400014

National Semiconductor

Local Area Network Databook

LOCAL AREA NETWORK
DATABOOK

1992 Edition

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

ABiC™
Abuseable™
Anadig™
ANS-R-TRAN™
APPS™
ASPECT™
Auto-Chem Deflasher™
BCP™
BI-FET™
BI-FET II™
BI-LINE™
BIPLAN™
BLC™
BLX™
BMAC™
Brite-Lite™
BSI™
CDD™
CheckTrack™
CIM™
CIMBUS™
CLASIC™
Clock✓Chek™
COMBO®
COMBO I™
COMBO II™
COPS™ microcontrollers
CRD™
DA4™
Datachecker®
DENSPAK™
DIB™
DISCERN™
DISTILL™
DNR®
DPVM™
E²CMOS™
ELSTAR™
Embedded System Processor™
E-Z-LINK™
FACT™
FACT Quiet Series™
FAIRCAD™
Fairtech™
FAST®
FASTr™
5-Star Service™
Flash™
GENIX™
GNX™
GTO™
HAMR™
HandiScan™
HEX 3000™
HPC™
I³L®
ICM™
INFOCHEX™
Integral ISE™
Intelisplay™
ISE™
ISE/06™
ISE/08™
ISE/16™
ISE32™
ISOPLANAR™
ISOPLANAR-Z™
KeyScan™
LERIC™
LMCMOS™
M²CMOS™
Macrobus™
Macrocomponent™
MAPL™
MAXI-ROM®
Meat✓Chek™
MenuMaster™
Microbus™ data bus
MICRO-DAC™
μtalker™
Microtalker™
MICROWIRE™
MICROWIRE/PLUS™
MOLE™
MPA™
MST™
Naked-8™
National®
National Semiconductor®
National Semiconductor Corp.®
NAX 800™
Nitride Plus™
Nitride Plus Oxide™
NML™
NOBUS™
NSC800™
NSCISE™
NSX-16™
NS-XC-16™
NTERCOM™
NURAM™
OPAL™
OXISS™
P²CMOS™
PC Master™
Perfect Watch™
Pharma✓Chek™
PLAN™
PLANAR™
PLAYER™
Plus-2™
Polycraft™
POSilink™
POSitalker™
Power + Control™
POWERplanar™
QUAD3000™
QUIKLOOK™
RAT™
RIC™
RTX16™
SABR™
Script✓Chek™
SCX™
SERIES/800™
Series 900™
Series 3000™
Series 32000®
Shelf✓Chek™
Simple Switcher™
SofChek™
SONIC™
SPIRE™
Staggered Refresh™
STAR™
Starlink™
STARPLEX™
ST-NIC™
Super-Block™
SuperChip™
SuperScript™
SYS32™
TapePak®
TDS™
TeleGate™
The National Anthem®
Time✓Chek™
TINA™
TLC™
Trapezoidal™
TRI-CODE™
TRI-POLY™
TRI-SAFE™
TRI-STATE®
TURBOTRANSCEIVER™
VIP™
VR32™
WATCHDOG™
XMOS™
XPU™
Z STAR™
883B/RETS™
883S/RETS™

abel™ is a trademark of Data I/O Corporation.
Apple®, Appletalk® and Macintosh® are registered trademarks of Apple Corporation.
COMPAQ® is a registered trademark of COMPAQ Corporation.
CP/M® is a registered trademark of Digital Research Corporation.
Dataphone® is a registered trademark of Dataphone Digital Service Corporation.
DECNET™ and VAX™ are trademarks of Digital Equipment Corporation.
General Motors® is a registered trademark of General Motors Corporation.
IBM®, MICROCHANNEL®, NETVIEW®, OS/2®, OS/2 Standard Edition®, PC-AT®, PS/2®, SNA® and System/2® are registered trademarks of International Business Machines Corporation.
Intel® is a registered trademark of Intel Corporation.
Microsoft®, MS® and MS-DOS® are registered trademarks of Microsoft Corporation.
NetWare™ is a trademark of Novell Inc.
PAL® is a registered trademark of and used under license from Advanced Micro Devices, Inc.
PC/TCP® is a registered trademark of FTP Software Inc.
Tandy® is a registered trademark of Tandy Corporation.
UNIX® is a registered trademark of AT&T Bell Laboratories Corporation.
3Com® is a registered trademark of 3Com Corporation.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090 1-800-272-9959
TWX (910) 339-9240

National Semiconductor

Product Status Definitions

Definition of Terms

Data Sheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This data sheet contains preliminary data, and supplementary data will be published at a later date. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

National Semiconductor Corporation reserves the right to make changes without further notice to any products herein to improve reliability, function or design. National does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

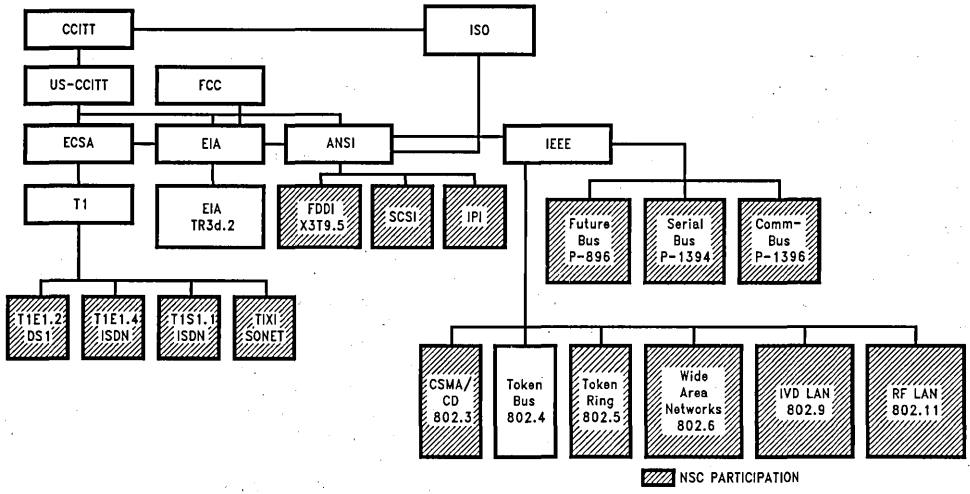
Table of Contents

Table of Contents (Continued)

Alpha-Numeric Index

National Semiconductor

Introduction to Local Area Network Standards and Products



TL/F/11257–1

PRO-ACTIVE COMMUNICATIONS STANDARDS INVOLVEMENT

Standards are crucial to the development and explosive growth of the communications industry. Several international committees and organizations develop and promote standards to ensure interoperability and interconnectivity. National Semiconductor Corporation takes an active part in the development and definition of these standards and produces a wide range of products that meet the requirements of these standards. Complete silicon solutions for IEEE 802.3 CSMA/CD Ethernet and FDDI are a part of National's communications portfolio, one of the broadest in the semiconductor industry.

FDDI (Fiber Distributed Data Interface)

The Fiber Distributed Data Interface is a 100 Mbit/sec. Local Area Network that uses a token passing access method and uses optical fiber as the medium with twisted pair wiring medium specifications to come in the future. Like Ethernet, FDDI has maximized the value of standardization and conformed to the Open Systems Interconnect (OSI) model and is specified by the ANSI X3T9.5 committee. The rich functionality that has been integrated into FDDI to meet the needs of a number of market segments means that it can be the one standard satisfying the requirements of broad, high-speed LANs. FDDI offers a low-cost means for bridging Ethernet to a high-speed fiber optic link, connecting local islands of Ethernet workgroups to similar islands located in another part of a company or office.

For FDDI to grow into a successful LAN, network equipment manufacturers need a practical and cost-effective standard chip set. National, already the leading supplier of IEEE 802.3 network silicon solutions, has developed a high-performance, cost-effective solution, the DP83200 FDDI chip set.

ETHERNET: IEEE 802.3

The IEEE 802.3 CSMA/CD Ethernet Protocol has become the most widely accepted standard for Local Area Networks connecting personal computers and workstations with information resources, servers and other peripherals.

The CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol defines how a node will gain access to transmit over the cable. The node first monitors the cable to ensure no transmissions are in progress (Carrier Sense). Any node may then decide to transmit (Multiple Access). If more than one node decides to transmit then all nodes must be able to detect this condition (Collision Detection), stop their transmissions and retry.

PHYSICAL LAYER SPECIFICATIONS

The four IEEE 802.3 Physical Layer Specifications, summarized in Table I, differ primarily in the cable schemes, including media and topology. 10Base5 (Thick Coax Ethernet) uses a thick double-shielded coaxial cable at 10 Mbit/sec. data rate in a bus configuration. 10Base2 (Thin Coax Ethernet) maintains the same 10 Mbit/sec. data rate and bus topology, but uses single-shielded coaxial cable. 1Base5 (StarLAN) uses inexpensive twisted pair cabling in a star topology at only 1 Mbit/sec. data rate. The newest IEEE 802.3 Physical Layer Specification, 10Base-T for Twisted Pair Ethernet, combines the installation convenience of inexpensive twisted pair cable and star topology of 1Base5 at the 10 Mbit/sec. data rate for performance compatibility with 10Base5 and 10Base2 Ethernet networks.

TABLE I. IEEE 802.3 Physical Layer Specifications

Parameter	10Base5	10Base2	1Base5	10Base-T
Designator	Thick Coax	Thin Coax	StarLAN	Twisted Pair
Data Rate	10 Mbit/sec.	10 Mbit/sec.	1 Mbit/sec.	10 Mbit/sec.
Segment Length	500 Meters	185 Meters	500 Meters	100 Meters Nominal
Topology	Bus (Multi-Point)	Bus (Multi-Point)	Star (Point-to-Point)	Star (Point-to-Point)
Cable Type	0.4" Diam. 50Ω, Double Shield Coax (RG11)	0.2" Diam. 50Ω, Single Shield Coax (RG58)	24 Gauge, 100Ω Twisted Pair	24 Gauge, 100Ω Twisted Pair
Connection	Precision TAP	BNC "T"	8-Pin, RJ-45	8-Pin, RJ-45

COMPREHENSIVE ETHERNET SOLUTIONS

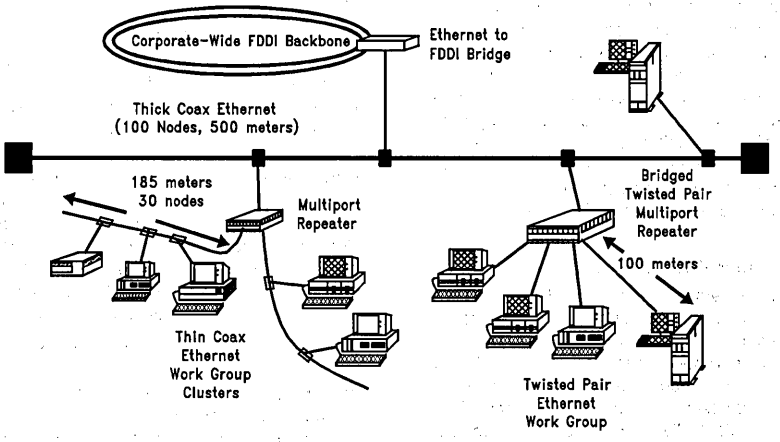
The demands for high performance, smaller chip count, and increased functionality have become significant factors in the selection of Ethernet silicon solutions. National offers a wide variety of Ethernet industry-standard silicon and development evaluation tools to meet the diverse needs of network product designers and manufacturers.

Integrated Network Interface Controller and ENDEC products for 8-bit and 16-bit designs include the single-chip DP83902 Serial Network Interface Controller for Twisted Pair (ST-NIC™), the DP83901 Serial Network Interface Controller (SNIC), the DP8390 Network Interface Controller (NIC), and the DP83910 CMOS Serial Network Interface (CSNI). 16-bit and 32-bit network interface controller and interconnectivity requirements are addressed by the DP83932 Systems Oriented Network Interface Controller (SONIC™). SONIC provides exceptional 16-bit or 32-bit Host-to-Ethernet interface management and performance with all of the major state-of-the-art CISC and RISC microprocessor architectures.

IEEE 802.3 Physical Layer Tansceiver products include the industry-standard DP8392 Coaxial Transceiver Interface (CTI) for 10Base5 and 10Base2; and the DP83922 Twisted Pair Transceiver Interface (TPI) for 10Base-T.

Repeater Interface Controller products are designed to meet the needs of intelligent Multi-Port Repeater or Hub designs. The DP83950 Repeater Interface Controller (RIC™) implements the IEEE 802.3 Repeater Specification, the 10Base-T Specification, and the Hub Management Draft Specification. Designs incorporating both RIC and SONIC provide the necessary hardware platform for Managed Hubs which support Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP) and other network management protocols. Simple, unmanaged Multi-Port Repeater or Hub design requirements are met by the Low End Repeater Interface Controller (LERIC™).

National offers an extensive selection of Evaluation Platforms, Demonstration Software, Application Notes, and System Briefs to assist network product designers and manufacturers both in silicon selection and product development.



TL/F/11257–2

FIGURE 1. Ethernet IEEE 802.3 Network

Section 1

Integrated Ethernet Network Interface Controller Products

Section 1 Contents

National Semiconductor

PRELIMINARY

DP83902 Serial Network Interface Controller for Twisted Pair (ST-NIC™)

General Description

The DP83902 Serial Network Interface Controller for Twisted Pair (ST-NIC) is a microCMOS VLSI device designed for easy implementation of CSMA/CD local area networks. These include Ethernet (10BASE5), Thin Ethernet (10BASE2) and Twisted-pair Ethernet (10BASE-T). The overall ST-NIC solution provides the Media Access Control (MAC) and Encode-Decode (ENDEC) with an AUI interface, and 10BASE-T transceiver functions in accordance with the IEEE 802.3 standards.

The DP83902's 10BASE-T transceiver fully complies with the IEEE standard. This functional block incorporates the receiver, transmitter, collision, heartbeat, loopback, jabber, and link integrity blocks as defined in the standard. The transceiver when combined with equalization resistors, transmit/receive filters, and pulse transformers provides a complete physical interface from the DP83902's ENDEC module and the twisted pair medium.

The integrated ENDEC module allows Manchester encoding and decoding via a differential transceiver and phase lock loop decoder at 10 Mbit/sec. Also included are collision detect translator and diagnostic loopback capability. The ENDEC module interfaces directly to the transceiver module, and also provides a fully IEEE compliant AUI (Attachment Unit Interface) for connection to other media transceivers. (Continued)

Features

- Single chip solution for IEEE 802.3, 10BASE-T
- Integrated controller, ENDEC, and transceiver
- Full AUI interface
- No external precision components required
- 3 levels of loopback supported

Transceiver Module

- Integrates transceiver electronics, including:
  - — Transmitter and receiver
  - — Collision detect, heartbeat and jabber timer
  - — Link integrity test
- Link disable for use with pre-standard twisted pair implementations
- Integrated smart receive squelch
- Polarity detection/correction

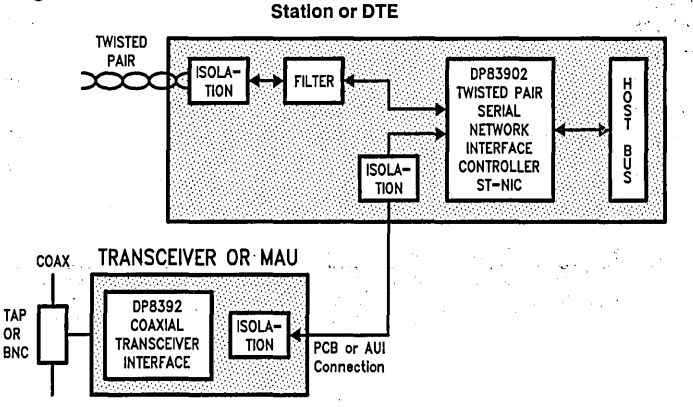
ENDEC Module

- 10 Mb/s Manchester encoding/decoding, plus clock recovery
- Transmitter half or full step mode
- Squelch on receive and collision pairs
- Lock time 5 bits typical
- Decodes Manchester data with up to ±18 ns jitter

MAC/Controller Module

- 100% DP8390 software/hardware compatible
- Dual 16-bit DMA channels
- 16-byte internal FIFO
- Efficient buffer management implementation
- Independent system and network clocks
- Supports physical, multicast and broadcast address filtering
- Network statistics storage

1.0 System Diagram



TL/F/11157–1

General Description (Continued)

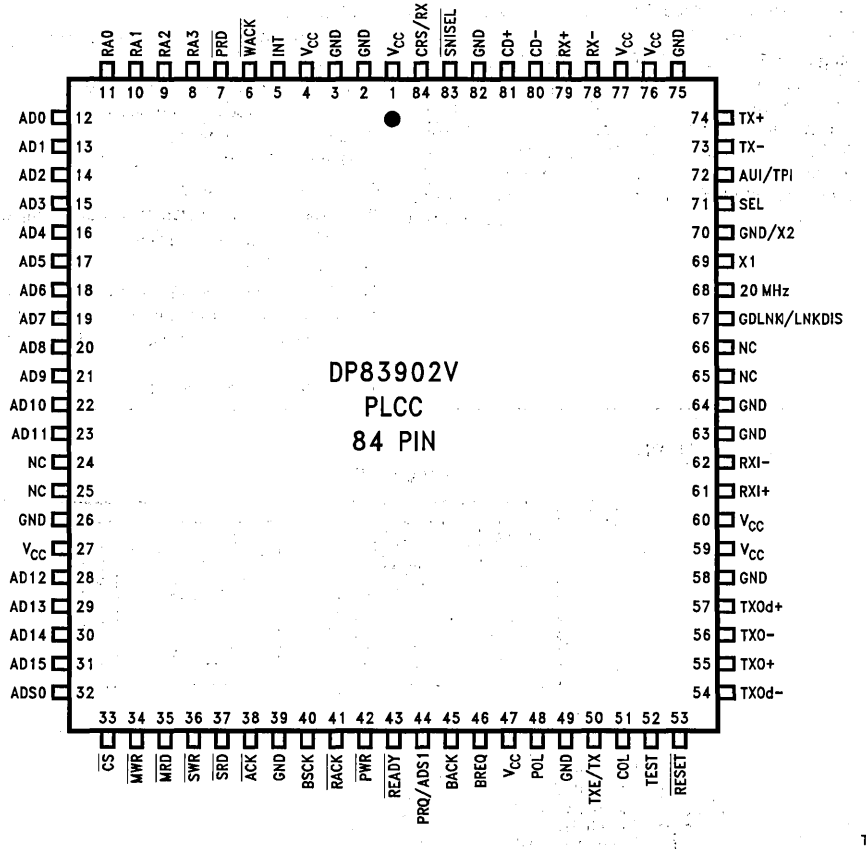
The Media Access Control function which is provided by the Network Interface Control module (NIC) provides simple and efficient packet transmission and reception control by means of unique dual DMA channels and an internal FIFO. Bus arbitration and memory control logic are integrated to reduce board cost and area overheads.

DP83902 provides a comprehensive single chip solution for 10BASE-T IEEE 802.3 networks and is designed for easy interface to other transceivers via the AUI interface.

Due to the inherent constraints of CMOS processing, isolation is required at the AUI differential signal interface for 10BASE5 and 10BASE2 applications. Capacitive or inductive isolation may be used.

Table Of Contents

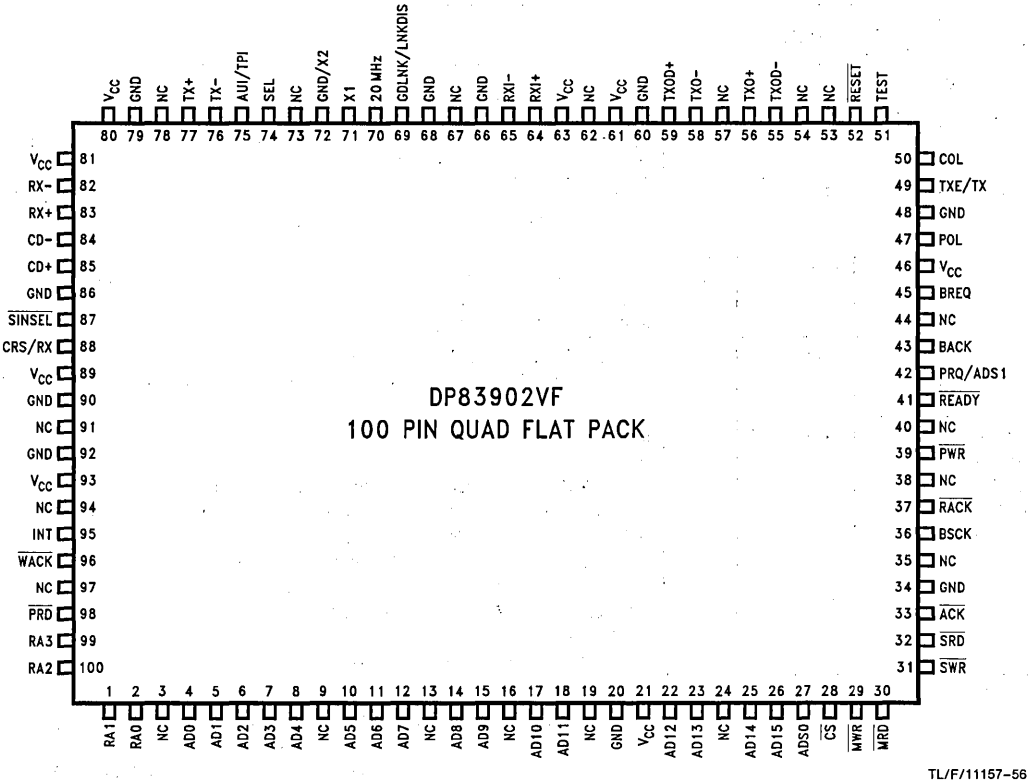
Connection Diagrams



TL/F/11157–2

Order Number DP83902V
See NS Package Number V84A

Connection Diagrams (Continued)



TL/F/11157–56

Order Number DP83902VF
See NS Package Number VF100B

2.0 Pin Description

PQFP Pin No.	PLCC Pin No.	Pin Name	I/O	Description
BUS INTERFACE PINS				
95	5	INT	O	**INTERRUPT:** Indicates that the DP83902 requires CPU attention after reception transmission or completion of DMA transfers. The interrupt is cleared by writing to the ISR (Interrupt Status Register). All interrupts are maskable.
96	6	$\overline{WACK}$	I	**WRITE ACKNOWLEDGE:** Issued from system to DP83902 to indicate that data has been written to the external latch. The DP83902 will begin a write cycle to place the data in local memory.
98	7	$\overline{PRD}$	O	**PORT READ:** Enables data from external latch on to local bus during a memory write cycle to local memory (remote write operation). This allows asynchronous transfer of data from the system memory to local memory.
99, 100, 1, 2	8–11	RA3–RA0	I	**REGISTER ADDRESS:** These four pins are used to select a register to be read or written. The state of these inputs is ignored when the DP83902 is not in slave mode ($\overline{CS}$ high).
4–8, 10–12, 14, 15, 17, 18, 22, 23, 25, 26	12–23, 28–31	AD0–AD15	I/O, Z	**MULTIPLEXED ADDRESS/DATA BUS:** • Register Access, with DMA inactive, $\overline{CS}$ low and $\overline{ACK}$ returned from DP83902, pins AD0–AD7 are used to read and write register data. AD8–AD15 float during I/O transfers, $\overline{SRD}$, $\overline{SWR}$ pins are used to select direction of transfer. • Bus Master with BACK input asserted. During t1 of memory cycle AD0–AD15 contain address. During t2, t3, t4 AD0–AD15 contain data (word transfer mode). During t2, t3, t4 AD0–AD7 contain data, AD8–AD15 contain address (byte transfer mode). Direction of transfer is indicated by DP83902 on $\overline{MWR}$, $\overline{MRD}$ lines.
27	32	ADS0	I/O, Z	**ADDRESS STROBE 0:** • Input: with DMA inactive and $\overline{CS}$ low, latches RA0–RA3 inputs on falling edge. If high, data present on RA0–RA3 will flow through latch. • Output: When Bus Master, latches address bits (AD0–AD15) to external memory during DMA transfers.
28	33	$\overline{CS}$	I	**CHIP SELECT:** Chip Select places controller in slave mode for μP access to internal registers. Must be valid through data portion of bus cycle. RA0–RA3 are used to select the internal register. $\overline{SWR}$ and $\overline{SRD}$ select direction of data transfer.
29	34	$\overline{MWR}$	O, Z	**MASTER WRITE STROBE:** (Strobe for DMA transfers) Active low during write cycles (t2, t3, tw) to buffer memory. Rising edge coincides with the presence of valid output data. TRI-STATE® until BACK asserted.
30	35	$\overline{MRD}$	O, Z	**MASTER READ STROBE:** (Strobe for DMA transfers) Active during read cycles (t2, t3, tw) to buffer memory. Input data must be valid on rising edge of MRD. TRI-STATE until BACK asserted.
31	36	$\overline{SWR}$	I	**SLAVE WRITE STROBE:** Strobe from CPU to write an internal register selected by RA0–RA3. Data is latched into the DP83902 on the rising edge of this input.
32	37	$\overline{SRD}$	I	**SLAVE READ STROBE:** Strobe from CPU to read an internal register selected by RA0–RA3. The register data is output when $\overline{SRD}$ goes low.
33	38	$\overline{ACK}$	O	**ACKNOWLEDGE:** Active low when DP83902 grants access to CPU. Used to insert WAIT states to CPU until DP83902 is synchronized for a register read or write operation.
36	40	BSCK	I	**BUS CLOCK:** This clock is used to establish the period of the DMA memory cycle. Four clock cycles (t1, t2, t3, t4) are used per DMA cycle. DMA transfers can be extended by one BSCK increment using the $\overline{READY}$ input.
37	41	$\overline{RACK}$	I	**READ ACKNOWLEDGE:** Indicates that the system DMA or host CPU has read the data placed in the external latch by the DP83902. The DP83902 will begin a read cycle to update the latch.
39	42	$\overline{PWR}$	O	**PORT WRITE:** Strobe used to latch data from the DP83902 into external latch for transfer to host memory during Remote Read transfers. The rising edge of $\overline{PWR}$ coincides with the presence of valid data on the local bus.

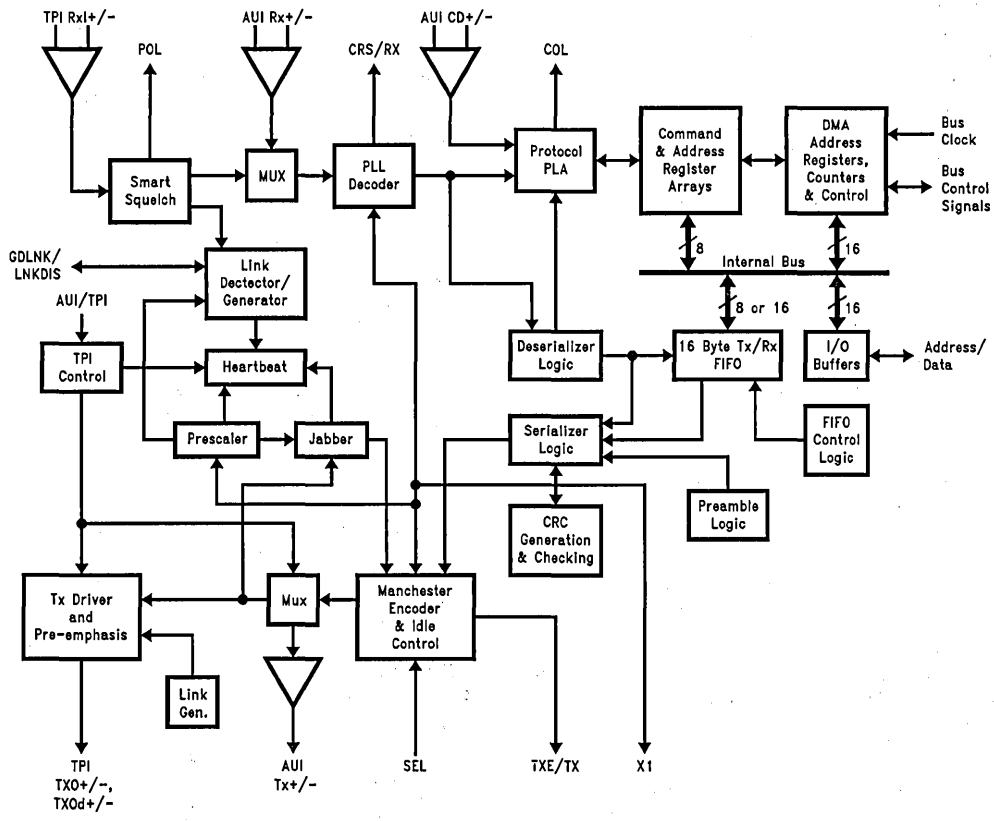
2.0 Pin Description (Continued)

PQFP Pin No.	PLCC Pin No.	Pin Name	I/O	Description
BUS INTERFACE PINS (Continued)				
41	43	$\overline{\text{READY}}$	I	**READY:** This pin is set high to insert wait states during a DMA transfer. The DP83902 will sample this signal at t3 during DMA transfers.
42	44	PRQ/ADS1	O, Z	**PORT REQUEST/ADDRESS STROBE 1** • 32-BIT MODE: If LAS is set in the Data Configuration Register, this line is programmed as ADS1. It is used to strobe addresses A16–A31 into external latches. (A16–A31 are the fixed addresses stored in RSAR0, RSAR1). ADS1 will remain at TRI-STATE until BACK is received. • 16-BIT MODE: If LAS is not set in the Data Configuration Register, this line is programmed as PRQ and is used for Remote DMA Transfers. The DP83902 initiates a single remote DMA read or write operation by asserting this pin. In this mode PRQ will be a standard logic output. **Note:** This line will power up as TRI-STATE until the Data Configuration Register is programmed.
43	45	BACK	I	**BUS ACKNOWLEDGE:** Bus Acknowledge is an active high signal indicating that the CPU has granted the bus to the DP83902. If immediate bus access is desired, BREQ should be tied to BACK. **Tying BACK to V_{CC} will result in a deadlock.**
45	46	BREQ	O	**BUS REQUEST:** Bus Request is an active high signal used to request the bus for DMA transfers. This signal is automatically generated when the FIFO needs servicing.
52	53	$\overline{\text{RESET}}$	I	**RESET:** Reset is active low and places the DP83902 in a reset mode immediately. No packets are transmitted or received by the DP83902 until STA bit is set. Affects Command Register, Interrupt Mask Register, Data Configuration Register and Transmit Configuration Register. The DP83902 will execute reset within 10 BSCK cycles.
NETWORK INTERFACE PINS				
47	48	POL	O	**POLARITY:** A TTL/MOS active high output. This signal is normally in the low state. When the TPI module detects seven consecutive link pulses or three consecutive received packets with reversed polarity POL, is asserted.
49	50	TXE/TX	O	**TRANSMIT ENABLE/TRANSMIT:** A TTL/MOS active high output. It is asserted for approximately 50 ms whenever the DP83902 transmits data in either AUI or TPI modes.
50	51	COL	O	**COLLISION:** A TTL/MOS active high output. It is asserted for approximately 50 ms whenever the DP83902 detects a collision in either the AUI or TPI modes.
51	52	TEST	I	**FACTORY TEST INPUT:** Used to check the chip's internal functions. This should be tied low during normal operation.
55, 56, 58, 59	54, 55, 56, 57	TXOd−, TXO+, TXO−, TXOd+	O	**TWISTED PAIR TRANSMIT OUTPUTS:** These high drive CMOS level outputs are resistively combined external to the chip to produce a differential output signal with equalization to compensate for Intersymbol Interference (ISI) on the twisted pair medium.
64, 65	61, 62	RXI+, RXI−	I	**TWISTED PAIR RECEIVE INPUTS:** These inputs feed a differential amplifier which passes valid data to the ENDEC module.
69	67	GDLNK/LNKDIS	I/O	**GOOD LINK/LINK DISABLE:** This pin has a dual function both input and output. The function is latched by the DP83902 on the rising edge of the Reset signal i.e.: on the chip returning to normal operation after reset. As an output this pin is configured as an open drain N-channel device and is suitable for driving a LED. It will be latched as output on removal of chip reset if connected to a LED or left open circuit. Under normal conditions (the twisted pair link is not broken) the output will be low, and the LED will be lit. The open drain output will be switched off if the twisted pair link has been detected to be broken. It is recommended that the color of the LED be green. This output will be pulled high in AUI mode, by an internal resistor of approximately 15 kΩ. When this pin, which has an internal pull-up resistor to V_{DD}, is tied low it becomes an input and the link integrity checking is disabled.

2.0 Pin Description (Continued)

PQFP Pin No.	PLCC Pin No.	Pin Name	I/O	Description
NETWORK INTERFACE PINS (Continued)				
70	68	20 MHz	O	**20 MHz:** This is a TTL/MOS level signal. It is a buffered version of the oscillator X2. It is suitable to drive external logic.
71	69	X1	I	**EXTERNAL OSCILLATOR INPUT**
72	70	GND/X2	I	**GROUND/X2:** This pin should normally be connected to ground. It is possible to use a crystal oscillator using X1 and GND/X2 if certain precautions are taken. Contact National Semiconductor for more information.
74	71	SEL	I	**MODE SELECT:** When high, TX+ and TX− are the same voltage in the idle state. When low, Transmit+ is positive with respect to Transmit− in the idle state, at the transformer's primary.
75	72	AUI/TPI	I	**AUI/TPI SELECT:** A TTL level active high input that selects either the AUI interface or the TPI module for interface with the ENDEC module. When high the AUI is selected, when low the TPI is selected.
76, 77	73, 74	TX−, TX+	O	**AUI TRANSMIT OUTPUT:** Differential driver which sends the encoded data to the transceiver. The outputs are source followers which require 270Ω pulldown resistors.
82, 83	78, 79	RX−, RX+	I	**AUI RECEIVE INPUT:** Differential receive input pair from the transceiver.
84, 85	80, 81	CD−, CD+	I	**AUI COLLISION INPUT:** Differential collision pair input from the transceiver.
87	83	$\overline{\text{SNISEL}}$	I	**FACTORY TEST INPUT:** For normal operation tied to V_{CC}. When low enables the ENDEC module to be tested independently of the DP83902 module.
88	84	CRS/RX	O	**CARRIER SENSE/RECEIVE:** A TTL/MOS level active high signal. It is asserted for approximately 50 ms whenever valid transmit or receive data is detected while in AUI mode or receive data is detected while in TPI mode.
POWER SUPPLY PINS (DIGITAL)				
21, 46, 89	1, 27, 47	V_{CC}		**POSITIVE 5V SUPPLY PINS**
20, 90	2, 26, 39, 49, 64	GND		**NEGATIVE (GROUND) SUPPLY PINS:** It is suggested that a decoupling capacitor be connected between the V_{CC} and GND pins.
POWER SUPPLY PINS (ANALOG)				
93	4	V_{CC}		**VCO 5V SUPPLY PIN:** Care should be taken to reduce noise on this pin as it supplies power to the analog VCO to the Phase Lock Loop.
92	3	GND		**VCO GROUND SUPPLY PIN:** Care should be taken to reduce noise on this pin as it supplies ground to the analog VCO to the Phase Lock Loop.
63	60	V_{CC}		**TPI RECEIVE 5V SUPPLY:** Power pin supplies 5V to the Twisted Pair Interface Receiver.
66	63	GND		**TPI RECEIVE GROUND:** Ground pin for the Twisted Pair Interface Receiver.
61	59	V_{CC}		**TPI TRANSMIT 5V SUPPLY:** Power pin supplies 5V to the Twisted Pair Interface Transmitter.
60	58	GND		**TPI TRANSMIT GROUND:** Ground pin for the Twisted Pair Interface Transmitter.
81	77	V_{CC}		**AUI RECEIVE 5V SUPPLY:** Power pin supplies 5V to the AUI Interface Receiver.
86	82	GND		**AUI RECEIVE GROUND:** Ground pin for the AUI Interface Receiver.
80	76	V_{CC}		**AUI TRANSMIT 5V SUPPLY:** Power pin supplies 5V to AUI Interface Transmitter.
79	75	GND		**AUI TRANSMIT GROUND:** Ground pin for the AUI Interface Transmitter.
NO CONNECTION				
3, 9, 13, 16, 19, 24, 35, 38, 40, 44, 53, 54, 57, 62, 67, 73, 78, 91, 94, 97	24, 25, 65, 66	NC		NO CONNECTION. Do not connect to these pins.

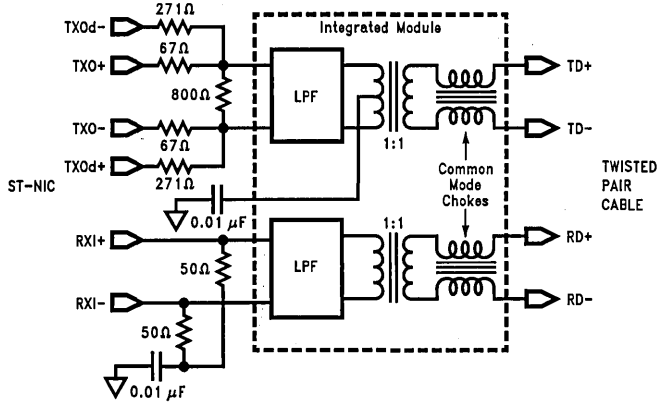
3.0 Block Diagram



TL/F/11157-3

FIGURE 1

Typical Connection to Twisted Pair Cable



TL/F/11157-4

Recommended integrated modules are:
1) Pulse Engineering PE65431
2) Belfuse 0556-2006-01 or 0556-3392-00
3) Valor FL1012.

ST-NIC Twisted Pair Interface

4.0 Functional Description (Refer to *Figure 1*)

TWISTED PAIR INTERFACE (TPI) MODULE

The TPI consists of five main logical functions:

a) The Smart Squelch, responsible for determining when valid data is present on the differential receive inputs (RXI±).

b) The Collision function checks for simultaneous transmission and reception of data on the TXO± and RXI± pins.

c) The Link Detector/Generator checks the integrity of the cable connecting the two twisted pair MAUs.

d) The Jabber disables the transmitter if it attempts to transmit a longer than legal packet.

e) The Tx Driver & Pre-emphasis transmits Manchester encoded data to the twisted pair network via the summing resistors and transformer/filter.

SMART SQUELCH

The ST-NIC implements an intelligent receive squelch on the RXI± differential inputs to ensure that impulse noise on the receive inputs will not be mistaken for a valid signal.

The squelch circuitry employs a combination of amplitude and timing measurements to determine the validity of data on the twisted pair inputs.

The diagram shows the operation of the smart squelch.

The signal at the start of packet is checked by the smart squelch and any pulses not exceeding the squelch level (either positive or negative, depending upon polarity) will be rejected. Once this first squelch level is overcome correctly the opposite squelch level must then be exceeded within 150 ns. Finally the signal must exceed the original squelch level within a further 150 ns to ensure that the input waveform will not be rejected. The checking procedure results in the loss of typically three bits at the beginning of each packet.

Only after all these conditions have been satisfied will a control signal be generated to indicate to the remainder of the circuitry that vaild data is present. At this time the smart squelch circuitry is reset.

Valid data is considered to be present until either squelch level has not been generated for a time longer than 150 ns, indicating End of Packet. Once good data has been detected the squelch levels are reduced to minimize the effect of noise causing premature End of Packet detection.

COLLISION

A collision is detected by the TPI module when the receive and transmit channels are active simultaneously. If the TPI is receiving when a collision is detected it is reported to the controller immediately. If, however, the TPI is transmitting when a collision is detected the collision is not reported until seven bits have been received while in the collision state. This prevents a collision being reported incorectly due to noise on the network. The signal to the controller remains for the duration of the collision.

Approximately 1 μs after the transmission of each packet a signal called the Signal Quality Error (SQE) consisting of typically 10 cycles of 10 MHz is generated. This 10 MHz signal, also called the Heartbeat, ensures the continued functioning of the collision circuitry.

LINK DETECTOR/GENERATOR

The link generator is a timer circuit that generates a link pulse as defined by the 10BASE-T specification that will be generated by the transmitter section. The pulse which is 100 ns wide is transmitted on the TXO+ output, every 16 ms, in the absence of transmit data.

The pulse is used to check the integrity of the connection to the remote MAU. The link detection circuit checks for valid pulses from the remote MAU and if valid link pulses are not received the link detector will disable the transmit, receive and collision detection functions.

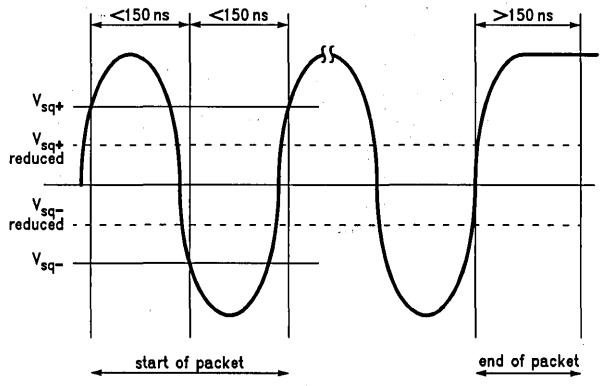
The GDLNK output can directly drive a LED to show that there is a good twisted pair link. For normal conditions the LED will be on. The link integrity function can be disabled as described in the Pin Description Section.

JABBER

The jabber timer monitors the transmitter and disables the transmission if the transmitter is active for greater than 26 ms. The transmitter is then disabled for the whole time that the ENDEC module's internal transmit enable is asserted. This signal has to be deasserted for approximately 750 ms (the unjab time) before the Jabber re-enables the transmit outputs.

TRANSMIT DRIVER

The transmitter consists of four signals, the true and complement Manchester encoded data (TXO±) and these signals delayed by 50 ns (TXOd±).

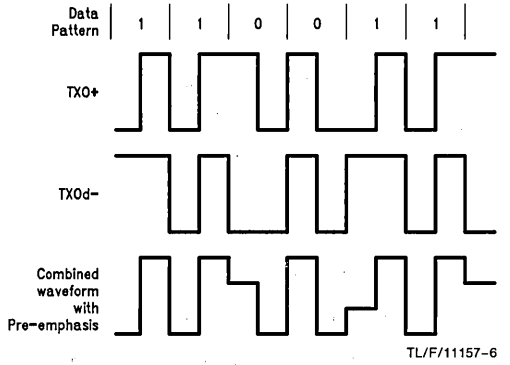


TL/F/11157–5

4.0 Functional Description (Continued)

These four signals are resistively combined, TXO+ with TXOd− and TXO− with TXOd+. This is known as digital pre-emphasis and is required to compensate for the twisted pair cable which acts like a low pass filter causing greater attenuation to the 10 MHz (50 ns) pulses of the Manchester encoded waveform than the 5 MHz (100 ns) pulses.

An example of how these signals are combined is shown in the following diagram.



TL/F/11157–6

The signal with pre-emphasis shown above is generated by resistively combining TXO+ and TXOd−. This signal along with its complement is passed to the transmit filter.

STATUS INFORMATION

Status information is provided by the ST-NIC on the CRS/RX, TXE/TX, COL and POL outputs as described in the pin description table. These outputs are suitable for driving status LEDs via an appropriate driver circuit.

The POL output is normally low, and will be driven high when seven consecutive link pulses or three consecutive receive packets are detected with reversed polarity. A polarity reversal can be caused by a wiring error at either end of the TPI cable. On detection of a polarity reversal the condition is latched and POL is asserted. The TPI corrects for this error internally and will decode received data correctly, eliminating the need to correct the wiring error.

ENCODER/DECODER (ENDEC) MODULE

The ENDEC consists of four main logical blocks:

a) The Manchester encoder accepts NRZ data from the controller, encodes the data to Manchester, and transmits it differentially to the transceiver, through the differential transmit driver.

b) The Manchester decoder receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends it to the controller.

c) The collision translator indicates to the controller the presence of a valid 10 MHz collision signal to the PLL.

MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The differential transmit pair, on the secondary of the transformer, drives up to 50 meters of twisted pair AUI cable. These outputs are source followers which require two 270Ω pull-down resistors to ground.

The DP83902 allows both half-step and full-step to be compatible with Ethernet and IEEE 802.3. With the SEL pin low (for Ethernet I). Transmit+ is positive with respect to Transmit− during idle; with SEL high (for IEEE 802.3), Transmit+ and Transmit− are equal in the idle state. This provides zero differential voltage to operate with transformer coupled loads.

MANCHESTER DECODER

The decoder consists of a differential receiver and a PLL to separate a Manchester decoded data stream into internal clock signals and data. The differential input must be externally terminated with two 39Ω resistors connected in series if the standard 78Ω transceiver drop cable is used. In thin Ethernet applications, these resistors are optional. To prevent noise from falsely triggering the decoder, a squelch circuit at the input rejects signals with levels less than −175 mV. Signals more negative than −300 mV are decoded. Data becomes valid typically within 5 bit times. The DP83902 may tolerate bit jitter up to 18 ns in the received data. The decoder detects the end of a frame when no more mid-bit transitions are detected.

COLLISION TRANSLATOR

When in AUI mode, when the Ethernet transceiver (DP8392 CTI) detects a collision, it generates a 10 MHz signal to the differential collision inputs (CD±) of the DP83902. When these inputs are detected active, the DP83902 uses this signal to back off its current transmission and reschedule another one.

The collision differential inputs are terminated the same way as the differential receive inputs. The squelch circuitry is also similar, rejecting pulses with levels less than −175 mV.

NIC (Media Access Control) MODULE

RECEIVE DESERIALIZER

The Receive Deserializer is activated when the input signal Carrier Sense is asserted to allow incoming bits to be shifted into the shift register by the receive clock. The serial receive data is also routed to the CRC generator/checker. The Receive Deserializer includes a synch detector which detects the SFD (Start of Frame Delimiter) to establish where byte boundaries within the serial bit stream are located. After every eight receive clocks, the byte wide data is transferred to the 16-byte FIFO and the Receive Byte Count is incremented. The first six bytes after the SFD are checked for valid comparison by the Address Recognition Logic. If the Address Recognition Logic does not recognize the packet, the FIFO is cleared.

CRC GENERATOR/CHECKER

During transmission, the CRC logic generates a local CRC field for the transmitted bit sequence. The CRC encodes all fields after the SFD. The CRC is shifted out MSB first following the last transmit byte. During reception the CRC logic generates a CRC field from the incoming packet. This local CRC is serially compared to the incoming CRC appended to the end of the packet by the transmitting node. If the local and received CRC match, a specific pattern will be generated and decoded to indicate no data errors. Transmission errors result in different pattern and are detected, resulting in rejection of a packet (if so programmed).

TRANSMIT SERIALIZER

The Transmit Serializer reads parallel data from the FIFO and serializes it for transmission. The serializer is clocked by

4.0 Functional Description (Continued)

the transmit clock generated internally. The serial data is also shifted into the CRC generator/checker. At the beginning of each transmission, the Preamble and Synch Generator append 62 bits of 1,0 preamble and a 1,1 synch pattern. After the last data byte of the packet has been serialized the 32-bit FCS field is shifted directly out of the CRC generator. In the event of a collision the Preamble and Synch generator is used to generate a 32-bit JAM pattern of all 1's.

ADDRESS RECOGNITION LOGIC

The address recognition logic compares the Destination Address Field (first 6 bytes of the received packet) to the Physical address registers stored in the Address Register Array. If any one of the six bytes does not match the pre-programmed physical address, the Protocol Control Logic rejects the packet. All multicast destination addresses are filtered using a hashing technique. (See register description.) If the multicast address indexes a bit that has been set in the filter bit array of the Multicast Address Register Array the packet is accepted, otherwise it is rejected by the Protocol Control Logic. Each destination address is also checked for all 1's which is the reserved broadcast address.

FIFO AND BUS OPERATIONS

Overview

To accommodate the different rates at which data comes from (or goes to) the network and goes to (or comes from) the system memory, the ST-NIC contains a 16-byte FIFO for buffering data between the media. The FIFO threshold is programmable. When the FIFO has filled to its programmed threshold, the local DMA channel transfers these bytes (or words) into local memory. It is crucial that the local DMA is given access to the bus within a minimum bus latency time; otherwise a FIFO underrun (or overrun) occurs.

FIFO underruns or overruns are caused by two conditions: (1) the bus latency is so long that the FIFO has filled (or emptied) from the network before the local DMA has serviced the FIFO and (2) the bus latency has slowed the throughput of the local DMA to a point where it is slower than the network data rate (10 Mbit/sec). This second condition is also dependent upon DMA clock and word width (byte wide or word wide). The worst case condition ultimately limits the overall bus latency which the ST-NIC can tolerate.

Beginning of Receive

At the beginning of reception, the ST-NIC stores the entire Address field of each incoming packet in the FIFO to determine whether the address matches the ST-NIC's Physical Address Registers or maps to one of its Multicast Registers. This causes the FIFO to accumulate 8 bytes. Furthermore, there are some synchronization delays in the DMA PLA. Thus, the actual time to when BREQ is asserted from the time the Start of Frame Delimiter (SFD) is detected is 7.8 μs. This operation affects the bus latencies at 2- and 4-byte thresholds during the first receive BREQ since the FIFO must be filled to 8 bytes (or 4 words) before issuing a BREQ.

End of Receive

When the end of a packet is detected by the ENDEC module, the ST-NIC enters its end of packet processing sequence, emptying its FIFO and writing the status information at the beginning of the packet. The ST-NIC holds onto the bus for the entire sequence. The longest time BREQ may be extended occurs when a packet ends just as the ST-NIC performs its last FIFO burst. The ST-NIC, in this case, performs a programmed burst transfer followed by flushing the remaining bytes in the FIFO, and completes by writing the header information to memory. The following steps occur during this sequence.

1. ST-NIC issues BREQ because the FIFO threshold has been reached.
2. During the burst, packet ends, resulting in BREQ extended.
3. ST-NIC flushes remaining bytes from FIFO.
4. ST-NIC performs internal processing to prepare for writing the header.
5. ST-NIC writes 4-byte (2-word) header.
6. ST-NIC de-asserts BREQ.

FIFO Threshold Detection

To assure that no overwriting of data in the FIFO, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO, effectively shortening the FIFO to 13 bytes. The FIFO logic also operates differently in Byte Mode and in Word Mode. In Byte Mode, a threshold is indicated when the n+1 byte has entered the FIFO; thus, with an 8-byte threshold, the ST-NIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until n+2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to an 8-byte threshold), BREQ is issued when the 10th byte has entered the FIFO.

Beginning of Transmit

Before transmitting, the ST-NIC performs a prefetch from memory to load the FIFO. The number of bytes prefetched is the programmed FIFO threshold. The next BREQ is not issued until after the ST-NIC actually begins transmitting data, i.e., after SFD.

Reading the FIFO

During normal operation, the FIFO must not be read. The ST-NIC will not issue an ACKnowledge back to the CPU if the FIFO is read. The FIFO should only be read during loopback diagnostics.

PROTOCOL PLA

The protocol PLA is responsible for implementing the IEEE 802.3 protocol, including collision recovery with random backoff. The Protocol PLA also formats packets during transmission and strips preamble and synch during reception.

DMA AND BUFFER CONTROL LOGIC

The DMA and Buffer Control Logic is used to control two 16-bit DMA channels. During reception, the local DMA stores packets in a receive buffer ring, located in buffer memory. During transmission the Local DMA uses programmed pointer and length registers to transfer a packet from local buffer memory to the FIFO. A second DMA channel is used as a slave DMA to transfer data between the local buffer memory and the host system. The Local DMA and Remote DMA are internally arbitrated, with the Local DMA channel having highest priority. Both DMA channels use a common external bus clock to generate all required bus timing. External arbitration is performed with a standard bus request, bus acknowledge handshake protocol.

5.0 Transmit/Receive Packet Encapsulation/Decapsulation

A standard IEEE 802.3 packet consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data, and Frame Check Sequence (FCS). The typical format is shown in *Figure 2*. The packets are Manchester encoded and decoded by the ENDEC module and transferred serially to the NIC module using NRZ data with a clock. All fields are of fixed length except for the data field. The ST-NIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the ENDEC to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC module. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern which consists of two consecutive 1's. The ST-NIC does not treat the SFD pattern as a byte, it detects only the two bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the ST-NIC: physical, multicast and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of "0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the ST-NIC to accept the packet. Multicast addresses begin with an MSB of "1". The ST-NIC filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a 6-bit value. This 6-bit value indexes a 64-bit array that filters the value. If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

SOURCE ADDRESS

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

LENGTH/TYPE FIELD

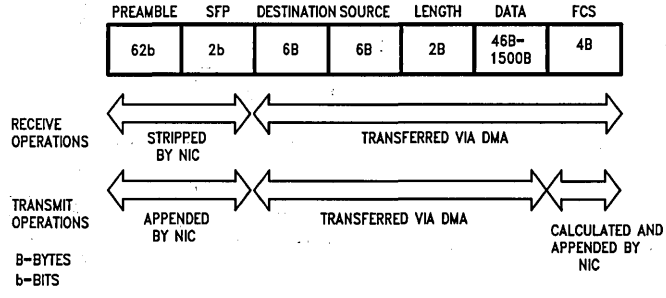
The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the ST-NIC.

DATA FIELD

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. **The ST-NIC does not strip or append pad bytes for short packets, or check for oversize packets.**

FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected. The AUTODIN II ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$) polynomial is used for the CRC calculations.



TL/F/11157–7

FIGURE 2

6.0 Direct Memory Access Control (DMA)

The DMA capabilities of the ST-NIC greatly simplify the use of the DP83902 in typical configurations. The local DMA channel transfers data between the FIFO and memory. On transmission, the packet is DMAed from memory to the FIFO in bursts. Should a collision occur (up to 15 times), the packet is retransmitted with no processor intervention. On reception, packets are DMAed from the FIFO to the receive buffer ring (as explained below).

A remote DMA channel is also provided on the ST-NIC to accomplish transfers between a buffer memory and system memory. The two DMA channels can alternatively be combined to form a single 32-bit address with 8- or 16-bit data.

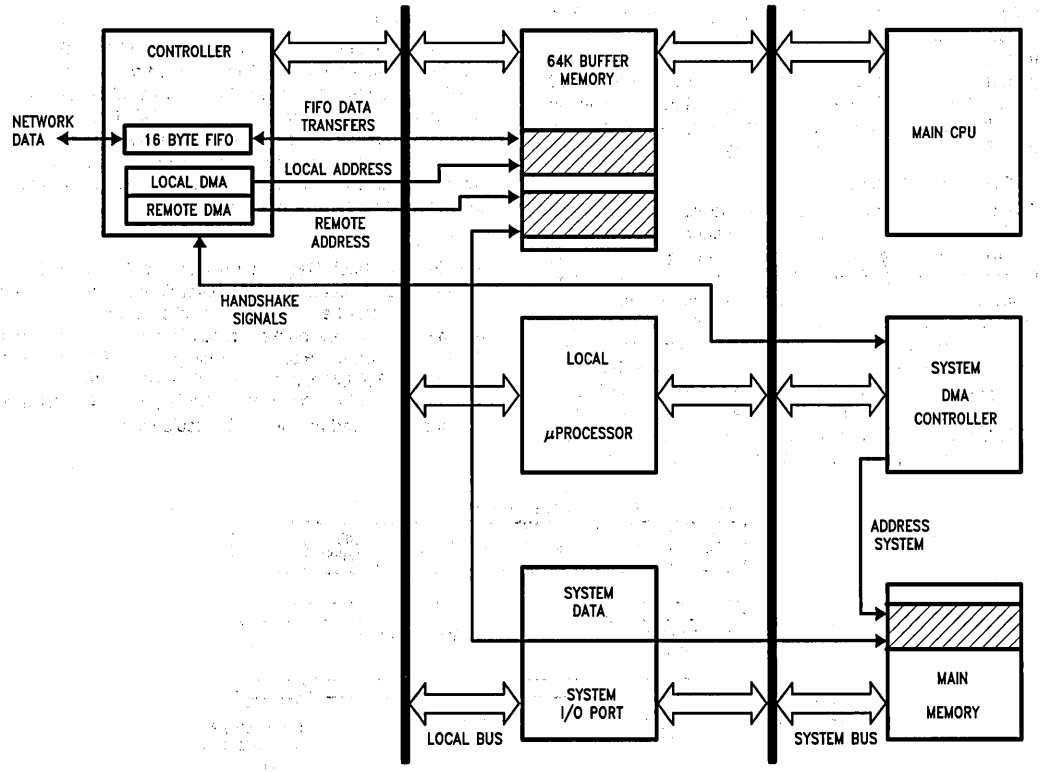
DUAL DMA CONFIGURATION

An example configuration using both the local and remote DMA channels is shown below. Network activity is isolated on a local bus, where the ST-NIC's local DMA channel performs burst transfers between the buffer memory and the ST-NIC's FIFO. The Remote DMA transfers data between the buffer memory and the host memory via a bidirectional I/O port. The Remote DMA provides local addressing capability and is used as a slave DMA by the host. Host side addressing must be provided by a host DMA or the CPU. The ST-NIC allows Local and Remote DMA operations to be interleaved.

SINGLE CHANNEL DMA OPERATION

If desirable, the two DMA channels can be combined to provide a 32-bit DMA address. The upper 16 bits of the 32-bit address are static and are used to point to a 64 kbyte (or 32k word) page of memory where packets are to be received and transmitted.

Dual Bus System



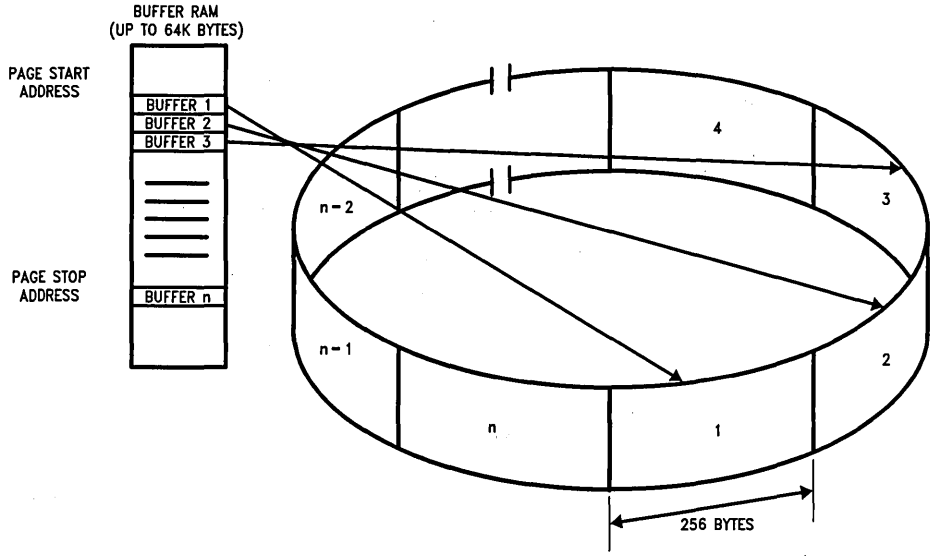
TL/F/11157–8

7.0 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256-byte (128 word) buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop Register. Ethernet packets consist of a distribution of shorter link control packets and longer data packets, the 256-byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers for storing packets is controlled by Buffer Management Logic in the ST-NIC. The Buffer Management Logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

At initialization, a portion of the 64 kbyte (or 32k word) address space is reserved for the receive buffer ring. Two 8-bit registers, The Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The ST-NIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.

ST-NIC Receive Buffer Ring



TL/F/11157–9

7.0 Packet Reception (Continued)

INITIALIZATION OF THE BUFFER RING

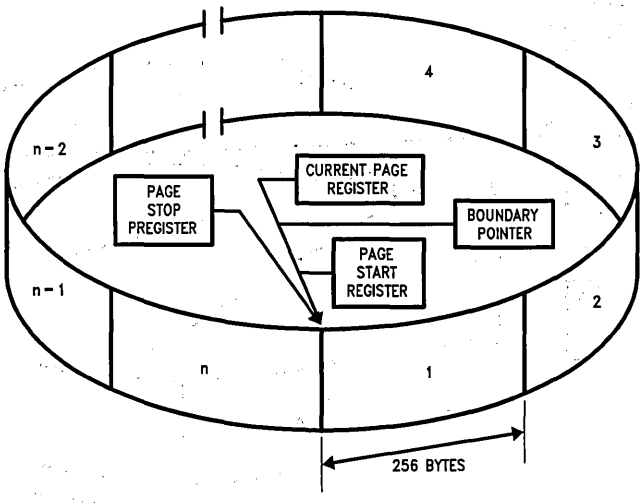
Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register (both described previously), the Current Page Register and the Boundary Pointer Register. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.

Note: The Page Start Register must not be initialized to 00H.

BEGINNING OF RECEPTION

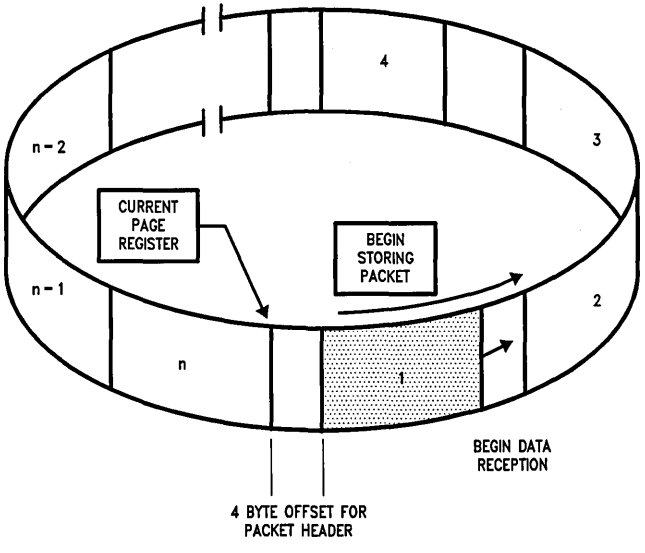
When the first packet begins arriving the ST-NIC begins storing the packet at the location pointed to by the Current Page Register. An offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.

Buffer Ring at Initialization



TL/F/11157–10

Received Packet Enters the Buffer Pages



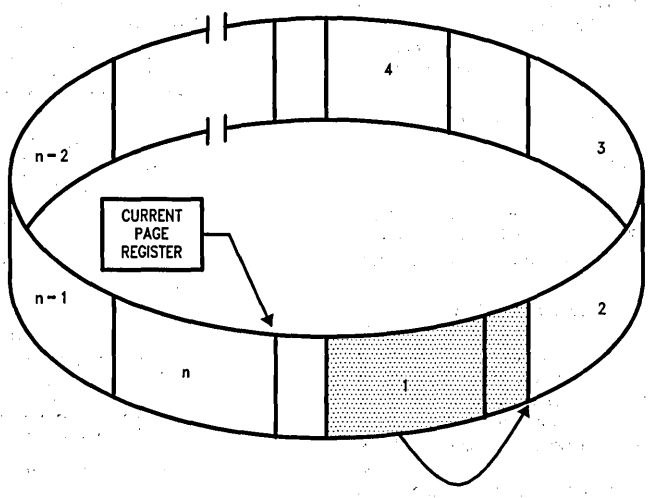
TL/F/11157–11

7.0 Packet Reception (Continued)

LINKING RECEIVE BUFFER PAGES

If the length of the packet exhausts the first 256-byte buffer, the DMA performs a forward link to the next buffer to store the remainder of the packet. For a maximum length packet the buffer logic will link six buffers to store the entire packet. Buffers cannot be skipped when linking, therefore a packet will always be stored in contiguous buffers. Before the next buffer can be linked, the Buffer Management Logic performs two comparisons. The first comparison tests for equality between the DMA address of the next buffer and the contents of the Page Stop Register. If the buffer address equals the Page Stop Register, the buffer management logic will restore the DMA to the first buffer in the Receive Buffer Ring value programmed in the Page Start Address Register. The second comparison tests for equality between the DMA address of the next buffer address and the contents of the Boundary Pointer Register. If the two values are equal the reception is aborted. The Boundary Pointer Register can be used to protect against overwriting any area in the receive buffer ring that has not yet been read. When linking buffers, buffer management will never cross this pointer, effectively avoiding any overwrites. If the buffer address does not match either the Boundary Pointer or Page Stop Address, the link to the next buffer is performed.

Linking Receive Buffer Pages



TL/F/11157–12

1) Check for = to PSTOP
2) Check for = to Boundary

7.0 Packet Reception (Continued)

Buffer Ring Overflow

If the Buffer Ring has been filled and the DMA reaches the Boundary Pointer Address, reception of the incoming packet will be aborted by the ST-NIC. Thus, the packets previously received and still contained in the Ring will not be destroyed.

In heavily loaded network which cause overflows of the Receive Buffer Ring, the ST-NIC may disable the local DMA and suspend further receptions even if the Boundary register is advanced beyond the Current register. To guarantee this will not happen, a software reset must be issued during all Receive Buffer Ring overflows (indicated by the OVW bit in the Interrupt Status Register). The following procedure is required to recover from a Receiver Buffer Ring Overflow.

If this routine is not adhered to, the ST-NIC may act in an unpredictable manner. It should also be noted that it is not permissible to service an overflow interrupt by continuing to empty packets from the receive buffer without implementing the prescribed overflow routine. A flow chart of the ST-NIC's overflow routine follows.

Note: It is necessary to define a variable in the driver, which will be called "Resend".

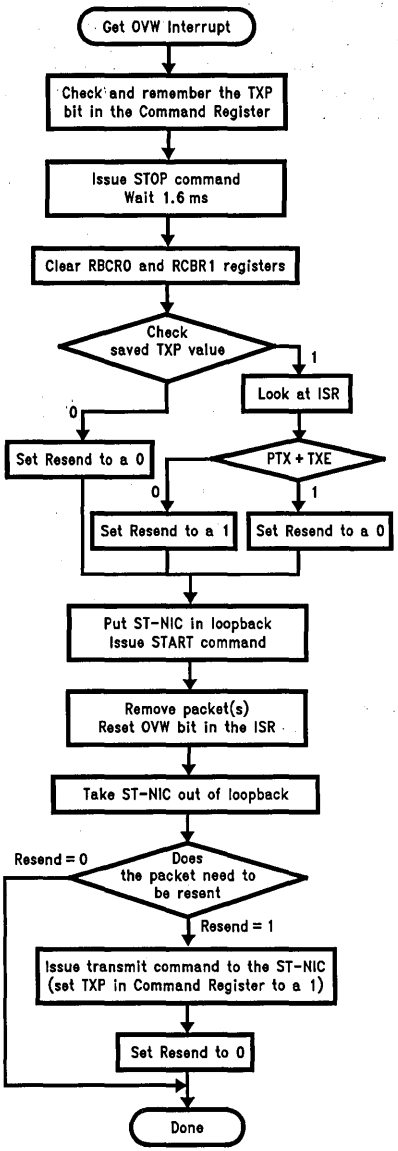
1. Read and store the value of the TXP bit in the ST-NIC Command Register.
2. Issue the STOP command to the ST-NIC. This is accomplished by setting the STP bit in the ST-NIC's Command Register. Writing 21H to the Command Register will stop the ST-NIC.
3. Wait for at least 1.6 ms. Since the ST-NIC will complete any transmission or reception that is in progress, it is necessary to time out for the maximum possible duration of an Ethernet transmission or reception. By waiting 1.6 ms this is achieved with some guard band added. Previously, it was recommended that the RST bit of the Interrupt Status Register be polled to insure that the pending transmission or reception is completed. This bit is not a reliable indicator and subsequently should be ignored.
4. Clear the ST-NIC's Remote Byte Count registers (RBCR0 and RBCR1).
5. Read the stored value of the TXP bit from step 1, above.

   If this value is a 0, set the "Resend" variable to a 0 and jump to step 6.

   If this value is a 1, read the ST-NIC's Interrupt Status Register. If either the Packet Transmitted bit (PTX) or Transmit Error bit (TXE) is set to a 1, set the "Resend" variable to a 0 and jump to step 6. If neither of these bits is set, place a 1 in the "Resend" variable and jump to step 6.

   This step determines if there was a transmission in progress when the stop command was issued in step 2. If there was a transmission in progress, the ST-NIC's ISR is read to determine whether or not the packet was recognized by the ST-NIC. If neither the PTX nor TXE bit was set, then the packet will essentially be lost and retransmitted only after a time-out takes place in the upper level software. By determining that the packet was lost at the driver level, a transmit command can be reissued to the ST-NIC once the overflow routine is completed (as in step 11). Also, it is possible for the ST-NIC to defer indefinitely, when it is stopped on a busy network. Step 5 also alleviates this problem. Step 5 is essential and should not be omitted from the overflow routine, in order for the ST-NIC to operate correctly.

Overflow Routine Flow Chart



TL/F/11157–57

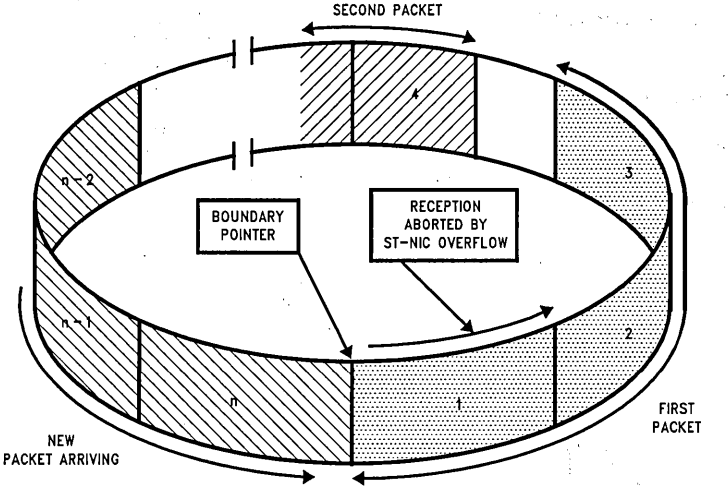
7.0 Packet Reception (Continued)

6. Place the ST-NIC in either mode 1 or mode 2 loopback. This can be accomplished by setting bits D2 and D1, of the Transmit Configuration Register, to "0,1" or "1,0", respectively.
7. Issue the START command to the ST-NIC. This can be accomplished by writing 22H to the Command Register. This is necessary to activate the ST-NIC's Remote DMA channel.
8. Remove one or more packets from the receive buffer ring.
9. Reset the overwrite warning (OVW, overflow) bit in the Interrupt Status Register.
10. Take the ST-NIC out of loopback. This is done by writing the Transmit Configration Register with the value it contains during normal operation. (Bits D2 and D1 should both be programmed to 0.)
11. If the "Resend" variable is set to a 1, reset the "Resend" variable and reissue the transmit command. This is done by writing a value of 26H to the Command Register. If the "Resend" variable is 0, nothing needs to be done.

Note 1: If Remote DMA is not being used, the ST-NIC does not need to be started before packets can be removed from the receive buffer ring. Hence, step 8 could be done before step 7.

Note 2: When the ST-NIC is in STOP mode, the Missed Talley Counter is disabled

Received Packet Aborted If It Hits Boundary



TL/F/11157–13

7.0 Packet Reception (Continued)

Enabling the ST-NIC On An Active Network

After the ST-NIC has been initialized the procedure for disabling and then re-enabling the ST-NIC on the network is similar to handling Receive Buffer Ring overflow as described previously.

1. Program Command Register for page 0 (Command Register = 21H)
2. Initialize Data Configuration Register (DCR)
3. Clear Remote Byte Count Registers (RBCR0, RBCR1)
4. Initialize Receive Configuration Register (RCR)
5. Place the ST-NIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6. Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7. Clear Interrupt Status Register (ISR) by writing 0FFH to it.
8. Initialize Interrupt Mask Register (IMR)
9. Program Command Register for page 1 (Command Register = 61H)
   i. Initialize Physical Address Registers (PAR0–PAR5)
   ii. Initialize Multicast Address Registers (MAR0–MAR7)
   iii. Initialize CURRent pointer
10. Put ST-NIC in START mode (Command Register = 22H). The local receive DMA is still not active since the ST-NIC is in LOOPBACK.
11. Initialize the Transmit Configuration for the intended value. The ST-NIC is now ready for transmission and reception.

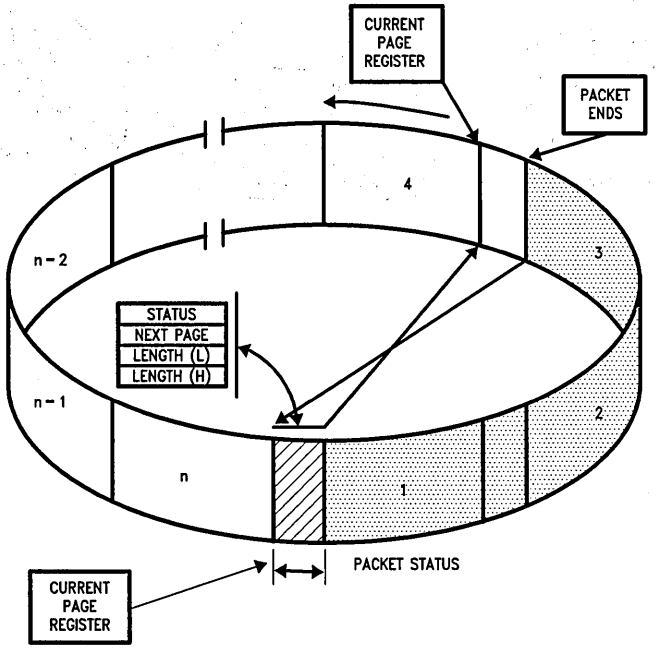
END OF PACKET OPERATIONS

At the end of the packet the ST-NIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

SUCCESSFUL RECEPTION

If the packet is successfully received, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. Note that the remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring. (The location of the next buffer had been previously calculated and temporarily stored in an internal scratchpad register.)

Termination of Received Packet—Packet Accepted



TL/F/11157–14

7.0 Packet Reception (Continued)

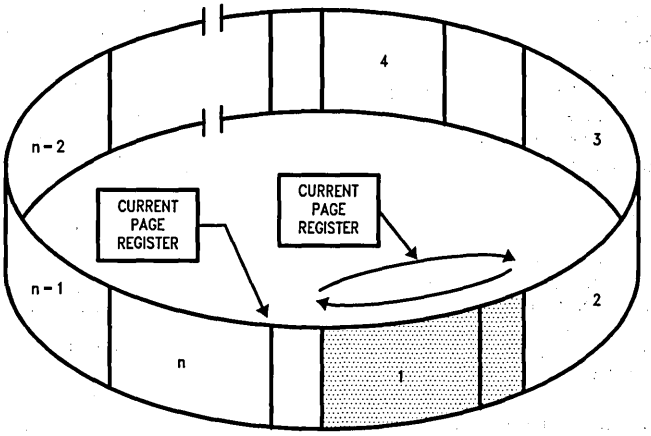
BUFFER RECOVERY FOR REJECTED PACKETS

If the packet is a runt packet or contains CRC or Frame Alignment errors, it is rejected. The buffer management logic resets the DMA back to the first buffer page used to store the packet (pointed to by CURR), recovering all buffers that had been used to store the rejected packet. This operation will not be performed if the ST-NIC is programmed to accept either runt packets or packets with CRC or Frame Alignment errors. The received CRC is always stored in buffer memory after the last byte of received data for the packet.

Error Recovery

If the packet is rejected as shown, the DMA is restored by the ST-NIC by reprogramming the DMA starting address pointed to by the Current Page Register.

Termination of Receive Packet—Packet Reject



TL/F/11157–15

7.0 Packet Reception (Continued)

REMOVING PACKETS FROM THE RING

Packets are removed from the ring using the Remote DMA or an external device. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the ST-NIC moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer.

The following is a suggested method for maintaining the Receive Buffer Ring pointers.

1. At initialization, set up a software variable (next_pkt) to indicate where the next packet will be read. At the beginning of each Remote Read DMA operation, the value of next_ pkt will be loaded into RSAR0 and RSAR1.
2. When initializing the ST-NIC set:
   BNDRY = PSTART
   CURR = PSTART + 1
   next_pkt = PSTART + 1
3. After a packet is DMAed from the Receive Buffer Ring, the Next Page Pointer (second byte in the ST-NIC buffer header is used to updte BNDRY and next_pkt.
   next_pkt = Next Page Pointer
   BNDRY = Next Page Pointer − 1
   If BNDRY <PSTART then BNDRY = PSTOP − 1

Note the size of the Receive Buffer Ring is reduced by one 256-byte buffer, this will not, however, impede the operation of the ST-NIC.

STORAGE FORMAT FOR RECEIVED PACKETS

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register.

AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 1		Receive Byte Count 0	
Byte 2		Byte 1	

BOS = 0, WTS = 1 in Data Configuration Register. This format is used with Series 32xxx, or 680x0 processors.

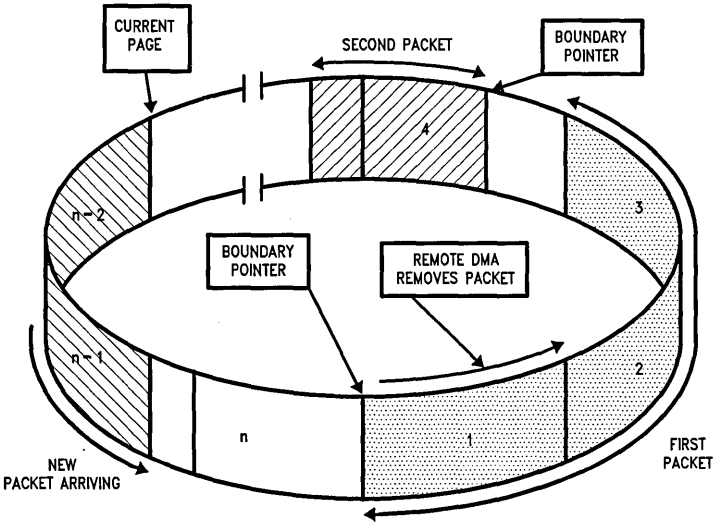
AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 0		Receive Byte Count 1	
Byte 1		Byte 2	

BOS = 1, WTS = 1 in Data Configuration Register. This format is used with 680x0 type processors. (**Note:** The Receive Count ordering remains the same for BOS = 0 or 1.)

Receive Status
Next Packet Pointer
Receive Byte Count 0
Receive Byte Count 1
Byte 0
Byte 1

BOS = 0, WTS = 0 in Data Configuration Register. This format is used with general 8-bit processors.

1st Received Packet Removed by Remote DMA



TL/F/11157–16

8.0 Packet Transmission

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0, 1). When the ST-NIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The ST-NIC will generate and append the preamble, synch and CRC fields.

General Transmit Packet Format

Transmit	Destination Address	6 Bytes
Byte	Source Address	6 Bytes
Count	Type/Length	2 Bytes
TBCR0, 1	Data Pad (If Data < 46 Bytes)	≥46 Bytes

TRANSMIT PACKET ASSEMBLY

The ST-NIC requires a contiguous assembled packet with the format shown. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes. The programmer is responsible for adding and stripping pad bytes.

TRANSMISSION

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR) is cleared and the ST-NIC begins to prefetch transmit data from memory (unless the ST-NIC is currently receiving). If the interframe gap has timed out the ST-NIC will begin transmission.

CONDITIONS REQUIRED TO BEGIN TRANSMISSION

In order to transmit a packet, the following three conditions must be met:

1. The Interframe Gap Timer has timed out the first 6.4 μs of the Interframe Gap.
2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started.)
3. If a collision has been detected the backoff timer has expired.

In typical systems the ST-NIC prefetchs the first burst of bytes before the 6.4 μs timer expires. The time during which ST-NIC transmits preamble can also be used to load the FIFO.

Note: If carrier sense is asserted before a byte has been loaded into the FIFO, the ST-NIC will become a receiver.

COLLISION RECOVERY

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

Note: NCR reads as zeroes if excessive collisions are encountered.

TRANSMIT PACKET ASSEMBLY FORMAT

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register.

D15 D8	D7 D0
Destination Address 1	Destination Address 0
Destination Address 3	Destination Address 2
Destination Address 5	Destination Address 4
Source Address 1	Source Address 0
Source Address 3	Source Address 2
Source Address 5	Source Address 4
Type/Length 1	Type/Length 0
Data 1	Data 0

BOS = 0, WTS = 1 in Data Configuration Register.

This format is used with Series 32xxx, or 808xx processors.

D15 D8	D7 D0
Destination Address 0	Destination Address 1
Destination Address 2	Destination Address 3
Destination Address 4	Destination Address 5
Source Address 0	Source Address 1
Source Address 2	Source Address 3
Source Address 4	Source Address 5
Type/Length 0	Type/Length 1
Data 0	Data 1

BOS = 1, WTS = 1 in Data Configuration Register.

This format is used with 680x0 type processors.

D1 D0
Destination Address 0
Destination Address 1
Destination Address 2
Destination Address 3
Destination Address 4
Destination Address 5
Source Address 0
Source Address 1
Source Address 2
Source Address 3
Source Address 4
Source Address 5

BOS = 0, WTS = 0 in Data Configuration Register.

This format is used with general 8-bit processors.

Note: All examples above will result in a transmission of a packet in order of DA0, DA1, DA2, DA3 . . . bits within each byte will be transmitted least significant bit first.

DA = Destination Address.

9.0 Remote DMA

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet.

Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) register pair and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address Register pair points to the beginning of the block to be moved while the Byte Count Register pair is used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

REMOTE WRITE

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

REMOTE READ

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

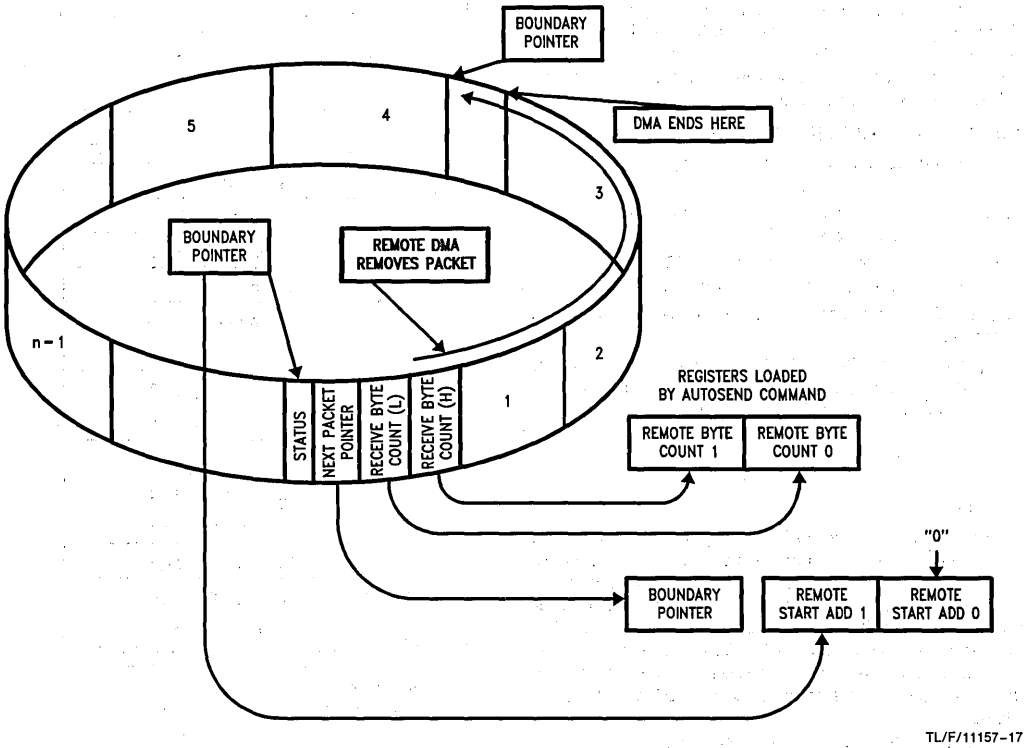
SEND PACKET COMMAND

The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a "Send Packet" Command. The DMA will be initialized to the value of the Boundary Pointer Register and the Remote Byte Count Register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop Register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring.

Note 1: In order for the ST-NIC to correctly execute the Send Packet Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 0FH.

Note 2: The Send Packet command cannot be used with 680x0 type processors.

Remote DMA Autoinitialization from Buffer Ring

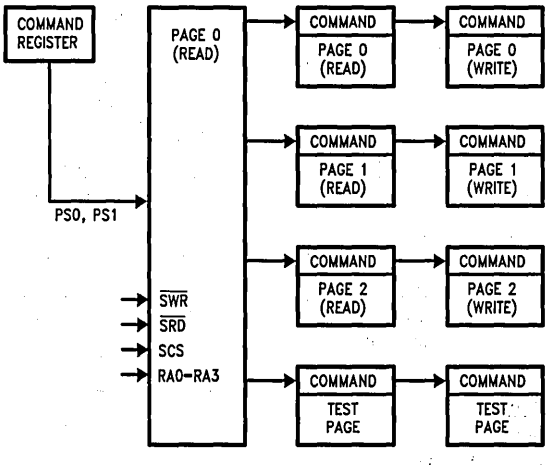


TL/F/11157–17

10.0 Internal Registers

All registers are 8-bit wide and mapped into four pages which are selected in the Command Register (PS0, PS1). Pins RA0–RA3 are used to address registers within each page. Page 0 registers are those registers which are commonly accessed during ST-NIC operation while page 1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.

10.1 REGISTER ADDRESS MAPPING



TL/F/11157–18

10.2 REGISTER ADDRESS ASSIGNMENTS

Page 0 Address Assignments (PS1 = 0, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

10.0 Internal Registers (Continued)

Page 1 Address Assignments (PS1 = 0, PS0 = 1)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

Page 2 Address Assignments (PS1 = 1, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Page Start Register (PSTART)	Current Local DMA Address 0 (CLDA0)
02H	Page Stop Register (PSTOP)	Current Local DMA Address 1 (CLDA1)
03H	Remote Next Packet Pointer	Remote Next Packet Pointer
04H	Transmit Page Start Address (TPSR)	Reserved
05H	Local Next Packet Pointer	Local Next Packet Pointer
06H	Address Counter (Upper)	Address Counter (Upper)
07H	Address Counter (Lower)	Address Counter (Lower)
08H	Reserved	Reserved
09H	Reserved	Reserved
0AH	Reserved	Reserved
0BH	Reserved	Reserved
0CH	Receive Configuration Register (RCR)	Reserved
0DH	Transmit Configuration Register (TCR)	Reserved
0EH	Data Configuration Register (DCR)	Reserved
0FH	Interrupt Mask Register (IMR)	Reserved

Note: Page 2 registers should only be accessed for diagnostic purposes. They should not be modified during normal operation.

Page 3 should never be modified.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS

COMMAND REGISTER (CR) 00H (READ/WRITE)

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands may be overlapped, but with the following rules: (1) If a transmit command overlaps with a remote DMA operation, bits RD0, RD1, and RD2 must be maintained for the remote DMA command when setting the TXP bit. Note, if a remote DMA command is re-issued when giving the transmit command, the DMA will complete immediately if the remote byte count register has not been reinitialized. (2) If a remote DMA operation overlaps a transmission, RD0, RD1, and RD2 may be written with the desired values and a "0" written to the TXP bit. Writing a "0" to this bit has no effect. (3) A remote write DMA may not overlap remote read operation or vice versa. Either of these operations must either complete or be aborted before the other operation may start. Bits PS1, PS0, RD2, and STP may be set any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

Bit	Symbol	Description
D0	STP	**Stop:** Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to 1. **STP powers up high.** **Note:** If the ST-NIC has previously been in start mode and the STP is set, both the STP and STA bits will remain set.
D1	STA	**Start:** This bit is used to activate the ST-NIC after either power up, or when the ST-NIC has been placed in a reset mode by software command or error. **STA powers up low.**
D2	TXP	**Transmit Packet:** This bit must be set to initiate the transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed.
D3, D4, and D5	RD0, RD1, and RD2	**Remote DMA Command:** These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. RD2 RD1 RD0 0 0 0 Not Allowed 0 0 1 Remote Read 0 1 0 Remote Write (Note 2) 0 1 1 Send Packet 1 X X Abort/Complete Remote DMA (Note 1) **Note 1:** If a remote DMA operation is aborted and the remote byte count has not decremented to zero. PRQ will remain high. A read acknowledge (RACK) on a write acknowledge (WACK) will reset PRQ low. **Note 2:** For proper operation of the Remote Write DMA, there are two steps which must be performed before using the Remote Write DMA. The steps are as follows: I) Write a non-zero value into RBCR0. II) Set bits RD2, RD1, and RD0 to 0, 0, and 1. III) Set RBCR0, 1 and RSAR0, 1. IV) Issue the Remote Write DMA Command (RD2, RD1, RD0 = 0, 1, 0).
D6 and D7	PS0 and PS1	**Page Select:** These two encoded bits select which register page is to be accessed with addresses RA0–3. PS1 PS0 0 0 Register Page 0 0 1 Register Page 1 1 0 Register Page 2 1 1 Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

INTERRUPT STATUS REGISTER (ISR) 07H (READ/WRITE)

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a "1" into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX

Bit	Symbol	Description
D0	PRX	**Packet Received:** Indicates packet received with no errors.
D1	PTX	**Packet Transmitted:** Indicates packet transmitted with no errors.
D2	RXE	**Receive Error:** Indicates that a packet was received with one or more of the following errors: — CRC Error — Frame Alignment Error — FIFO Overrun — Missed Packet
D3	TXE	**Transmit Error:** Set when packet transmitted with one or more of the following errors: — Excessive Collisions — FIFO Underrun
D4	OVW	**Overwrite Warning:** Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer)
D5	CNT	**Counter Overflow:** Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	**Remote DMA Complete:** Set when Remote DMA operation has been completed.
D7	RST	**Reset Status:** Set when ST-NIC enters reset state and cleared when a Start Command is issued to the CR. This bit is also set when a Receive Buffer Ring overflow occurs and is cleared when one or more packets have been removed from the ring. Writing to this bit has no effect. **Note:** This bit does not generate an interrupt, it is merely a status indicator.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

INTERRUPT MASK REGISTER (IMR) 0FH (WRITE)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set, an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. **The IMR powers up to all zeroes.**

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Bit	Symbol	Description
D0	PRXE	**Packet Received Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet received
D1	PTXE	**Packet Transmitted Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted
D2	RXEE	**Receive Error Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet received with error
D3	TXEE	**Transmit Error Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error
D4	OVWE	**Overwrite Warning Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet
D5	CNTE	**Counter Overflow Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set
D6	RDCE	**DMA Complete Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed
D7	Reserved	Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

DATA CONFIGURATION REGISTER (DCR) 0EH (WRITE)

This Register is used to program the ST-NIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO thresholds. **The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.**

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	LS	LAS	BOS	WTS

Bit	Symbol	Description
D0	WTS	**Word Transfer Select** 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers ; WTS establishes byte or word transfers for both Remote and Local DMA transfers **Note:** When word-wide mode is selected up to 32k words are addressable; A0 remains low.
D1	BOS	**Byte Order Select** 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32xxx, 80x86) 1: MS byte placed on AD7–AD0 and LS byte on AD15–A8. (680x0) ; Ignored when WTS is low
D2	LAS	**Long Address Select** 0: Dual 16-bit DMA mode 1: Single 32-bit DMA mode ; When LAS is high, the contents of the Remote DMA registers RSAR0, 1 are issued as A16–A31 Power up high
D3	LS	**Loopback Select** 0: Loopback mode selected. Bits D1 and D2 of the TCR must also be programmed for Loopback operation 1: Normal Operation
D4	ARM	**Auto-Initialize Remote** 0: Send Command not executed, all packets removed from Buffer Ring under program control 1: Send Command executed, Remote DMA auto-initialized to remove packets from Buffer Ring **Note:** Send Command cannot be used with 680x0 byte processors.
D5 and D6	FT0 and FT1	**FIFO Threshold Select:** Encoded FIFO threshold. Establishes point at which bus is requested when filling or emptying the FIFO. During reception, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled serially from the network before bus request (BREQ) is asserted. **Note:** FIFO threshold setting determines the DMA burst length. Receive Thresholds FT1 FT0 Word Wide Byte Wide 0 0 1 Word 2 Bytes 0 1 2 Words 4 Bytes 1 0 4 Words 8 Bytes 1 1 6 Words 12 Bytes During transmission, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled from the Local DMA before BREQ is asserted. Thus, the transmission threshold is 16 bytes less the received threshold.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

TRANSMIT CONFIGURATION REGISTER (TCR) 0DH (WRITE)

The transmit configuration establishes the actions of the transmitter section of the ST-NIC during transmission of a packet on the network. **LB1 and LB0 which select loopback mode power up as 0.**

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

Bit	Symbol	Description
D0	CRC	**Inhibit CRC** 0: CRC appended by transmitter 1: CRC inhibited by transmitter In loopback mode CRC can be enabled or disabled to test the CRC logic
D1 and D2	LB0 and LB1	**Encoded Loopback Control:** These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 places the ENDEC Module in loopback mode and that D3 of the DCR must be set to zero for loopback operation. LB1 LB0 Mode 0 0 0 Normal Operation (LPBK = 0) Mode 1 0 1 Internal NIC Module Loopback (LPBK = 0) Mode 2 1 0 Internal ENDEC Module Loopback (LPBK = 1) Mode 3 1 1 External Loopback (LPBK = 0)
D3	ATD	**Auto Transmit Disable:** This bit allows another station to disable the ST-NIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.
D4	OFST	**Collision Offset Enable:** This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n, 10)}$ slot times for first three collisions, then follows standard backoff. (For the first three collisions, the station has higher average backoff delay making a low priority mode.)
D5	Reserved	Reserved
D6	Reserved	Reserved
D7	Reserved	Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

TRANSMIT STATUS REGISTER (TSR) 04H (READ)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	—	PTX

Bit	Symbol	Description
D0	PTX	**Packet Transmitted:** Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0")
D1	Reserved	Reserved
D2	COL	**Transmit Collided:** Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	**Transmit Aborted:** Indicates the ST-NIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16.)
D4	CRS	**Carrier Sense Lost:** This bit is set when carrier is lost during transmission of the packet. Transmission is not aborted on loss of carrier.
D5	FU	**FIFO Underrun:** If the ST-NIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	**CD Heartbeat:** Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 μs of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	**Out of Window Collision:** Indicates that a collision occurred after a slot time (51.2 μs). Transmissions rescheduled as in normal collisions.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

RECEIVE CONFIGURATION REGISTER (RCR) 0CH (WRITE)

This register determines operation of the ST-NIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

Bit	Symbol	Description
D0	SEP	**Save Errored Packets** 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	**Accept Runt Packets:** This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	**Accept Broadcast:** Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	**Accept Multicast:** Enables the receiver to accept a packet with a multicast address. All multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	**Promiscuous Physical:** Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	**Monitor Mode:** Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	Reserved	Reserved
D7	Reserved	Reserved

Note: D2 and D3 are "OR'd" together, i.e., if D2 and D3 are set the ST-NIC will accept broadcast and multicast addresses as well as its own physical address. To establish full promiscuous mode, bits D2, D3, and D4 should be set. In addition the multicast hashing array must be set to all 1's in order to accept all multicast addresses.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

RECEIVE STATUS REGISTER (RSR) 0CH (READ)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the ST-NIC which relinguishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

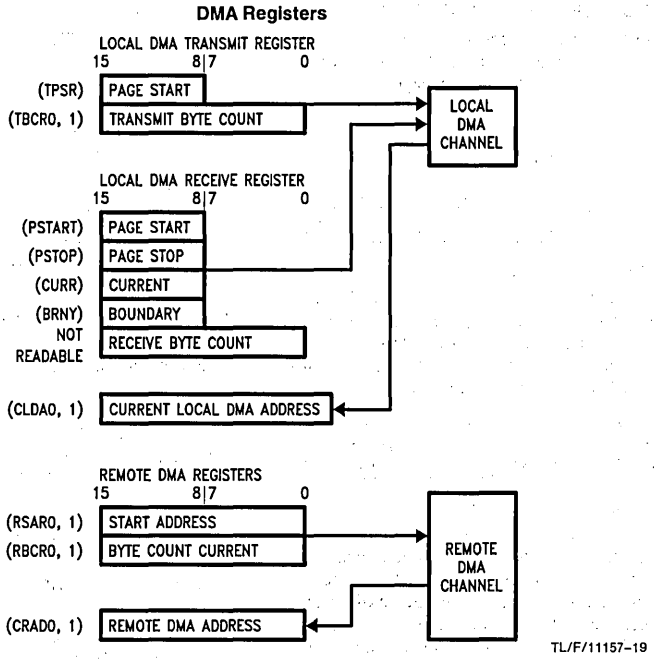
Bit	Symbol	Description
D0	PRX	**Packet Received Intact:** Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	**CRC Error:** Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	**Frame Alignment Error:** Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at the last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	**FIFO Overrun:** This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	**Missed Packet:** Set when a packet intended for node cannot be accepted by ST-NIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	**Physical/Multicast Address:** Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Physical Address Match
D6	DIS	**Receiver Disabled:** Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	**Deferring:** Set when internal Carrier Sense or Collision signals are generated in the ENDEC module. If the transceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.

Note: Following coding applies to CRC and FAE bits.

FAE	CRC	Type of Error
0	0	No Error (Good CRC and <6 Dribble Bits)
0	1	CRC Error
1	0	Illegal, Will Not Occur
1	1	Frame Alignment Error and CRC Error

10.0 Internal Registers (Continued)

10.4 DMA REGISTERS



The DMA Registers are partitioned into groups; Transmit, Receive and Remote DMA Registers. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive Registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary Registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.

Note: In the figure above, registers are shown as 8 or 16 bits wide. Although some registers are 16-bit internal registers, all registers are accessed as 8-bit registers. Thus the 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0, TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus. Thus, they are shifted to positions 15–8 in the diagram above.

10.5 TRANSMIT DMA REGISTERS

TRANSMIT PAGE START REGISTER (TPSR)

This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256-byte page boundaries. The bit assignment is shown below. The values placed in bits D7–D0 will be used to initialize the higher order address (A8–A15) of the Local DMA for transmission. The lower order bits (A7–A0) are initialized to zero.

Bit Assignment

	7	6	5	4	3	2	1	0
TPSR	A15	A14	A13	A12	A11	A10	A9	A8

(A7–A0 Initialized to Zero)

TRANSMIT BYTE COUNT REGISTER 0, 1 (TBCR0, TBCR1)

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64 Kbytes. The ST-NIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
TBCR1	L15	L14	L13	L12	L11	L10	L9	L8

	7	6	5	4	3	2	1	0
TBCR0	L7	L6	L5	L4	L3	L2	L1	L0

10.0 Internal Registers (Continued)

10.6 LOCAL DMA RECEIVE REGISTERS

PAGE START AND STOP REGISTERS (PSTART, PSTOP)

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the ST-NIC uses fixed 256-byte buffers aligned on page boundaries only the upper 8 bits of the start and stop address are specified.

PSTART, PSTOP Bit Assignment

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

BOUNDARY (BNRY) REGISTER

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT PAGE REGISTER (CURR)

This register is used internally by the Buffer Management Logic as a backup register for reception. CURR contains the address of the first buffer to be used for a packet reception and is used to restore DMA pointers in the event of receive errors. This register is initialized to the same value as PSTART and should not be written to again unless the controller is Reset.

	7	6	5	4	3	2	1	0
CURR	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT LOCAL DMA REGISTER 0,1 (CLDA0, 1)

These two registers can be accessed to determine the current local DMA address.

	7	6	5	4	3	2	1	0
CLDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CLDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.7 REMOTE DMA REGISTERS

REMOTE START ADDRESS REGISTERS (RSAR0, 1)

Remote DMA operations are programmed via the Remote Start Address (RSAR0, 1) and Remote Byte Count (RBCR0, 1) registers. The Remote Start Address is used to point to the start of the block of data to be transferred and the Remote Byte Count is used to indicate the length of the block (in bytes).

	7	6	5	4	3	2	1	0
RSAR1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
RSAR0	A7	A6	A5	A4	A3	A2	A1	A0

REMOTE BYTE COUNT REGISTERS (RBCR0, 1)

	7	6	5	4	3	2	1	0
RBCR1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8

	7	6	5	4	3	2	1	0
RBCR0	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

Note: RSAR0 programs the start address bits A0–A7.
RSAR1 programs the start address bits A8–A15.
Address incremented by two for word transfers, and by one for byte transfers. Byte Count decremented by two for word transfers and by one for byte transfers.
RBCR0 programs LSB byte count.
RBCR1 programs MSB byte count.

CURRENT REMOTE DMA ADDRESS (CRDA0, CRDA1)

The Current Remote DMA Registers contain the current address of the Remote DMA. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
CRDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CRDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.8 PHYSICAL ADDRESS REGISTERS (PAR0–PAR5)

The physical address registers are used to compare the destination address of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte-wide basis. The bit assignment shown below relates the sequence in PAR0–PAR5 to the bit sequence of the received packet.

	D7	D6	D5	D4	D3	D2	D1	D0
PAR0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
PAR1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
PAR2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
PAR3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
PAR4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
PAR5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40

					Destination Address			Source	
P/S	DA0	DA1	DA2	DA3	...	DA46	DA47	SA0	...

Note: P/S = Preamble, Synch
DA0 = Physical/Multicast Bit

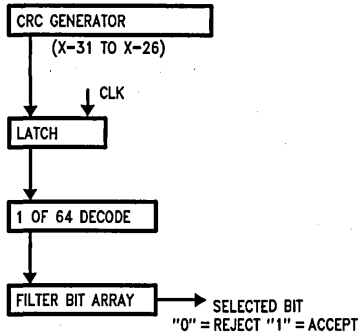
10.9 MULTICAST ADDRESS REGISTERS (MAR0–MAR7)

The multicast address registers provide filtering of multicast addresses hashed by the CRC logic. All destination addresses are fed through the CRC logic and as the last bit of the destination address enters the CRC, the 6 most signifi-

10.0 Internal Registers (Continued)

cant bits of the CRC generator are latched. These 6 bits are then decoded by a 1 of 64 decode to index a unique filter bit (FB0–63) in the multicast address registers. If the filter bit selected is set, the multicast packet is accepted. The system designer would use a program to determine which filter bits to set in the multicast registers. All multicast filter bits that correspond to multicast address accepted by the node are then set to one. To accept all multicast packets all of the registers are set to all ones.

Note: Although the hashing algorithm does not guarantee perfect filtering of multicast address, it will perfectly filter up to 64 multicast addresses if these addresses are chosen to map into unique locations in the multicast filter.



TL/F/11157–53

	D7	D6	D5	D4	D3	D2	D1	D0
MAR0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MAR1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MAR2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MAR3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MAR4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MAR5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MAR6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MAR7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

If address Y is found to hash to the value 32 (20H), then FB32 in MAR4 should be initialized to "1". This will cause the ST-NIC to accept any multicast packet with the address Y.

10.10 NETWORK TALLY COUNTERS

Three 8-bit counters are provided for monitoring the number of CRC errors, Frame Alignment Errors and Missed Packets. The maximum count reached by any counter is 192 (C0H). These registers will be cleared when read by the CPU. The count is recorded in binary in CT0–CT7 of each Tally Register.

Frame Alignment Error Tally (CNTR0)

This counter increments every time a packet is received with a Frame Alignment Error. The packet must have been recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR0	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

CRC Error Tally (CNTR1)

This counter is incremented every time a packet is received with a CRC error. The packet must first be recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR1	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

Frames Lost Tally Register (CNTR2)

This counter is incremented if a packet cannot be received due to lack of buffer resources. In monitor mode, this counter will count the number of packets that pass the address recognition logic.

	7	6	5	4	3	2	1	0
CNTR2	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

FIFO

This is an 8-bit register that allows the CPU to examine the contents of the FIFO after loopback. The FIFO will contain the last 8 data bytes transmitted in the loopback packet. Sequential reads from the FIFO will advance a pointer in the FIFO and allow reading of all 8 bytes.

	7	6	5	4	3	2	1	0
FIFO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Note: The FIFO should only be read when the ST-NIC has been programmed in loopback mode.

NUMBER OF COLLISIONS (NCR)

This register contains the number of collisions a node experiences when attempting to transmit a packet. If no collisions are experienced during a transmission attempt, the COL bit of the TSR will not be set and the contents of NCR will be zero. If there are excessive collisions, the ABT bit in the TSR will be set and the contents of NCR will be zero. The NCR is cleared after the TXP bit in the CR is set.

	7	6	5	4	3	2	1	0
NCR	—	—	—	—	NC3	NC2	NC1	NC0

11.0 Initialization Procedures

The ST-NIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the ST-NIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The ST-NIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the ST-NIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

Initialization Sequence

The following initialization procedure is mandatory.

1. Program Command Register for Page 0 (Command Register = 21H)
2. Initialize Data Configuration Register (DCR)
3. Clear Remote Byte Count Registers (RBCR0, RBCR1)
4. Initialize Receive Configuration Register (RCR)
5. Place the ST-NIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6. Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7. Clear Interrupt Status Register (ISR) by writing 0FFH to it.
8. Initialize Interrupt Mask Register (IMR)
9. Program Command Register for page 1 (Command Register = 61H)
   I) Initialize Physical Address Registers (PAR0–PAR5)
   II) Initialize Multicast Address Registers (MAR0–MAR5)
   III) Initialize CURRent pointer
10. Put ST-NIC in START mode (Command Register = 22H).
11. Initialize the Transmit Configuration Register for the intended value. The ST-NIC is now ready for transmission and reception.

Before receiving packets, the user must specify the location of the Receive Buffer Ring. This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Register must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

12.0 Loopback Diagnostics

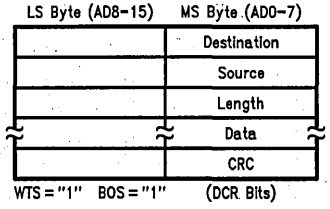
Three forms of local loopback are provided on the ST-NIC. The user has the ability to loopback through the deserializer on the controller, through the ENDEC module or the Transceiver. **Because of the half duplex architecture of the ST-NIC, loopback testing is a special mode of operation with the following restrictions:**

Restrictions During Loopback

The FIFO is split into two halves, one half is used for transmission and the other for reception. Only 8-bit fields can be fetched from memory so two tests are required for 16-bit systems to verify integrity of the entire data path. During loopback the maximum latency from the assertion of BREQ to BACK is 2.0 μs. Systems that wish to use the loopback test but do not meet this latency can limit the loopback to 7 bytes without experiencing underflow. Only the last 8 bytes of the loopback packet are retained in the FIFO. The last 8 bytes can be read through the FIFO register which will advance through the FIFO to allow reading the receive packet sequentially.

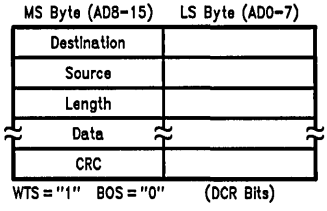
Destination Address	= (6 Bytes) Station Physical Address
Source Address	|
Length	2 Bytes
Data	= 46 to 1500 Bytes
CRC	Appended by ST-NIC if CRC = "0" in TCR

When in word-wide mode with Byte Order Select set, the loopback packet must be assembled in the even byte location as shown below. (Loopback only operates with byte wide transfers.)



TL/F/11157–54

When in word-wide mode with Byte Order Select low, the following format must be used for the loopback packet.



TL/F/11157–55

Note: When using loopback in word mode 2n bytes must be programmed in TBCR0, 1. Where n = actual number of bytes assembled in even or odd location.

12.0 Loopback Diagnostics (Continued)

To initiate a loopback the user first assembles the loopback packet then selects the type of loopback using the Transmit Configuration register bits LB0, LB1. The transmit configuration register must also be set to enable or disable CRC generation during transmission. The user then issues a normal transmit command to send the packet. During loopback the receiver checks for an address match and if CRC bit in the TCR is set, the receiver will also check the CRC. The last 8 bytes of the loopback packet are buffered and can read out of the FIFO using the FIFO read port.

Loopback Modes

MODE 1: Loopback through the NIC Module (LB1 = 0, LB0 = 1): If this loopback is used, the NIC Modules's serializer is connected to the deserializer.

MODE 2: Loopback through the ENDEC Module (LB1 = 1, LB0 = 0): If the loopback is to be performed through the SNI, the ST-NIC provides a control (LPBK) that forces the ENDEC module to loopback all signals.

MODE 3: Loopback to cable (LB1 = 1, LB0 = 1). Packets can be transmitted to the cable in loopback mode to check all of the transmit and receive paths and the cable itself.

Note: Collision and Carrier Sense can be generated by the ENDEC module and are masked by the NIC module. It is not possible to go directly between the loopback modes, it is necessary to return to normal operation (00H) when changing modes.

Reading the Loopback Packet

The last 8 bytes of a received packet can be examined by 8 consecutive reads of the FIFO register. The FIFO pointer is incremented after the rising edge of the CPU's read strobe by internally synchronizing and advancing the pointer. This may take up to four bus clock cycles, if the pointer has not been incremented by the time the CPU reads the FIFO register again, the ST-NIC will insert wait states.

Note: The FIFO may only be read during Loopback. Reading the FIFO at any other time will cause the ST-NIC to malfunction.

Alignment of the Received Packet in the FIFO

Reception of the packet in the FIFO begins at location zero, after the FIFO pointer reaches the last location in the FIFO, the pointer wraps to the top of the FIFO overwriting the previously received data. This process is continued until the last byte is received. The ST-NIC then appends the received byte count in the next two locations of the FIFO. The contents of the Upper Byte Count are also copied to the next FIFO location. The number of bytes used in the loopback packet determines the alignment of the packet in the FIFO.

The alignment for a 64-byte packet is shown below.

FIFO Location	FIFO Contents	
0	Lower Byte Count	First Byte Read
1	Upper Byte Count	Second Byte Read
2	Upper Byte Count	•
3	Last Byte	•
4	CRC1	•
5	CRC2	•
6	CRC3	•
7	CRC4	Last Byte Read

For the following alignment in the FIFO the packet length should be (N x 8) + 5 Bytes. Note that if the CRC bit in the TCR is set, CRC will not be appended by the transmitter. If the CRC is appended by the transmitter, the 1st four bytes, bytes N-3 to N, correspond to the CRC.

FIFO Location	FIFO Contents	
0	Byte N-4	First Byte Read
1	Byte N-3 (CRC1)	Second Byte Read
2	Byte N-2 (CRC2)	•
3	Byte N-1 (CRC3)	•
4	Byte N (CRC4)	•
5	Lower Byte Count	•
6	Upper Byte Count	Last Byte Read
7	Upper Byte Count	

LOOPBACK TESTS

Loopback capabilities are provided to allow certain tests to be performed to validate operation of the DP83902 ST-NIC prior to transmitting and receiving packets on a live network. Typically these tests may be performed during power up of a node. The diagnostic provides support to verify the following:

1. Verify integrity of data path. Received data is checked against transmitted data.
2. Verify the CRC logic's capability to generate good CRC on transmit, verify CRC on receive (good or bad CRC).

12.0 Loopback Diagnostics (Continued)

3. Verify that the Address Recognition Logic can
   a) Recognize address match packets
   b) Reject packets that fail to match an address

LOOPBACK OPERATION IN THE ST-NIC

Loopback is a modified form of transmission using only half of the FIFO. This places certain restrictions on the use of loopback testing. When loopback mode is selected in the TCR, the FIFO is split. A packet should be assembled in memory with programming of TPSR and TBCR0, TBCR1 registers. When the transmit command is issued the following operations occur:

Transmitter Actions

1. Data is transferred from memory by the DMA until the FIFO is filled. For each transfer TBCR0 and TBCR1 are decremented. (Subsequent burst transfers are initiated when the number of bytes in the FIFO drops below the programmed threshold.)
2. The ST-NIC generates 56 bits of preamble followed by an 8-bit synch pattern.
3. Data transferred from FIFO to serializer.
4. If CRC = 1 in TCR, the CRC is not calculated by ST-NIC, and the last byte transmitted is the last byte from the FIFO (Allows software CRC to be appended). If CRC = 0, ST-NIC calculates and appends four bytes of CRC.
5. At end of Transmission PTX bit set in ISR.

Receiver Actions

1. Wait for synch, all preamble stripped.
2. Store packet in FIFO, increment receive byte count for each incoming byte.
3. If CRC = 1 in TRC, receiver checks incoming packet for CRC errors. If CRC = 0 in TCR, receiver does not check CRC errors, CRC error bit always set in RSR (for address matching packets).
4. At the end of receive, the receive byte count is written into the FIFO, and the receive status register is updated. The PRX bit is typically set in the RSR even if the address does not match. If CRC errors are forced, the packet must match the address filters in order for the CRC error bit in the RSR to be set.

EXAMPLES

The following examples show what results can be expected from a properly operating ST-NIC during loopback. The restrictions and results of each type of loopback are listed for reference. The loopback tests are divided into two sets of tests. One to verify the data path, CRC generation and byte count through all three paths. The second set of tests uses internal loopback to verify the receiver's CRC checking and address recognition. For all of the tests the DCR was programmed to 40H.

Path	TCR	RCR	TSR	RSR	ISR
ST-NIC Internal	02	00	53 (Note 1)	02 (Note 2)	02 (Note 3)

Note 1: Since carrier sense and collision detect are generated in the ENDEC module, they are blocked during NIC loopback. Carrier and CD heartbeat are not seen and the CRS and CDH bits are set.

Note 2: CRC errors are always indicated by the receiver if CRC is appended by the transmitter.

Note 3: Only the PTX bit in the ISR is set, the PRX bit is only set if status is written to memory. In loopback this action does not occur and the PRX bit remains 0 for all loopback modes.

Note 4: All values are hex.

Path	TCR	RCR	TSR	RSR	ISR
ST-NIC Internal	04	00	43 (Note 1)	02	02

Note 1: CDH is set, CRS is not set since it is generated by the external encoder/decoder.

Path	TCR	RCR	TSR	RSR	ISR
ST-NIC External	06	00	03 (Note 1)	02	02 (Note 2)

Note 1: CDH and CRS should not be set. The TSR however, could also contain 01H, 03H, 07H and a variety of other values depending on whether collisions were encountered or the packet was deferred.

Note 2: The ISR will contain 08H if packet is not transmittable.

Note 3: During external loopback the ST-NIC is now exposed to network traffic. It is therefore possible for the contents of both the Receive portion of the FIFO and the RSR to be corrupted by any other packet on the network. Thus in a live network the contents of the FIFO and RSR should not be depended on. The ST-NIC will still abide by the standard CSMA/CD protocol in external loopback mode. (i.e., The network will not be disturbed by the loopback packet.)

Note 4: All values are hex.

CRC AND ADDRESS RECOGNITION

The next three tests exercise the address recognition logic and CRC. These tests should be performed using internal loopback only so that the ST-NIC is isolated from interference from the network. These tests also require the capability to generate CRC in software.

The address recognition logic cannot be directly tested. The CRC and FAE bits in the RSR are only set if the address in the packet matches the address filters. If errors are expected to be set and they are not set, the packet has been rejected on the basis of an address mismatch. The following sequence of packets will test the address recognition logic. The DCR should be set to 40H and the TCR should be set to 03H with a software generated CRC.

Packet Contents			Results
Test	Address	CRC	RSR
Test A	Matching	Good	01 (Note 1)
Test B	Matching	Bad	02 (Note 2)
Test C	Non-Matching	Bad	01

Note 1: Status will read 21H if multicast address used.

Note 2: Status will read 22H if multicast address used.

Note 3: In test A, the RSR is set up. In test B the address is found to match since the CRC is flagged as bad. Test C proves that the address recognition logic can distinguish a bad address and does not notify the RSR of the bad CRC. The receiving CRC is proven to work in test A and test B.

Note 4: All values are hex.

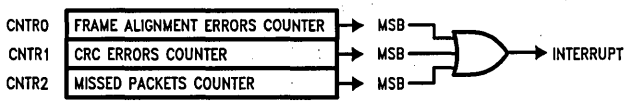
NETWORK MANAGEMENT FUNCTIONS

Network management capabilities are required for maintenance and planning of a local area network. The ST-NIC supports the minimum requirement for network management in hardware, the remaining requirements can be met with software counts. There are three events that software alone can not track during reception of packets: CRC errors, Frame Alignment errors, and missed packets.

12.0 Loopback Diagnostics (Continued)

Since errored packets can be rejected, the status associated with these packets is lost unless the CPU can access the Receive Status Register before the next packer arrives. In situations where another packet arrives very quickly, the CPU may have no opportunity to do this. The ST-NIC counts the number of packets with CRC errors and Frame Alignment errors. 8-Bit counters have been selected to reduce overhead. The counters will generate interrupts whenever their MSBs are set so that a software routine can accumulate the network statistics and reset the counter before overflow occurs. The counters are sticky so that when they reach a count of 192 (C0H) counting is halted. An additional counter is provided to count the number of packets the ST-NIC misses due to buffer overflow or being offline.

The structure of the counters is shown below:



TL/F/11157–20

Additional information required for network management is available in the Receive and Transmit Status Registers. Transmit status is available after each transmission for information regarding events during transmission.

Typically, the following statistics might be gathered in software:

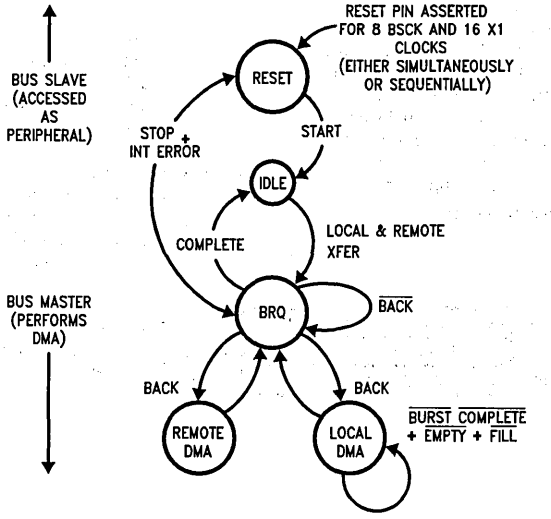
Traffic: Frames Sent OK
Frames Received OK
Multicast Frames Received
Packets Lost Due to Lack of Resources
Retries/Packet

Errors: CRC Errors
Alignment Errors
Excessive Collisions
Packet with Length Errors
Heartbeat Failure

13.0 Bus Arbitration and Timing

The ST-NIC operates in three possible modes:

- BUS MASTER (WHILE PERFORMING DMA)
- BUS SLAVE (WHILE BEING ACCESSED BY CPU)
- IDLE



TL/F/11157–21

Upon power-up the ST-NIC is in an indeterminate state. After receiving a hardware reset the ST-NIC is a bus slave in the Reset State, the receiver and transmitter are both disabled in this state. The reset state can be re-entered under three conditions, soft reset (Stop Command), hard reset (RESET input) or an error that shuts down the receiver of transmitter (FIFO underflow or overflow). After initialization of registers, the ST-NIC is issued a Start command and the ST-NIC enters Idle state. Until the DMA is required the ST-NIC remains in idle state. The idle state is exited by a request from the FIFO on the case of receiver or transmit, or from the Remote DMA in the case of Remote DMA operation. After acquiring the bus in a BREQ/BACK handshake the Remote or Local DMA transfer is completed and the ST-NIC re-enters the idle state.

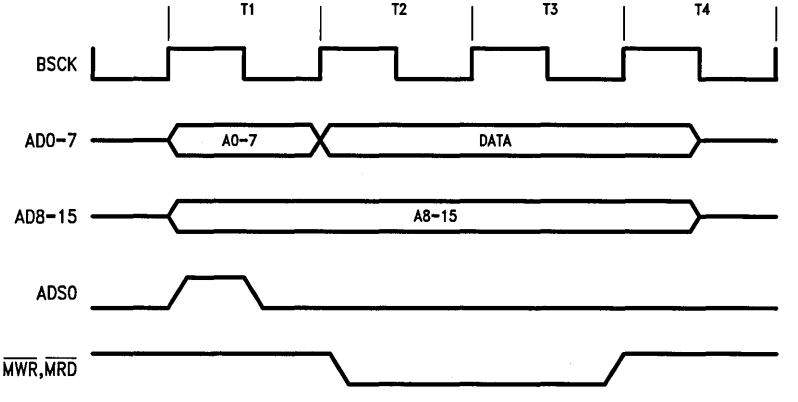
DMA TRANSFERS TIMING

The DMA can be programmed for the following types of transfers:

16-Bit Address, 8-bit Data Transfer
16-Bit Address, 16-bit Data Transfer
32-Bit Address, 8-bit Data Transfer
32-Bit Address, 16-bit Data Transfer

All DMA transfers use BSCK for timing. 16-Bit Address modes require 4 BSCK cycles as shown below:

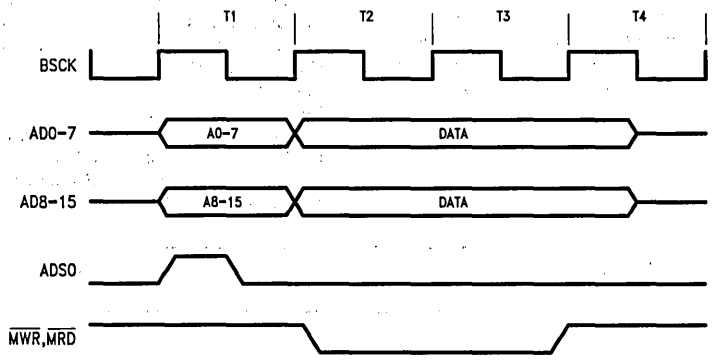
16-Bit Address, 8-Bit Data



TL/F/11157–22

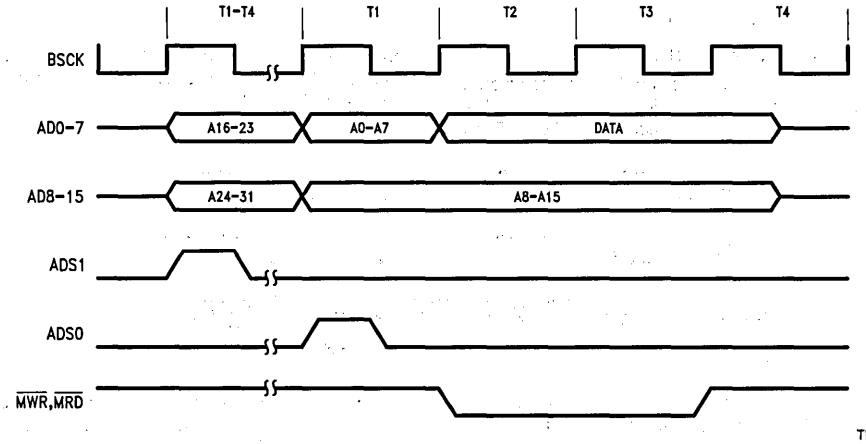
13.0 Bus Arbitration and Timing (Continued)

16-Bit Address, 16-Bit Data



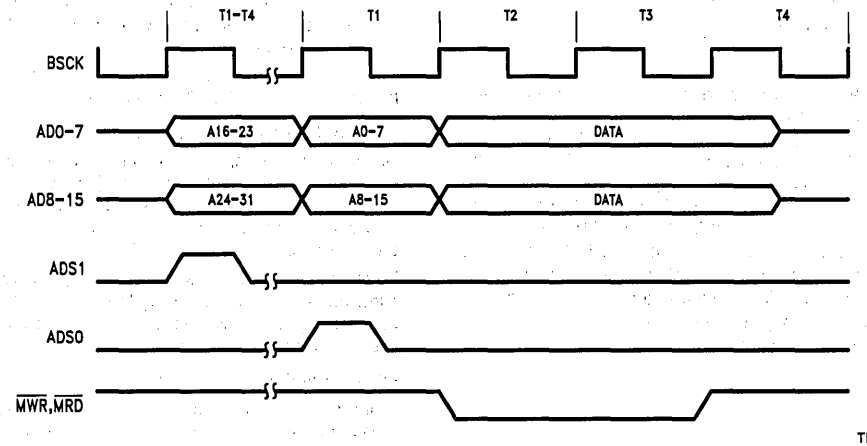
TL/F/11157-23

32-Bit Address, 8-Bit Data



TL/F/11157-24

32-Bit Address, 16-Bit Data



TL/F/11157-25

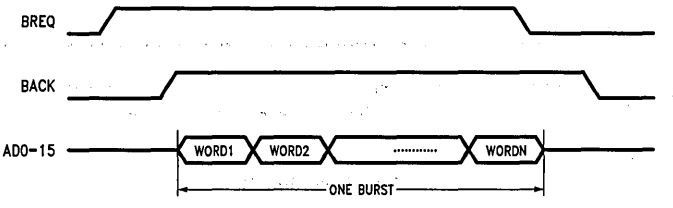
Note: In 32-bit address mode, ADS1 at TRI-STATE after the first T1–T4 states: thus, a 4.7k pull-down resistor is required for 32-bit address.

13.0 Bus Arbitration and Timing (Continued)

When in 32-bit mode four additional BSCK cycles are required per burst. The first bus cycle (T1'–T4') of each burst is used to output the upper 16-bit addresses. This 16-bit address is programmed in RSAR0 and RSAR1 and points to a 64k page of system memory. All transmitted or received packets are constrained to reside within this 64k page.

FIFO BURST CONTROL

All Local DMA transfer are burst transfers, once the DMA requests the bus and the bus is acknowledged, the DMA will transfer an exact burst of bytes programmed in the Data Configuration Register (DCR) then relinquish the bus. If there are remaining bytes in the FIFO the next burst will not be initiated until the FIFO threshold is exceeded. If desired the DMA can empty/fill the FIFO when it acquires the bus. If BACK is removed during the transfer, the burst transfer will be aborted. **(DROPPING BACK DURING A DMA CYCLE IS NOT RECOMMENDED.)**

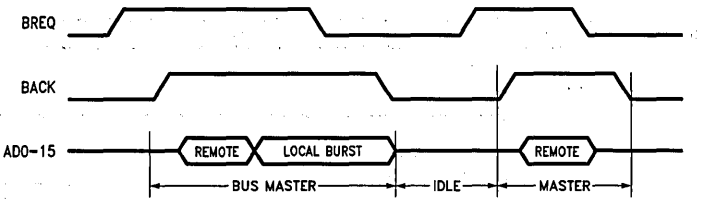


TL/F/11157–26

where N = 1, 2, 4, or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode.

INTERLEAVED LOCAL OPERATION

If a remote DMA transfer is initiated or in progress when a packet is being received or transmitted, the Remote DMA transfer will be interrupted for higher priority Local DMA transfers. When the Local DMA transfer is completed the Remote DMA will rearbitrate for the bus and continue its transfers. This is illustrated below:



TL/F/11157–27

Note that if the FIFO requires service while a remote DMA is in progress, BREQ is not dropped and the Local DMA burst is appended to the Remote Transfer. When switching from a local transfer to a remote transfer, however, BREQ is dropped and raised again. This allows the CPU or other devices to fairly contend for the bus.

FIFO AND BUS OPERATIONS

Overview

To accommodate the different rates at which data comes from (or goes to) the network and goes to (or comes from) the system memory, the ST-NIC contains a 16-byte FIFO for buffering data between the bus and the media. The FIFO threshold is programmable, allowing filling (or emptying) the FIFO at different rates. When the FIFO has filled to its programmed threshold, the local DMA channel transfers these bytes (or words) into local memory. It is crucial that the local DMA is given access to the bus within a minimum bus latency time; otherwise a FIFO underrun (or overrun) occurs.

To understand FIFO underruns or overruns, there are two causes which produce this condition—

1. the bus latency is so long that the FIFO has filled (or emptied) from the network before the local DMA has serviced the FIFO.
2. the bus latency or bus data rate has slowed the throughput of the local DMA to a point where it is slower than the network data rate (10 Mb/s). This second condition is also dependent upon DMA clock and word width (byte wide or word wide).

The worst case condition ultimately limits the overall bus latency which the ST-NIC can tolerate.

FIFO Underrun and Transmit Enable

During transmission, if a FIFO underrun occurs, the Transmit enable (TXE) output may remain high (active). Generally, this will cause a very large packet to be transmitted onto the network. The jabber feature of the transceiver will terminate the transmission, and reset TXE.

To prevent this problem, a properly designed system will not allow FIFO underruns by giving the ST-NIC a bus acknowledge within time shown in the maximum bus latency curves shown and described later.

FIFO at the Beginning of Receive

At the beginning of reception, the ST-NIC stores entire Address field of each incoming packet in the FIFO to deter-

13.0 Bus Arbitration and Timing (Continued)

mine whether the packet matches its Physical Address Registers or maps to one of its Multicast Registers. This causes the FIFO to accumulate 8 bytes. Furthermore, there are some synchronization delays in the DMA PLA. Thus, the actual time that BREQ is asserted from the time the Start of Frame Delimiter (SFD) is detected is 7.8 μs. This operation affects the bus latencies at 2- and 4-byte thresholds during the first receive BREQ since the FIFO must be filled to 8 bytes (or 4 words) before issuing a BREQ.

FIFO Operation at the End of Receive

When Carrier Sense goes low, the ST-NIC enters its end of packet processing sequence, emptying its FIFO and writing the status information at the beginning of the packet, *Figure 5*. The ST-NIC holds onto the bus for the entire sequence. The longest time BREQ may be extended occurs when a packet ends just as the ST-NIC performs its last FIFO burst. The ST-NIC, in this case, performs a programmed burst transfer followed by flushing the remaining bytes in the FIFO, and completes by writing the header information to memory. The following steps occur during this sequence.

1. ST-NIC issues BREQ because the FIFO threshold has been reached
2. During the burst, packet ends, resulting in BREQ extended.
3. ST-NIC flushes remaining bytes from FIFO
4. ST-NIC performs internal processing to prepare for writing the header.
5. ST-NIC writes 4-byte (2-word) header
6. ST-NIC deasserts BREQ

End of Packet Processing (EOPP) times for 10 MHz and 20 MHz have been tabulated in the table below.

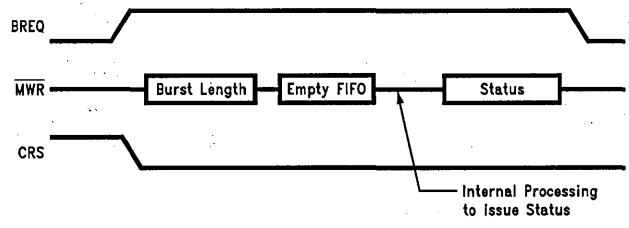
Mode	Threshold	Bus Clock	EOPP
Byte	2 Bytes	10 MHz	7.0 μs
	4 Bytes		8.6 μs
	8 Bytes		11.0 μs
Byte	2 Bytes	20 MHz	3.6 μs
	4 Bytes		4.2 μs
	8 Bytes		5.0 μs
Word	2 Bytes	10 MHz	5.4 μs
	4 Bytes		6.2 μs
	8 Bytes		7.4 μs
Word	2 Bytes	20 MHz	3.0 μs
	4 Bytes		3.2 μs
	8 Bytes		3.6 μs

End of Packet Processing Times for Various FIFO Thresholds, Bus Clocks and Transfer Modes

Threshold Detection (Bus Latency)

To assure that no overwriting of data in the FIFO occurs, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO, effectively shortening the FIFO to 13 bytes. The FIFO logic also operates differently in Byte Mode and in Word Mode. In Byte Mode, a threshold is indicated when the n+1 byte has entered the FIFO; thus, with an 8-byte threshold, the ST-NIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n+2 bytes have entered the FIFO. Thus, with a 4-word threshold (equivalent to 8-byte threshold), BREQ is issued when the 10th byte has entered the FIFO. The two graphs, following, indicate the maximum allowable bus latency for Word and Byte transfer modes.

End of Packet Processing



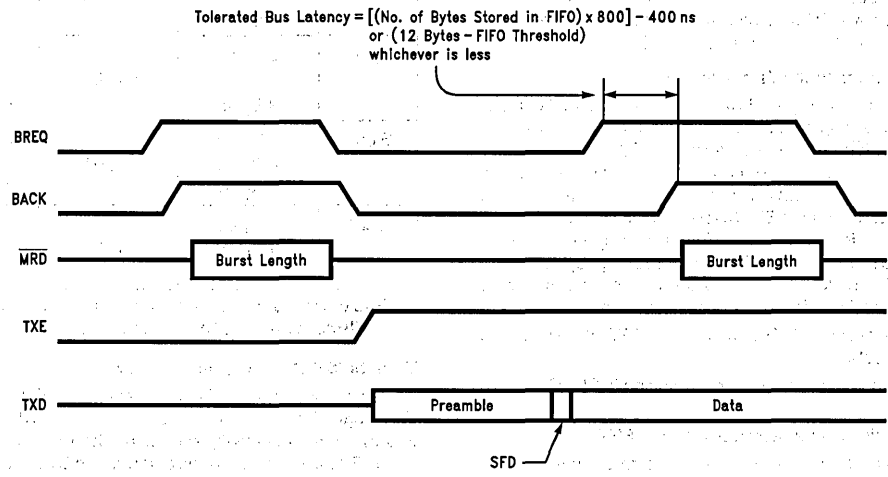
TL/F/11157–58

13.0 Bus Arbitration and Timing (Continued)

The FIFO at the Beginning of Transmit

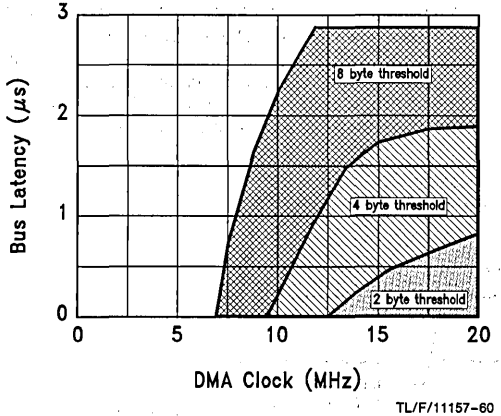
Before transmitting, the ST-NIC performs a prefetch from memory to load the FIFO. The number of bytes prefetched is the programmed FIFO threshold. The next BREQ is not issued until after the ST-NIC actually begins transmitting data, i.e., after SFD. The Transmit Prefetch diagram illustrates this process.

Transmit Prefetch Timing



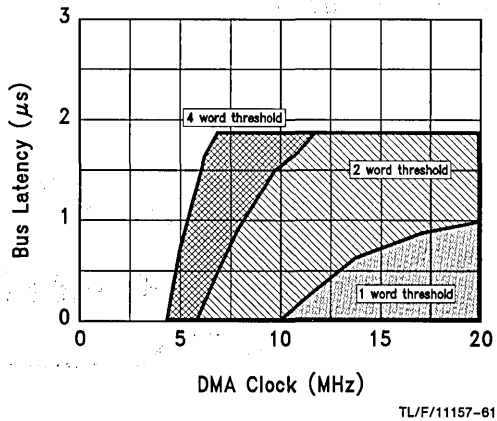
TL/F/11157–59

Maximum Bus Latency for Byte Mode



TL/F/11157–60

Maximum Bus Latency for Word Mode



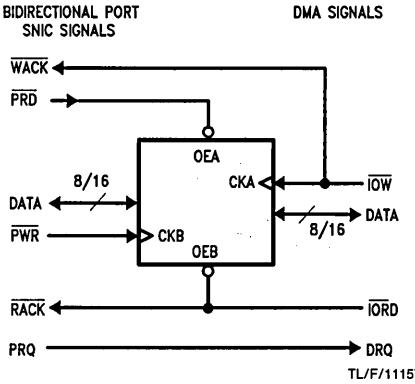
TL/F/11157–61

13.0 Bus Arbitration and Timing (Continued)

REMOTE DMA-BIDIRECTIONAL PORT CONTROL

The Remote DMA transfers data between the local buffer memory and a bidirectional port (memory to I/O transfer). This transfer is arbited on a byte by byte basis versus the burst transfer used for Local DMA transfers. This bidirectional port is also read/written by the host. All transfers through this port are asynchronous. At any one time transfers are limited to one direction, either from the port to local buffer memory (Remote Write) or from local buffer memory to the port (Remote Read).

Bus Handshake Signals for Remote DMA Transfers



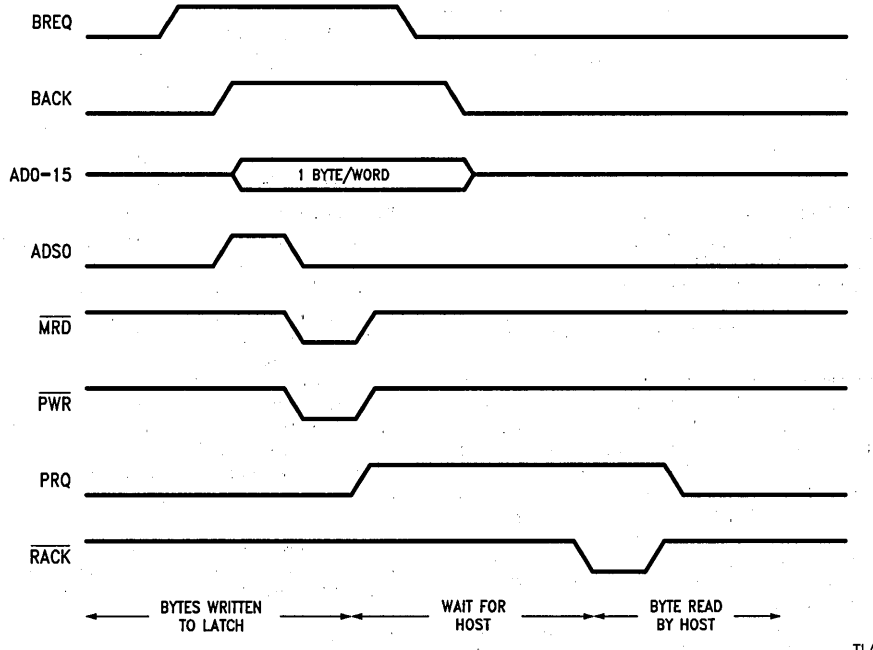
TL/F/11157–28

REMOTE READ TIMING

1. The DMA reads byte/word from local buffer memory and writes byte/word into latch, increments the DMA address and decrements the byte count (RBCR0, 1).
2. A Request Line (PRQ) is asserted to inform the system that a byte is available.
3. The system reads the port, the read strobe (RACK) is used as an acknowledge by the Remote DMA and it goes back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.

Note that in order for the Remote DMA to transfer a byte from memory to the latch, it must arbitrate access to the local bus via a BREQ, BACK handshake. After each byte or word is transferred to the latch, BREQ is dropped. If a Local DMA is in progress, the Remote DMA is held off until the local DMA is complete.



TL/F/11157–29

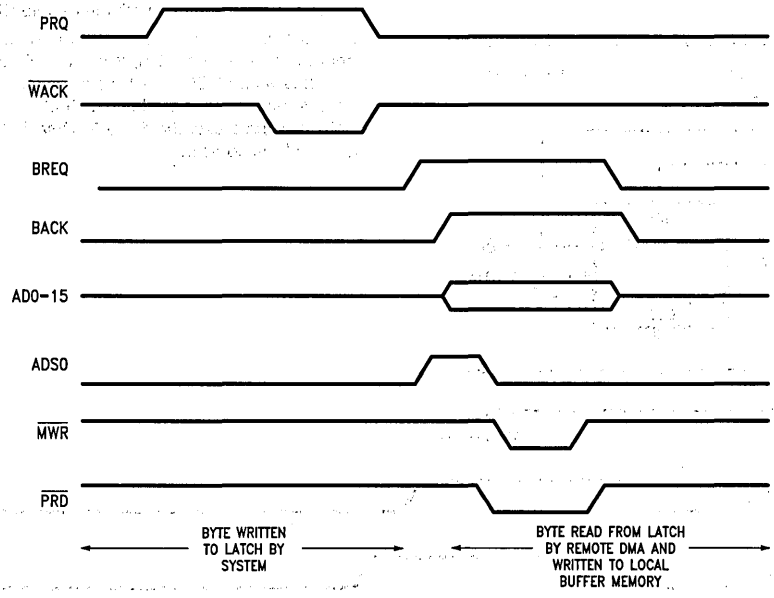
13.0 Bus Arbitration and Timing (Continued)

REMOTE WRITE TIMING

A Remote Write operation transfers data from the I/O port to the local buffer RAM. The ST-NIC initiates a transfer by requesting a byte/word via the PRQ. The system transfers a byte-word to the latch via $\overline{IOW}$. This write strobe is detected by the ST-NIC and PRQ is removed. By removing the PRQ, the Remote DMA holds off further transfers into the latch until the current byte/word has been transferred from the latch, PRQ is reasserted and the next transfer can begin.

1. ST-NIC asserts PRQ. System writes byte/word into latch. ST-NIC removes PRQ.
2. Remote DMA reads contents of port and writes byte/word to local buffer memory, increments address and decrements byte count (RBCR0, 1).
3. Go back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.



TL/F/11157–30

REMOTE DMA WRITE SPECIAL CONSIDERATIONS

Setting PRQ Using the Remote Read

Under certain conditions the ST-NIC bus state machine may issue $\overline{MWR}$ and $\overline{PRD}$ before PRQ for the first DMA transfer of a Remote Write Command. If this occurs this could cause data corruption, or cause the remote DMA count to be different from the main CPU count causing the system to "lock up".

To prevent this condition when implementing a Remote DMA Write, the Remote DMA Write command should first be preceded by a Remote DMA Read command to insure that the PRQ signal is asserted before the ST-NIC starts its port read cycle. The reason for this is that the state machine that asserts PRQ runs independently of the state machine that controls the DMA signals. The DMA machine assumes that PRQ is asserted, but actually may not be. To remedy this situation, a single Remote Read cycle should be inserted before the actual DMA Write Command is given. This will ensure that PRQ is asserted when the Remote DMA Write is subsequently executed. This single Remote Read cycle is called a "dummy Remote Read". In order for the dummy Remote Read cycle to operate correctly, the Start Address should be programmed to a known, safe location in the buffer memory space, and the Remote Byte count should be programmed to a value greater than 1. This will ensure that the master read cycle is performed safely, eliminating the possibility of data corruption.

Remote Write with High Speed Buses

When implementing the Remote DMA Write solution with high speed buses and CPU's, timing may cause the system to hang. Therefore additional considerations are required.

A problem occurs when the system can execute the dummy Remote Read and then start the Remote Write before the ST-NIC has had a chance to execute the Remote Read. If this happens the PRQ signal will not get set, and the Remote Byte Count and Remote Start Address for the Remote Write operation could be corrupted. This is shown by the hatched waveforms in the following timing diagram. The execution of the Remote Read can be delayed by the local DMA operations (particularly during end-of-packet processing).

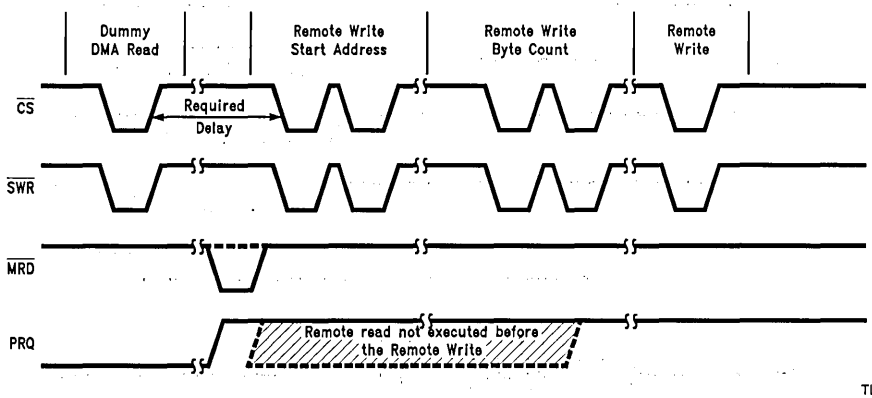
To ensure the dummy Remote Read does execute, a delay must be inserted between writing the Remote Read Command, and starting to write the Remote Write Start Address. (This time is designated in the next figure by the delay arrows.) The recommended method to avoid this problem is after the Remote Read command is given, to poll both bytes of the Current Remote DMA Address Registers. When the address has incremented PRQ has been set. Software should recognize this and then start the Remote Write.

13.0 Bus Arbitration and Timing (Continued)

An additional caution for high speed systems is that the polling must follow guidelines specified in the Time Between Chip Selects section. That is, there must be at least 4 bus clocks between chip selects. (For example when BSCK = 20 MHz, then this time should be 200 ns).

The general flow for executing a Remote Write is:

1. Set Remote Byte Count to a value >1 and Remote Start Address to unused RAM (one location before the transmit start address is usually a safe location).
2. Issue the "dummy" Remote Read command.
3. Read the Current Remote DMA Address (CRDA) (both bytes).
4. Compare to previous CRDA value if different go to 6.
5. Delay and jump to 3.
6. Set up for the Remote Write command, by setting the Remote Byte Count and the Remote Start Address (note that if the Remote Byte count in step 1 can be set to the transmit byte count plus one, and the Remote Start Address to one less, these will now be incremented to the correct values.)
7. Issue the Remote Write command.



TL/F/11157–62

Note: The dashed lines indicate incorrect timing as described in text.

Timing Diagram for Dummy Remote Read

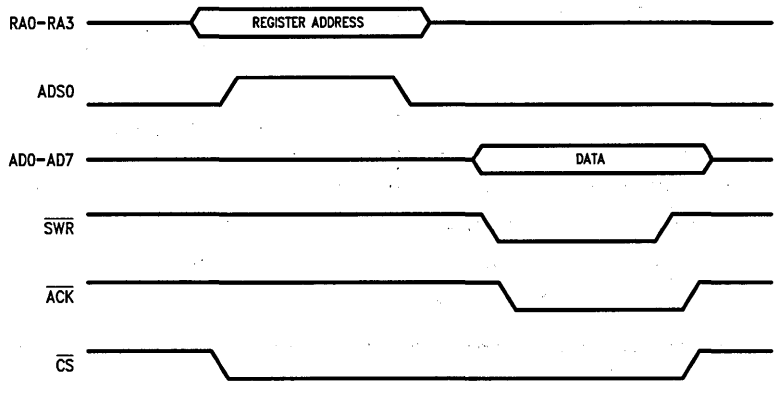
13.0 Bus Arbitration and Timing (Continued)

SLAVE MODE TIMING

When $\overline{CS}$ is low, the ST-NIC becomes a bus slave. The CPU can then read or write any internal registers. All register accesses are byte wide. The timing for register access is shown below. The host CPU accesses internal registers with four address lines, RA0–RA3, $\overline{SRD}$ and $\overline{SWR}$ strobes.

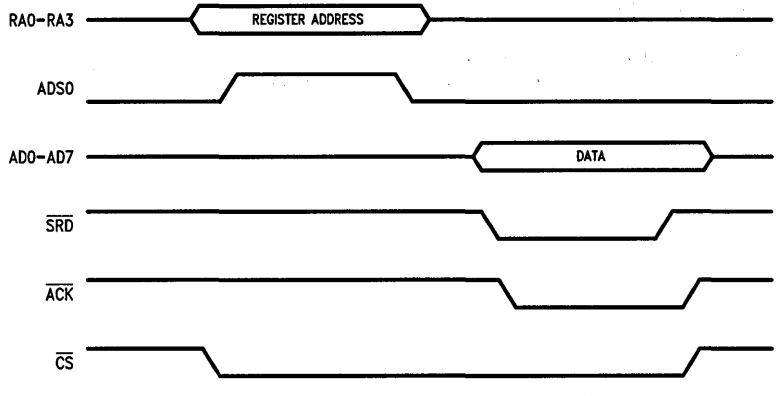
ADS0 is used to latch the address when interfacing to a multiplexed, address data bus. Since the ST-NIC may be a local bus master when the host CPU attempts to read or write to the controller, an $\overline{ACK}$ line is used to hold off the CPU until the ST-NIC leaves master mode. Some number of BSCK cycles is also required to allow the ST-NIC to synchronize to the read or write cycles.

Write to Register



TL/F/11157–31

Read from Register

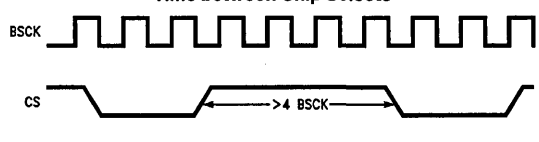


TL/F/11157–32

TIME BETWEEN CHIP SELECTS

The ST-NIC requires that successive chip selects be no closer than 4 bus clocks (BSCK) together. If the condition is violated, the ST-NIC may glitch $\overline{ACK}$. CPUs that operate from pipelined instructions (i.e., 386) or have a cache (i.e., 486) can execute consecutive I/O cycles very quickly. The solution is to delay the execution of consecutive I/O cycles by either breaking the pipeline or forcing the CPU to access outside its cache.

Time between Chip Selects



TL/F/11157–63

14.0 Preliminary Electrical Characteristics

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (PD)	800 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C
ESD Rating (R_{ZAP} = 1.5k, C_{ZAP} = 120 pF)	1.5 kV
Clamp Diode Current	±20 mA

Note: *Absolute Maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

Note: *All specifications in this datasheet are valid only if the mandatory isolation is employed and all differential signals are taken to exist at the AUI or TPI side of the isolation.*

Preliminary DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage (Notes 1, 4)	I_{OH} = −20 μA I_{OH} = −2.0 mA	V_{CC} −0.1 3.5		V V
V_{OL}	Minimum Low Level Output Voltage (Notes 1, 4)	I_{OL} = 20 μA I_{OL} = 2.0 mA		0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage (Note 2)		2.0		V
V_{IH2}	Minimum High Level Input Voltage For RACK WACK (Note 2)		2.7		V
V_{IL}	Maximum Low Level Input Voltage (Note 2)			0.8	V
V_{IL2}	Maximum Low Level Input Voltage For RACK, WACK (Note 2)			0.6	V
V_{LOL}	Good Link Output Voltage	I_{OL} = 16 mA		0.4	V
I_{IN}	Input Current	V_I = V_{CC} or GND	−1.0	+1.0	μA
I_{OZ}	Minimum TRI-STATE Output Leakage Current (Note 5)	V_{OUT} = V_{CC} or GND	−10	+10	μA
I_{CC}	Average Supply Current (Note 3)	X1 = 20 MHz Clock I_{OUT} = 0 μA V_{IN} = V_{CC} or GND		140	mA

Note 1: These levels are tested dynamically using a limited amount of functional test patterns, please refer to AC test load.

Note 2: Limited functional test patterns are performed at these input levels. The majority of functional tests are performed at levels of 0V and 3V.

Note 3: This is measured with a 0.1 μF bypass capacitor between V_{CC} and GND.

Note 4: The low drive CMOS compatible V_{OH} and V_{OL} limits are not tested directly. Detailed device characterization validates that this specification can be guaranteed by testing the high drive TTL compatible V_{OL} and V_{OH} specification.

Note 5: RA0–RA3, $\overline{PRD}$, $\overline{WACK}$, BREQ and INT pins are used as outputs in test mode and as a result are tested as if they are TRI-STATE input/outputs. For these pins the input leakage specification is I_{OZ}.

14.0 Preliminary Electrical Characteristics (Continued)

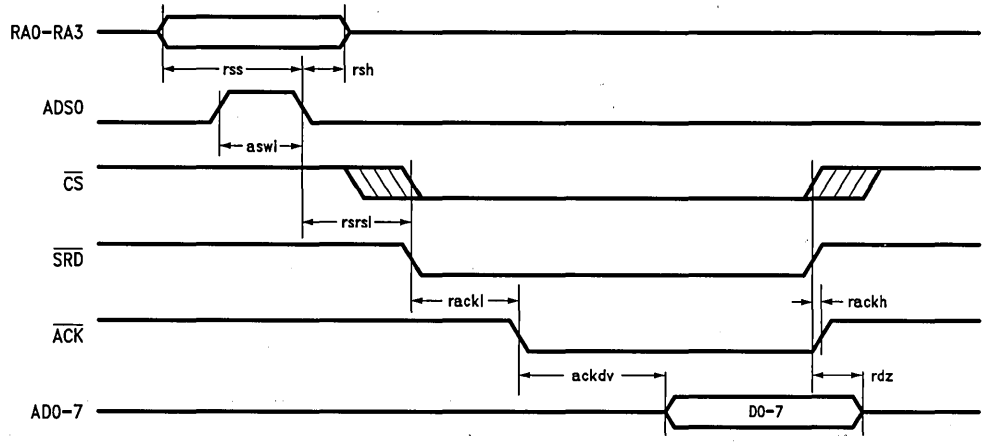
Preliminary DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified. (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
AUI INTERFACE PINS (TX±, RX±, and CD±)					
V_{OD}	Diff. Output Voltage (TX±)	78Ω Termination, and 270Ω from Each to GND	±550	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±) (Note 1)	78Ω Termination, and 270Ω from Each to GND	Typical: 40 mV		
V_U	Undershoot Voltage (TX±) (Note 1)	78Ω Termination, and 270Ω from Each to GND	Typical: 80 mV		
V_{DS}	Diff. Squelch Threshold (RX± and CD±) (Note 1)		−175	−300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± and CD±) (Note 1)		0	5.25	V
OSCILLATOR PINS (X1 AND X2)					
V_{IH}	X1 Input High Voltage	X1 is Connected to an Oscillator and GND/X2 is Grounded	2.0		V
V_{IL}	X1 Input Low Voltage	X1 is Connected to an Oscillator and GND/X2 is Grounded		0.8	V
I_{OSC}	X1 Input Current	GND/X2 is Grounded $V_{IN} = V_{CC}$ or GND		3	mA
TWISTED PAIR INTERFACE PINS (TXO±, TXOd±, and RXI±)					
R_{TOL}	TXOd±, TXO± Low Level Output Resistance	I_{OL} = 25 mA		15	Ω
R_{TOH}	TXOd±, TXO± High Level Output Resistance	I_{OH} = 25 mA		15	Ω
V_{SRON}	Receive Threshold Turn-On Voltage		±300	±585	mV
V_{SROFF}	Receive Threshold Turn-Off Voltage (Note 1)		±175	V_{SRON} − 100	mV
V_{DIFF}	Differential Mode Input Voltage Range (Note 1)	V_{CC} = 5.0V	−3.1	+3.1	V

Note 1: This parameter is guaranteed by design and is not tested.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary

Register Read (Latched Using ADS0)



TL/F/11157–33

Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	13		ns
aswi	Address Strobe Width In	15		ns
ackdv	Acknowledge Low to Data Valid		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 3)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns
rsrsl	Register Select to Slave Read Low, Latched RS0–3	10		ns

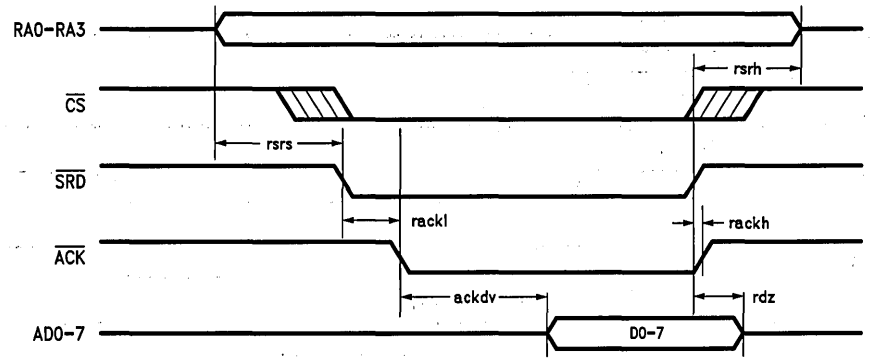
Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SRD}$ are low and the ST-NIC has synchronized to the register access. The ST-NIC will insert an integral number of Bus Clock cycles until it is synchronized. In Dual Bus systems additional cycles will be used for a local or remote DMA to complete. Wait states must be issued to the CPU until $\overline{ACK}$ is asserted low.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SRD}$. If $\overline{CS}$ is asserted after $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$. $\overline{CS}$ can be de-asserted concurrently with $\overline{SRD}$ or after $\overline{SRD}$ is de-asserted.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Register Read (Non-Latched, ADS0 = 1)



TL/F/11157-34

Symbol	Parameter	Min	Max	Units
rsrs	Register Select to Read Setup (Notes 1, 3)	10		ns
rsrh	Register Select Hold from Read	0		ns
ackdv	$\overline{ACK}$ Low to Valid Data		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 2)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Note 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns

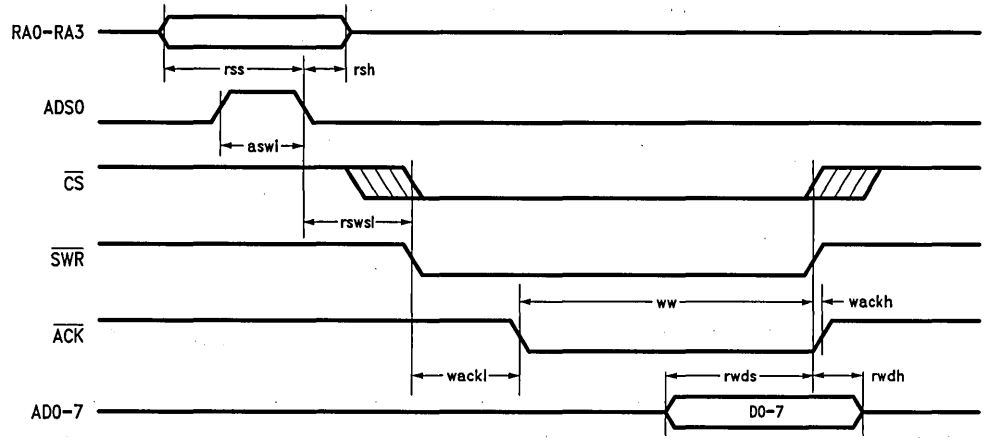
Note 1: rsrs includes flow-through time of latch.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

Note 3: $\overline{CS}$ may be asserted before of after RA0–3, and $\overline{SRD}$, since address decode begins when $\overline{ACK}$ is asserted. If $\overline{CS}$ is asserted after RA0–3, and $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Register Write (Latched Using ADS0)



TL/F/11157–35

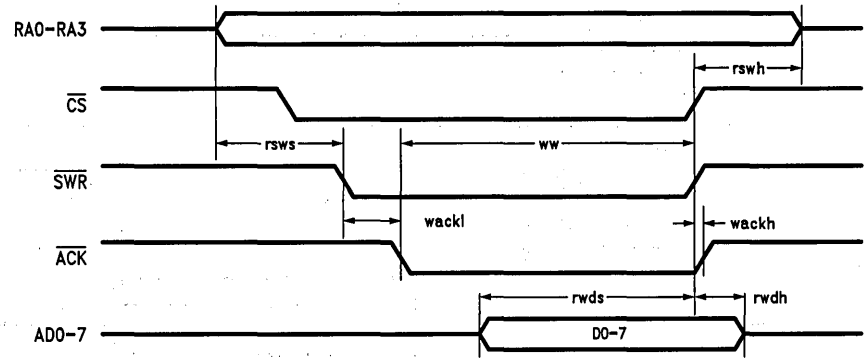
Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	17		ns
aswi	Address Strobe Width In	15		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
ww	Write Strobe Width from $\overline{ACK}$	50		ns
wackh	Write Strobe High to $\overline{ACK}$ High		30	ns
wackl	Write Low to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rswsl	Register Select to Write Strobe Low	10		ns

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the ST-NIC has synchronized to the register access. In Dual Bus Systems additional cycles will be used for a local DMA or Remote DMA to complete.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SWR}$. If $\overline{CS}$ is asserted after $\overline{SWR}$, wackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Register Write (Non-Latched, ADS0 = 1)



TL/F/11157–36

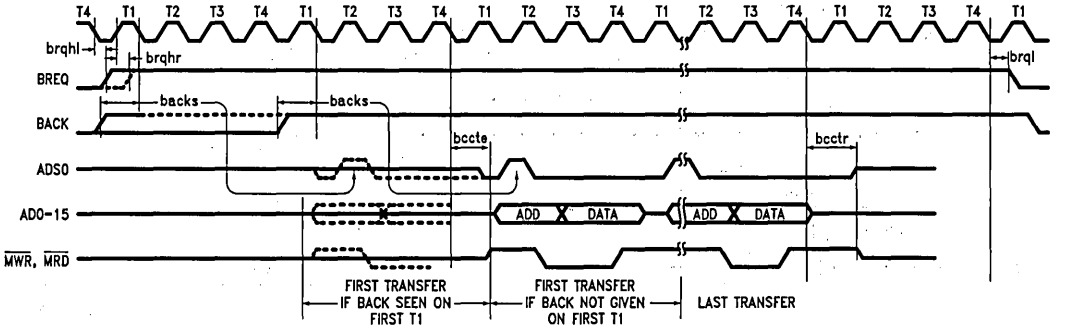
Symbol	Parameter	Min	Max	Units
rsws	Register Select to Write Setup (Note 1)	15		ns
rswh	Register Select Hold from Write	0		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
wackl	Write Low to $\overline{ACK}$ Low (Note 2)		n*bcyc + 30	ns
wackh	Write High to $\overline{ACK}$ High		30	ns
ww	Write Width from $\overline{ACK}$	50		ns

Note 1: Assumes ADS0 is high when RA0–3 changing.

Note 2: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the ST-NIC has synchronized to the register access. In Dual Bus systems additional cycles will be used for a local DMA or remote DMA to complete.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

DMA Control, Bus Arbitration



TL/F/11157–37

Symbol	Parameter	Min	Max	Units
brqhl	Bus Clock to Bus Request High for Local DMA		50	ns
brqhr	Bus Clock to Bus Request High for Remote DMA		45	ns
brql	Bus Request Low from Bus Clock		60	ns
backs	Acknowledge Setup to Bus Clock (Note 1)	2		ns
bccte	Bus Clock to Control Enable		60	ns
bcctr	Bus Clock to Control Release (Notes 2, 3)		70	ns

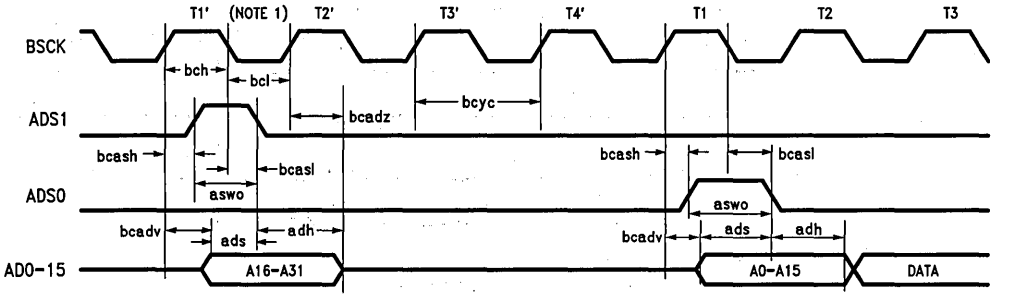
Note 1: BACK must be setup before T1 after BREQ is asserted. Missed setup will slip the beginning of the DMA by four bus clocks. The Bus Latency will influence the allowable FIFO threshold.

Note 2: During remote DMA transfers only, a single bus transfer is performed. During local DMA operations burst mode transfers are performed.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

DMA Address Generation



TL/F/11157–38

Symbol	Parameter	Min	Max	Units
bcyc	Bus Clock Cycle Time (Note 2)	50	125	ns
bch	Bus Clock High Time	20		ns
bcl	Bus Clock Low Time	20		ns
bcash	Bus Clock to Address Strobe High		34	ns
bcasl	Bus Clock to Address Strobe Low		44	ns
aswo	Address Strobe Width Out	bch		ns
bcadv	Bus Clock to Address Valid		45	ns
bcadz	Bus Clock to Address TRI-STATE (Note 3)	15	55	ns
ads	Address Setup to ADS0/1 Low	bch − 15		ns
adh	Address Hold from ADS0/1 Low	bcl − 5		ns

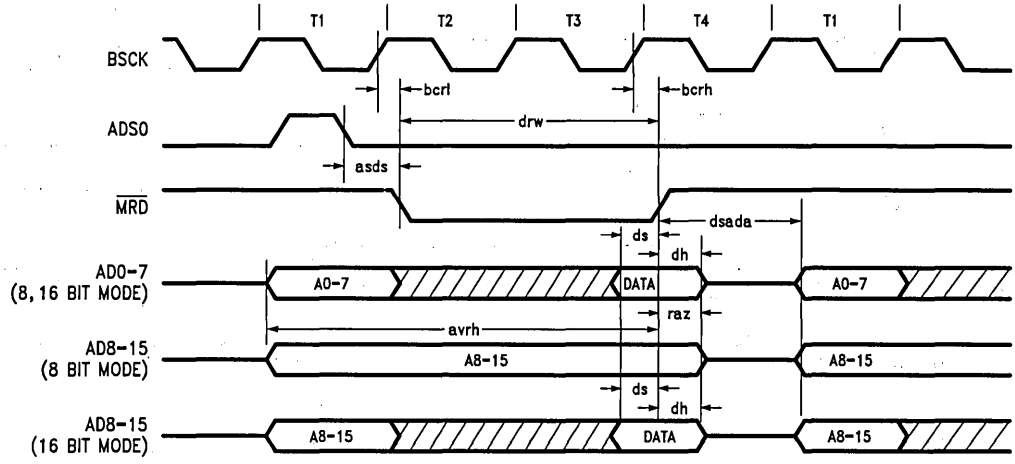
Note 1: Cycles T1', T1', T3' ≤ T4' are only issued for the first transfer in a burst 32-bit mode has been selected.

Note 2: The rate of bus clock must be high enough to support transfers to/from the FIFO at a rate greater than the serial network transfers from/to the FIFO.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

DMA Memory Read



TL/F/11157-39

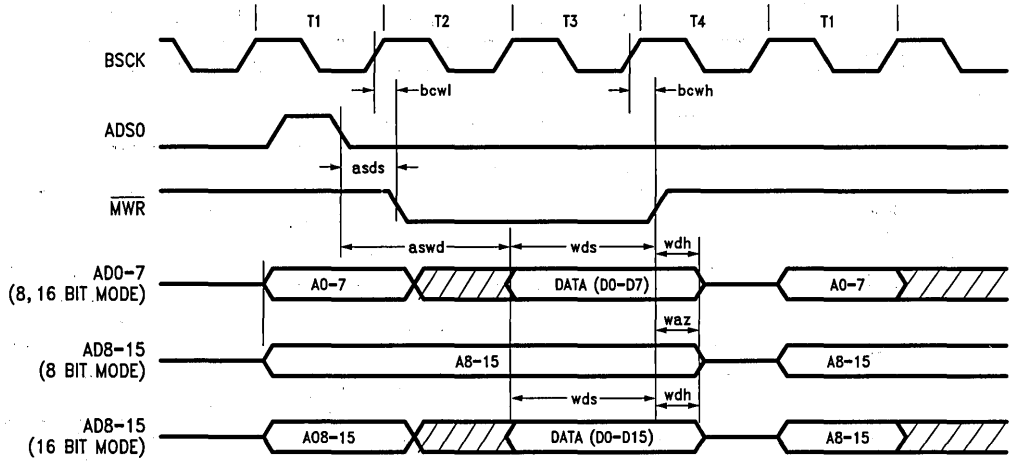
Symbol	Parameter	Min	Max	Units
bcrl	Bus Clock to Read Strobe Low		43	ns
bcrh	Bus Clock to Read Strobe High		40	ns
ds	Data Setup to Read Strobe High	22		ns
dh	Data Hold from Read Strobe High	0		ns
drw	DMA Read Strobe Width Out	2*bcyc − 15		ns
raz	Memory Read High to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
dsada	Data Strobe to Address Active	bcyc − 10		ns
avrh	Address Valid to Read Strobe High	3*bcyc − 18		ns

Note 1: During a burst A8–A15 are not TRI-STATE if byte wide transfers are selected. On the last transfer A8–A15 are TRI-STATE as shown above.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

DMA Memory Write



TL/F/11157-40

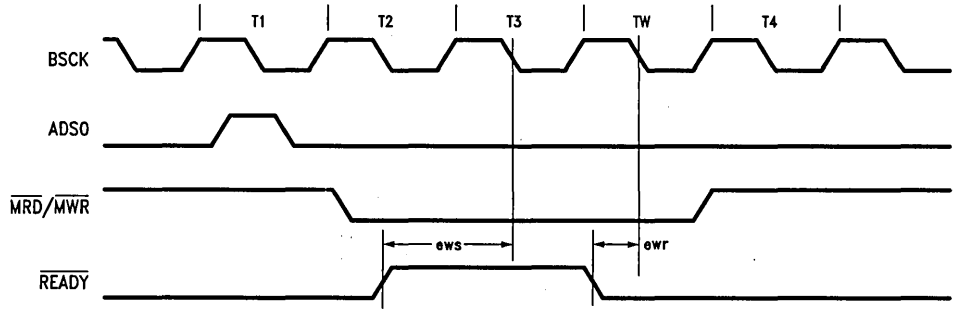
Symbol	Parameter	Min	Max	Units
bcwl	Bus Clock to Write Strobe Low		40	ns
bcwh	Bus Clock to Write Strobe High		40	ns
wds	Data Setup to $\overline{MWR}$ High	2*bcyc − 30		ns
wdh	Data Hold from $\overline{MWR}$ Low	bch + 7		ns
waz	Write Strobe to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
aswd	Address Strobe to Write Data Valid		bcl + 30	ns

Note 1: When using byte mode transfers A8–A15 are only TRI-STATE on the last transfer, waz timing is only valid for last transfer in a burst.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Wait State Insertion



TL/F/11157–41

Symbol	Parameter	Min	Max	Units
ews	External Wait Setup to T3 0Clock (Note 1)	10		ns
ewr	External Wait Release Time (Note 1)	15		ns

Note 1: The addition of wait states affects the count of deserialized bytes and is limited to a number of bus clock cycles depending on the bus clock and network rates. The allowable wait states are found in the table below. (Assumes 10 Mbit/sec data rate.)

BSCK (MHz)	# of Wait States	
	Byte Transfer	Word Transfer
8	0	1
10	0	1
12	1	2
14	1	2
16	1	2
18	2	3
20	2	4

Table assumes 10 MHz network clock.

The number of allowable wait states in byte mode can be calculated using:

$$\#W_{(\text{byte mode})} = \left(\frac{8\ tnw}{4.5\ tbsck} - 1\right)$$

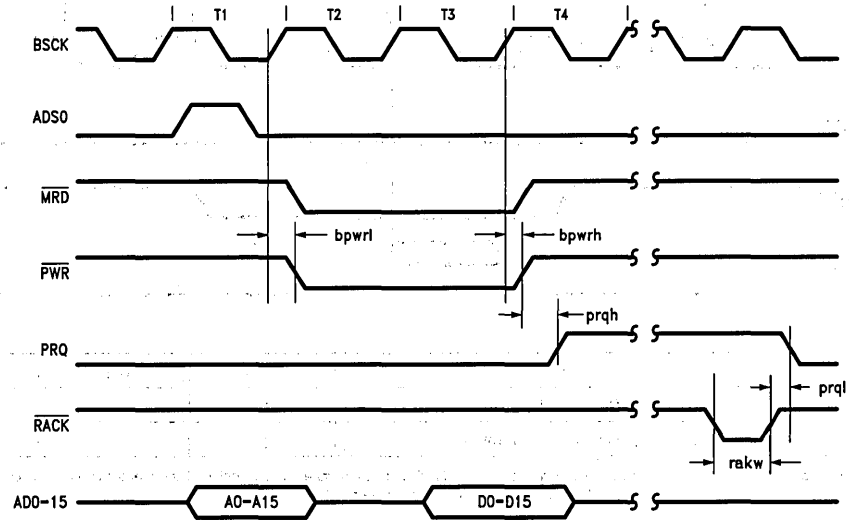
#W = Number of Wait States
tnw = Network Clock Period
tbsck = BSCK Period

The number of allowable wait states in word mode can be calculated using:

$$\#W_{(\text{word mode})} = \left(\frac{5\ tnw}{2\ tbsck} - 1\right)$$

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Remote DMA (Read, Send Command)



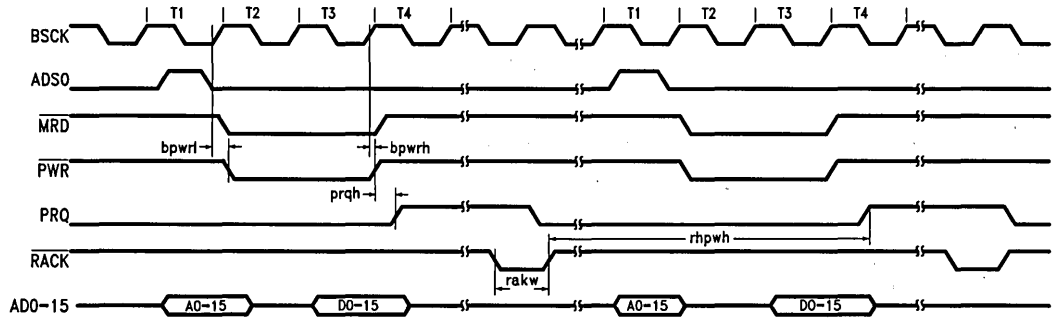
TL/F/11157–42

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		60	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Remote DMA (Read, Send Command) Recovery Time



TL/F/11157-43

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		60	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns
rhpwh	Read Acknowledge High to Next Port Write Cycle (Notes 2, 3, 4)	11		BSCK

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether or not a local DMA is pending.

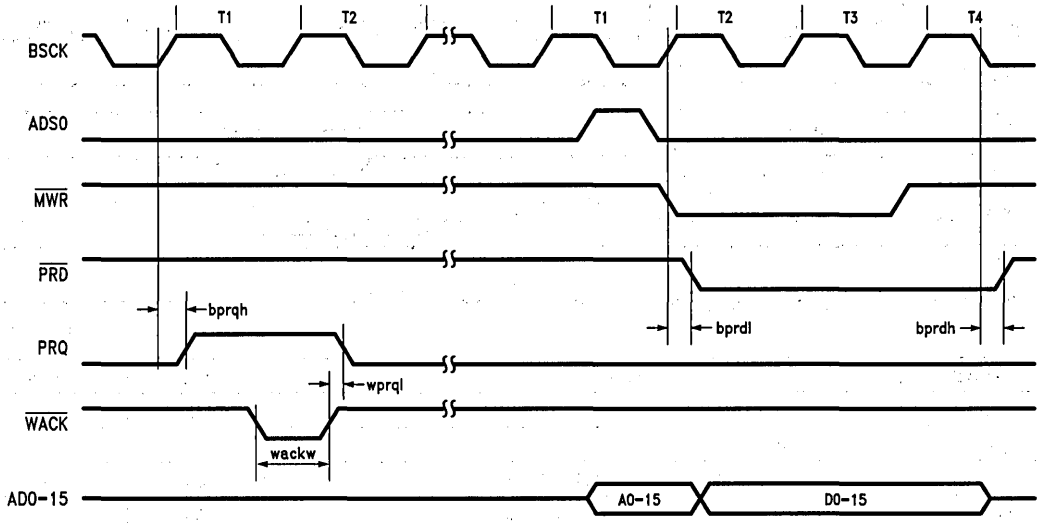
Note 2: This is not a measured value but guaranteed by design.

Note 3: $\overline{RACK}$ must be high for a minimum of 7 BSCK.

Note 4: Assumes no local DMA interleave, no $\overline{CS}$, and immediate BACK.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Remote DMA (Write Cycle)



TL/F/11157-44

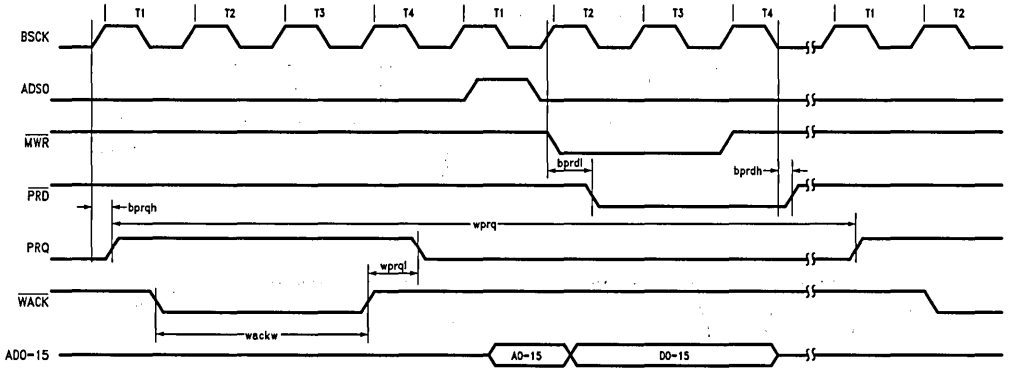
Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{WACK}$ to Port Request Low		52	ns
wackw	$\overline{WACK}$ Pulse Width	25		ns
bprdl	Bus Clock to Port Read Low (Note 2)		55	ns
bprdh	Bus Clock to Port Read High		40	ns

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

Remote DMA (Write Cycle) Recovery Time



TL/F/11157-45

Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{WACK}$ to Port Request Low		50	ns
wackw	$\overline{WACK}$ Pulse Width	25		ns
bprdl	Bus Clock to Port Read Low (Note 2)		55	ns
bprdh	Bus Clock to Port Read High		40	ns
wprq	Remote Write Port Request to Port Request Time (Notes 3, 4, 5)	12		BSCK

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

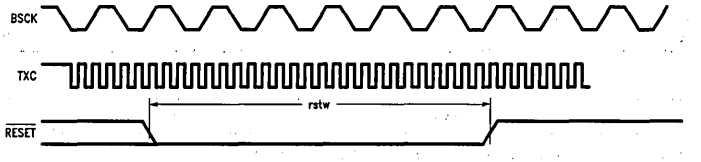
Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BSCK and whether a local DMA is pending.

Note 3: Assuming wackw < 1 BSCK, and no local DMA interleave, no $\overline{CS}$, immediate $\overline{BACK}$, and $\overline{WACK}$ goes high before T4.

Note 4: $\overline{WACK}$ must be high for a minimum of 7 BSCK.

Note 5: This is not a measured value but guaranteed by design.

Reset Timing



TL/F/11157-64

Symbol	Parameter	Min	Max	Units
rstw	Reset Pulse Width (Note 1)	8		BSCK Cycles or TXC Cycles (Note 2)

Note 1: The RESET pulse requires that BSCK and TXC be stable. On power up, $\overline{RESET}$ should not be raised until BSCK and TXC have become stable. Several registers are affected by $\overline{RESET}$. Consult the register descriptions for details.

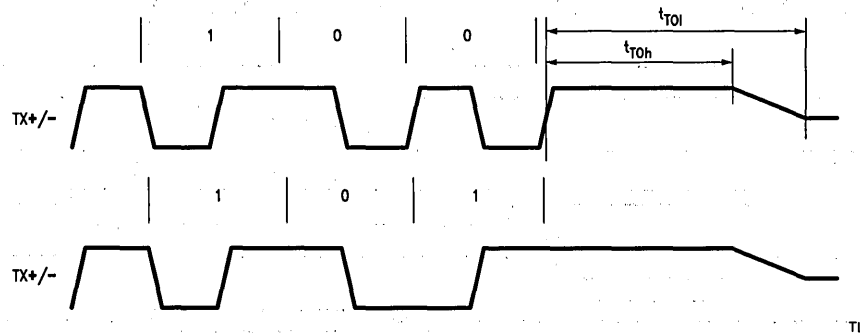
Note 2: The slower of BSCK or TXC clocks will determine the minimum time for the $\overline{RESET}$ signal to be low.

If BSCK < TXC then $\overline{RESET}$ = 8 × BSCK

If TXC < BSCK then $\overline{RESET}$ = 8 × TXC

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

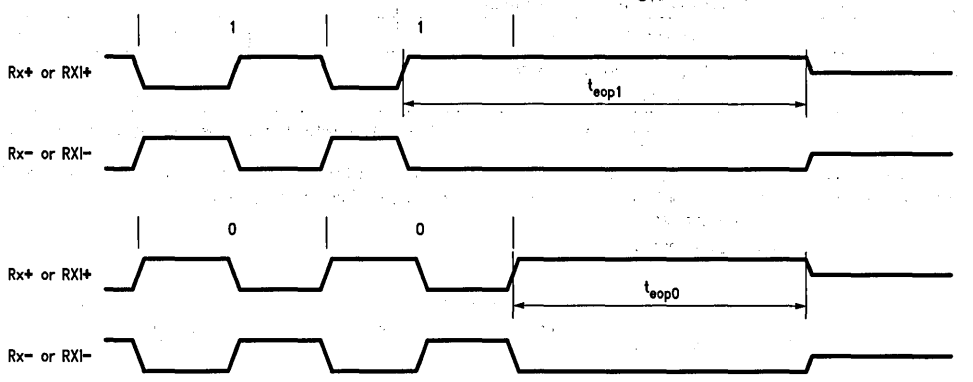
AUI Transmit Timing (End of Packet)



TL/F/11157–46

Symbol	Parameter	Min	Max	Units
t_{TOh}	Transmit Output High before Idle (Half Step)	200		ns
t_{TOI}	Transmit Output Idle Time (Half Step)		8000	ns

AUI/TPI Receive End of Packet Timing



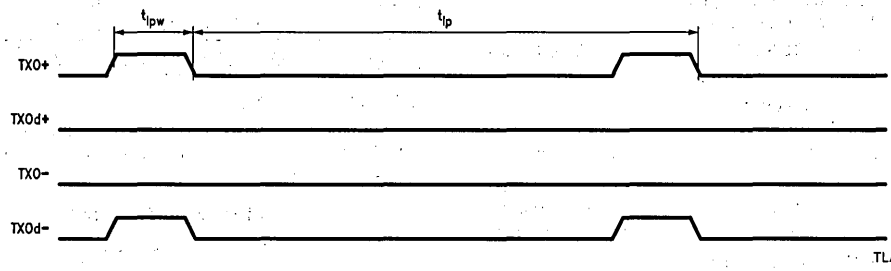
TL/F/11157–47

Symbol	Parameter	Min	Max	Units
teop1	Receive End of Packet Hold Time after Logic "1" (Note 1)	225		ns
teop0	Receive End of Packet Hold Time after Logic "0" (Note 1)	225		ns

Note 1: This parameter is guaranteed by design and is not tested.

15.0 Switching Characteristics AC Specs DP83902 **Note:** All Timing is Preliminary (Continued)

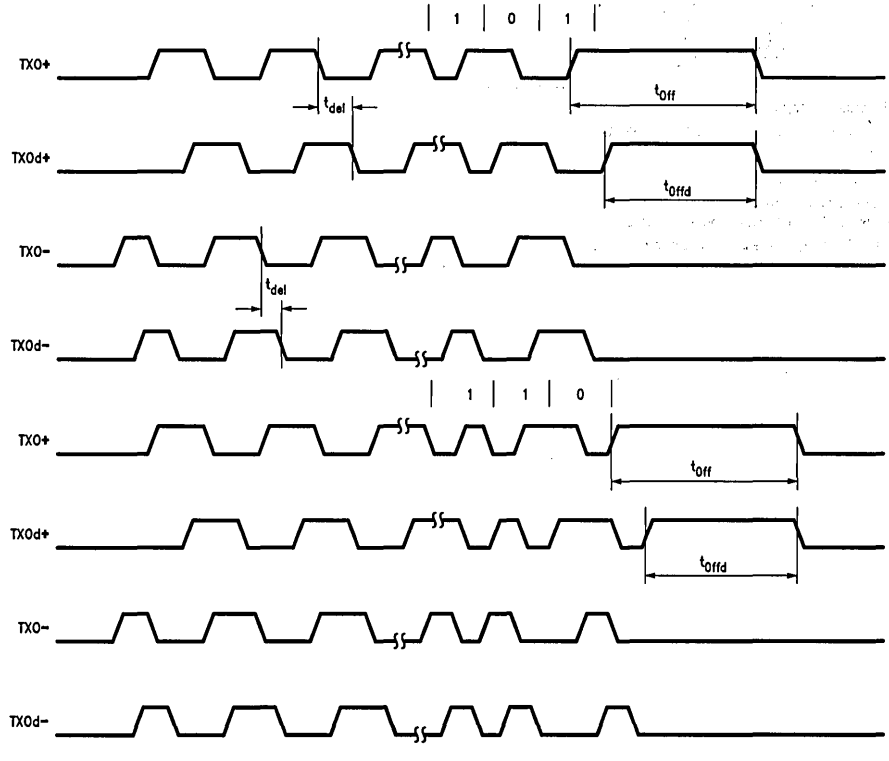
Link Pulse Timing



TL/F/11157–48

Symbol	Parameter	Min	Max	Units
tip	Time between Link Output Pulses	8	24	ms
tipw	Link Integrity Output Pulse Width	80	130	ns

TPI Transmit and End of Packet Timing



TL/F/11157–49

Symbol	Parameter	Min	Max	Units
tdel	Pre-Emphasis Output Delay (TXO ± to TXOd ±) (Note 1)	46	54	ns
toff	Transmit Hold Time at End of Packet (TXO ±) (Note 1)	250		ns
toffd	Transmit Hold Time at End of Packet (TXOd ±) (Note 1)	200		ns

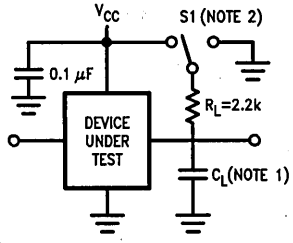
Note 1: This parameter is guaranteed by design and is not tested.

16.0 AC Timing Test Conditions

All specifications are valid only if the mandatory isolation is employed and all differential signals are taken to be at the AUI side of the pulse transformer.

Input Pulse Levels (TTL/CMOS)	GND to 3.0V
Input Rise and Fall Times (TTL/CMOS)	5 ns
Input and Output Reference Levels (TTL/CMOS)	1.3V
Input Pulse Levels (Diff.)	−350 mV to −1315 mV
Input and Output Reference Levels (Diff.)	50% Point of the Differential
TRI-STATE Reference Levels	Float (ΔV) ±0.5V

Output Load (See Figure Below)



TL/F/11157–50

Note 1: 50 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.

S1 = V_{CC} for V_{OL} test.

S1 = GND for V_{OH} test.

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active high to High Impedance measurements.

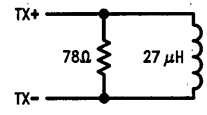
Pin Capacitance T_A = 25°C, f = 1 MHz

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	7	pF
C_{OUT}	Output Capacitance	7	pF

DERATING FACTOR

Output timing is measured with a purely capacitive load of 50 pF. The following correction factor can be used for other loads: $C_L \geq 50$ pF + 0.3 ns/pF.

AUI Transmit Test Load



TL/F/11157–51

Note: In the above diagram, the TX+ and TX− signals are taken from the AUI side of the isolation (pulse transformer). The pulse transformer used for all testing is the Pulse Engineering PE64103.

National Semiconductor

DP83901A
Serial Network Interface Controller (SNIC)

General Description

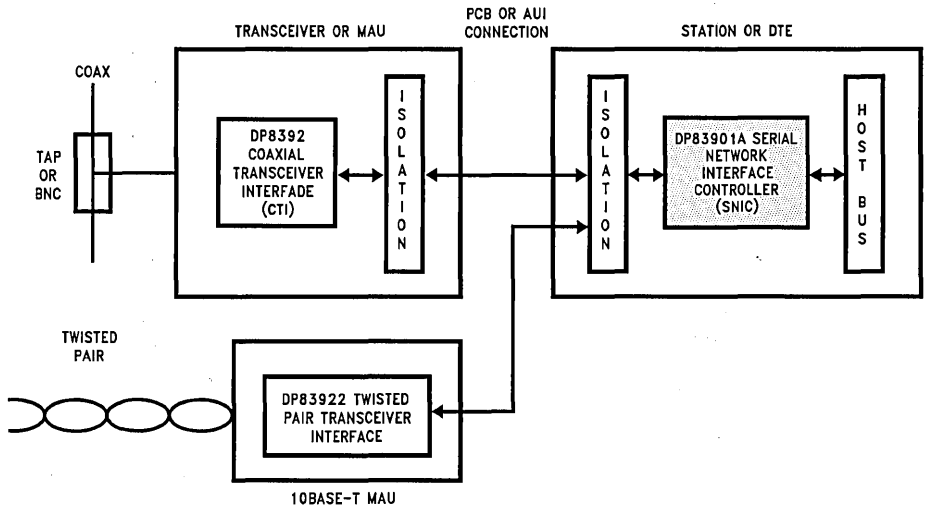
The DP83901A Serial Network Interface Controller (SNIC) is a microCMOS VLSI device designed for easy implementation of CSMA/CD local area networks. These include Ethernet (10BASE5), Thin Ethernet (10BASE2) and Twisted-pair Ethernet (10BASE-T). The overall SNIC solution provides the Media Access Control (MAC) and Encode-Decode (ENDEC) functions in accordance with the IEEE 802.3 standard.

The integrated ENDEC module allows Manchester encoding and decoding via a differential transceiver and phase lock loop at 10 Mbit/sec. Also included is a collision detect translator and diagnostic loopback capability. (Continued)

Features

- Compatible with IEEE 802.3, 10BASE5, 10BASE2, 10BASE-T
- Dual 16-byte DMA channels
- 16-byte internal FIFO
- Network statistics storage
- Supports physical, multicast and broadcast address filtering
- 10 Mbit/sec Manchester encoding and decoding plus clock recovery
- No external precision components required
- Efficient buffer management implementation
- Transmitter can be selected for half or full step mode
- Integrated squelch on receive and collision pairs
- 3 levels of loopback supported
- Utilizes independent system and network clocks
- Lock Time 5 bits typical
- Decodes Manchester data with up to ±18 ns jitter

1.0 System Diagram



TL/F/10469–1

General Description (Continued)

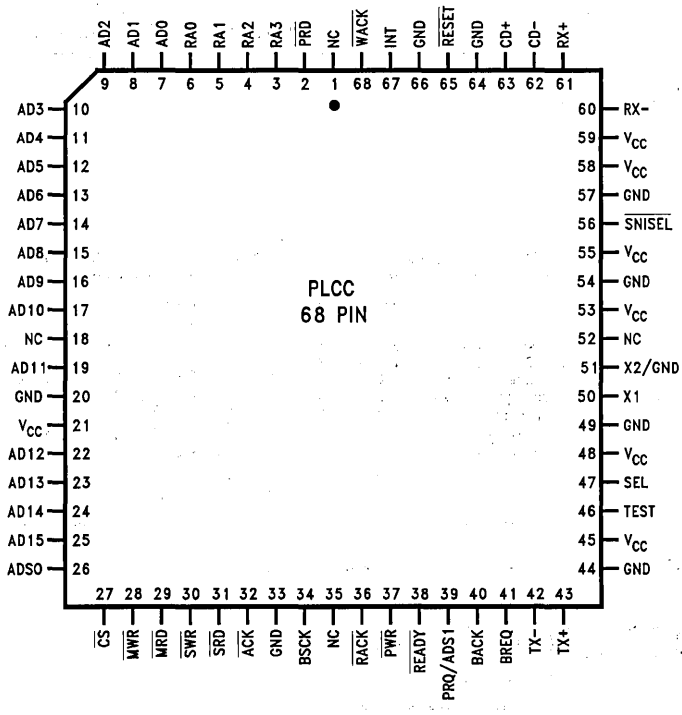
The MAC function (NIC) provides simple and efficient packet transmission and reception control by means of unique dual DMA channels and an internal FIFO. Bus arbitration and memory control logic are integrated to reduce board cost and area overheads.

SNIC used in conjunction with the DP8392 Coaxial Transceiver Interface (CTI) provides a comprehensive 2 chip solution for IEEE 802.3 networks and is designed for easy interface to the latest 10BASE-T transceivers.

Due to the inherent constraints of CMOS processing, isolation is required at the differential signal interfaces for 10BASE5 and 10BASE2 applications. Capacitive or inductive isolation may be used.

Table Of Contents

Connection Diagram



TL/F/10469–2

Top View

Order Number DP83901AV
See NS Package Number V68A

Pin Description

Pin No	Pin Name	I/O	Description
BUS INTERFACE PINS			
2	$\overline{PRD}$	O	**PORT READ:** Enables data from external latch on to local bus during a memory write cycle to local memory (remote write operation). This allows asynchronous transfer of data from the system memory to local memory.
3–6	RA0–RA3	I	**REGISTER ADDRESS:** These four pins are used to select a register to be read or written. The state of these inputs is ignored when the NIC is not in slave mode ($\overline{CS}$ high).
7–17, 19, 22–25	AD0–AD15	I/O, Z	**MULTIPLEXED ADDRESS/DATA BUS:** • Register Access, with DMA inactive, $\overline{CS}$ low and $\overline{ACK}$ returned from SNIC, pins AD0–AD7 are used to read and write register data. AD8–AD15 float during I/O transfers, $\overline{SRD}$, $\overline{SWR}$ pins are used to select direction of transfer. • Bus Master with BACK input asserted. During t1 of memory cycle AD0–AD15 contain address. During t2, t3, t4 AD0–AD15 contain data (word transfer mode). During t2, t3, t4 AD0–AD7 contain data, AD8–AD15 contain address (byte transfer mode). Direction of transfer is indicated by SNIC on $\overline{MWR}$, $\overline{MRD}$ lines.
26	ADS0	I/O, Z	**ADDRESS STROBE 0:** • Input: with DMA inactive and $\overline{CS}$ low, latches RA0–RA3 inputs on falling edge. If high, data present on RA0–RA3 will flow through latch. • Output: When Bus Master, latches address bits (A0–A15) to external memory during DMA transfers.
27	$\overline{CS}$	O	**CHIP SELECT:** Chip Select places controller in slave mode for μP access to internal registers. Must be valid through data portion of bus cycle. RA0–RA3 are used to select the internal register. $\overline{SWR}$ and $\overline{SRD}$ select direction of data transfer.
28	$\overline{MWR}$	O, Z	**MASTER WRITE STROBE:** (Strobe for DMA transfers) Active low during write cycles (t2, t3, tw) to buffer memory. Rising edge coincides with the presence of valid output data. TRI-STATE® until BACK asserted.
29	$\overline{MRD}$	O, Z	**MASTER READ STROBE:** (Strobe for DMA transfers) Active during read cycles (t2, t3, tw) to buffer memory. Input data must be valid on rising edge of MRD. TRI-STATE until BACK asserted.
30	$\overline{SWR}$	I	**SLAVE WRITE STROBE:** Strobe from CPU to write an internal register selected by RA0–RA3. Data is latched into the SNIC on the rising edge of this input.
31	$\overline{SRD}$	I	**SLAVE READ STROBE:** Strobe from CPU to read an internal register selected by RA0–RA3. The register data is output when $\overline{SRD}$ goes low.
32	$\overline{ACK}$	O	**ACKNOWLEDGE:** Active low when SNIC grants access to CPU. Used to insert WAIT states to CPU until SNIC is synchronized for a register read or write operation.
34	BSCK	I	**BUS CLOCK:** This clock is used to establish the period of the DMA memory cycle. Four clock cycles (t1, t2, t3, t4) are used per DMA cycle. DMA transfers can be extended by one BSCK increments using the READY input.
36	$\overline{RACK}$	I	**READ ACKNOWLEDGE:** Indicates that the system DMA or host CPU has read the data placed in the external latch by the SNIC. The SNIC will begin a read cycle to update the latch.
37	$\overline{PWR}$	O	**PORT WRITE:** Strobe used to latch data from the SNIC into external latch for transfer to host memory during Remote Read transfers. The rising edge of $\overline{PWR}$ coincides with the presence of valid data on the local bus.
38	$\overline{READY}$	I	**READY:** This pin is set high to insert wait states during a DMA transfer. The SNIC will sample this signal at t3 during DMA transfers.

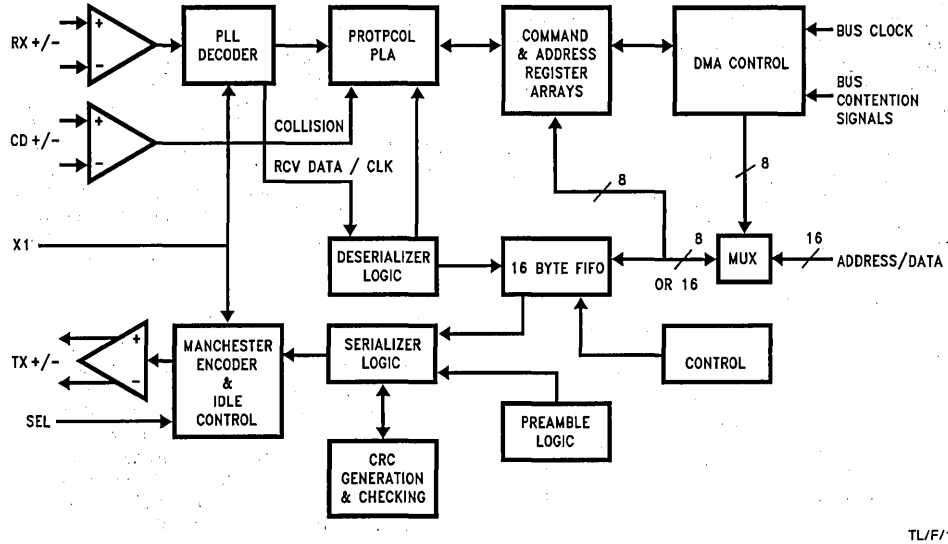
Pin Description (Continued)

Pin No	Pin Name	I/O	Description
BUS INTERFACE PINS (Continued)			
39	PRQ/ADS1	O, Z	**PORT REQUEST/ADDRESS STROBE 1** • 32-BIT MODE: If LAS is set in the Data Configuration Register, this line is programmed as $\overline{ADS1}$. It is used to strobe addresses A16–A31 into external latches. (A16–A31 are the fixed addresses stored in RSAR0, RSAR1). $\overline{ADS1}$ will remain at TRI-STATE until BACK is received. • 16-BIT MODE: If LAS is not set in the Data Configuration Register, this line is programmed as PRQ and is used for Remote DMA Transfers. The SNIC initiates a single remote DMA read or write operation by asserting this pin. In this mode PRQ will be a standard logic output. **Note:** This line will power up as TRI-STATE until the Data Configuration Register is programmed.
40	BACK	I	**BUS ACKNOWLEDGE:** Bus Acknowledge is an active high signal indicating that the CPU has granted the bus to the SNIC. If immediate bus access is desired, BREQ should be tied to BACK. **Tying BACK to V_{CC} will result in a deadlock.**
41	BREQ	O	**BUS REQUEST:** Bus Request is an active high signal used to request the bus for DMA transfers. This signal is automatically generated when the FIFO needs servicing.
65	$\overline{RESET}$	I	**RESET:** Reset is active low and places the SNIC in a reset immediately, no packets are transmitted or received by the SNIC until STA bit is set. Affects Command Register, Interrupt Mask Register, Data Configuration Register and Transmit Configuration Register. The SNIC will execute reset within 10 BSCK cycles and TXC cycles.
67	INT	O	**INTERRUPT:** Indicates that the SNIC requires CPU attention after reception transmission or completion of DMA transfers. The interrupt is cleared by writing to the ISR (Interrupt Service Register). All interrupts are maskable.
68	$\overline{WACK}$	I	**WRITE ACKNOWLEDGE:** Issued from system to SNIC to indicate that data has been written to the external latch. The SNIC will begin a write cycle to place the data in local memory.
NETWORK INTERFACE PINS			
42, 43	TX− TX+	O	**TRANSMIT OUTPUT:** Differential driver which sends the encoded data to the transceiver. The outputs are source followers which require 270Ω pulldown resistors.
46	TEST	I	**FACTORY TEST INPUT:** Used to check the chip's internal functions. Tied low during normal operation.
47	SEL	I	**MODE SELECT:** When high, Transmit+ and Transmit− are the same voltage in the idle state. When low, Transmit+ is positive with respect to Transmit− in the idle state, at the transformer's primary.
50	X1	I	**EXTERNAL OSCILLATOR INPUT**
51	GND/X2	O	**GROUND/X2:** This in should normally be connected to ground. It is possible to use a crystal oscillator using X1 and GND/X2 if certain precautions are taken. Contact National Semiconductor for more information.
56	$\overline{SNISEL}$	I	**FACTORY TEST INPUT:** For normal operation tied to V_{CC}. When low enables the ENDEC module to be tested independently of the SNIC module.
60, 61	RX− RX+	I	**RECEIVE INPUT:** Differential receive input pair from the transceiver.
62, 63	CD− CD+	I	**COLLISION INPUT:** Differential collision pair input from the transceiver.

Pin Description (Continued)

Pin No	Pin Name	I/O	Description
POWER SUPPLY PINS			
21, 48, 53, 55	V_{CC}		**DIGITAL POSITIVE 5V SUPPLY PINS:**
20, 33, 49 54, 66	GND		**DIGITAL NEGATIVE (GROUND) SUPPLY PINS:** It is suggested that a decoupling capacitor be connected between the V_{CC} and GND pins.
59	V_{CC}		**AUI RECEIVE 5V SUPPLY:** Power pin supplies 5V to the AUI receiver.
64	GND		**AUI RECEIVE GROUND:** Ground pin for AUI receiver.
45	V_{CC}		**AUI TRANSMIT 5V SUPPLY:** Power pin supplies 5V to the AUI transmitter.
44	GND		**AUI TRANSMIT GROUND:** Ground pin for AUI transmitter
58	V_{CC}		**VCO 5V SUPPLY:** Care should be taken to reduce noise on this pin as it supplies 5V to the ENDEC's Phase Lock Loop.
57	GND		**VCO GROUND PIN:** Care should be taken to reduce noise on this pin as it is the ground to the ENDEC's Phase Lock Loop.
NO CONNECTION			
1, 18, 35, 52	NC		**NO CONNECTION:** Do not connect to these pins.

3.0 Block Diagram



TL/F/10469-3

FIGURE 1

4.0 Functional Description (Refer to *Figure 1*)

ENCODER/DECODER (ENDEC) MODULE

The ENDEC consists of four main logical blocks:

a) The Manchester encoder accepts NRZ data from the controller, encodes the data to Manchester, and transmits it differentially to the transceiver, through the differential transmit driver.

b) The Manchester decoder receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends it to the controller.

c) The collision translator indicates to the controller the presence of a valid 10 MHz collision signal to the PLL.

MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The differential transmit pair, on the secondary of the employed transformer, drives up to 50 meters of twisted pair AUI cable. These outputs are source followers which require two 270Ω pull-down resistors to ground.

The DP83901A allows both half-step and full-step to be compatible with Ethernet and IEEE 802.3. With the SEL pin low (for Ethernet I). Transmit+ is positive with respect to Transmit− during idle; with SEL high (for IEEE 802.3), Transmit+ and Transmit− are equal in the idle state. This provides zero differential voltage to operate with transformer coupled loads.

MANCHESTER DECODER

The decoder consists of a differential receiver and a PLL to separate a Manchester decoded data stream into internal clock signals and data. The differential input must be externally terminated with two 39Ω resistors connected in series if the standard 78Ω transceiver drop cable is used, in thin Ethernet applications, these resistors are optional. To prevent noise from falsely triggering the decoder, a squelch circuit at the input rejects signals with levels less than −175 mV. Signals more negative than −300 mV and a duration greater than 30 ns are decoded. Data becomes valid typically within 5 bit times. The DP83901A may tolerate bit jitter up to 18 ns in the received data. The decoder detects the end of a frame when no more mid-bit transitions are detected.

COLLISION TRANSLATOR

When the Ethernet transceiver (DP8392 CTI) detects a collision, it generates a 10 MHz signal to the differential collision inputs (CD±) of the DP83901A. When these inputs are detected active, the DP83901A uses this signal to back off its current transmission and reschedule another one.

The collision differential inputs are terminated the same way as the differential receive inputs. The squelch circuitry is also similar, rejecting pulses with levels less than −175 mV.

NIC (Media Access Control) MODULE

RECEIVE DESERIALIZER

The Receive Deserializer is activated when the input signal Carrier Sense is asserted to allow incoming bits to be shifted into the shift register by the receive clock. The serial receive data is also routed to the CRC generator/checker. The Receive Deserializer includes a synch detector which detects the SFD (Start of Frame Delimiter) to establish where byte boundaries within the serial bit stream are located. After every eight receive clocks, the byte wide data is transferred to the 16-byte FIFO and the Receive Byte Count is incremented. The first six bytes after the SFD are checked for valid comparison by the Address Recognition Logic. If the Address Recognition Logic does not recognize the packet, the FIFO is cleared.

CRC GENERATOR/CHECKER

During transmission, the CRC logic generates a local CRC field for the transmitted bit sequence. The CRC encodes all fields after the SFD. The CRC is shifted out MSB first following the last transmit byte. During reception the CRC logic generates a CRC field from the incoming packet. This local CRC is serially compared to the incoming CRC appended to the end of the packet by the transmitting node. If the local and received CRC match, a specific pattern will be generated and decoded to indicate no data errors. Transmission errors result in different pattern and are detected, resulting in rejection of a packet (if so programmed).

TRANSMIT SERIALIZER

The Transmit Serializer reads parallel data from the FIFO and serializes it for transmission. The serializer is clocked by the transmit clock generated internally. The serial data is also shifted into the CRC generator/checker. At the beginning of each transmission, the Preamble and Synch Generator append 62 bits of 1,0 preamble and a 1,1 synch pattern. After the last data byte of the packet has been serialized the 32-bit FCS field is shifted directly out of the CRC generator. In the event of a collision the Preamble and Synch generator is used to generate a 32-bit JAM pattern of all 1's.

ADDRESS RECOGNITION LOGIC

The address recognition logic compares the Destination Address Field (first 6 bytes of the received packet) to the Physical address registers stored in the Address Register Array. If any one of the six bytes does not match the pre-programmed physical address, the Protocol Control Logic rejects the packet. All multicast destination addresses are filtered using a hashing technique. (See register description.) If the multicast address indexes a bit that has been set in the filter bit array of the Multicast Address Register Array the packet is accepted, otherwise it is rejected by the Proto-

4.0 Functional Description (Continued)

col Control Logic. Each destination address is also checked for all 1's which is the reserved broadcast address.

FIFO AND BUS OPERATIONS

Overview

To accommodate the different rates at which data comes from (or goes to) the network and goes to (or comes from) the system memory, the SNIC contains a 16-byte FIFO for buffering data between the media. The FIFO threshold is programmable, allowing filling (or emptying) the FIFO at different rates. When the FIFO has filled to its programmed threshold, the local DMA channel transfers these bytes (or words) into local memory. It is crucial that the local DMA is given access to the bus within a minimum bus latency time; otherwise a FIFO underrun (or overrun) occurs.

FIFO underruns or overruns are caused by two conditions: (1) the bus latency is so long that the FIFO has filled (or emptied) from the network before the local DMA has serviced the FIFO and (2) the bus latency has slowed the throughput of the local DMA to point where it is slower than the network data rate (10 Mbit/sec). This second condition is also dependent upon DMA clock and word width (byte wide or word wide). The worst case condition ultimately limits the overall bus latency which the SNIC can tolerate.

Beginning of Receive

At the beginning or reception, the SNIC stores entire Address field of each incoming packet in the FIFO to determine whether the packet matches its Physical Address Registers or maps to one of its Multicast Registers. This causes the FIFO to accumulate 8 bytes. Furthermore, there are some synchronization delays in the DMA PLA. Thus, the actual time that BREQ is asserted from the time the Start of Frame Delimiter (SFD) is detected is 7.8 μs. This operation affects the bus latencies at 2 and 4-byte thresholds during the first receive BREQ since the FIFO must be filled to 8 bytes (or 4 words) before issuing a BREQ.

End of Receive

When the end of a packet is detected by the ENDEC module, the SNIC enters its end of packet processing sequence, emptying its FIFO and writing the status information at the beginning of the packet. The SNIC holds onto the bus for the entire sequence. The longest time BREQ may be extended occurs when a packet ends just as the SNIC performs its last FIFO burst. The SNIC, in this case, performs a programmed burst transfer followed by flushing the remaining bytes in the FIFO, and completed by writing the header information to memory. The following steps occur during this sequence.

1. SNIC issues BREQ because the FIFO threshold has been reached.
2. During the burst, packet ends, resulting in BREQ extended.
3. SNIC flushes remaining bytes from FIFO.
4. SNIC performs internal processing to prepare for writing the header.
5. SNIC writes 4-byte (2-word) header.
6. SNIC de-asserts BREQ.

FIFO Threshold Detection

To assure that no overwriting of data in the FIFO, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO, effectively shortening the FIFO to 13 bytes. The FIFO logic also operates differently in Byte Mode and in Word Mode. In Byte Mode, a threshold is indicated when the n+1 byte has entered the FIFO; thus, with an 8-byte threshold, the SNIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n+2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to 8-byte threshold), BREQ is issued when the 10th byte has entered the FIFO.

Beginning of Transmit

Before transmitting, the SNIC performs a prefetch from memory to load the FIFO. The number of bytes prefetched is the programmed FIFO threshold. The next BREQ is not issued until after the SNIC actually begins transmitting data, i.e., after SFD.

Reading the FIFO

During normal operation, the FIFO must not be read. The SNIC will not issue an ACKnowledge back to the CPU if the FIFO is read. The FIFO should only be read during loopback diagnostics.

PROTOCOL PLA

The protocol PLA is responsible for implementing the IEEE 802.3 protocol, including collision recovery with random backoff. The Protocol PLA also formats packets during transmission and strips preamble and synch during reception.

DMA AND BUFFER CONTROL LOGIC

The DMA and Buffer Control Logic is used to control two 16-bit DMA channels. During reception, the local DMA stores packets in a receive buffer ring, located in buffer memory. During transmission the Local DMA uses programmed pointer and length registers to transfer a packet from local buffer memory to the FIFO. A second DMA channel is used as a slave DMA to transfer data between the local buffer memory and the host system. The Local DMA and Remote DMA are internally arbitrated, with the Local DMA channel having highest priority. Both DMA channels use a common external bus clock to generate all required bus timing. External arbitration is performed with a standard bus request, bus acknowledge handshake protocol.

5.0 Transmit/Receive Packet Encapsulation/Decapsulation

A standard IEEE 802.3 packet consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data, and Frame Check Sequence (FCS). The typical format is shown in *Figure 2.* The packets are Manchester encoded and decoded by the ENDEC module and transferred serially to the NIC module using NRZ data with a clock. All fields are of fixed length except for the data field. The SNIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the ENDEC to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC module. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern which consists of two consecutive 1's. The SNIC does not treat the SFD pattern as a byte, it detects only the two bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the SNIC: physical, multicast and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of "0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the SNIC to accept the packet. Multicast addresses begin with an MSB of "1". The SNIC filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a 6-bit value. This 6-bit value indexes a 64-bit array that filters the value. If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

SOURCE ADDRESS

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

LENGTH FIELD

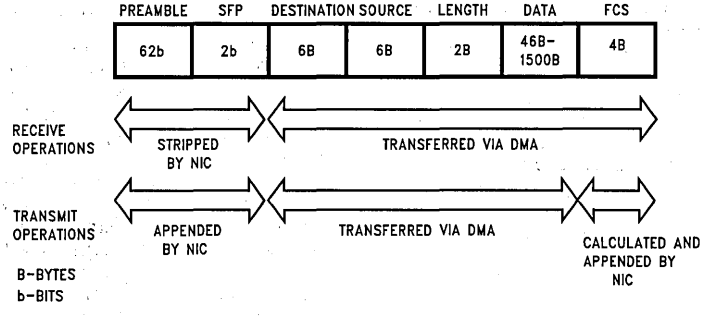
The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the SNIC.

DATA FIELD

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. **The SNIC does not strip or append pad bytes for short packets, or check for oversize packets.**

FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected. The AUTODIN II ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$) polynomial is used for the CRC calculations.



TL/F/10469-5

FIGURE 2

6.0 Direct Memory Access Control (DMA)

The DMA capabilities of the SNIC greatly simplify the use of the DP83901A in typical configurations. The local DMA channel transfers data between the FIFO and memory. On transmission, the packet is DMA'd from memory to the FIFO in bursts. Should a collision occur (up to 15 times), the packet is retransmitted with no processor intervention. On reception, packets are DMAed from the FIFO to the receive buffer ring (as explained below).

A remote DMA channel is also provided on the SNIC to accomplish transfers between a buffer memory and system memory. The two DMA channels can alternatively be combined to form a single 32-bit address with 8- or 16-bit data.

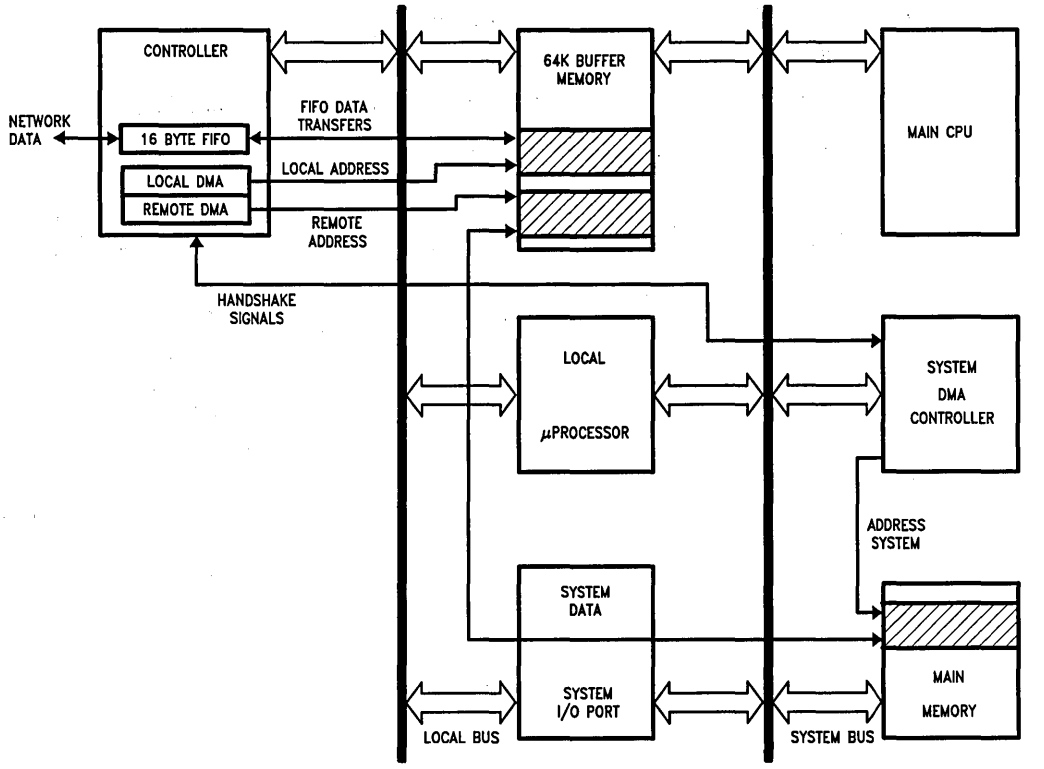
DUAL DMA CONFIGURATION

An example configuration using both the local and remote DMA channels is shown below. Network activity is isolated on a local bus, where the SNIC's local DMA channel performs burst transfers between the buffer memory and the SNIC's FIFO. The Remote DMA transfers data between the buffer memory and the host memory via a bidirectional I/O port. The Remote DMA provides local addressing capability and is used as a slave DMA by the host. Host side addressing must be provided by a host DMA or the CPU. The SNIC allows Local and Remote DMA operations to be interleaved.

SINGLE CHANNEL DMA OPERATION

If desirable, the two DMA channels can be combined to provide a 32-bit DMA address. The upper 16 bits of the 32-bit address are static and are used to point to a 64 kbyte (or 32k word) page of memory where packets are to be received and transmitted.

Dual Bus System



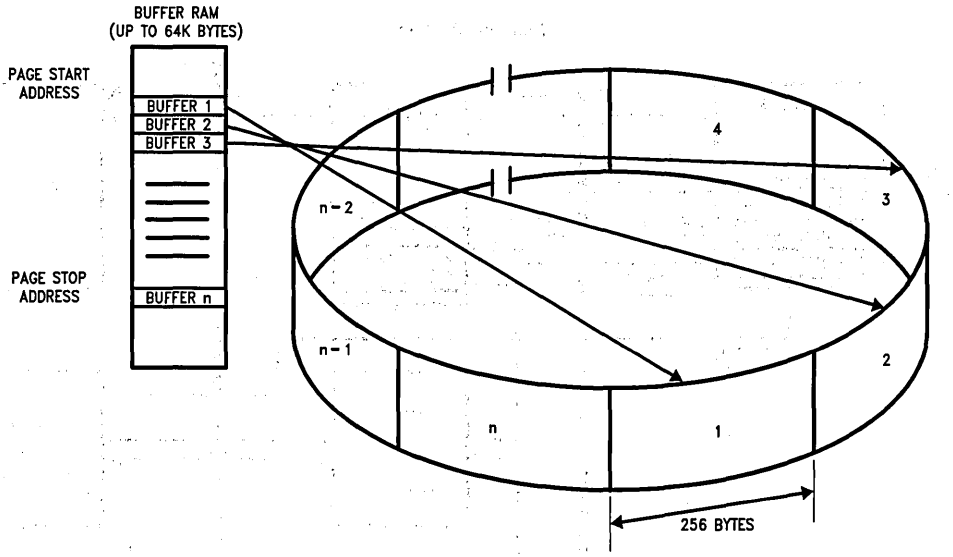
TL/F/10469–6

7.0 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256-byte (128 word) buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop Register. Ethernet packets consist of a distribution of shorter link control packets and longer data packets, the 256-byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers for storing packets is controlled by Buffer Management Logic in the SNIC. The Buffer Management Logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

At initialization, a portion of the 64 kbyte (or 32k word) address space is reserved for the receive buffer ring. Two 8-bit registers, The Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The SNIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.

NIC Receive Buffer Ring



TL/F/10469–7

7.0 Packet Reception (Continued)

INITIALIZATION OF THE BUFFER RING

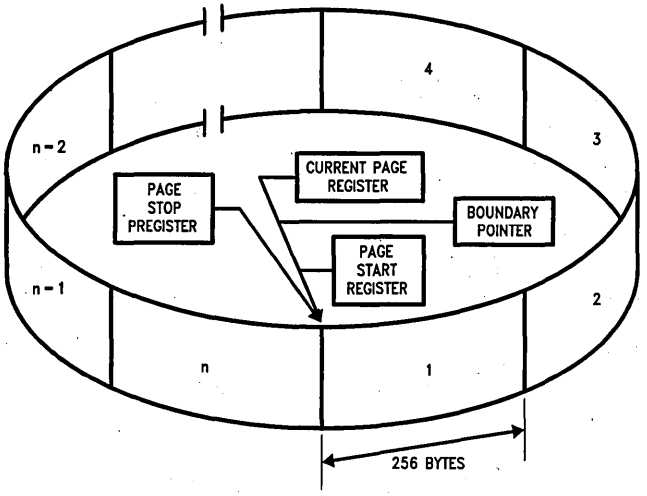
Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register (both described previously), the Current Page Register and the Boundary Pointer Register. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.

Note: The Page Start Register must not be initialized to 00H.

BEGINNING OF RECEPTION

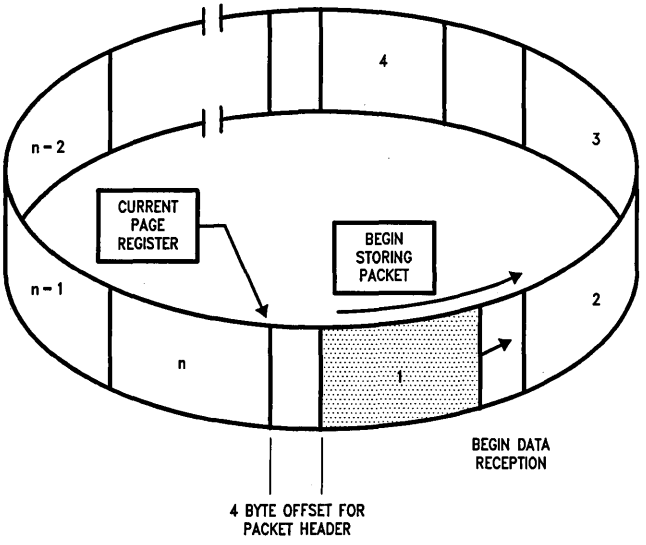
When the first packet begins arriving the SNIC begins storing the packet at the location pointed to by the Current Page Register. An offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.

Buffer Ring at Initialization



TL/F/10469–8

Received Packet Enters the Buffer Pages



TL/F/10469–9

7.0 Packet Reception (Continued)

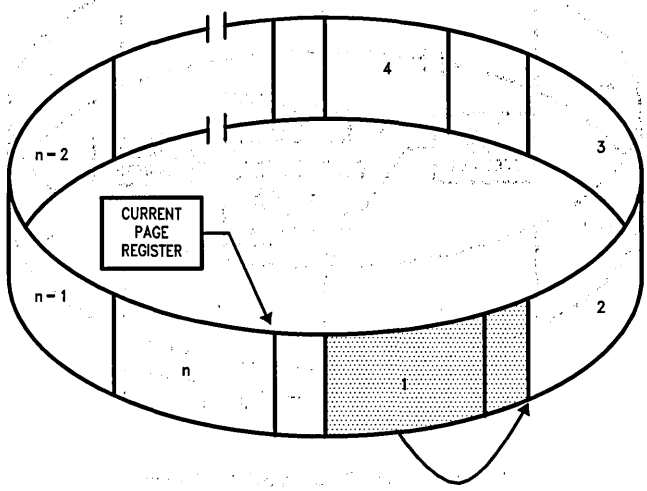
LINKING RECEIVE BUFFER PAGES

If the length of the packet exhausts the first 256-byte buffer, the DMA performs a forward link to the next buffer to store the remainder of the packet. For a maximal length packet the buffer logic will link six buffers to store the entire packet. Buffers cannot be skipped when linking, a packet will always be stored in contiguous buffers. Before the next buffer can be linked, the Buffer Management Logic performs two comparisons. The first comparison tests for equality between the DMA address of the next buffer and the contents of the Page Stop Register. If the buffer address equals the Page Stop Register, the buffer management logic will restore the DMA to the first buffer in the Receive Buffer Ring value programmed in the Page Start Address Register. The second comparison tests for equality between the DMA address of the next buffer address and the contents of the Boundary Pointer Register. If the two values are equal the reception is aborted. The Boundary Pointer Register can be used to protect against overwriting any area in the receive buffer ring that has not yet been read. When linking buffers, buffer management will never cross this pointer, effectively avoiding any overwrites. If the buffer address does not match either the Boundary Pointer or Page Stop Address, the link to the next buffer is performed.

Linking Buffers

Before the DMA can enter the next contiguous 256-byte buffer, the address is checked for equality to PSTOP and to the Boundary Pointer. If neither are reached, the DMA is allowed to use the next buffer.

Linking Receive Buffer Pages



TL/F/10469–10

1) Check for = to PSTOP
2) Check for = to Boundary

7.0 Packet Reception (Continued)

Buffer Ring Overflow

If the Buffer Ring has been filled and the DMA reaches the Boundary Pointer Address, reception of the incoming packet will be aborted by the SNIC. Thus, the packets previously received and still contained in the Ring will not be destroyed.

In heavily loaded network which cause overflows of the Receive Buffer Ring, the SNIC may disable the local DMA and suspend further receptions even if the Boundary register is advanced beyond the Current register. To guarantee this will not happen, a software reset must be issued during all Receive Buffer Ring overflows (indicated by the OVW bit in the Interrupt Status Register). The following procedure is required to recover from a Receiver Buffer Ring Overflow.

If this routine is not adhered to, the SNIC may act in an unpredictable manner. It should also be noted that it is not permissible to service an overflow interrupt by continuing to empty packets from the receive buffer without implementing the prescribed overflow routine. A flow chart of the SNIC's overflow routine can be found on the next page.

Note: It is necessary to define a variable in the driver, which will be called "Resend".

1. Read and store the value of the TXP bit in the SNIC's Command Register.
2. Issue the STOP command to the SNIC. This is accomplished by setting the STP bit in the SNIC's Command Register. Writing 21H to the Command Register will stop the SNIC.
3. Wait for at least 1.6 ms. Since the SNIC will complete any transmission or reception that is in progress, it is necessary to time out for the maximum possible duration of an Ethernet transmission or reception. By waiting 1.6 ms this is achieved with some guard band added. Previously, it was recommended that the RST bit of the Interrupt Status Register be polled to insure that the pending transmission or reception is completed. This bit is not a reliable indicator and subsequently should be ignored.
4. Clear the SNIC's Remote Byte Count registers (RBCR0 and RBCR1).
5. Read the stored value of the TXP bit from step 1, above.

   If this value is a 0, set the "Resend" variable to a 0 and jump to step 6.

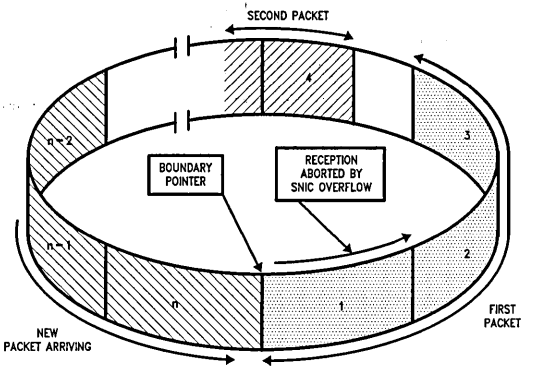
   If this value is a 1, read the SNIC's Interrupt Status Register. If either the Packet Transmitted bit (PTX) or Transmit Error bit (TXE) is set to a 1, set the "Resend" variable to a 0 and jump to step 6. If neither of these bits is set, place a 1 in the "Resend" variable and jump to step 6.

   This step determines if there was a transmission in progress when the stop command was issued in step 2. If there was a transmission in progress, the SNIC's ISR is read to determine whether or not the packet was recognized by the SNIC. If neither the PTX nor TXE bit was set, then the packet will essentially be lost and re-transmitted only after a time-out takes place in the upper level software. By determining that the packet was lost at the driver level, a transmit command can be reissued to the SNIC once the overflow routine is completed (as in step 11). Also, it is possible for the SNIC to defer indefinitely, when it is stopped on a busy network. Step 5 also alleviates this problem. Step 5 is essential and should not be omitted from the overflow routine, in order for the SNIC to operate correctly.
6. Place the SNIC in either mode 1 or mode 2 loopback. This can be accomplished by setting bits D2 and D1, of the Transmit Configuration Register, to "0,1" or "1,0", respectively.
7. Issue the START command to the SNIC. This can be accomplished by writing 22H to the Command Register. This is necessary to activate the SNIC's Remote DMA channel.
8. Remove one or more packets from the receive buffer ring.
9. Reset the overwrite warning (OVW, overflow) bit in the Interrupt Status Register.
10. Take the SNIC out of loopback. This is done by writing the Transmit Configuration Register with the value it contains during normal operation. (Bits D2 and D1 should both be programmed to 0.)
11. If the "Resend" variable is set to a 1, reset the "Resend" variable and reissue the transmit command. This is done by writing a value of 26H to the Command Register. If the "Resend" variable is 0, nothing needs to be done.

Note 1: If Remote DMA is not being used, the SNIC does not need to be started before packets can be removed from the receive buffer ring. Hence, step 8 could be done before step 7.

Note 2: When the SNIC is in STOP mode, the Missed Talley Counter is disabled.

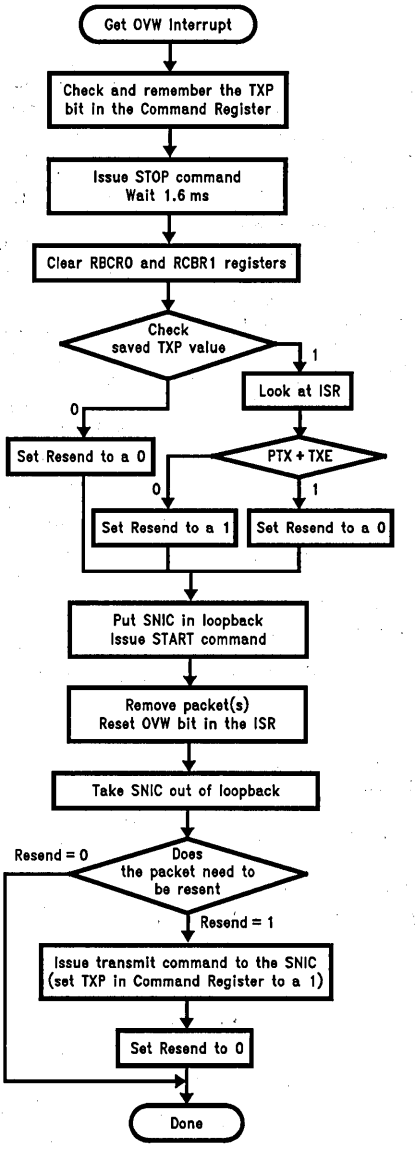
Received Packet Aborted If It Hits Boundary



TL/F/10469–11

7.0 Packet Reception (Continued)

Overflow Routine Flow Chart



TL/F/10469–52

7.0 Packet Reception (Continued)

Enabling the SNIC On An Active Network

After the SNIC has been initialized the procedure for disabling and then re-enabling the SNIC on the network is similar to handling Receive Buffer Ring overflow as described previously.

1. Program Command Register for page 0 (Command Register = 21H)
2. Initialize Data Configuration Register (DCR)
3. Clear Remote Byte Count Registers (RBCR0, RBCR1)
4. Initialize Receive Configuration Register (RCR)
5. Place the SNIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6. Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7. Clear Interrupt Status Register (ISR) by writing 0FFH to it.
8. Initialize Interrupt Mask Register (IMR)
9. Program Command Register for page 1 (Command Register = 61H)
   i) Initialize Physical Address Registers (PAR0–PAR5)
   ii) Initialize Multicast Address Registers (MAR0–MAR7)
   iii) Initialize CURRent pointer
10. Put SNIC in START mode (Command Register = 22H). The local receive DMA is still not active since the SNIC is in LOOPBACK.
11. Initialize the Transmit Configuration for the intended value. The SNIC is now ready for transmission and reception.

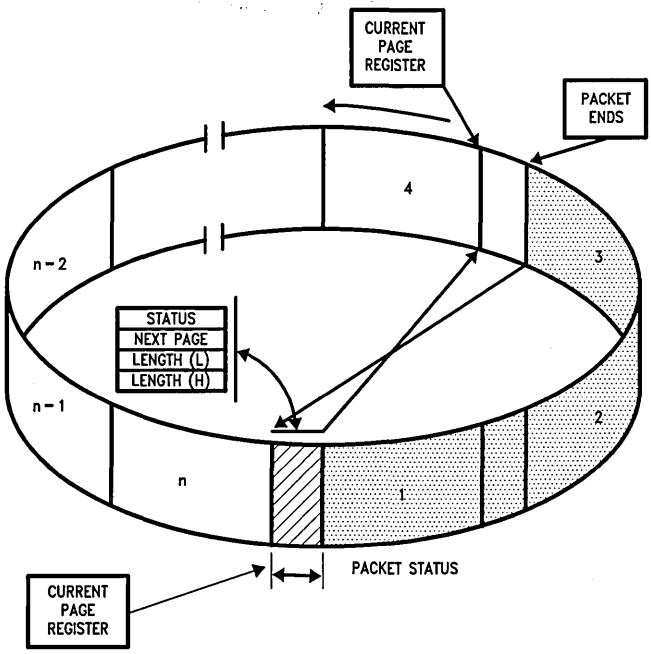
END OF PACKET OPERATIONS

At the end of the packet the SNIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

SUCCESSFUL RECEPTION

If the packet is successfully received, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. Note that the remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring. (The location of the next buffer had been previously calculated and temporarily stored in an internal scratchpad register.)

Termination of Received Packet—Packet Accepted



TL/F/10469–12

7.0 Packet Reception (Continued)

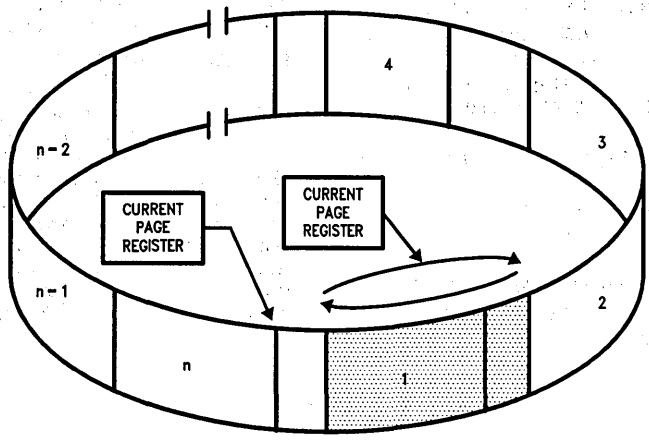
BUFFER RECOVERY FOR REJECTED PACKETS

If the packet is a runt packet or contains CRC or Frame Alignment errors, it is rejected. The buffer management logic resets the DMA back to the first buffer page used to store the packet (pointed to by CURR), recovering all buffers that had been used to store the rejected packet. This operation will not be performed if the SNIC is programmed to accept either runt packets or packets with CRC or Frame Alignment errors. The received CRC is always stored in buffer memory after the last byte of received data for the packet.

Error Recovery

If the packet is rejected as shown, the DMA is restored by the SNIC by reprogramming the DMA starting address pointed to by the Current Page Register.

Termination of Receive Packet—Packet Reject



TL/F/10469–13

7.0 Packet Reception (Continued)

REMOVING PACKETS FROM THE RING

Packets are removed from the ring using the Remote DMA or an external device. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the SNIC moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer.

The following is a suggested method for maintaining the Receive Buffer Ring pointers.

1. At initialization, set up a software variable (next_pkt) to indicate where the next packet will be read. At the beginning of each Remote Read DMA operation, the value of next_ pkt will be loaded into RSAR0 and RSAR1.
2. When initializing the SNIC set:
   BNDRY = PSTART
   CURR = PSTART + 1
   next_pkt = PSTART + 1
3. After a packet is DMAed from the Receive Buffer Ring, the Next Page Pointer (second byte in the SNIC buffer header is used to updte BNDRY and next_pkt.
   next_pkt = Next Page Pointer
   BNDRY = Next Page Pointer − 1
   If BNDRY <PSTART then BNDRY = PSTOP − 1

Note the size of the Receive Buffer Ring is reduced by one 256-byte buffer, this will not, however, impede the operation of the SNIC.

STORAGE FORMAT FOR RECEIVED PACKETS

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register.

AD15 AD8	AD7 AD0
Next Packet Pointer	Receive Status
Receive Byte Count 1	Receive Byte Count 0
Byte 2	Byte 1

BOS = 0, WTS = 1 in Data Configuration Register. This format is used with Series 32xxx, or 808xx processors.

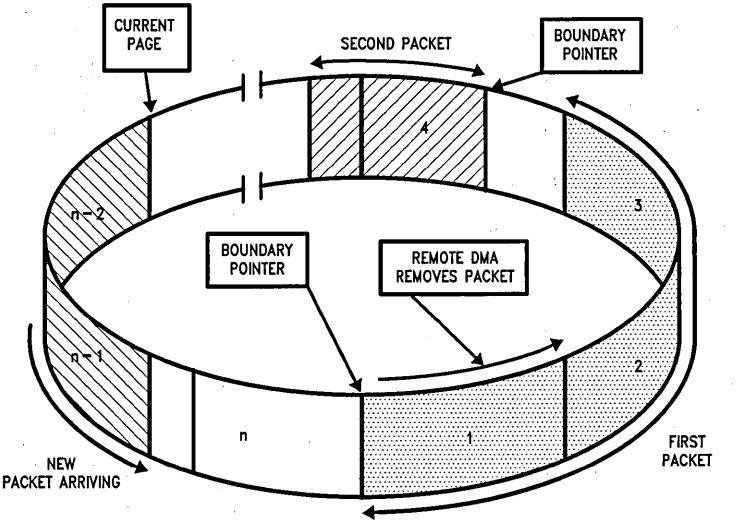
AD15 AD8	AD7 AD0
Next Packet Pointer	Receive Status
Receive Byte Count 0	Receive Byte Count 1
Byte 1	Byte 2

BOS = 1, WTS = 1 in Data Configuration Register. This format is used with 680x0 type processors. **(Note:** The Receive Count ordering remains the same for BOS = 0 or 1.)

Receive Status
Next Packet Pointer
Receive Byte Count 0
Receive Byte Count 1
Byte 0
Byte 1

BOS = 0, WTS = 0 in Data Configuration Register. This format is used with general 8-bit processors.

1st Received Packet Removed by Remote DMA



TL/F/10469–14

8.0 Packet Transmission

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0, 1). When the SNIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The SNIC will generate and append the preamble, synch and CRC fields.

General Transmit Packet Format

Transmit	Destination Address	6 Bytes
Byte	Source Address	6 Bytes
Count	Type/Length	2 Bytes
TBCR0, 1	Data	≥46 Bytes
	Pad (If Data < 46 Bytes)	

TRANSMIT PACKET ASSEMBLY

The SNIC requires a contiguous assembled packet with the format shown. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes. The programmer is responsible for adding and stripping pad bytes.

TRANSMISSION

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR) is cleared and the SNIC begins to prefetch transmit data from memory (unless the SNIC is currently receiving). If the interframe gap has timed out the SNIC will begin transmission.

CONDITIONS REQUIRED TO BEGIN TRANSMISSION

In order to transmit a packet, the following three conditions must be met:

1. The Interframe Gap Timer has timed out the first 6.4 μs of the Interframe Gap.
2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started.)
3. If a collision had been detected then before transmission the packet time must have timed out.

In typical systems the SNIC prefetchs the first burst of bytes before the 6.4 μs timer expires. The time during which SNIC transmits preamble can also be used to load the FIFO.

Note: If carrier sense is asserted before a byte has been loaded into the FIFO, the SNIC will become a receiver.

COLLISION RECOVERY

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

Note: NCR reads as zeroes if excessive collisions are encountered.

TRANSMIT PACKET ASSEMBLY FORMAT

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register.

D15 D8	D7 D0
Destination Address 1	Destination Address 0
Destination Address 3	Destination Address 2
Destination Address 5	Destination Address 4
Source Address 1	Source Address 0
Source Address 3	Source Address 2
Source Address 5	Source Address 4
Type/Length 1	Type/Length 0
Data 1	Data 0

BOS = 0, WTS = 1 in Data Configuration Register.

This format is used with Series 32xxx, or 808xx processors.

D15 D8	D7 D0
Destination Address 0	Destination Address 1
Destination Address 2	Destination Address 3
Destination Address 4	Destination Address 5
Source Address 0	Source Address 1
Source Address 2	Source Address 3
Source Address 4	Source Address 5
Type/Length 0	Type/Length 1
Data 0	Data 1

BOS = 1, WTS = 1 in Data Configuration Register.

This format is used with 680x0 type processors.

D1 D0
Destination Address 0
Destination Address 1
Destination Address 2
Destination Address 3
Destination Address 4
Destination Address 5
Source Address 0
Source Address 1
Source Address 2
Source Address 3
Source Address 4
Source Address 5

BOS = 0, WTS = 0 in Data Configuration Register.

This format is used with general 8-bit processors.

Note: All examples above will result in a transmission of a packet in order of DA0, DA1, DA3 ... bits within each byte will be transmitted least significant bit first.

DA = Destination Address.

9.0 Remote DMA

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet.

Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address Register pair points to the beginning of the block to be moved while the Byte Count Register pair is used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

REMOTE WRITE

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches a count of zero.

REMOTE READ

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

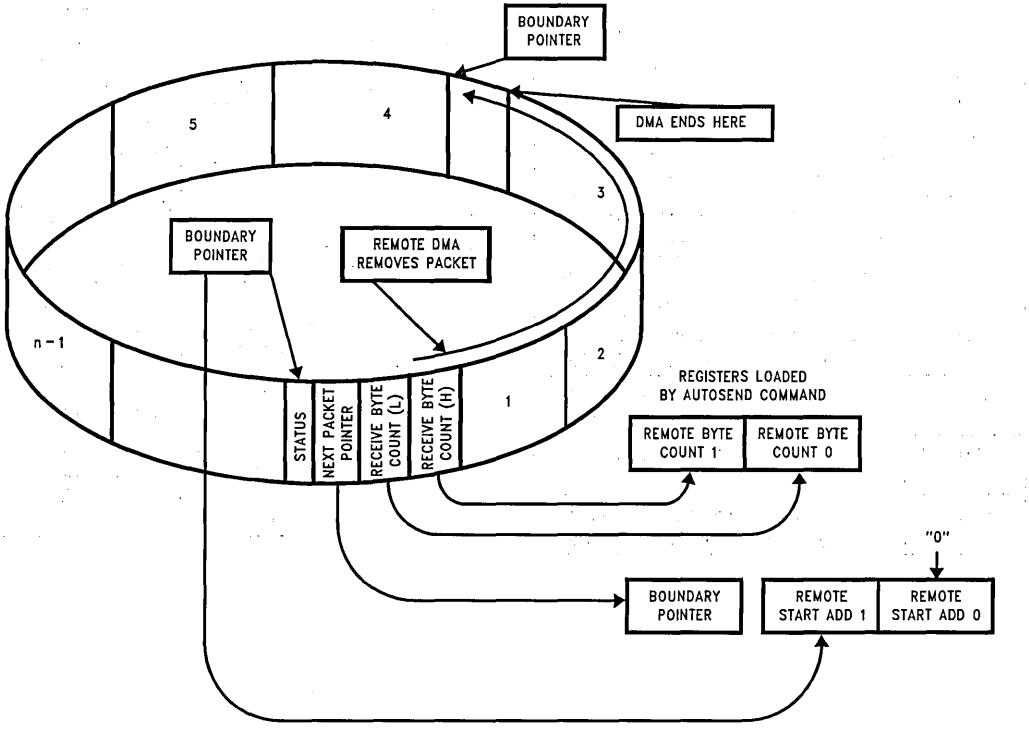
SEND PACKET COMMAND

The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a "Send Packet" Command. The DMA will be initialized to the value of the Boundary Pointer Register and the Remote Byte Count Register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop Register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring.

Note 1: In order for the SNIC to correctly execute the Send Packet Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 0FH.

Note 2: The Send Packet command cannot be used with 680x0 type processors.

Remote DMA Autoinitialization from Buffer Ring

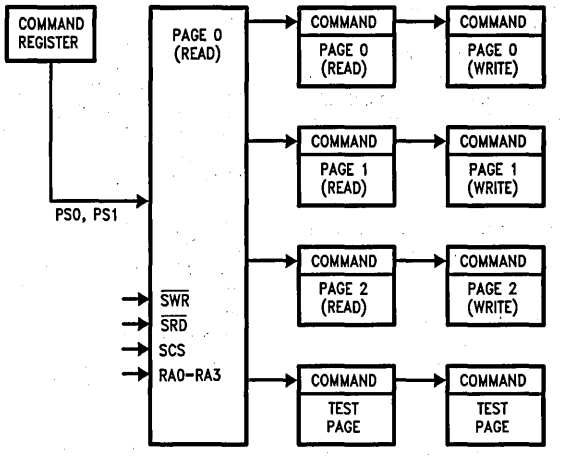


TL/F/10469–15

10.0 Internal Registers

All registers are 8-bit wide and mapped into four pages which are selected in the Command Register (PS0, PS1). Pins RA0–RA3 are used to address registers within each page. Page 0 registers are those registers which are commonly accessed during SNIC operation while page 1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.

10.1 REGISTER ADDRESS MAPPING



TL/F/10469–16

10.2 REGISTER ADDRESS ASSIGNMENTS

Page 0 Address Assignments (PS1 = 0, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

10.0 Internal Registers (Continued)

Page 1 Address Assignments (PS1 = 0, PS0 = 1)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

Page 2 Address Assignments (PS1 = 1, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Page Start Register (PSTART)	Current Local DMA Address 0 (CLDA0)
02H	Page Stop Register (PSTOP)	Current Local DMA Address 1 (CLDA1)
03H	Remote Next Packet Pointer	Remote Next Packet Pointer
04H	Transmit Page Start Address (TPSR)	Reserved
05H	Local Next Packet Pointer	Local Next Packet Pointer
06H	Address Counter (Upper)	Address Counter (Upper)
07H	Address Counter (Lower)	Address Counter (Lower)
08H	Reserved	Reserved
09H	Reserved	Reserved
0AH	Reserved	Reserved
0BH	Reserved	Reserved
0CH	Receive Configuration Register (RCR)	Reserved
0DH	Transmit Configuration Register (TCR)	Reserved
0EH	Data Configuration Register (DCR)	Reserved
0FH	Interrupt Mask Register (IMR)	Reserved

Note: Page 2 registers should only be accessed for diagnostic purposes. They should not be modified during normal operation.

Page 3 should never be modified.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS

COMMAND REGISTER (CR) 00H (READ/WRITE)

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands may be overlapped, but with the following rules: (1) If a transmit command overlaps with a remote DMA operation, bits RD0, RD1, and RD2 must be maintained for the remote DMA command when setting the TXP bit. Note, if a remote DMA command is re-issued when giving the transmit command, the DMA will complete immediately if the remote byte count register has not been reinitialized. (2) If a remote DMA operation overlaps a transmission, RD0, RD1, and RD2 may be written with the desired values and a "0" written to the TXP bit. Writing a "0" to this bit has no effect. (3) A remote write DMA may not overlap remote read operation or visa versa. Either of these operations must either complete or be aborted before the other operation may start. Bits PS1, PS0, RD2, and STP may be set any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

Bit	Symbol	Description
D0	STP	**Stop:** Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to 1. **STP powers up high.** **Note:** If the SNIC has previously been in start mode and the STP is set, both the STP and STA bits will remain set.
D1	STA	**Start:** This bit is used to activate the SNIC after either power up, or when the SNIC has been placed in a reset mode by software command or error. **STA powers up low.**
D2	TXP	**Transmit Packet:** This bit must be set to initiate transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed.
D3, D4, and D5	RD0, RD1, and RD2	**Remote DMA Command:** These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. RD2 RD1 RD0 0 0 0 Not Allowed 0 0 1 Remote Read 0 1 0 Remote Write (Note 2) 0 1 1 Send Packet 1 X X Abort/Complete Remote DMA (Note 1) **Note 1:** If a remote DMA operation is aborted and the remote byte count has not decremented to zero. PRQ will remain high. A read acknowledge (RACK) on a write acknowledge (WACK) will reset PRQ low. **Note 2:** For proper operation of the Remote Write DMA, there are two steps which must be performed before using the Remote Write DMA. The steps are as follows: I) Write a non-zero value into RBCR0. II) Set bits RD2, RD1, and RD0 to 0, 0, and 1. III) Issue the Remote Write DMA Command (RD2, RD1, RD0 = 0, 1, 0).
D6 and D7	PS0 and PS1	**Page Select:** These two encoded bits select which register page is to be accessed with addresses RA0–3. PS1 PS0 0 0 Register Page 0 0 1 Register Page 1 1 0 Register Page 2 1 1 Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

INTERRUPT STATUS REGISTER (ISR) 07H (READ/WRITE)

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a "1" into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX

Bit	Symbol	Description
D0	PRX	**Packet Received:** Indicates packet received with no errors.
D1	PTX	**Packet Transmitted:** Indicates packet transmitted with no errors.
D2	RXE	**Receive Error:** Indicates that a packet was received with one or more of the following errors: — CRC Error — Frame Alignment Error — FIFO Overrun — Missed Packet
D3	TXE	**Transmit Error:** Set when packet transmitted with one or more of the following errors: — Excessive Collisions — FIFO Underrun
D4	OVW	**Overwrite Warning:** Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer)
D5	CNT	**Counter Overflow:** Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	**Remote DMA Complete:** Set when Remote DMA operation has been completed.
D7	RST	**Reset Status:** Set when SNIC enters reset state and cleared when a Start Command is issued to the CR. This bit is also set when a Receive Buffer Ring overflow occurs and is cleared when one or more packets have been removed from the ring. Writing to this bit has no effect. **Note:** This bit does not generate an interrupt, it is merely a status indicator.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

INTERRUPT MASK REGISTER (IMR) 0FH (WRITE)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set, an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. **The IMR powers up to all zeroes.**

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Bit	Symbol	Description
D0	PRXE	**Packet Received Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet received
D1	PTXE	**Packet Transmitted Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted
D2	RXEE	**Receive Error Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet received with error
D3	TXEE	**Transmit Error Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error
D4	OVWE	**Overwrite Warning Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet
D5	CNTE	**Counter Overflow Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set
D6	RDCE	**DMA Complete Interrupt Enable** 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed
D7	Reserved	Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

DATA CONFIGURATION REGISTER (DCR) 0EH (WRITE)

This Register is used to program the SNIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO thresholds. **The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.**

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	LS	LAS	BOS	WTS

Bit	Symbol	Description
D0	WTS	**Word Transfer Select** 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers ; WTS establishes byte or word transfers for both Remote and Local DMA transfers **Note:** When word-wide mode is selected up to 32k words are addressable; A0 remains low.
D1	BOS	**Byte Order Select** 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32xxx, 80x86) 1: MS byte placed on AD7–AD0 and LS byte on AD15–A8. (680x0) : Ignored when WTS is low
D2	LAS	**Long Address Select** 0: Dual 16-bit DMA mode 1: Single 32-bit DMA mode ; When LAS is high, the contents of the Remote DMA registers RSAR0, 1 are issued as A16–A31 Power up high
D3	LS	**Loopback Select** 0: Loopback mode selected. Bits D1 and D2 of the TCR must also be programmed for Loopback operation 1: Normal Operation
D4	ARM	**Auto-Initialize Remote** 0: Send Command not executed, all packets removed from Buffer Ring under program control 1: Send Command executed, Remote DMA auto-initialized to remove packets from Buffer Ring **Note:** Send Command cannot be used with 680x0 byte processors.
D5 and D6	FT0 and FT1	**FIFO Threshold Select:** Encoded FIFO threshold. Establishes point at which bus is requested when filling or emptying the FIFO. During reception, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled serially from the network before bus request (BREQ) is asserted. **Note:** FIFO threshold setting determines the DMA burst length. Receive Thresholds FT1 FT0 Word Wide Byte Wide 0 0 1 Word 2 Bytes 0 1 2 Words 4 Bytes 1 0 4 Words 8 Bytes 1 1 6 Words 12 Bytes During transmission, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled from the Local DMA before BREQ is asserted. Thus, the transmission threshold is 16 bytes less the received threshold.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

TRANSMIT CONFIGURATION REGISTER (TCR) 0DH (WRITE)

The transmit configuration establishes the actions of the transmitter section of the SNIC during transmission of a packet on the network. **LB1 and LB0 which select loopback mode power up as 0.**

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

Bit	Symbol	Description
D0	CRC	**Inhibit CRC** 0: CRC appended by transmitter 1: CRC inhibited by transmitter In loopback mode CRC can be enabled or disabled to test the CRC logic
D1 and D2	LB0 and LB1	**Encoded Loopback Control:** These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 places the ENDEC Module in loopback mode and that D3 of the DCR must be set to zero for loopback operation. LB1 LB0 Mode 0 0 0 Normal Operation (LPBK = 0) Mode 1 0 1 Internal NIC Module Loopback (LPBK = 0) Mode 2 1 0 Internal ENDEC Module Loopback (LPBK = 1) Mode 3 1 1 External Loopback (LPBK = 0)
D3	ATD	**Auto Transmit Disable:** This bit allows another station to disable the SNIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.
D4	OFST	**Collision Offset Enable:** This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n,\,10)}$ slot times for first three collisions, then follows standard backoff. (For the first three collisions, the station has higher average backoff delay making a low priority mode.)
D5	Reserved	Reserved
D6	Reserved	Reserved
D7	Reserved	Reserved

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

TRANSMIT STATUS REGISTER (TSR) 04H (READ)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	—	PTX

Bit	Symbol	Description
D0	PTX	**Packet Transmitted:** Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0")
D1	Reserved	Reserved
D2	COL	**Transmit Collided:** Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	**Transmit Aborted:** Indicates the SNIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16.)
D4	CRS	**Carrier Sense Lost:** This bit is set when carrier is lost during transmission of the packet. Transmission is not aborted on loss of carrier.
D5	FU	**FIFO Underrun:** If the SNIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	**CD Heartbeat:** Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 μs of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	**Out of Window Collision:** Indicates that a collision occurred after a slot time (51.2 μs). Transmissions rescheduled as in normal collisions.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

RECEIVE CONFIGURATION REGISTER (RCR) 0CH (WRITE)

This register determines operation of the SNIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

Bit	Symbol	Description
D0	SEP	**Save Errored Packets** 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	**Accept Runt Packets:** This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	**Accept Broadcast:** Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	**Accept Multicast:** Enables the receiver to accept a packet with a multicast address, all multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	**Promiscuous Physical:** Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	**Monitor Mode:** Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	Reserved	Reserved
D7	Reserved	Reserved

Note: D2 and D3 are "OR'd" together, i.e., if D2 and D3 are set the SNIC will accept broadcast and multicast addresses as well as its own physical address. To establish full promiscuous mode, bits D2, D3, and D4 should be set. In addition the multicast hashing array must be set to all 1's in order to accept all multicast addresses.

10.0 Internal Registers (Continued)

10.3 REGISTER DESCRIPTIONS (Continued)

RECEIVE STATUS REGISTER (RSR) 0CH (READ)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the SNIC which relinguishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

Bit	Symbol	Description
D0	PRX	**Packet Received Intact:** Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	**CRC Error:** Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	**Frame Alignment Error:** Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	**FIFO Overrun:** This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	**Missed Packet:** Set when a packet intended for node cannot be accepted by SNIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	**Physical/Multicast Address:** Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Physical Address Match
D6	DIS	**Receiver Disabled:** Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	**Deferring:** Set when internal Carrier Sense or Collision signals are generated in the ENDEC module. If the transceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.

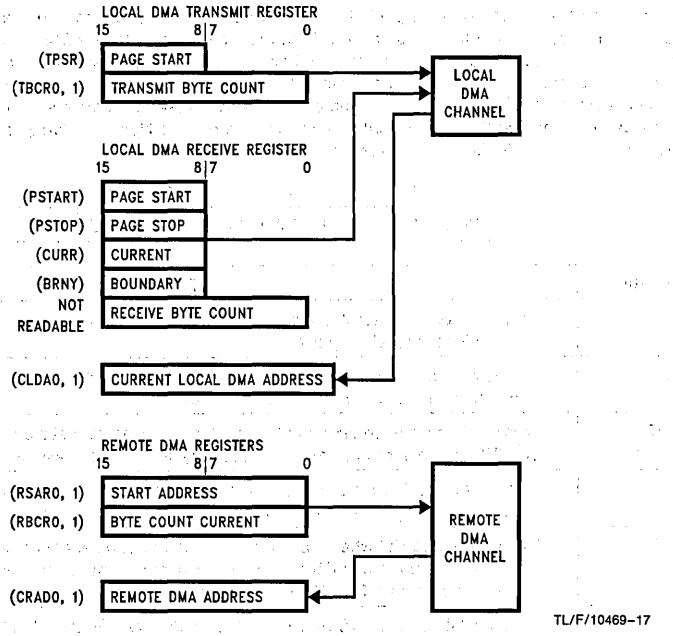
Note: Following coding applies to CRC and FAE bits.

FAE	CRC	Type of Error
0	0	No Error (Good CRC and <6 Dribble Bits)
0	1	CRC Error
1	0	Illegal, Will Not Occur
1	1	Frame Alignment Error and CRC Error

10.0 Internal Registers (Continued)

10.4 DMA REGISTERS

DMA Registers



TL/F/10469–17

The DMA Registers are partitioned into groups; Transmit, Receive and Remote DMA Registers. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive Registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary Registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.

Note: In the figure above, registers are shown as 8 or 16 bits wide. Although some registers are 16-bit internal registers, all registers are accessed as 8-bit registers. Thus the 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0, TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus. Thus, they are shifted to positions 15–8 in the diagram above.

10.5 TRANSMIT DMA REGISTERS

TRANSMIT PAGE START REGISTER (TPSR)

This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256-byte page boundaries. The bit assignment is shown below. The values placed in bits D7–D0 will be used to initialize the higher order address (A8–A15) of the Local DMA for transmission. The lower order bits (A7–A0) are initialized to zero.

Bit Assignment

	7	6	5	4	3	2	1	0
TPSR	A15	A14	A13	A12	A11	A10	A9	A8

(A7–A0 Initialized to Zero)

TRANSMIT BYTE COUNT REGISTER 0, 1 (TBCR0, TBCR1)

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64 Kbytes. The SNIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
TBCR1	L15	L14	L13	L12	L11	L10	L9	L8

	7	6	5	4	3	2	1	0
TBCR0	L7	L6	L5	L4	L3	L2	L1	L0

10.0 Internal Registers (Continued)

10.6 LOCAL DMA RECEIVE REGISTERS

PAGE START AND STOP REGISTERS (PSTART, PSTOP)

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the SNIC uses fixed 256-byte buffers aligned on page boundaries only the upper 8 bits of the start and stop address are specified.

PSTART, PSTOP Bit Assignment

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

BOUNDARY (BNRY) REGISTER

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT PAGE REGISTER (CURR)

This register is used internally by the Buffer Management Logic as a backup register for reception. CURR contains the address of the first buffer to be used for a packet reception and is used to restore DMA pointers in the event of receive errors. This register is initialized to the same value as PSTART and should not be written to again unless the controller is Reset.

	7	6	5	4	3	2	1	0
CURR	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT LOCAL DMA REGISTER 0,1 (CLDA0, 1)

These two registers can be accessed to determine the current local DMA address.

	7	6	5	4	3	2	1	0
CLDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CLDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.7 REMOTE DMA REGISTERS

REMOTE START ADDRESS REGISTERS (RSAR0, 1)

Remote DMA operations are programmed via the Remote Start Address (RSAR0, 1) and Remote Byte Count (RBCR0, 1) registers. The Remote Start Address is used to point to the start of the block of data to be transferred and the Remote Byte Count is used to indicate the length of the block (in bytes).

	7	6	5	4	3	2	1	0
RSAR1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
RSAR0	A7	A6	A5	A4	A3	A2	A1	A0

REMOTE BYTE COUNT REGISTERS (RBCR0, 1)

	7	6	5	4	3	2	1	0
RBCR1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8

	7	6	5	4	3	2	1	0
RBCR0	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

Note: RSAR0 programs the start address bits A0–A7.
RSAR1 programs the start address bits A8–A15.
Address incremented by two for word transfers, and by one for byte transfers. Byte Count decremented by two for word transfers and by one for byte transfers.
RBCR0 programs LSB byte count.
RBCR1 programs MSB byte count.

CURRENT REMOTE DMA ADDRESS (CRDA0, CRDA1)

The Current Remote DMA Registers contain the current address of the Remote DMA. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
CRDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CRDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.8 PHYSICAL ADDRESS REGISTERS (PAR0–PAR5)

The physical address registers are used to compare the destination address of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte-wide basis. The bit assignment shown below relates the sequence in PAR0–PAR5 to the bit sequence of the received packet.

	D7	D6	D5	D4	D3	D2	D1	D0
PAR0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
PAR1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
PAR2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
PAR3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
PAR4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
PAR5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40

						Destination Address			Source	
P/S	DA0	DA1	DA2	DA3	...	DA46	DA47	SA0	...	

Note: P/S = Preamble, Synch
DA0 = Physical/Multicast Bit

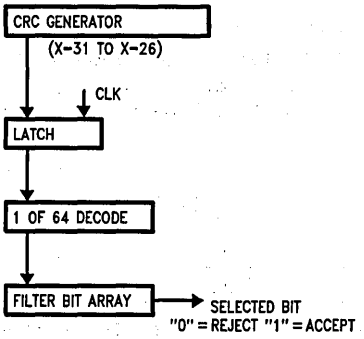
10.9 MULTICAST ADDRESS REGISTERS (MAR0–MAR7)

The multicast address registers provide filtering of multicast addresses hashed by the CRC logic. All destination addresses are fed through the CRC logic and as the last bit of the destination address enters the CRC, the 6 most signifi-

10.0 Internal Registers (Continued)

cant bits of the CRC generator are latched. These 6 bits are then decoded by a 1 of 64 decode to index a unique filter bit (FB0–63) in the multicast address registers. If the filter bit selected is set, the multicast packet is accepted. The system designer would use a program to determine which filter bits to set in the multicast registers. All multicast filter bits that correspond to multicast address accepted by the node are then set to one. To accept all multicast packets all of the registers are set to all ones.

Note: Although the hashing algorithm does not guarantee perfect filtering of multicast address, it will perfectly filter up to 64 multicast addresses if these addresses are chosen to map into unique locations in the multicast filter.



TL/F/10469–18

	D7	D6	D5	D4	D3	D2	D1	D0
MAR0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MAR1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MAR2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MAR3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MAR4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MAR5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MAR6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MAR7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

If address Y is found to hash to the value 32 (20H), then FB32 in MAR4 should be initialized to "1". This will cause the SNIC to accept any multicast packet with the address Y.

10.10 NETWORK TALLY COUNTERS

Three 8-bit counters are provided for monitoring the number of CRC errors, Frame Alignment Errors and Missed Packets. The maximum count reached by any counter is 192 (C0H). These registers will be cleared when read by the CPU. The count is recorded in binary in CT0–CT7 of each Tally Register.

Frame Alignment Error Tally (CNTR0)

This counter increments every time a packet is received with a Frame Alignment Error. The packet must have been recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR0	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

CRC Error Tally (CNTR1)

This counter is incremented every time a packet is received with a CRC error. The packet must first be recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR1	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

Frames Lost Tally Register (CNTR2)

This counter is incremented if a packet cannot be received due to lack of buffer resources. In monitor mode, this counter will count the number of packets that pass the address recognition logic.

	7	6	5	4	3	2	1	0
CNTR2	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

FIFO

This is an 8-bit register that allows the CPU to examine the contents of the FIFO after loopback. The FIFO will contain the last 8 data bytes transmitted in the loopback packet. Sequential reads from the FIFO will advance a pointer in the FIFO and allow reading of all 8 bytes.

	7	6	5	4	3	2	1	0
FIFO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Note: The FIFO should only be read when the SNIC has been programmed in loopback mode.

NUMBER OF COLLISIONS (NCR)

This register contains the number of collisions a node experiences when attempting to transmit a packet. If no collisions are experienced during a transmission attempt, the COL bit of the TSR will not be set and the contents of NCR will be zero. If there are excessive collisions, the ABT bit in the TSR will be set and the contents of NCR will be zero. The NCR is cleared after the TXP bit in the CR is set.

	7	6	5	4	3	2	1	0
NCR	—	—	—	—	NC3	NC2	NC1	NC0

11.0 Initialization Procedures

The SNIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the SNIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The SNIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the SNIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

Initialization Sequence

The following initialization procedure is mandatory.

1. Program Command Register for Page 0 (Command Register = 21H)
2. Initialize Data Configuration Register (DCR)
3. Clear Remote Byte Count Registers (RBCR0, RBCR1)
4. Initialize Receive Configuration Register (RCR)
5. Place the SNIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6. Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7. Clear Interrupt Status Register (ISR) by writing 0FFH to it.
8. Initialize Interrupt Mask Register (IMR)
9. Program Command Register for page 1 (Command Register = 61H)
   I) Initialize Physical Address Registers (PAR0–PAR5)
   II) Initialize Multicast Address Registers (MAR0–MAR5)
   III) Initialize CURRent pointer
10. Put SNIC in START mode (Command Register = 22H).
11. Initialize the Transmit Configuration for the intended value. The SNIC is now ready for transmission and reception.

Before receiving packets, the user must specify the location of the Receive Buffer Ring. This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Register must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

12.0 Loopback Diagnostics

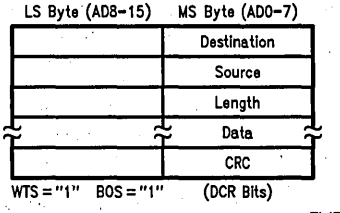
Three forms of local loopback are provided on the SNIC. The user has the ability to loopback through the deserializer on the controller, through the ENDEC module or the Coax Transceiver on the DP83901A SNIC. **Because of the half duplex architecture of the SNIC, loopback testing is a special mode of operation with the following restrictions:**

Restrictions During Loopback

The FIFO is split into two halves, one used for transmission the other for reception. Only 8-bit fields can be fetched from memory so two tests are required for 16-bit systems to verify integrity of the entire data path. During loopback the maximum latency from the assertion of BREQ to BACK is 2.0 μs. Systems that wish to use the loopback test yet do not meet this latency can limit the loopback to 7 bytes without experiencing underflow. Only the last 8 bytes of the loopback packet are retained in the FIFO. The last 8 bytes can be read through the FIFO register which will advance through the FIFO to allow reading the receive packet sequentially.

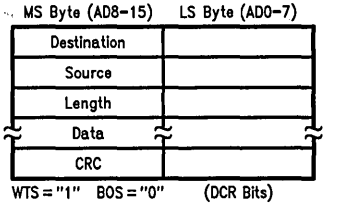
Destination Address	= (6 Bytes) Station Physical Address
Source Address	|
Length	2 Bytes
Data	= 46 to 1500 Bytes
CRC	Appended by SNIC if CRC = "0" in TCR

When in word-wide mode with Byte Order Select set, the loopback packet must be assembled in the even byte location as shown below. (The loopback only operated with byte wide transfers.)



TL/F/10469–50

When in word-wide mode with Byte Order Select low, the following format must be used for the loopback packet.



TL/F/10469–51

Note: When using loopback in word mode 2n bytes must be programmed in TBCR0, 1. Where n = actual number of bytes assembled in even or odd location.

12.0 Loopback Diagnostics (Continued)

To initiate a loopback the user first assembles the loopback packet then selects the type of loopback using the Transmit Configuration register bits LB0, LB1. The transmit configuration register must also be set to enable or disable CRC generation during transmission. The user then issues a normal transmit command to send the packet. During loopback the receiver checks for an address match and if CRC bit in the TCR is set, the receiver will also check the CRC. The last 8 bytes of the loopback packet are buffered and can read out of the FIFO using FIFO read port.

Loopback Modes

MODE 1: Loopback through the NIC Module (LB1 = 0, LB0 = 1): If this loopback is used, the NIC Modules's serializer is connected to the deserializer.

MODE 2: Loopback through the ENDEC Module (LB1 = 1, LB0 = 0): If the loopback is to be performed through the SNI, the SNIC provides a control (LPBK) that forces the ENDEC module to loopback all signals.

MODE 3: Loopback to Coax (LB1 = 1, LB0 = 1). Packets can be transmitted to the coax in loopback mode to check all of the transmit and receive paths and the coax itself.

Note: Collision and Carrier Sense can be generated by the ENDEC module and are masked by the NIC module. It is not possible to go directly between the loopback modes, it is necessary to return to normal operation (00H) when changing modes.

Note: The FIFO may only be read during Loopback. Reading the FIFO at any other time will cause the SNIC to malfunction.

Reading the Loopback Packet

The last 8 bytes of a received packet can be examined by 8 consecutive reads of the FIFO register. The FIFO pointer is incremented after the rising edge of the CPU's read strobe by internally synchronizing and advancing the pointer. This may take up to four bus clock cycles, if the pointer has not been incremented by the time the CPU reads the FIFO register again, the SNIC will insert wait states.

Note: The FIFO may only be read during Loopback. Reading the FIFO at any other time will cause the SNIC to malfunction.

Alignment of the Received Packet in the FIFO

Reception of the packet in the FIFO begins at location zero, after the FIFO pointer reaches the last location in the FIFO, the pointer wraps to the top of the FIFO overwriting the previously received data. This process continued until the last byte is received. The SNIC then appends the received byte count in the next two locations of the FIFO. The contents of the Upper Byte Count are also copied to the next FIFO location. The number of bytes used in the loopback packet determined the alignment of the packet in the FIFO.

The alignment for a 64-byte packet is shown below.

FIFO Location	FIFO Contents	
0	Lower Byte Count	First Byte Read
1	Upper Byte Count	Second Byte Read
2	Upper Byte Count	•
3	Last Byte	•
4	CRC1	•
5	CRC2	•
6	CRC3	•
7	CRC4	Last Byte Read

For the following alignment in the FIFO the packet length should be (N x 8) + 5 Bytes. Note that if the CRC bit in the TCR is set, CRC will not be appended by the transmitter. If the CRC is appended by the transmitter, the 1st four bytes, bytes N-3 to N, correspond to the CRC.

FIFO Location	FIFO Contents	
0	Byte N-4	First Byte Read
1	Byte N-3 (CRC1)	Second Byte Read
2	Byte N-2 (CRC2)	•
3	Byte N-1 (CRC3)	•
4	Byte N (CRC4)	•
5	Lower Byte Count	•
6	Upper Byte Count	Last Byte Read
7	Upper Byte Count	

LOOPBACK TESTS

Loopback capabilities are provided to allow certain tests to be performed to validate operation of the DP83901A SNIC prior to transmitting and receiving packets on a live network. Typically these tests may be performed during power up of a node. The diagnostic provides support to verify the following:

1. Verify integrity of data path. Received data is checked against transmitted data.
2. Verify CRC logic's capability to generate good CRC on transmit, verify CRC on receive (good or bad CRC).

12.0 Loopback Diagnostics (Continued)

3. Verify that the Address Recognition Logic can
   a) Recognize address match packets
   b) Reject packets that fail to match an address

LOOPBACK OPERATION IN THE SNIC

Loopback is a modified form of transmission using only half of the FIFO. This places certain restrictions on the use of loopback testing. When loopback mode is selected in the TCR, the FIFO is split. A packet should be assembled in memory with programming of TPSR and TBCR0, TBCR1 registers. When the transmit command is issued the following operations occur:

Transmitter Actions

1. Data is transferred from memory by the DMA until the FIFO is filled. For each transfer TBCR0 and TBCR1 are decremented. (Subsequent burst transfers are initiated when the number of bytes in the FIFO drops below the programmed threshold.)
2. The SNIC generates 56 bits of preamble followed by an 8-bit synch pattern.
3. Data transferred from FIFO to serializer.
4. If CRC = 1 in TCR, no CRC calculated by SNIC, the last byte transmitted is the last byte from the FIFO (Allows software CRC to be appended). If CRC = 0, SNIC calculates and appends four bytes of CRC.
5. At end of Transmission PTX bit set in ISR.

Receiver Actions

1. Wait for synch, all preamble stripped.
2. Store packet in FIFO, increment receive byte count for each incoming byte.
3. If CRC = 1 in TRC, receiver checks incoming packet for CRC errors. If CRC = 0 in TCR, receiver does not check CRC errors, CRC error bit always set in RSR (for address matching packets).
4. At end of receive, receive byte count written into FIFO, receive status register is updated. The PRX bit is typically set in the RSR even if the address does not match. If CRC errors are forced, the packet must match the address filters in order for the CRC error bit in the RSR to be set.

EXAMPLES

The following examples show what results can be expected from a properly operating SNIC during loopback. The restrictions and results of each type of loopback are listed for reference. The loopback tests are divided into two sets of tests. One to verify the data path, CRC generation and byte count through all three paths. The second set of tests uses internal loopback to verify the receiver's CRC checking and address recognition. For all of the tests the DCR was programmed to 40H.

Path	TCR	RCR	TSR	RSR	ISR
SNIC Internal	02	00	53 (Note 1)	02 (Note 2)	02 (Note 3)

Note 1: Since carrier sense and collision detect are generated in the ENDEC module. They are blocked during NIC loopback, carrier and CD heartbeat are not seen and the CRS and CDH bits are set.

Note 2: CRC errors are always indicated by receiver if CRC is appended by the transmitter.

Note 3: Only the PTX bit in the ISR is set, the PRX bit is only set if status is written to memory. In loopback this action does not occur and the PRX bit remains 0 for all loopback modes.

Note 4: All values are hex.

Path	TCR	RCR	TSR	RSR	ISR
SNIC Internal	04	00	43 (Note 1)	02	02

Note 1: CDH is set, CRS is not set since it is generated by the external encoder/decoder.

Path	TCR	RCR	TSR	RSR	ISR
SNIC External	06	00	03 (Note 1)	02	02 (Note 2)

Note 1: CDH and CRS should not be set. The TSR however, could also contain 01H, 03H, 07H and a variety of other values depending on whether collisions were encountered or the packet was deferred.

Note 2: Will contain 08H if packet is not transmittable.

Note 3: During external loopback the SNIC is now exposed to network traffic, it is therefore possible for the contents of both the Receive portion of the FIFO and the RSR to be corrupted by any other packet on the network. Thus in a live network the contents of the FIFO and RSR should not be depended on. The SNIC will still abide by the standard CSMA/CD protocol in external loopback mode. (i.e., The network will not be disturbed by the loopback packet.)

Note 4: All values are hex.

CRC AND ADDRESS RECOGNITION

The next three tests exercise the address recognition logic and CRC. These tests should be performed using internal loopback only so that the SNIC is isolated from interference from the network. These tests also require the capability to generate CRC in software.

The address recognition logic cannot be directly tested. The CRC and FAE bits in the RSR are only set if the address in the packet matches the address filters. If errors are expected to be set and they are not set, the packet has been rejected on the basis of an address mismatch. The following sequence of packets will test the address recognition logic. The DCR should be set to 40H, the TCR should be set to 03H with a software generated CRC.

Packet Contents			Results
Test	Address	CRC	RSR
Test A	Matching	Good	01 (Note 1)
Test B	Matching	Bad	02 (Note 2)
Test C	Non-Matching	Bad	01

Note 1: Status will read 21H if multicast address used.

Note 2: Status will read 22H if multicast address used.

Note 3: In test A, the RSR is set up. In test B the address is found to match since the CRC is flagged as bad. Test C proves that the address recognition logic can distinguish a bad address and does not notify the RSR of the bad CRC. The receiving CRC is proven to work in test A and test B.

Note 4: All values are hex.

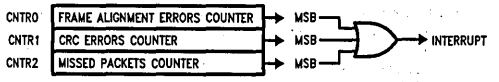
NETWORK MANAGEMENT FUNCTIONS

Network management capabilities are required for maintenance and planning of a local area network. The SNIC supports the minimum requirement for network management in hardware, the remaining requirements can be met with software counts. There are three events that software alone can not track during reception of packets: CRC errors, Frame Alignment errors, and missed packets.

12.0 Loopback Diagnostics (Continued)

Since errored packets can be rejected, the status associated with these packets is lost unless the CPU can access the Receive Status Register before the next packer arrives. In situations where another packet arrives very quickly, the CPU may have no opportunity to do this. The SNIC counts the number of packets with CRC errors and Frame Alignment errors. 8-Bit counters have been selected to reduce overhead. The counters will generate interrupts whenever their MSBs are set so that a software routine can accumulate the network statistics and reset the counter before overflow occurs. The counters are sticky so that when they reach a count of 192 (C0H) counting is halted. An additional counter is provided to count the number of packets the SNIC misses due to buffer overflow or being offline.

The structure of the counters is shown below:



TL/F/10469–19

Additional information required for network management is available in the Receive and Transmit Status Registers. Transmit status is available after each transmission for information regarding events during transmission.

Typically, the following statistics might be gathered in software:

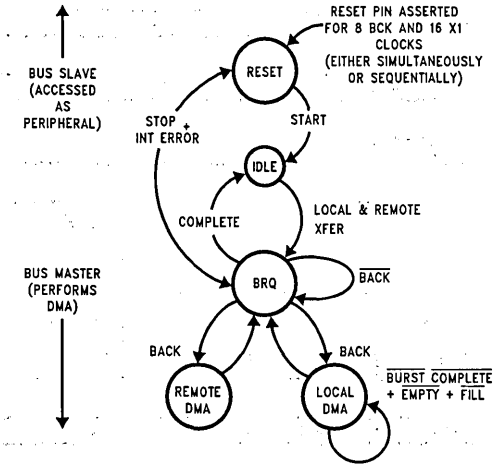
Traffic: Frames Sent OK
Frames Received OK
Multicast Frames Received
Packets Lost Due to Lack of Resources
Retries/Packet

Errors: CRC Errors
Alignment Errors
Excessive Collisions
Packet with Length Errors
Heartbeat Failure

13.0 Bus Arbitration and Timing

The SNIC operates in three possible modes:

- BUS MASTER (WHILE PERFORMING DMA)
- BUS SLAVE (WHILE BEING ACCESSED BY CPU)
- IDLE



TL/F/10469–20

Upon power-up the SNIC is in an indeterminate state. After receiving a hardware reset the SNIC is a bus slave in the Reset State, the receiver and transmitter are both disabled in this state. The reset state can be re-entered under three conditions, soft reset (Stop Command), hard reset (RESET input) or an error that shuts down the receiver of transmitter (FIFO underflow or overflow). After initialization of registers, the SNIC is issued a Start command and the SNIC enters Idle state. Until the DMA is required the SNIC remains in idle state. The idle state is exited by a request from the FIFO on the case of receiver or transmit, or from the Remote DMA in the case of Remote DMA operation. After acquiring the bus in a BREQ/BACK handshake the Remote or Local DMA transfer is completed and the SNIC re-enters the idle state.

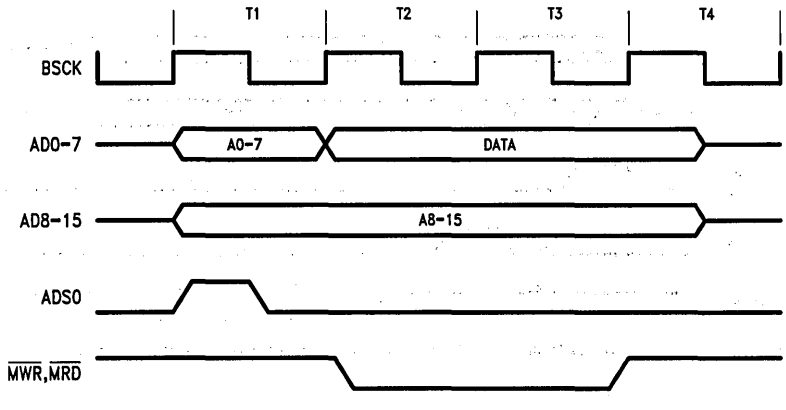
DMA TRANSFERS TIMING

The DMA can be programmed for the following types of transfers:

16-Bit Address, 8-bit Data Transfer
16-Bit Address, 16-bit Data Transfer
32-Bit Address, 8-bit Data Transfer
32-Bit Address, 16-bit Data Transfer

All DMA transfers use BSCK for timing. 16-Bit Address modes require 4 BSCK cycles as shown below:

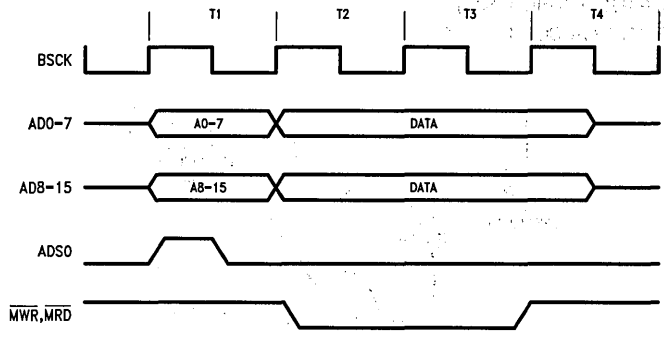
16-Bit Address, 8-Bit Data



TL/F/10469–21

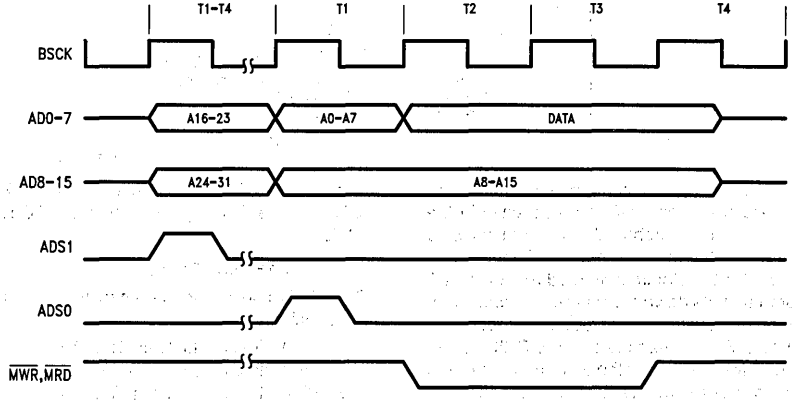
13.0 Bus Arbitration and Timing (Continued)

16-Bit Address, 16-Bit Data



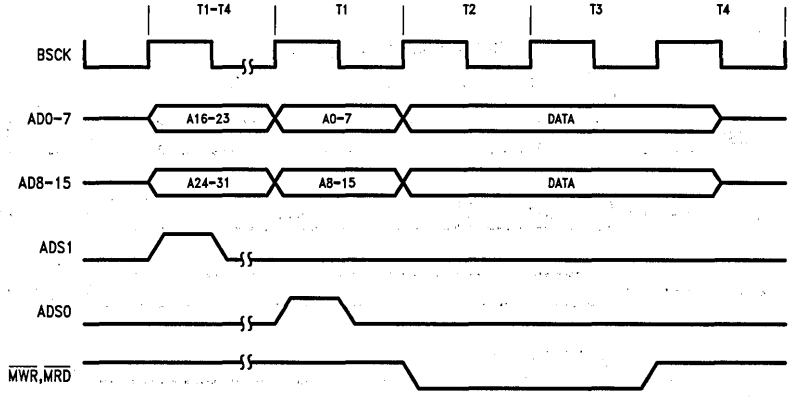
TL/F/10469-22

32-Bit Address, 8-Bit Data



TL/F/10469-23

32-Bit Address, 16-Bit Data



TL/F/10469-24

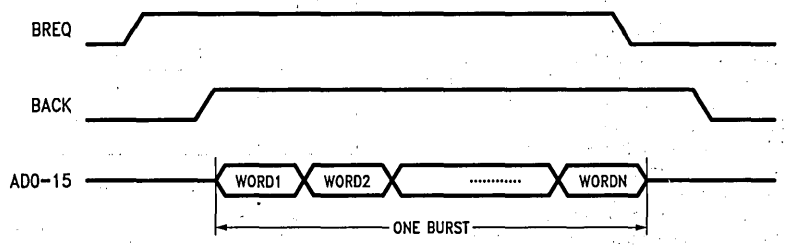
Note: In 32-bit address mode, ADS1 at TRI-STATE after the first T1–T4 states: thus, a 4.7k pull-down resistor is required for 32-bit address.

13.0 Bus Arbitration and Timing (Continued)

When in 32-bit mode four additional BSCK cycles are required per burst. The first bus cycle (T1'–T4') of each burst is used to output the upper 16-bit addresses. This 16-bit address is programmed in RSAR0 and RSAR1 and points to a 64k page of system memory. All transmitted or received packets are constrained to reside within this 64k page.

FIFO BURST CONTROL

All Local DMA transfer are burst transfers, once the DMA requests the bus and the bus is acknowledged, the DMA will transfer an exact burst of bytes programmed in the Data Configuration Register (DCR) then relinquish the bus. If there are remaining bytes in the FIFO the next burst will not be initiated until the FIFO threshold is exceeded. If desired the DMA can empty/fill the FIFO when it acquires the bus. If BACK is removed during the transfer, the burst transfer will be aborted. **(DROPPING BACK DURING A DMA CYCLE IS NOT RECOMMENDED.)**

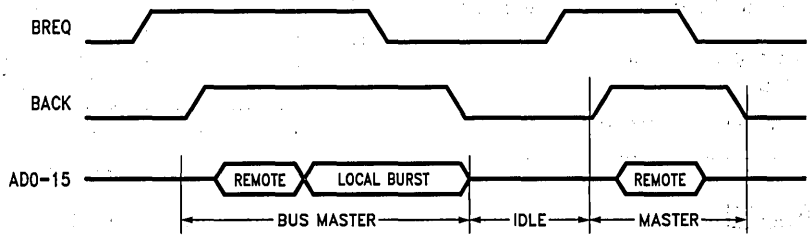


TL/F/10469–25

where N = 1, 2, 4, or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode.

INTERLEAVED LOCAL OPERATION

If a remote DMA transfer is initiated or in progress when a packet is being received or transmitted, the Remote DMA transfer will be interrupted for higher priority Local DMA transfers. When the Local DMA transfer is completed the Remote DMA will rearbitrate for the bus and continue its transfers. This is illustrated below:



TL/F/10469–26

Note that if the FIFO requires service while a remote DMA is in progress, BREQ is not dropped and the Local DMA burst is appended to the Remote Transfer. When switching from a local transfer to a remote transfer, however, BREQ is dropped and raised again. This allows the CPU or other devices to fairly contend for the bus.

FIFO AND BUS OPERATIONS

Overview

To accommodate the different rates at which data comes from (or goes to) the network and goes to (or comes from) the system memory, the SNIC contains a 16-byte FIFO for buffering data between the bus and the media. The FIFO threshold is programmable, allowing filling (or emptying) the FIFO at different rates. When the FIFO has filled to its programmed threshold, the local DMA channel transfers these bytes (or words) into local memory. It is crucial that the local DMA is given access to the bus within a minimum bus latency time; otherwise a FIFO underrun (or overrun) occurs.

To understand FIFO underruns or overruns, there are two causes which produce this condition—

1. the bus latency is so long that the FIFO has filled (or emptied) from the network before the local DMA has serviced the FIFO.
2. the bus latency or bus data rate has slowed the throughput of the local DMA to a point where it is slower than the network data rate (10 Mb/s). This second condition is also dependent upon DMA clock and word width (byte wide or word wide).

The worst case condition ultimately limits the overall bus latency which the SNIC can tolerate.

FIFO Underrun and Transmit Enable

During transmission, if a FIFO underrun occurs, the Transmit enable (TXE) output may remain high (active). Generally, this will cause a very large packet to be transmitted onto the network. The jabber feature of the transceiver will terminate the transmission, and reset TXE.

To prevent this problem, a properly designed system will not allow FIFO underruns by giving the SNIC a bus acknowledge within time shown in the maximum bus latency curves shown and described later.

13.0 Bus Arbitration and Timing (Continued)

FIFO AT THE BEGINNING OF RECEIVE

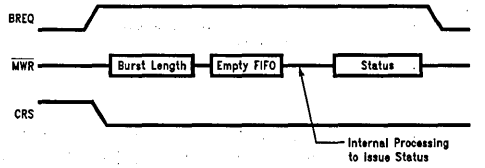
At the beginning of reception, the SNIC stores entire Address field of each incoming packet in the FIFO to determine whether the packet matches its Physical Address Register or maps to one of its Multicast Registers. This causes the FIFO to accumulate 8 bytes. Furthermore, there are some synchronization delays in the DMA PLA. Thus, the actual time that BREQ is asserted from the time the Start of Frame Delimiter (SFD) is detected is 7.8 μs. This operation affects the bus latencies at 2 byte and 4 byte thresholds during the first receive BREQ since the FIFO must be filled to 8 bytes (or 4 words) before issuing a BREQ.

FIFO Operation at the End of Receive

When Carrier Sense goes low, the SNIC enters its end of packet processing sequence, emptying its FIFO and writing the status information at the beginning of the packet, *Figure 5*. The SNIC holds onto the bus for the entire sequence. The longest time BREQ may be extended occurs when a packet ends just as the SNIC performs its last FIFO burst. The SNIC, in this case, performs a programmed burst transfer followed by flushing the remaining bytes in the FIFO, and completes by writing the header information to memory. The following steps occur during this sequence.

1. SNIC issues BREQ because the FIFO threshold has been reached
2. During the burst, packet ends, resulting in BREQ extended.
3. SNIC flushes remaining bytes from FIFO
4. SNIC performs internal processing to prepare for writing the header.
5. SNIC writes 4-byte (2-word) header
6. SNIC deasserts BREQ

End of Packet Processing



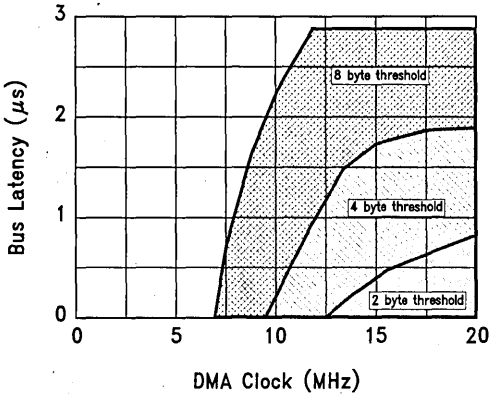
TL/F/10469–53

End of Packet Processing (EOPP) times for 10 MHz and 20 MHz have been tabulated below.

End of Packet Processing Times for Various FIFO Thresholds, Bus Clocks and Transfer Modes

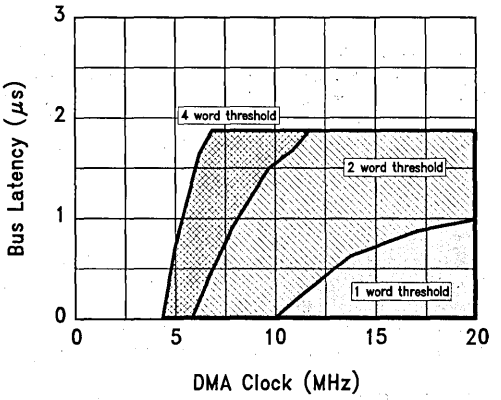
Mode	Threshold	Bus Clock	EOPP
Byte	2 bytes	10 MHz	7.0 μs
	4 bytes		8.6 μs
	8 bytes		11.0 μs
Byte	2 bytes	20 MHz	3.6 μs
	4 bytes		4.2 μs
	8 bytes		5.0 μs
Word	2 bytes	10 MHz	5.4 μs
	4 bytes		6.2 μs
	8 bytes		7.4 μs
Word	2 bytes	20 MHz	3.0 μs
	4 bytes		3.2 μs
	8 bytes		3.6 μs

Maximum Bus Latency for Byte Mode



TL/F/10469–54

Maximum Bus Latency for Word Mode



TL/F/10469–55

13.0 Bus Arbitration and Timing (Continued)

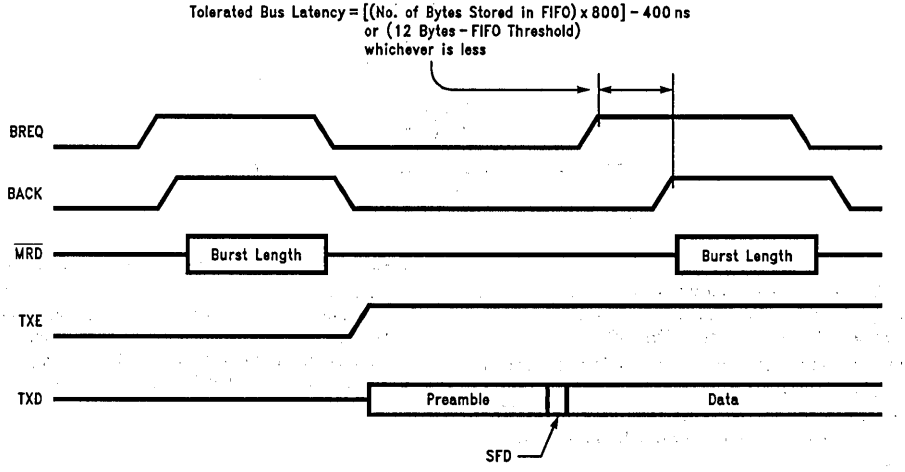
Threshold Detection (Bus Latency)

To assure that no overwriting of data in the FIFO occurs, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO, effectively shortening the FIFO to 13 bytes. The FIFO logic also operates differently in Byte Mode and in Word Mode. In Byte Mode, a threshold is indicated when the n+1 byte has entered the FIFO; thus, with an 8 byte threshold, the SNIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n+2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to 8 byte threshold), BREQ is issued when the 10th byte has entered the FIFO. The two graphs indicate the maximum allowable bus latency for Word or Byte transfer modes.

The FIFO at the Beginning of Transmit

Before transmitting, the SNIC performs a prefetch from memory to load the FIFO. The number of bytes prefetched is the programmed FIFO threshold. The next BREQ is not issued until after the SNIC actually begins transmitting data, i.e., after SFD. The Transmit Prefetch diagram illustrates this process.

Transmit Prefetch Timing



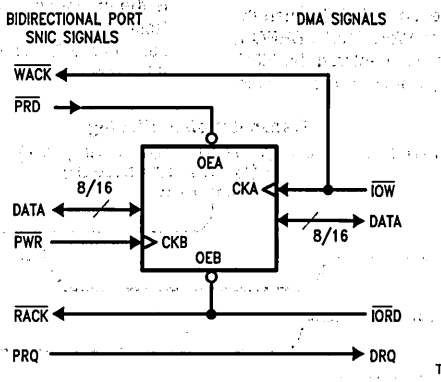
TL/F/10469–56

13.0 Bus Arbitration and Timing (Continued)

REMOTE DMA-BIDIRECTIONAL PORT CONTROL

The Remote DMA transfers data between the local buffer memory and a bidirectional port (memory to I/O transfer). This transfer is arbited on a byte by byte basis versus the burst transfer used for Local DMA transfers. This bidirectional port is also read/written by the host. All transfers through this port are asynchronous. At any one time transfers are limited to one direction, either from the port to local buffer memory (Remote Write) or from local buffer memory to the port (Remote Read).

Bus Handshake Signals for Remote DMA Transfers



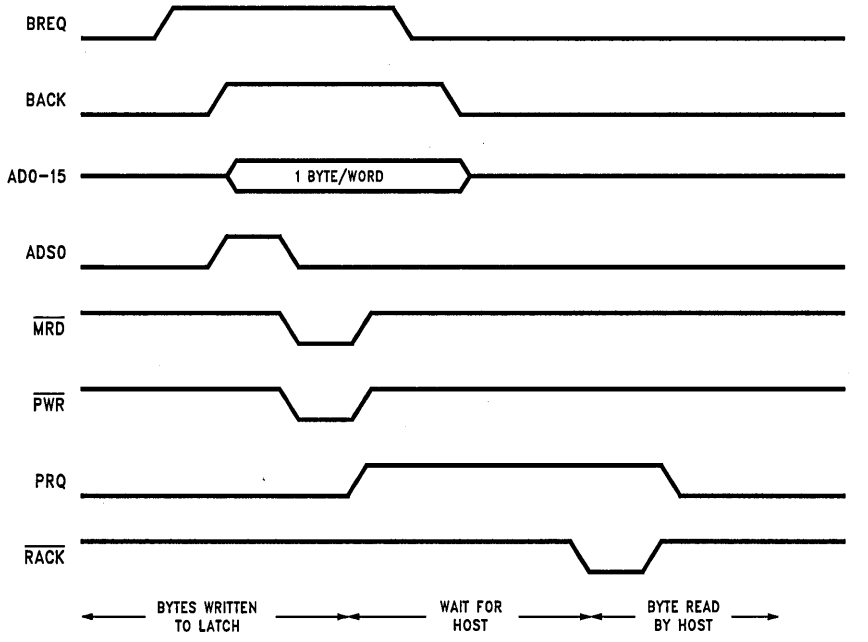
TL/F/10469–27

REMOTE READ TIMING

1. The DMA reads byte/word from local buffer memory and writes byte/word into latch, increments the DMA address and decrements the byte count (RBCR0, 1).
2. A Request Line (PRQ) is asserted to inform the system that a byte is available.
3. The system reads the port, the read strobe (RACK) is used as an acknowledge by the Remote DMA and it goes back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.

Note that in order for the Remote DMA to transfer a byte from memory to the latch, it must arbitrate access to the local bus via a BREQ, BACK handshake. After each byte or word is transferred to the latch, BREQ is dropped. If a Local DMA is in progress, the Remote DMA is held off until the local DMA is complete.



TL/F/10469–28

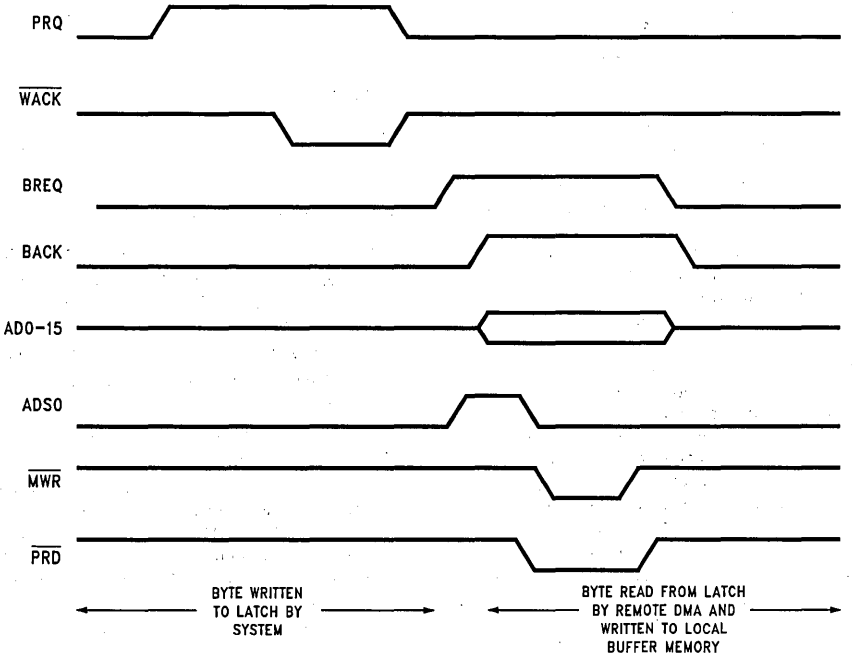
13.0 Bus Arbitration and Timing (Continued)

REMOTE WRITE TIMING

A Remote Write operation transfers data from the I/O port to the local buffer RAM. The SNIC initiates a transfer by requesting a byte/word via the PRQ. The system transfers a byte-word to the latch via $\overline{IOW}$, this write strobe is detected by the SNIC and PRQ is removed. By removing the PRQ, the Remote DMA holds off further transfers into the latch until the current byte/word has been transferred from the latch, PRQ is reasserted and the next transfer can begin.

1. SNIC asserts PRQ. System writes byte/word into latch. SNIC removes PRQ.
2. Remote DMA reads contents of port and writes byte/word to local buffer memory, increments address and decrements byte count (RBCR0, 1).
3. Go back to step 1.

   Steps 1–3 are repeated until the remote DMA is complete.



TL/F/10469–29

REMOTE DMA WRITE

Setting PRQ Using the Remote Read

Under certain conditions the SNIC's bus state machine may issue $\overline{MWR}$ and $\overline{PRD}$ before PRQ for the first DMA transfer of a Remote Write Command. If this occurs this could cause data corruption, or cause the remote DMA count to be different from the main CPU count causing the system to "lock up."

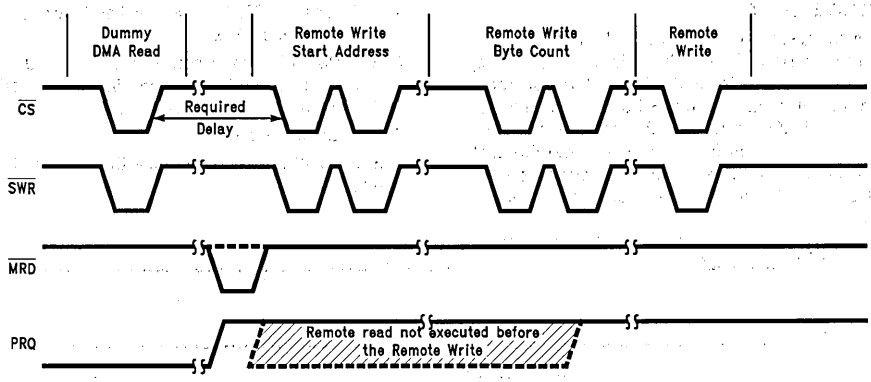
To prevent this condition when implementing a Remote DMA Write, the Remote DMA Write command should first be preceded by a Remote DMA Read command to insure that the PRQ signal is asserted before the SNIC starts its port read cycle. The reason for this is that the state machine that asserts PRQ runs independently of the state machine that controls the DMA signals. The DMA machine assumes that PRQ is asserted, but actually may not be. To remedy this situation, a single Remote Read cycle should be inserted before the actual DMA Write Command is given. This will ensure that PRQ is asserted when the Remote DMA Write is subsequently executed. This single Remote Read cycle is called a "dummy Remote Read." In order for the dummy Remote Read cycle to operate correctly, the Start Address should be programmed to a known, safe location in the buffer memory space, and the Remote Byte Count should be programmed to a value greater than 1. This will ensure that the master read cycle is performed safely, eliminating the possibility of data corruption.

Remote Write with High Speed Buses

When implementing the Remote DMA Write solution with high speed buses and CPU's, timing problems may cause the system to hang. Therefore additional considerations are required.

The problem occurs when the system can execute the dummy Remote Read and then start the Remote Write before the SNIC has had a chance to execute the Remote Read. If this happens the PRQ signal will not get set, and the Remote Byte Count and Remote Start Address for the Remote

13.0 Bus Arbitration and Timing (Continued)



TL/F/10469–57

Note: The dashed lines indicate incorrect timing as described in the text.

FIGURE 9. Timing Diagram for Dummy Remote Read

Write operation could be corrupted. This is shown by the hatched waveforms in the timing diagram of *Figure 9*. The execution of the Remote Read can be delayed by the local DMA operations (particularly during end-of-packet processing).

To ensure the dummy Remote Read does execute, a delay must be inserted between writing the Remote Read Command, and starting to write the Remote Write State Address. (This time is designated in *Figure 9* by the delay arrows.) The recommended method to avoid this problem is after the Remote Read command is given, to poll both bytes of the Current Remote DMA Address Registers. When the address has incremented PRQ has been set. Software should recognize this and then start the Remote Write.

An additional caution for high speed systems is that the polling must follow guidelines specified in Time Between Chip Select section. That is, there must be at least 4 bus clocks between chip selects (for example when BSCK = 20 MHz, then this time should be 200 ns).

The general flow for executing a Remote Write is:

1. Set Remote Byte Count to a value >1 and Remote Start Address to unused RAM (one location before the transmit start address is usually a safe location).
2. Issue the "dummy" Remote Read command.
3. Read the Current Remote DMA Address (CRDA) (both bytes).
4. Compare to previous CRDA value if different go to 6.
5. Delay and jump to 3.
6. Set up for the Remote Write command, by setting the Remote Byte Count and the Remote Start Address (note that if Remote Byte count in step 1 can be set to the transmit byte count plus one, and the Remote Start Address to one less, these will now be incremented to the correct values.)
7. Issue the Remote Write command.

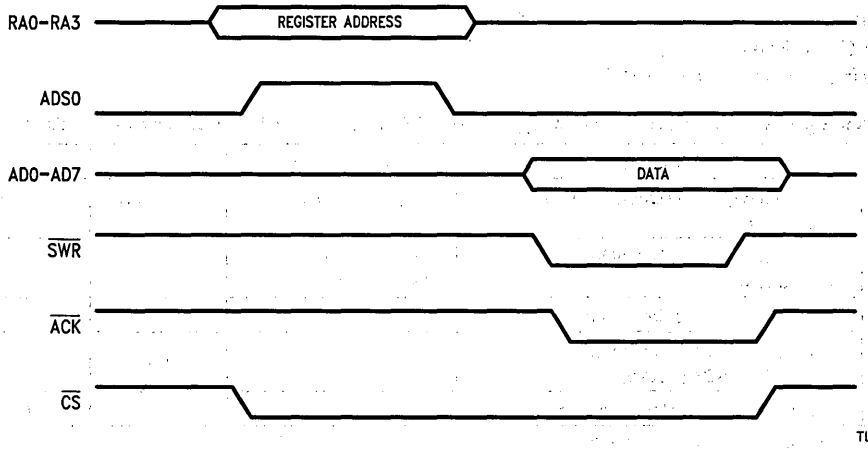
13.0 Bus Arbitration and Timing (Continued)

SLAVE MODE TIMING

When $\overline{CS}$ is low, the SNIC becomes a bus slave. The CPU can then read or write any internal registers. All register access is byte wide. The timing for register access is shown below. The host CPU accesses internal registers with four address lines, RA0–RA3, $\overline{SRD}$ and $\overline{SWR}$ strobes.

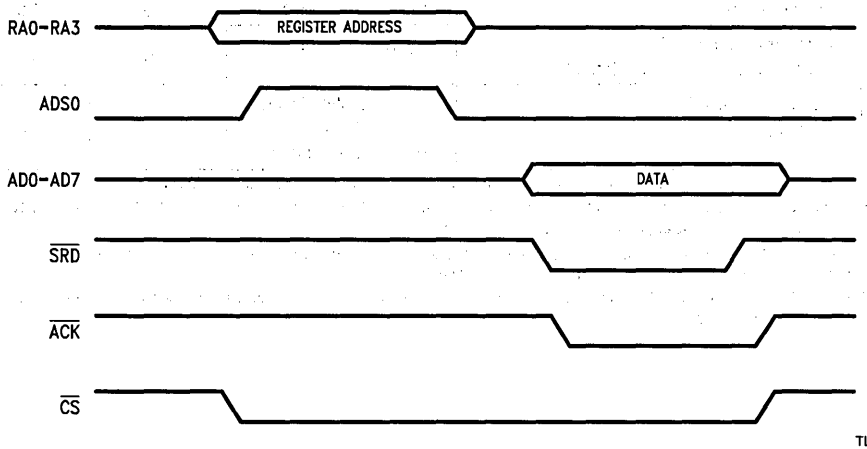
ADS0 is used to latch the address when interfacing to a multiplexed, address data bus. Since the SNIC may be a local bus master when the host CPU attempts to read or write to the controller, an $\overline{ACK}$ line is used to hold off the CPU until the SNIC leaves master mode. Some number of BSCK cycles is also required to allow the SNIC to synchronize to the read or write cycles.

Write to Register



TL/F/10469–30

Read from Register

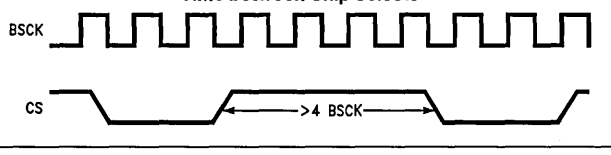


TL/F/10469–31

TIME BETWEEN CHIP SELECTS

The SNIC requires that successive chip selects be no closer than 4 bus clocks (BSCK) together. If the condition is violated, the SNIC may glitch $\overline{ACK}$. CPUs that operate from pipelined instructions (i.e., 386) or have a cache (i.e., 486) can execute consecutive I/O cycles very quickly. The solution is to delay the execution of consecutive I/O cycles by either breaking the pipeline or forcing the CPU to access outside its cache.

Time between Chip Selects



TL/F/10469–58

14.0 Preliminary Electrical Characteristics

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (PD)	800 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C
ESD Rating (R_{ZAP} = 1.5k, C_{ZAP} = 120 pF)	1.5 kV

Note: *Absolute Maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

Note: *All specifications in this datasheet are valid only if the mandatory isolation is employed and all differential signals are taken to exist at the AUI side of the isolation.*

Preliminary DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage (Notes 1, 4)	I_{OH} = −20 μA I_{OH} = −2.0 mA	V_{CC} −0.1 3.5		V V
V_{OL}	Minimum Low Level Output Voltage (Notes 1, 4)	I_{OL} = 20 μA I_{OL} = 2.0 mA		0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage (Note 2)		2.0		V
V_{IH2}	Minimum High Level Input Voltage For RACK WACK (Note 2)		2.7		V
V_{IL}	Minimum Low Level Input Voltage (Note 2)			0.8	V
V_{IL2}	Minimum Low Level Input Voltage For RACK, WACK (Note 2)			0.6	V
I_{IN}	Input Current	V_I = V_{CC} or GND	−1.0	+1.0	μA
I_{OZ}	Minimum TRI-STATE Output Leakage Current (Note 5)	V_{OUT} = V_{CC} or GND	−10	+10	μA
I_{CC}	Average Supply Current (Note 3)	X1 = 20 MHz Clock I_{OUT} = 0 μA V_{IN} = V_{CC} or GND		110	mA

Note 1: These levels are tested dynamically using a limited amount of functional test patterns, please refer to AC test load.

Note 2: Limited functional test patterns are performed at these input levels. The majority of functional tests are performed at levels of 0V and 3V.

Note 3: This is measured with a 0.1 μF bypass capacitor between V_{CC} and GND.

Note 4: The low drive CMOS compatible V_{OH} and V_{OL} limits are not tested directly. Detailed device characterization validates that this specification can be guaranteed by testing the high drive TTL compatible V_{OL} and V_{OH} specification.

Note 5: RA0–RA3, $\overline{PRD}$, $\overline{WACK}$, BREQ and INT pins are used as outputs in test mode and as a result are tested as if they are TRI-STATE input/outputs. For these pins the input leakage specification is I_{OZ}.

14.0 Preliminary Electrical Characteristics (Continued)

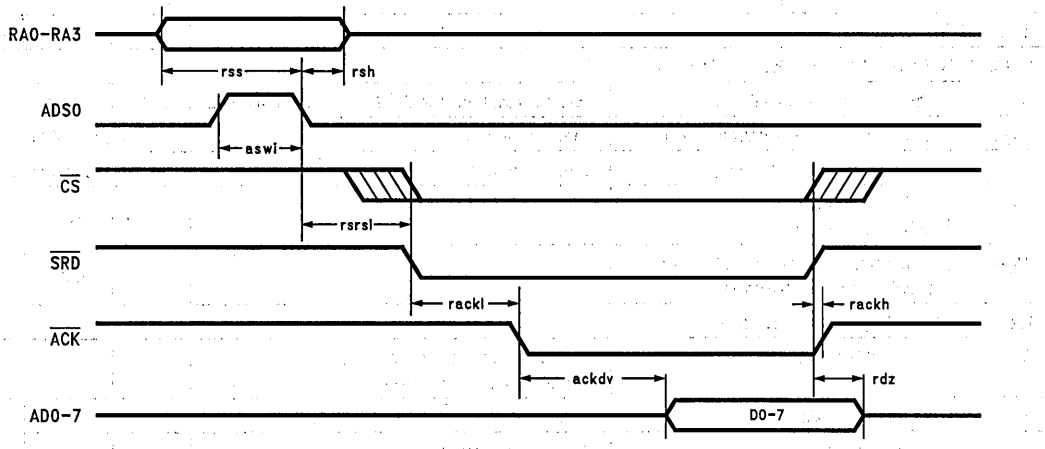
Preliminary DC Specifications T_A = 0°C to 70°C, V = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
DIFFERENTIAL PINS (TX±, RX±, and CD±)					
V_{OD}	Diff. Output Voltage (TX±)	78Ω Termination, and 270Ωs from each to GND	±550	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±)	Same as Above	Typical: 40 mV		
V_U	Undershoot Voltage (TX±)	Same as Above	Typical: 80 mV		
V_{DS}	Diff. Squelch Threshold (RX± and CD±)		−175	−300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± and CD±) (Note 1)		0	5.25	V
OSCILLATOR PINS (X1 and GND/X2)					
V_{IH}	X1 Input High Voltage	X1 is Connected to an Oscillator and GND/X2 is Grounded	2.0		V
V_{IL}	X1 Input Low Voltage	Same as Above		0.8	V
I_{OSC}	X1 Input Current	GND/X2 is Grounded $V_{IN} = V_{CC}$ or GND		+3	mA

Note 1: This parameter is guaranteed by the isolation and is not tested.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary

Register Read (Latched Using ADS0)



TL/F/10469–32

Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	13		ns
aswi	Address Strobe Width In	15		ns
ackdv	Acknowledge Low to Data Valid		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 3)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns
rsrsl	Register Select to Slave Read Low, Latched RS0–3	10		ns

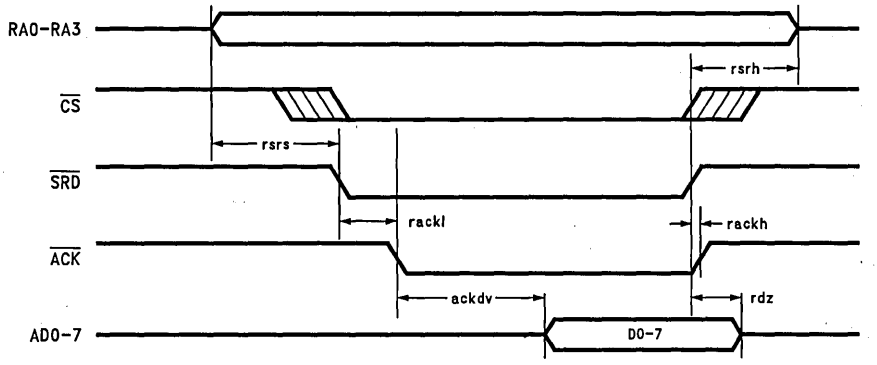
Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SRD}$ are low and the SNIC has synchronized to the register access. The SNIC will insert an integral number of Bus Clock cycles until it is synchronized. In Dual Bus systems additional cycles will be used for a local or remote DMA to complete. Wait states must be issued to the CPU until $\overline{ACK}$ is asserted low.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SRD}$. If $\overline{CS}$ is asserted after $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$. $\overline{CS}$ can be de-asserted concurrently with $\overline{SRD}$ or after $\overline{SRD}$ is de-asserted.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Register Read (Non-Latched, ADS0 = 1)



TL/F/10469–33

Symbol	Parameter	Min	Max	Units
rsrs	Register Select to Read Setup (Notes 1, 3)	10		ns
rsrh	Register Select Hold from Read	0		ns
ackdv	$\overline{ACK}$ Low to Valid Data		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 2)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Note 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns

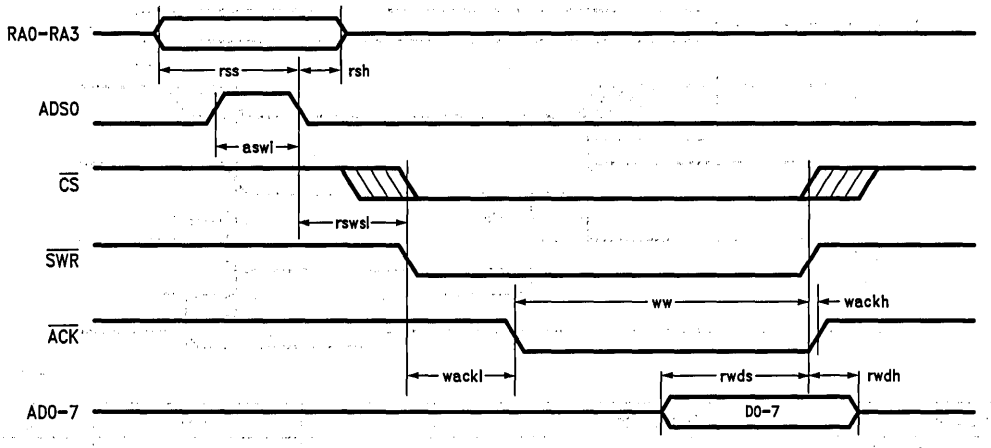
Note 1: rsrs includes flow-through time of latch.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

Note 3: $\overline{CS}$ may be asserted before of after RA0–3, and $\overline{SRD}$, since address decode begins when $\overline{ACK}$ is asserted. If $\overline{CS}$ is asserted after RA0–3, and $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Register Write (Latched Using ADS0)



TL/F/10469–34

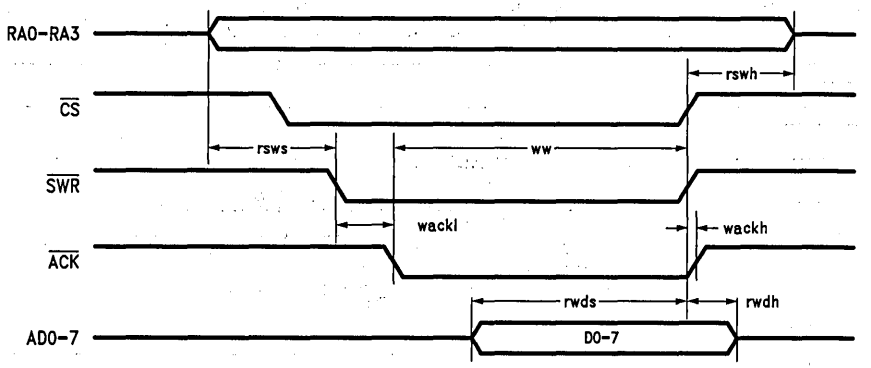
Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	17		ns
aswi	Address Strobe Width In	15		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
ww	Write Strobe Width from $\overline{ACK}$	50		ns
wackh	Write Strobe High to $\overline{ACK}$ High		30	ns
wackl	Write Low to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rswsl	Register Select to Write Strobe Low	10		ns

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the SNIC has synchronized to the register access. In Dual Bus Systems additional cycles will be used for a local DMA or Remote DMA to complete.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SWR}$. If $\overline{CS}$ is asserted after $\overline{SWR}$, wackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Register Write (Non-Latched, ADS0 = 1)



TL/F/10469-35

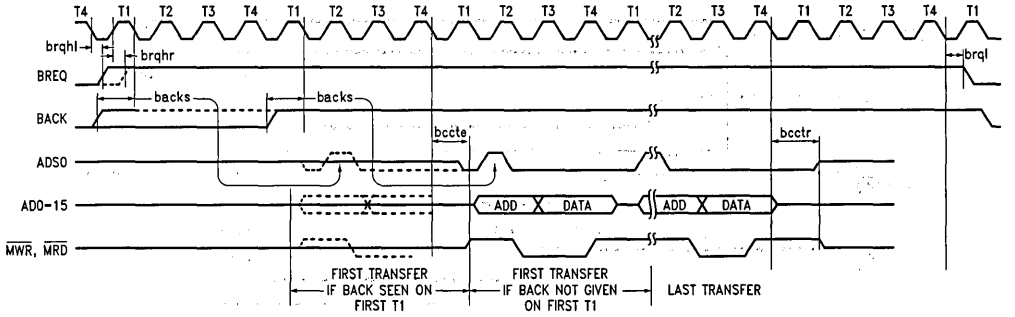
Symbol	Parameter	Min	Max	Units
rsws	Register Select to Write Setup (Note 1)	15		ns
rswh	Register Select Hold from Write	0		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
wackl	Write Low to $\overline{ACK}$ Low (Note 2)		n*bcyc + 30	ns
wackh	Write High to $\overline{ACK}$ High		30	ns
ww	Write Width from $\overline{ACK}$	50		ns

Note 1: Assumes ADS0 is high when RA0–3 changing.

Note 2: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the SNIC has synchronized to the register access. In Dual Bus systems additional cycles will be used for a local DMA or remote DMA to complete.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

DMA Control, Bus Arbitration



TL/F/10469–36

Symbol	Parameter	Min	Max	Units
brqhl	Bus Clock to Bus Request High for Local DMA		50	ns
brqhr	Bus Clock to Bus Request High for Remote DMA		45	ns
brql	Bus Request Low from Bus Clock		60	ns
backs	Acknowledge Setup to Bus Clock (Note 1)	2		ns
bccte	Bus Clock to Control Enable		60	ns
bcctr	Bus Clock to Control Release (Notes 2, 3)		70	ns

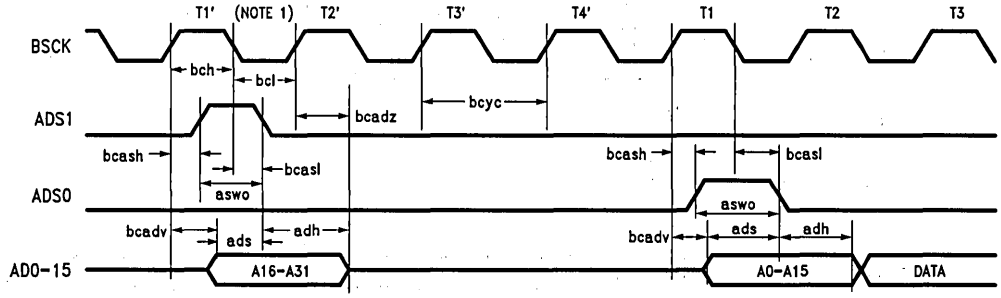
Note 1: BACK must be setup before T1 after BREQ is asserted. Missed setup will slip the beginning of the DMA by four bus clocks. The Bus Latency will influence the allowable FIFO threshold and transfer mode (empty/fill vs exact burst transfer).

Note 2: During remote DMA transfers only, a single bus transfer is performed. During local DMA operations burst mode transfers are performed.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

DMA Address Generation



TL/F/10469-37

Symbol	Parameter	Min	Max	Units
bcyc	Bus Clock Cycle Time (Note 2)	50	125	ns
bch	Bus Clock High Time	20		ns
bcl	Bus Clock Low Time	20		ns
bcash	Bus Clock to Address Strobe High		34	ns
bcasl	Bus Clock to Address Strobe Low		44	ns
aswo	Address Strobe Width Out	bch		ns
bcadv	Bus Clock to Address Valid		45	ns
bcadz	Bus Clock to Address TRI-STATE (Note 3)	15	55	ns
ads	Address Setup to ADS0/1 Low	bch − 15		ns
adh	Address Hold from ADS0/1 Low	bcl − 5		ns

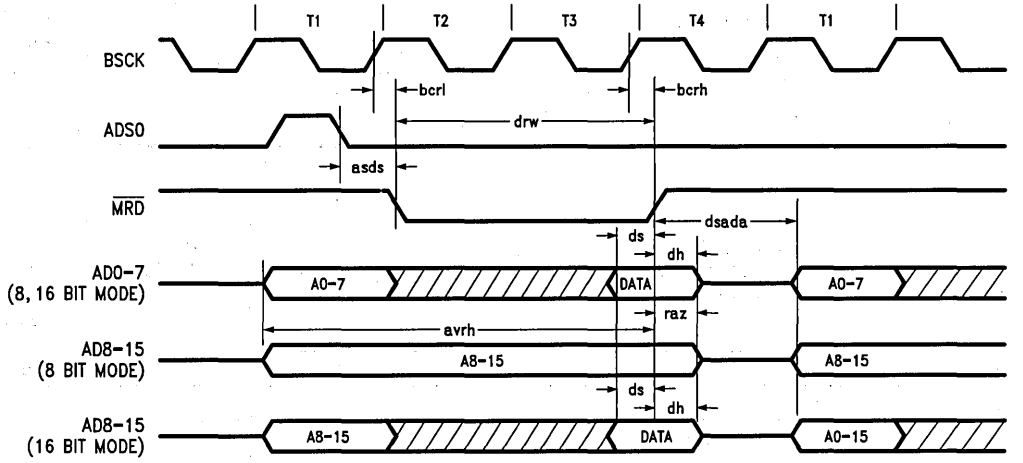
Note 1: Cycles T1', T1', T3' ≤ T4' are only issued for the first transfer in a burst 32-bit mode has been selected.

Note 2: The rate of bus clock must be high enough to support transfers to/from the FIFO at a rate greater than the serial network transfers from/to the FIFO.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

DMA Memory Read



TL/F/10469–38

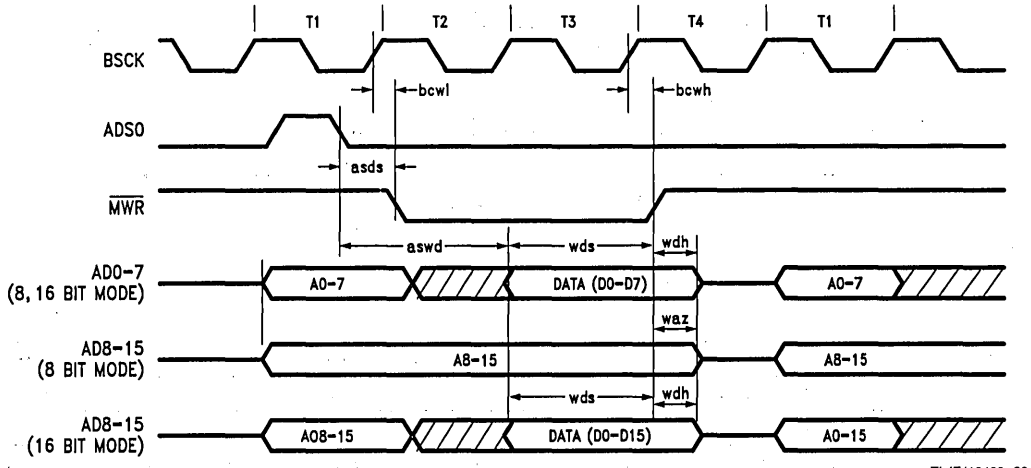
Symbol	Parameter	Min	Max	Units
bcrl	Bus Clock to Read Strobe Low		43	ns
bcrh	Bus Clock to Read Strobe High		40	ns
ds	Data Setup to Read Strobe High	22		ns
dh	Data Hold from Read Strobe High	0		ns
drw	DMA Read Strobe Width Out	2*bcyc − 15		ns
raz	Memory Read High to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
dsada	Data Strobe to Address Active	bcyc − 10		ns
avrh	Address Valid to Read Strobe High	3*bcyc − 18		ns

Note 1: During a burst A8–A15 are not TRI-STATE if byte wide transfers are selected. On the last transfer A8–A15 are TRI-STATE as shown above.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

DMA Memory Write



TL/F/10469-39

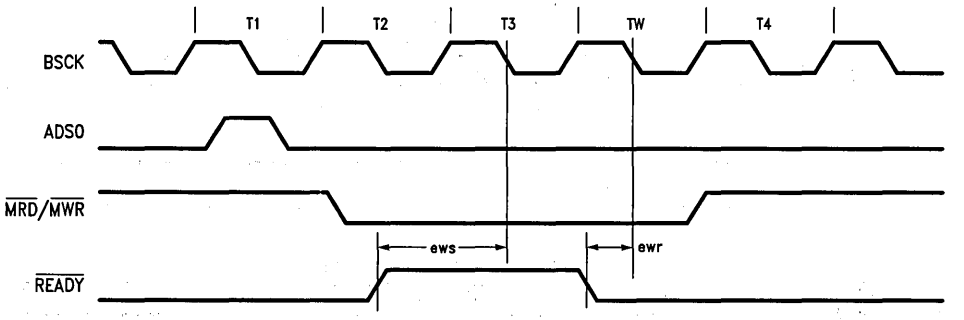
Symbol	Parameter	Min	Max	Units
bcwl	Bus Clock to Write Strobe Low		40	ns
bcwh	Bus Clock to Write Strobe High		40	ns
wds	Data Setup to $\overline{WR}$ High	2*bcyc − 30		ns
wdh	Data Hold from $\overline{WR}$ Low	bch + 7		ns
waz	Write Strobe to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
aswd	Address Strobe to Write Data Valid		bcl + 30	ns

Note 1: When using byte mode transfers A8–A15 are only TRI-STATE on the last transfer, waz timing is only valid for last transfer in a burst.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Wait State Insertion



TL/F/10469–40

Symbol	Parameter	Min	Max	Units
ews	External Wait Setup to T3 0Clock (Note 1)	10		ns
ewr	External Wait Release Time (Note 1)	15		ns

Note 1: The addition of wait states affects the count of deserialized bytes and is limited to a number of bus clock cycles depending on the bus clock and network rates. The allowable wait states are found in the table below. (Assumes 10 Mbit/sec data rate.)

BSCK (MHz)	# of Wait States	
	Byte Transfer	Word Transfer
8	0	1
10	0	1
12	1	2
14	1	2
16	1	2
18	2	3
20	2	4

Table assumes 10 MHz network clock.

The number of allowable wait states in byte mode can be calculated using:

$$\#W_{(\text{byte mode})} = \left(\frac{8\ \text{tnw}}{4.5\ \text{tbsck}} - 1\right)$$

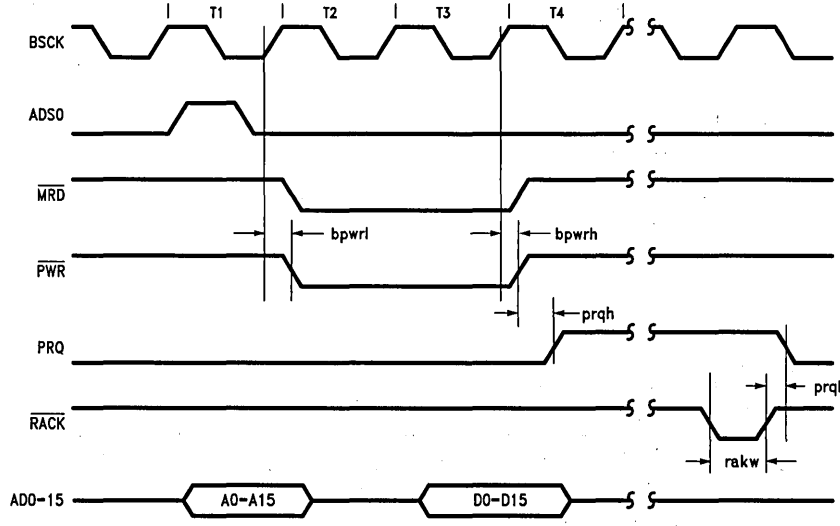
#W = Number of Wait States
tnw = Network Clock Period
tbsck = BSCK Period

The number of allowable wait states in word mode can be calculated using:

$$\#W_{(\text{word mode})} = \left(\frac{5\ \text{tnw}}{2\ \text{tbsck}} - 1\right)$$

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Remote DMA (Read, Send Command)



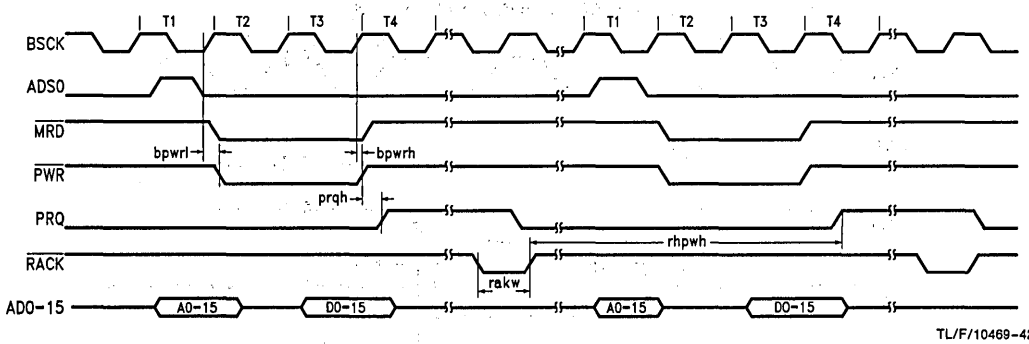
TL/F/10469–41

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		60	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generted relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Remote DMA (Read, Send Command) Recovery Time



TL/F/10469-42

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		60	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns
rhpwh	Read Acknowledge High to Next Port Write Cycle (Notes 2, 3, 4)	11		BSCK

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

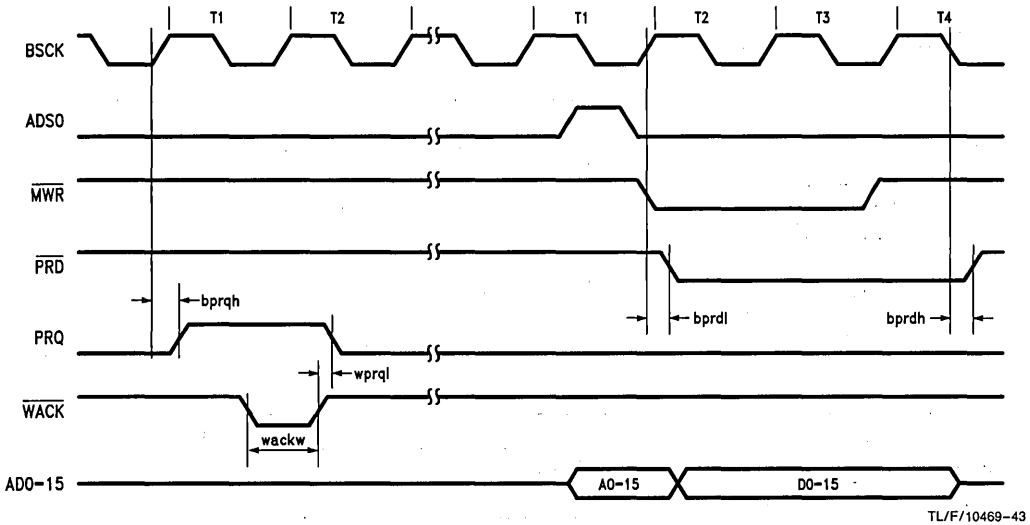
Note 2: This is not a measured value but guaranteed by design.

Note 3: $\overline{RACK}$ must be high for a minimum of 7 BSCK.

Note 4: Assumes no local DMA interleave, no $\overline{CS}$, and immediate BACK.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Remote DMA (Write Cycle)



TL/F/10469-43

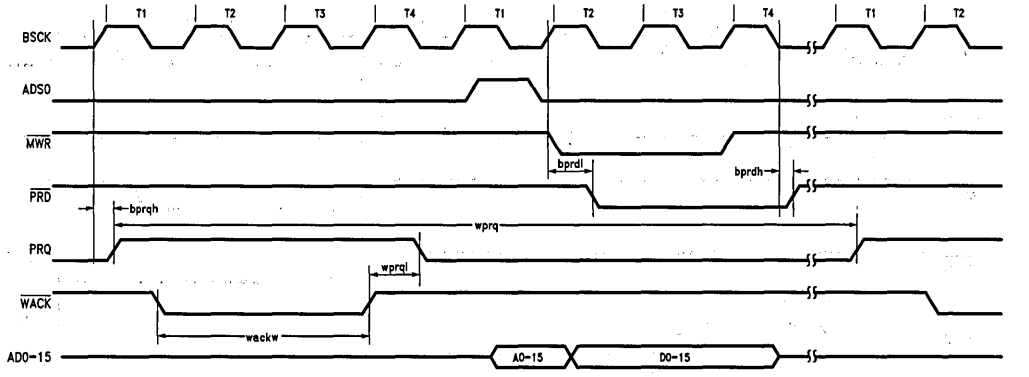
Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{WACK}$ to Port Request Low		52	ns
wackw	$\overline{WACK}$ Pulse Width	25		ns
bprdl	Bus Clock to Port Read Low (Note 2)		55	ns
bprdh	Bus Clock to Port Read High		40	ns

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Remote DMA (Write Cycle) Recovery Time



TL/F/10469–44

Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{WACK}$ to Port Request Low		50	ns
wackw	$\overline{WACK}$ Pulse Width	25		ns
bprdl	Bus Clock to Port Read Low (Note 2)		55	ns
bprdh	Bus Clock to Port Read High		40	ns
wprq	Remote Write Port Request to Port Request Time (Notes 3, 4, 5)	12		BSCK

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BSCK and whether a local DMA is pending.

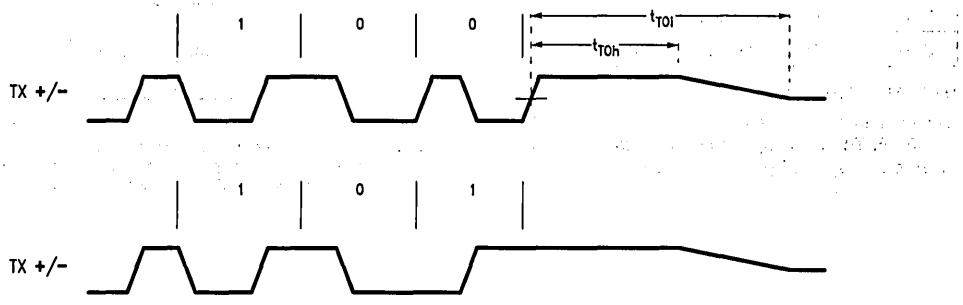
Note 3: Assuming wackw < 1 BSCK, and no local DMA interleave, no $\overline{CS}$, immediate $\overline{BACK}$, and $\overline{WACK}$ goes high before T4.

Note 4: $\overline{WACK}$ must be high for a minimum of 7 BSCK.

Note 5: This is not a measured value but guaranteed by design.

15.0 Switching Characteristics AC Specs DP83901A **Note:** All Timing is Preliminary (Continued)

Transmit Timing (End of Packet)

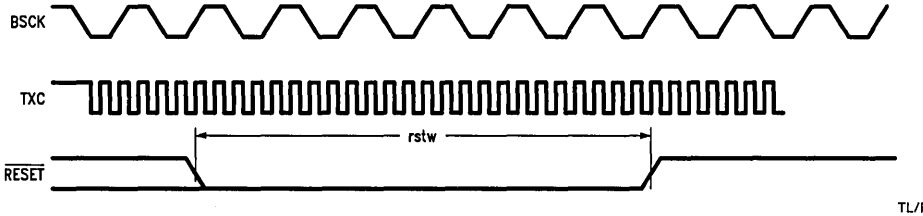


TL/F/10469–47

TRANSMIT SPECIFICATIONS (End of Packet)

Symbol	Parameter	Min	Max	Units
t_{TOH}	Transmit Output High before Idle (Half Step)	200		ns
t_{TOI}	Transmit Output Idle Time to ±40 mV(Half Step)		8000	ns

Reset Timing



TL/F/10469–45

Symbol	Parameter	Min	Max	Units
rstw	Reset Pulse Width (Note 1)	8		BSCK Cycles or TXC Cycles (Note 2)

Note 1: The RESET pulse requires the BSCK and TXC be stable. On power up, $\overline{\text{RESET}}$ should not be raised until BSCK and TXC have become stable. Several registers are affected by $\overline{\text{RESET}}$. Consult the register descriptions for details.

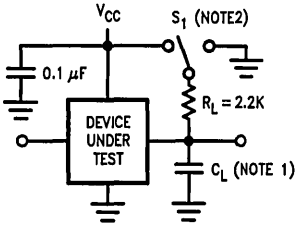
Note 2: The slower of BSCK or TXC clocks will determine the minimum time for the $\overline{\text{RESET}}$ signal to be low. TXC is X1 divided by 2.
If BSCK < TXC then $\overline{\text{RESET}}$ = 8 × BSCK
If TXC < BSCK then $\overline{\text{RESET}}$ = 8 × TXC

16.0 AC Timing Test Conditions

All specifications are valid only if the mandatory isolation is employed and all differential signals are taken to be at the AUI side of the pulse transformer.

Input Pulse Levels (TTL/CMOS)	GND to 3.0V
Input Rise and Fall Times (TTL/CMOS)	5 ns
Input and Output Reference Levles (TTL/CMOS)	1.3V
Input Pulse Levels (Diff.)	−350 mV to −1315 mV
Input and Output Reference Levels (Diff.)	50% Point of the Differential
TRI-STATE Reference Levels	Float (ΔV) ±0.5V

Output Load (See Figure Below)



TL/F/10469–48

Note 1: 50 pF, includes scope and jig capacitance

Note 2: S1 = Open for timing tests for push pull outputs.
S1 = V_{CC} for V_{OL} test.
S1 = GND for V_{OH} test.
S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.
S1 = GND for High Impedance to active high and active high to High Impedance measurements.

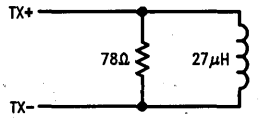
Pin Capacitance $T_A = 25°C, f = 1$ MHz

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	7	pF
C_{OUT}	Output Capacitance	7	pF

DERATING FACTOR

Output timings are measured with a purely capacitive load for 50 pF. The following correction factor can be used for other loads: $C_L \geq 50$ pF + 0.3 ns/pF.

AUI Transmit Load



TL/F/10469–49

Note: In the above diagram, the TX+ and TX− signals are taken from the AUI side of the isolation (pulse transformer). The pulse transformer used for all testing is the Pulse Engineering PE64103.

National Semiconductor

DP8390D/NS32490D Network Interface Controller

General Description

The DP8390D/NS32490D Network Interface Controller (NIC) is a microCMOS VLSI device designed to ease interfacing with CSMA/CD type local area networks including Ethernet, Thin Ethernet (Cheapernet) and StarLAN. The NIC implements all Media Access Control (MAC) layer functions for transmission and reception of packets in accordance with the IEEE 802.3 Standard. Unique dual DMA channels and an internal FIFO provide a simple yet efficient packet management design. To minimize system parts count and cost, all bus arbitration and memory support logic are integrated into the NIC.

The NIC is the heart of a three chip set that implements the complete IEEE 802.3 protocol and node electronics as shown below. The others include the DP8391 Serial Network Interface (SNI) and the DP8392 Coaxial Transceiver Interface (CTI).

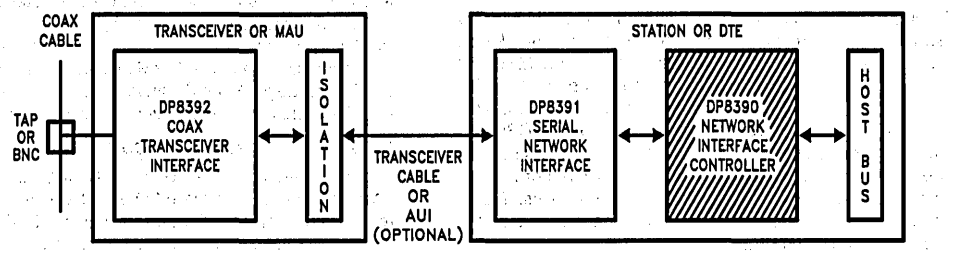
Features

- Compatible with IEEE 802.3/Ethernet II/Thin Ethernet/StarLAN
- Interfaces with 8-, 16- and 32-bit microprocessor systems
- Implements simple, versatile buffer management
- Requires single 5V supply
- Utilizes low power microCMOS process
- Includes
  - — Two 16-bit DMA channels
  - — 16-byte internal FIFO with programmable threshold
  - — Network statistics storage
- Supports physical, multicast, and broadcast address filtering
- Provides 3 levels of loopback
- Utilizes independent system and network clocks

Table of Contents

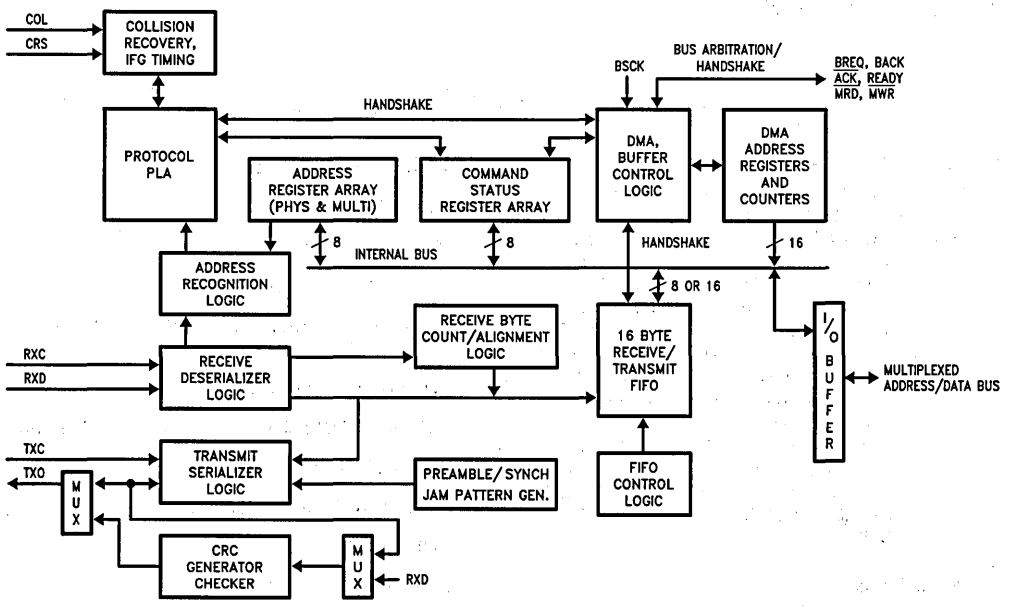
1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Thin Ethernet Local Area Network Chip Set



TL/F/8582–1

2.0 Block Diagram



TL/F/8582–2

FIGURE 1

3.0 Functional Description

(Refer to *Figure 1*)

RECEIVE DESERIALIZER

The Receive Deserializer is activated when the input signal Carrier Sense is asserted to allow incoming bits to be shifted into the shift register by the receive clock. The serial receive data is also routed to the CRC generator/checker. The Receive Deserializer includes a synch detector which detects the SFD (Start of Frame Delimiter) to establish where byte boundaries within the serial bit stream are located. After every eight receive clocks, the byte wide data is transferred to the 16-byte FIFO and the Receive Byte Count is incremented. The first six bytes after the SFD are checked for valid comparison by the Address Recognition Logic. If the Address Recognition Logic does not recognize the packet, the FIFO is cleared.

CRC GENERATOR/CHECKER

During transmission, the CRC logic generates a local CRC field for the transmitted bit sequence. The CRC encodes all fields after the synch byte. The CRC is shifted out MSB first following the last transmit byte. During reception the CRC logic generates a CRC field from the incoming packet. This local CRC is serially compared to the incoming CRC appended to the end of the packet by the transmitting node. If the local and received CRC match, a specific pattern will be generated and decoded to indicate no data errors. Transmission errors result in a different pattern and are detected, resulting in rejection of a packet.

TRANSMIT SERIALIZER

The Transmit Serializer reads parallel data from the FIFO and serializes it for transmission. The serializer is clocked by the transmit clock generated by the Serial Network Interface (DP8391). The serial data is also shifted into the CRC generator/checker. At the beginning of each transmission, the Preamble and Synch Generator append 62 bits of 1,0 preamble and a 1,1 synch pattern. After the last data byte of the packet has been serialized the 32-bit FCS field is shifted directly out of the CRC generator. In the event of a collision the Preamble and Synch generator is used to generate a 32-bit JAM pattern of all 1's

ADDRESS RECOGNITION LOGIC

The address recognition logic compares the Destination Address Field (first 6 bytes of the received packet) to the Physical address registers stored in the Address Register Array. If any one of the six bytes does not match the pre-programmed physical address, the Protocol Control Logic rejects the packet. All multicast destination addresses are filtered using a hashing technique. (See register description.) If the multicast address indexes a bit that has been set in the filter bit array of the Multicast Address Register Array the packet is accepted, otherwise it is rejected by the Protocol Control Logic. Each destination address is also checked for all 1's which is the reserved broadcast address.

FIFO AND FIFO CONTROL LOGIC

The NIC features a 16-byte FIFO. During transmission the DMA writes data into the FIFO and the Transmit Serializer reads data from the FIFO and transmits it. During reception the Receive Deserializer writes data into the FIFO and the DMA reads data from the FIFO. The FIFO control logic is used to count the number of bytes in the FIFO so that after a preset level, the DMA can begin a bus access and write/read data to/from the FIFO before a FIFO underflow//overflow occurs.

3.0 Functional Description (Continued)

Because the NIC must buffer the Address field of each incoming packet to determine whether the packet matches its Physical Address Registers or maps to one of its Multicast Registers, the first local DMA transfer does not occur until 8 bytes have accumulated in the FIFO.

To assure that there is no overwriting of data in the FIFO, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO; this effectively shortens the FIFO to 13 bytes. In addition, the FIFO logic operates differently in Byte Mode than in Word Mode. In Byte Mode, a threshold is indicated when the n + 1 byte has entered the FIFO; thus, with an 8-byte threshold, the NIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n + 2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to an 8-byte threshold), BREQ is issued when the 10th byte has entered the FIFO.

PROTOCOL PLA

The protocol PLA is responsible for implementing the IEEE 802.3 protocol, including collision recovery with random backoff. The Protocol PLA also formats packets during transmission and strips preamble and synch during reception.

DMA AND BUFFER CONTROL LOGIC

The DMA and Buffer Control Logic is used to control two 16-bit DMA channels. During reception, the Local DMA stores packets in a receive buffer ring, located in buffer memory. During transmission the Local DMA uses programmed pointer and length registers to transfer a packet from local buffer memory to the FIFO. A second DMA channel is used as a slave DMA to transfer data between the local buffer memory and the host system. The Local DMA and Remote DMA are internally arbitrated, with the Local DMA channel having highest priority. Both DMA channels use a common external bus clock to generate all required bus timing. External arbitration is performed with a standard bus request, bus acknowledge handshake protocol.

4.0 Transmit/Receive Packet Encapsulation/Decapsulation

A standard IEEE 802.3 packet consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data, and Frame Check Sequence (FCS). The typical format is shown in *Figure 2*. The packets are Manchester encoded and decoded by the DP8391 SNI and transferred serially to the NIC using NRZ data with a clock. All fields are of fixed length except for the data field. The NIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the SNI (DP8391) to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern which consists of two consecutive 1's. The NIC does not treat the SFD pattern as a byte, it detects only the two bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the NIC: physical, multicast, and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of "0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the NIC to accept the packet. Multicast addresses begin with an MSB of "1". The DP8390D filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a 6-bit value. This 6-bit value indexes a 64-bit array that filters the value. If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

SOURCE ADDRESS

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

LENGTH FIELD

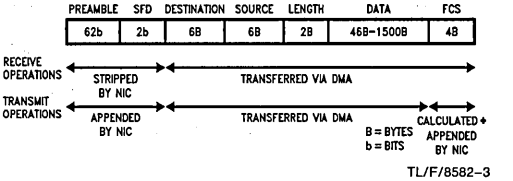
The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the NIC.

DATA FIELD

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. **The NIC does not strip or append pad bytes for short packets, or check for oversize packets.**

FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected. The AUTODIN II ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$) polynomial is used for the CRC calculations.

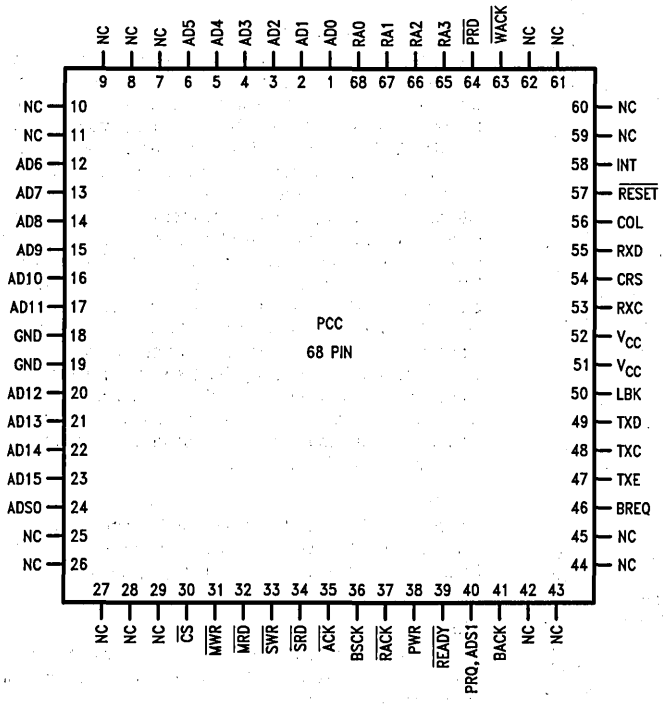


TL/F/8582–3

FIGURE 2

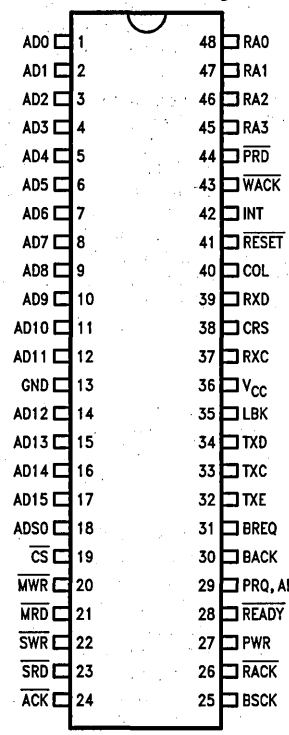
Connection Diagrams

Plastic Chip Carrier



TL/F/8582-5

Dual-In-Line Package



TL/F/8582-4

Order Number DP8390DN or DP8390DV
See NS Package Number N48A or V68A

5.0 Pin Descriptions

BUS INTERFACE PINS

Symbol	DIP Pin No	Function	Description
AD0–AD15	1–12 14–17	I/O,Z	**MULTIPLEXED ADDRESS/DATA BUS:** • Register Access, with DMA inactive, $\overline{CS}$ low and $\overline{ACK}$ returned from NIC, pins AD0–AD7 are used to read/write register data. AD8–AD15 float during I/O transfers. $\overline{SRD}$, $\overline{SWR}$ pins are used to select direction of transfer. • Bus Master with BACK input asserted. During t1 of memory cycle AD0–AD15 contain address. During t2, t3, t4 AD0–AD15 contain data (word transfer mode). During t2, t3, t4 AD0–AD7 contain data, AD8–AD15 contain address (byte transfer mode). Direction of transfer is indicated by NIC on $\overline{MWR}$, $\overline{MRD}$ lines.
ADS0	18	I/O,Z	**ADDRESS STROBE 0** • Input with DMA inactive and $\overline{CS}$ low, latches RA0–RA3 inputs on falling edge. If high, data present on RA0–RA3 will flow through latch. • Output when Bus Master, latches address bits (A0–A15) to external memory during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{CS}$	19	I	**CHIP SELECT:** Chip Select places controller in slave mode for μP access to internal registers. Must be valid through data portion of bus cycle. RA0–RA3 are used to select the internal register. $\overline{SWR}$ and $\overline{SRD}$ select direction of data transfer.
$\overline{MWR}$	20	O,Z	**MASTER WRITE STROBE:** Strobe for DMA transfers, active low during write cycles (t2, t3, tw) to buffer memory. Rising edge coincides with the presence of valid output data. TRI-STATE® until BACK asserted.
$\overline{MRD}$	21	O,Z	**MASTER READ STROBE:** Strobe for DMA transfers, active during read cycles (t2, t3, tw) to buffer memory. Input data must be valid on rising edge of $\overline{MRD}$. TRI-STATE until BACK asserted.
$\overline{SWR}$	22	I	**SLAVE WRITE STROBE:** Strobe from CPU to write an internal register selected by RA0–RA3.
$\overline{SRD}$	23	I	**SLAVE READ STROBE:** Strobe from CPU to read an internal register selected by RA0–RA3.
$\overline{ACK}$	24	O	**ACKNOWLEDGE:** Active low when NIC grants access to CPU. Used to insert WAIT states to CPU until NIC is synchronized for a register read or write operation.
RA0–RA3	45–48	I	**REGISTER ADDRESS:** These four pins are used to select a register to be read or written. The state of these inputs is ignored when the NIC is not in slave mode ($\overline{CS}$ high).
$\overline{PRD}$	44	O	**PORT READ:** Enables data from external latch onto local bus during a memory write cycle to local memory (remote write operation). This allows asynchronous transfer of data from the system memory to local memory.
$\overline{WACK}$	43	I	**WRITE ACKNOWLEDGE:** Issued from system to NIC to indicate that data has been written to the external latch. The NIC will begin a write cycle to place the data in local memory.
INT	42	O	**INTERRUPT:** Indicates that the NIC requires CPU attention after reception transmission or completion of DMA transfers. The interrupt is cleared by writing to the ISR. All interrupts are maskable.
$\overline{RESET}$	41	I	**RESET:** Reset is active low and places the NIC in a reset mode immediately, no packets are transmitted or received by the NIC until STA bit is set. Affects Command Register, Interrupt Mask Register, Data Configuration Register and Transmit Configuration Register. The NIC will execute reset within 10 BUSK cycles.
BREQ	31	O	**BUS REQUEST:** Bus Request is an active high signal used to request the bus for DMA transfers. This signal is automatically generated when the FIFO needs servicing.
BACK	30	I	**BUS ACKNOWLEDGE:** Bus Acknowledge is an active high signal indicating that the CPU has granted the bus to the NIC. If immediate bus access is desired, BREQ should be tied to BACK. **Tying BACK to V_{CC} will result in a deadlock.**
PRQ, ADS1	29	O,Z	**PORT REQUEST/ADDRESS STROBE 1** • 32-BIT MODE: If LAS is set in the Data Configuration Register, this line is programmed as ADS1. It is used to strobe addresses A16–A31 into external latches. (A16–A31 are the fixed addresses stored in RSAR0, RSAR1.) ADS1 will remain at TRI-STATE until BACK is received. • 16-BIT MODE: If LAS is not set in the Data Configuration Register, this line is programmed as PRQ and is used for Remote DMA Transfers. In this mode PRQ will be a standard logic output. **NOTE: This line will power up as TRI-STATE until the Data Configuration Register is programmed.**
$\overline{READY}$	28	I	**READY:** This pin is set high to insert wait states during a DMA transfer. The NIC will sample this signal at t3 during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{PWR}$	27	O	**PORT WRITE:** Strobe used to latch data from the NIC into external latch for transfer to host memory during Remote Read transfers. The rising edge of $\overline{PWR}$ coincides with the presence of valid data on the local bus.
$\overline{RACK}$	26	I	**READ ACKNOWLEDGE:** Indicates that the system DMA or host CPU has read the data placed in the external latch by the NIC. The NIC will begin a read cycle to update the latch.
BSCK	25	I	This clock is used to establish the period of the DMA memory cycle. Four clock cycles (t1, t2, t3, t4) are used per DMA cycle. DMA transfers can be extended by one BSCK increments using the $\overline{READY}$ input.
NETWORK INTERFACE PINS			
COL	40	I	**COLLISION DETECT:** This line becomes active when a collision has been detected on the coaxial cable. During transmission this line is monitored after preamble and synch have been transmitted. At the end of each transmission this line is monitored for CD heartbeat.
RXD	39	I	**RECEIVE DATA:** Serial NRZ data received from the ENDEC, clocked into the NIC on the rising edge of RXC.
CRS	38	I	**CARRIER SENSE:** This signal is provided by the ENDEC and indicates that carrier is present. This signal is active high.
RXC	37	I	**RECEIVE CLOCK:** Re-synchronized clock from the ENDEC used to clock data from the ENDEC into the NIC.
LBK	35	O	**LOOPBACK:** This output is set high when the NIC is programmed to perform a loopback through the StarLAN ENDEC.
TXD	34	O	**TRANSMIT DATA:** Serial NRZ Data output to the ENDEC. The data is valid on the rising edge of TXC.
TXC	33	I	**TRANSMIT CLOCK:** This clock is used to provide timing for internal operation and to shift bits out of the transmit serializer. TXC is nominally a 1 MHz clock provided by the ENDEC.
TXE	32	O	**TRANSMIT ENABLE:** This output becomes active when the first bit of the packet is valid on TXD and goes low after the last bit of the packet is clocked out of TXD. This signal connects directly to the ENDEC. This signal is active high.
POWER			
V_{CC}	36		+5V DC is required. It is suggested that a decoupling capacitor be connected between these pins. It is essential to provide a path to ground for the GND pin with the lowest possible impedance.
GND	13		

6.0 Direct Memory Access Control (DMA)

The DMA capabilities of the NIC greatly simplify use of the DP8390D in typical configurations. The local DMA channel transfers data between the FIFO and memory. On transmission, the packet is DMA'd from memory to the FIFO in bursts. Should a collision occur (up to 15 times), the packet is retransmitted with no processor intervention. On reception, packets are DMAed from the FIFO to the receive buffer ring (as explained below).

A remote DMA channel is also provided on the NIC to accomplish transfers between a buffer memory and system memory. The two DMA channels can alternatively be combined to form a single 32-bit address with 8- or 16-bit data.

DUAL DMA CONFIGURATION

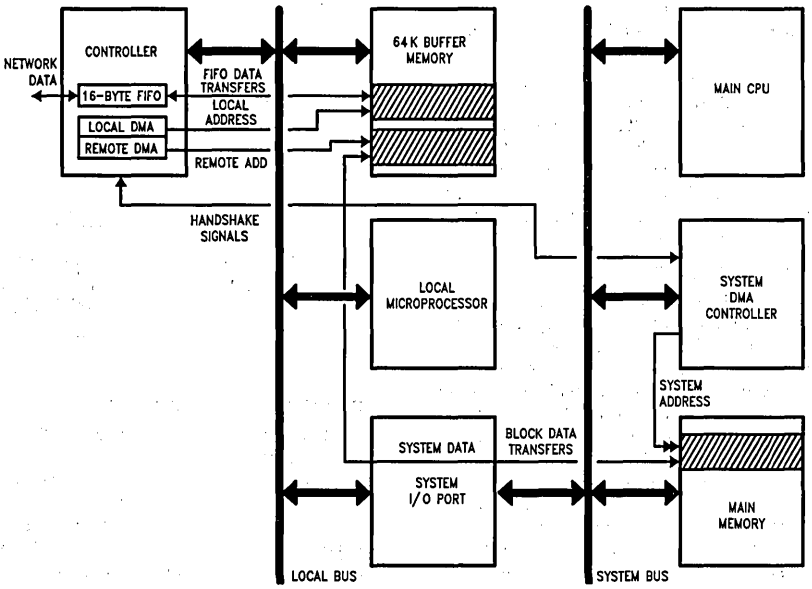
An example configuration using both the local and remote DMA channels is shown below. Network activity is isolated on a local bus, where the NIC's local DMA channel performs burst transfers between the buffer memory and the NIC's FIFO. The Remote DMA transfers data between the buffer memory and the host memory via a bidirectional I/O port. The Remote DMA provides local addressing capability and is used as a slave DMA by the host. Host side addressing must be provided by a host DMA or the CPU. The NIC allows Local and Remote DMA operations to be interleaved.

SINGLE CHANNEL DMA OPERATION

If desirable, the two DMA channels can be combined to provide a 32-bit DMA address. The upper 16 bits of the 32-bit address are static and are used to point to a 64k byte (or 32k word) page of memory where packets are to be received and transmitted.

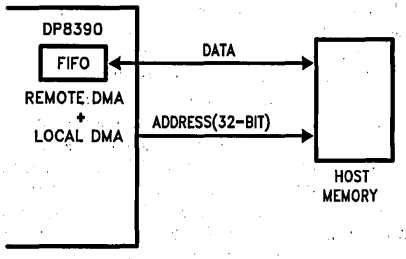
6.0 Direct Memory Access Control (DMA) (Continued)

Dual Bus System



TL/F/8582–55

32-Bit DMA Operation

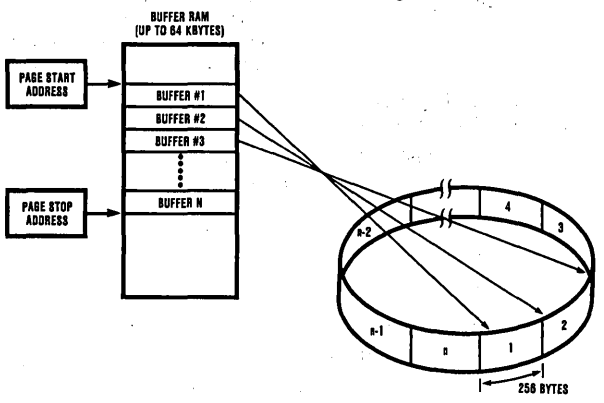


TL/F/8582–6

7.0 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256 byte (128 word) buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop Register. Ethernet packets consist of a distribution of shorter link control packets and longer data packets, the 256 byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers

NIC Receive Buffer Ring



TL/F/8582–7

7.0 Packet Reception (Continued)

for storing packets is controlled by Buffer Management Logic in the NIC. The Buffer Management Logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

At initialization, a portion of the 64k byte (or 32k word) address space is reserved for the receive buffer ring. Two eight bit registers, the Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The NIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.

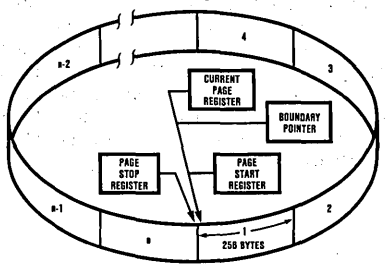
INITIALIZATION OF THE BUFFER RING

Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register (both described previously), the Current Page Register and the Boundary Pointer Register. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.

Note 1: At initialization, the Page Start Register value should be loaded into both the Current Page Register and the Boundary Pointer Register.

Note 2: The Page Start Register must not be initialized to 00H.

Receive Buffer Ring At Initialization

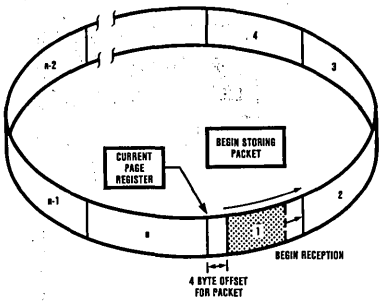


TL/F/8582–30

BEGINNING OF RECEPTION

When the first packet begins arriving the NIC begins storing the packet at the location pointed to by the Current Page Register. An offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.

Received Packet Enters Buffer Pages



TL/F/8582–31

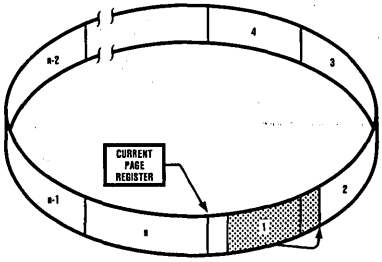
LINKING RECEIVE BUFFER PAGES

If the length of the packet exhausts the first 256 byte buffer, the DMA performs a forward link to the next buffer to store the remainder of the packet. For a maximal length packet the buffer logic will link six buffers to store the entire packet. Buffers cannot be skipped when linking, a packet will always be stored in contiguous buffers. Before the next buffer can be linked, the Buffer Management Logic performs two comparisons. The first comparison tests for equality between the DMA address of the next buffer and the contents of the Page Stop Register. If the buffer address equals the Page Stop Register, the buffer management logic will restore the DMA to the first buffer in the Receive Buffer Ring value programmed in the Page Start Address Register. The second comparison tests for equality between the DMA address of the next buffer address and the contents of the Boundary Pointer Register. If the two values are equal the reception is aborted. The Boundary Pointer Register can be used to protect against overwriting any area in the receive buffer ring that has not yet been read. When linking buffers, buffer management will never cross this pointer, effectively avoiding any overwrites. If the buffer address does not match either the Boundary Pointer or Page Stop Address, the link to the next buffer is performed.

Linking Buffers

Before the DMA can enter the next contiguous 256 byte buffer, the address is checked for equality to PSTOP and to the Boundary Pointer. If neither are reached, the DMA is allowed to use the next buffer.

Linking Receive Buffer Pages

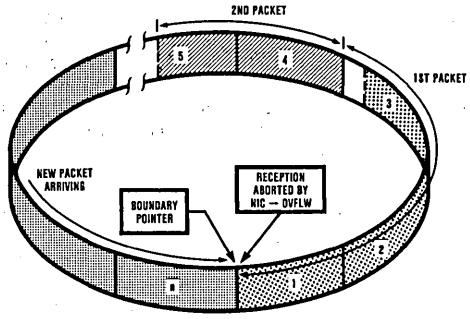


1) Check for = to PSTOP
2) Check for = to Boundary

TL/F/8582–32

7.0 Packet Reception (Continued)

Received Packet Aborted If It Hits Boundary Pointer



TL/F/8582–8

Buffer Ring Overflow

If the Buffer Ring has been filled and the DMA reaches the Boundary Pointer Address, reception of the incoming packet will be aborted by the NIC. Thus, the packets previously received and still contained in the Ring will not be destroyed.

In a heavily loaded network environment the local DMA may be disabled, preventing the NIC from buffering packets from the network. To guarantee this will not happen, a software reset must be issued during all Receive Buffer Ring overflows (indicated by the OVW bit in the Interrupt Status Register). **The following procedure is required to recover from a Receiver Buffer Ring Overflow.**

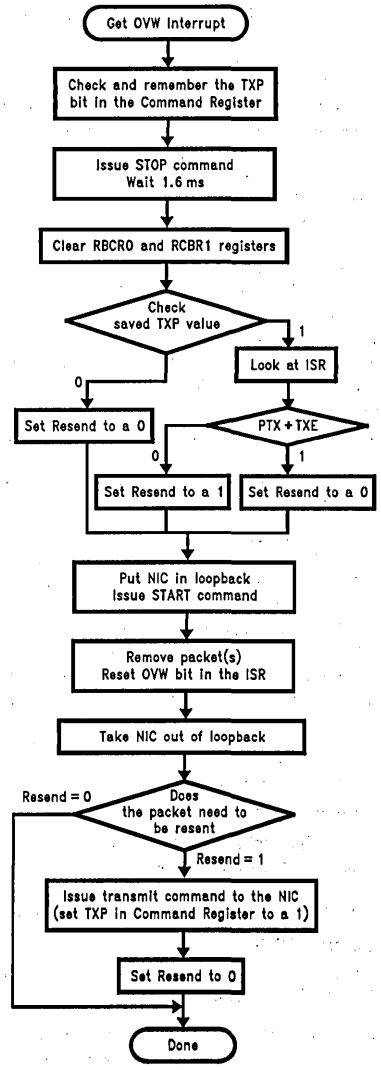
If this routine is not adhered to, the NIC may act in an unpredictable manner. It should also be noted that it is not permissible to service an overflow interrupt by continuing to empty packets from the receive buffer without implementing the prescribed overflow routine. A flow chart of the NIC's overflow routine can be found at the right.

Note: It is necessary to define a variable in the driver, which will be called "Resend".

1. Read and store the value of the TXP bit in the NIC's Command Register.
2. Issue the STOP command to the NIC. This is accomplished be setting the STP bit in the NIC's Command Register. Writing 21H to the Command Register will stop the NIC.

Note: If the STP is set when a transmission is in progress, the RST bit may not be set. In this case, the NIC is guaranteed to be reset after the longest packet time (1500 bytes = 1.2 ms). For the DP8390D (but not for the DP8390B), the NIC will be reset within 2 microseconds after the STP bit is set and Loopback mode 1 is programmed.

3. Wait for at least 1.6 ms. Since the NIC will complete any transmission or reception that is in progress, it is necessary to time out for the maximum possible duration of an Ethernet transmission or reception. By waiting 1.6 ms this is achieved with some guard band added. Previously, it was recommended that the RST bit of the Interrupt Status Register be polled to insure that the pending transmission or reception is completed. This bit is not a reliable indicator and subsequently should be ignored.
4. Clear the NIC's Remote Byte Count registers (RBCR0 and RBCR1).



TL/F/8582–95

Overflow Routine Flow Chart

5. Read the stored value of the TXP bit from step 1, above.

   If this value is a 0, set the "Resend" variable to a 0 and jump to step 6.

   If this value is a 1, read the NIC's Interrupt Status Register. If either the Packet Transmitted bit (PTX) or Transmit Error bit (TXE) is set to a 1, set the "Resend" variable to a 0 and jump to step 6. If neither of these bits is set, place a 1 in the "Resend" variable and jump to step 6.

   This step determines if there was a transmission in progress when the stop command was issued in step 2. If there was a transmission in progress, the NIC's ISR is read to determine whether or not the packet was recognized by the NIC. If neither the PTX nor TXE bit was set,

7.0 Packet Reception (Continued)

then the packet will essentially be lost and re-transmitted only after a time-out takes place in the upper level software. By determining that the packet was lost at the driver level, a transmit command can be reissued to the NIC once the overflow routine is completed (as in step 11). Also, it is possible for the NIC to defer indefinitely, when it is stopped on a busy network. Step 5 also alleviates this problem. Step 5 is essential and should not be omitted from the overflow routine, in order for the NIC to operate correctly.

6. Place the NIC in either mode 1 or mode 2 loopback. This can be accomplished by setting bits D2 and D1, of the Transmit Configuration Register, to "0,1" or "1,0", respectively.
7. Issue the START command to the NIC. This can be accomplished by writing 22H to the Command Register. This is necessary to activate the NIC's Remote DMA channel.
8. Remove one or more packets from the receive buffer ring.
9. Reset the overwrite warning (OVW, overflow) bit in the Interrupt Status Register.
10. Take the NIC out of loopback. This is done by writing the Transmit Configuration Register with the value it contains during normal operation. (Bits D2 and D1 should both be programmed to 0.)
11. If the "Resend" variable is set to a 1, reset the "Resend" variable and reissue the transmit command. This is done by writing a value of 26H to the Command Register. If the "Resend" variable is 0, nothing needs to be done.

Note: If Remote DMA is not being used, the NIC does not need to be started before packets can be removed from the receive buffer ring. Hence, step 8 could be done before step 7.

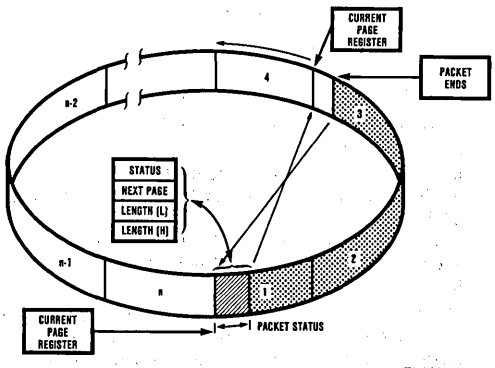
END OF PACKET OPERATIONS

At the end of the packet the NIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

SUCCESSFUL RECEPTION

If the packet is successfully received as shown, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. Note that the remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring. (The location of the next buffer had been previously calculated and temporarily stored in an internal scratchpad register.)

Termination of Received Packet—Packet Accepted

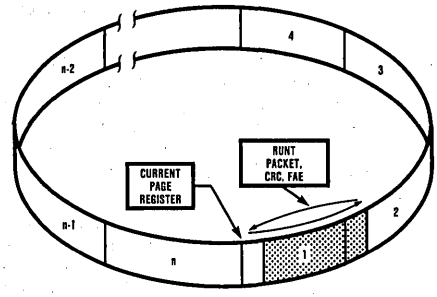


TL/F/8582–10

BUFFER RECOVERY FOR REJECTED PACKETS

If the packet is a runt packet or contains CRC or Frame Alignment errors, it is rejected. The buffer management logic resets the DMA back to the first buffer page used to store the packet (pointed to by CURR), recovering all buffers that had been used to store the rejected packet. This operation will not be performed if the NIC is programmed to accept either runt packets or packets with CRC or Frame Alignment errors. The received CRC is always stored in buffer memory after the last byte of received data for the packet.

Termination of Received Packet—Packet Rejected



TL/F/8582–13

Error Recovery

If the packet is rejected as shown, the DMA is restored by the NIC by reprogramming the DMA starting address pointed to by the Current Page Register.

REMOVING PACKETS FROM THE RING

Packets are removed from the ring using the Remote DMA or an external device. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the NIC moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer.

The following is a suggested method for maintaining the Receive Buffer Ring pointers.

1. At initialization, set up a software variable (next_pkt) to indicate where the next packet will be read. At the beginning of each Remote Read DMA operation, the value of next_pkt will be loaded into RSAR0 and RSAR1.
2. When initializing the NIC set:
   BNDRY = PSTART
   CURR = PSTART + 1
   next_pkt = PSTART + 1

7.0 Packet Reception (Continued)

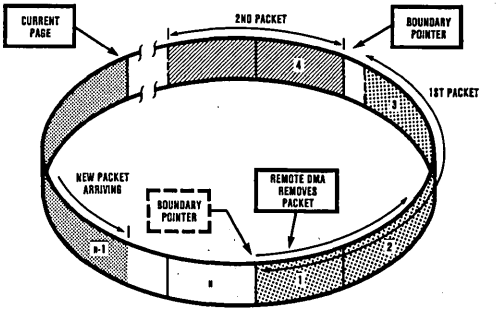
3. After a packet is DMAed from the Receive Buffer Ring, the Next Page Pointer (second byte in NIC buffer header) is used to update BNDRY and next_pkt.

   next_pkt = Next Page Pointer
   BNDRY = Next Page Pointer − 1
   If BNDRY < PSTART then BNDRY = PSTOP − 1

Note the size of the Receive Buffer Ring is reduced by one 256-byte buffer; this will not, however, impede the operation of the NIC.

In StarLAN applications using bus clock frequencies greater than 4 MHz, the NIC does not update the buffer header information properly because of the disparity between the network and bus clock speeds. The lower byte count is copied twice into the third and fourth locations of the buffer header and the upper byte count is not written. The upper byte count, however, can be calculated from the current next page pointer (second byte in the buffer header) and the previous next page pointer (stored in memory by the CPU). The following routine calculates the upper byte count and allows StarLAN applications to be insensitive to bus clock speeds. Next_pkt is defined similarly as above.

1st Received Packet Removed By Remote DMA



TL/F/8582–57

upper byte count = next page pointer − next_pkt − 1
if (upper byte count) < 0 then
upper byte count = (PSTOP − next_pkt) +
(next page pointer − PSTART) − 1
if (lower byte count) > 0 fch then
upper byte count = upper byte count + 1

STORAGE FORMAT FOR RECEIVED PACKETS

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register.

Storage Format

AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 1		Receive Byte Count 0	
Byte 2		Byte 1	

BOS = 0, WTS = 1 in Data Configuration Register.

This format used with Series 32000 808X type processors.

AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 0		Receive Byte Count 1	
Byte 1		Byte 2	

BOS = 1, WTS = 1 in Data Configuration Register.

This format used with 68000 type processors.

Note: The Receive Byte Count ordering remains the same for BOS=0 or 1.

AD7	AD0
Receive Status	
Next Packet Pointer	
Receive Byte Count 0	
Receive Byte Count 1	
Byte 0	
Byte 1	

BOS = 0, WTS = 0 in Data Configuration Register.

This format used with general 8-bit CPUs.

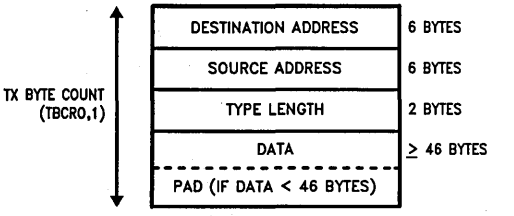
8.0 Packet Transmission

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0,1). When the NIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The NIC will generate and append the preamble, synch and CRC fields.

TRANSMIT PACKET ASSEMBLY

The NIC requires a contiguous assembled packet with the format shown. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes. The programmer is responsible for adding and stripping pad bytes.

General Transmit Packet Format



TL/F/8582–58

8.0 Packet Transmission (Continued)

TRANSMISSION

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR) is cleared and the NIC begins to prefetch transmit data from memory (unless the NIC is currently receiving). If the interframe gap has timed out the NIC will begin transmission.

CONDITIONS REQUIRED TO BEGIN TRANSMISSION

In order to transmit a packet, the following three conditions must be met:

1. The Interframe Gap Timer has timed out the first 6.4 μs of the Interframe Gap (See appendix for Interframe Gap Flowchart)
2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started)
3. If the NIC had collided, the backoff timer has expired.

In typical systems the NIC has already prefetched the first burst of bytes before the 6.4 μs timer expires. The time during which NIC transmits preamble can also be used to load the FIFO.

Note: If carrier sense is asserted before a byte has been loaded into the FIFO, the NIC will become a receiver.

COLLISION RECOVERY

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

Note: NCR reads as zeroes if excessive collisions are encountered.

TRANSMIT PACKET ASSEMBLY FORMAT

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register.

D15 D8	D7 D0
DA1	DA0
DA3	DA2
DA5	DA4
SA1	DA0
SA3	DA2
SA5	DA4
T/L1	T/L0
DATA 1	DATA 0

BOS = 0, WTS = 1 in Data Configuration Register.

This format is used with Series 32000, 808X type processors.

D15 D8	D7 D0
DA0	DA1
DA2	DA3
DA4	DA5
SA0	SA1
SA2	SA3
SA4	SA5
T/L0	T/L1
DATA 0	DATA 1

BOS = 1, WTS = 1 in Data Configuration Register.

This format is used with 68000 type processors.

D7 D0
DA0
DA1
DA2
DA3
DA4
DA5
SA0
SA1
SA2
SA3

BOS = 0, WTS = 0 in Data Configuration Register.

This format is used with general 8-bit CPUs.

Note: All examples above will result in a transmission of a packet in order of DA0, DA1, DA2, DA3 . . . bits within each byte will be transmitted least significant bit first.

DA = Destination Address

SA = Source Address

T/L = Type/Length Field

9.0 Remote DMA

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet.

Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address Register pair points to the beginning of the block to be moved while the Byte Count Register pair is used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

9.0 Remote DMA (Continued)

REMOTE WRITE

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches a count of zero.

REMOTE READ

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA Address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count Register reaches zero.

REMOTE DMA WRITE

Setting PRQ Using the Remote Read

Under certain conditions the NIC's bus state machine may issue /MWR and /PRD before PRQ for the first DMA transfer of a Remote Write Command. If this occurs this could cause data corruption, or cause the remote DMA count to be different from the main CPU count causing the system to "lock up".

To prevent this condition when implementing a Remote DMA Write, the Remote DMA Write command should first be preceded by a Remote DMA Read command to insure that the PRQ signal is asserted before the NIC starts its port read cycle. The reason for this is that the state machine that asserts PRQ runs independently of the state machine that controls the DMA signals. The DMA machine assumes that PRQ is asserted, but actually may not be. To remedy this situation, a single Remote Read cycle should be inserted before the actual DMA Write Command is given. This will ensure that PRQ is asserted when the Remote DMA Write is subsequently executed. This single Remote Read cycle is called a "dummy Remote Read." In order for the dummy Remote Read cycle to operate correctly, the Start Address should be programmed to a known, safe location in the buffer memory space, and the Remote Byte Count should be progammed to a value greater than 1. This will ensure that the master read cycle is performed safely, eliminating the possiblity of data corruption.

Remote Write with High Speed Buses

When implementing the Remote DMA Write solution in previous section with high speed buses and CPU's, timing problems may cause the system to hang. Therefore additional considerations are required.

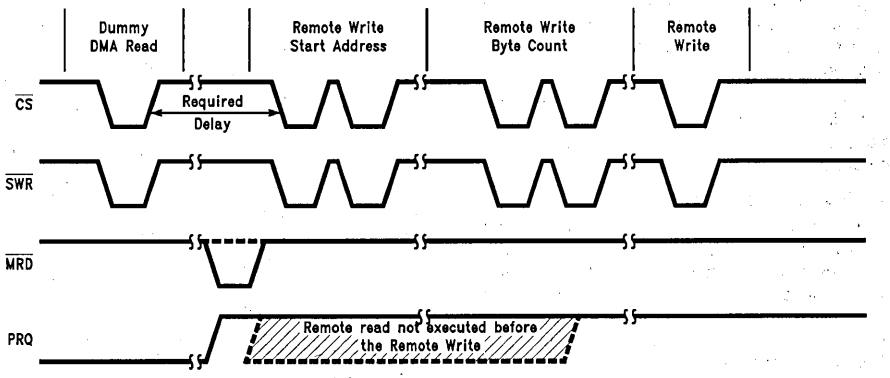
The problem occurs when the system can execute the dummy Remote Read and then start the Remote Write before the NIC has had a chance to execute the Remote Read. If this happens the PRQ signal will not get set, and the Remote Byte Count and Remote Start Address for the Remote Write operation could be corrupted. This is shown by the hatched waveforms in the timing diagram below. The execution of the Remote Read can be delayed by the local DMA operations (particularly during end-of-packet processing).

To ensure the dummy Remote Read does execute, a delay must be inserted between writing the Remote Read Command, and starting to write the Remote Write Start Address. (This time is designated in figure below by the delay arrows.) The recommended method to avoid this problem is, after the Remote Read command is given, to poll both bytes of the Current Remote DMA Address Registers. When the address has incremented, PRQ has been set. Software should recognize this and then start the Remote Write.

An additional caution for high speed systems is that the polling must follow guidelines specified at the end of Section 13. That is, there must be at least 4 bus clocks between chip selects. (For example, when BSCK = 20 MHz, then this time should be 200 ns.)

The general flow for executing a Remote Write is:

1. Set Remote Byte Count to a value >1 and Remote Start Address to unused RAM (one location before the transmit start address is usually a safe location).



TL/F/8582-96

Timing Diagram for Dummy Remote Read

Note: The dashed lines indicate incorrect timing.

9.0 Remote DMA (Continued)

2. Issue the "dummy" Remote Read command.
3. Read the Current Remote DMA Address (CRDA) (both bytes).
4. Compare to previous CRDA value if different go to 6.
5. Delay and jump to 3.
6. Set up for the Remote Write command, by setting the Remote Byte Count and the Remote Start Address (note that if the Remote Byte count in step 1 can be set to the tramsmit byte count plus one, and the Remote Start Address to one less, these will now be incremented to the correct values.)
7. Issue the Remote Write command.

FIFO AND BUS OPERATIONS

Overview

To accommodate the different rates at which data comes from (or goes to) the network and goes to (or comes from) the system memory, the NIC contains a 16-byte FIFO for buffering data between the bus and the media. The FIFO threshold is programmable, allowing filling (or emptying) the FIFO at different rates. When the FIFO has filled to its programmed threshold, the local DMA channel transfers these bytes (or words) into local memory. It is crucial that the local DMA is given access to the bus within a minimum bus latency time; otherwise a FIFO underrun (or overrun) occurs.

To understand FIFO underruns or overruns, there are two causes which produce this condition—

1) the bus latency is so long that the FIFO has filled (or emptied) from the network before the local DMA has serviced the FIFO.
2) the bus latency or bus data rate has slowed the throughput of the local DMA to point where it is slower than the network data rate (10 Mb/s). This second condition is also dependent upon DMA clock and word width (byte wide or word wide).

The worst case condition ultimately limits the overall bus latency which the NIC can tolerate.

FIFO Underrun and Transmit Enable

During transmission, if a FIFO underrun occurs, the Transmit enable (TXE) output may remain high (active). Generally, this will cause a very large packet to be transmitted onto the network. The jabber feature of the transceiver will terminate the transmission, and reset TXE.

To prevent this problem, a properly designed system will not allow FIFO underruns by giving the NIC a bus acknowledge within time shown in the maximum bus latency curves shown and described later.

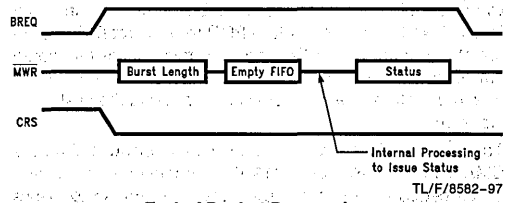
FIFO at the Beginning of Receive

At the beginning of reception, the NIC stores entire Address field of each incoming packet in the FIFO to determine whether the packet matches its Physical Address Registers or maps to one of its Multicast Registers. This causes the FIFO to accumulate 8 bytes. Furthermore, there are some synchronization delays in the DMA PLA. Thus, the actual time that BREQ is asserted from the time the Start of Frame Delimiter (SFD) is detected is 7.8 μs. This operation affects the bus latencies at 2 and 4 byte thresholds during the first receive BREQ since the FIFO must be filled to 8 bytes (4 words) before issuing a BREQ.

FIFO Operation at the End of Receive

When Carrier Sense goes low, the NIC enters its end of packet processing sequence, emptying its FIFO and writing the status information at the beginning of the packet, figure below. This NIC holds onto the bus for the entire sequence. The longest time BREQ may be extended occurs when a packet ends just as the NIC performs its last FIFO burst. The NIC, in this case, performs a programmed burst transfer followed by flushing the remaining bytes in the FIFO, and completes by writing the header information to memory. The following steps occur during this sequence.

1) NIC issues BREQ because the FIFO threshold has been reached.
2) During the burst, packet ends, resulting in BREQ extended.
3) NIC flushes remaining bytes from FIFO.
4) NIC performs internal processing to prepare for writing the header.
5) NIC writes 4-byte (2-word) header.
6) NIC deasserts BREQ.



TL/F/8582–97

End of Packet Processing

End of Packet Processing (EOPP) times for 10 MHz and 20 MHz have been tabulated in the table below.

End of Packet Processing Times for Various FIFO Thresholds, Bus Clocks and Transfer Modes

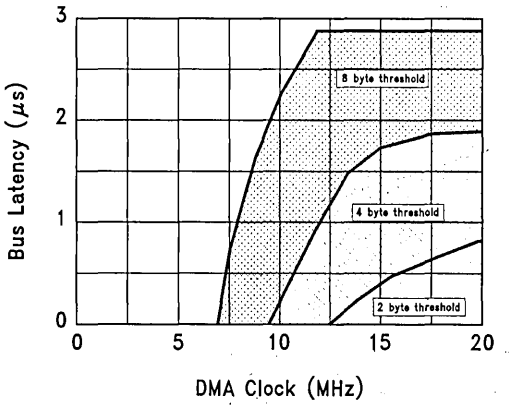
Mode	Threshold	Bus Clock	EOPP
Byte	2 bytes	10 MHz	7.0 μs
	4 bytes		8.6 μs
	8 bytes		11.0 μs
Byte	2 bytes	20 MHz	3.6 μs
	4 bytes		4.2 μs
	8 bytes		5.0 μs
Word	2 bytes	10 MHz	5.4 μs
	4 bytes		6.2 μs
	8 bytes		7.4 μs
Word	2 bytes	20 MHz	3.0 μs
	4 bytes		3.2 μs
	8 bytes		3.6 μs

Threshold Detection (Bus Latency)

To assure that no overwriting of data in the FIFO, the FIFO logic flags a FIFO overrun as the 13th byte is written into the FIFO, effectively shortening the FIFO to 13 bytes. The FIFO logic also operates differently in Byte Mode and in Word Mode. In Byte Mode, a threshold is indicated when the n+1

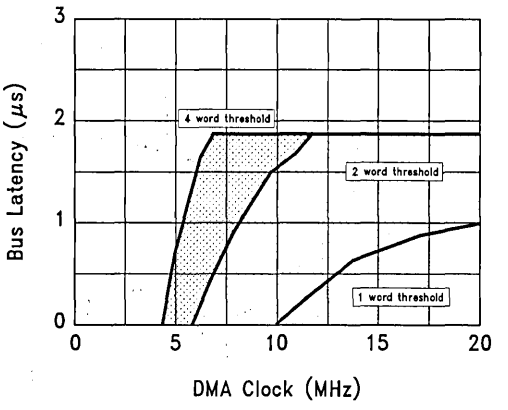
9.0 Remote DMA (Continued)

Maximum Bus Latency for Byte Mode



TL/F/8582-98

Maximum Bus Latency for Word Mode



TL/F/8582-99

byte has entered the FIFO; thus, with an 8 byte threshold, the NIC issues Bus Request (BREQ) when the 9th byte has entered the FIFO. For Word Mode, BREQ is not generated until the n+2 bytes have entered the FIFO. Thus, with a 4 word threshold (equivalent to 8 byte threshold), BREQ is issued when the 10th byte has entered the FIFO. The two graphs, the figures above, indicate the maximum allowable bus latency for Word and Byte transfer modes.

The FIFO at the Beginning of Transmit

Before transmitting, the NIC performs a prefetch from memory to load the FIFO. The number of bytes prefetched is the programmed FIFO threshold. The next BREQ is not issued until after the NIC actually begins trasmitting data, i.e., after SFD. The Transmit Prefetch diagram illustrates this process.

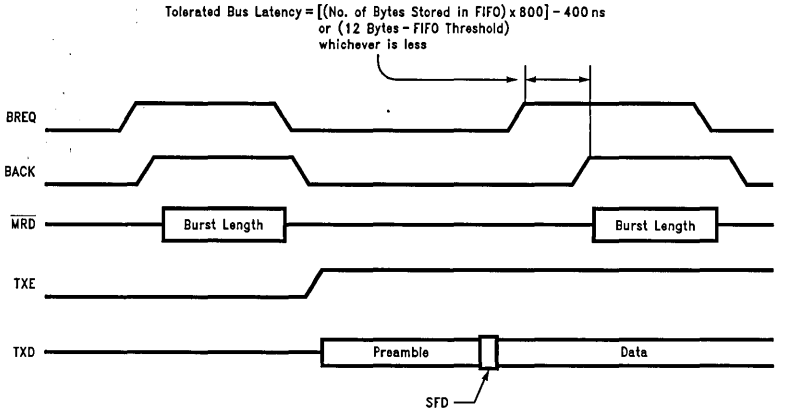
SEND PACKET COMMAND

The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a "Send Packet" Command. The DMA will be initialized to the value of the Boundary Pointer Register and the Remote Byte Count Register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop Register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring.

Note 1: In order for the NIC to correctly execute the Send Packet Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 0FH.

Note 2: The Send Packet command cannot be used with 68000 type processors.

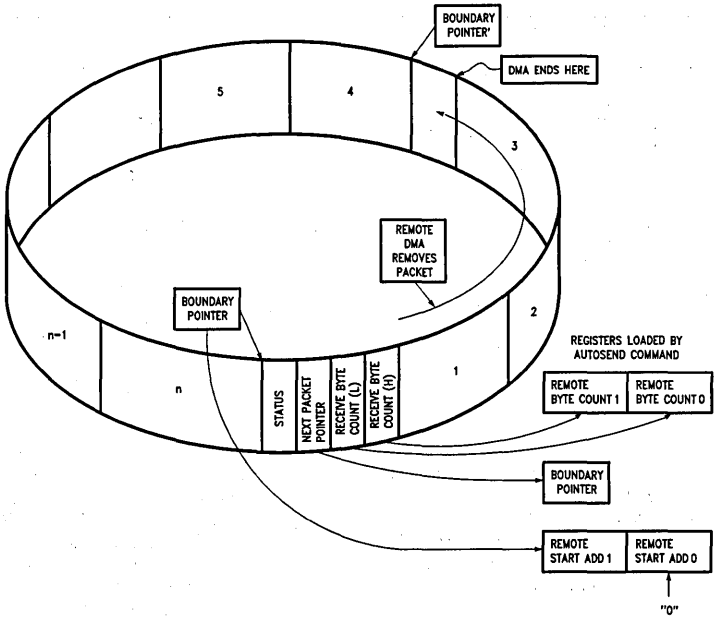
Transmit Prefetch Timing



TL/F/8582-A0

9.0 Remote DMA (Continued)

Remote DMA Autoinitialization from Buffer Ring

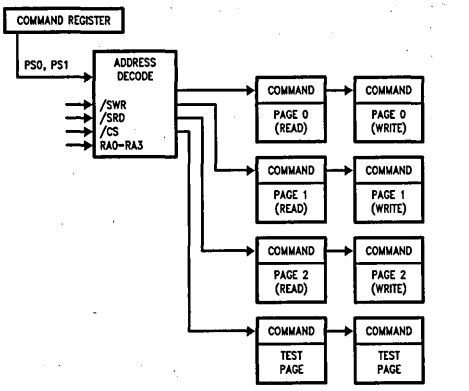


TL/F/8582–59

10.0 Internal Registers

All registers are 8-bit wide and mapped into two pages which are selected in the Command Register (PS0, PS1). Pins RA0–RA3 are used to address registers within each page. Page 0 registers are those registers which are commonly accessed during NIC operation while page 1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.

10.1 REGISTER ADDRESS MAPPING



TL/F/8582–60

10.0 Internal Registers (Continued)

10.2 REGISTER ADDRESS ASSIGNMENTS

Page 0 Address Assignments (PS1 = 0, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 (Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

Page 1 Address Assignments (PS1 = 0, PS0 = 1)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

10.0 Internal Registers (Continued)

Page 2 Address Assignments (PS1 = 1, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Page Start Register (PSTART)	Current Local DMA Address 0 (CLDA0)
02H	Page Stop Register (PSTOP)	Current Local DMA Address 1 (CLDA1)
03H	Remote Next Packet Pointer	Remote Next Packet Pointer
04H	Transmit Page Start Address (TPSR)	Reserved
05H	Local Next Packet Pointer	Local Next Packet Pointer
06H	Address Counter (Upper)	Address Counter (Upper)
07H	Address Counter (Lower)	Address Counter (Lower)
08H	Reserved	Reserved
09H	Reserved	Reserved
0AH	Reserved	Reserved
0BH	Reserved	Reserved
0CH	Receive Configuration Register (RCR)	Reserved
0DH	Transmit Configuration Register (TCR)	Reserved
0EH	Data Configuration Register (DCR)	Reserved
0FH	Interrupt Mask Register (IMR)	Reserved

Note: Page 2 registers should only be accessed for diagnostic purposes. They should not be modified during normal operation.

Page 3 should never be modified.

10.0 Internal Registers (Continued)

10.3 Register Descriptions

COMMAND REGISTER (CR) 00H (READ/WRITE)

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands may be overlapped, but with the following rules: (1) If a transmit command overlaps with a remote DMA operation, bits RD0, RD1, and RD2 must be maintained for the remote DMA command when setting the TXP bit. Note, if a remote DMA command is re-issued when giving the transmit command, the DMA will complete immediately if the remote byte count register have not been re-initialized. (2) If a remote DMA operation overlaps a transmission, RD0, RD1, and RD2 may be written with the desired values and a "0" written to the TXP bit. Writing a "0" to this bit has no effect. (3) A remote write DMA may not overlap remote read operation or visa versa. Either of these operations must either complete or be aborted before the other operation may start.

Bits PS1, PS0, RD2, and STP may be set any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

Bit	Symbol	Description
D0	STP	**STOP:** Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to a 1. **STP powers up high.** **Note:** If the NIC has previously been in start mode and the STP is set, both the STP and STA bits will remain set.
D1	STA	**START:** This bit is used to activate the NIC after either power up, or when the NIC has been placed in a reset mode by software command or error. **STA powers up low.**
D2	TXP	**TRANSMIT PACKET:** This bit must be set to initiate transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed. **Note:** Before the transmit command is given, the STA bit must be set and the STP bit reset.
D3, D4, D5	RD0, RD1, RD2	**REMOTE DMA COMMAND:** These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. RD2 RD1 RD0 0 0 0 Not Allowed 0 0 1 Remote Read 0 1 0 Remote Write (Note 2) 0 1 1 Send Packet 1 X X Abort/Complete Remote DMA (Note 1) **Note 1:** If a remote DMA operation is aborted and the remote byte count has not decremented to zero, PRQ (pin 29, DIP) will remain high. A read acknowledge ($\overline{RACK}$) on a write acknowledge ($\overline{WACK}$) will reset PRQ low. **Note 2:** For proper operation of the Remote Write DMA, there are two steps which must be performed before using the Remote Write DMA. The steps are as follows: i) Write a non-zero value into RBCR0. ii) Set bits RD2, RD1, RD0 to 0, 0, 1. iii) Set RBCR0, 1 and RSAR0, 1 iv) Issue the Remote Write DMA Command (RD2, RD1, RD0 = 0, 1, 0)
D6, D7	PS0, PS1	**PAGE SELECT:** These two encoded bits select which register page is to be accessed with addresses RA0–3. PS1 PS0 0 0 Register Page 0 0 1 Register Page 1 1 0 Register Page 2 1 1 Reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

INTERRUPT STATUS REGISTER (ISR) 07H (READ/WRITE)

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a "1" into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED:** Indicates packet received with no errors.
D1	PTX	**PACKET TRANSMITTED:** Indicates packet transmitted with no errors.
D2	RXE	**RECEIVE ERROR:** Indicates that a packet was received with one or more of the following errors: —CRC Error —Frame Alignment Error —FIFO Overrun —Missed Packet
D3	TXE	**TRANSMIT ERROR:** Set when packet transmitted with one or more of the following errors: —Excessive Collisions —FIFO Underrun
D4	OVW	**OVERWRITE WARNING:** Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer).
D5	CNT	**COUNTER OVERFLOW:** Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	**REMOTE DMA COMPLETE:** Set when Remote DMA operation has been completed.
D7	RST	**RESET STATUS:** Set when NIC enters reset state and cleared when a Start Command is issued to the CR. This bit is also set when a Receive Buffer Ring overflow occurs and is cleared when one or more packets have been removed from the ring. Writing to this bit has no effect. **NOTE:** This bit does not generate an interrupt, it is merely a status indicator.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

INTERRUPT MASK REGISTER (IMR) 0FH (WRITE)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. **The IMR powers up all zeroes.**

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Bit	Symbol	Description
D0	PRXE	**PACKET RECEIVED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received.
D1	PTXE	**PACKET TRANSMITTED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted.
D2	RXEE	**RECEIVE ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received with error.
D3	TXEE	**TRANSMIT ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error.
D4	OVWE	**OVERWRITE WARNING INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet.
D5	CNTE	**COUNTER OVERFLOW INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set.
D6	RDCE	**DMA COMPLETE INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed.
D7	reserved	reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

DATA CONFIGURATION REGISTER (DCR) 0EH (WRITE)

This Register is used to program the NIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO threshholds. **The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.**

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	LS	LAS	BOS	WTS

Bit	Symbol	Description
D0	WTS	**WORD TRANSFER SELECT** 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers ; WTS establishes byte or word transfers for both Remote and Local DMA transfers **Note:** When word-wide mode is selected, up to 32k words are addressable; A0 remains low.
D1	BOS	**BYTE ORDER SELECT** 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32000, 8086) 1: MS byte placed on AD7–AD0 and LS byte on AD15–AD8. (68000) ; Ignored when WTS is low
D2	LAS	**LONG ADDRESS SELECT** 0: Dual 16-bit DMA mode 1: Single 32-bit DMA mode ; When LAS is high, the contents of the Remote DMA registers RSAR0,1 are issued as A16–A31 Power up high.
D3	LS	**LOOPBACK SELECT** 0: Loopback mode selected. Bits D1, D2 of the TCR must also be programmed for Loopback operation. 1: Normal Operation.
D4	AR	**AUTO-INITIALIZE REMOTE** 0: Send Command not executed, all packets removed from Buffer Ring under program control. 1: Send Command executed, Remote DMA auto-initialized to remove packets from Buffer Ring. **Note:** Send Command cannot be used with 68000 type processors.
D5, D6	FT0, FT1	**FIFO THRESHHOLD SELECT:** Encoded FIFO threshhold. Establishes point at which bus is requested when filling or emptying the FIFO. During reception, the FIFO threshold indicates the number of bytes (or words) the FIFO has filled serially from the network before bus request (BREQ) is asserted. **Note:** FIFO threshold setting determines the DMA burst length. RECEIVE THRESHOLDS FT1 FT0 Word Wide Byte Wide 0 0 1 Word 2 Bytes 0 1 2 Words 4 Bytes 1 0 4 Words 8 Bytes 1 1 6 Words 12 Bytes During transmission, the FIFO threshold indicates the numer of bytes (or words) the FIFO has filled from the Local DMA before BREQ is asserted. Thus, the transmission threshold is 16 bytes less the receive threshold.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

TRANSMIT CONFIGURATION REGISTER (TCR) 0DH (WRITE)

The transmit configuration establishes the actions of the transmitter section of the NIC during transmission of a packet on the network. **LB1 and LB0 which select loopback mode power up as 0.**

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

Bit	Symbol	Description
D0	CRC	**INHIBIT CRC** 0: CRC appended by transmitter 1: CRC inhibited by transmitter ; In loopback mode CRC can be enabled or disabled to test the CRC logic.
D1, D2	LB0, LB1	**ENCODED LOOPBACK CONTROL:** These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 sets the LPBK pin high, this places the SNI in loopback mode and that D3 of the DCR must be set to zero for loopback operation. LB1 LB0 Mode 0 0 0 Normal Operation (LPBK = 0) Mode 1 0 1 Internal Loopback (LPBK = 0) Mode 2 1 0 External Loopback (LPBK = 1) Mode 3 1 1 External Loopback (LPBK = 0)
D3	ATD	**AUTO TRANSMIT DISABLE:** This bit allows another station to disable the NIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 0: Normal Operation 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.
D4	OFST	**COLLISION OFFSET ENABLE:** This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n,10)}$ slot times for first three collisions, then follows standard backoff. (For first three collisions station has higher average backoff delay making a low priority mode.)
D5	reserved	reserved
D6	reserved	reserved
D7	reserved	reserved

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

TRANSMIT STATUS REGISTER (TSR) 04H (READ)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	—	PTX

Bit	Symbol	Description
D0	PTX	**PACKET TRANSMITTED:** Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0").
D1	reserved	reserved
D2	COL	**TRANSMIT COLLIDED:** Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	**TRANSMIT ABORTED:** Indicates the NIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16).
D4	CRS	**CARRIER SENSE LOST:** This bit is set when carrier is lost during transmission of the packet. Carrier Sense is monitored from the end of Preamble/Synch until TXEN is dropped. Transmission is not aborted on loss of carrier.
D5	FU	**FIFO UNDERRUN:** If the NIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	**CD HEARTBEAT:** Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 μs of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	**OUT OF WINDOW COLLISION:** Indicates that a collision occurred after a slot time (51.2 μs). Transmissions rescheduled as in normal collisions.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

RECEIVE CONFIGURATION REGISTER (RCR) 0CH (WRITE)

This register determines operation of the NIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

Bit	Symbol	Description
D0	SEP	**SAVE ERRORED PACKETS** 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	**ACCEPT RUNT PACKETS:** This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	**ACCEPT BROADCAST:** Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	**ACCEPT MULTICAST:** Enables the receiver to accept a packet with a multicast address, all multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	**PROMISCUOUS PHYSICAL:** Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	**MONITOR MODE:** Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	reserved	reserved
D7	reserved	reserved

Note: D2 and D3 are "OR'd" together, i.e., if D2 and D3 are set the NIC will accept broadcast and multicast addresses as well as its own physical address. To establish full promiscuous mode, bits D2, D3, and D4 should be set. In addition the multicast hashing array must be set to all 1's in order to accept all multicast addresses.

10.0 Internal Registers (Continued)

10.3 Register Descriptions (Continued)

RECEIVE STATUS REGISTER (RSR) 0CH (READ)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the NIC which relinquishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED INTACT:** Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	**CRC ERROR:** Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	**FRAME ALIGNMENT ERROR:** Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	**FIFO OVERRUN:** This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	**MISSED PACKET:** Set when packet intended for node cannot be accepted by NIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	**PHYSICAL/MULTICAST ADDRESS:** Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Broadcast Address Match
D6	DIS	**RECEIVER DISABLED:** Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	**DEFERRING:** Set when CRS or COL inputs are active. If the transceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.

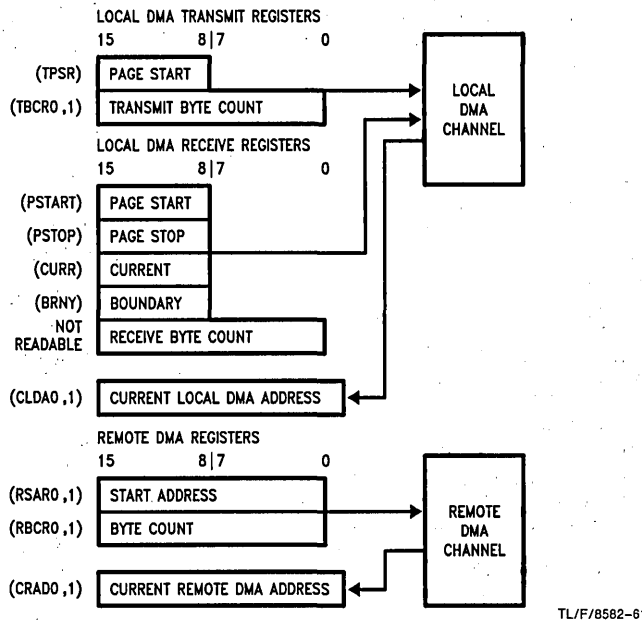
Note: Following coding applies to CRC and FAE bits

FAE	CRC	Type of Error
0	0	No Error (Good CRC and <6 Dribble Bits)
0	1	CRC Error
1	0	Illegal, will not occur
1	1	Frame Alignment Error and CRC Error

10.0 Internal Registers (Continued)

10.4 DMA REGISTERS

DMA Registers



TL/F/8582–61

The DMA Registers are partitioned into three groups; Transmit, Receive and Remote DMA Registers. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive Registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary Registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.

Note: In the figure above, registers are shown as 8 or 16 bits wide. Although some registers are 16-bit internal registers, all registers are accessed as 8-bit registers. Thus the 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0 and TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus. Thus they are shifted to positions 15-8 in the diagram above.

10.5 TRANSMIT DMA REGISTERS

TRANSMIT PAGE START REGISTER (TPSR)

This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256-byte page boundaries. The bit assignment is shown below. The values placed in bits D7–D0 will be used to initialize the higher order address (A8–A15) of the Local DMA for transmission. The lower order bits (A7–A0) are initialized to zero.

Bit Assignment

	7	6	5	4	3	2	1	0
TPSR	A15	A14	A13	A12	A11	A10	A9	A8

(A7–A0 Initialized to zero)

TRANSMIT BYTE COUNT REGISTER 0,1 (TBCR0, TBCR1)

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64k bytes. The NIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
TBCR1	L15	L14	L13	L12	L11	L10	L9	L8
	7	6	5	4	3	2	1	0
TBCR0	L7	L6	L5	L4	L3	L2	L1	L0

10.6 LOCAL DMA RECEIVE REGISTERS

PAGE START STOP REGISTERS (PSTART, PSTOP)

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the NIC uses fixed 256-byte buffers aligned on page boundaries only the upper eight bits of the start and stop address are specified.

PSTART,PSTOP bit assignment

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

BOUNDARY (BNRY) REGISTER

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

10.0 Internal Registers (Continued)

CURRENT PAGE REGISTER (CURR)

This register is used internally by the Buffer Management Logic as a backup register for reception. CURR contains the address of the first buffer to be used for a packet reception and is used to restore DMA pointers in the event of receive errors. This register is initialized to the same value as PSTART and should not be written to again unless the controller is Reset.

	7	6	5	4	3	2	1	0
CURR	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT LOCAL DMA REGISTER 0,1 (CLDA0,1)

These two registers can be accessed to determine the current Local DMA Address.

	7	6	5	4	3	2	1	0
CLDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CLDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.7 REMOTE DMA REGISTERS

REMOTE START ADDRESS REGISTERS (RSAR0,1)

Remote DMA operations are programmed via the Remote Start Address (RSAR0,1) and Remote Byte Count (RBCR0,1) registers. The Remote Start Address is used to point to the start of the block of data to be transferred and the Remote Byte Count is used to indicate the length of the block (in bytes).

	7	6	5	4	3	2	1	0
RSAR1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
RSAR0	A7	A6	A5	A4	A3	A2	A1	A0

6.4.3.2 REMOTE BYTE COUNT REGISTERS (RBCR0,1)

	7	6	5	4	3	2	1	0
RBCR1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8

	7	6	5	4	3	2	1	0
RBCR0	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

Note:

RSAR0 programs the start address bits A0–A7.
RSAR1 programs the start address bits A8–A15.
Address incremented by two for word transfers, and by one for byte transfers.
Byte Count decremented by two for word transfers and by one for byte transfers.
RBCR0 programs LSB byte count.
RBCR1 programs MSB byte count.

CURRENT REMOTE DMA ADDRESS (CRDA0, CRDA1)

The Current Remote DMA Registers contain the current address of the Remote DMA. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
CRDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CRDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.8 PHYSICAL ADDRESS REGISTERS (PAR0–PAR5)

The physical address registers are used to compare the destination address of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte-wide basis. The bit assignment shown below relates the sequence in PAR0–PAR5 to the bit sequence of the received packet.

	D7	D6	D5	D4	D3	D2	D1	D0
PAR0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
PAR1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
PAR2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
PAR3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
PAR4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
PAR5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40

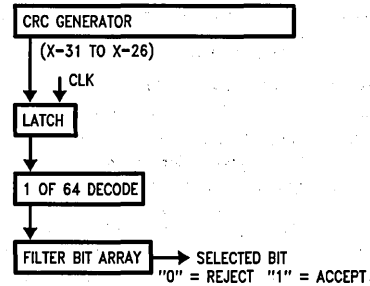
	Destination Address							Source
P/S	DA0	DA1	DA2	DA3	.....	DA46	DA47	SA0 ...

Note:
P/S = Preamble, Synch
DA0 = Physical/Multicast Bit

10.9 MULTICAST ADDRESS REGISTERS (MAR0–MAR7)

The multicast address registers provide filtering of multicast addresses hashed by the CRC logic. All destination addresses are fed through the CRC logic and as the last bit of the destination address enters the CRC, the 6 most significant bits of the CRC generator are latched. These 6 bits are then decoded by a 1 of 64 decode to index a unique filter bit (FB0–63) in the multicast address registers. If the filter bit selected is set, the multicast packet is accepted. The system designer would use a program to determine which filter bits to set in the multicast registers. All multicast filter bits that correspond to multicast address accepted by the node are then set to one. To accept all multicast packets all of the registers are set to all ones.

Note: Although the hashing algorithm does not guarantee perfect filtering of multicast address, it will perfectly filter up to 64 multicast addresses if these addresses are chosen to map into unique locations in the multicast filter.



TL/F/8582–62

10.0 Internal Registers (Continued)

	D7	D6	D5	D4	D3	D2	D1	D0
MAR0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MAR1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MAR2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MAR3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MAR4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MAR5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MAR6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MAR7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

If address Y is found to hash to the value 32 (20H), then FB32 in MAR4 should be initialized to "1". This will cause the NIC to accept any multicast packet with the address Y.

NETWORK TALLY COUNTERS

Three 8-bit counters are provided for monitoring the number of CRC errors, Frame Alignment Errors and Missed Packets. The maximum count reached by any counter is 192 (C0H). These registers will be cleared when read by the CPU. The count is recorded in binary in CT0–CT7 of each Tally Register.

Frame Alignment Error Tally (CNTR0)

This counter is incremented every time a packet is received with a Frame Alignment Error. The packet must have been recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR0	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

CRC Error Tally (CNTR1)

This counter is incremented every time a packet is received with a CRC error. The packet must first be recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR1	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

Frames Lost Tally Register (CNTR2)

This counter is incremented if a packet cannot be received due to lack of buffer resources. In monitor mode, this counter will count the number of packets that pass the address recognition logic.

	7	6	5	4	3	2	1	0
CNTR2	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

FIFO

This is an eight bit register that allows the CPU to examine the contents of the FIFO after loopback. The FIFO will contain the last 8 data bytes transmitted in the loopback packet. Sequential reads from the FIFO will advance a pointer in the FIFO and allow reading of all 8 bytes.

	7	6	5	4	3	2	1	0
FIFO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Note: The FIFO should only be read when the NIC has been programmed in loopback mode.

NUMBER OF COLLISIONS (NCR)

This register contains the number of collisions a node experiences when attempting to transmit a packet. If no collisions are experienced during a transmission attempt, the COL bit of the TSR will not be set and the contents of NCR will be zero. If there are excessive collisions, the ABT bit in the TSR will be set and the contents of NCR will be zero. The NCR is cleared after the TXP bit in the CR is set.

	7	6	5	4	3	2	1	0
NCR	—	—	—	—	NC3	NC2	NC1	NC0

11.0 Initialization Procedures

The NIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the NIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The NIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the NIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

Initialization Sequence

The following initialization procedure is mandatory.

1) Program Command Register for Page 0 (Command Register = 21H)
2) Initialize Data Configuration Register (DCR)
3) Clear Remote Byte Count Registers (RBCR0, RBCR1)
4) Initialize Receive Configuration Register (RCR)
5) Place the NIC in LOOPBACK mode 1 or 2 (Transmit Configuration Register = 02H or 04H)
6) Initialize Receive Buffer Ring: Boundary Pointer (BNDRY), Page Start (PSTART), and Page Stop (PSTOP)
7) Clear Interrupt Status Register (ISR) by writing 0FFh to it.
8) Initialize Interrupt Mask Register (IMR)
9) Program Command Register for page 1 (Command Register = 61H)
   i)Initialize Physical Address Registers (PAR0-PAR5)
   ii)Initialize Multicast Address Registers (MAR0-MAR7)
   iii)Initialize CURRent pointer
10) Put NIC in START mode (Command Register = 22H). The local receive DMA is still not active since the NIC is in LOOPBACK.
11) Initialize the Transmit Configuration for the intended value. The NIC is now ready for transmission and reception.

11.0 Initialization Procedures

(Continued)

Before receiving packets, the user must specify the location of the Receive Buffer Ring. This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Registers must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

12.0 Loopback Diagnostics

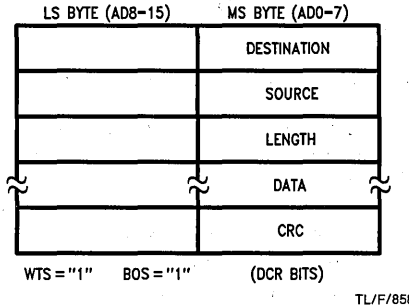
Three forms of local loopback are provided on the NIC. The user has the ability to loopback through the deserializer on the DP8390D NIC, through the DP8391 SNI, and to the coax to check the link through the transceiver circuitry. **Because of the half duplex architecture of the NIC, loopback testing is a special mode of operation with the following restrictions:**

Restrictions During Loopback

The FIFO is split into two halves, one used for transmission the other for reception. Only 8-bit fields can be fetched from memory so two tests are required for 16-bit systems to verify integrity of the entire data path. During loopback the maximum latency from the assertion of BREQ to BACK is 2.0 μs. Systems that wish to use the loopback test yet do not meet this latency can limit the loopback packet to 7 bytes without experiencing underflow. Only the last 8 bytes of the loopback packet are retained in the FIFO. The last 8 bytes can be read through the FIFO register which will advance through the FIFO to allow reading the receive packet sequentially.

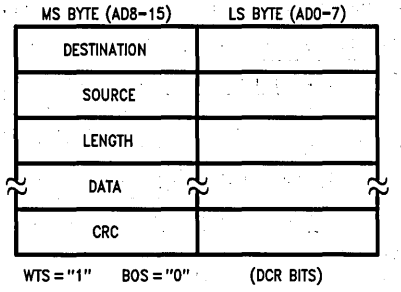
DESTINATION ADDRESS	= (6 bytes) Station Physical Address
SOURCE ADDRESS	|
LENGTH	2 bytes
DATA	= 46 to 1500 bytes
CRC	Appended by NIC if CRC = "0" in TCR

When in word-wide mode with Byte Order Select set, the loopback packet must be assembled in the even byte locations as shown below. (The loopback only operates with byte wide transfers.)



TL/F/8582-15

When in word-wide mode with Byte Order Select low, the following format must be used for the loopback packet.



TL/F/8582-16

Note: When using loopback in word mode 2n bytes must be programmed in TBCR0, 1. Where n = actual number of bytes assembled in even or odd location.

To initiate a loopback the user first assembles the loopback packet then selects the type of loopback using the Transmit Configuration register bits LB0, LB1. The transmit configuration register must also be set to enable or disable CRC generation during transmission. The user then issues a normal transmit command to send the packet. During loopback the receiver checks for an address match and if CRC bit in the TCR is set, the receiver will also check the CRC. The last 8 bytes of the loopback packet are buffered and can be read out of the FIFO using the FIFO read port.

Loopback Modes

MODE 1: Loopback Through the Controller (LB1 = 0, LB0 = 1).

If the loopback is through the NIC then the serializer is simply linked to the deserializer and the receive clock is derived from the transmit clock.

MODE 2: Loopback Through the SNI (LB1 = 1, LB0 = 0).

If the loopback is to be performed through the SNI, the NIC provides a control (LPBK) that forces the SNI to loopback all signals.

MODE 3: Loopback to Coax (LB1 = 1, LB0 = 1).

Packets can be transmitted to the coax in loopback mode to check all of the transmit and receive paths and the coax itself.

Note: In MODE 1, CRS and COL lines are not indicated in any status register, but the NIC will still defer if these lines are active. In MODE 2, COL is masked and in MODE 3 CRS and COL are not masked. It is not possible to go directly between the loopback modes, it is necessary to return to normal operation (00H) when changing modes.

Reading the Loopback Packet

The last eight bytes of a received packet can be examined by 8 consecutive reads of the FIFO register. The FIFO pointer is incremented after the rising edge of the CPU's read strobe by internally synchronizing and advancing the pointer. This may take up to four bus clock cycles, if the pointer has not been incremented by the time the CPU reads the FIFO register again, the NIC will insert wait states

Note: The FIFO may only be read during Loopback. Reading the FIFO at any other time will cause the NIC to malfunction.

12.0 Loopback Diagnostics (Continued)

Alignment of the Received Packet in the FIFO

Reception of the packet in the FIFO begins at location zero, after the FIFO pointer reaches the last location in the FIFO, the pointer wraps to the top of the FIFO overwriting the previously received data. This process continues until the last byte is received. The NIC then appends the received byte count in the next two locations of the FIFO. The contents of the Upper Byte Count are also copied to the next FIFO location. The number of bytes used in the loopback packet determines the alignment of the packet in the FIFO. The alignment for a 64-byte packet is shown below.

FIFO LOCATION	FIFO CONTENTS	
0	LOWER BYTE COUNT	→ First Byte Read
1	UPPER BYTE COUNT	→ Second Byte Read
2	UPPER BYTE COUNT	•
3	LAST BYTE	•
4	CRC1	•
5	CRC2	•
6	CRC3	•
7	CRC4	→ Last Byte Read

For the following alignment in the FIFO the packet length should be (N × 8) + 5 Bytes. Note that if the CRC bit in the TCR is set, CRC will not be appended by the transmitter. If the CRC is appended by the transmitter, the last four bytes, bytes N-3 to N, correspond to the CRC.

FIFO LOCATION	FIFO CONTENTS	
0	BYTE N-4	→ First Byte Read
1	BYTE N-3 (CRC1)	AR Second Byte Read
2	BYTE N-2 (CRC2)	•
3	BYTE N-1 (CRC3)	•
4	BYTE N (CRC4)	•
5	LOWER BYTE COUNT	•
6	UPPER BYTE COUNT	→ Last Byte Read
7	UPPER BYTE COUNT	

LOOPBACK TESTS

Loopback capabilities are provided to allow certain tests to be performed to validate operation of the DP8390D NIC prior to transmitting and receiving packets on a live network. Typically these tests may be performed during power up of a node. The diagnostic provides support to verify the following:

1) Verify integrity of data path. Received data is checked against transmitted data.
2) Verify CRC logic's capability to generate good CRC on transmit, verify CRC on receive (good or bad CRC).
3) Verify that the Address Recognition Logic can
   a) Recognize address match packets
   b) Reject packets that fail to match an address

LOOPBACK OPERATION IN THE NIC

Loopback is a modified form of transmission using only half of the FIFO. This places certain restrictions on the use of loopback testing. When loopback mode is selected in the TCR, the FIFO is split. A packet should be assembled in memory with programming of TPSR and TBCR0,TBCR1 registers. When the transmit command is issued the following operations occur:

Transmitter Actions

1) Data is transferred from memory by the DMA until the FIFO is filled. For each transfer TBCR0 and TBCR1 are decremented. (Subsequent burst transfers are initiated when the number of bytes in the FIFO drops below the programmed threshold.)
2) The NIC generates 56 bits of preamble followed by an 8-bit synch pattern.
3) Data transferred from FIFO to serializer.
4) If CRC=1 in TCR, no CRC calculated by NIC, the last byte transmitted is the last byte from the FIFO (Allows software CRC to be appended). If CRC=0, NIC calculates and appends four bytes of CRC.
5) At end of Transmission PTX bit set in ISR.

Receiver Actions

1) Wait for synch, all preamble stripped.
2) Store packet in FIFO, increment receive byte count for each incoming byte.
3) If CRC=0 in TCR, receiver checks incoming packet for CRC errors. If CRC=1 in TCR, receiver does not check CRC errors, CRC error bit always set in RSR (for address matching packets).
4) At end of receive, receive byte count written into FIFO, receive status register is updated. The PRX bit is typically set in the RSR even if the address does not match. If CRC errors are forced, the packet must match the address filters in order for the CRC error bit in the RS to be set.

EXAMPLES

The following examples show what results can be expected from a properly operating NIC during loopback. The restrictions and results of each type of loopback are listed for reference. The loopback tests are divided into two sets of tests. One to verify the data path, CRC generation and byte count through all three paths. The second set of tests uses internal loopback to verify the receiver's CRC checking and address recognition. For all of the tests the DCR was programmed to 40h.

PATH	TCR	RCR	TSR	RSR	ISR
NIC Internal	02	00	53(1)	02(2)	02(3)

Note 1: Since carrier sense and collision detect inputs are blocked during internal loopback, carrier and CD heartbeat are not seen and the CRS and CDH bits are set.

Note 2: CRC errors are always indicated by receiver if CRC is appended by the transmitter.

Note 3: Only the PTX bit in the ISR is set, the PRX bit is only set if status is written to memory. In loopback this action does not occur and the PRX bit remains 0 for all loopback modes.

Note 4: All values are hex.

12.0 Loopback Diagnostics (Continued)

PATH	TCR	RCR	TSR	RSR	ISR
NIC External	04	00	43(1)	02	02

Note 1: CDH is set, CRS is not set since it is generated by the external encoder/decoder.

PATH	TCR	RCR	TSR	RSR	ISR
NIC External	06	00	03(1)	02	02(2)

Note 1: CDH and CRS should not be set. The TSR however, could also contain 01H,03H,07H and a variety of other values depending on whether collisions were encountered or the packet was deferred.

Note 2. Will contain 08H if packet is not transmittable.

Note 3: During external loopback the NIC is now exposed to network traffic, it is therefore possible for the contents of both the Receive portion of the FIFO and the RSR to be corrupted by any other packet on the network. Thus in a live network the contents of the FIFO and RSR should not be depended on. The NIC will still abide by the standard CSMA/CD protocol in external loopback mode. (i.e. The network will not be disturbed by the loopback packet).

Note 4: All values are hex.

CRC AND ADDRESS RECOGNITION

The next three tests exercise the address recognition logic and CRC. These tests should be performed using internal loopback only so that the NIC is isolated from interference from the network. These tests also require the capability to generate CRC in software.

The address recognition logic cannot be directly tested. The CRC and FAE bits in the RSR are only set if the address of the packet matches the address filters. If errors are expected to be set and they are not set, the packet has been rejected on the basis of an address mismatch. The following sequence of packets will test the address recognition logic. The DCR should be set to 40H, the TCR should be set to 03H with a software generated CRC.

Packet Contents			Results
Test	**Address**	**CRC**	**RSR**
Test A	Matching	Good	01(1)
Test B	Matching	Bad	02(2)
Test C	Non-Matching	Bad	01

Note 1: Status will read 21H if multicast address used.

Note 2: Status will read 22H if multicast address used.

Note 3: In test A, the RSR is set up. In test B the address is found to match since the CRC is flagged as bad. Test C proves that the address recognition logic can distinguish a bad address and does not notify the RSR of the bad CRC. The receiving CRC is proven to work in test A and test B.

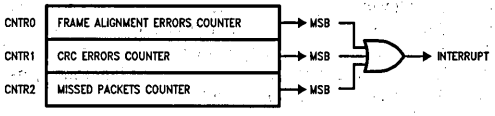
Note 4: All values are hex.

NETWORK MANAGEMENT FUNCTIONS

Network management capabilities are required for maintenance and planning of a local area network. The NIC supports the minimum requirement for network management in hardware, the remaining requirements can be met with software counts. There are three events that software alone can not track during reception of packets: CRC errors, Frame Alignment errors, and missed packets.

Since errored packets can be rejected, the status associated with these packets is lost unless the CPU can access the Receive Status Register before the next packet arrives. In situations where another packet arrives very quickly, the CPU may have no opportunity to do this. The NIC counts the number of packets with CRC errors and Frame Alignment errors. 8-bit counters have been selected to reduce overhead. The counters will generate interrupts whenever their MSBs are set so that a software routine can accumulate the network statistics and reset the counters before overflow occurs. The counters are sticky so that when they reach a count of 192 (C0H) counting is halted. An additional counter is provided to count the number of packets NIC misses due to buffer overflow or being offline.

The structure of the counters is shown below:



TL/F/8582–63

Additional information required for network management is available in the Receive and Transmit Status Registers. Transmit status is available after each transmission for information regarding events during transmission.

Typically, the following statistics might be gathered in software:

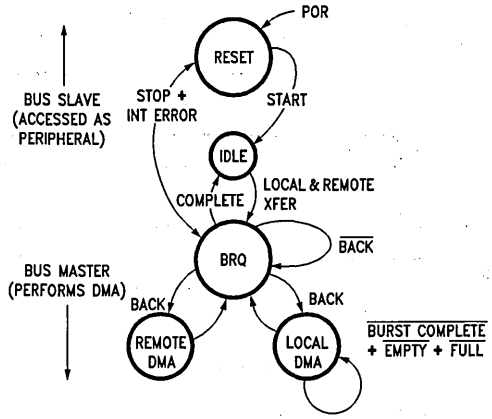
Traffic:
- Frames Sent OK
- Frames Received OK
- Multicast Frames Received
- Packets Lost Due to Lack of Resources
- Retries/Packet

Errors:
- CRC Errors
- Alignment Errors
- Excessive Collisions
- Packet with Length Errors
- Heartbeat Failure

13.0 Bus Arbitration and Timing

The NIC operates in three possible modes:

- BUS MASTER (WHILE PERFORMING DMA)
- BUS SLAVE (WHILE BEING ACCESSED BY CPU)
- IDLE



TL/F/8582-64

Upon power-up the NIC is in an indeterminant state. After receiving a Hardware Reset the NIC comes up as a slave in the Reset State. The receiver and transmitter are both disabled in this state. The reset state can be reentered under three conditions, soft reset (Stop Command), hard reset (RESET input) or an error that shuts down the receiver or transmitter (FIFO underflow or overflow). After initialization of registers, the NIC is issued a Start command and the NIC enters Idle state. Until the DMA is required the NIC remains in an idle state. The idle state is exited by a request from the FIFO in the case of receive or transmit, or from the Remote/DMA in the case of Remote DMA operation. After acquiring the bus in a BREQ/BACK handshake the Remote or Local DMA transfer is completed and the NIC reenters the idle state.

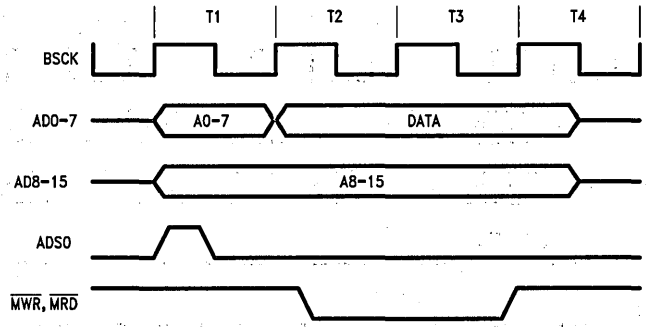
DMA TRANSFERS TIMING

The DMA can be programmed for the following types of transfers:

16-Bit Address, 8-bit Data Transfer
16-Bit Address, 16-bit Data Transfer
32-Bit Address, 8-bit Data Transfer
32-Bit Address, 16-bit Data Transfer

All DMA transfers use BSCK for timing. 16-Bit Address modes require 4 BSCK cycles as shown below:

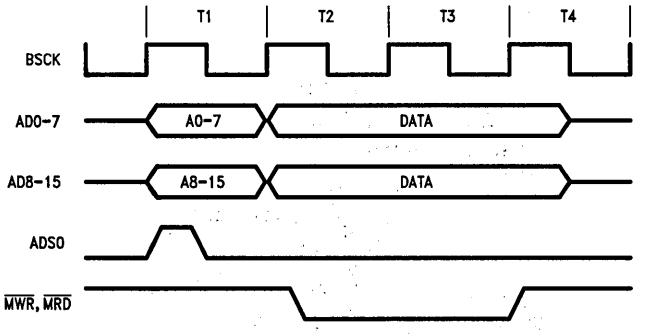
16-Bit Address, 8-Bit Data



TL/F/8582-65

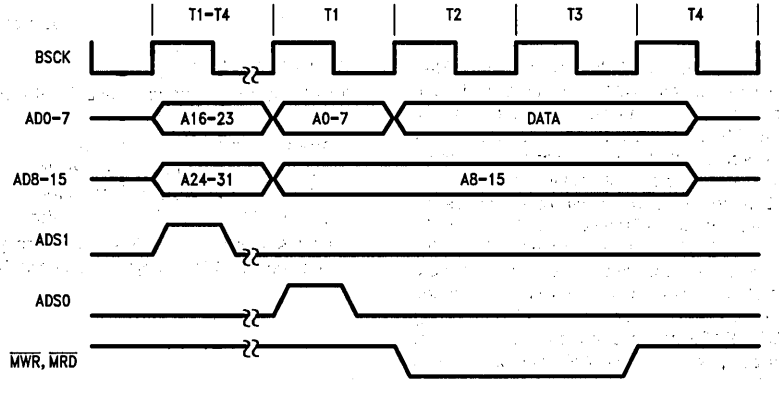
13.0 Bus Arbitration and Timing (Continued)

16-Bit Address, 16-Bit Data



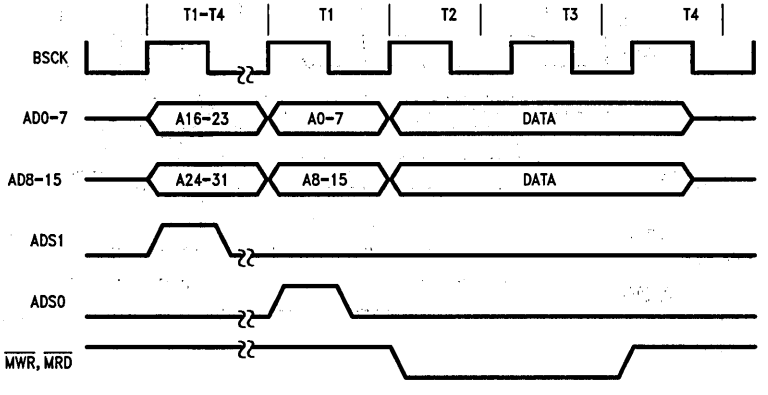
TL/F/8582-66

32-Bit Address, 8-Bit Data



TL/F/8582-67

32-Bit Address, 16-Bit Data



TL/F/8582-68

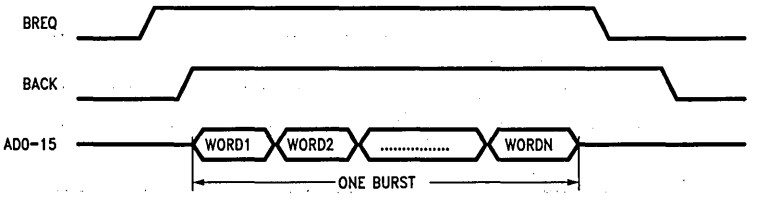
Note: In 32-bit address mode, ADS1 is at TRI-STATE after the first T1–T4 states; thus, a 4.7k pull-down resistor is required for 32-bit address mode.

13.0 Bus Arbitration and Timing (Continued)

When in 32-bit mode four additional BSCK cycles are required per burst. The first bus cycle (T1′–T4′) of each burst is used to output the upper 16-bit addresses. This 16-bit address is programmed in RSAR0 and RSAR1 and points to a 64k page of system memory. All transmitted or received packets are constrained to reside within this 64k page.

FIFO BURST CONTROL

All Local DMA transfers are burst transfers, once the DMA requests the bus and the bus is acknowledged, the DMA will transfer an exact burst of bytes programmed in the Data Configuration Register (DCR) then relinquish the bus. If there are remaining bytes in the FIFO the next burst will not be initiated until the FIFO threshold is exceeded. If desired the DMA can empty/fill the FIFO when it acquires the bus. If BACK is removed during the transfer, the burst transfer will be aborted. **(DROPPING BACK DURING A DMA CYCLE IS NOT RECOMMENDED.)**

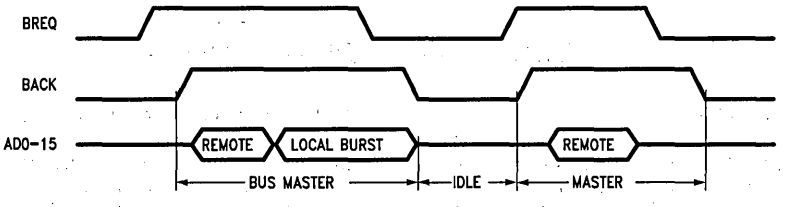


TL/F/8582–69

where N = 1, 2, 4, or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode

INTERLEAVED LOCAL OPERATION

If a remote DMA transfer is initiated or in progress when a packet is being received or transmitted, the Remote DMA transfer will be interrupted for higher priority Local DMA transfers. When the Local DMA transfer is completed the Remote DMA will rearbitrate for the bus and continue its transfers. This is illustrated below:



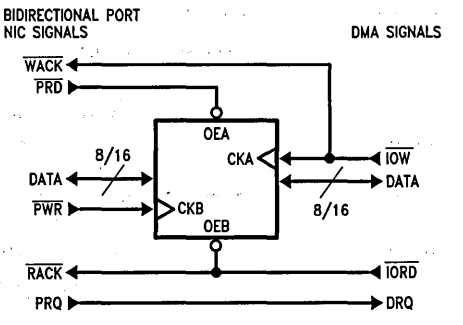
TL/F/8582–70

Note that if the FIFO requires service while a remote DMA is in progress, BREQ is not dropped and the Local DMA burst is appended to the Remote Transfer. When switching from a local transfer to a remote transfer, however, BREQ is dropped and raised again. This allows the CPU or other devices to fairly contend for the bus.

REMOTE DMA-BIDIRECTIONAL PORT CONTROL

The Remote DMA transfers data between the local buffer memory and a bidirectional port (memory to I/O transfer). This transfer is arbited on a byte by byte basis versus the burst transfer used for Local DMA transfers. This bidirectional port is also read/written by the host. All transfers through this port are asynchronous. At any one time transfers are limited to one direction, either from the port to local buffer memory (Remote Write) or from local buffer memory to the port (Remote Read).

Bus Handshake Signals for Remote DMA Transfers



TL/F/8582–71

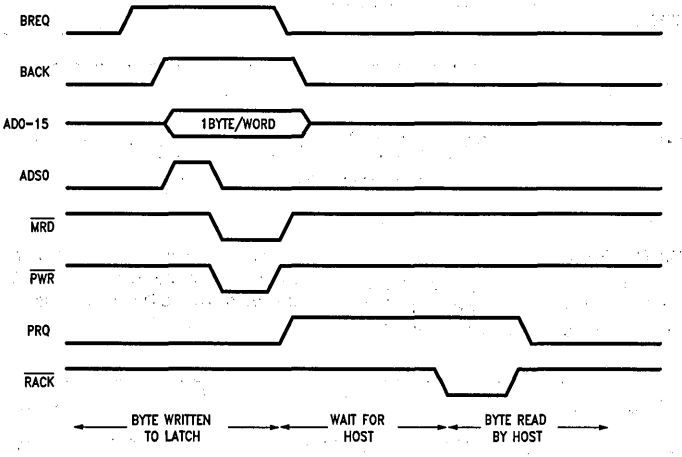
13.0 Bus Arbitration and Timing (Continued)

REMOTE READ TIMING

1) The DMA reads byte/word from local buffer memory and writes byte/word into latch, increments the DMA address and decrements the byte count (RBCR0,1).
2) A Request Line (PRQ) is asserted to inform the system that a byte is available.
3) The system reads the port, the read strobe ($\overline{RACK}$) is used as an acknowledge by the Remote DMA and it goes back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.

Note that in order for the Remote DMA to transfer a byte from memory to the latch, it must arbitrate access to the local bus via a BREQ, BACK handshake. After each byte or word is transferred to the latch, BREQ is dropped. If a Local DMA is in progress, the Remote DMA is held off until the local DMA is complete.



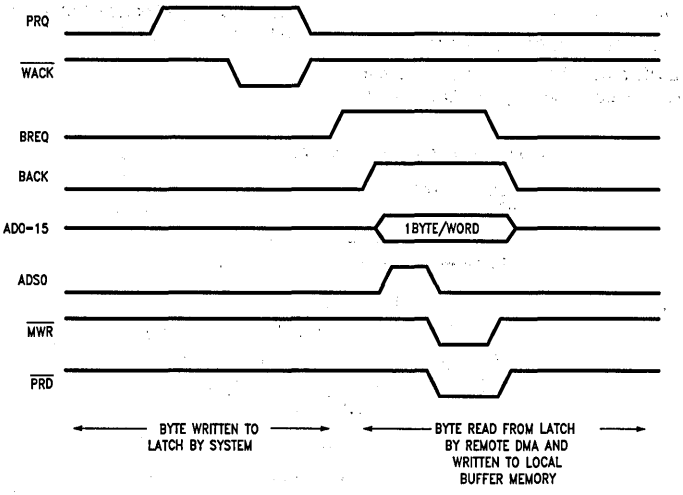
TL/F/8582–72

REMOTE WRITE TIMING

A Remote Write operation transfers data from the I/O port to the local buffer RAM. The NIC initiates a transfer by requesting a byte/word via the PRQ. The system transfers a byte/word to the latch via $\overline{IOW}$, this write strobe is detected by the NIC and PRQ is removed. By removing the PRQ, the Remote DMA holds off further transfers into the latch until the current byte/word has been transferred from the latch, PRQ is reasserted and the next transfer can begin.

1) NIC asserts PRQ. System writes byte/word into latch. NIC removes PRQ.
2) Remote DMA reads contents of port and writes byte/word to local buffer memory, increments address and decrements byte count (RBCR0,1).
3) Go back to step 1.

Steps 1–3 are repeated until the remote DMA is complete.



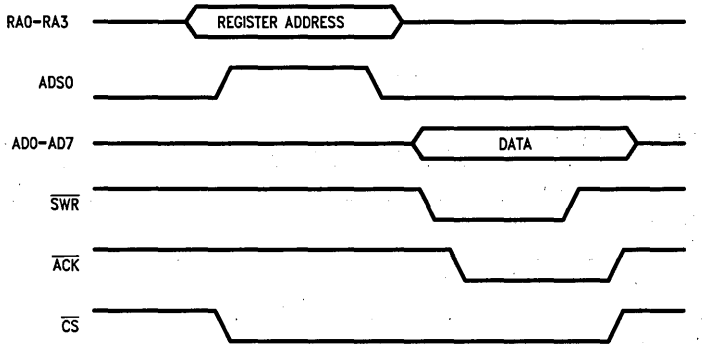
TL/F/8582–73

13.0 Bus Arbitration and Timing (Continued)

SLAVE MODE TIMING

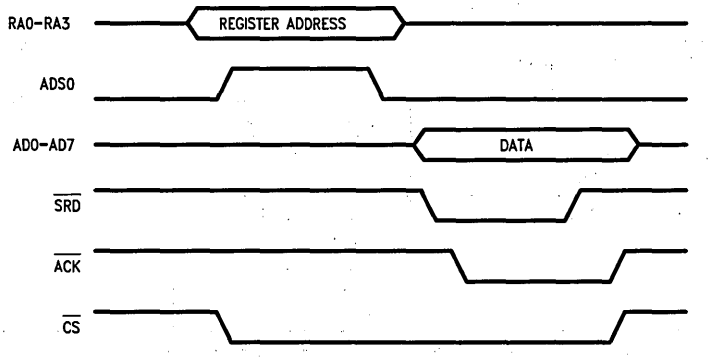
When $\overline{CS}$ is low, the NIC becomes a bus slave. The CPU can then read or write any internal registers. All register access is byte wide. The timing for register access is shown below. The host CPU accesses internal registers with four address lines, RA0–RA3, $\overline{SRD}$ and $\overline{SWR}$ strobes. ADS0 is used to latch the address when interfacing to a multiplexed, address data bus. Since the NIC may be a local bus master when the host CPU attempts to read or write to the controller, an $\overline{ACK}$ line is used to hold off the CPU until the NIC leaves master mode. Some number of BSCK cycles is also required to allow the NIC to synchronize to the read or write cycle.

Write to Register



TL/F/8582–74

Read from Register

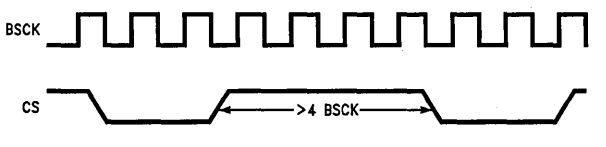


TL/F/8582–75

TIME BETWEEN CHIP SELECTS

The NIC requires that successive chip selects be no closer than 4 bus clocks (BSCK) together, below. If the condition is violated, the NIC may glitch/ACK. CPUs that operate from pipelined instructions (i.e. 386) or have a cache (i.e. 486) can execute consecutive I/O cycles very quickly. The solution is to delay the execution of consecutive I/O cycles by either breaking the pipeline or forcing the CPU to access outisde it's cache.

Time between Chip Selects



TL/F/8582–A1

14.0 Preliminary Electrical Characteristics

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (PD)	500 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C
ESD rating (R_{ZAP} = 1.5k, C_{ZAP} = 120 pF)	1600V

Preliminary DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified

Symbol	Parameter	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage (Notes 1, 4)	I_{OH} = −20 μA I_{OH} = −2.0 mA	V_{CC} − 0.1 3.5		V V
V_{OL}	Minimum Low Level Output Voltage (Notes 1, 4)	I_{OL} = 20 μA I_{OL} = 2.0 mA		0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage (Note 2)		2.0		V
V_{IH2}	Minimum High Level Input Voltage for RACK, WACK (Note 2)		2.7		V
V_{IL}	Minimum Low Level Input Voltage (Note 2)			0.8	V
V_{IL2}	Minimum Low Level Input Voltage For RACK, WACK (Note 2)			0.6	V
I_{IN}	Input Current	V_I = V_{CC} or GND	−1.0	+1.0	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND	−10	+10	μA
I_{CC}	Average Supply Current (Note 3)	TXCK = 10 MHz RXCK = 10 MHz BSCK = 20 MHz I_{OUT} = 0 μA V_{IN} = V_{CC} or GND		40	mA

Note 1: These levels are tested dynamically using a limited amount of functional test patterns, please refer to AC Test Load.

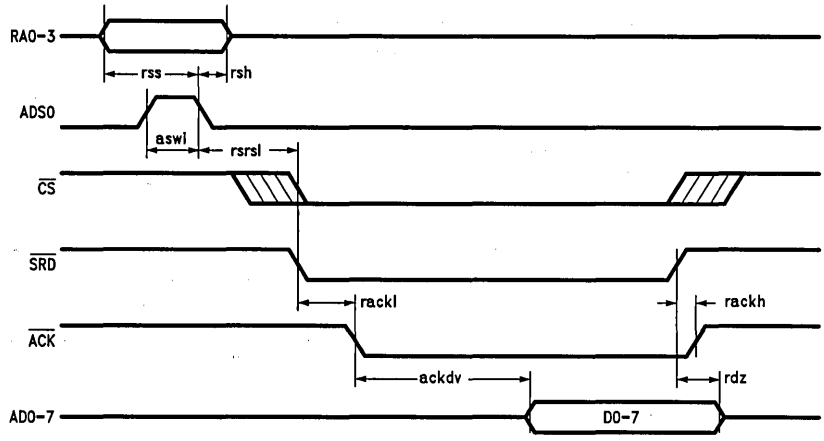
Note 2: Limited functional test patterns are performed at these input levels. The majority of functional tests are performed at levels of 0V and 3V.

Note 3: This is measured with a 0.1 μF bypass capacitor between V_{CC} and GND.

Note 4: The low drive CMOS compatible V_{OH} and V_{OL} limits are not tested directly. Detailed device characterization validates that this specification can be guaranteed by testing the high drive TTL compatible V_{OL} and V_{OH} specification.

15.0 Switching Characteristics AC Specs DP8390D **Note:** All Timing is Preliminary

Register Read (Latched Using ADS0)



TL/F/8582–76

Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	13		ns
aswi	Address Strobe Width In	15		ns
ackdv	Acknowledge Low to Data Valid		55	ns
rdz	Read Strobe to Data TRI-STATE	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Notes 1, 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns
rsrsl	Register Select to Slave Read Low, Latched RS0–3 (Note 2)	10		ns

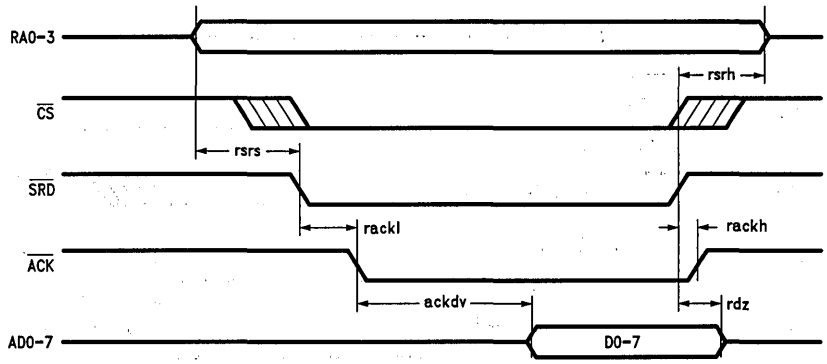
Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SRD}$ are low and the NIC has synchronized to the register access. The NIC will insert an integral number of Bus Clock cycles until it is synchronized. In Dual Bus systems additional cycles will be used for a local or remote DMA to complete. Wait states must be issued to the CPU until $\overline{ACK}$ is asserted low.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SRD}$. If $\overline{CS}$ is asserted after $\overline{SRD}$, rackl is referenced from falling edge of $\overline{CS}$. $\overline{CS}$ can be de-asserted concurrently with $\overline{SRD}$ or after $\overline{SRD}$ is de-asserted.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

Register Read (Non Latched, ADS0 = 1)



TL/F/8582-77

Symbol	Parameter	Min	Max	Units
rsrs	Register Select to Read Setup (Notes 1, 3)	10		ns
rsrh	Register Select Hold from Read	0		ns
ackdv	$\overline{ACK}$ Low to Valid Data		55	ns
rdz	Read Strobe to Data TRI-STATE (Note 2)	15	70	ns
rackl	Read Strobe to $\overline{ACK}$ Low (Note 3)		n*bcyc + 30	ns
rackh	Read Strobe to $\overline{ACK}$ High		30	ns

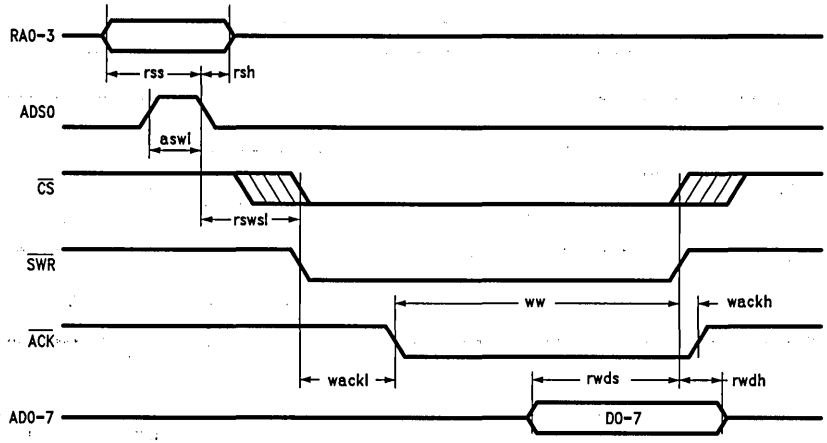
Note 1: rsrs includes flow-through time of latch.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

Note 3: $\overline{CS}$ may be asserted before or after RA0–3, and $\overline{SRD}$, since address decode begins when $\overline{ACK}$ is asserted. If $\overline{CS}$ is asserted after RA0-3, and $\overline{SRD}$, rack1 is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics (Continued)

Register Write (Latched Using ADS0)



TL/F/8582-78

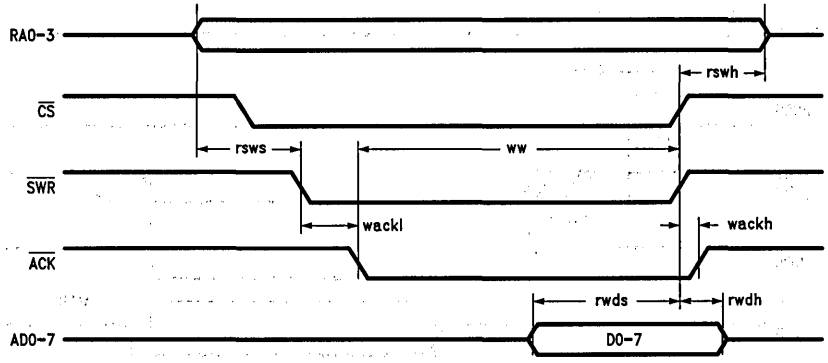
Symbol	Parameter	Min	Max	Units
rss	Register Select Setup to ADS0 Low	10		ns
rsh	Register Select Hold from ADS0 Low	17		ns
aswi	Address Strobe Width In	15		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
ww	Write Strobe Width from $\overline{ACK}$	50		ns
wackh	Write Strobe High to $\overline{ACK}$ High		30	ns
wackl	Write Low to $\overline{ACK}$ Low (Notes 1, 2)		n*bcyc + 30	ns
rswsl	Register Select to Write Strobe Low	10		ns

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus Systems additional cycles will be used for a local DMA or Remote DMA to complete.

Note 2: $\overline{CS}$ may be asserted before or after $\overline{SWR}$. If $\overline{CS}$ is asserted after $\overline{SWR}$, wackl is referenced from falling edge of $\overline{CS}$.

15.0 Switching Characteristics (Continued)

Register Write (Non Latched, ADS0 = 1)



TL/F/8582–79

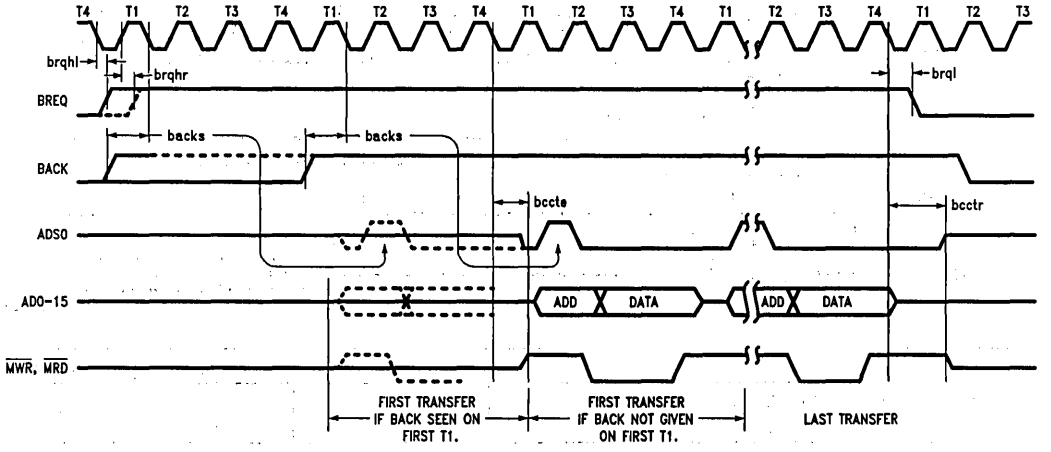
Symbol	Parameter	Min	Max	Units
rsws	Register Select to Write Setup (Note 1)	15		ns
rswh	Register Select Hold from Write	0		ns
rwds	Register Write Data Setup	20		ns
rwdh	Register Write Data Hold	21		ns
wackl	Write Low to $\overline{ACK}$ Low (Note 2)		n*bcyc + 30	ns
wackh	Write High to $\overline{ACK}$ High		30	ns
ww	Write Width from $\overline{ACK}$	50		ns

Note 1: Assumes ADS0 is high when RA0–3 changing.

Note 2: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus systems additional cycles will be used for a local DMA or remote DMA to complete.

15.0 Switching Characteristics (Continued)

DMA Control, Bus Arbitration



TL/F/8582–80

Symbol	Parameter	Min	Max	Units
brqhl	Bus Clock to Bus Request High for Local DMA		43	ns
brqhr	Bus Clock to Bus Request High for Remote DMA		38	ns
brql	Bus Request Low from Bus Clock		55	ns
backs	Acknowledge Setup to Bus Clock (Note 1)	2		ns
bccte	Bus Clock to Control Enable		60	ns
bcctr	Bus Clock to Control Release (Notes 2, 3)		70	ns

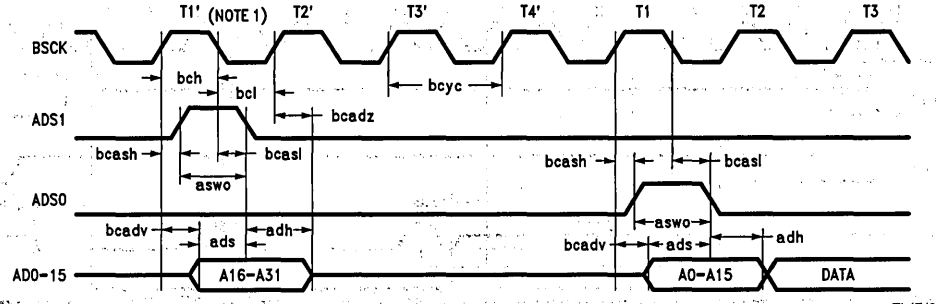
Note 1: BACK must be setup before T1 after BREQ is asserted. Missed setup will slip the beginning of the DMA by four bus clocks. The Bus Latency will influence the allowable FIFO threshold and transfer mode (empty/fill vs exact burst transfer).

Note 2: During remote DMA transfers only, a single bus transfer is performed. During local DMA operations burst mode transfers are performed.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Address Generation



TL/F/8582-81

Symbol	Parameter	Min	Max	Units
bcyc	Bus Clock Cycle Time (Note 2)	50	1000	ns
bch	Bus Clock High Time	22.5		ns
bcl	Bus Clock Low Time	22.5		ns
bcash	Bus Clock to Address Strobe High		34	ns
bcasl	Bus Clock to Address Strobe Low		44	ns
aswo	Address Strobe Width Out	bch		ns
bcadv	Bus Clock to Address Valid		45	ns
bcadz	Bus Clock to Address TRI-STATE (Note 3)	15	55	ns
ads	Address Setup to ADS0/1 Low	bch − 15		ns
adh	Address Hold from ADS0/1 Low	bcl − 5		ns

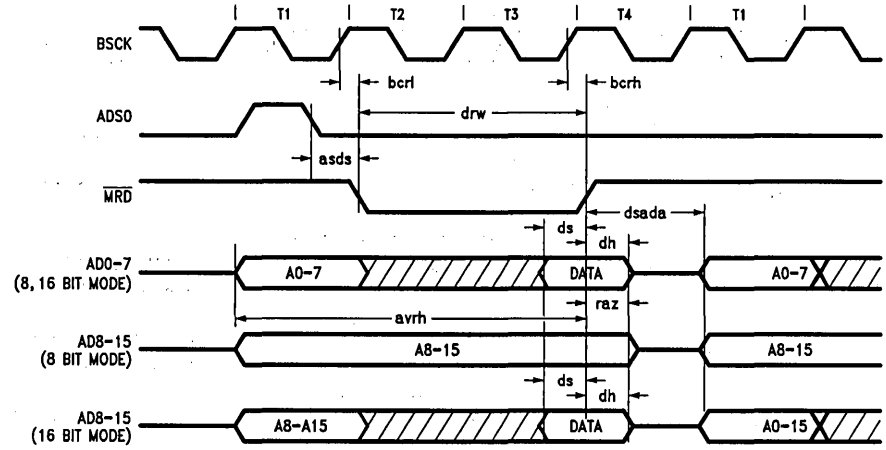
Note 1: Cycles T1', T2', T3', T4' are only issued for the first transfer in a burst when 32-bit mode has been selected.

Note 2: The rate of bus clock must be high enough to support transfers to/from the FIFO at a rate greater than the serial network transfers from/to the FIFO.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Memory Read



TL/F/8582–82

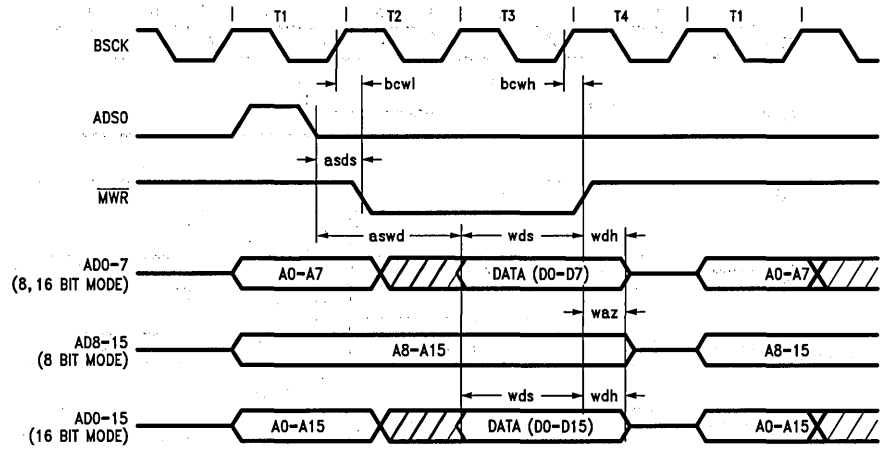
Symbol	Parameter	Min	Max	Units
bcrl	Bus Clock to Read Strobe Low		43	ns
bcrh	Bus Clock to Read Strobe High		40	ns
ds	Data Setup to Read Strobe High	25		ns
dh	Data Hold from Read Strobe High	0		ns
drw	DMA Read Strobe Width Out	2*bcyc − 15		ns
raz	Memory Read High to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
dsada	Data Strobe to Address Active	bcyc − 10		ns
avrh	Address Valid to Read Strobe High	3*bcyc − 15		ns

Note 1: During a burst A8–A15 are not TRI-STATE if byte wide transfers are selected. On the last transfer A8–A15 are TRI-STATE as shown above.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

DMA Memory Write



TL/F/8582-83

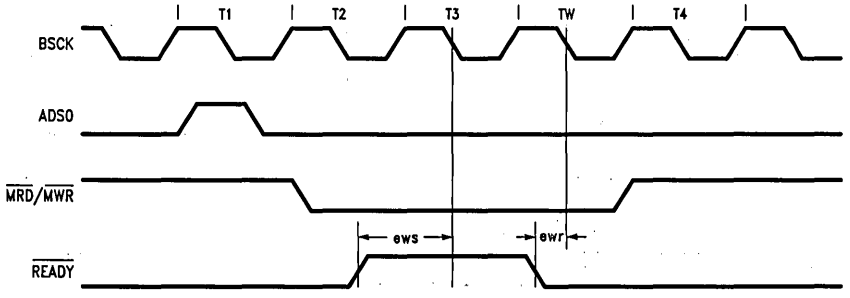
Symbol	Parameter	Min	Max	Units
bcwl	Bus Clock to Write Strobe Low		40	ns
bcwh	Bus Clock to Write Strobe High		40	ns
wds	Data Setup to $\overline{WR}$ High	2*bcyc − 30		ns
wdh	Data Hold from $\overline{WR}$ Low	bch + 7		ns
waz	Write Strobe to Address TRI-STATE (Notes 1, 2)		bch + 40	ns
asds	Address Strobe to Data Strobe		bcl + 10	ns
aswd	Address Strobe to Write Data Valid		bcl + 30	ns

Note 1: When using byte mode transfers A8–A15 are only TRI-STATE on the last transfer, waz timing is only valid for last transfer in a burst.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within bch + 15 ns, enabling other devices to drive these lines with no contention.

15.0 Switching Characteristics (Continued)

Wait State Insertion



TL/F/8582–45

Symbol	Parameter	Min	Max	Units
ews	External Wait Setup to T3 ↓ Clock (Note 1)	10		ns
ewr	External Wait Release Time (Note 1)	15		ns

Note 1: The addition of wait states affects the count of deserialized bytes and is limited to a number of bus clock cycles depending on the bus clock and network rates. The allowable wait states are found in the table below. (Assumes 10 Mbit/sec data rate.)

BUSCK (MHz)	# of Wait States	
	Byte Transfer	Word Transfer
8	0	1
10	0	1
12	1	2
14	1	2
16	1	3
18	2	3
20	2	4

Table assumes 10 MHz network clock.

The number of allowable wait states in byte mode can be calculated using:

$$\#W_{(\text{byte mode})} = \left(\frac{8\ tnw}{4.5\ tbsck} - 1\right)$$

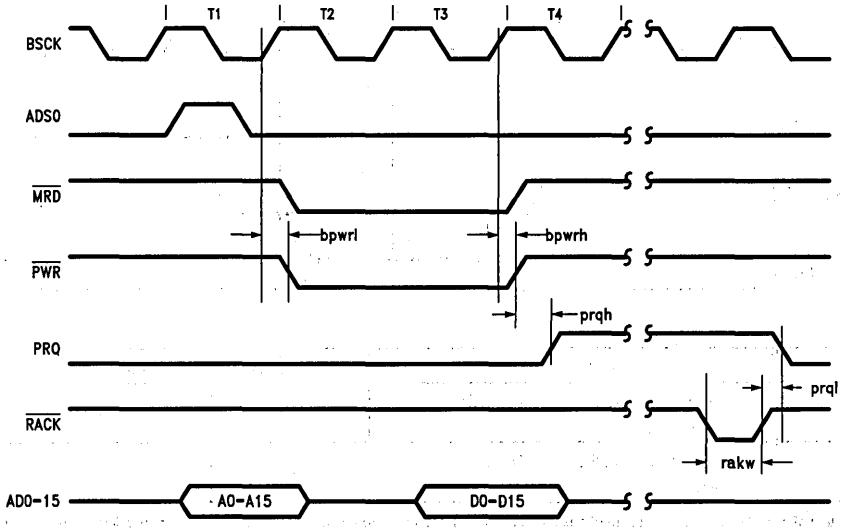
#W = Number of Wait States
tnw = Network Clock Period
tbsck = BSCK Period

The number of allowable wait states in word mode can be calculated using:

$$\#W_{(\text{word mode})} = \left(\frac{5\ tnw}{2\ tbsck} - 1\right)$$

15.0 Switching Characteristics (Continued)

Remote DMA (Read, Send Command)



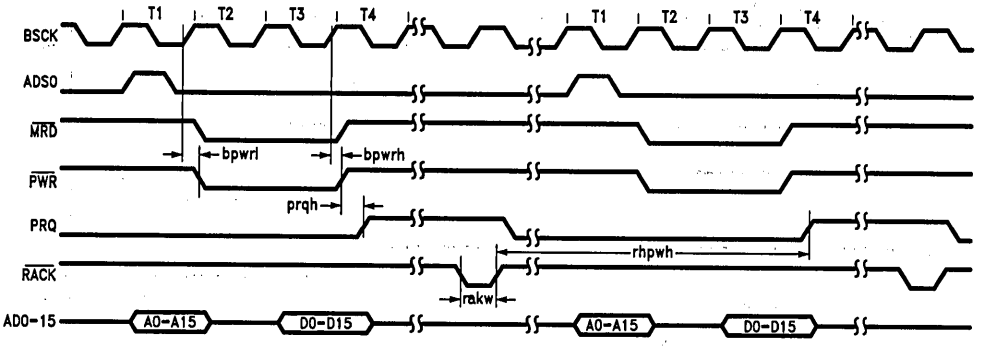
TL/F/8582-84

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		45	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns

Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

15.0 Switching Characteristics (Continued)

Remote DMA (Read, Send Command) Recovery Time



TL/F/8582–85

Symbol	Parameter	Min	Max	Units
bpwrl	Bus Clock to Port Write Low		43	ns
bpwrh	Bus Clock to Port Write High		40	ns
prqh	Port Write High to Port Request High (Note 1)		30	ns
prql	Port Request Low from Read Acknowledge High		45	ns
rakw	Remote Acknowledge Read Strobe Pulse Width	20		ns
rhpwh	Read Acknowledge High to Next Port Write Cycle (Notes 2,3,4)	11		BUSCK

Note 1: Start of next transfer is dependent on where $\overline{\text{RACK}}$ is generated relative to BSCK and whether a local DMA is pending.

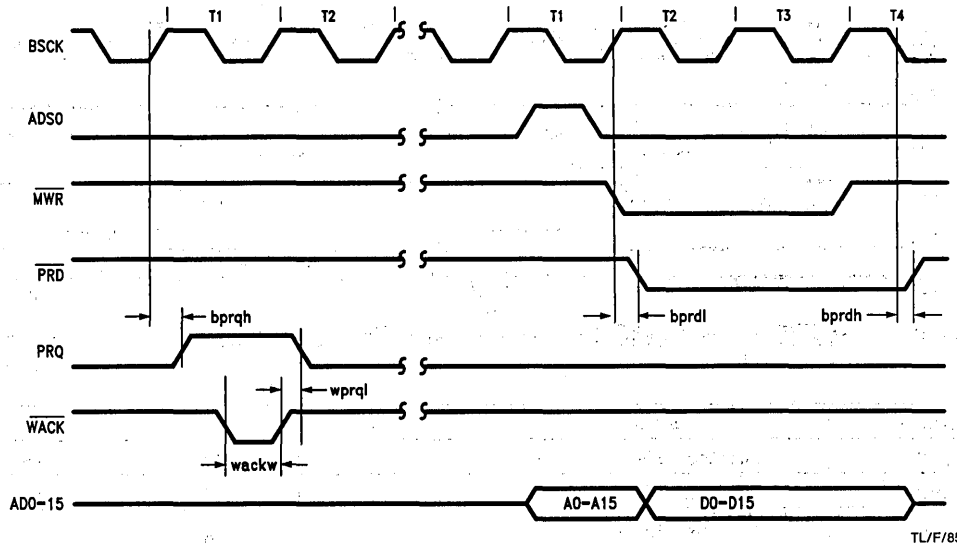
Note 2: This is not a measured value but guaranteed by design.

Note 3: $\overline{\text{RACK}}$ must be high for a minimum of 7 BUSCK.

Note 4: Assumes no local DMA interleave, no $\overline{\text{CS}}$, and immediate BACK.

15.0 Switching Characteristics (Continued)

Remote DMA (Write Cycle)



TL/F/8582-86

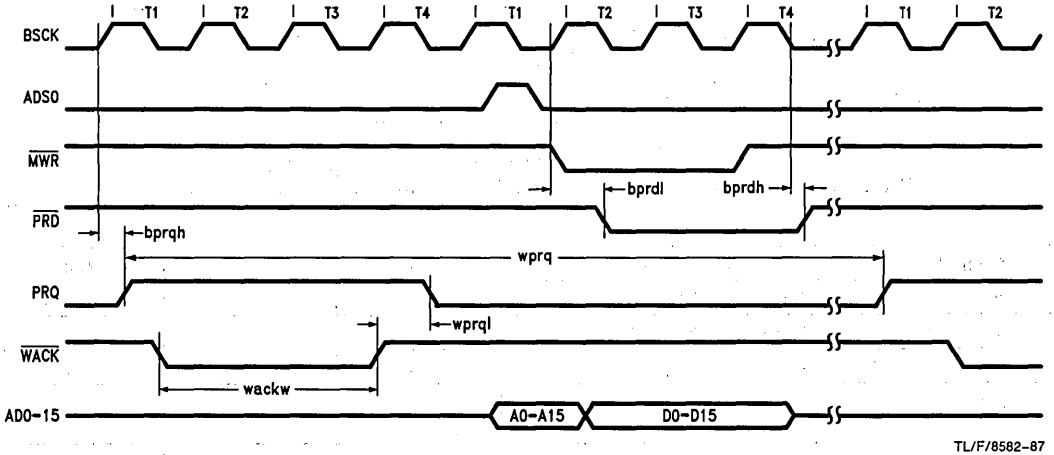
Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		42	ns
wprql	$\overline{WACK}$ to Port Request Low		45	ns
wackw	$\overline{WACK}$ Pulse Width	20		ns
bprdl	Bus Clock to Port Read Low (Note 2)		40	ns
bprdh	Bus Clock to Port Read High		40	ns

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BUSCK and whether a local DMA is pending.

15.0 Switching Characteristics (Continued)

Remote DMA (Write Cycle) Recovery Time



TL/F/8582–87

Symbol	Parameter	Min	Max	Units
bprqh	Bus Clock to Port Request High (Note 1)		40	ns
wprql	$\overline{WACK}$ to Port Request Low		45	ns
wackw	$\overline{WACK}$ Pulse Width	20		ns
bprdl	Bus Clock to Port Read Low (Note 2)		40	ns
bprdh	Bus Clock to Port Read High		40	ns
wprq	Remote Write Port Request to Port Request Time (Notes 3,4,5)	12		BUSCK

Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BUSCK and whether a local DMA is pending.

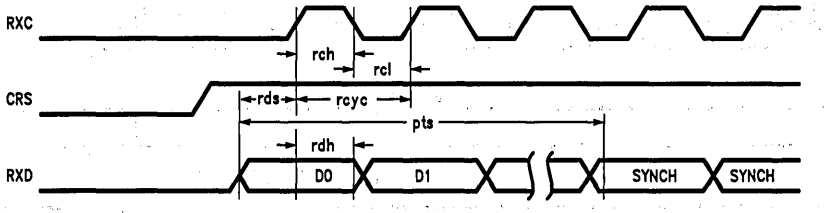
Note 3: Assuming wackw < 1 BUSCK, and no local DMA interleave, no $\overline{CS}$, immediate $\overline{BACK}$, and $\overline{WACK}$ goes high before T4.

Note 4: $\overline{WACK}$ must be high for a minimum of 7 BUSCK.

Note 5: This is not a measured value but guaranteed by design.

15.0 Switching Characteristics (Continued)

Serial Timing—Receive (Beginning of Frame)



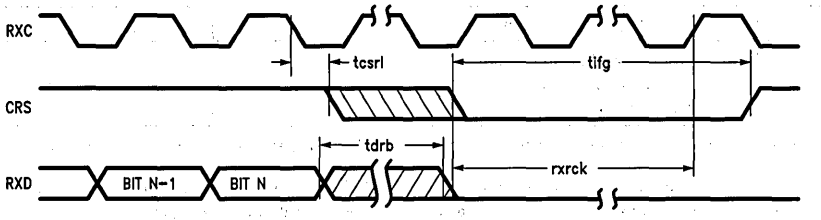
TL/F/8582–88

Symbol	Parameter	Min	Max	Units
rch	Receive Clock High Time	40		ns
rcl	Receive Clock Low Time	40		ns
rcyc	Receive Clock Cycle Time	800	1200	ns
rds	Receive Data Setup Time to Receive Clock High (Note 1)	20		ns
rdh	Receive Data Hold Time from Receive Clock High	17		ns
pts	First Preamble Bit to Synch (Note 2)	8		rcyc cycles

Note 1: All bits entering NIC must be properly decoded, if the PLL is still locking, the clock to the NIC should be disabled or CRS delayed. Any two sequential 1 data bits will be interpreted as Synch.

Note 2: This is a minimum requirement which allows reception of a packet.

Serial Timing—Receive (End of Frame)



TL/F/8582–89

Symbol	Parameter	Min	Max	Units
rxrck	Minimum Number of Receive Clocks after CRS Low (Note 1)	5		rcyc cycles
tdrb	Maximum of Allowed Dribble Bits/Clocks (Note 2)		3	rcyc cycles
tifg	Receive Recovery Time (Notes 4,5)		40	rcyc cycles
tcrsl	Receive Clock to Carrier Sense Low (Note 3)	0	1	rcyc cycles

Note 1: The NIC requires a minimum number of receive clocks following the de-assertion of carrier sense (CRS). These additional clocks are provided by the DP8391 SNI. If other decoder/PLLs are being used additional clocks should be provided. Short clocks or glitches are not allowed.

Note 2: Up to 5 bits of dribble bits can be tolerated without resulting in a receive error.

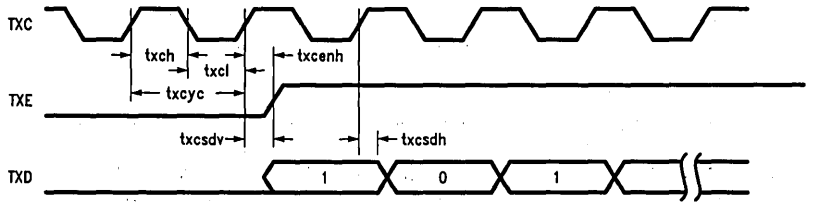
Note 3: Guarantees to only load bit N, additional bits up to tdrb can be tolerated.

Note 4: This is the time required for the receive state machine to complete end of receive processing. This parameter is not measured but is guaranteed by design. This is not a measured parameter but is a design requirement.

Note 5: CRS must remain de-asserted for a minimum of 2 RXC cycles to be recognized as end of carrier.

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Beginning of Frame)

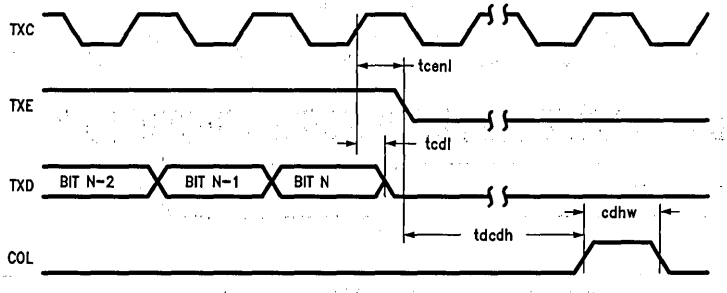


TL/F/8582–90

Symbol	Parameter	Min	Max	Units
txch	Transmit Clock High Time	36		ns
txcl	Transmit Clock Low Time	36		ns
txcyc	Transmit Clock Cycle Time	800	1200	ns
txcenh	Transmit Clock to Transmit Enable High (Note 1)		48	ns
txcsdv	Transmit Clock to Serial Data Valid		67	ns
txcsdh	Serial Data Hold Time from Transmit Clock High	10		ns

Note 1: The NIC issues TXEN coincident with the first bit of preamble. The first bit of preamble is always a 1.

Serial Timing—Transmit (End of Frame, CD Heartbeat)



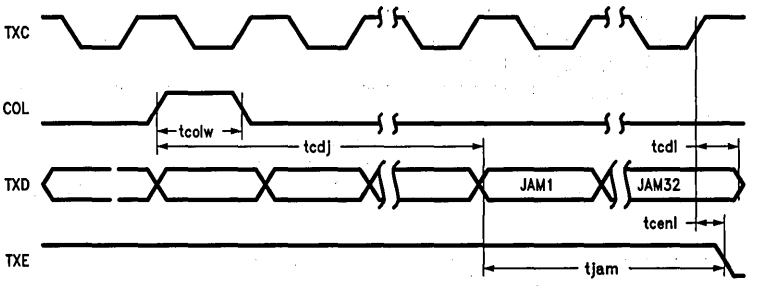
TL/F/8582–91

Symbol	Parameter	Min	Max	Units
tcdl	Transmit Clock to Data Low		55	ns
tcenl	Transmit Clock to TXEN Low		55	ns
tdcdh	TXEN Low to Start of Collision Detect Heartbeat (Note 1)	0	64	txcyc cycles
cdhw	Collision Detect Width	2		txcyc cycles

Note 1: If COL is not seen during the first 64 TX clock cycles following de-assertion of TXEN, the CDH bit in the TSR is set.

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Collision)



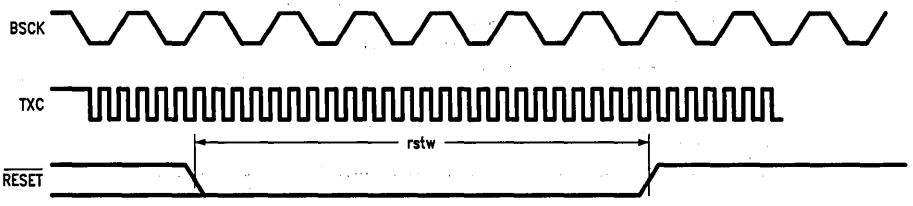
TL/F/8582–92

Symbol	Parameter	Min	Max	Units
tcolw	Collision Detect Width	2		txcyc cycles
tcdj	Delay from Collision to First Bit of Jam (Note 1)		8	txcyc cycles
tjam	Jam Period (Note 2)		32	txcyc cycles

Note 1: The NIC must synchronize to collision detect. If the NIC is in the middle of serializing a byte of data the remainder of the byte will be serialized. Thus the jam pattern will start anywhere from 1 to 8 TXC cycles after COL is asserted.

Note 2: The NIC always issues 32 bits of jam. The jam is all 1's data.

Reset Timing



TL/F/8582–93

Symbol	Parameter	Min	Max	Units
rstw	Reset Pulse Width (Note 1)	8		BSCK Cycles or TXC Cycles (Note 2)

Note 1: The RESET pulse requires that BSCK and TXC be stable. On power up, $\overline{\text{RESET}}$ should not be raised until BSCK and TXC have become stable. Several registers are affected by $\overline{\text{RESET}}$. Consult the register descriptions for details.

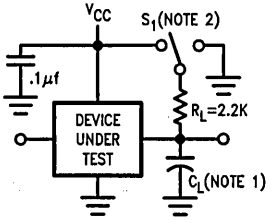
Note 2: The slower of BSCK or TXC clocks will determine the minimum time for the $\overline{\text{RESET}}$ signal to be low.

If BSCK < TXC then $\overline{\text{RESET}}$ = 8 × BSCK

If TXC < BSCK then $\overline{\text{RESET}}$ = 8 × TXC

AC Timing Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	5 ns
Input and Output Reference Levels	1.3V
TRI-STATE Reference Levels	Float (ΔV) ±0.5V
Output Load (See Figure below)	



TL/F/8582–94

Note 1: C_L = 50 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.

S1 = V_{CC} for V_{OL} test.

S1 = GND for V_{OH} test.

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active high to High Impedance measurements.

Pin Capacitance T_A = 25°C, f = 1 MHz

Parameter	Description	Typ	Max	Unit
C_{IN}	Input Capacitance	7	15	pF
C_{OUT}	Output Capacitance	7	15	pF

Note: This parameter is sampled and not 100% tested.

DERATING FACTOR

Output timings are measured with a purely capacitave load for 50 pF. The following correction factor can be used for other loads:

$C_L \geq 50$ pf: +0.3 ns/pF (for all outputs except TXE, TXD, and LBK)

National Semiconductor

PRELIMINARY

DP83910A
CMOS Serial Network Interface

General Description

The DP83910A CMOS Serial Network Interface (SNI) is a direct-pin equivalent of the bipolar DP8391 SNI and provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Thin-Ethernet type local area networks. The SNI interfaces the DP8390 Network Interface Controller (NIC) to the DP8392 CTI or an Ethernet transceiver cable. When transmitting, the SNI converts non-return-to-zero (NRZ) data from the controller into Manchester data and sends the converted data differentially to the transceiver. Conversely, when receiving, a Phase Lock Loop decodes the 10 Mbit/s data from the transceiver into NRZ data for the controller.

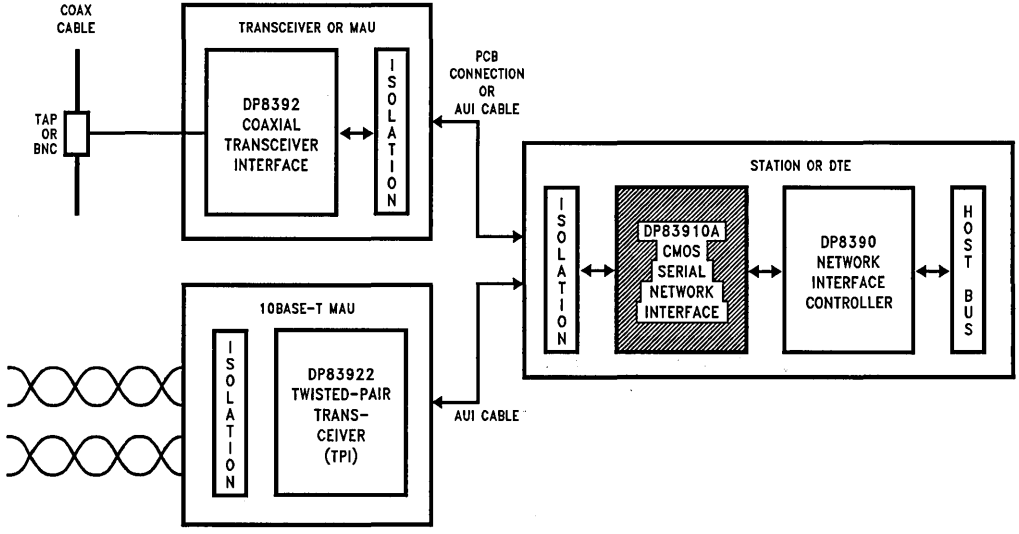
The DP83910A operates in conjunction with the DP8392 Coaxial Transceiver Interface (CTI) and the DP8390 Network Interface Controller (NIC) to form a three-chip set that implements a complete IEEE 802.3 compatible network as shown below. The DP83910A is a functionally complete Manchester encoder/decoder including a balanced driver and receiver, on-board crystal oscillator, collision signal translator, and a diagnostic loopback feature. The DP83910A, fabricated CMOS, typically consumes less than 70 mA of current. However, as a result of being CMOS, the DP83910A's differential signals must be isolated in both Ethernet and thin wire Ethernet.

Features

- Compatible with Ethernet I, IEEE 802.3; 10BASE5, 10BASE2, and 10BASE-T
- Designed to interface with 10BASE-T transceivers
- Functional and pin-out duplicate of the DP8391
- 10 Mbits/s Manchester encoding/decoding with receive clock recovery
- Requires no precision components
- Decodes Manchester data with up to 18 ns of jitter
- Loopback capability for diagnostics
- Externally selectable half or full step modes of operation at transmit output
- Squelch circuitry at the receive and collision inputs to reject noise
- TTL/MOS compatible controller interface

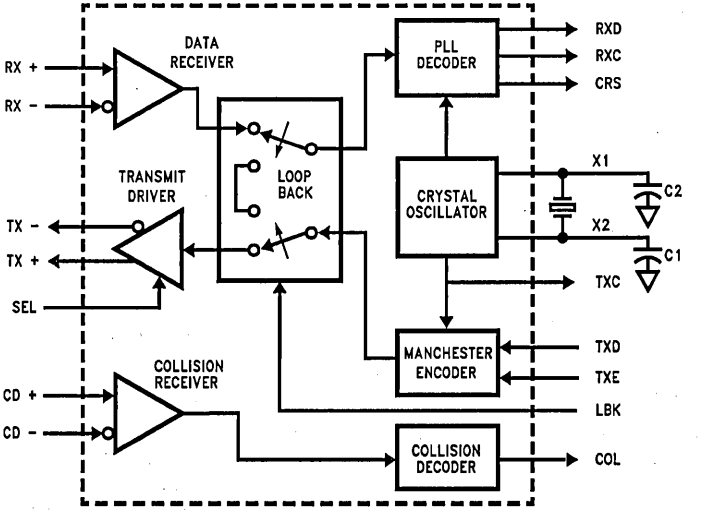
1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Thin-Ethernet/10 BaseT Local Area Network Chip Set



TL/F/9365–1

2.0 Block Diagram



TL/F/9365-2

3.0 Functional Description

The DP83910A consists of five main logical blocks:

a) The oscillator generates the 10 MHz transmit clock signal for system timing.

b) The Manchester encoder accepts NRZ data from the controller, encodes the data to Manchester, and transmits it differentially to the transceiver, through the differential transmit driver.

c) The Manchester decoder receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends it to the controller.

d) The collision translator indicates to the controller the presence of a valid 10 MHz collision signal to the PLL.

e) The loopback circuitry, when asserted, routes the data from the Manchester encoder back to the PLL decoder.

3.1 OSCILLATOR

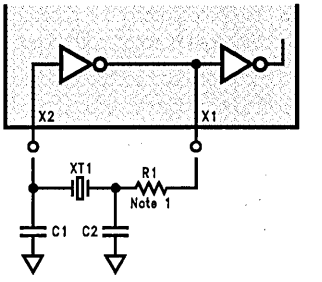
The oscillator is controlled by a 20 MHz parallel resonant crystal connected between X1 and X2 or by an external clock on X1. The 20 MHz output of the oscillator is divided by 2 to generate the 10 MHz transmit clock for the controller. The oscillator also provides internal clock signals to the encoding and decoding circuits.

If a crystal is connected to the DP83910A, it is recommended that the circuit shown in *Figure 1* be used and that the components used meet the following:

Crystal XT1: AT cut parallel resonant crystal
Series Resistance: $\leq 10\Omega$
Specified Load Capacitance: 13.5 pF
Accuracy: 0.005% (50 ppm)

C1, C2: Load Capacitor, 27 pF.

The resistor, R1, in *Figure 1* may be required in order to minimize frequency drift due to changes in the V_{CC} supply voltage. If R1 is required, it's value must be carefully selected. R1 decreases the loop gain. Thus, if R1 is made too large, the loop gain will be greatly reduced and the crystal will not oscillate. If R1 is made too small, normal variations in the V_{CC} may cause the oscillation frequency to drift out of specification. As the first rule of thumb, the value of R1



TL/F/9365-15

Note 1: The resistor R1 may be required in order to minimize frequency drift due to changes in the V_{CC}. See text description.

FIGURE 1. Crystal Connection to DP83910A
(see text for component values)

should be made equal to five times the motional resistance of the crystal.

The motional resistance of 20 MHz crystals is usually in the range of 10Ω to 30Ω. This implies that a reasonable value for R1 should be in the range of 50Ω–150Ω.

The decision of whether or not to include R1 should be based upon measured variations of crystal frequency as each of the circuit parameters is varied.

According to the IEEE 802.3 standard, the entire oscillator circuit (crytsal and amplifier) must be accurate to 0.01%. When using a crystal, the X1 pin is not guaranteed to provide a TTL compatible logic output, and should not be used to drive external standard logic. If additional logic needs to be driven, then an external oscillator should be used, as described in the following.

3.2 OSCILLATOR MODULE OPERATION

If the designer wishes to use a crystal clock oscillator, one that provides the following should be employed:
1) TTL or CMOS output with a 0.01% frequency tolerance
2) 40%–60% duty cycle
3) ≥ 2 TTL load output drive (I_{OL} = 3.2 mA)

3.0 Functional Description (Continued)

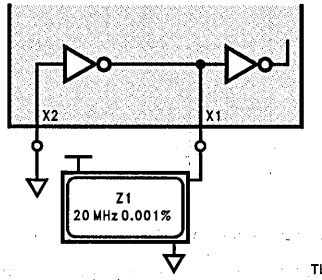
The circuit is shown in *Figure 2*. (Additional output drive may be necessary if the oscillator must also drive other components.) When using a clock oscillator it is still recommended that the designer connect the oscillator output to the X1 pin and tie the X2 pin to ground.

3.3 MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The encoder begins operation when the Transmit Enable input (TXE) goes high and converts clock and NRZ data to Manchester data for the transceiver. For the duration of TXE remaining high, the Transmitted Data (TXD) is encoded for the transmit-driver pair (TX±). TXD must be valid on the rising edge of Transmit Clock (TXC). Transmission ends when TXE goes low. The last transition is always positive; it occurs at the center of the bit cell if the last bit is a one, or at the end of the bit cell if the last bit is a zero.

The differential transmit pair from the secondary of the isolation transformer drives up to 50 meters of twisted pair AUI cable. These outputs are source followers which require two 270Ω pull-down resistors to ground.

The DP83910A allows both half-step and full-step to be compatible with Ethernet I and IEEE 802.3. With the SEL pin low (for Ethernet I), transmit+ is positive with respect to transmit− during idle; with SEL high (for IEEE 802.3), transmit+ and transmit− are equal in the idle state. This provides zero differential voltage to operate with transformer coupled loads.



TL/F/9365–16

FIGURE 2. DP83910A Connection for Oscillator Module

3.4 MANCHESTER DECODER

The decoder consists of a differential receiver and a PLL to separate Manchester encoded data stream into clock signals and NRZ data. The differential input must be externally terminated with two 39Ω resistors connected in series if the standard 78Ω transceiver drop cable is used; in Thin-Ethernet applications, these resistors are optional. To prevent noise from falsely triggering the decoder, a squelch circuit at the input rejects signals with levels less than −175 mV. Once the input exceeds the squelch requirements, Carrier Sense (CRS) is asserted. Receive data (RXD) and receive clock (RXC) become valid typically within 6 bit times. The DP83910A may tolerate bit jitter up to 18 ns in the received data.

The decoder detects the end of a frame when no more midbit transitions are detected. Within one and a half bit times after the last bit, carrier sense is de-asserted. Receive clock stays active for five more bit times after CRS goes low to guarantee the receive timings of the DP8390 NIC.

3.5 COLLISION TRANSLATOR

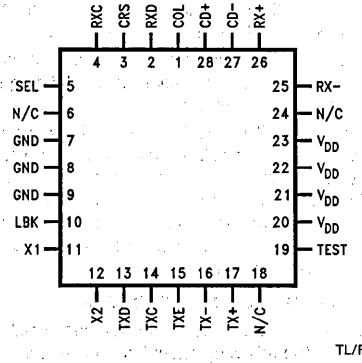
When the Ethernet transceiver (DP8392 CTI) detects a collision, it generates a 10 MHz signal to the differential collision inputs (CD±) of the DP83910A. When these inputs are detected active, the DP83910A translates the 10 MHz signal to an active high level for the controller. The controller uses this signal to back off its current transmission and reschedule another one.

The collision differential inputs are terminated the same way as the differential receive inputs. The squelch circuitry is also similar, rejecting pulses with levels less than −175 mV.

3.6 LOOPBACK FUNCTIONS

When the Loopback input (LBK) is asserted high, the DP83910A redirects its transmitted data back into its receive path. This feature provides a convenient method for testing both chip and system level integrity. The transmit driver and receive input circuitry are disabled in loopback mode.

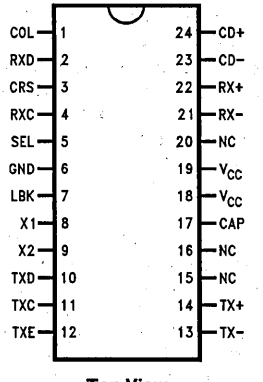
4.0 Connection Diagrams



TL/F/9365–17

Top View

Order Number DP83910AV
See NS Package Number V28A

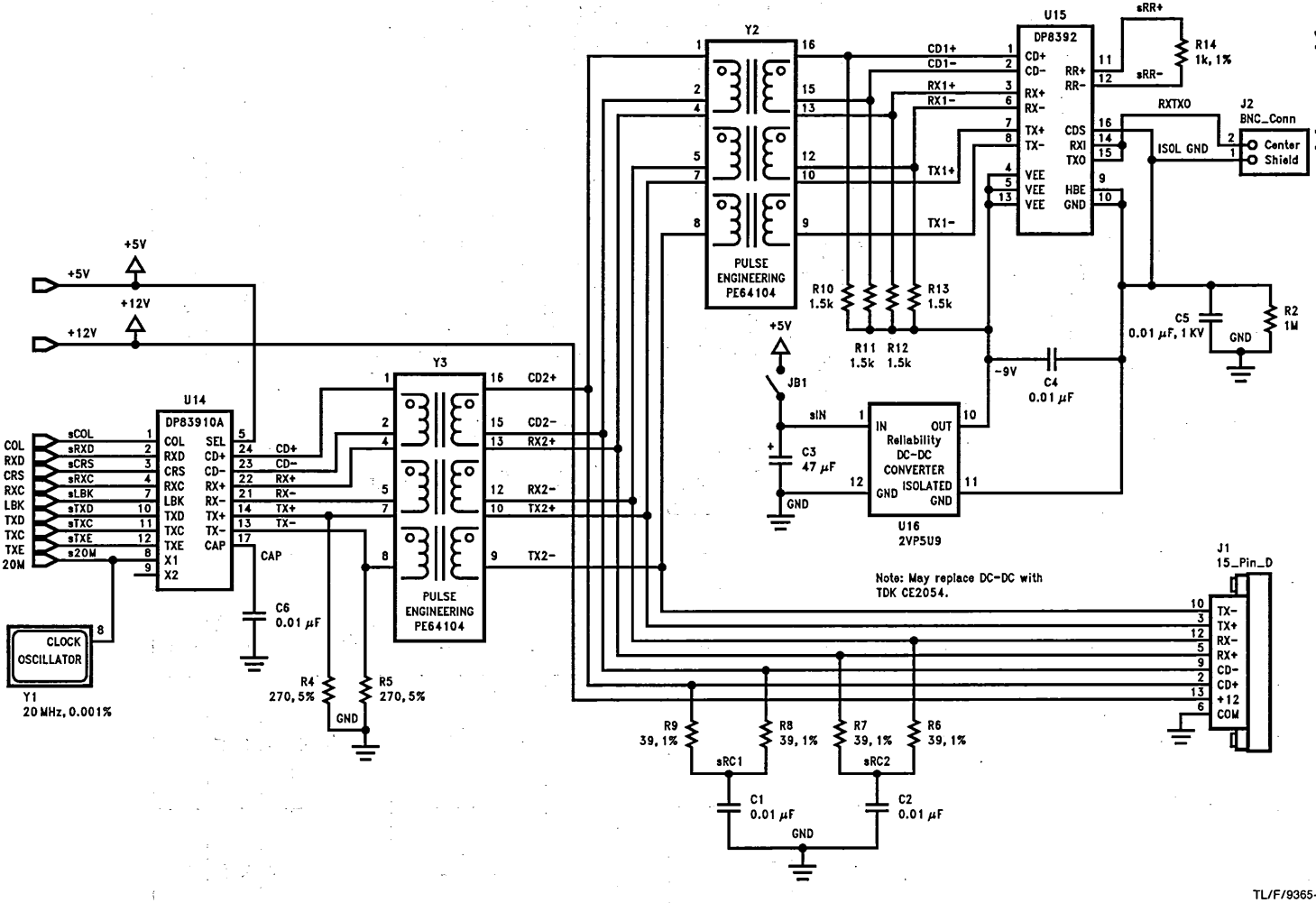


TL/F/9365–18

Top View

Order Number DP83910AN
See NS Package Number N24C

Interface for Ethernet and Thin Wire Ethernet Using Single Jumper for Thin/Thick Selection



TL/F/9365-3

6.0 Pin Descriptions

24-Pin DIP	28-Pin PCC	Name	I/O	Description
1	1	COL	O	**COLLISION DETECT OUTPUT:** Generates an active high signal when 10 MHz collision signal is detected.
2	2	RXD	O	**RECEIVE DATA OUTPUT:** NRZ data output from the PLL. This signal must be sampled on the rising edge of receive clock.
3	3	CRS	O	**CARRIER SENSE:** Asserted on the first valid high-to-low transition on the RX± pair. Remains active until 1.5 bit times after the last bit in data.
4	4	RXC	O	**RECEIVE CLOCK:** The receive clock from the Manchester data after the PLL has locked. Remains active 5 bit times after deasserting CRS.
5	5	SEL	I	**MODE SELECT:** When high, transmit+ and transmit− are the same voltage in the idle state. When low, transmit+ is positive with respect to transmit− in the idle state, at the transformer's primary.
6	7 8 9	V_{SS} V_{SS} V_{SS}		**GROUND PIN**
7	10	LBK	I	**LOOPBACK:** When high, the loopback mode is enabled.
8	11	X1	I	**CRYSTAL OR EXTERNAL OSCILLATOR INPUT**
9	12	X2	O	**CRYSTAL FEEDBACK OUTPUT:** Used in crystal connections only. Connected to ground when using an external oscillator.
10	13	TXD	I	**TRANSMIT DATA INPUT:** NRZ data input from the controller. The data is combined with the transmit clock to produce Manchester data. TXD is sampled on the rising edge of transmit clock.
11	14	TXC	O	**TRANSMIT CLOCK:** The 10 MHz clock derived from the 20 MHz oscillator.
12	15	TXE	I	**TRANSMIT ENABLE:** The encoder begins operation when this input is asserted high.
13 14	16 17	TX− TX+	O	**TRANSMIT OUTPUT:** Differential line driver which sends the encoded data to the transceiver. The outputs are source followers which require 270Ω pull-down resistors.
15	6	NC		**NO CONNECTION:** This may be tied to V_{SS} for the PLCC version to be compatible with the DP8391.
16	18	NC		**NO CONNECTION**
17	19	TEST	I	**FACTORY TEST INPUT:** Used to check the chip's internal functions. May be tied low or have a 0.01 μf bypass capacitor to ground (for compatibility with the bipolar DP8391) during normal operation.
18 19	20 21 22 23	V_{DD} V_{DD} V_{DD} V_{DD}		**POWER CONNECTION**
20	24	NC		**NO CONNECTION**
21 22	25 26	RX− RX+	I	**RECEIVE INPUT:** Differential receive input pair from the transceiver.
23 24	27 28	CD− CD+	I	**COLLISION INPUT:** Differential collision pair input from the transceiver.

7.0 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} +0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} +0.5V
Differential Input Voltage	−5.5 to +16V
Differential Output Voltage	0 to 16V
Power Dissipation	500 mW
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	260°C
ESD (R_{ZAP} = 1.5 kΩ, C_{ZAP} = 120 pF)	≥2 kV (Pin 4 = 1.5 kV)

Note: *Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

***Note:** An asterisk following a parameter's symbol indicates that the parameter has been characterized but not tested.

Note: All specifications in this datasheet are valid only if the mandatory isolation is employed and all differential signals are taken to exist at the AUI side of the pulse transformer.

8.0 DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5%

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Controller Interface Pins (COL, RXD, CRS, RXC, SEL, LBK, TXD, TXC and TXE)						
V_{IH}	Input High Voltage		2.0			V
V_{IL}	Input Low Voltage				0.8	V
I_{IN}	Input Leakage	V_{IN} = V_{CC} or GND	−1.0		1.0	μA
V_{OH}	Output High Voltage	(TTL) I_{OH} = 2.0 mA (CMOS) I_{OH} = 20 μA	3.5 V_{CC} − 0.1			V V
V_{OL}	Output Low Voltage	(TTL) I_{OL} = 2.0 mA (CMOS) I_{OL} = 20 μA			0.4 0.1	V V
I_{CCO}	Operating V_{CC} Supply Current (Note 1)	10 Mbit/sec			70	mA
I_{CCS}	Stand By V_{CC} Supply Current (Note 2)	10 Mbit/sec			65	mA
Differential Pins (TX±, RX±, and CD±)						
V_{OD}	Diff. Output Voltage (TX±)	78Ω Termination, and 270Ω from each to GND *(Figure 4)*	±550		±1200	mV
V_{OB}*	Diff. Output Voltage Imbalance (TX±)	78Ω Termination, and 270Ω from each to GND *(Figure 4)*		40		mV
V_U*	Undershoot Voltage (TX±)	78Ω Termination, and 270Ω from each to GND *(Figure 4)*		100		mV
V_{DS}	Diff. Squelch Threshold (RX± and CD±)		−175		−300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± and CD±) (Note 3)		0		5.5	V
Oscillator Pins (X1 and X2)						
V_{IH}	X1 Input High Voltage	X1 is connected to an oscillator, and X2 is grounded	2.0			V
V_{IL}	X1 Input Low Voltage	X1 is connected to an oscillator, and X2 is grounded			0.8	V
I_{OSC}	X1 Input Current	X1 is connected to an oscillator, and X2 is grounded. V_{IN} = V_{CC} or GND	−2		+2	mA

Note 1: This measurement was made while the DP83910A was undergoing transmission, reception, and collision detection. Also, this value was not measured instantaneously, but averaged over a span of several milliseconds. (V_{IN} = 2.4V or 0.4V and I_o = 0 mA).

Note 2: This measurement was made while the DP83910A was sitting idle with TXE low. Also, this value was not measured instantaneously, but averaged over a span of several milliseconds. (V_{IN} = 2.4V or 0.4V and I_o = 0 mA).

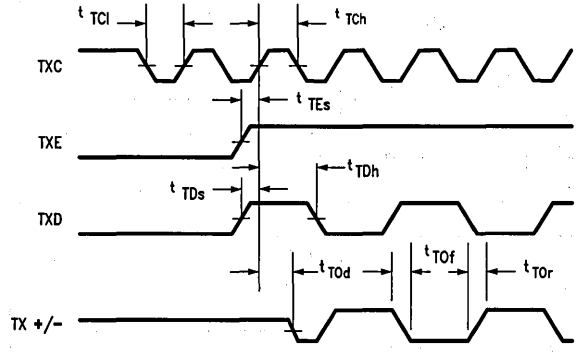
Note 3: This parameter is guaranteed by design and is not tested.

9.0 Switching Characteristics $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$

Oscillator Specification

Symbol	Parameter	Min	Max	Units
t_{XTH}	X1 to Transmit Clock High	5	30	ns
t_{XTL}	X1 to Transmit Clock Low	5	30	ns

Transmit Timing (Start of Packet)



TL/F/9365–4

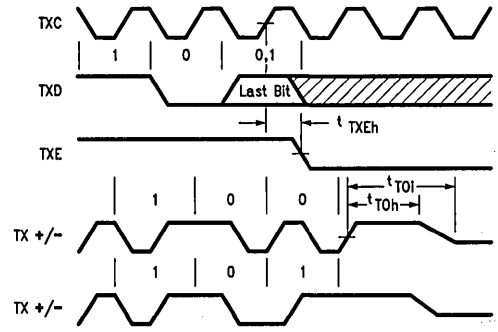
Transmit Specifications (Start of Packet)

Symbol	Parameter	Min	Max	Units
t_{TCh}	Transmit Clock High Time (Note 1)	40	60	ns
t_{TCl}	Transmit Clock Low Time (Note 1)	40		ns
t_{TCc}*	Transmit Clock Cycle Time (Note 1)	99.99	100.01	ns
t_{TCr}*	Transmit Clock Rise Time (20% to 80%) (C_L = 30 pF)		8	ns
t_{TCf}*	Transmit Clock Fall Time (80% to 20%) (C_L = 30 pF)		8	ns
t_{TEs}	Transmit Enable Setup Time to Rising Edge of TXC (Note 1)	20		ns
t_{TDs}	Transmit Data Setup Time from Rising Edge of TXC (Note 1)	20		ns
t_{TDh}	Transmit Data Hold Time from Rising Edge of TXC	0		ns
t_{TOd}	Transmit Output Delay from Rising Edge of TXC (Note 1)		65	ns
t_{TOf}*	Transmit Output Fall Time (80% to 20%)		7	ns
t_{TOr}*	Transmit Output Rise Time (20% to 80%)		7	ns
t_{TOj}*	Transmit Output Jitter	0.5 Typical		ns

Note 1: This parameter is measured using the fifty percent point of each clock edge.

9.0 Switching Characteristics (Continued)

Transmit Timing (End of Packet)

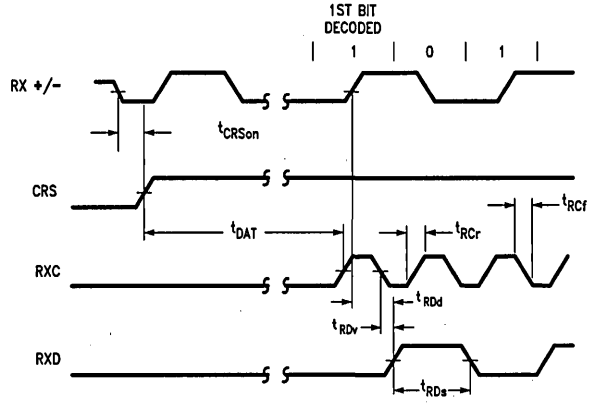


TL/F/9365–5

Transmit Specifications (End of Packet)

Symbol	Parameter	Min	Max	Units
t_{TXEh}	Transmit Enable Hold Time from Rising Edge of TXC	0		ns
t_{TOh}	Transmit Output High before Idle (Half Step)	200		ns
t_{TOi}*	Transmit Output Idle Time (Half Step)		8000	ns

Receive Timing (Start of Packet)



TL/F/9365–6

Receiver Specifications (Start of Packet)

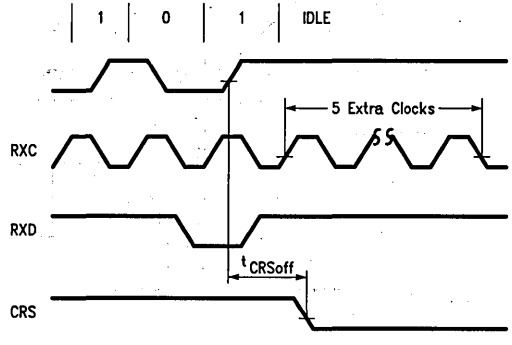
Symbol	Parameter	Min	Max	Units
t_{RCd}	Receive Clock Duty Cycle (Note 1)	40	60	%
t_{RCr}*	Receive Clock Rise Time (20% to 80%, C_{TL} = 30 pF)		7	ns
t_{RCf}*	Receive Clock Fall Time (80% to 20%, C_{TL} = 30 pF)		7	ns
t_{CRSon}	Carrier Sense Turn On Delay		70	ns
t_{DAT}	Decoder Acquisition Time		700	ns
t_{RDd}	Receive Data Output Delay		150	ns
t_{RDs}	Receive Data Output Stable after Going Valid	90		ns
t_{Dtor}	Differential Inputs Turn-On Pulse (Note 2)	30		ns
t_{RDV}	Receive Data Output Valid from Falling Edge of RXC		10	ns

Note 1: This parameter is measured using the fifty percent point of each clock edge.

Note 2: This parameter was characterized with a differential input of −375 mV on the receive pair inputs.

9.0 Switching Characteristics (Continued)

Receive Timing (End of Packet)



TL/F/9365–7

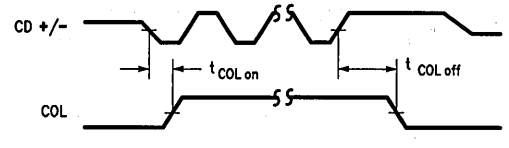
Receiver Specifications (End of Packet)

Symbol	Parameter	Min	Max	Units
t_{CRSoff}	Carrier Sense Turn Off Delay (Note 1)		155	ns
t_{RXCh}	Minimum Number of RXCs after CRS Low (Note 2)	5		Bit Times

Note 1: When CRS goes low, it will go low a minimum of 2 receive clocks.

Note 2: The DP8390 Network Interface Controller (NIC) requires a minimum of 5 receive clocks after CRS goes low to function properly.

Collision Timing



TL/F/9365–8

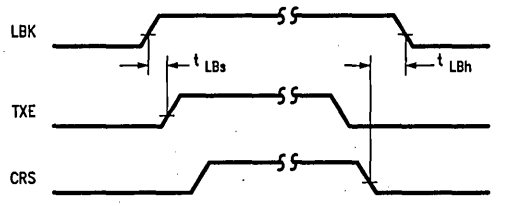
Collision Specifications

Symbol	Parameter	Min	Max	Units
t_{COLon}	Collision Turn On Delay		60	ns
t_{COLoff}	Collision Turn Off Delay		350	ns
t_{Dtoc}	Differential Inputs Turn-On Pulse (Squelch, Note 1)	30		ns

Note 1: This parameter was characterized with a differential input of −375 mV on the collision input pair.

9.0 Switching Characteristics (Continued)

Loopback Timing



TL/F/9365–9

Loopback Specifications

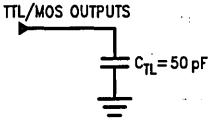
Symbol	Parameter	Min	Max	Units
t_{LBs}	Loopback Setup Time (Note 1)	50		ns
t_{LBh}	Loopback Hold Time (Note 1)	1000		ns

Note 1: This parameter is guaranteed by design and is not tested.

AC Timing Test Conditions

All specifications are valid only if the mandatory isolation is employed and all differential signals are taken to be at the AUI side of the pulse tranformer.

Input Pulse Levels (TTL/CMOS)	GND to 3.0V
Input Rise and Fall Times (TTL/CMOS)	5 ns
Input and Output Reference Levels (TTL/CMOS)	1.3V
Input Pulse Levels (Diff.)	−350 to −1315 mV
Input and Output Reference Levels (Diff.)	50% Point of the Differential

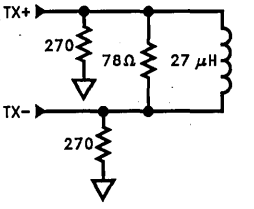


TL/F/9365–10

FIGURE 3

Capacitance T_A = 25°C, f = 1 MHz

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	7	pF
C_{OUT}	Output Capacitance	7	pF



TL/F/9365–12

FIGURE 4

Note: In the above diagram, the TX+ and TX− signals are taken from the AUI side of the isolation (pulse transformer). The pulse transformer used for all testing is the Pulse Engineering PE64103.

National Semiconductor

DP8391A/NS32491A Serial Network Interface

General Description

The DP8391A Serial Network Interface (SNI) provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Cheapernet type local area networks. The SNI interfaces the DP8390 Network Interface Controller (NIC) to the Ethernet transceiver cable. When transmitting, the SNI converts non-return-to-zero (NRZ) data from the controller and clock pulses into Manchester encoding and sends the converted data differentially to the transceiver. The opposite process occurs on the receive path, where a digital phase-locked loop decodes 10 Mbit/s signals with as much as ±18 ns of jitter.

The DP8391A SNI is a functionally complete Manchester encoder/decoder including ECL like balanced driver and receivers, on board crystal oscillator, collision signal translator, and a diagnostic loopback circuit.

The SNI is part of a three chip set that implements the complete IEEE compatible network node electronics as shown below. The other two chips are the DP8392 Coax Transceiver Interface (CTI) and the DP8390 Network Interface Controller (NIC).

Incorporated into the CTI are the transceiver, collision and jabber functions. The Media Access Protocol and the buffer management tasks are performed by the NIC. There is an isolation requirement on signal and power lines between the CTI and the SNI. This is usually accomplished by using a set of miniature pulse transformers that come in a 16-pin plastic DIP for signal lines. Power isolation, however, is done by using a DC to DC converter.

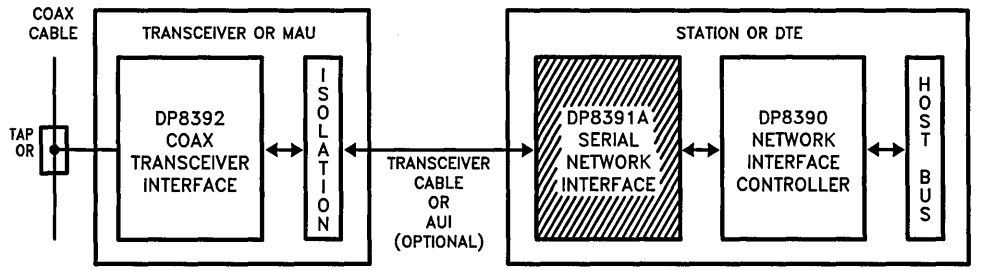
Features

- Compatible with Ethernet II, IEEE 802.3; 10Base5, 10Base2, and 10Base-T
- 10 Mb/s Manchester encoding/decoding with receive clock recovery
- Patented digital phase locked loop (DPLL) decoder requires no precision external components
- Decodes Manchester data with up to ±18 ns of jitter
- Loopback capability for diagnostics
- Externally selectable half or full step modes of operation at transmit output
- Squelch circuits at the receive and collision inputs reject noise
- High voltage protection at transceiver interface (16V)
- TTL/MOS compatible controller interface
- Connects directly to the transceiver (AUI) cable

Table of Contents

1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set



TL/F/9357–1

2.0 Block Diagram

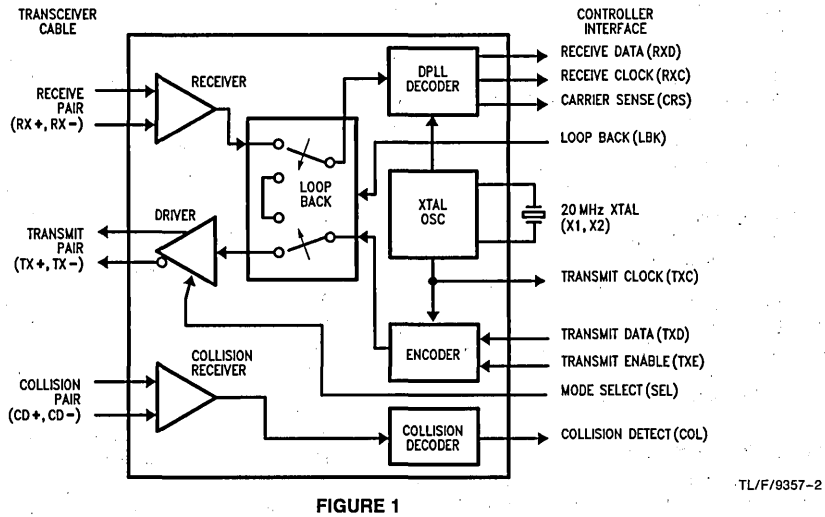


FIGURE 1

3.0 Functional Description

The SNI consists of five main logical blocks:

a) the oscillator—generates the 10 MHz transmit clock signal for system timing.

b) the Manchester encoder and differential output driver—accepts NRZ data from the controller, performs Manchester encoding, and transmits it differentially to the transceiver.

c) the Manchester decoder—receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends them to the controller.

d) the collision translator—indicates to the controller the presence of a valid 10 MHz signal at its input.

e) the loopback circuitry—when asserted, switches encoded data instead of receive input signals to the digital phase-locked loop.

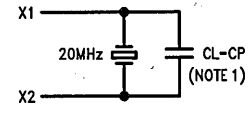
3.1 OSCILLATOR

The oscillator is controlled by a 20 MHz parallel resonant crystal connected between X1 and X2 or by an external clock on X1. The 20 MHz output of the oscillator is divided by 2 to generate the 10 MHz transmit clock for the controller. The oscillator also provides internal clock signals to the encoding and decoding circuits.

Crystal Specification

Resonant frequency	20 MHz
Tolerance	±0.001% at 25°C
Stability	±0.005% 0–70°C
Type	AT-Cut
Circuit	Parallel Resonance

The 20 MHz crystal connection to the SNI requires special care. The IEEE 802.3 standard requires a 0.01% absolute accuracy on the transmitted signal frequency. Stray capacitance can shift the crystal's frequency out of range, causing the transmitted frequency to exceed its 0.01% tolerance. The frequency marked on the crystal is usually measured with a fixed shunt capacitance (C_L) that is specified in the crystal's data sheet. This capacitance for 20 MHz crystals is typically 20 pF. The capacitance between the X1 and X2 pins of the SNI, of the PC board traces and the plated through holes plus any stray capacitance such as the socket capacitance, if one is used, should be estimated or measured. Once the total sum of these capacitances is determined, the value of additional external shunt capacitance required can be calculated. This capacitor can be a fixed 5% tolerance component. The frequency accuracy should be measured during the design phase at the transmit clock pin (TXC) for a given pc layout. *Figure 2* shows the crystal connection.



CL = Load capacitance specified by the crystal's manufacturer

CP = Total parasitic capacitance including:
a) SNI input capacitance between X1 and X2 (typically 5 pF)
b) PC board traces, plated through holes, socket capacitances

Note 1: When using a Viking (San Jose) VXB49N5 crystal, the external capacitor is not required, as the C_L of the crystal matches the input capacitance of the DP8391A.

FIGURE 2. Crystal Connection

3.2 MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The encoder combines clock and data information for the transceiver. Data encoding and transmission begins with the transmit enable input (TXE) going high. As long as TXE re-

3.0 Functional Description (Continued)

mains high, transmit data (TXD) is encoded out to the transmit-driver pair (TX±). The transmit enable and transmit data inputs must meet the setup and hold time requirements with respect to the rising edge of transmit clock. Transmission ends with the transmit enable input going low. The last transition is always positive at the transmit output pair. It will occur at the center of the bit cell if the last bit is one, or at the boundary of the bit cell if the last bit is zero.

The differential line driver provides ECL like signals to the transceiver with typically 5 ns rise and fall times. It can drive up to 50 meters of twisted pair AUI Ethernet transceiver cable. These outputs are source followers which need external 270Ω pulldown resistors to ground. Two different modes, full-step or half-step, can be selected with SEL input. With SEL low, transmit + is positive with respect to transmit − in the idle state. With SEL high, transmit + and transmit − are equal in the idle state, providing zero differential voltage to operate with transformer coupled loads. *Figures 4, 5* and *6* illustrate the transmit timing.

3.3 MANCHESTER DECODER

The decoder consists of a differential input circuitry and a digital phase-locked loop to separate Manchester encoded data stream into clock signals and NRZ data. The differential input should be externally terminated if the standard 78Ω transceiver drop cable is used. Two 39Ω resistors connected in series and one optional common mode bypass capacitor would accomplish this. A squelch circuit at the input rejects signals with pulse widths less than 5 ns (negative going), or with levels less than −175 mV. Signals more negative than −300 mV and with a duration greater than 30 ns are always decoded. This prevents noise at the input from falsely triggering the decoder in the absence of a valid signal. Once the input exceeds the squelch requirements, carrier sense (CRS) is asserted. Receive data (RXD) and receive clock (RXC) become available typically within 6 bit times. At this point the digital phase-locked loop has locked to the incoming signal. The DP8391A decodes a data frame with up to ±18 ns of jitter correctly.

The decoder detects the end of a frame when the normal mid-bit transition on the differential input ceases. Within one and a half bit times after the last bit, carrier sense is de-asserted. Receive clock stays active for five more bit times before it goes low and remains low until the next frame. *Figures 7, 8* and *9* illustrate the receive timing.

3.4 COLLISION TRANSLATOR

The Ethernet transceiver detects collisions on the coax cable and generates a 10 MHz signal on the transceiver cable. The SNI's collision translator asserts the collision detect output (COL) to the DP8390 controller when a 10 MHz signal is present at the collision inputs. The controller uses this signal to back off transmission and recycle itself. The collision detect output is de-asserted within 350 ns after the 10 MHz input signal disappears.

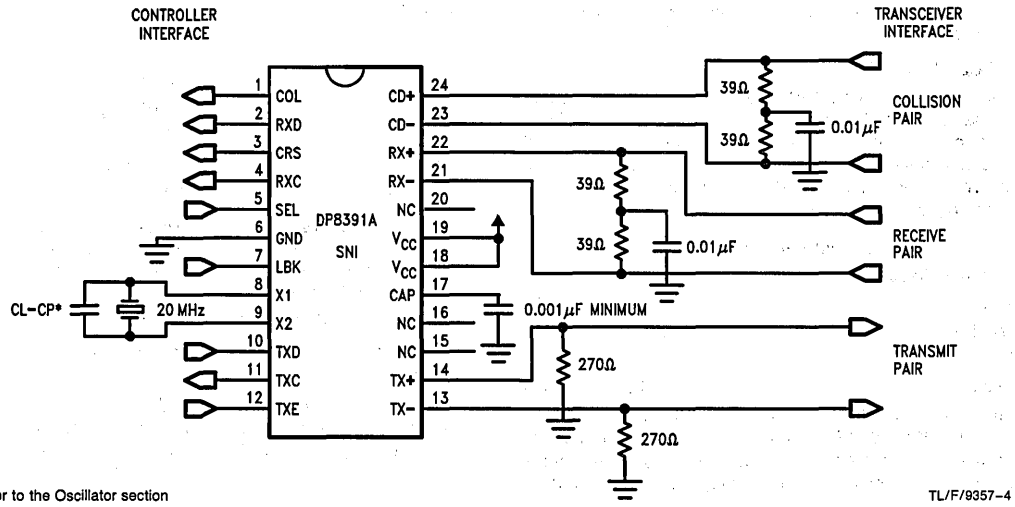
The collision differential inputs (+ and −) should be terminated in exactly the same way as the receive inputs. The collision input also has a squelch circuit that rejects signals with pulse widths less than 5 ns (negative going), or with levels less than −175 mV. *Figure 10* illustrates the collision timing.

3.5 LOOPBACK FUNCTIONS

Logic high at loopback input (LBK) causes the SNI to route serial data from the transmit data input, through its encoder, returning it through the phase-locked-loop decoder to receive data output. In loopback mode, the transmit driver is in idle state and the receive and collision input circuitries are disabled.

4.0 Connection Diagram

Top View



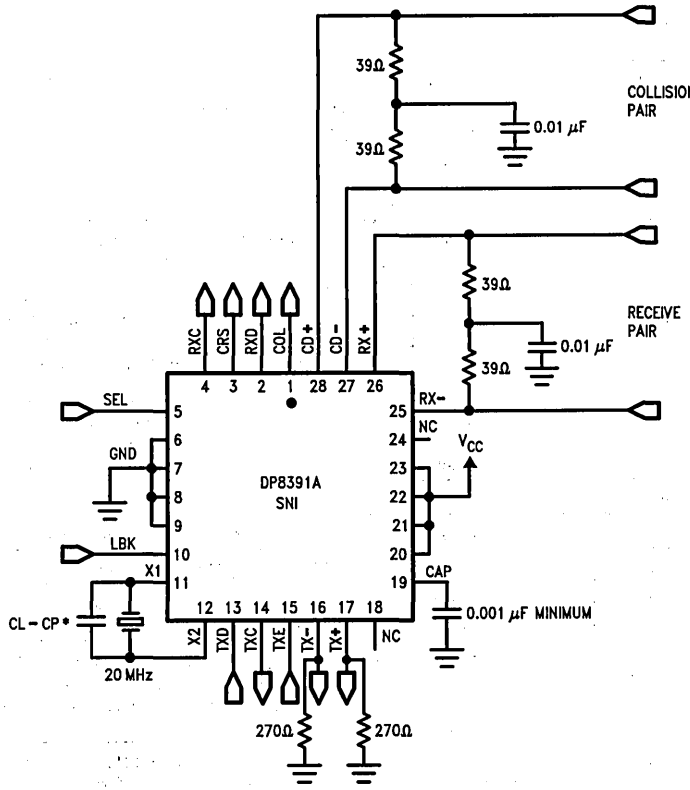
*Refer to the Oscillator section

TL/F/9357–4

FIGURE 3a

Order Number DP8391AN
See NS Package Number N24C

PCC Connection Diagram



TL/F/9357–5

*Refer to the Oscillator section

FIGURE 3b
Order Number DP8391AV
NS Package Number V28A

5.0 Pin Descriptions

Pin No. (DIP)	Pin No. (PCC)	Name	I/O	Description
1	1	COL	O	**Collision Detect Output.** A TTL/MOS level active high output. A 10 MHz (+25%−15%) signal at the collision input will produce a logic high at COL output. When no signal is present at the collision input, COL output will go low.
2	2	RXD	O	**Receive Data Output.** A TTL/MOS level signal. This is the NRZ data output from the digital phase-locked loop. This signal should be sampled by the controller at the rising edge of receive clock.
3	3	CRS	O	**Carrier Sense.** A TTL/MOS level active high signal. It is asserted when valid data from the transceiver is present at the receive input. It is de-asserted one and a half bit times after the last bit at receive input.
4	4	RXC	O	**Receive Clock.** A TTL/MOS level recovered clock. When the phase-locked loop locks to a valid incoming signal a 10 MHz clock signal is activated on this output. This output remains low during idle (5 bit times after activity ceases at receive input).
5	5	SEL	I	**Mode Select.** A TTL level input. When high, transmit + and transmit − outputs are at the same voltage in idle state providing a "zero" differential. When low, transmit + is positive with respect to transmit − in idle state.
6	6–9	GND		**Negative Supply Pins.**
7	10	LBK	I	**Loopback.** A TTL level active high on this input enables the loopback mode.
8	11	X1	I	**Crystal or External Frequency Source Input (TTL).**
9	12	X2	O	**Crystal Feedback Output.** This output is used in the crystal connection only. It must be left open when driving X1 with an external frequency source.
10	13	TXD	I	**Transmit Data.** A TTL level input. This signal is sampled by the SNI at the rising edge of transmit clock when transmit enable input is high. The SNI combines transmit data and transmit clock signals into a Manchester encoded bit stream and sends it differentially to the transceiver.
11	14	TXC	O	**Transmit Clock.** A TTL/MOS level 10 MHz clock signal derived from the 20 MHz oscillator. This clock signal is always active.
12	15	TXE	I	**Transmit Enable.** A TTL level active high data encoder enable input. This signal is also sampled by the SNI at the rising edge of transmit clock.
13 14	16 17	TX− TX+	O	**Transmit Output.** Differential line driver which sends the encoded data to the transceiver. These outputs are source followers and require 270Ω pulldown resistors to GND.
15 16	18	NC		**No Connection.**
17	19	CAP	O	**Bypass Capacitor.** A ceramic capacitor (greater than 0.001 μF) must be connected from this pin to GND.
18 19	20–23	VCC		**Positive Supply Pins.** A 0.1 μF ceramic decoupling capacitor must be connected across VCC and GND as close to the device as possible.
20	24	NC		**No Connection.**
21 22	25 26	RX− RX+	I	**Receive Input.** Differential receive input pair from the transceiver.
23 24	27 28	CD− CD+	I	**Collision Input.** Differential collision input pair from the transceiver.

6.0 Absolute Maximum Ratings

Supply Voltage (V_{CC})	7V
Input Voltage (TTL)	0 to 5.5V
Input Voltage (differential)	−5.5 to +16V
Output Voltage (differential)	0 to 16V
Output Current (differential)	−40 mA
Storage Temperature	−65° to 150°C
Lead Temperature (soldering, 10 sec)	300°C
Package Power Rating for DIP at 25°C (PC Board Mounted) Derate Linearly at the rate of 23.8 mW/°C	2.95W*
Package Power Rating for PCC at 25°C Derate Linearly at the rate of 15.4 mW/°C	1.92W*

*For actual power dissipation of the device please refer to Section 7.0.

ESD rating	2000V

Recommended Operating Conditions

Supply Voltage (V_{CC})	5V ± 5%
Ambient Temperature (DIP)	0° to 70°C
(PCC)	0° to 55°C

Note: *Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

7.0 Electrical Characteristics

V_{CC} = 5V ±5%, T_A = 0°C to 70°C for DIP and 0°C to 55°C for PCC (Notes 1 & 2)

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{IH}	Input High Voltage (TTL)		2.0		V
V_{IHX1a}	Input High Voltage (X1)	No Series Resistor	2.0	V_{CC}−1.5	V
V_{IHX1b}	Input High Voltage (X1)	1k Series Resistor	2.0	V_{CC}	V
V_{IL}	Input Low Voltage (TTL and X1)			0.8	V
I_{IH}	Input High Current (TTL) Input High Current (RX± CD±)	V_{IN} = V_{CC} V_{IN} = V_{CC}		50 500	μA μA
I_{IL}	Input Low Current (TTL) Input Low Current (RX± CD±)	V_{IN} = 0.5V V_{IN} = 0.5V		−300 −700	μA μA
V_{CL}	Input Clamp Voltage (TTL)	I_{IN} = −12 mA		−1.2	V
V_{OH}	Ouptut High Voltage (TTL/MOS)	I_{OH} = −100 μA	3.5		V
V_{OL}	Output Low Voltage (TTL/MOS)	I_{OL} = 8 mA		0.5	V
I_{OS}	Output Short Circuit Current (TTL/MOS)		−40	−200	mA
V_{OD}	Differential Output Voltage (TX±)	78Ω termination, and 270Ω from each to GND	±550	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±)	same as above		±40	mV
V_{DS}	Diff. Squelch Threshold (RX± CD±)		−175	−300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± CD±)		−5.25	5.25	V
I_{CC}	Power Supply Current	10Mbit/s		270	mA

8.0 Switching Characteristics

V_{CC} = 5V ±5%, T_A = 0°C to 70°C for DIP and 0°C to 55°C for PCC (Note 2)

Symbol	Parameter	Figure	Min	Typ	Max	Units
OSCILLATOR SPECIFICATION						
t_{XTH}	X1 to Transmit Clock High	12	8		20	ns
t_{XTL}	X1 to Transmit Clock Low	12	8		20	ns
TRANSMIT SPECIFICATION						
t_{TCd}	Transmit Clock Duty Cycle at 50% (10 MHz)	12	42	50	58	%
t_{TCr}	Transmit Clock Rise Time (20% to 80%)	12			8	ns
t_{TCf}	Transmit Clock Fall Time (80% to 20%)	12			8	ns
t_{TDs}	Transmit Data Setup Time to Transmit Clock Rising Edge	4 & 12	20			ns
t_{TDh}	Transmit Data Hold Time from Transmit Clock Rising Edge	4 & 12	0			ns
t_{TEs}	Transmit Enable Setup Time to Trans. Clock Rising Edge	4 & 12	20			ns
t_{TEh}	Transmit Enable Hold Time from Trans. Clock Rising Edge	5 & 12	0			ns

Note 1: All currents into device pins are positive, all currents out of device pins are negative. All voltages are referenced to ground unless otherwise specified.

Note 2: All typicals are given for V_{CC} = 5V and T_A = 25°C.

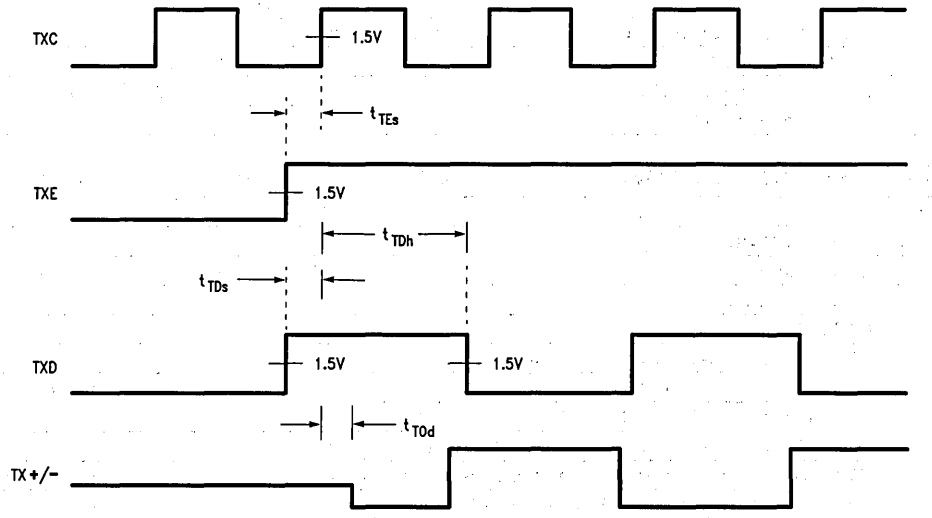
8.0 Switching Characteristics

V_{CC} = 5V ±5%, T_A = 0°C to 70°C for DIP and 0°C to 55°C for PCC (Note 2) (Continued)

Symbol	Parameter	Figure	Min	Typ	Max	Units
TRANSMIT SPECIFICATION (Continued)						
t_{TOd}	Transmit Output Delay from Transmit Clock Rising Edge	4 & 12			50	ns
t_{TOr}	Transmit Output Rise Time (20% to 80%)	12			7	ns
t_{TOf}	Transmit Output Fall Time (80% to 20%)	12			7	ns
t_{TOj}	Transmit Output Jitter	12		±0.25		ns
t_{TOh}	Transmit Output High Before Idle in Half Step Mode	5 & 12	200			ns
t_{TOi}	Transmit Output Idle Time in Half Step Mode	5 & 12			800	ns
RECEIVE SPECIFICATION						
t_{RCd}	Receive Clock Duty Cycle at 50% (10 MHz)	12	40	50	60	%
t_{RCr}	Receive Clock Rise Time (20% to 80%)	12			8	ns
t_{RCf}	Receive Clock Fall Time (80% to 20%)	12			8	ns
t_{RDr}	Receive Data Rise Time (20% to 80%)	12			8	ns
t_{RDf}	Receive Data Fall Time (80% to 20%)	12			8	ns
t_{RDs}	Receive Data Stable from Receive Clock Rising Edge	7 & 12	±40			ns
t_{CSon}	Carrier Sense Turn On Delay	7 & 12			50	ns
t_{CSoff}	Carrier Sense Turn Off Delay	8, 9 & 12			160	ns
t_{DAT}	Decoder Acquisition Time	7		0.6	1.80	μs
t_{Drej}	Differential Inputs Rejection Pulse Width (Squelch)	7	5		30	ns
t_{Rd}	Receive Throughput Delay	8 & 12			150	ns
COLLISION SPECIFICATION						
t_{COLon}	Collision Turn On Delay	10 & 12			50	ns
t_{COLoff}	Collision Turn Off Delay	10 & 12			350	ns
LOOPBACK SPECIFICATION						
t_{LBs}	Loopback Setup Time	11	20			ns
t_{LBh}	Loopback Hold Time	11	0			ns

Note 2: All typicals are given for V_{CC} = 5V and T_A = 25°C.

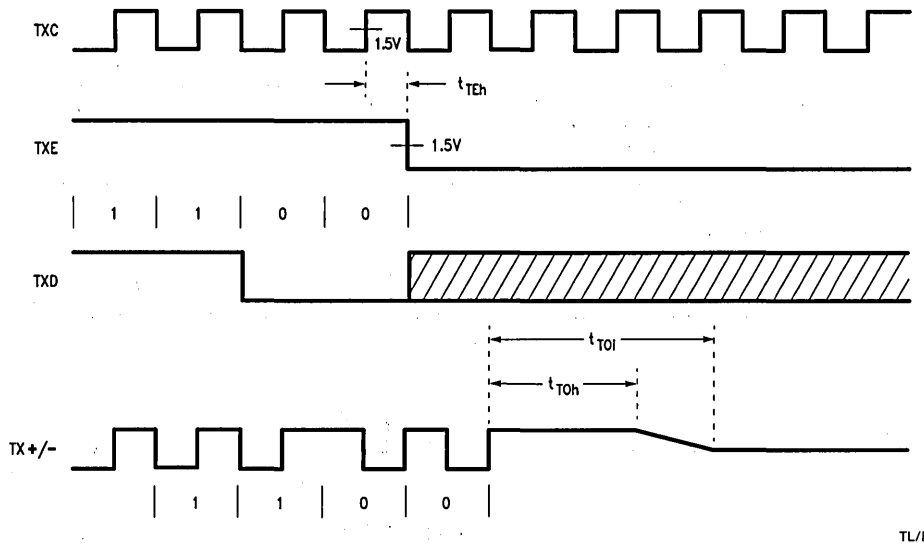
9.0 Timing and Load Diagrams



TL/F/9357–6

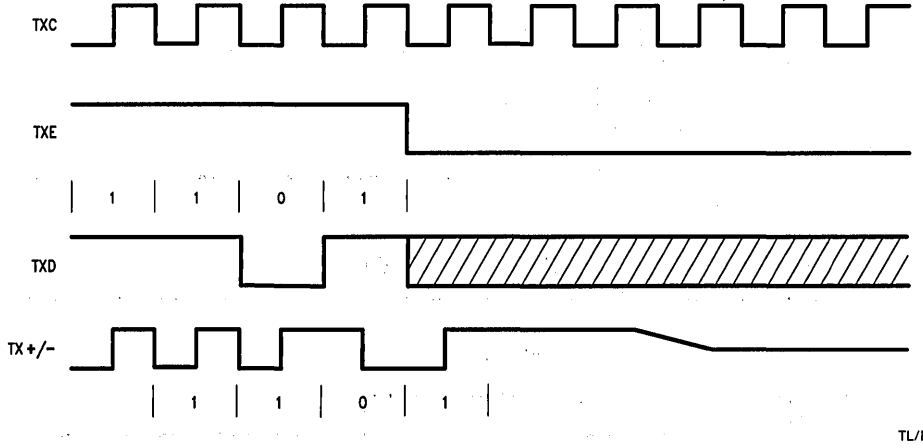
FIGURE 4. Transmit Timing - Start of Transmission

9.0 Timing and Load Diagrams (Continued)



TL/F/9357–7

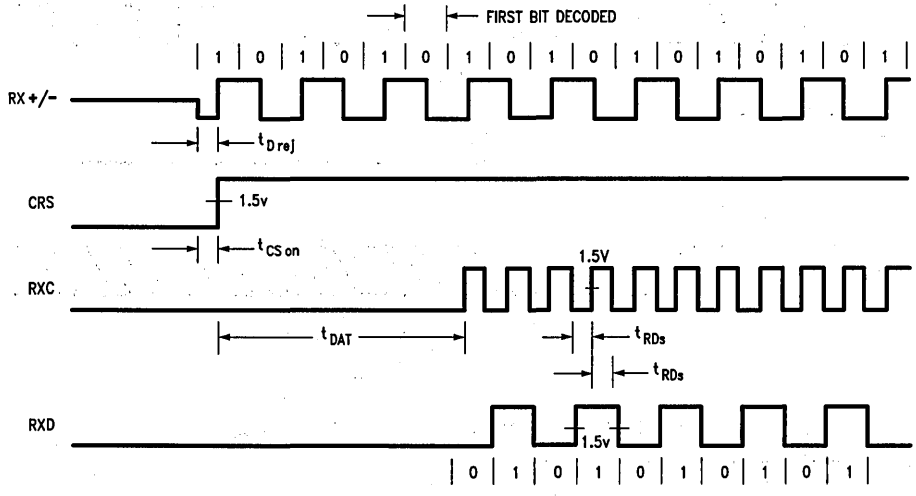
FIGURE 5. Transmit Timing - End of Transmission (last bit = 0)



TL/F/9357–8

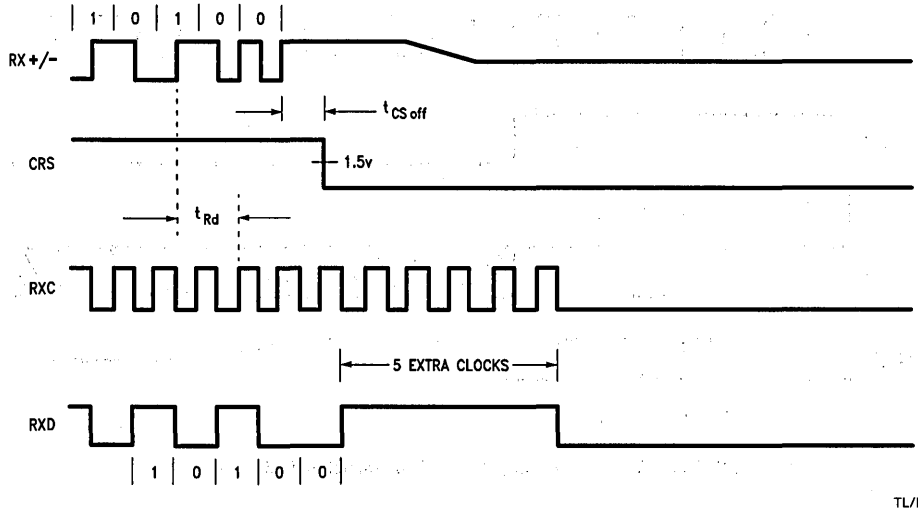
FIGURE 6. Transmit Timing - End of Transmission (last bit = 1)

9.0 Timing and Load Diagrams (Continued)



TL/F/9357–9

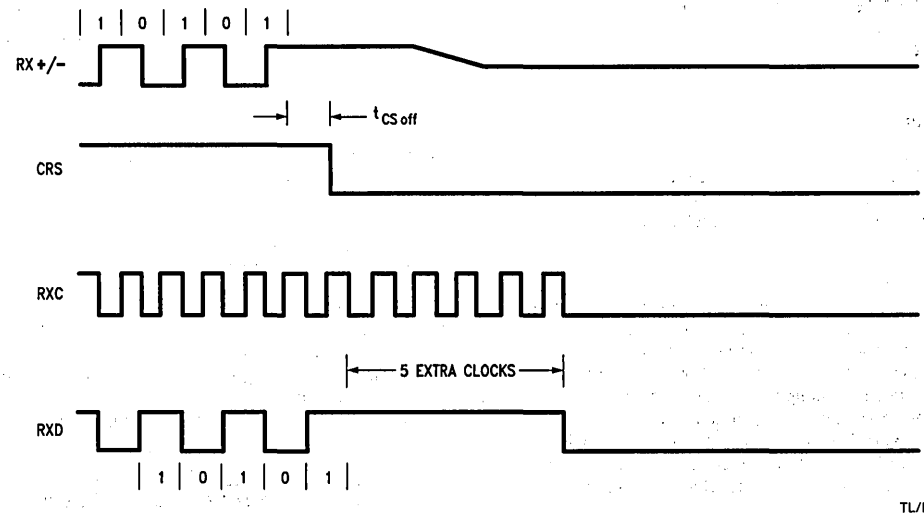
FIGURE 7. Receive Timing - Start of Packet



TL/F/9357–10

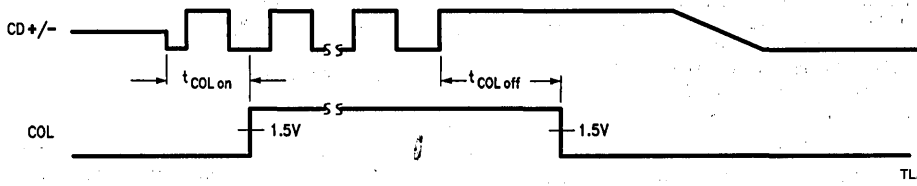
FIGURE 8. Receive Timing - End of Packet (last bit = 0)

9.0 Timing and Load Diagrams (Continued)



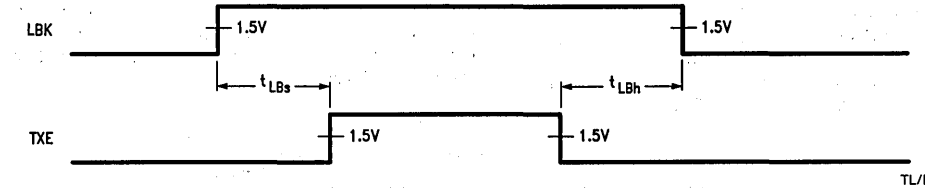
TL/F/9357-11

FIGURE 9. Receive Timing - End of Packet (last bit = 1)



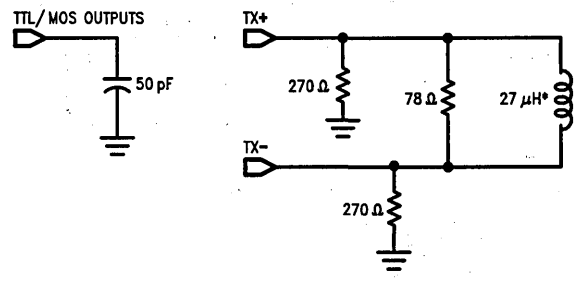
TL/F/9357-12

FIGURE 10. Collision Timing



TL/F/9357-13

FIGURE 11. Loopback Timing



TL/F/9357-14

*27 μH transformer is used for testing purposes, 100 μH transformers (Valor, LT1101, or Pulse Engineering 64103) are recommended for application use.

FIGURE 12. Test Loads

DP8390 Network Interface Controller: An Introductory Guide

National Semiconductor
Application Note 475

OVERVIEW

A general description of the DP8390 Network Interface Controller (NIC) is given in this application note. The emphasis is placed on how it operates, and how it can be used. This description should be read in conjunction with the DP8390 data sheet.

1.0 INTRODUCTION

The DP8390 Network Interface Controller provides all the Media Access Control layer functions required for transmission and reception of packets in accordance with the IEEE 802.3 CSMA/CD standard. The controller was designed to act as an advanced peripheral and serve as a complete interface between the system and the network. The onboard FIFO and DMA channels work together to form a straight-forward packet management scheme, providing (local) DMA transfers at up to 10 megabytes per second while tolerating typical bus latencies.

A second set of DMA channels (remote DMA) is provided on chip, and is integrated into the packet management scheme to aid in the system interface. The DP8390 was designed with the popular 8, 16 and 32 bit microprocessors in mind, and gives system designers several architectural options. The NIC is fabricated using National Semiconductor's double metal 2 micron microCMOS process, yielding high speed with very low power dissipation.

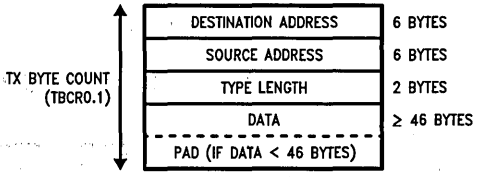
2.0 METHOD OF OPERATION

The NIC is used as a standard peripheral device and is controlled through an array of on-chip registers. These registers are used during initialization, packet transmission and reception, and remote DMA operations. At initialization, the physical address and multicast filters are set, the receiver, transmitter and data paths are configured, the DMA channels are prepared, and the appropriate interrupts are masked. The Command Register (CR) is used to initiate transmission and remote DMA operations.

Upon packet reception, end of packet transmission, remote DMA completion or error conditions, an interrupt is generated to indicate that an action should be taken. The processor's interrupt driven routine then reads the Interrupt Status Register (ISR) to determine the type of interrupt that occurred, and performs the appropriate actions.

3.0 PACKET TRANSMISSION

The NIC transmits packets in accordance with the CSMA/CD protocol, scheduling retransmission of packets up to 15 times on collisions according to the truncated binary exponential backoff algorithm. No additional processor intervention is required once the transmit command is given.

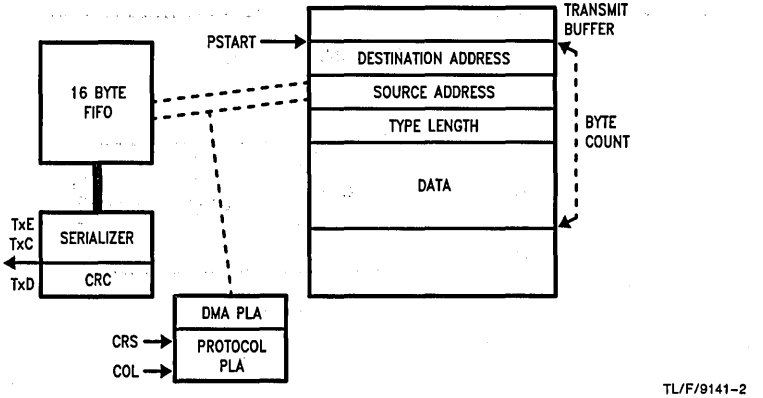


TL/F/9141–1

FIGURE 1. Transmit Packet Format

3.1 Transmission Setup

After a packet that conforms to the IEEE 820.3 specification is set up in memory, with 6 bytes of the destination address, followed by 6 bytes of the source address, followed by the data byte count and the data, it is ready for transmission (see *Figure 1*). To transmit a packet, the NIC is given the starting address of the packet (TPSR), the length of the packet (TPCR0, TBCR1), and then the PTX (transmit packet) bit of the Command Register is set to initiate the transmission (see *Figure 2*).



TL/F/9141–2

FIGURE 2. Packet Transmission

3.2 Transmission Process

Once the transmit command is given, if no reception is in progress, the transmit prefetch begins. The high speed local DMA channel bursts data into the NIC's FIFO. After the first DMA transfer of the prefetch burst, if no carrier is present on the network, and the NIC is not deferring, the TXE (transmit enable) signal is asserted and the transmission begins. After the 62 bits of preamble (alternating ONEs and ZEROs) and the start of frame delimiter (two ONEs) are sent out, the data in the FIFO is serialized, and sent out as NRZ data (pin TxD) with a clock (TxC), while the CRC is calculated. When the FIFO reaches a threshold (X bytes empty) a new DMA burst is initiated. This process continues until the byte count reaches zero. After the last byte is serialized, the four bytes of the calculated CRC are serialized and appended to complete the packet.

Should a collision occur, the current transmission stops, a jam sequence (32 Ones) transmitted (to ensure that every node senses a collision), and a retransmission of the packet is scheduled according to the truncated Binary Exponential Backoff Routine.

3.3 Transmission Status

After the transmission is complete, an interrupt is generated and either the PTX bit (complete packet transmitted) or the TXE bit (packet transmission aborted) of the ISR (Interrupt Status Register) is set. The interrupt driven routine then reads the TSR to find out details of the transmission. If the PTX bit is set, the TSR can reveal if a carrier was present when the transmission was initiated (DFR), if the carrier was lost during the transmission (CRS—this would point to a short somewhere on the network), if the collision detect circuitry is working properly (CDH), and if collision occurred (COL). Whenever a collision is encountered during transmission, the collision count register (NCR) is incremented. Should a collision occur outside the 512 bit window (slot time), the OWC (Out of Window Collision) bit of the TSR is set.

The TXE bit of the ISR is set if 16 collisions or a FIFO underrun occurs. If the transmission is aborted due to 16 collisions, the ABT bit of the TSR is set. (If this occurs it is likely that there is an open somewhere on the network.) If the local DMA channel can not fill the FIFO faster than data is sent to the network, the FU bit (FIFO Underrun) of the TSR is set and the transmission is also aborted. This is a result of a system bandwidth problem and points to a system design flaw. System bandwidth considerations are discussed further in Section 5.1.3.

4.0 PACKET RECEPTION

The bus topology used in CSMA/CD networks allows every node to receive every packet transmitted on the network. The receive filters determine which packets will be buffered to memory. Since every packet is not of interest, only packets having a destination address that passes the node's receive filters will be transferred into memory. The NIC offers many options for the receive filters and implements a complete packet management scheme for storage of incoming packets.

4.1 Reception Process

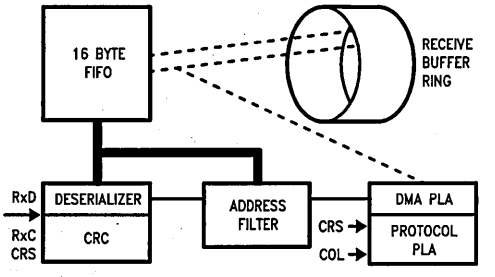
When a carrier is first sensed on the network (i.e. CRS signal is active), the controller sees the alternating ONE - ZERO preamble and begins checking for two consecutive ONEs, denoting the start of frame delimiter (SFD). Once the SFD is detected, the serial stream of data is deserialized and pushed into the FIFO, a byte at a time. As the data is being transferred into the FIFO, the first six bytes are checked against the receive address filters. If an address match occurs, the packet is DMAed from the FIFO into the receive buffer ring. If the address does not match, the packet is not buffered and the FIFO is reset.

Each time the FIFO threshold is reached, a DMA burst begins and continues for the proper number of transfers. DMA bursts continue until the end of the packet (Section 5.1.2). At the end of a reception, the NIC prepares for an immediate reception while writing the status of the previous reception to memory. An interrupt is issued to indicate that a packet was received, and is ready to be processed.

The CRC generator is free running and is reset whenever the SFD is detected. At every byte boundary the calculated value of the CRC is compared with the last four received bytes. When the CRS signal goes LOW, denoting the end of a packet, if the calculated CRC matches the received CRC on the last byte boundary, the packet is a good packet and is accepted. However, if the calculated and received CRCs do not match on the last byte boundary before CRS goes LOW, a CRC error is flagged (CRC bit of RSR set) and the packet is rejected, i.e. the receive buffer ring pointer (CURR) is not updated (Section 4.5). If the CRS signal does not go LOW on a byte boundary and a CRC error occurs, the incoming packet is misaligned, and a frame alignment error is flagged (FAE bit of RSR set). Frame alignment errors only occur with CRC errors.

4.2 Address Matches

The first bit received after the SFD indicates whether the incoming packet has a physical or multicast address. A ZERO indicates a physical address, that is, a unique map-



TL/F/9141–3

FIGURE 3. Packet Reception

ping between the received address and the node's 48 bit physical address as programmed at intialization (PAR0–PAR5). A ONE indicates a multicast address, meaning a packet intended for more than one node.

Multicast addressing is useful where one node needs to send a packet to multiple nodes, as in a query command. Multicast addressing provides a very fast way to perform address filtering in realtime, by using an on-chip hashing table. A hashing algorithm based on the CRC is used to map the multicast address into the 64 bit Multicast Address Filter (MAF0–7).

After the CRC has been calculated on the destination address, the upper six bits of the CRC are used as an index into the Multicast Address Filter (MAF). If the selected filter bit is ONE, the packet is accepted, if the MAF bit is ZERO the packet is not accepted.

A special multicast address is the broadcast address, which denotes a packet intended to be received by all nodes. The broadcast packet has an address of all ONEs (this address also maps into a bit in the MAF).

The DP8390 also provides the ability to accept all packets on the network with a physical address. Promiscuous mode causes any packet with a physical address to be buffered into memory. To receive all multicast packets it is nesessary to set all of the MAF bits to ONE.

4.3 Network Statistics

Three eight bit counters are provided for monitoring receive packet errors. After an address match occurs if a Frame Alignment or CRC error occurs, or if a packet is lost due to insufficient buffer resources (see below), the appropriate counter is incremented. These counters are cleared when read. The counters trigger an interrupt when they reach a value of 128 (if not masked) to force the processor to read (and thus clear) their contents. The counters have a maximum value of 192, providing a large latency between when the interrupt is asserted and when the counter overflows. When a CNT interrupt occurs, all three tally counters should be read and added into larger counters maintained by the processor.

4.4 Setting the Receive Configuration Register

The Receive Configuration Register (RCR) is used in conjunction with the physical and multicast addresses to determine which packets should be accepted and placed in the receive buffer ring. The RCR is initialized to accept physical, multicast and/or broadcast packets, or alternatively to place the receiver in promiscuous mode to accept all packets with a physical address. If the MON bit of the RCR is set, placing the receiver in monitor mode, the receiver still checks the addresses of incoming packets according to the set up address filter, and network statistics are still gathered, but packets are not buffered into memory.

The minimum packet size in standard 802.3 networks is 64 bytes long. Packets less than 64 bytes are considered runt packets and are normally rejected. However, in some applications it may be desirable to accept such packets. By setting the AR bit of the RCR, runt packets are accepted.

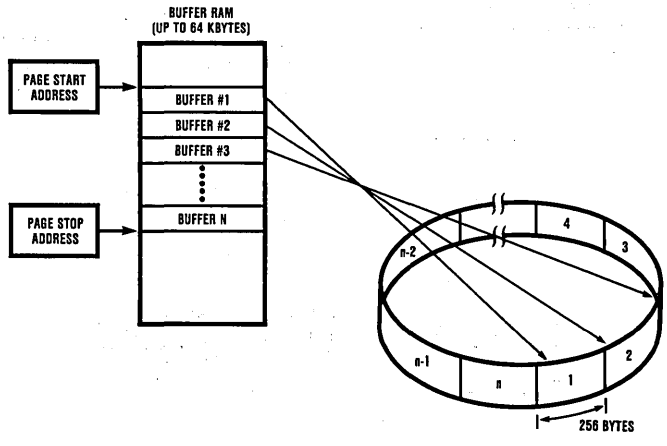
For diagnostic purposes it may be desirable to examine errored packets, and not overwrite them with good packets as is done in normal operation. By setting the SEP bit of the RCR, errored packets are saved and their status is written to memory.

4.5 Receive Buffer Ring

As packets are received they are placed into the receive buffer ring, and as they are processed they are removed from this ring. At initialization, an area of memory is allocated to act as the receive buffer ring, and the NIC's buffer management scheme then makes efficient use of this memory. The efficiency is helped significantly because the ring pointers are contained on chip, and the DMA channels can work at up to a 10 Mbyte/sec transfer rate. A second DMA channel, the remote DMA channel, is available for transferring packets out of the receive buffer ring.

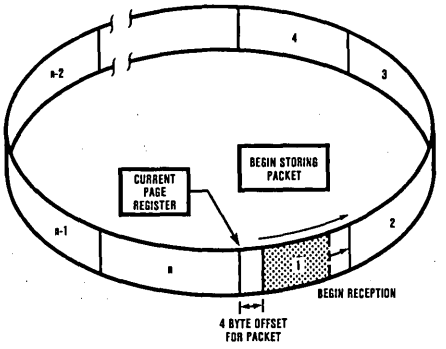
The employed buffer management scheme effectively works as a large packet FIFO. This buffer management scheme is very appropriate for most networking applications because packets are generally processed in the order they are received.

Four pointers are used to control the ring; the (1) page start (PSTART) and (2) page stop (PSTOP) pointers to determine the size of the buffer ring, the (3) current page (CURR) pointer, to determine where the next packet will be loaded,

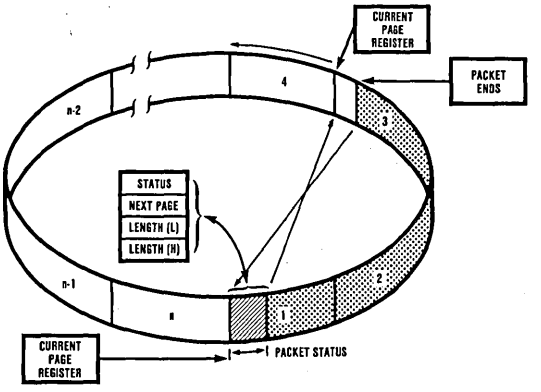


TL/F/9141–4

FIGURE 4. The Receive Buffer



TL/F/9141-5



TL/F/9141-6

FIGURE 5. Receive Packet Buffering

and the (4) boundary (BNRY) pointer, to show where the next packet to be unloaded (or processed) lies. As packets are received, the boundary pointer follows the current page pointer around the ring. The page start and stop pointers remain unchanged during operation.

The receive buffer ring is divided into 256 byte buffers, and these buffers are linked together as required by the received packets (see *Figure 4*). Up to 256 of these buffers can be linked together in the receive buffer ring, yielding a maximum buffer size of 64K bytes. Since all NIC registers are 8 bits wide, the ring pointers refer to 256 byte boundaries within a 64K byte space.

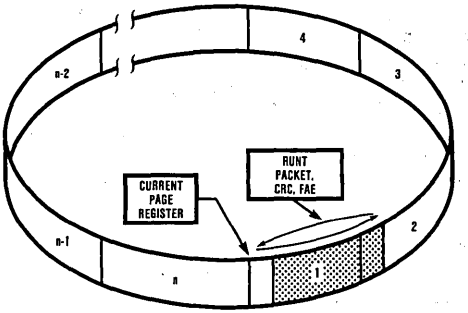
At initialization, PSTART register is loaded with the beginning page address of the ring, and PSTOP is loaded with the ending page address of the ring.

On a valid reception, the packet is placed in the ring at the page pointed to by CURR plus a 4 byte offset (see *Figure 5*). The packet is transferred to the ring, a DMA burst at a time. When necessary, buffers are automatically linked together, until the complete packet is received. The last and first buffers of the ring buffer are linked just as the first and seconds buffers. At the end of a reception, the status from the Receive Status Register (RSR), a pointer to the next packet, and the byte count of the current packet are written into the 4 byte offset.

If a receive error occurs (FAE, CRC) CURR is not updated at the end of a reception, so the next packet received overwrites the bad packet (see *Figure 6*). This feature can be disabled (by setting the save errored packet (SEP) bit in the RCR) to allow examination of errored packets.

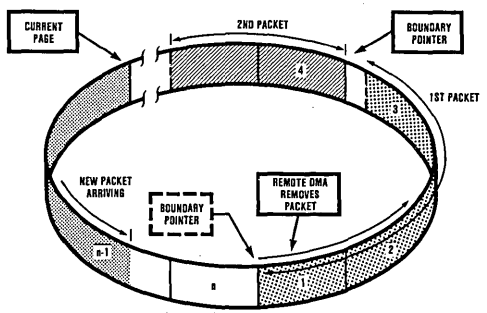
At receiving nodes, collision fragments may be seen as runt packets. A runt packet is a packet less than 64 bytes (512 bits) long, and since a collision must occur in the first 512 bit times, the packet will be truncated to less than 64 bytes. After runt packets are received, the CURR is not updated, so the next packet received will overwrite the runt packet. This standard feature can also be suppressed by setting the AR bit in the TCR. This is useful when it is desirable to examine collision fragments, and in non-standard applications where smaller packets are desirable.

Once packets are in the receive ring they must be processed. However, the amount of processing that occurs while the packet is in the buffer ring varies according to the implementation. As packets are removed from the buffer ring, the boundary pointer (BNRY) must be updated. The BNRY always follows CURR around the ring (see *Figure 7*).



TL/F/9141-7

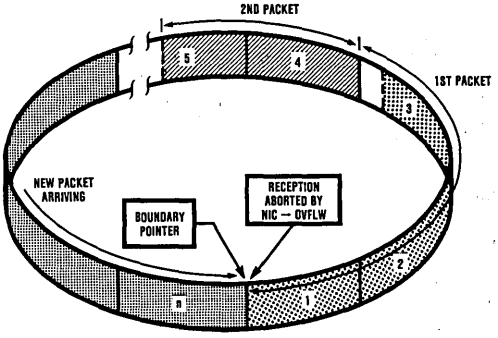
FIGURE 6. Packet Rejection



TL/F/9141-8

FIGURE 7. Removing Packets From Receive Buffer Ring

If the current local DMA address ever reaches BNRY, the ring is full. In this case, the current and any additional receptions are aborted and tallied until the BNRY pointer is updated. Packets already present in the ring will not be overwritten (see *Figure 8*). All missed packets will increment the missed packet tally counter. When enough memory is allocated for the receive buffer ring, the overwrite warning (setting of the OVW bit of the ISR) should seldom occur.



TL/F/9141–9

FIGURE 8. Receive Buffer Ring Overwrite Protection

A second set of DMA channels has been included on the DP8390 to aid in the transfer of packets out of the buffer ring. These Remote DMA channels can work in close co-operation with the receive buffer ring to provide a very effective system interface (§7).

If the BNRY is placed outside of the buffer ring, no overwrite protection will be present, and incoming packets may overwrite packets that have not been processed. This may be useful when evaluating the DP8390, but in normal operation it is not recommended.

When the CURR and BNRY pointers are equal, the buffer ring can either be completely empty or completely full. To ensure that the NIC does not misinterpret this condition, it is necessary to guarantee that the value of the BNRY pointer does not equal the value of the CURR pointer. It is recommended that the BNRY pointer be kept one less than CURR pointer when the ring is empty, and only be equal to CURR when the ring is full, as shown below.

1. Use a variable (NXTPKT) to indicate from where the next packet will be removed (possibly using Remote DMA)
2. At initialization set:
   BNRY = PSTART
   CURR = PSTART + 1
   NXTPKT = PSTART + 1
3. After each packet is removed from the ring, use the next packet pointer in the header information (the second byte of the header), HNXTPKT, and set:
   NXTPKT = HNXTPKT
   BNRY = HNXTPKT − 1
   If BNRY < PSTART then BNRY = PSTOP − 1

The above procedure is not necessary if the Send Packet Command is used to remove packets from the ring as explained in section 7.

5.0 SYSTEM/NETWORK INTERFACE

The DP8390 offers considerable flexibility when designing a system/network interface. This flexibility allows the designer to choose the appropriate price/performance combination while easing the actual design process.

5.1 Interfacing Considerations

Several features have been included on the NIC to allow it to easily be integrated into many systems. The size of the data paths, the byte ordering, and the bus latencies are all programmable. In addition, the clock the DMA channels use is not coupled to the network clock, so the NIC's DMA can easily be integrated into memory systems.

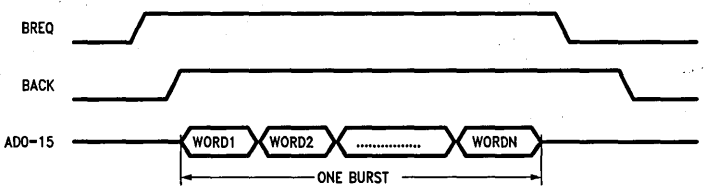
5.1.1 Data Path

The NIC can interface with 8, 16, and 32 bit microprocessors. The data paths are configurable for both byte- wide and word-wide transfers (bit WTS in DCR). When in word-wide mode, the byte ordering is programmable to accommodate both popular byte ordering schemes. All NIC registers are 8 bits wide to allow 8, 16 and 32 bit processors to access them with no additional hardware. If the NIC's 16 address lines (64K bytes) do not provide an adequate address space, the two DMA channels can be concatenated to form a 32 bit DMA address (bit LAS in DCR).

5.1.2 Local DMA

The DMA transfers between the FIFO and memory during transmission and reception occur in bursts. The bursts begin when the FIFO threshold is reached. Since only a single FIFO is required (because a node cannot receive and transmit simultaneously), the threshold takes on different meanings during transmission and reception. During reception the FIFO threshold refers to the number of bytes in the FIFO. During transmission the FIFO threshold refers to the number of empty bytes in the FIFO (16 - # bytes in FIFO). The FIFO threshold is set to 2, 4, 8 or 12 bytes (1, 2, 4 or 6 words) in the DCR (bits FT0, FT1).

The number of transfers that occur in a burst depends on whether the Exact Transfer or Empty/Fill mode is used (bit BMS in DCR). When in Exact Transfer mode, a number of bytes/words equal to the FIFO threshold will be transferred in each burst. The Empty/Fill mode continues the transfers until the FIFO is empty, during receptions, and full, during transmissions (see *Figure 9*).



TL/F/9141–10

where N = 1, 2, 4 or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode

FIGURE 9. Local DMA Burst

Before a burst can begin, the NIC must first arbitrate to become master of the bus. It requests the bus by activating the BREQ signal and waiting for acknowledgment with the BACK signal. Once the NIC becomes the master of the bus, the byte/word transfers may begin. The frequency of the DMA clock is not related to the network clock, and can be input (pin 25) as any frequency up to 20 MHz. For 10 Mbit/sec networks the DMA clock can be as slow as 6 MHz. This allows tailoring of the DMA channel, to the system. The local DMA channel can burst data into and out of the FIFO at up to 10 Mbyte/sec (8X the speed of standard Ethernet). This means that during transmission or reception the network interface could require as little as one eighth of the bus bandwidth.

5.1.3 Bus Analysis

Two parameters useful in analysis of bus systems are the Bus Latency and the Bus Utilization. The Bus Latency is the maximum time between the NIC assertion of BREQ and the system granting of BACK. This is of importance because of the finite size of the NIC's internal FIFO. If the bus latency becomes too great, the FIFO overflows during reception (FIFO overrun error), and becomes empty during transmission (FIFO underrun error). Both conditions result in an error that aborts the reception or transmission. In a well designed system these errors should never occur. The Bus Utilization is the fraction of time the NIC is the master of the bus. It is desirable to minimize the time the NIC occupies the bus, in order to maximize its use by the rest of the system. When designing a system it is necessary to guarantee the NIC a certain Bus Latency, and it is desirable to minimize the Bus Utilization required by the NIC.

Associated with each DMA burst is a DMA set up and recovery time. When a packet is being transferred either to or from memory it will be transferred in a series of bursts. If more byte/word transfers are accomplished in each burst, fewer bursts are required to transfer the complete packet, and less time is spent on DMA set up and recovery. Thus, when longer bursts are used, less bus bandwidth is required to complete the same packet transfer.

The Empty(/Fill) mode guarantees longer bursts because as the byte/word transfers are taking place, serialized data is still filling(/emptying) the FIFO, and these additional bytes/words must also be transferred out of(/into) the FIFO. The least NIC bus utilization occurs when the bursts are as long as possible. This occurs when the threshold is as high as possible, and Empty/Fill mode is used. The determination of the threshold is related to the maximum bus latency the system can guarantee the NIC.

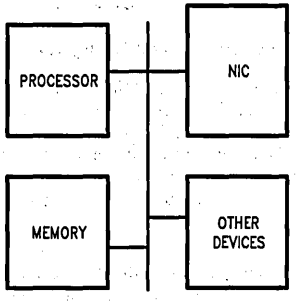
If the NIC is required to guarantee other devices a certain bus latency, it can only remain master of the bus for a certain amount of time. In this case, the Exact Transfer burst mode is desirable because the NIC only remains master of the bus for a certain amount of time.

6.0 INTERFACE OPTIONS

The network interface can be incorporated into systems in several ways. The network interface can be controlled by either a system processor or a dedicated processor, and can utilize either system memory or buffer memory. This section covers the basic interface architectures.

6.1 Single Bus System

The least complex implementation places the NIC on the same bus as the processor (see *Figure 10*). The DP8390 acts as both a master and a slave on this bus; a master during DMA bursts, and a slave during NIC register accesses. This architecture is commonly seen on motherboards in personal computers and low cost workstations, but until recently without an integrated network interface. A major issue in such designs is the bus bandwidth for use by the processor. The DP8390 is particularly suitable for such applications because of its bus utilization characteristics. During transmissions and receptions, the only time the NIC becomes a bus master, the DP8390 can require as little as one- eighth the bus bandwidth. In addition, the previously mentioned bus tailoring features, ease its integration into such systems.



TL/F/9141–11

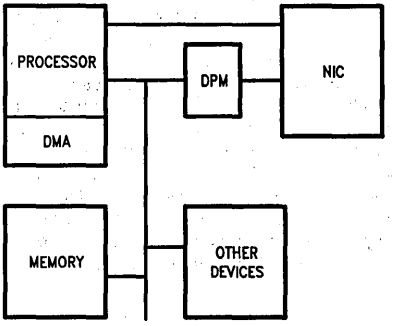
FIGURE 10. Single Bus Configuration

The design must only be able to guarantee the NIC a maximum bus latency (<1.6 μs for 10 Mbit/s networks), because of the finite size of the on-chip FIFO. In bus systems where the NIC is the highest priority device, this should present no problem. However, if the bus contains other devices such as Disk, DMA and Graphic controllers that require the bus for more than 10 μs during high priority or real time activities, meeting this maximum bus latency criteria could present a problem.

Likewise, many existing single bus systems make no provision for external devices to become bus masters, and if they do, it is only under several restrictions. In such cases, an interface without the mentioned bus latency restrictions is highly desirable.

6.2 Dual Port Memory

One popular method of increasing the apparent bus latency of an interface, has the added effect of shielding the system bus from the high priority network bandwidth. In this application, the Dual Port Memory (DPM) allows the system bus to access the memory through one port, while the network interface accesses it through the other port. In this way, all of the high priority network bandwidth is localized on a dedicated bus, with little effect on the system bus (see *Figure 11*).



TL/F/9141–12

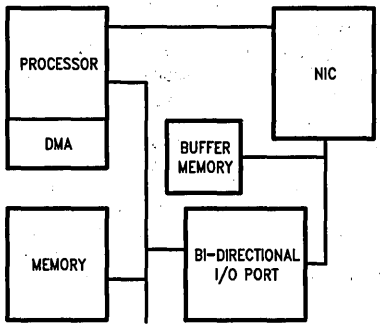
FIGURE 11. DPM Configuration

Dual Port Memories are typically smaller than the main memory and little, if any, processing can occur while the packets are in the DPM. Therefore, the processor (or if available, DMA controller) must transfer data between the DPM and the main memory before beginning packet processing. In this example, the DPM acts as a large packet FIFO.

Such configurations provide popular solutions. Aside from the extra complexity of the software and the DPM contention logic, higher performance can be achieved.

6.3 Dual Port Memory Equivalent

The functional equivalent of a Dual Port Memory implementation can be realized for low cost with the DP8390. This configuration makes use of the NIC's Remote DMA capabilities and requires only a buffer memory, and a bidirectional I/O port (see *Figure 12*). The complete network interface, with 8k x 8 of buffer memory, easily fits onto a half size IBM-PC card (as in the Network Interface Adapter, NIA, for the IBM-PC.)



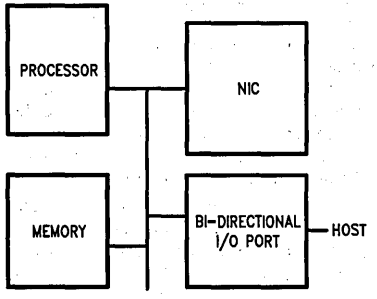
TL/F/9141–13

FIGURE 12. DPM Equivalent Configuration

The high priority network bandwidth is decoupled from the system bus, and the system interracts with the buffer memory using a lower priority bi-directional I/O port. For example, when a packet is received the local DMA channel transfers it into the buffer memory, part of which has been configured as the receive buffer ring. The remote DMA channel then transfers the packet on a byte by byte (or word by word) basis to the I/O port. At this point, as in the previous example, the processor (or if available, DMA channel), through a completely asynchronous protocol, transfers the packet into the main memory.

6.4 Dual Processor Configuration

For higher performance applications, it is desirable to offload the lower-level packet processing functions from the main system (see *Figure 11*). A processor placed on a local bus with the NIC, memory and a bi-directional I/O port could accomplish these lower-level tasks, and communicate with the system processor through a higher level protocol. This processor could be responsible for sending acknowledgement packets, establishing and breaking logical links, assembling and disassembling files, executing remote procedure calls, etc.



TL/F/9141–14

FIGURE 13. Dual Processor Configuration

7.0 REMOTE DMA

A second set of DMA channels is built into the DP8390 to aid in the system integration (as discussed above). Using a simple asynchronous protocol, the Remote DMA channels are used to transfer data between dedicated network memory, and common system memory. In normal operation, the remote DMA channels transfer data between the network memory and an I/O port, and the system transfers between the I/O port and the system memory. The system transfers are typically accomplished using either the processor, or a DMA controller.

The Remote DMA channels work in both directions; pending transmission packets are transferred into the network memory, and received packets are transferred out of the network memory. Transfers into the network memory are known as remote write operations, and transfers out of the network memory are known as remote read operations. A special remote read operation, send packet, automatically removes the next packet from the receive buffer ring.

7.1 Performing Remote DMA Operations

Before beginning a remote DMA operation, the controller must be informed of the network memory it will be using. Both the starting address (RSAR0,1) and length (RBCR0,1) are set before initiating the remote DMA operation. The remote DMA operation begins by setting the appropriate bits in the Command Register (RD0–RD3). When the remote DMA operation is complete (all of the bytes transferred), the RDC bit (Remote DMA Complete) in the ISR (Interrupt Status Register) is set and the processor receives an interrupt, whereupon it takes the appropriate action. When the Send packet command is used, the controller automatically loads the starting address, and byte count from the receive buffer ring for the remote read operation, and upon completion updates the boundary pointer (BNRY) for the receive buffer ring. Only one remote DMA operation can be active at a time.

7.2 Hardware Considerations

The Remote DMA capabilities of the NIC were designed to require minimal external components and provide a simple implementation. An eight bit bi-directional port can be implemented using just two 374 latches (see the DP8390 Hardwre Design Guide). All of the control circuitry is provided on the DP8390. In addition, bus arbitration with the local DMA is accomplished within the NIC in such a way as to not lock out other devices on the bus (see the DP8390 Datasheet).

PC-AT® Compatible DP83902 ST-NIC Ethernet Evaluation Board (DP83902EB-AT)

National Semiconductor
Application Note 752

OVERVIEW

The National Semiconductor ST-NIC Evaluation Board design provides IBM® AT's and AT Compatibles with Thick Ethernet, Thin Ethernet, and Twisted Pair connections. This low parts count Evaluation Board is compatible with the AT bus and requires only a ½ size slot for insertion. The board uses the DP83902 (ST-NIC) to interface to twisted pair Ethernet. The ST-NIC also has an AUI Port which allows interface to thick wire Ethernet, or thin wire Ethernet by the addition of the DP8392 Coaxial Transceiver Interface (CTI). The dual DMA (local and remote) capabilities of the ST-NIC, along with 16 Kbytes of buffer RAM, allow the entire Network Interface Adapter to appear as a standard I/O Port to the system. The NIC module's local DMA channel buffers packets between the local memory (16 Kbytes of buffer RAM) and the network, while the NIC module's remote DMA channel passes data between the local memory and the system by way of an I/O Port. This I/O Port architecture which isolates the CPU from the network traffic proves to be the simplest method to interface the DP83902 to the system.

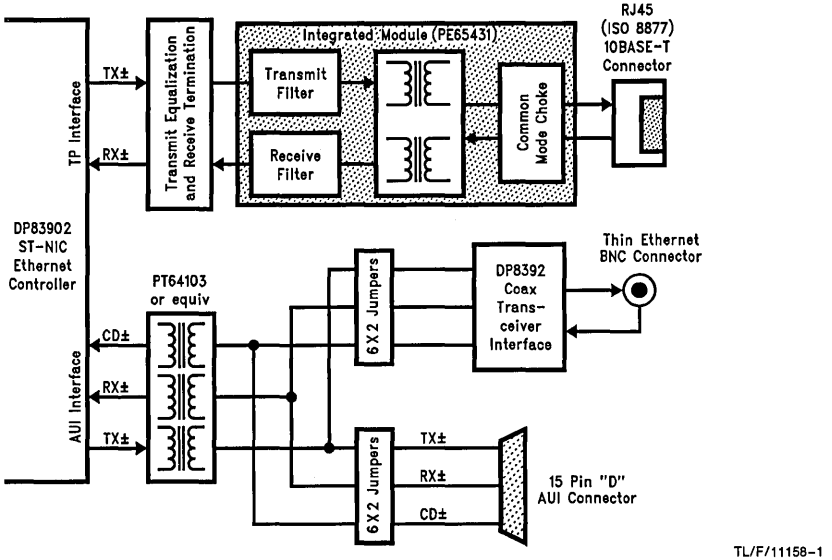
HARDWARE FEATURES

- Half-size IBM PC-AT I/O Card Form Factor
- Utilizes DP83902 Serial Network Interface Controller for Twisted Pair (ST-NIC)
- 16 Kbyte on-board Packet Buffer
- Simple I/O port interface to IBM PC-AT
- Interfaces to Thick Ethernet, Thin Ethernet, and Twisted Pair
- Boot EPROM Socket

The detailed schematics for this design are shown at the end of this document.

NETWORK INTERFACE OPTIONS

The evaluation board supports three physical layer options: Thick Ethernet, Thin Ethernet, and Twisted Pair. The block diagram for these interfaces can be seen in *Figure 1*. When using Thick Ethernet, a drop cable is connected to an external transceiver which is in turn connected to a standard Ethernet network, eliminating the need for an internal transceiver. This configuration may be obtained by connecting the pins on JB3 while leaving JB2 open, and connecting JB9 (AUTP) low.



TL/F/11158–1

FIGURE 1. Physical Layer Adapter Interface Block Diagram

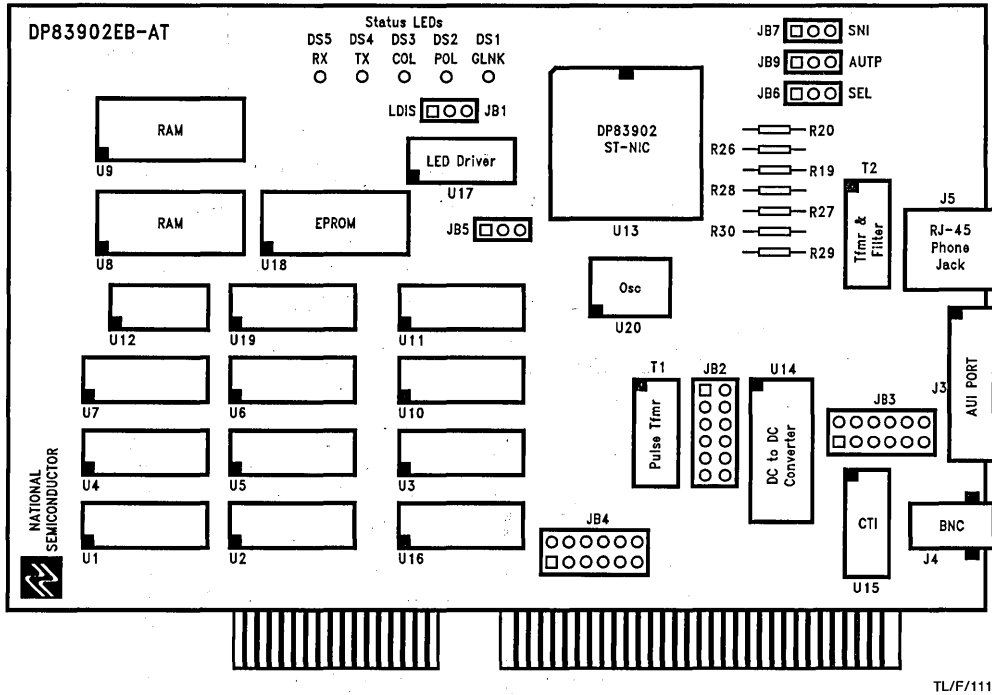
When using Thin Ethernet, a transceiver (the CTI) is available on-board to allow the direct connection to the network via the evaluation board. This transceiver (the CTI) forms the link between the differential ECL signals of the SNI module and the non-differential ECL signal of the thin-wire coaxial cable. A DC-DC Convertor is provided on the board to supply the CTI with −9V isolated voltage source. The Thin Ethernet solution is made by connecting the pins on JB2, leaving JB3 open and JB9 (AUTP) should be connected to Ground.

When using the Twisted Pair, JB9 (AUTP) needs to be connected to V_{CC}. The ST-NIC allows direct connection to the network using the RJ-45 phone jack. The remaining circuitry includes pre-emphasis resistors, a filter, a transformer/filter and a common mode choke. The transformer/filter decouples the DC component and eliminates any possible voltage spikes.

The diagram in *Figure 2* illustrates the layout of the board. It shows the various jumpers, ICs, LEDs, and the connectors for the three physical layer options. The transmit pre-emphasis resistors R27–R30 provide equalization to the twisted pair transmit outputs. This boosts the higher harmonics of the signal in order to compensate for losses in these harmonics over the twisted pair cable. R19 and R20 are 50Ω each and when combined form the required 100Ω termination on the receive side.

BUS INTERFACE

The block diagram, *Figure 3*, illustrates the architecture of the ST-NIC Evaluation Board. The ST-NIC Board as seen by the system appears only to be an I/O port. With this architecture the ST-NIC board has its own local bus to access the board memory. The system never has to intrude further than the I/O ports for any packet data operation. This I/O architecture isolates the system bus and the local bus, thereby preventing interference by the system when the ST-NIC is doing real-time accesses such as transmitting and receiving packets.



TL/F/11158–2

FIGURE 2. Layout of ST-NIC Evaluation Board

BOARD ARCHITECTURE

I/O Map of ST-NIC Board

The ST-NIC Board requires a 32-byte I/O space to allow for decoding the data buffers, the reset port, and the ST-NIC registers. The first 16 bytes (300h–30Fh) are used to address the ST-NIC registers (8 bits wide) and the next 8 bytes (310h–317h) are used to address the data buffers which are 16 bits wide. Finally, the reset port (also software selectable) may be addressed by 318h–31Fh.

TABLE I. I/O MAP in PC AT

Address	Part Addressed
300h–30Fh	ST-NIC Chip Select
310h–317h	Data Buffers
318h–31Fh	Reset

Although in the description above the I/O map is positioned at the addresses 300–31F, it may also be placed in the following address spaces: 320–33F, 340–35F, 360–37F. These alternate address spaces may be selected by the two jumpers pins JP1 and JP0 (refer to JB4 in *Figure 4*).

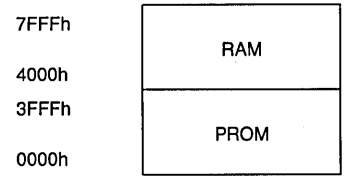
TABLE II. Optional Address Spaces

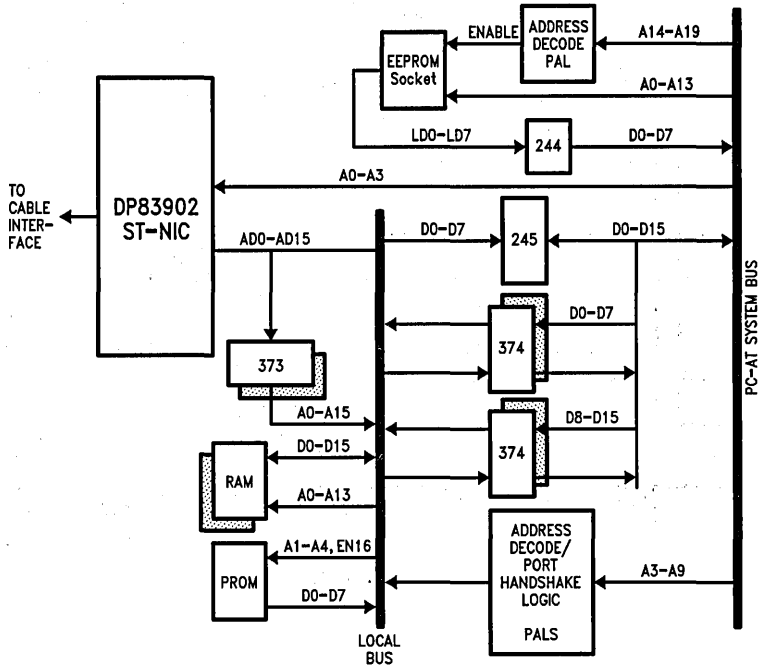
JUMPERS		
JP1	JP0	I/O Address Space
ON	ON	300h–31FH
ON	ON	320h–33FH
OFF	ON	340h–35FH
OFF	OFF	360h–37FH

DP83902's Local Memory Map

There are only two items mapped into the local memory space. These two items being the 8K x 16 buffer RAM and the ID address PROM. The buffer RAM is used for temporary storage of transmit and receive packets.

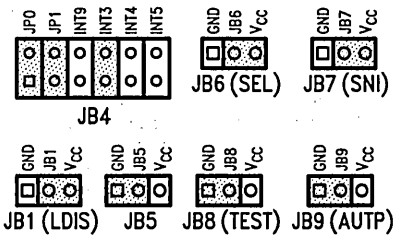
TABLE III. ST-NIC's Local Memory Map





TL/F/11158–3

FIGURE 3. Block Diagram of ST-NIC Evaluation Board's System Interface



TL/F/11158–4

FIGURE 4. Factory Configuration for JB1, JB4, JB5, JB6, JB7, JB8 and JB9

For transmit packets, the remote DMA puts data from the I/O ports into the RAM and the local DMA moves the data from the RAM to the ST-NIC. For the receive packets, the local DMA carries the data from the ST-NIC to the RAM and the remote DMA moves the data from the RAM to the I/O ports. The ID address PROM (74S288 32 x 8) contains the physical address of the evaluation board. Each PROM holds its own unique physical address which is installed during its manufacture. Besides this address, the PROM also contains some identification bytes that can be checked by the driver software. At the initialization of the evaluation board the software commands the ST-NIC to transfer the PROM data to the I/O Port where it is read by the CPU. The CPU then loads the ST-NIC's physical address registers. The following chart shows the contents of the PROM.

TABLE IV. PROM Contents

PROM Location	Location Contents
00h	Ethernet Address 0 (Most Significant Byte)
01h	Ethernet Address 1
02h	Ethernet Address 2
03h	Ethernet Address 3
04h	Ethernet Address 4
05h	Ethernet Address 5
06h–0Dh	00h
0Eh, 0Fh	57h
10h–15h	Ethernet Address 0 thru 5
16h–1Dh	Reserved
1Eh, 1Fh	42h

Data and Address Paths

The following paragraph may be better understood by looking at the block diagram shown earlier in *Figure 3*. Twenty address lines from the PC go onto the ST-NIC Board, but only four of them actually go to the ST-NIC. These four addresses along with the NIOR (low-asserted I/O read) or NIOW (low-asserted I/O write) and the CS (ST-NIC chip select signal) allow the PC to read or write to the ST-NIC's registers. If the system wants to read from or write to the ST-NIC registers, the data (only 8 bits) must pass through the 245 buffer. All of the packet data will pass through the I/O ports (the 374's). Each 374 is unidirectional and can only drive 8 bits, therefore it is necessary to have four 374's. Two of which drive data from the ports to the board memory and two of which drive the data from the ports to the AT bus. Even the PROM, which can only be addressed by the ST-NIC, sends its 8 bits of data out through the 374's. When the PROM does this, two of the 374's will be enabled but only the lower 8 bits will be read by the system. The RAM is also accessed by the ST-NIC. However, it is addressed by 14 bits and drives out 16 bits of data. The PALs receive 7 address lines among many other signals such as NIOR, NIOW, NACK, MRD, etc. With these signals the PALs do all of the decodes, such as selecting the ST-NIC Board, the ST-NIC chip, the RAM, and the PROM.

EPROM SOCKET

The EPROM socket is provided so that the user may add an EPROM to the system. This EPROM would normally contain a program and a driver to enable the PC-AT to be booted up through the network. The chips necessary to interface the EPROM to the system are the 27128 (EPROM), a 16L8 (PAL), and a 74ALS244 (buffer). Also, JB5 must be placed in the proper selection as described in the jumper section. The PAL decodes SA14–SA19, along with SMRDC (system memory read), in order to generate the EPROMEN signal. This signal, issued when the PC wants to execute the program stored in the EPROM, enables the EPROM and the 244 buffer.

EVALUATION BOARD OPERATION

The following pages will describe the slave accesses to the ST-NIC and the local DMA and remote DMA operation.

Register Operations

Accesses to the board are register operations to the DP83902, which are done to set up the ST-NIC and to control the operation of the ST-NIC's DMA channels.

REGISTER READ

To begin the register read, the CPU drives the four address lines (SA0–SA3) to the ST-NIC and the SA3–SA9 address lines to the PAL. These address lines are decoded by the PAL in order to generate a chip select to the ST-NIC. The CPU also drives the NIOR line which the ST-NIC sees as the NSRD (slave read). Once the ST-NIC receives this NSRD, it then sends out a high assertion on NACK, acknowledging that it is in slave mode but not yet ready to complete the read. The NACK signal is used by the PAL to assert the IOCHRDY (used to insert wait states) signal false. The ST-NIC then drives out the data from its internal registers to the 245 buffer. The 245 buffer is then enabled and the data is driven onto the AT BUS. When the ST-NIC is ready, it asserts NACK true and the PAL asserts IOCHRDY true. As a result, NIOR is driven high by the CPU, thereby deasserting the NSRD. On the rising edge of the NIOR, the data which is on the AT BUS is latched into the system. The addresses are removed at the same time, causing the ST-NIC chip select to become deasserted, ending the register read cycle.

REGISTER WRITE

To begin the register write, the CPU drives the SA0–SA3 address lines to the ST-NIC and the SA4–SA9 address lines to the PAL. With these address lines, the PAL decodes to 300–30F (the ST-NIC registers) thereby enabling the chip select for the ST-NIC. The CPU then drives the NIOW strobe which the ST-NIC sees as NSWR (slave write). Once the ST-NIC receives this NSWR it sends back a low assertion on NACK to acknowledge that it is in slave mode and ready to perform the write. When the CPU receives this signal, it puts data out onto the AT BUS where it goes into the 245 buffer. The 245 buffer then drives the data to the ST-NIC, but the data is not latched into the ST-NIC until the rising edge of NIOW. The system drives NIOW high, thereby deasserting the NSWR and latching the data. The addresses also are taken away and the chip select then goes high (deasserted). This ends the cycle of the register write.

Remote Transfers

Remote DMA transfers are operations performed by the ST-NIC on the board. These operations occur when the ST-NIC is programmed to transfer packet data between the PC-AT and the card's on-board RAM. These transfers take place through the I/O Port interfacing.

REMOTE READ

To program the ST-NIC for a remote read, the CPU must make five slave accesses to the ST-NIC. The CPU must write the Remote Start Address (2 bytes), the Remote Byte Count (2 bytes), and issue the Remote DMA Read Command. The addresses and byte count require two transfers because they are both 16 bits, yet only 8 bits can be written per transfer.

Once the ST-NIC has received all of the above data, it drives out BREQ and waits for BACK. The ST-NIC immediately receives BACK because it is tied to the BREQ line. (BREQ can be tied to BACK because there are no other devices contending for the local bus.) After receiving BACK, the ST-NIC drives out the address from which the data is required to be read. This address flows into the 373's and is latched by ADS0. From here, the address flows to the RAM. The RAM waits until it receives MRD from the ST-NIC and then it drives the data into the 374 ports. The 374 ports then latch the data on the rising edge of the PWR strobe from the ST-NIC. PRQ is then sent out by the ST-NIC to let the system know that there is data waiting in the ports.

If the AT reads the I/O ports before the ST-NIC has loaded the 374's, then the port request (PRQ) from the ST-NIC will not yet be driven. This unasserted PRQ signal causes the AT's ready line to be set low, indicating that the ST-NIC has yet to load the data. After the data is in the ports, the system must then read the 374 data ports. This begins with the AT driving out an address which is decoded (inside the PAL) to the data I/O Ports (310–31F). The PAL then drives RACK to the ST-NIC, indicating that the CPU is ready to accept data. This RACK signal then reads the data from the 374 ports onto the AT BUS. The system deasserts NIOR which finishes the cycle.

REMOTE WRITE

Like the remote read, the remote write cycle also begins with five slave accesses into the internal registers. The CPU must write the Remote Start Address (2 bytes), the Remote Byte Count (2 bytes), and issue the Remote DMA Write Command. The ST-NIC then issues a PRQ. The CPU responds by sending an NIOW, indicating that it is ready to write to the ports. The CPU also drives out the address which corresponds to the I/O Ports. This address goes into the PAL and helps to decode to WACK. This WACK signal latches the data into the 374 ports The ST-NIC issues a BREQ and immediately receives a BACK since the two lines are tied together. (BREQ can be tied to BACK because there are no other devices contending for the local bus.) The ST-NIC, upon receiving the BACK, drives out address lines to the 373's. These address lines are latched by ADS0 and then are driven to the RAM. ST-NIC sends out a PRD and a NMWR which drives the data from the 374 ports into the already specified address of the onboard memory. PRD and NMWR are then deasserted and the cycle ends.

Network Transfers

Transfers to and from the network are controlled by the DP83902's local DMA channel which transfers packet data to/from the ST-NIC's internal FIFO from/to the card buffer's RAM.

RECEIVE

The data comes off of the network, is deserialized and is stored in the FIFO inside of the ST-NIC. The ST-NIC then issues a BREQ and immediately receives BACK since the lines are tied together. After receiving BACK, the ST-NIC drives the address lines to the 373's. The 373's are latched by ADS0 and the address is allowed to flow to the RAM. Then the ST-NIC drives out NMWR along with the data from the FIFO. The data flows into the RAM under the address given earlier. The NMWR strobe is then deasserted, ending the cycle.

TRANSMIT

To begin the transmit cycle, the ST-NIC issues a BREQ and waits for the BACK. Since the BREQ and BACK lines are tied together, the BACK signal is received immediately. Upon reception of this signal, the ST-NIC drives out the address to the 373's which latch the address with the ADS0 strobe. The address then flows to the onboard memory. NMRD, driven by ST-NIC, causes the RAM to drive the data out of the given address and into the ST-NIC. The ST-NIC then latches the data into the FIFO on the rising edge of NMRD. This high assertion of NMRD signifies the ending of this cycle. From the FIFO, the data is serialized and transmitted onto the network.

BOARD CONFIGURATION

On the DP83902EB-AT ST-NIC AT board, there are nine jumper blocks as seen in the diagram below. The following pages wiill explain how to configure these jumpers.

Physical Layer

TABLE V. Physical Layer Selection

JB9	JB2	JB3	Physical Layer Selected
High	X	X	Twisted Pair
Low	On	Off	Thin Ethernet
Low	Off	On	Thick Ethernet

If JB9 is tied to V_{CC} then the twisted pair interface will be selected. If JB2 is closed while JB3 is open, and JB9 is connected to Ground, then the Thin Ethernet option will be selected. And finally if JB3 is closed while JB2 is open, and JB9 is low, then the Thick Ethernet option will be selected.

Interrupt Lines, Board Addresses, and EPROM Addresses

On JB4, there are six possible connections. Four of these are to select an interrupt line. The available interrupt lines include INT3, INT4, INT5, and INT9. The last two possible connections, JP1 and JP0, are used to select the base address for the board. However, if JB5 is connected to V_{CC}, then these last two connections select the address of the EPROM also. The possible selections and the jumpers are shown in Table VI. The factory configuration uses the INT3 line for interrupts and has JP1 and JP0 in the on position. This factory configuration is shown in *Figure 4*, along with the factory configurations for JB1, JB5, JB6, JB7, JB8, and JB9. The square pin indicates pin 1 of the jumpers.

TABLE VI. Base Address and EPROM Address

JP1	JP0	Base Address	EPROM Address
On	On	300h–31Fh	C800h
On	Off	320h–33Fh	CC00h
Off	On	340h–35Fh	D000h
Off	Off	360h–37Fh	D400h

PAL EQUATIONS

PAL #1 (U1)

In this first PAL, the output signals are NIO16, NIOEN, NSTNICB, and NCSROM. (The N's before the signals indicate that the signal is low asserted.) Since it is necessary to assert NIO16 as soon as possible, this first PAL has been selected to be a 10 ns "D" PAL. The NIO16 signal must be TRI-STATE® when it is not asserted. Therefore, we use an enable signal (NIOEN) which is equal to the decode for the I/O Ports (310–31F) and NAEN high (NAEN high signifies that the system DMA does not have control of the bus). The enable signal (NIOEN) loops back into the PAL to bring NIO16 out of TRI-STATE. The NIO16 signal is set to zero so that whenever it is enabled it will be asserted.

The STNICB signal consists of simple address decodes along with NAEN. The addresses decode to one of four address slots which were earlier mentioned in the board configuration section. The NCSROM is a very simple signal as it consists only of AD14 and NMRD. AD14 comes from the ST-NIC and selects either the PROM (when low) or the onboard RAM (when high).

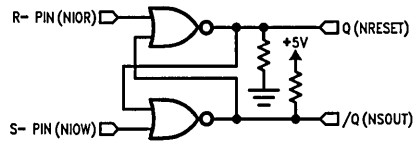
PAL 1

```
module iodec;
flag '-r1';
title '
date: 9/13/89
functions:
ST-NIC BOARD DECODE, IO16 DECODE, AND CHIP SELECT PROM';
u1 device 'p16l8';

"input pins:
NEN16, NAEN, SA9          pin 1, 2, 3;
SA8, SA7, SA6             pin 4, 5, 6;
SA5, SA4, SA3             pin 7, 8, 9;
JP0, JP1, NMRD            pin 13, 14, 15;
A14                       pin 16;
"output pins:
NSTNICB, NIOEN, NIO16     pin 12, 17, 18;
NCSROM                    pin 19;

"constants
    X = .X.;
    Z = .Z.;

equations
NSTNICB = !(!NAEN & SA9 & SA8 & !SA7 & !SA6 & !SA5 & !JP1 & !JP0
          # !NAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & !JP1 & JP0
          # !NAEN & SA9 & SA8 & !SA7 & SA6 & !SA5 & JP1 & !JP0
          # !NAEN & SA9 & SA8 & !SA7 & SA6 & SA5 & JP1 & JP0);
NIOEN = !(!NAEN & SA9 & SA8 & !SA7 & !SA6 & !SA5 & !JP1 & !JP0
        & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & !JP1 & JP0
         & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & SA6 & !SA5 & JP1 & !JP0
         & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & SA6 & SA5 & JP1 & JP0
         & !NEN16 & SA4 & !SA3);
NCSROM = !(!A14 & !NMRD);
enable NIO16 = !NIOEN;
       NIO16 = 0;
end iodec;
```

TL/F/11158–5

PAL #2

In this PAL, there are eight outputs which include NRESET, NSOUT, NRDYEN, NIOCHRDY, NCS, NRACK, and NWACK. The first two outptus (NRESET and NSOUT) are part of an R–S flip flop as shown below:



TL/F/11158–6

FIGURE 5. RS Flip-Flop

NRESET is given by the NOR of the high asserted R-input pin and the NSOUT signal. NSOUT is given by the NOR of the high asserted S-input pin and the NRESET signal. The NOR gates are enabled by the low assertion of NRSTDRV. When the system first boots up, it will disable the NOR gates by asserting the RSTDRV signal. But due to the pull-up and pull-down resistors, the output <NRESET, NSOUT> will be set to <0, 1>. Once RSTDRV becomes deasserted, the output will remain at <0, 1>. The only way to get out of reset is to assert the S-pin high which is done by an NIOW and an address decode to 318–31F. After the system has booted up, the ST-NIC may be reset through software. This would be done by setting the R-pin high with an NIOR and an address decode to 318–31F. To escape from reset, we once again set the S-pin high with an NIOW and address decode of 318–31F. The above description of logic is also shown in Truth Table VII.

TABLE VII. R–S Flip Flop Truth Table

R (NIOR)	S (NIOW)	Q (NRESET)	$\overline{Q}$ (NSOUT)
0	0	0	1
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1
0	1	1	0

By using the NIOR and NIOW which are never asserted at the same time, this insures that the R-pin and the S-pin will never be asserted at the same time. The next two signals (NRDYEN and NIOCHRDY) are quite similar to NIOEN and NIO16 in PAL#1. All of the decode takes place in the enable signal (NRDYEN). This decode consists of addresses 300–30F without NACK or the addresses 310–318 without PRQ. If the NRDYEN signal is asserted, then NIOCHRDY will be driven low. At all other times, the NIOCHRDY strobe will be in TRI-STATE. This PAL must also be a 10 ns "D" PAL.

NCS is decoded by NSTNICB (from PAL #1) along with the low assertion of SA4 and either NIOR or NIOW. Its decode is in the address range of 300–30F. The last two signals are NRACK and NWACK. NRACK occurs with an address decode to 310–318, an NIOR, and a PRQ. The NWACK signal only differs from the NRACK by the NIOR/NIOW signal and therefore consists of an address decode to 310–31F, an NIOW and a PRQ. INT is just sent through the PAL to be buffered. The buffered signal which comes out of the PAL is INTO.

PAL2

```
module reset;
flag '-r1';
title '
date: 9/13/89
functions:
RESET LATCH, STNIC SELECT, IOCHRDY, RACK, WACK,
BUFFER INTERRUPT';
u2 device 'p16l8';

"input pins:
NSTNICB, NIOW, NIOR      pin 1, 2, 3;
RSTDRV, NACK, PRQ        pin 4, 5, 6;
SA4, SA3, INT            pin 7, 8, 9;

"output pins:
INTO, NRACK, NWACK       pin 12, 13, 14;
NRESET, NSOUT, NRDYEN    pin 15, 16, 17;
NIOCHRDY, NCS            pin 18, 19;

"constants
    X = .X.;
    Z = .Z.;

equations
NCS = !(!NSTNICB & !NIOR & !SA4 # !NSTNICB &!NIOW &!SA4);
NRACK = !(!NSTNICB & PRQ & !NIOR & SA4 & !SA3);
NWACK = !(!NSTNICB & PRQ & !NIOW & SA4 & !SA3);
NRDYEN = !(!NSTNICB & !NIOR & !SA4 & NACK
         # !NSTNICB & !NIOW & !SA4 & NACK
         # !NSTNICB & !PRQ & !NIOR & SA4 & !SA3
         # !NSTNICB & !PRQ & !NIOW & SA4 & !SA3);
enable NIOCHRDY =!NRDYEN;
       NIOCHRDY = 0;
enable NRESET = !RSTDRV;
       NRESET = !(!NSTNICB & !NIOR & SA4 & SA3 # NSOUT);
enable NSOUT = !RSTDRV;
       NSOUT = !(!NIOW # NRESET);
INTO = INT;
end reset;
```

TL/F/11158–7

PAL #3

The third PAL only does a decode to enable the optional EPROM. This decode consists of an address decode to C800h, CC00h, D000h, or D400h depending on JP1 and JP0 as shown in the board configuration section. JP2 must also be jumpered for selection of the EPROM. NAEN, a low asserted signal should be high to indicate that the DMA does not have control of the bus and the NSMRDC signal should be asserted high since the CPU is doing a system memory read.

PAL 3

```
module epromdec;
flag '-r0';
title '
date: 9/13/89
function:
EPROM DECODE';

u16 device 'p16l8';

"input pins:

A0, A13, EN16            pin 1, 2, 3;
SMRDC, SA19, SA18        pin 4, 5, 6;
A17, SA16, SA15          pin 7, 8, 9;
A14, NAEN, JP2           pin 11, 13, 14;
P0, JP1                  pin 15, 16;

"output pins:

EPROMEN                  pin 19;
A013                     pin 12;

"constants
    X = .X.;

equations

NEPROMEN = !(SA19 & SA18 & !SA17 & !SA16 & SA15 & !SA14 & !NAEN
         & JP2 & !JP1 & !JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & !SA16 & SA15 & SA14 & !NAEN
         & JP2 & !JP1 & JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & SA16 & !SA15 & !SA14 & !NAEN
         & JP2 & JP1 & !JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & SA16 & !SA15 & SA14 & !NAEN
         & JP2 & JP1 & JP0 & !NSMRDC);

A013 = !(!A0 & EN16 # !A13 & !EN16);

end epromdec;
```

TL/F/11158–8

BILL OF MATERIALS FOR DP83902EB-AT ST-NIC ETHERNET ADAPTER

CAPACITORS

C1, C20.01 μF		50V	10%	Monolythic
C3	4.7 μF	25V	20%	Tantalum
C4	0.01 μF	1 KV	10%	Ceramic Disk
C5	0.01 μF	50V	10%	Ceramic Disk
C9..C10	0.01 μF	50V	20%	Monolythic
C11	0.75 μF	1 KV		Spark Gap
C12..C19	0.01 μF	50V	20%	Monolythic
C20	4.7 μF 25V	20%	Tantalum	
C21	4.7 μF 25V	20%	Tantalum	
C22..C29	0.01 μF	50V	20%	Monolythic
C30..C32	4.7 μF 25V	20%	Tantalum	
C33..C37	0.01 μF	50V	20%	Monolythic
C38..C42	4.7 μF 25V	20%	Tantalum	

RESISTORS

R1..R3	10K	1/4W	5%
R4..R6	4.7K	1/4W	5%
R7..R10	39.2	1/4W	1%
R11	1M	1/2W	5%
R12, R13	270	1/4W	5%
R14..R17	1.5K	1/4W	5%
R18	1K	1/4W	1%
R19..R20	TBD	1/4W	1%
R21	420	1/4W	5%
R22..R25	430	1/4W	5%
R26	4.7K	1/4W	5%
R27..R30	TBD	1/4W	1%
R31..R34	4.7K	1/4W	5%

IC's

U1, U2	16L8D	PAL
U16	16L8B	PAL
U3	74ALS245	
U4..U7	74ALS374	
U8, U9	HM6264	8K x 8 STATIC RAM 100 ns
U10, U11	74AS373	
U12	74S288	PROM
U13	DP83902	ST-NIC
U15	DP8392	CTI
U17	74HC04N	
U18	27128	EPROM (not supplied on board)
U19	74ALS244	
U20	20 MHz 0.01%	Crystal Oscillator

MAGNETICS

U14 PM7102 VALOR DC-DC Convertor.
T1 PE64103 Pulse Engineering
T2 RX and TX Filter & Transformer. Pulse Engineering PE65431.

MISCELLANEOUS

DS1	GREEN	5mm LOW CURRENT LED	CURRENT=I_F=3.5m
DS2	AMBER	5mm LOW CURRENT LED	CURRENT=I_F=3.5 mA
DS3	RED	5mm LOW CURRENT LED	CURRENT=I_F=2.0 mA
DS4	YELLOW	5mm LOW CURRENT LED	CURRENT=I_F=2.0 mA
DS5	GREEN	5mm LOW CURRENT LED	CURRENT=I_F=3.5 mA
JB1		1x3 SHUNT BLOCK WITH .1" SPACING BETWEEN PINS	
JB2		2x6 SHUNT BLOCK WITH .1" SPACING BETWEEN PINS	
JB3		2x6 SHUNT BLOCK WITH .1" SPACING BETWEEN PINS	
JB4		2x6 SHUNT BLOCK WITH .1" SPACING BETWEEN PINS	
JB5-JB9		1x3 SHUNT BLOCK WITH .1" SPACING BETWEEN PINS	

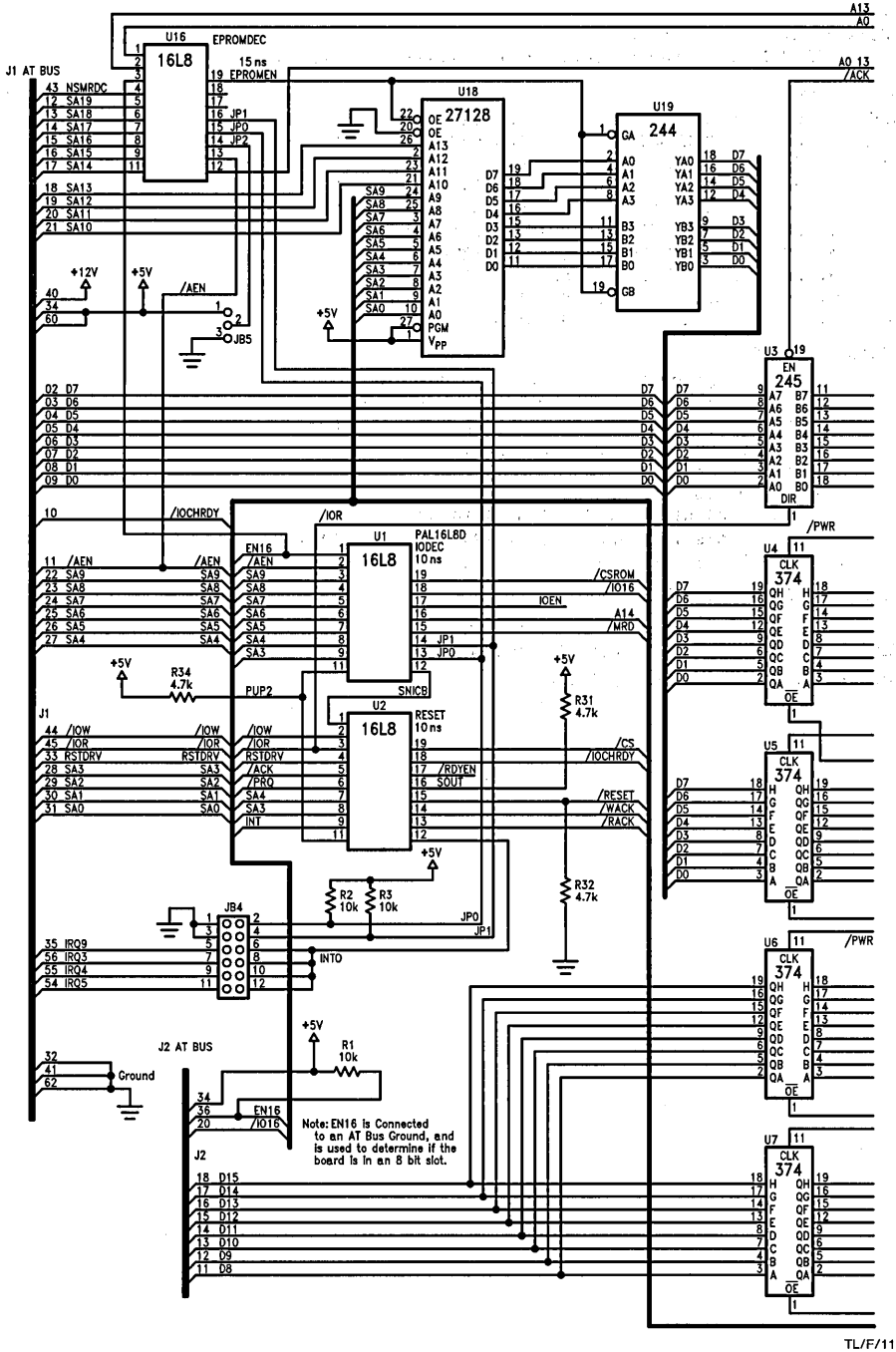
SOCKETS/MECHANICAL

S1-S3	20 PIN 0.3" DUAL IN-LINE FOR U1, U2, U16 (PAL)
S4	28 PIN DUAL IN-LINE SOCKET FOR U18 (EPROM)
S5	84 PIN PLCC SOCKET FOR U13 (ST-NIC) AMP SOCKET
S8	BRACKET FOR MOUNTING IN PC-AT, SLOT G44 Basic Blank,
J3	RJ-45 CONNECTOR AMP 520252-4
J4	BNC CONNECTOR RT/A Low Pro Amp 227161-7
J5	15 PIN D CONNECTOR Female AMP 747247-4 (or 747845-4) MAXCON SUB D Slide Lock MDA 51220-1

BOARD ATTACHMENT COMPONENTS

1) Screw: Bind Head Slotted 4-40 x .250, Steel, (90277A106)
2) Washer: Lock Ext #4, Zinc/Steel, (91114A005)
3) Washer: Flat #4, Zinc-CRS, (90126A005)

Bus Interface/NIC Section

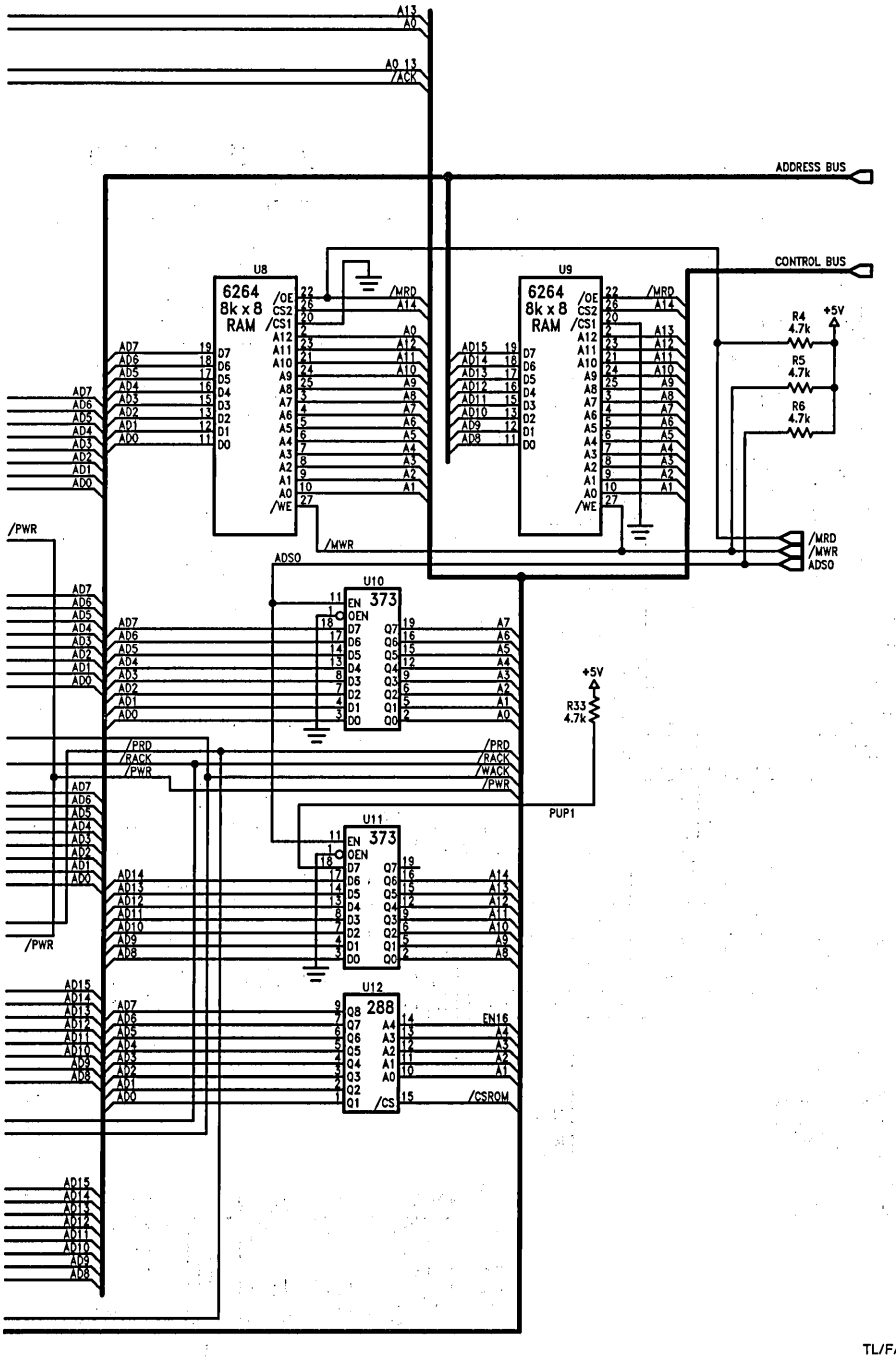


TL/F/11158–9

Note: All resistors to be 5%, ¼W unless otherwise indicated

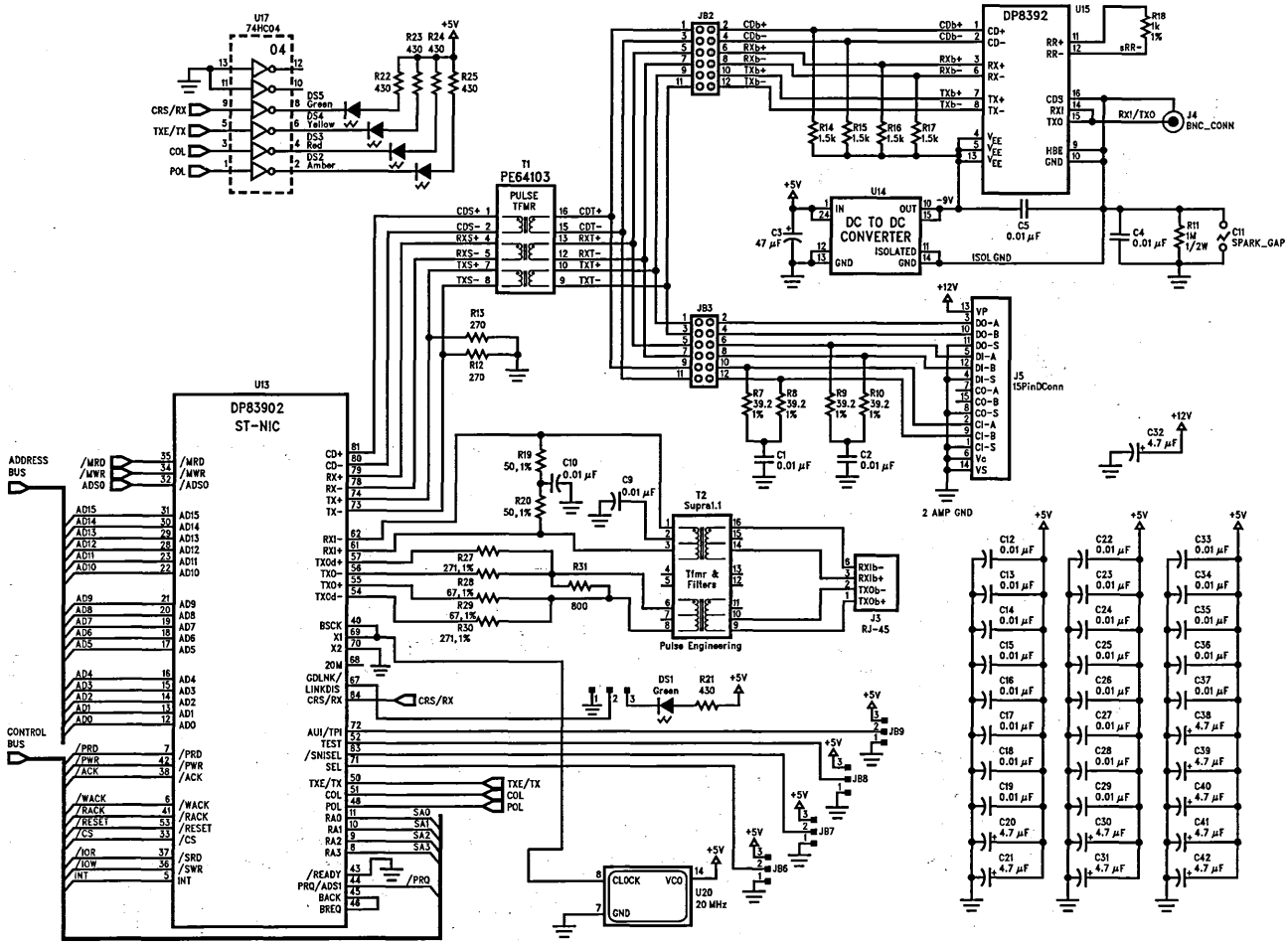
Note: EN16 is actually a ground signal on the AT Bus J2 Connector. This signal is used to determine whether 8- or 16-bit mode should be used.

DP83902EB-AT ST-NIC Ethernet Evaluation Board Schematic
(Bus Interface/NIC Section)



TL/F/11158–10

DP83902EB-AT ST-NIC Ethernet Evaluation Board Schematic (Continued)
(Bus Interface/NIC Section)



TL/F/11158–11

Note: All resistors to be 5%, ¼W unless otherwise indicated

Note: Capacitors 39 to 42 are for the V_{CC} signals off of the AT Bus.

DP83902EB-AT PC-AT Ethernet Evaluation Board

DP839EB-ATN IBM® PC-AT® Compatible DP83901 Serial Network Interface Controller (SNIC) Evaluation Board

National Semiconductor
Application Note 729
Larry Wakeman

OVERVIEW

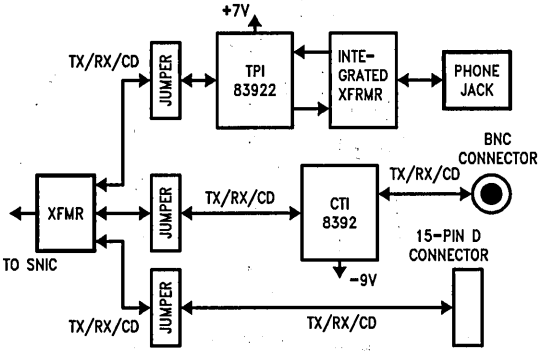
The National Semiconductor SNIC Evaluation Board design provides IBM AT's and AT Compatibles with Thick Ethernet, Thin Ethernet, and Twisted Pair connections. This low parts count evaluation board is compatible with the AT bus and requires only a ½ size slot for insertion. Besides using the DP83901 Serial Network Interface Controller (SNIC), the Coaxial Transceiver Interface (CTI) and the Twisted Pair Interface (TPI) are also employed. The dual DMA (local and remote) capabilities of the SNIC, along with 16 kbytes of buffer RAM, allow the entire Network Interface Adapter to appear as a standard I/O Port to the system. The NIC's local DMA channel buffers packets between the local memory (16 kbytes of buffer RAM) and the network, while the NIC's remote DMA channel passes data between the local memory and the system by way of an I/O Port. This I/O Port architecture which isolates the CPU from the network traffic proves to be the simplest method to interface the DP83901 to the system.

HARDWARE FEATURES

- Fits in half-size IBM PC-AT I/O card form factor
- Utilizes DP83901 Serial Network Interface Controller (SNIC) and DP83922 Twisted Pair Transceiver (TPI)
- 16 kbyte on-board packet buffer
- Simple I/O port interface to IBM PC-AT
- Interfaces to thick Ethernet, thin Ethernet, and twisted pair
- Boot EPROM socket

NETWORK INTERFACE OPTIONS

The evaluation board supports three physical layer options: Thick Ethernet, Thin Ethernet, and Twisted Pair. These can be seen in *Figure 1*. When using Thick Ethernet, a drop cable is connected to an external transceiver which is in turn connected to a standard Ethernet network. For this physical layer, there is no need for an internal transceiver since it already has an external one. This configuration may be obtained by connecting the pins on JB3 while leaving JB1 and JB2 open. When using Thin Ethernet, a transceiver (the CTI) is available on-board to allow the direct connection to the network via the evaluation board. This transceiver (the CTI) forms the link between the differential ECL signals of the SNI and the non-differential ECL signal of the thin-wire coaxial cable. For proper operation, the CTI requires a DC–DC Converter to provide an isolated ground and a −9V source. In order to put this Thin Ethernet solution into operation, the pins on JB2 need to be connected while JB1 and JB3 should be open.



TL/F/10800–1

FIGURE 1. Physical Layer Adapter Interface Block Diagram

When using the Twisted Pair (telephone cable), another transceiver (the TPI) is available on-board which also allows direct connection to the network. The voltage regulator (the LM317) provides the TPI chip with a constant +7V supply. The remaining portion of the TPI circuit includes a common mode choke and a transformer. The transformer decouples the DC component and eliminates any possible voltage spikes. This twisted pair interface can be selected by using JB1 and leaving JB2 and JB3 open. Each of these three physical layer options can be selected simply by the placement of a jumper block (no software changes are needed).

BUS INTERFACE

The block diagram, *Figure 2*, illustrates the architecture of the SNIC Evaluation Board. The SNIC Board as seen by the system appears only to be an I/O port. With this architecture the SNIC board has its own local bus to access the board memory. The system never has to intrude further than the I/O ports for any packet data operation. This I/O architecture isolates the system bus and the local bus, thereby preventing interference by the system when the SNIC is doing real-time accesses such as transmitting and receiving packets.

BOARD ARCHITECTURE

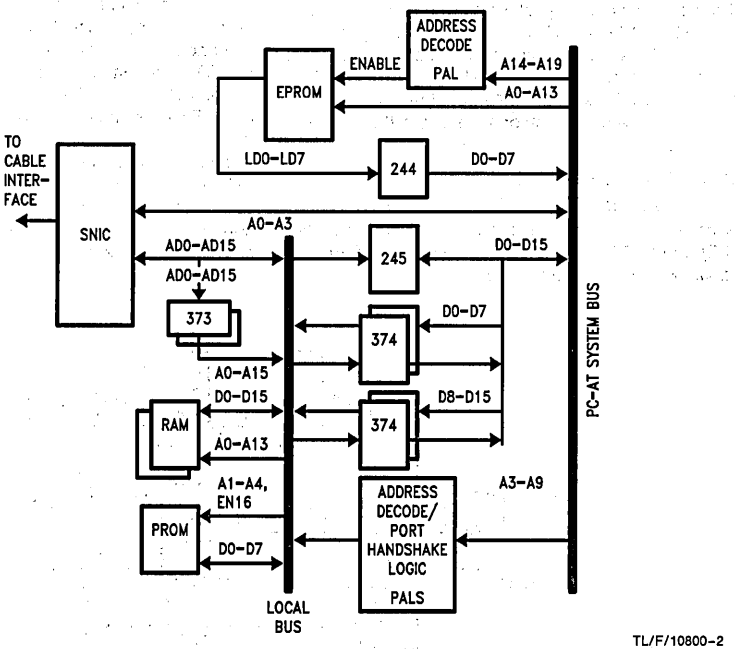
I/O Map of SNIC Board

The SNIC Board requires a 32-byte I/O space to allow for decoding the data buffers, the reset port, and the SNIC registers. The first 16 bytes (300h–30Fh) are used to address the SNIC registers (8 bits wide) and the next 8 bytes (310h–317h) are used to address the data buffers which are 16 bits wide. Finally, the reset port may be addressed by 318h–31Fh.

TABLE I. I/O Map In PC-AT

Address	Part Addressed
300h–30Fh	SNIC Chip Select
310h–317h	Data Buffers
318h–31Fh	Reset

Although in the description above the I/O map is positioned at the addresses 300–31F, it may also be placed in the following address spaces: 320–33F, 340–35F, 360–37F. These alternate address spaces may be selected by the two jumpers (JP1 and JP0).



TL/F/10800–2

FIGURE 2. Block Diagram of SNIC Evaluation Board's System Interface

TABLE II. Optional Address Spaces

Jumpers		
JP1	JP0	I/O Address Space
On	On	300h-31Fh
On	Off	320h-33Fh
Off	On	340h-35Fh
Off	Off	360h-37Fh

DP83901's Local Memory Map

There are only two items mapped into the local memory space. These two items being the 8k x 16 buffer RAM and the ID address PROM. The buffer RAM is used for temporary storage of transmit and receive packets. For transmit packets, the remote DMA puts data from the I/O ports into the RAM and the local DMA moves the data from the RAM to the SNIC. For the receive packets, the local DMA carries the data from the SNIC to the RAM and the remote DMA moves the data from the RAM to the I/O ports. The ID address PROM (74S288 32 x 8) contains the physical address of the evaluation board. Each PROM holds its own unique physical address which is installed during its manufacture. Besides this address, the PROM also contains a checksum. This checksum, calculated by exclusive OR-ing the six address bytes with each other, is provided in order to check the addresses. At the initialization of the evaluation board the software commands the SNIC to transfer the PROM data to the I/O Port where it is read by the CPU. The CPU then verifies the checksum and loads the SNIC's physical address registers. The following chart shows the contents of the PROM.

TABLE III. SNIC's Local Memory Map

7FFFh	RAM
4000h	
3FFFh	PROM
0000h	

TABLE IV. PROM Contents

PROM Location	Location Contents
00h	Ethernet Address 0 (Most Significant Byte)
01h	Ethernet Address 1
02h	Ethernet Address 2
03h	Ethernet Address 3
04h	Ethernet Address 4
05h	Ethernet Address 5
06h–0Dh	00h
0Eh	57h
0Fh	57h
10h–15h	Same as 00h–05h
16h–1Dh	00h
1Eh	42h
1Fh	42h

Data and Address Paths

The following paragraph may be better understood by looking at the block diagram shown earlier, in *Figure 2*. Twenty address lines from the PC go onto the SNIC Board, but only four of them actually go to the SNIC. These four addresses along with the NIOR (low-asserted I/O read) or NIOW (low-asserted I/O write) and the $\overline{CS}$ (SNIC chip select signal) allow the PC to read or write to the SNIC's registers. If the system wants to read from or write to the SNIC registers, the data (only 8 bits) must pass through the 245 buffer. All of the packet data will pass through the I/O ports (the 374's). Each 374 is unidirectional and can only drive 8 bits, therefore it is necessary to have four 374's. Two of which drive data from the ports to the board memory and two of which drive the data from the ports to the AT bus. Even the PROM, which can only be addressed by the SNIC, sends its 8 bits of data out through the 374's. When the PROM does this, two of the 374's will be enabled but only the lower 8 bits will be read by the system. The RAM is also accessed by the SNIC. However, it is addressed by 14 bits and drives out 16 bits of data. The PALs® receive 7 address lines among many other signals such as NIOR, NIOW, NACK, MRD, etc. With these signals the PALs do all of the decodes, such as selecting the SNIC Board, the SNIC chip, the RAM, and the PROM.

EPROM Socket

The EPROM socket is provided so that the user may add an EPROM to the board. This EPROM would normally contain a program and a driver to enable the PC-AT to be booted up through the network. The chips necessary to interface the EPROM to the system are the 27128 (EPROM), a 16L8 (PAL), and a 74ALS244 (buffer). Also, JB8 must be placed in the proper selection as described in the jumper section. A PAL decodes SA14–SA19, along with SMRDC (system memory read), in order to generate the EPROMEN signal. This signal, issued when the PC desires to execute the program contained in the EPROM, enables the EPROM and the 244 buffer.

EVALUATION BOARD OPERATION

The following pages will describe the slave accesses to the SNIC and the local DMA and remote DMA operation.

Register Operations

Accesses to the board are register operations to the DP83901, which are done to set up the SNIC and to control the operation of the SNIC's DMA channels.

Register Read

To begin the register read, the CPU drives the four address lines (SA0–SA3) to the SNIC and the SA3–SA9 address lines to the PAL. These address lines are decoded by the PAL in order to generate a chip select to the SNIC. The CPU also drives the NIOR line which the SNIC sees as the NSRD (slave read). Once the SNIC receives this NSRD, it then sends out a high assertion on NACK, acknowledging that it is in slave mode but not yet ready to complete the read. The NACK signal is used by the PAL to assert the IOCHRDY signal false. The SNIC then drives out the data from its internal registers to the 245 buffer. The 245 buffer is then enabled and the data is driven onto the AT BUS. When the SNIC is ready, it asserts NACK true and the PAL asserts IOCHRDY true. As a result, NIOR is driven high by

the CPU, thereby deasserting the NSRD. On the rising edge of the NIOR, the data which is on the AT BUS is latched into the system. The addresses are removed at the same time, causing the SNIC chip select to become deasserted and therefore ending the register read cycle.

Register Write

To begin the register write, the CPU drives the SA0–SA3 address lines to the SNIC and the SA4–SA9 address lines to the PAL. With these address lines, the PAL decodes to 300–30F (the SNIC registers) thereby enabling the chip select for the SNIC. The CPU then drives the NIOW strobe which the SNIC sees as NSWR (slave write). Once the SNIC receives this NSWR it sends back a low assertion on NACK to acknowledge that it is in slave mode and ready to perform the write. When the CPU receives this signal, it puts data out onto the AT BUS where it goes into the 245 buffer. The 245 buffer then drives the data to the SNIC, but the data is not latched into the SNIC until the rising edge of NIOW. The system drives NIOW high, thereby deasserting the NSWR and latching the data. The addresses also are taken away and the chip select then goes high (deasserted). This thereby ends the cycle of the register write.

Remote Transfers

Remote DMA transfers are operations performed by the SNIC on the board. These operations occur when the SNIC is programmed to transfer packet data between the PC-AT and the card's on-board RAM. These transfers take place through the I/O Port interfacing.

Remote Read

To program the SNIC for a remote read, the CPU must make five slave accesses to the SNIC. The CPU must write the Remote Start Address (2 bytes), the Remote Byte Count (2 bytes) and issue the Remote DMA Read Command. The addresses and byte count require two transfers because they are both 16 bits, yet only 8 bits can be written per transfer.

Once the SNIC has received all of the above data, it drives out BREQ and waits for BACK. For this design the SNIC immediately receives the BACK because it is tied to the BREQ line (BREQ can be tied to BACK because there are no other devices contending for the local bus). After receiving the BACK, the SNIC drives out the address from which the data will be read. This address flows into the 373's and is latched by ADS0. From here, the address flows to the RAM. The RAM waits until it receives NMRD from the SNIC and then it drives the data out of the address it was given and into the 374 ports. The 374 ports then latch the data on the rising edge of the NPWR strobe from the SNIC. PRQ is then sent out by the SNIC to let the system know that there is data waiting in the ports.

If the AT reads the I/O ports before the SNIC has loaded the 374's, then the port request (PRQ) from the SNIC will not yet be driven. This unasserted PRQ signal causes the AT's ready line to be set low, indicating that the SNIC has yet to load the data. After the data is in the ports, the system must then read the 374 data ports. This begins with the AT driving out an address which is decoded (inside the PAL) to the data I/O Ports (310–317). The PAL then drives RACK to the SNIC, indicating that the CPU is ready to accept data. This RACK signal then reads the data from the 374 ports onto the AT BUS. The system deasserts NIOR which finishes the cycle.

Remote Write

Like the remote read, the remote write cycle also begins with five slave accesses into the internal registers. The CPU must write the Remote Start Address (2 bytes), the Remote Byte Count (2 bytes) and issue the Remote DMA Read Command. The SNIC then issues a PRQ. The CPU responds by sending a NIOW, indicating that it is ready to write to the ports. The CPU also drives out the address which corresponds to the I/O Ports. This address goes into the PAL and helps to decode to WACK. This WACK signal latches the data into the 374 ports. The SNIC issues a BREQ and immediately receives a BACK since the two lines are tied together (BREQ can be tied to BACK because there are no other devices contending for the local bus). The SNIC, upon receiving the BACK, drives out address lines to the 373's. These address lines are latched by ADS0 and then are driven to the RAM. SNIC sends out a PRD and a MWR which drives the data from the 374 ports into the already specified address of the onboard memory. Soon afterwards, the PRD and the MWR are deasserted and the cycle ends.

Network Transfers

Transfers to and from the network are controlled by the DP83901's local DMA channel which transfers packet data to/from the SNIC's internal FIFO from/to the card's buffer RAM.

Receive

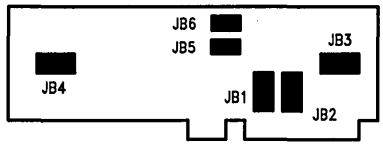
The data comes off of the network, is deserialized and is stored in the FIFO inside of the SNIC. The SNIC then issues a BREQ and immediately receives BACK since the lines are tied together. After receiving BACK, the SNIC drives the address lines to the 373's. The 373's are latched by ADS0 and the address is allowed to flow to the RAM. Then the SNIC drives out NMWR along with the data from the FIFO. The data flows into the RAM at the address given earlier. After this, the NMWR strobe is deasserted thereby causing the cycle to end.

Transmit

To begin the transmit cycle, the SNIC issues a BREQ and waits for BACK. Since BREQ and BACK lines are tied together, BACK signal is received immediately. Upon reception of this signal, the SNIC drives out the address to the 373's which latch the address with the ADS0 strobe. The address then flows to the onboard memory. NMRD, driven by SNIC, causes the RAM to drive the data out of the given address and into the SNIC. The SNIC then latches the data into the FIFO on the rising edge of NMRD. This high assertion of NMRD signifies the ending of this cycle. From the FIFO, the data is serialized and transmitted onto the network.

BOARD CONFIGURATION

On this SNIC-AT board, there are six jumper blocks as seen in the diagram below. The following pages will explain how to configure these jumpers.



TL/F/10800–3

Physical Layer

TABLE V. Physical Layer Selection

JB1	JB2	JB3	Physical Layer Selected
on	off	off	twisted pair
off	on	off	thin ethernet
off	off	on	thick ethernet

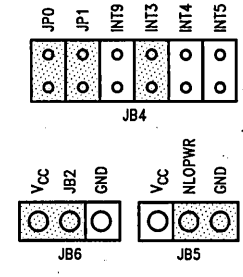
If JB1 is closed while JB2 and JB3 are open, then the twisted pair interface will be selected. If JB2 is closed while JB1 and JB3 are open, then the thin ethernet will be selected. And finally if JB3 is closed while JB1 and JB2 are open, then the thick ethernet will be selected.

Interrupt Lines, Board Addresses, and EPROM Addresses

On JB4, there are six possible connections. Four of these are to select an interrupt line. The available interrupt lines include INT3, INT4, INT5, and INT9. The last two possible connections, JP1 and JP0, are used to select the base address for the board. However, if JB6 is connected to V_{CC}, then these last two connections select the address of the EPROM also. The possible selections and the jumpers which should be on (closed) are shown in Table VI. The factory configuration uses the INT3 line for interrupts and has JP1 and JP0 in the on position. This factory configuration is shown in *Figure 3*, along with the factory configurations for JB5 and JB6.

TABLE VI. Base Address and EPROM Address

JP1	JP0	Base Address	EPROM Address
on	on	300h–31Fh	C800h
on	off	320h–33Fh	CC00h
off	on	340h–35Fh	D000h
off	off	360h–37Fh	D400h



TL/F/10800–4

FIGURE 3. Factory Configuration for JP4, JP5, and JP6

Low Power SNIC Mode

This low power mode is entirely dependent on JB5. The NLOPWR signal is low-asserted, so in most cases this signal will be jumpered to V_{CC}. However, if LAN transmissions are not needed for an extended length of time, the NLOPWR signal may be jumpered to ground. This would turn off the SNI circuitry and conserve power. This feature is primarily provided because the DP83901 enables this function, but can not be practically used in this design.

PAL EQUATIONS

PAL #1 (U1)

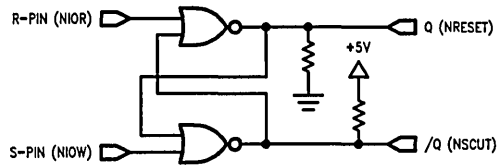
In this first PAL, the output signals are NIO16, NIOEN, NSNICB, and NCSROM. (The N's before the signals indicate that the signal is low asserted). Since it is necessary to assert NIO16 as soon as possible, this first PAL has been selected to be a 10 ns "D" PAL. The NIO16 signal must be TRI-STATE® when it is not asserted. Therefore, we use an enable signal (NIOEN) which is equal to the decode for the I/O Ports (310–31F) and NAEN high. (NAEN high signifies that the system DMA does not have control of the bus.) The enable signal (NIOEN) loops back into the PAL to bring NIO16 out of TRI-STATE. The NIO16 signal is set to zero so that whenever it is enabled it will be asserted.

The SNICB signal consists of simple address decodes along with NAEN. The addresses decode to one of four address slots which were earlier mentioned in the board configuration section. The NCSROM is a very simple signal as it consists only of AD14 and NMRD. AD14 comes from the SNIC and selects either the PROM (when low) or the onboard RAM (when high).

```
PAL 1
module iodec;
flag '-rl';
title
date:9/13/89
functions:
SNIC BOARD DECODE, IO16 DECODE, AND CHIP SELECT PROM';
ul device 'pl618';
"input pins:
NEN16, NAEN, SA9          pin 1, 2, 3;
SA8, SA7, SA6             pin 4, 5, 6;
SA5, SA4, SA3             pin 7, 8, 9;
JP0, JP1, NMRD            pin 13, 14, 15;
A14                       pin 16;

"output pins:
NSNICB, NIOEN, NIO16      pin 12, 17, 18;
NCSROM                    pin 19;

"constants
    X = .X.;
    Z = .Z.;

equations
NSNICB = !(!NAEN & SA9 & SA8 & !SA7 & !SA6 & !SA5 & !JP1 & !JP0
          # !NAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & !JP1 & JP0
          # !NAEN & SA9 & SA8 & !SA7 & SA6 & !SA5 & JP1 & !JP0
          # !NAEN & SA9 & SA8 & !SA7 & SA6 & SA5 & JP1 & JP0);
NIOEN = !(!NAEN & SA9 & SA8 & !SA7 & !SA6 & !SA5 & !JP1 & !JP0
         & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & !JP1 & JP0
         & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & SA6 & !SA5 & JP1 & !JP0
         & !NEN16 & SA4 & !SA3
       # !NAEN & SA9 & SA8 & !SA7 & SA6 & SA5 & JP1 & JP0
         & !NEN16 & SA4 & !SA3);
NCSROM = !(!A14 & !NMRD);
enable NIO16 = !NIOEN;
       NIO16 = 0;
end iodec;
```

PAL #2

In this PAL, there are eight outputs which include NRESET, NSOUT, NRDYEN, NIOCHRDY, NCS, NRACK, and NWACK. The first two outputs (NRESET and NSOUT) are part of an R-S flip-flop as shown below:



TL/F/10800–5

FIGURE 4. RS Flip-Flop

NRESET is given by the NOR of the high asserted R-input pin and the NSOUT signal. NSOUT is given by the NOR of the high asserted S-input pin and the NRESET signal. The NOR gates are enabled by the low assertion of NRSTDRV. When the system first boots up, it will disable the NOR gates by asserting the RSTDRV signal. But due to the pull-up and pull-down resistors, the output <NRESET, NSOUT> will be set to <0,1>. Once RSTDRV becomes deasserted, the output will remain at <0,1>. The only way to get out of reset is to assert the S-pin high which is done by an NIOW and an address decode to 318–31F. After the system has booted up, the SNIC may be reset through software. This would be done by setting the R-pin high with an NIOR and an address decode to 318–31F. To escape from reset, we once again set the S-pin high with an NIOW and address decode of 318–31F. The above description of logic is also shown in Truth Table VII.

TABLE VII. R-S Flip-Flop Truth Table

R (NIOR)	S (NIOW)	Q (NRESET)	$\overline{Q}$ (NSOUT)
0	0	0	1
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1
0	1	1	0

By using the NIOR and NIOW which are never asserted at the same time, this insures that the R-pin and the S-pin will never be asserted at the same time. The next two signals (NRDYEN and NIOCHRDY) are quite similar to NIOEN and NIO16 in PAL #1. All of the decode takes place in the enable signal (NRDYEN). This decode consists of addresses 300–30F without NACK or the addresses 310–318 without PRQ. If the NRDYEN signal is asserted, then NIOCHRDY will be driven low. At all other times, the NIOCHRDY strobe will be in TRI-STATE. NCS is decoded by NSNICB (from PAL #1) along with the low assertion of SA4 and either NIOR or NIOW. Its decode is in the address range of 300–30F. The last two signals are NRACK and NWACK. NRACK occurs with an address decode to 310–31F, an NIOR, and a PRQ. The NWACK signal only differs from the NRACK by the NIOR/NIOW signal and therefore consists of an address decode to 310–318, an NIOW, and a PRQ. INT is just sent through the PAL to be buffered. The buffered signal which comes out of the PAL is INTO.

```
PAL 2
module reset;
flag '-rl';
title '
date:9/13/89
functions:
RESET LATCH, SNIC SELECT, IOCHRDY, RACK, WACK,
BUFFER INTERRUPT';
u2 device 'pl6l8';
"input pins:
NSNICB, NIOW, NIOR      pin 1, 2, 3;
RSTDRV, NACK, PRQ       pin 4, 5, 6;
SA4, SA3, INT           pin 7, 8, 9;
"output pins:
INTO, NRACK, NWACK      pin 12, 13, 14;
NRESET, NSOUT, NRDYEN   pin 15, 16, 17;
NIOCHRDY, NCS           pin 18, 19;
"constants
    X = .X.;
    Z = .Z.;
equations
NCS = !(!NSNICB & !NIOR & !SA4 # !NSNICB & !NIOW & !SA4);
NRACK = !(!NSNICB & PRQ & !NIOR & SA4 & !SA3);
NWACK = !(!NSNICB & PRQ & !NIOW & SA4 & !SA3);
NRDYEN = !(!NSNICB & !NIOR & !SA4 & NACK
         # !NSNICB & !NIOW & !SA4 & NACK
         # !NSNICB & !PRQ & !NIOR & SA4 & !SA3
         # !NSNICB & !PRQ & !NIOW & SA4 & !SA3);
enable NIOCHRDY =!NRDYEN;
       NIOCHRDY = 0;
enable NRESET = !RSTDRV;
       NRESET = !(!NSNICB & !NIOR & SA4 & SA3 # NSOUT);
enable NSOUT = RSTDRV;
       NSOUT = !(!NIOW # NRESET);
INTO = INT;
end reset;
```

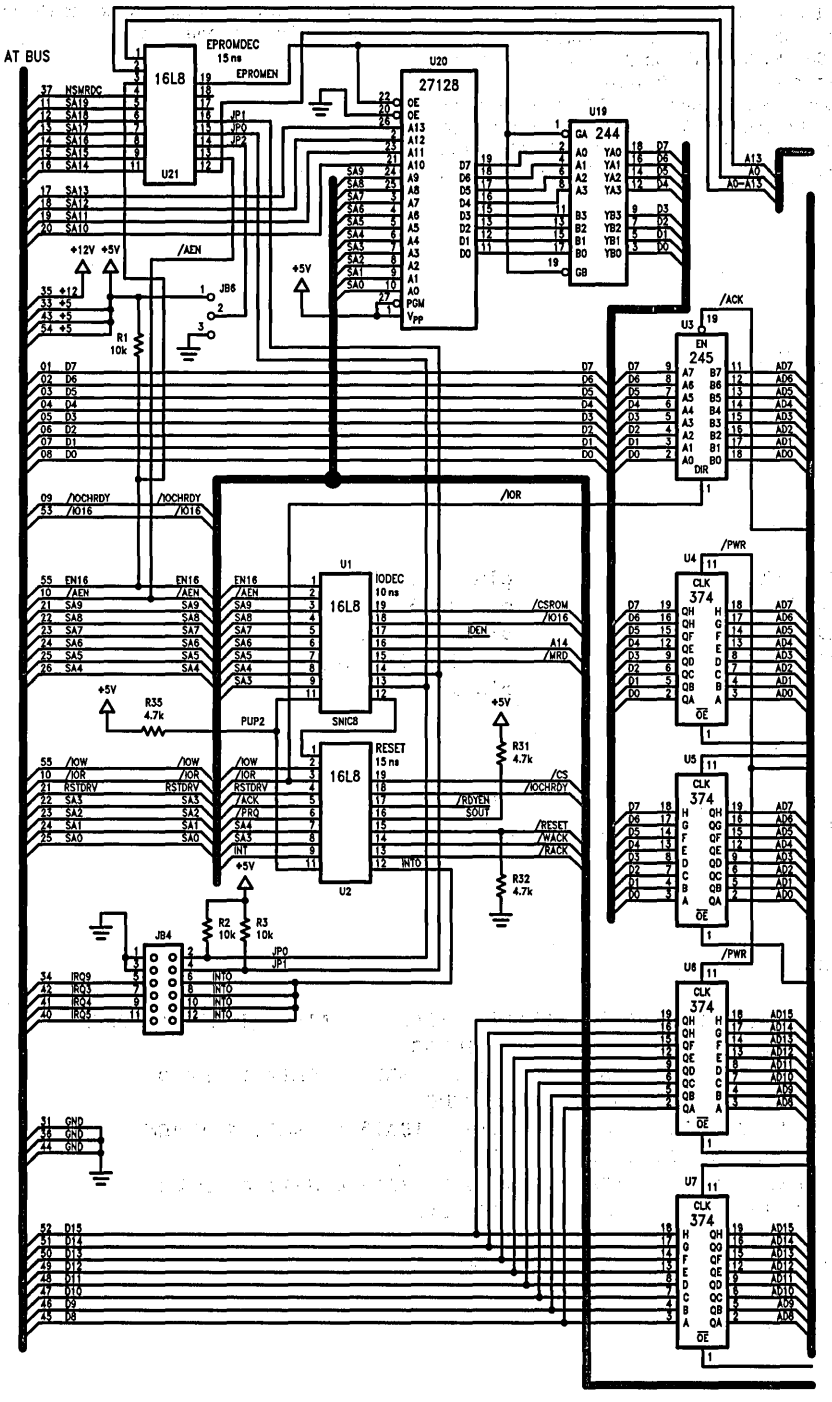
PAL #3

The third PAL only does a decode to enable the optional EPROM. This decode consists of an address decode to C800h, CC00h, D000h, or D400h depending on JP1 and JP0 as shown in the board configuration section. JP2 must also be jumpered for selection of the EPROM. NAEN, a low asserted signal should be high to indicate that the DMA does not have control of the bus and the NSMRDC signal should be asserted low since the CPU is doing a system memory read.

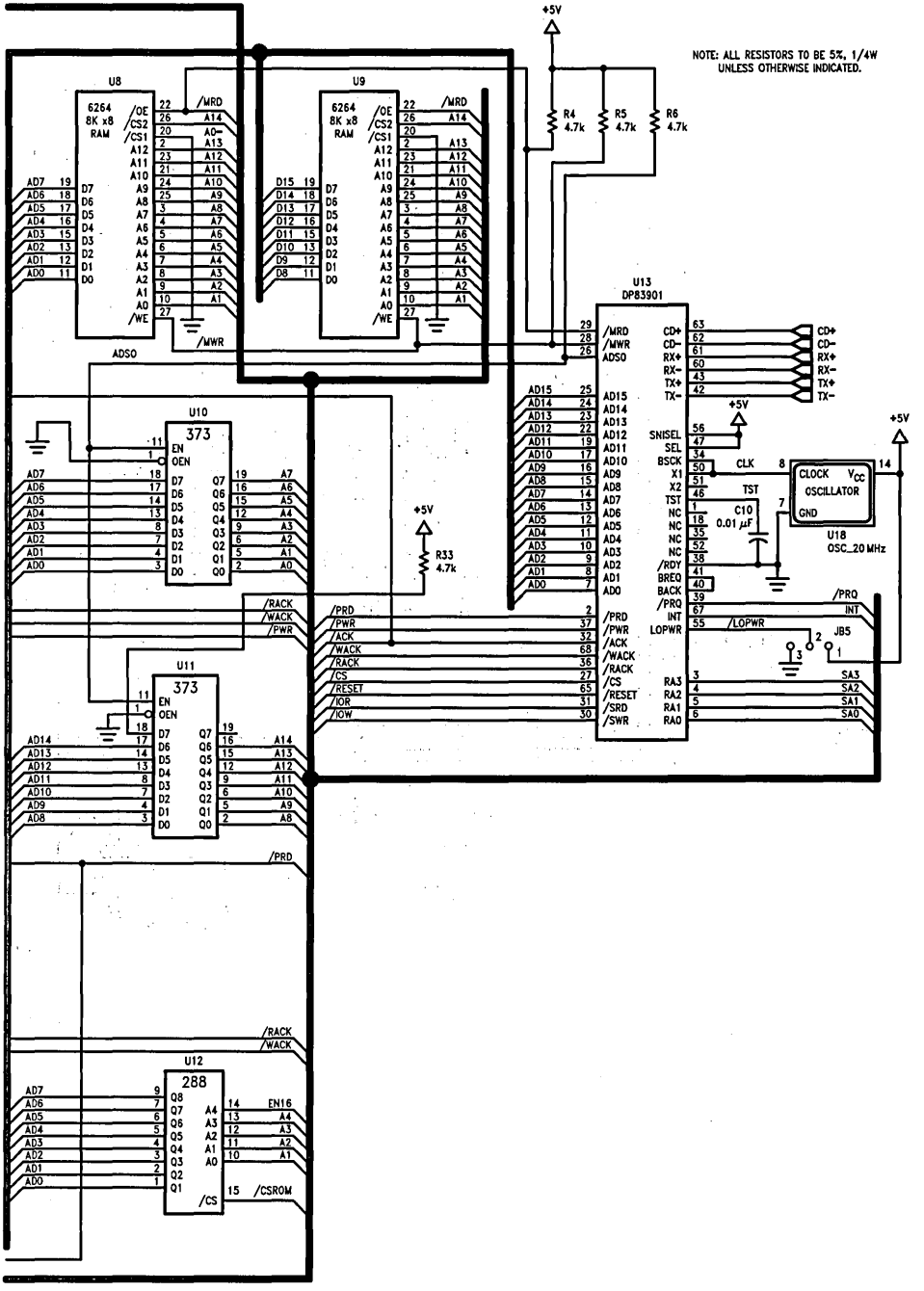
A013 output is used to generate a signal to the lower byte RAM. This signal will route A0 to the RAM when EN16 is high enabling 8-bit operation.

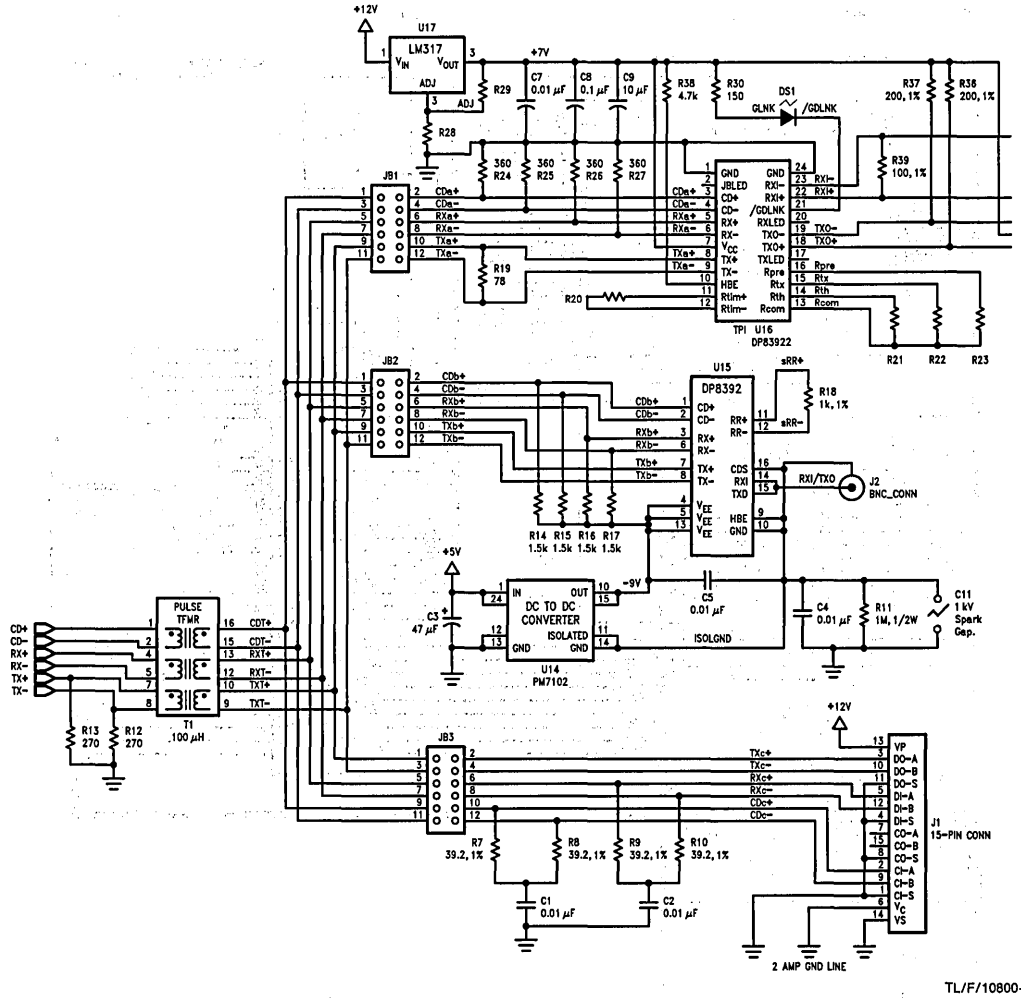
```
PAL 3
module epromdec;
flag '-r0';
title '
date:9/13/90
function:
EPROM DECODE';

u21 device 'p16l8';

"input pins:

A0, A13, EN16                   pin 1, 2, 3;
SMRDC, SA19, SA18               pin 4, 5, 6;
A17, SA16, SA15                 pin 7, 8, 9;
A14, NAEN, JP2                  pin 11, 13, 14;
P0, JP1                         pin 15, 16;

"output pins:

A013                            pin 12;
EPROMEN                         pin 19;

"constants
    X = .X.;

equations

NEPROMEN = !(SA19 & SA18 & !SA17 & !SA16 & SA15 & !SA14 & !NAEN
         & JP2 & !JP1 & !JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & !SA16 & SA15 & SA14 & !NAEN
         & JP2 & !JP1 & JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & SA16 & !SA15 & !SA14 & !NAEN
         & JP2 & JP1 & !JP0 & !NSMRDC
      # SA19 & SA18 & !SA17 & SA16 & !SA15 & SA14 & !NAEN
         & JP2 & JP1 & JP0 & !NSMRDC);

A013 = !(!A0 & EN16 # !A13 & !EN16);

end epromdec;
```

TL/F/10800–10

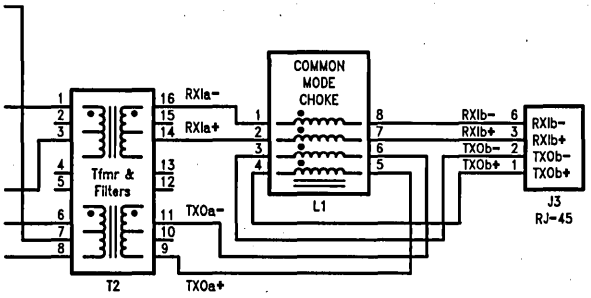

AT BUS
EPROMDEC
15 ns
16L8
U21
EPROMEN
JP1
JP0
JP2
U20
27128
U19
244
+12V
+5V
/AEN
JB6
R1
10k
/ACK
U3
EN
245
DIR
/IOR
/IOCHRDY
/IO16
U1
IODEC
10 ns
/CSROM
IOEN
A14
/MRD
SNIC8
PUP2
R35
4.7k
RESET
15 ns
U2
/CS
/IOCHRDY
/RDYEN
SOUT
/RESET
/WACK
/RACK
R31
4.7k
R32
4.7k
JB4
R2
10k
R3
10k
JP0
JP1
/PWR
U4
U5
U6
U7
CLK
374
OE
GND

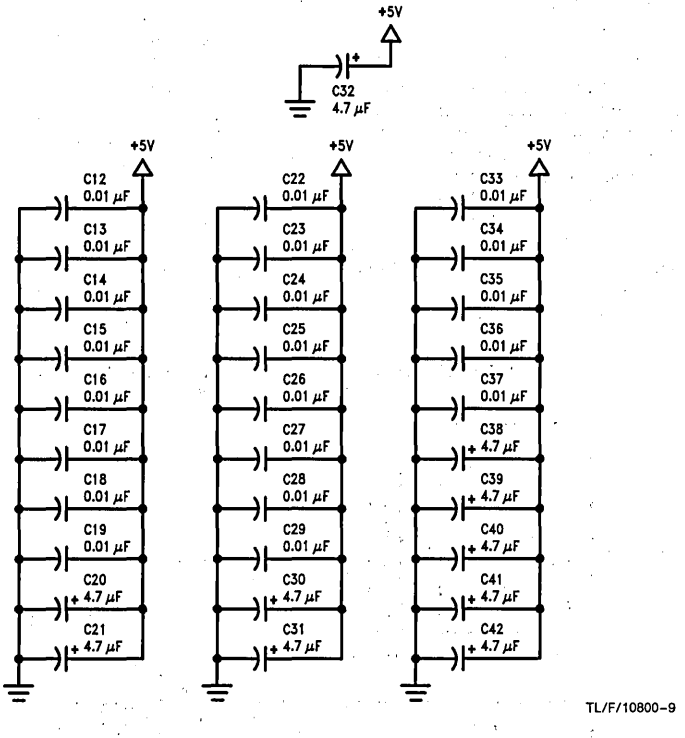
TL/F/10800-6


NOTE: ALL RESISTORS TO BE 5%, 1/4W
UNLESS OTHERWISE INDICATED.
+5V
R4 4.7k
R5 4.7k
R6 4.7k
U8
6264 8K x8 RAM
U9
6264 8K x8 RAM
/MRD
/MWR
/WE
/OE
/CS1
/CS2
ADSO
U10
373
U11
373
U12
288
/CSROM
U13
DP83901
R33 4.7k
C10 0.01 µF
TST
CLK
U18 OSC_20 MHz
CLOCK OSCILLATOR
JB5
/PRD
/PWR
/ACK
/WACK
/RACK
/CS
/RESET
/IOR
/IOW
/SRD
/SWR
/PRQ
/LOPWR
INT
CD+
CD-
RX+
RX-
TX+
TX-
SNISEL
SEL
BSCK
X1
X2
TST
NC
/RDY
BREQ
BACK
LOPWR
RA3
RA2
RA1
RA0
SA3
SA2
SA1
SA0
EN16
TL/F/10800-7


+12V
U17
LM317
VIN VOUT
ADJ
+7V
R29
C7 0.01 µF
C8 0.1 µF
C9 10 µF
R38 4.7k
R30 150
DS1
GLNK /GDLNK
R37 200, 1%
R36 200, 1%
R28
360 R24
360 R25
360 R26
360 R27
R39 100, 1%
JB1
CDa+ CDa− RXa+ RXa− TXa+ TXa−
GND JBLED CD+ CD− RX+ VCC TX+ TX− HBE Rtim+ Rtim−
GND RXI+ RXI− /GDLNK RXLED TXO− TXO+ TXLED Rpre Rtx Rth Rcom
TXO− TXO+
R19 78
R20
TPI U16
DP83922
R21 R22 R23
JB2
CDb+ CDb− RXb+ RXb− TXb+ TXb−
U15
DP8392
CD+ CD− RX+ RX− TX+ TX− VEE VEE VEE
RR+ RR− CDS RXI TXD HBE GND
sRR+ sRR−
R18 1k, 1%
RXI/TXO
J2
BNC_CONN
R14 R15 R16 R17
1.5k 1.5k 1.5k 1.5k
+5V
C3 47 µF
DC TO DC CONVERTER
IN OUT ISOLATED GND GND
U14
PM7102
−9V
C5 0.01 µF
C4 0.01 µF
R11 1M, 1/2W
C11 1 kV Spark Gap
ISOLGND
CD+ CD− RX+ RX− TX+ TX−
PULSE TFMR
CDT+ CDT− RXT+ RXT+ TXT− TXT+
T1
100 µH
R13 270
R12 270
JB3
TXc+ TXc− RXc+ RXc− CDc+ CDc−
R7 39.2, 1%
R8 39.2, 1%
R9 39.2, 1%
R10 39.2, 1%
C1 0.01 µF
C2 0.01 µF
+12V
VP DO-A DO-B DO-S DI-A DI-B DI-S CO-A CO-B CO-S CI-A CI-B CI-S VC VS
J1
15-PIN CONN
2 AMP GND LINE

TL/F/10800–8


COMMON
MODE
CHOKE
Tfmr &
Filters
T2
L1
J3
RJ-45
RXIa−
RXIa+
TXOa−
TXOa+
RXIb−
RXIb+
TXOb−
TXOb+


+5V
C32
4.7 µF
C12
0.01 µF
C13
0.01 µF
C14
0.01 µF
C15
0.01 µF
C16
0.01 µF
C17
0.01 µF
C18
0.01 µF
C19
0.01 µF
C20
4.7 µF
C21
4.7 µF
C22
0.01 µF
C23
0.01 µF
C24
0.01 µF
C25
0.01 µF
C26
0.01 µF
C27
0.01 µF
C28
0.01 µF
C29
0.01 µF
C30
4.7 µF
C31
4.7 µF
C33
0.01 µF
C34
0.01 µF
C35
0.01 µF
C36
0.01 µF
C37
0.01 µF
C38
4.7 µF
C39
4.7 µF
C40
4.7 µF
C41
4.7 µF
C42
4.7 µF

TL/F/10800–9

Low Power Ethernet with the CMOS DP83910 Serial Network Interface

National Semiconductor
Application Note 622
William Harmon

INTRODUCTION

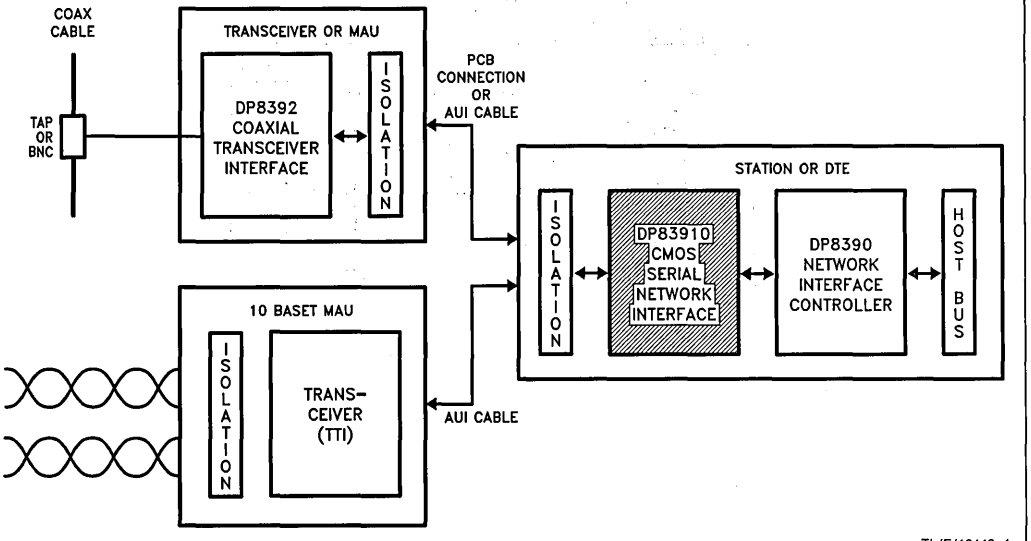
This application note discusses the features of, and implementation techniques for, National Semiconductor's CMOS Serial Network Interface (SNI), the DP83910. Also, a comparison of the CMOS SNI to National's bipolar SNI (DP8391) on several key issues will be provided. In general, the DP83910 provides a low power Attachment Unit Interface (AUI) for a Carrier-Sense Multiple Access with Collision Detect (CSMA/CD) Ethernet system. In fact, when used in conjunction with National Semiconductor's Network Interface Controller (NIC, DP8390) and Coaxial Transceiver Interface (CTI, DP8392), the DP83910 provides for a complete IEEE 802.3 Ethernet and/or thin wire Ethernet solution, as shown in *Figure 1*.

FUNCTIONAL DESCRIPTION OF THE DP83910

The CMOS SNI operates as an interface between an Ethernet transceiver and a local area network data controller. A functional block diagram of the DP83910 is shown in *Figure 2*. The primary function of this interface is to perform the encoding and decoding that is necessary for the differential pair Manchester encoded data of the transceiver and the Non-Return-to-Zero (NRZ) serial data of the NIC to be compatible with each other. In the case of a transmission, the SNI translates the NRZ serial data from a network controller's transmit data line into differential pair Manchester encoded data on a transceiver's transmit pair. In order to perform this operation, the NRZ bit stream is first received by the Manchester encoder block of the SNI. Once the bit stream is encoded, it is transmitted out differentially on to the transmit differential pair through the transmit driver. When a reception takes place, the differential receive data from a transceiver is converted from Manchester encoded data into NRZ serial data and a receive clock, which are passed to the receive data and receive clock inputs of the Network Interface Controller. In executing this sequence, the DP83910's data receiver takes the Manchester data from the differential receive lines and passes it to the phase locked loop (PLL) decoder block. The PLL block then decodes the data and generates a data receive clock and a stream of NRZ serial data, which is presented to the NIC. In the case of National Semiconductor's Network Interface Controller, the DP8390, the serial NRZ signals are called TXD and RXD.

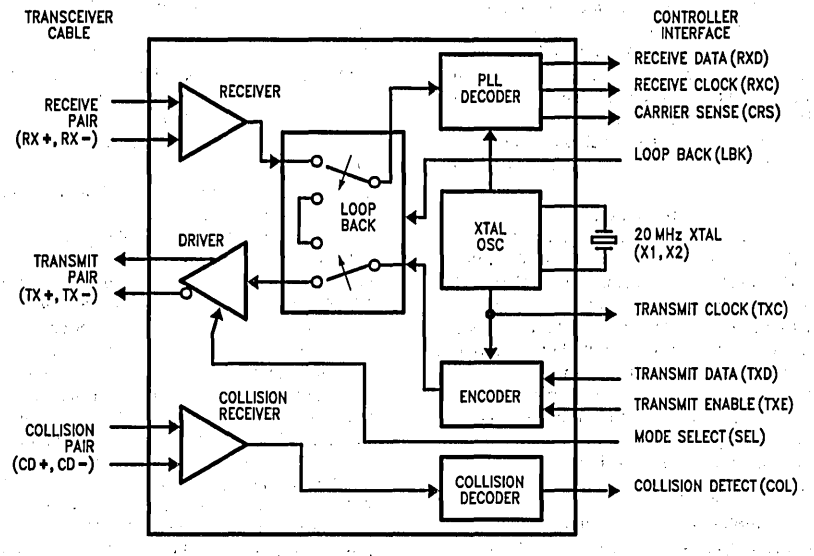
In addition to performing the Manchester encoding and decoding function, the DP83910 also provides several important network signals to the network controller. A diagram of the interface between National Semiconductor's NIC and the CMOS SNI can be found in *Figure 3*. The first of these signals is carrier sense (CRS), which indicates to the controller that data is present on the SNI's receive differential pair. Secondly, the SNI provides the network controller with a collision detection signal (COL), which informs the controller that a collision is taking place somewhere on the net-

IEEE 802.3 Compatible Ethernet/Thin Ethernet/10BaseT Local Area Network Chip Set



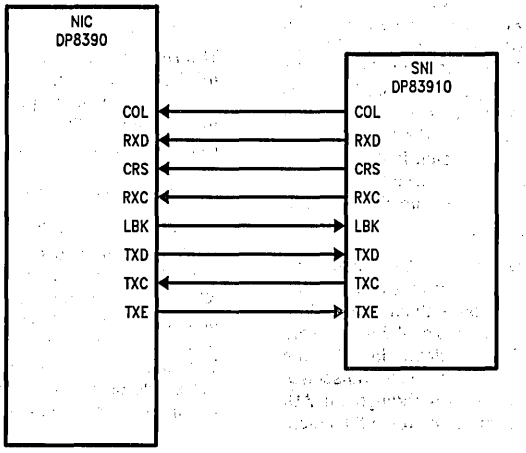
TL/F/10446–1

FIGURE 1. A Block Level Diagram of an Ethernet Node



TL/F/10446–2

FIGURE 2. DP83910 Block Diagram



TL/F/10446–3

FIGURE 3. Interface between the DP8390 and DP83910

work. The SNI itself is informed of the collision when its collision receiver detects a 10 MHz signal on the differential collision input pair. Finally, the DP83910 provides both the receive and transmit clocks (RXC and TXC, respectively). The transmit clock is a divide by two derivative of the SNI's oscillator inputs (X1 and X2), while the receive clock is generated directly from the frequency of the input data to the PLL.

The DP83910 can also be placed in a loopback mode, in order to check the SNI's receive and transmit interface to the network controller. In loopback, as pictured above, the SNI's Manchester encoder block is essentially connected directly to the PLL decoder block. This allows for the validation of the Manchester encoding and decoding process without the variable of random network traffic. The SNI is placed in loopback mode when the loopback pin (LBK) is driven high.

COMPARING THE DP83910 WITH THE DP8391

The DP83910 is basically a CMOS version of the existing National Semiconductor bipolar SNI, the DP8391. The functionality of the two parts is identical. However, there are a few differences that exist between the two parts, in spite of the fact that they can be implemented as pin for pin compatible. The most fundamental difference between the two parts is the process under which each is manufactured. The DP83910 SNI is fabricated in a CMOS process, while the DP8391 is made in a bipolar process. As a result of this, the level of average power supply current needed by the DP83910 is approximately 75 percent less than the 270 mA required by the DP8391. Another significant difference between the two parts is the CMOS SNI's need for a pulse transformer to be placed between all of its differential signals and those of the transceiver, regardless of whether a drop cable or thin wire Ethernet configuration is being imple-

mented. This is necessary due to the fact that the CMOS process will not guarantee the IEEE 802.3 16V fail safe specification if no isolation is provided to the differential signals that go to the AUI cable. One consequence of the transformer requirement is that National Semiconductor defines the AUI interface at the transceiver side of the transformer and only guarantees the correct operation of the CMOS SNI when the pulse transformer is employed in the system.

In addition to the above process related differences, there are still two non-process related differences, which need to be mentioned. First, the phase locked loop in the bipolar SNI is digital, while the phase locked loop of the CMOS SNI is analog. This is functionally transparent when designing with the DP83910; however, it does provide for a significant savings in power consumption. Finally, it should be noted that pin 17 (TEST) on the bipolar SNI is required to be tied to ground through a capacitor, while the same pin on the CMOS SNI can either be implemented in the same manner or connected directly to ground. A list of all the above mentioned differences can be found in Table I.

DESIGNING WITH THE DP83910

In developing the DP83910, National Semiconductor performed extensive testing in its own Local Area Network Laboratory to assure that the CMOS SNI would provide an easily implemented low power controller/transceiver interface for Ethernet system designers. This development and testing assured that the DP83910 was IEEE 802.3 and Ethernet compatible, able to interface with industry standard transceivers (Ethernet, Twisted Pair Ethernet, and Fiber Optic Ethernet), and is capable of having the National Semiconductor DP8391 as a pin-for-pin replacement. In *Figures 4* and *5*, two methods of implementing the DP83910 with the DP8392 are demonstrated. One significant feature of both designs is that it is possible to directly substitute a DP8391 for the CMOS SNI and maintain the same functional quality.

The DP83910 Transmitter Operation

When operating as a transmitter, the DP83910 combines NRZ data received from the controller with a clock signal, which the SNI generates, and encodes them into a Manchester serial bit stream. This encoded signal then appears differentially at the SNI's TX± output. In Ethernet (10Base5) applications, this signal is sent to the transceiver or the Medium Attachment Unit (MAU) through an AUI transceiver cable. This cable, which can be up to 50 meters in length, typically consists of four individually shielded twisted wire pairs (TX±, RX±, CD±, and PWR/GND), which are covered by an additional overall shield. The transmit signal pair, which has a differential characteristic impedance of 78Ω, should be terminated at the receiving end of the cable. It should be noted that each of the TX+ and TX− source follower outputs needs to be connected to ground through a 270Ω pull down resistor.

When employing the CMOS SNI, it is important to place a pulse transformer between the differential transmit pair on the DP83910 and the differential transmit signal on the AUI cable or CTI, as shown in *Figures 4* and *5*. This transformer is required in order to provide the necessary isolation for the CMOS SNI to meet the IEEE 802.3 16V fail safe specification. However, the pulse transformer does reduce the transmission of noise onto the transceiver cable. Also, it should be noted that more inductive transformers will decrease the magnitude of the undershoot. Furthermore, it is imperative that the designer guarantee the inductive load seen between the DP83910's AUI interface and the CTI receiver be greater than 27 μH. Transformers with 50 μH to 150 μH loading, such as the Pulse Engineering PE64103 and Nano Pulse NP5417, are recommended, since they will minimize the inductive undershoot on the SNI's TX± output pair and reduce the noise seen by the CTI's differential transmit input pair. It is important that the selected pulse transformer doesn't excessively increase the rise and fall time nor lower the output amplitude despite the fact that it reduces the undershoot.

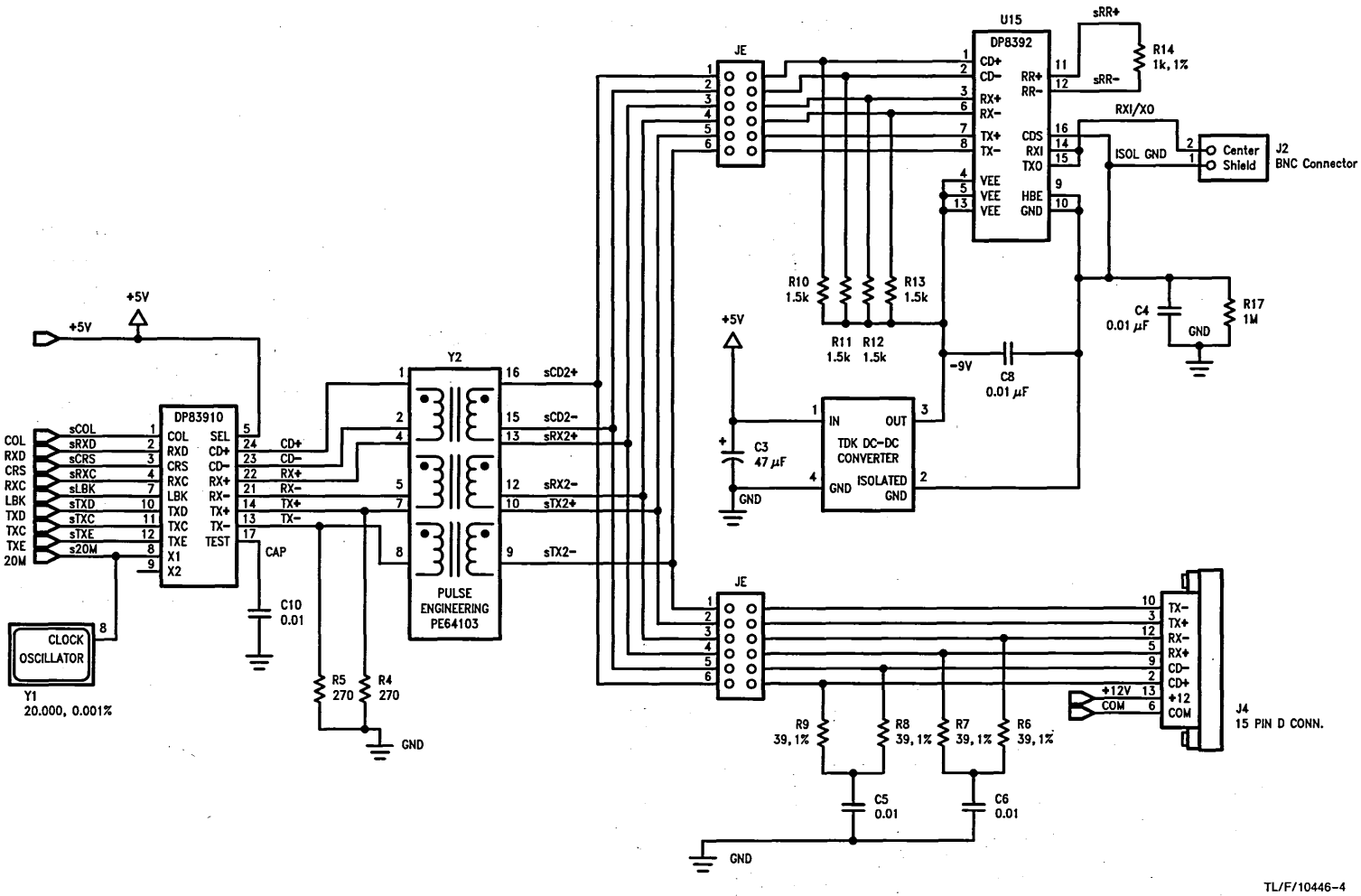
The DP83910 provides both half and full step modes. The IEEE 802.3 standard requires the use of half step mode, in which the transmit output goes to differential zero in idle. In full step mode, the transmitter enters idle and stays at a fixed level. This will eventually allow the pulse transformer to completely saturate. The desired mode of operation is chosen through the Mode Select pin (SEL) on the SNI.

The DP83910 Data Receiver Operation

While performing reception, the CMOS SNI receives differential Manchester encoded serial data and converts it into NRZ serial data and a receive clock. The Manchester encoded data, which is received from the CTI or AUI cable, must be isolated before it reaches the SNI. Hence, the DP83910 requires that there be a pulse transformer on the SNI's side of the AUI interface. The actual employment of this transformer can be seen in both *Figures 4* and *5*. This

TABLE I. Comparison of the DP8391 and DP83910

	DP8391	DP83910
Process	Bipolar	CMOS
Power Consumption (Typical)	270 mA	70 mA
Pulse Transformer (At DTE Side of AUI Interface)	Optional	Required
Phase Locked Loop	Digital	Analog
Pin 17	PLL Filter/Capacitor Required	Test Pin/Capacitor Optional



TL/F/10446-4

FIGURE 4. Interface for Ethernet and Thin Wire Ethernet

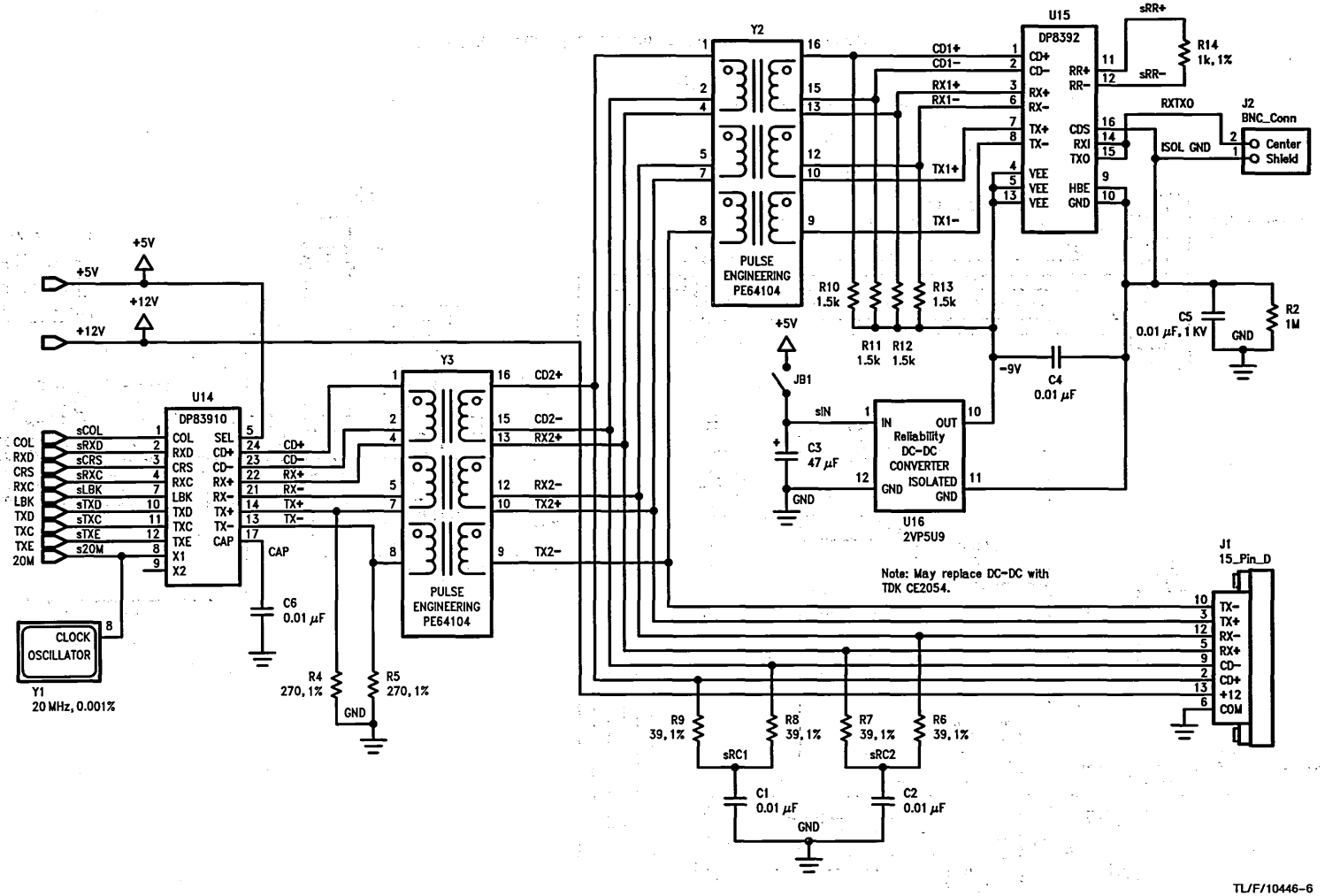


FIGURE 5. Single Switch Solution for Ethernet and Thin Wire Ethernet

transformer is mandatory and it forms part of the internal DC biasing circuit used for the differential receivers. Furthermore, the transformer is also needed to isolate the transceiver cable against the 16V voltage fault specification in the IEEE 802.3 standard. The performance of the differential receiver is not greatly affected by the selection of a pulse transformer. As a result, the pulse transformer selected for the transmitter design will also work correctly for the RX± data receiver. It should be noted here that the collision receiver is very similar to the data receiver and requires the same isolation. The collision input will be discussed more in the following section.

Once the data arrives at the receiver inputs of the SNI, it is amplified and then decoded by the analog phase locked loop, which can receive Manchester data with ±20 ns of random jitter. During the decoding process, the incoming signal is converted into NRZ data and a receive clock, which are sent to a network controller. Also, the differential data receiver has a built in filter to provide a static noise margin. This filter enables the SNI to reject signals that do not exceed the input squelch voltage and have less than a 30 ns pulse width.

Furthermore, since the DP83910 and pulse transformer constitute the AUI interface, the physical connection between the AUI and the MAU interfaces is defined as being on the MAU side of the pulse transformer. In light of this, it is permissible, when incorporating the CMOS SNI in a thin wire Ethernet application, to have a 78Ω resistance appear across the differential receive and collision inputs to the CTI, as shown in *Figure 5*.

The DP83910 Collision Pair Operation

In addition to the data receiver, the DP83910 also provides a differential receiver for the collision pair, which is driven by the transceiver. This 10 MHz active signal, from the AUI Interface, is converted to a TTL signal, digitally stretched, and sent to the controller as the Collision Detect Output (COL). Just as with the data receiver, the differential collision receiver has a built in filter that rejects pulses that do not exceed the input squelch voltage level and have a pulse width less than 30 ns.

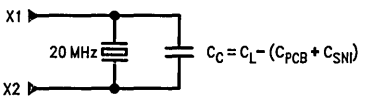
Optimal Ethernet and Thin Wire Ethernet Interface

If it is necessary to design a LAN board that minimizes the number of switching devices (jumpers) to alternate between Ethernet and thin wire Ethernet, the solution in *Figure 5* could be employed. This solution, in contrast to the six jumper solution in *Figure 4*, requires only one switch, which enables and disables the power supply to the CTI. In the case of thin wire Ethernet, power would be supplied to the CTI, while during drop cable Ethernet operation the unused CTI would be powered down. Hence, no excessive power is required when thin wire Ethernet is not in use. Furthermore, since there is only one switch, it may be feasible to implement that switch with a transistor as opposed to a jumper. The advantage to using a transistor is that the Ethernet/thin wire Ethernet option can now be made to be software selectable. This is accomplished by developing a control signal, which the software can issue to switch the transistor. Also, in looking at *Figure 5*, it is seen that two pulse transformers are used. The first transformer (Y3) is required by the CMOS SNI, for the reasons previously mentioned. The second pulse transformer (Y2), however, is used to isolate the powered-down CTI from the AUI cable interface, when Ethernet is being used. As in *Figure 4*, the application in *Figure 5* allows the direct substitution of a bipolar SNI, the DP8391, for the CMOS SNI.

The DP83910 Oscillator Inputs

The oscillator inputs of the CMOS SNI can be driven with a crystal or an oscillator. In either case, the SNI oscillator must be driven with a 20 MHz signal that provides for the transmitted frequency to be accurate within 0.01% as specified in IEEE 802.3 standard. When using an oscillator, the output of the oscillator should be tied to input X1 of the SNI and the X2 input of the SNI should be left unconnected or grounded. However, the employment of a crystal to generate the 20 MHz signal at the SNI's oscillator inputs requires a great deal of care. The frequency of the crystal is usually measured with a fixed load capacitance (C_L, typically 20 pF), which is specified in the crystal's data sheet. In order to prevent any distortion in the transmitted frequency, the total capacitance across the crystal's leads should equal its specified load capacitance. The capacitance that is seen by the crystal's leads is the sum of the stray PC board capacitance (C_{PCB}) and the capacitance looking into the X1 and X2 inputs (C_{SNI}). If this capacitance is smaller than the crystal's load capacitance, a correctional capacitance (C_C) can be placed across the crystal's leads. This correctional capacitance would equal the difference between the crystal's load capacitance and the sum of the stray PC board capacitance and the SNI's X1 and X2 input capacitance. It should be noted that the input capacitance of the SNI that is seen across X1 and X2 is approximately a negligible 0.5 pF. *Figure 6* displays a possible crystal setup. The selected crystal should meet the following specifications:

Resonant frequency	20 MHz
Tolerance	±0.001% at 25°C
Stability	±0.005% at 0°C–70°C
Type	AT cut
Circuit	Parallel Resonance

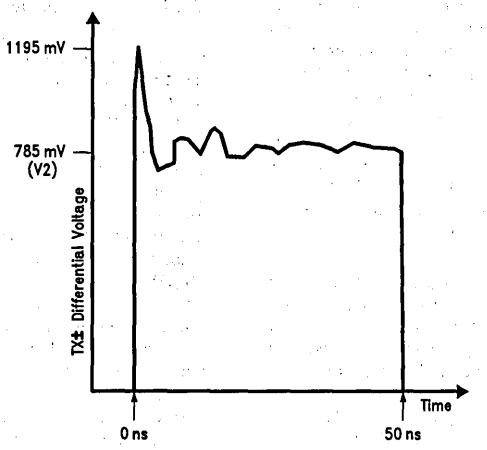


TL/F/10446–5

FIGURE 6. SNI Oscillator Input Circuit

Improving Transmitter Overshoot

Upon transitioning from a differential voltage of one polarity to another polarity (i.e., positive to negative), the magnitude of the differential transmit signal will reach a peak value. This peak at the transition points in the differential transmit waveform is referred to as the overshoot voltage. The overshoot voltage of the DP83910 is below the maximum allowable 1315 mV value that appears in the IEEE 802.3 standard. However, the IEEE standard also defines the overshoot voltage to be no greater than 1.12 times the nominal value (IEEE calls this nominal value V2). The DP83910 exceeds this particular segment of the overshoot specification, as shown in *Figure 7*. However, exceeding the allowable overshoot voltage value, as the CMOS SNI does, will have no functional affect on a system. Furthermore, the overshoot voltage can be altered to adhere to the IEEE 802.3 specification by placing a capacitor across the differential transmit pair at the primary (SNI side) of the required pulse transformer. This capacitor should be in the range of 40 pF to 50 pF and will not degrade the performance of the CMOS SNI or system in any way. It should also be mentioned that the DP8391, the bipolar SNI, will still be a pin-for-pin replacement for the CMOS SNI, in a design which employs the capacitor for improving the overshoot.



TL/F/10446–7

FIGURE 7. TX± Differential Overshoot Voltage

Ethernet Network Interface Adapter for the Apple Macintosh II NuBus

National Semiconductor
Application Note 686
Andrew C. Pagnon

INTRODUCTION

The gradual move in recent years towards distributed processing units with a need for these to communicate and the increasing demand for peripheral usage optimization has resulted in the fast growth of Local Area Networks. In an attempt to standardize communications between networks the International Standards Organization (ISO) has proposed a seven layer reference model called the Open System Interconnect (OSI) which provides an independent framework for all the emerging network standards. The IEEE has defined a number of these standards (802.3 to 802.6) covering the two lower layer functions, physical and data link layers.

National Semiconductor provides a three chip set which supports the Ethernet/Thin-wire Ethernet standard (a subset of IEEE802.3) the DP8390 Network Interface Controller (NIC), DP83910 Serial Network Interface (SNI), and DP8392 Coaxial Transceiver Interface (CTI).

The aim of this application note is to describe the implementation of a Macintosh II, IIx and IIcx to Ethernet/Thin-wire Ethernet interface solution using the NSC chip set. This solution takes the form of a network interface adapter card which on one side plugs into any of the six Macintosh II NuBus expansion slots and on the other supports two physical layer options, Ethernet and thin-wire Ethernet.

The board easily interfaces to the Macintosh II NuBus interface with few external components. This application note assumes the reader is familiar with NSC's Ethernet chip set and the Macintosh II NuBus.

The note begins with a hardware overview of the adapter card, and a background description of the NuBus interface. This is followed by a detailed description of hardware supported by the main sequencer/arbitrator state diagram. This covers arbitration and a detailed description of all the cycle types implemented on the card. The PAL equations and part list are included at the end of the note along with a detailed schematic and timing diagrams.

HARDWARE OVERVIEW

The main function of this adapter card is to transfer Ethernet packet data to/from the Macintosh CPU via NuBus during LAN transmissions and receptions. The card supports a NuBus interface to the CPU and an Ethernet interface to the network. Data transfers between the interfaces are routed on the card's local bus through 8k words of shared buffer memory which temporarily stores ethernet packet data, thus decoupling data transfers across the two interfaces. The 8k buffer memory can be expanded to 32k by simply replacing the memory ICs.

Figure 1 shows a simplified block diagram of the adapter. Besides the basic DP8390 chip set this diagram illustrates the connection of the slot and cycle decode logic used to select the card, and generate read/write cycles. The arbiter controls whether the NIC or NuBus can access the buffer RAM. The RAM contains the transmit/received packet data, and the ROMs (actually one chip) contain the Ethernet Address and the Macintosh configuration information. The address bus interface latches store the NuBus address from the multiplexed address/data bus, and the data bus interface consists of buffers and latches to assemble the 16-bit RAM buffer data into a 32-bit word for the NuBus.

Transmission/Reception

For Ethernet transmissions the host CPU writes data into the transmit area of the adapter card buffer memory over the NuBus interface. The host CPU then sets up the NIC to transmit the data by writing to its internal registers. The NIC responds by fetching the data into its internal FIFO using its local DMA channel, from where it is sent to the SNI-CTI and onto the Ethernet cable. Once the data has been transmitted the NIC issues an interrupt back to the host CPU and sets a status bit in its internal register.

For Ethernet receptions data is loaded from the Ethernet cable into the internal FIFO of the NIC from the SNI and CTI. When a programmable threshold is reached in the FIFO, the NIC transfers the data into the receive area of the adapter card buffer memory using its local DMA channel. Once a complete packet has been loaded into memory, the NIC sets up a pointer in its internal register, issues an interrupt to the host CPU and sets a status bit in its internal register. The host processor responds by reading the packet from the adapter card memory over the NuBus interface and updating the packet pointers stored in the NICs internal register.

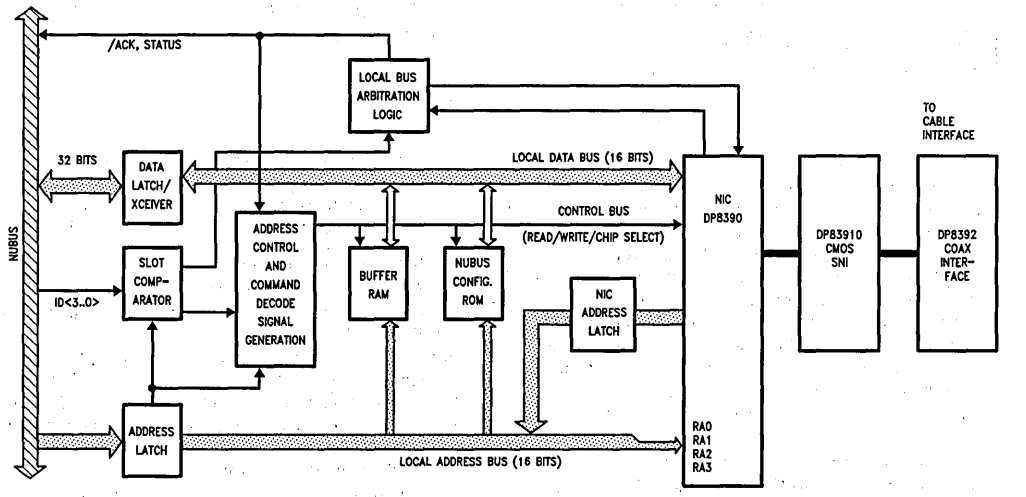
General Adapter Architecture Considerations

A shared memory architecture has been chosen for this design to maximize data throughput while not adding any extra cost or intelligence on the card. The buffer memory is mapped into the NuBus address space and a NuBus slave interface plus local bus arbitration logic is implemented on the adapter card. The reasoning for this decision is given below.

The DP8390 efficiently supports an input/output port architecture, in which the adapter card makes use of the NICs Remote DMA facility to transfer network data between the buffer RAM and an input/output port interfacing to the NuBus and to the host CPU. This implementation is a slightly less expensive option than others however the throughput of the port interface is somewhat limiting, and there are no memory addressing limitations on the NuBus that would require and I/O port technique.

A bus master architecture, in which the adapter card can gain ownership of the host CPU bus and transfer data directly into system memory is significantly more costly and with the current generation of controllers will not yield significantly better performance across NuBus without going to the expense of adding an on-card processor.

Thus, using a buffer RAM that is addressed directly by the NuBus and the Ethernet Controller, provides the flexibility of reading/writing data via the NuBus at fast speeds, and since the DP8390's local DMA only utilizes a small percentage of the RAM's total access time (12%) the RAM is mostly free for NuBus activity.



TL/F/10805–1

FIGURE 1. General Block Diagram for the NuBus Shared RAM Adapter

Once the Shared Adapter RAM approach is chosen another architectural consideration is the width of the Buffer RAM and of the CPU bus transfers supported. Table I shows the options considered. The 16-bit Buffer memory and 32-bit NuBus transfer option was chosen as the best compromise on data throughput and component count/cost.

General Hardware Overview

Figure 2 shows a more detailed block diagram of the Adapter. This shows each block, and details the chips used to implement each block.

The Ethernet to buffer memory interface is implemented using National Semiconductor chip set. The DP8390 Network Interface Controller is a CMOS VLSI device designed to ease interfacing with IEEE 802.3 Ethernet type Local Area Networks. It implements all Media Access Control MAC functions (a subset of the ISO data link layer) for transmission and reception of packets in accordance with the IEEE 802.3 standard. Its dual DMA channels and internal FIFO provide a simple yet efficient packet management design. All bus arbitration and memory support logic required by its dual DMA channels are integrated into the NIC.

The DP83910 Serial Network Interface is a CMOS combined analog and digital device which provides the Manchester encoding and decoding functions of IEEE 802.3 Ethernet type Local Area Networks. It contains ECL like balanced drivers and receivers, collision signal translator and a diagnostic loopback circuit.

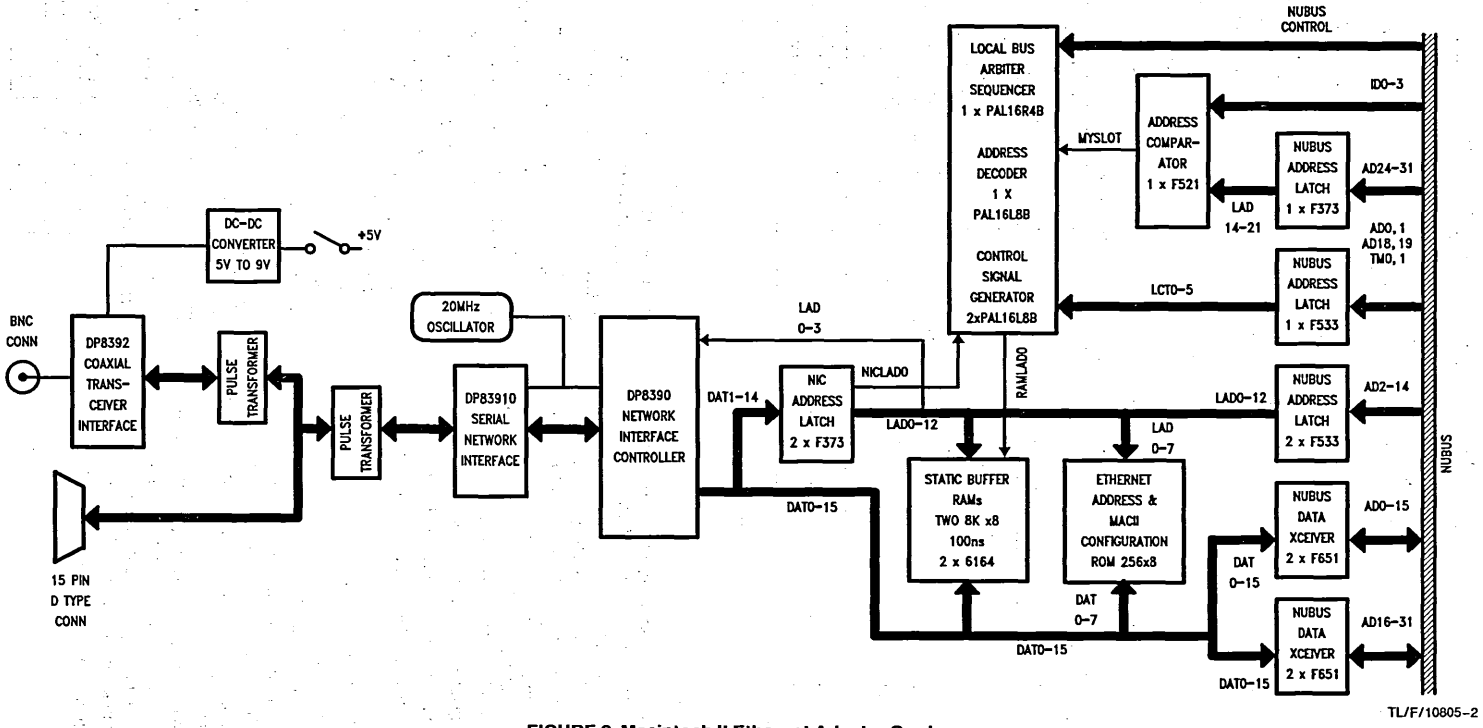
The DP8392 Coaxial Transceiver Interface is a bipolar device used as a coaxial cable line driver/receiver for IEEE 802.3 Ethernet Local Area Networks. In Ethernet applications the transceiver is usually mounted within a dedicated enclosure (Media Access Unit) and connected to the SNI via a drop cable, while for Thin-wire Ethernet (low cost version of Ethernet) the CTI is mounted on the same board as the SNI. Signal and power isolation requirements are met by placing a set of pulse transformers between the SNI and the CTI, and using a DC to DC converter to provide the CTI's −9V supply.

The adapter card supports a 32-bit NuBus interface to the host CPU, implemented using synchronous sequencer logic in the registered PAL 16R4. This interface supports transfers to the Network Interface Controller registers, the "Ethernet address/Mac configuration" ROM and the buffer memory. The two card interfaces must request use of the local bus before they can initiate a transfer to any of the on card devices. These requests are processed by arbitration logic which gives priority to the Ethernet interface.

The 256 x 8 ROM (LS471) contains the unique Physical Address assigned to each Etherent board and the Configuration data required on each NuBus board which supplies identifying information about the board. This ROM can also contain device driver data.

The address/data interface to NuBus consists of four F651s data transceivers to transfer 32-bit data from/onto the NuBus, and three F533s and an F373 used to latch the address and the transfer mode signals onto the adapter card. Data on NuBus is inverted and byte swapped, so inverting transceivers and latches are used. An exception to this is the F373 which is a non-inverting version of the F533 used to latch AD24–31 which are then compared with the ID lines of the particular NuBus slot. Also the card performs a hardware byte swap on the NuBus data.

On NuBus transfers, an F521 8-bit comparator aids in the address decode function by determining whether or not the transfer is intended for the adapter card.



TL/F/10805-2

FIGURE 2. Macintosh II Ethernet Adapter Card

TABLE I. Adapter Card NuBus RAM Width Options

NuBus Transfer Width	Buffer Memory Width	NuBus Clock Beats	NuBus Maximum Data Rate	Percent Bus Usage	Extra Devices Required
16 Bits	16 Bits	4	32 Mbits/sec	31%	None
32 Bits	16 Bits	5	53 Mbits/sec	19%	2-Transceivers 1-PAL16L8
32 Bits	32 Bits	4	64 Mbits/sec	15.5%	2-Transceivers 2-RAM 8k x 8
32 Bits	32 Bits	3	80 Mbits/sec	12.5%	2-Transceivers 2-RAM 8k x 8 (80 ns)

TABLE II. Adapter Card Cycle Type Decoding

AD0	AD1	TM0	TM1	AD18	AD19	Cycle Type
X	X	X	L	H	L	RAM Read
X	L	L	H	H	L	RAM Write Byte 0, 1
L	L	H	H	H	L	RAM Write (Byte 0)
H	L	H	H	H	L	RAM Write (Byte 1)
X	H	X	H	H	L	No Action
X	X	X	L	H	H	ROM Read
X	X	X	H	H	H	Bus Error (ROM Write)
X	X	X	X	L	L	Bus Error (No Device)
X	X	H	X	L	H	NIC Write
X	X	L	X	L	H	NIC Read

NUBUS BACKGROUND

This section describes the NuBus implemented in the Macintosh II expansion slots, and the Ethernet adapter card implementation of its interface. It covers NuBus' main features and signals as used by the card, followed by a description of the address space and addressing modes and ending with a description of the NuBus interface protocol.

NuBus is the bus chosen by Apple to drive the expansion slots of the Macintosh II. Its main features are:

- 32-bit wide multiplexed address data lines
- Synchronous 10 MHz clock cycle (75% duty cycle)
- Read and Write cycles (Mac II does not support block transfers)
- I/O and interrupts are memory mapped
- Geographical addressing lines

Each slot has its own geographical addressing lines onto the adapter board. This is illustrated in *Figure 3* by the super slot space in which each card has its own 256 Mbytes of memory space. Therefore no board configuration is required. (Described later.)

The Ethernet adapter card only implements a NuBus slave interface and therefore arbitration logic to gain bus mastership has not been implemented. Note also that no parity generating/checking logic has been implemented either.

The Ethernet adapter board uses the following NuBus signals:

- Clock
- Reset
- Card Slot Identification
- Non-Master Request
- Address/Data Signals/AD<31..0>
- Control Signals /TM<1..0>, /START, /ACK(nowledge)

The evaluation board supports single word transfers to the NIC registers, Address/Configuration ROM and the Buffer RAM. During the Start clock of a NuBus cycle the NuBus address and transfer mode lines are decoded on the card as shown in *Figure 3,* and as follows:

- TM<1..0> determine the type of transfer (read/write),
- AD<24..31> determine which NuBus slot is accessed.
- AD<19..18> determine which adapter card device is accessed (NIC, ROM or RAM)
- AD<15..2> are used to access a particular location within the device.
- AD<0..1>&<20..23> are ignored by the address decode

The adapter card responds to all NuBus cycles which address the card by generating an acknowledge signal ACK for one clock period and driving a status code on the transfer mode lines. Only two of the four NuBus defined transfer modes are supported by the card, transfer complete or bus error (see Table III).

TABLE III. NuBus Status Codes

TMO	TM1	Acknowledge
H	H	Transfer Complete
L	H	Bus Error

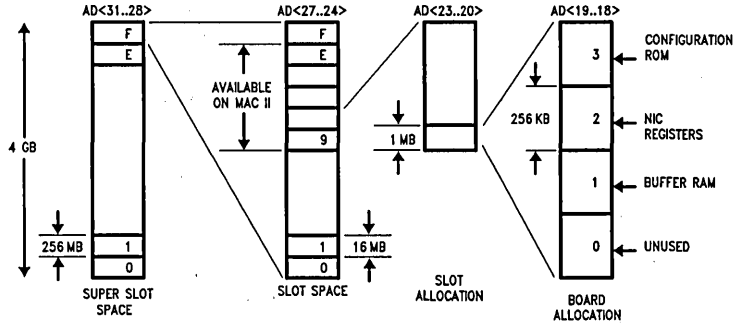


FIGURE 3. NuBus and Adapter Card Address Space Mapping

If the CPU requests a transfer to the adapter card NuBus slot which does not address a device on the card, or requests a write transfer to ROM, the board will respond with an error status encoding of the TM<1..0> lines during the ACK clock of the NuBus cycle. Otherwise a "Bus Transfer Complete" code is returned.

Typical NuBus read and write cycle timings are shown in *Figures 4* and *5*. The first NuBus cycle asserts the START line going low and presenting the address and the transfer mode. A number of clock cycles may follow before the last cycle presents the data and status on the bus, and asserting the ACK signal.

The adapter card does not implement the two other status codes supported by NuBus, "bus time out error", and "try again later". The design of the adapter card ensures that all NuBus cycles will be acknowledged within the NuBus timeout period.

NuBus Address Space

With a 32-bit architecture, the NuBus provides a 4 Gigabytes of address space, *Figure 3*. The 4 Gigabyte space is divided into sixteen 256 Megabyte Super Slots. The Super Slot being accessed is determined by decoding AD<31..28>. The top Super Slot is divided into sixteen Slot spaces by 16 Megabytes each determined by decoding AD<27..24>. Six of these slots ($9 to $E) are implemented as NuBus expansion board connectors on the MAC II. The interface adapter board may be plugged into any of these connectors. No hardware configuration on the adapter card is required.

24/32 Bit Addressing Modes

The adapter card, by ignoring address bits AD<23..20>, supports both 32- and 24-bit addressing modes.

When addressing the card in 24-bit mode, addresses of the form "$sx xxxx" where s is the slot number can be used. The Mac II hardware translates this into a 32-bit address of the form "$Fs0x xxxx".

When addressing the card in 32-bit mode addresses of the form "$Fsxx xxxx" can be used. Note that as the adapter card ignores address bits AD<23..20>, addresses of the form "$Fssx xxxx" access the same adapter card location in both 32- and 24-bit modes, and as Apple have indicated that to ensure compatibility with future versions of the Macintosh designers should not rely on 24-bit mode addressing, it is suggested that addresses of the form "$Fssx xxxx" are always used.

Although supporting 24-bit mode addressing limits the memory range of each slot from 16M to 1M, this sufficiently covers the need of an Ethernet adapter card and simplifies software development. The Macintosh slot manager puts the system in 24-bit addressing mode by default and the memory manager plus some toolbox routines do not currently function properly in 32-bit mode.

Adapter Card Address Space

Once the slot is selected, the Network Interface Adapter's memory space is subdivided into four 256 kbyte blocks of memory determined by decoding AD<19..18>.

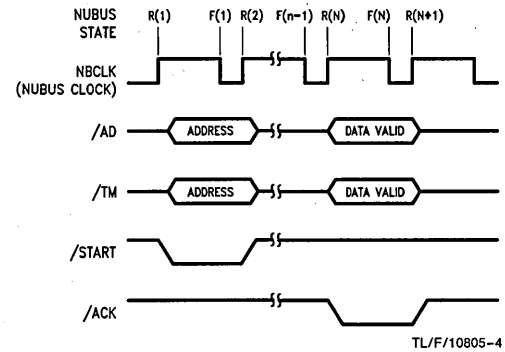


FIGURE 4. NuBus Read Cycle

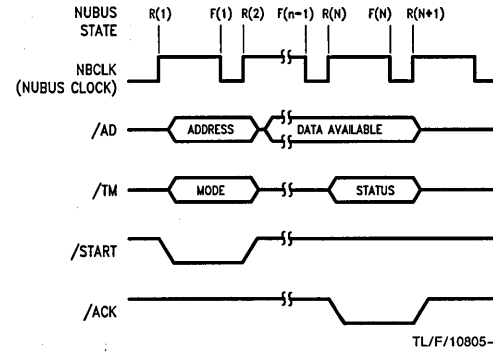


FIGURE 5. NuBus Write Cycle

Figure 3 shows the sub-division of the Super Slot, Slot, and Adapter Board address space.

It is important to note the following points when developing high level software to address the adapter card. If the NIC registers or the ROM are accessed as 8-bit devices, (by declaring a pointer to a character in Macintosh Programmers Workshop (MPW) for example) or as 16-bit devices (by declaring a pointer to a short integer), incrementing these pointers will usually only modify the transfer mode (by incrementing AD<1..0>) rather than increment the address to the device (NIC or RAM). It is therefore recommended that if this form of addressing is used, the RAM, ROM and NIC are declared as 32-bit devices (by accessing them with a pointer to an integer). This will ensure only word transfers take place and each NuBus address increment will also increment the address to the onboard device.

NuBus Timing Diagrams

The NuBus clock has a 100 ns period (10 MHz) with a 75% duty cycle (75 ns "high" and 25 ns "low"). NuBus signals are driven at the rising edge and sampled at the falling edge of the clock. A transfer (read/write cycle) is initiated when the master asserts /START, drives the address on AD<31..0>, and drives the transfer mode signals TM<1..0> with the appropriate code to indicate the desired transfer. A transfer is completed when the slave responds by asserting ACK and driving the transfer mode signals with the appropriate status code. Please refer to *Figure 4* and *Figure 5* for the NuBus read/write cycles.

For a read operation, once the master has acquired the bus, a read bus transaction involves the following steps:

R(1): The bus master asserts/START and the appropriate /ADx and /TMx lines to initate the transfer.

F(1): The bus slaves sample the /ADx and /TMx lines.

R(2): The bus master releases the /ADx, /TMx, and /START lines and waits for /ACK.

R(N): The bus slave places the requested data onto the /ADx lines, asserts /ACK, and places the appropriate status code on /TM0 and TM1 lines. (Note N may be from 2 to 256)

F(N): The bus master samples the /ADx, /ACK, and TMx lines to receive the data and note and error condition.

R(N+1): The bus slave releases the /ADx and /ACK lines and the /TMx lines. This may be the R(1) transition of the next transaction.

For a write operation, once the master has acquired the bus, a write bus transaction involves the following steps:

R(1): The bus master assert/START and the appropriate /ADx and /TMx lines to initiate the transfer.

F(1): The bus slaves sample the /ADx and /TMx lines.

R(2): The bus master places the data on the /ADx lines, releases /TMx, and /START lines and waits for /ACK.

F(2)–F(N): The bus slave samples the /ADx lines to capture the data. The data may be sampled before or during the assertion of /ACK.

R(N): The bus slave asserts /ACK, and places the appropriate status code on /TM0 and TM1 lines when the data is accepted. (Note N may be from 2 to 256)

F(N): The bus master samples the /ACK and TMx lines to determine the end of a transaction.

R(N+1): The bus master releases the /ADx while the bus slave releases the /ACK lines and the /TMx lines.

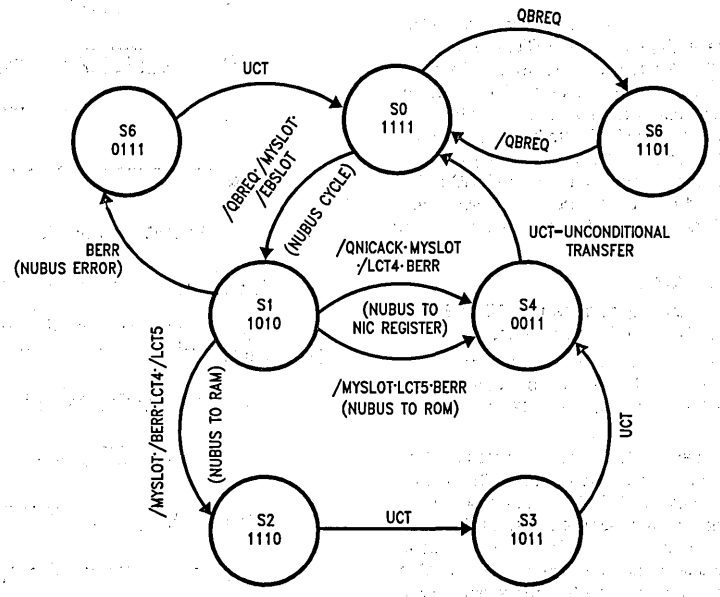


FIGURE 6. State Diagram Sequencer

TABLE IV. Adapter Card Cycle State Sequence for Various Cycle Types and Corresponding Diagrams

Cycle Type	State Sequence	Functional Diagram	Timing Diagram
NuBus to RAM Read	S0 → S1 → S2 → S3 → S4 → S0	*Figure 12*	*Figure A-4*
NuBus to RAM Write	S0 → S1 → S2 → S3 → S4 → S0	*Figure 12*	*Figure A-5*
NuBus to ROM Read	S0 → S1 → S4 → S0	*Figure 8*	*Figure A-1*
NuBus to NIC Read	S0 → S1 → S4 → S0	*Figure 10*	*Figure A-2*
NuBus to NIC Write	S0 → S1 → S4 → S0	*Figure 10*	*Figure A-3*
NuBus Bus Error	S0 → S1 → S5 → S0	*Figure 8*	
NIC to RAM Read/Write	S0 → S6 → S0	*Figure 14*	

DETAILED HARDWARE DESCRIPTION

The card's main function is to transfer Ethernet packet data from the NuBus interface to the Ethernet cable during packet transmission, and from the Ethernet cable to the NuBus interface during packet reception, via an 8k x 16 Buffer RAM, expandable to 32k x 16. In addition to this the NuBus interface is allowed direct read and write access to the NICs registers to control and monitor the NIC's operations, and read access to the "Ethernet address/Mac configuration" ROM.

This transfer of packet data from Ethernet to host CPU is executed in two distinct stages, transfers between host CPU and buffer Memory, and transfers between buffer memory and Ethernet. The former is performed by the on-card NuBus slave interface whereas the latter is performed by the NIC chip set.

A synchronous sequencer/arbiter implemented in a PAL16R4 running on the 10 MHz NuBus clock controls all transfers supported by the adapter card. The state diagram for its operation is shown in *Figure 6.* States S1 to S5 support the NuBus slave interface, and state S6 supports the NICs interface. State S0 is the idle state. The sequence of states for each bus cycle type is shown in Table IV.

Arbitration for Local Card Bus by NIC/NuBus

All addressable devices on the card (Buffer RAM, NIC registers and ROM) share the common non-multiplexed local address and data buses. The two potential masters of this bus, the NuBus interface and the NIC, request access to the bus. Arbitration logic and the state sequencer resolve these requests. The sequencer only responds to master's requests during the idle state (S0). Therefore cycles already in progress are always allowed to complete before the bus is reallocated. Cycles always complete by returning to the idle state (S0). This prevents bus contention on the common local address bus at switchover time (NuBus/NIC) and enables arbitration to take place after every NuBus cycle or NIC local DMA burst. The NIC is given priority over the NuBus interface, so that if a NIC and a NuBus interface request are active when the sequencer is idle, the NIC cycle will be serviced first. The following bus latency discussion shows there is no real need to give one master priority over the other.

Bus Latency Requirements

This is defined as the time between a master issuing a request and receiving an acknowledge.

The NuBus interface bus latency allowable is determined by its 25.6 μs bus timeout period. The maximum NuBus bus latency that can be expected on this card, that is, the longest Bus Request from the NIC, occurs if a packet ends just as the NIC performs its last FIFO burst. The local DMA burst plus the End of Packet processing operations add up to just over 4 μs, well within the allowable 25.6 μs.

The NIC bus latency allowable is determined by the need to prevent its internal FIFO from overflowing during packet reception. The worst case bus latency the NIC can accommodate running on a 20 MHz clock is a little more than 1 μs (refer to the DP8390 Datasheet addendum). The maximum NIC bus latency that can be expected on this card, that is, the longest NuBus cycle to the card, is 0.5 μs (five NuBus Clocks), well within the allowable 1 μs.

Again, Table IV shows all the cycles supported by the card, and the state sequence followed by each one. The figures quoted display the timing diagram for each case.

NUBUS MASTER CYCLES

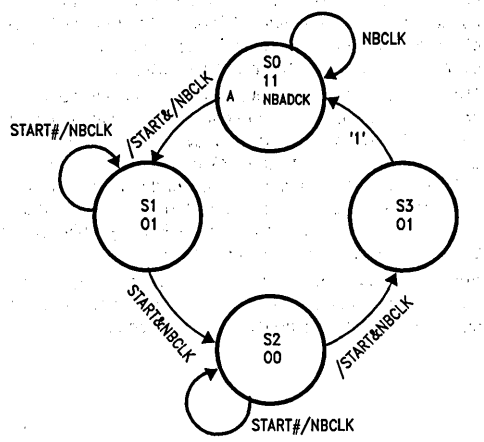
The CPU initiates NuBus cycles by driving the address and transfer mode onto NuBus and asserting the Start signal during the first clock of the bus access.

The adapter card latches the address onto the local bus on the falling edge of the first clock. A transparent latch is used so that the address is available on the local bus as soon as it is driven onto NuBus and address recognition can begin. A small asynchronous state machine in a control PAL® generates the address latching signal, which opens the latch when START becomes active and latches the address on the next falling edge of the clock. *Figure 7* shows the state machine diagram.

The address is enabled onto the local bus by NBADOE provided that the NIC is not already the bus master nor has issued a request while the sequencer is in the idle state.

The top 8 bits of the address are compared with the slot ID driven from NuBus.

The card, in response to the start signal, generates an enable signal (EBSLOT) which allows the sequencer to proceed onto the next NuBus cycle state (S1), provided the NIC bus request is not active and the above address comparison is successful (MYSLOT is active). This enable signal is cleared at the end of the NuBus cycle when ACK becomes active and prevents addresses not generated with a START signal from triggering the sequencer onto the S1 state. *Figure A-4* shows a detailed timing diagram of the address recognition operation. Note that FAST devices have been selected for the address latches and comparator to enable address recognition to meet the set up time of the sequencer.



TL/F/10805–7

A = State Variable (Not Used)

NBADCK = NuBus Address Clock

FIGURE 7. State Diagram for Clock to Latch NuBus Address onto Ethernet Card

States S1 to S4 cover the data portion of a NuBus transfer. The signal DASB (Data Strobe) is generated to qualify all the data enabling signals (NICCS, ROMCS, RAMOE, NBDBOE, DBNBOE). Note that during S2 DASB has to be released so that the state has a separate state number. This only affects NuBus to RAM cycles. Therefore during S2 the signal TOP is used to qualify the data enabling signals (RAMOE, NBDBOE, DBNBOE).

During S1 the address is further decoded with four possible outcomes. Each of the possible transfers initiated by NuBus are described.

Bus Error Transfer

If the address is not recognized by any of the on-card devices or it is recognized by the ROM with transfer mode defining a write cycle, a bus error condition is flagged to the CPU. The sequencer enters state S5 where the DASB signal is cleared, ACK is generated to signal the end of the cycle, and a bus error code is driven onto the NuBus transfer mode lines. The sequencer then returns to idle on the next clock beat. *Figure 8* shows a timing diagram for a NuBus error cycle.

NuBus ROM Transfer

If the address and transfer mode are decoded as a read cycle to ROM, the address decoding PAL generates a ROM enabling signal to the ROM chip select input which drives its data onto the local bus. The control PALs generate the "Data bus to NuBus output enable" signal DBNBOE to enable this data from the local bus onto NuBus. The sequencer transfers to state S4 on the next clock beat where ACK is driven onto NuBus together with the "transfer complete" code on the transfer mode signals. *Figures 8* and *A-1* show a basic and detailed timing diagram of the ROM read operation. Note that ROM set up times are easily met. Very slow ROMs can be used on this design, up to 135 ns data enable time or 210 ns address access time.

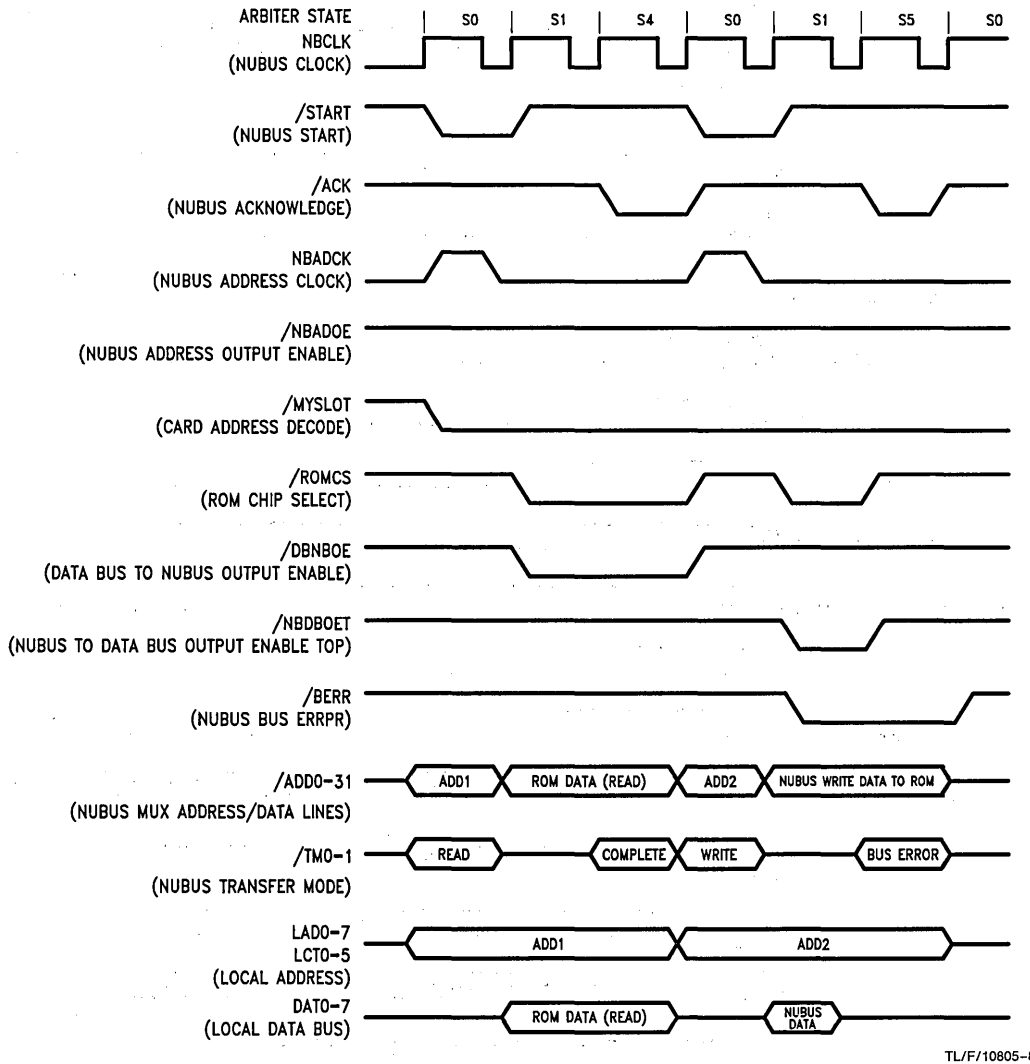
NuBus NIC Transfer

If the address and transfer mode are decoded as a read or write cycle to the NIC register, the address decoding PAL generates the NIC chip select signal NICCS, and the bottom four bits of the local address are sent to the NIC to select one of sixteen possible NIC registers. The sequencer, *Figure 6,* remains in state S1 until the NIC generates acknowledge signal NICACK. This signal is synchronized to the NuBus clock before it is fed into the synchronous state sequencer PAL. The sequencer then proceeds onto state S4.

See *Figure 9* for a functional timing diagram of NuBus to NIC read and write cycles.

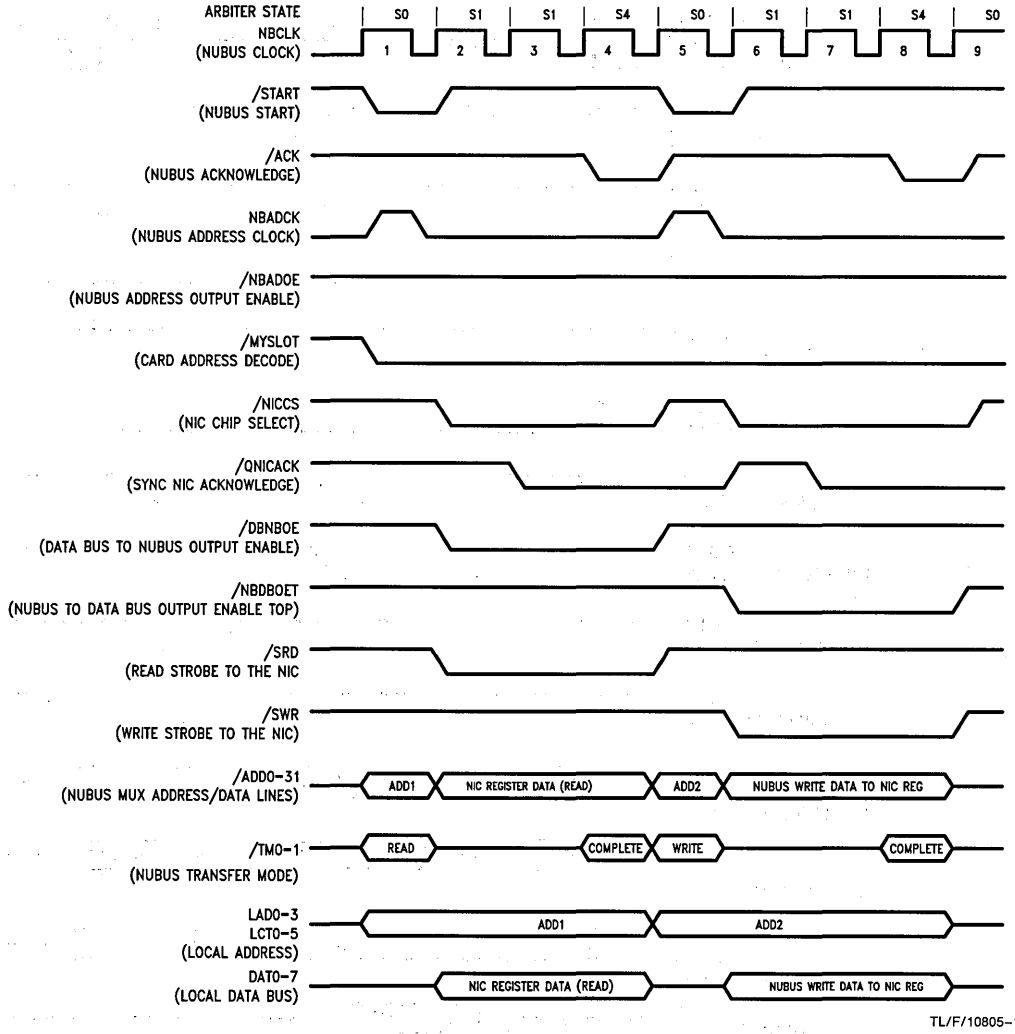
If the cycle is a write, the control PAL generates the NBDBOET signal to enable the NuBus write data onto the local bus, and a small asynchronous state machine in the PAL generates the write enable signal to the NIC, SWR. *Figure 10* shows the state machine diagram. This signal is cleared on the falling edge of the clock during the S4 state to provide the necessary write data hold time to the NIC. See *Figure A-3* detailed timing diagram.

If the cycle is a read, the control PAL generates DBNBOE to enable the NIC read data from the local bus onto NuBus. See *Figure A-2* for a detailed timing diagram.



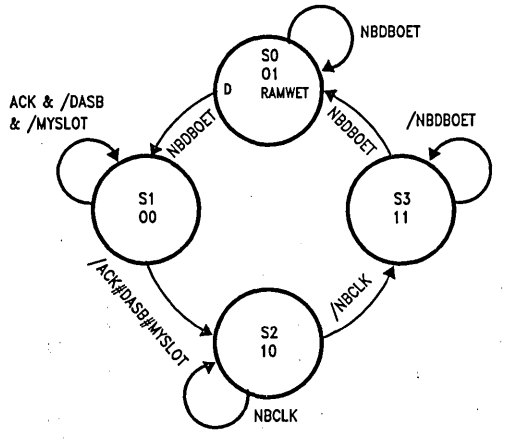
TL/F/10805–8

FIGURE 8. NuBus to ROM Read and Bus Error Cycles



TL/F/10805–10

FIGURE 9. NuBus to NIC Register Read and Write Cycle

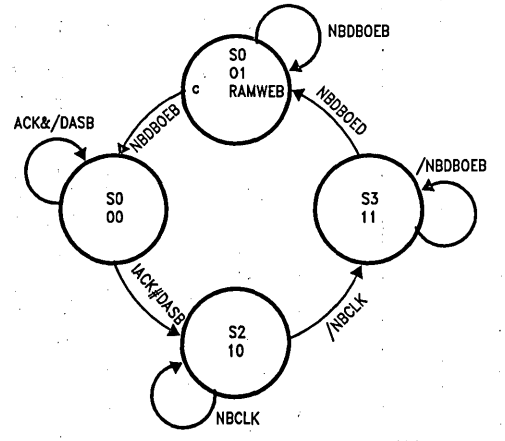


TL/F/10805–9

D = State Variable

RAMWET = RAM Write Enable Top

FIGURE 10. Write Enable Top State Diagram



TL/F/10805–11

C = State Variable (Not Used)

RAMWEB = RAM Write Enable Bottom

FIGURE 11. Write Enable Bottom State Diagram

NuBus to RAM Transfer

The address and transfer mode are decoded as a read or write cycle to the buffer RAM. The adapter card supports 32-bit NuBus transfers to the 16-bit buffer RAM. This is done by reading/writing to the RAM twice on every NuBus to RAM access, once during states S1 and S2 to access the least significant 16 bits of the NuBus word, and again during states S3 and S4 to access the most significant 16 bits of the NuBus word, after having incremented the bottom local address bit to the RAM (see the state diagram *Figure 6*). Therefore the NIC sees the buffer memory as an 8k x 16 RAM whereas the NuBus sees it as a 4k x 32 RAM. *Figure 12* shows a basic timing diagram for a NuBus to RAM read and write cycle.

For read cycles' data from the RAM read during states S1 and S2 is stored in the top two NuBus transceivers by setting the transceivers in storage mode and clocking them on the falling edge of the NuBus clock during state S2 with the TBCK (Top Bus Clock) signal. This data is enabled onto NuBus bits 16–31 with the signal NBDBOET. During states S3 and S4 the next RAM location is read and its data driven onto NuBus bits 0–15 through the bottom two NuBus transceivers, which are not set in storage mode (real time data mode).

Therefore by the time the adapter card drives ACK back to NuBus during state S4, the least significant 16 bits of data, corresponding to the first RAM location read, which were stored during S2, are being enabled onto NuBus bits 16–31 through the top two transceivers, and the most significant 16 bits of data, corresponding to the second RAM location read, are being enabled onto NuBus bits 0–15 through the bottom two NuBus transceivers. *Figure A-4* at the end of this note, shows a detailed timing diagram of a NuBus to RAM read cycle.

Note that the adapter card performs a hardware byte swap of NuBus data through the transceivers, so that the least significant byte of data from the RAM (Bits 0–7 on the local data bus of the first RAM location read) are driven onto byte 3 of NuBus (bits 24–31). This byte will be carried on byte lane 3 in the MACII system onto byte 3 of the MC68020 (data line bits 0–7).

For write cycles, during states S1 and S2, the top two NuBus transceivers (NuBus bits 16–31) are enabled onto the local data bus and their NuBus write data written into the Buffer RAM, with the bottom local address bit clear. During the next two states S3 and S4 the bottom two NuBus transceivers (NuBus bits 0–15) are enabled onto the local data bus and their NuBus write data written into the next Buffer RAM location with the bottom local address bit set. Two separate write enable signals are generated (RAMWET and RAMWEB) and ANDed together on the card to generate RAMWE. Two small asynchronous state machines are used to generate these signals. *Figure 10* and *11* show their state diagram. *Figure A-5* shows a detailed timing diagram of a NuBus write cycle to RAM.

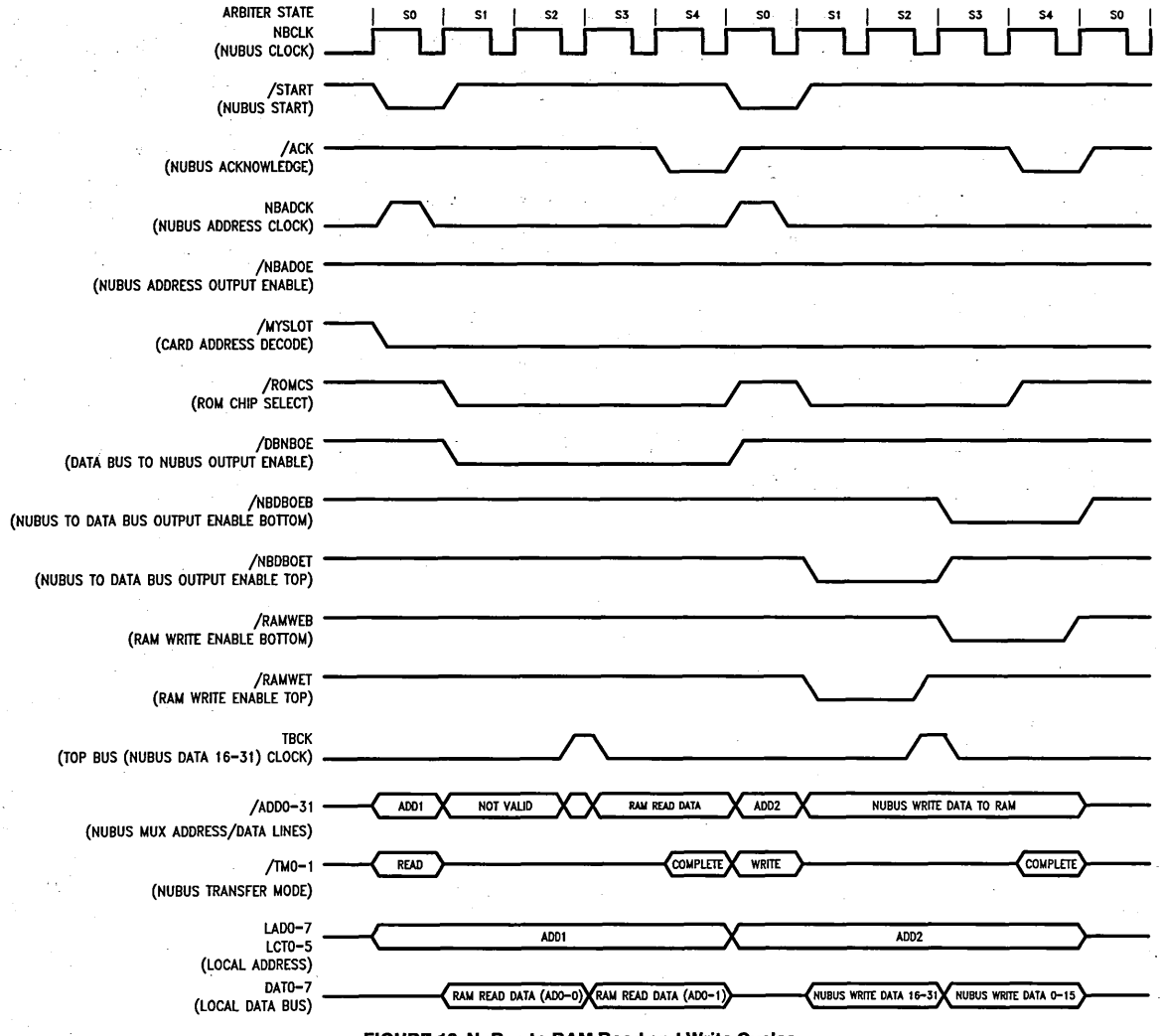
Supporting 32-bit transfers on NuBus rather than 16 only introduces one extra wait state per NuBus cycle to the RAM while doubling the data throughput per transfer.

NIC MASTER CYCLE

The NIC initiates local DMA cycles by driving its Bus Request line active. The sequencer/arbiter PAL, if in idle state S0, will enter state S6 where it acknowledges the request and hands over control of the local bus to the NIC. Any requests from the NuBus interface will be held until the NIC completes its local DMA burst and clears its request line allowing the sequencer PAL to return to the idle state S0.

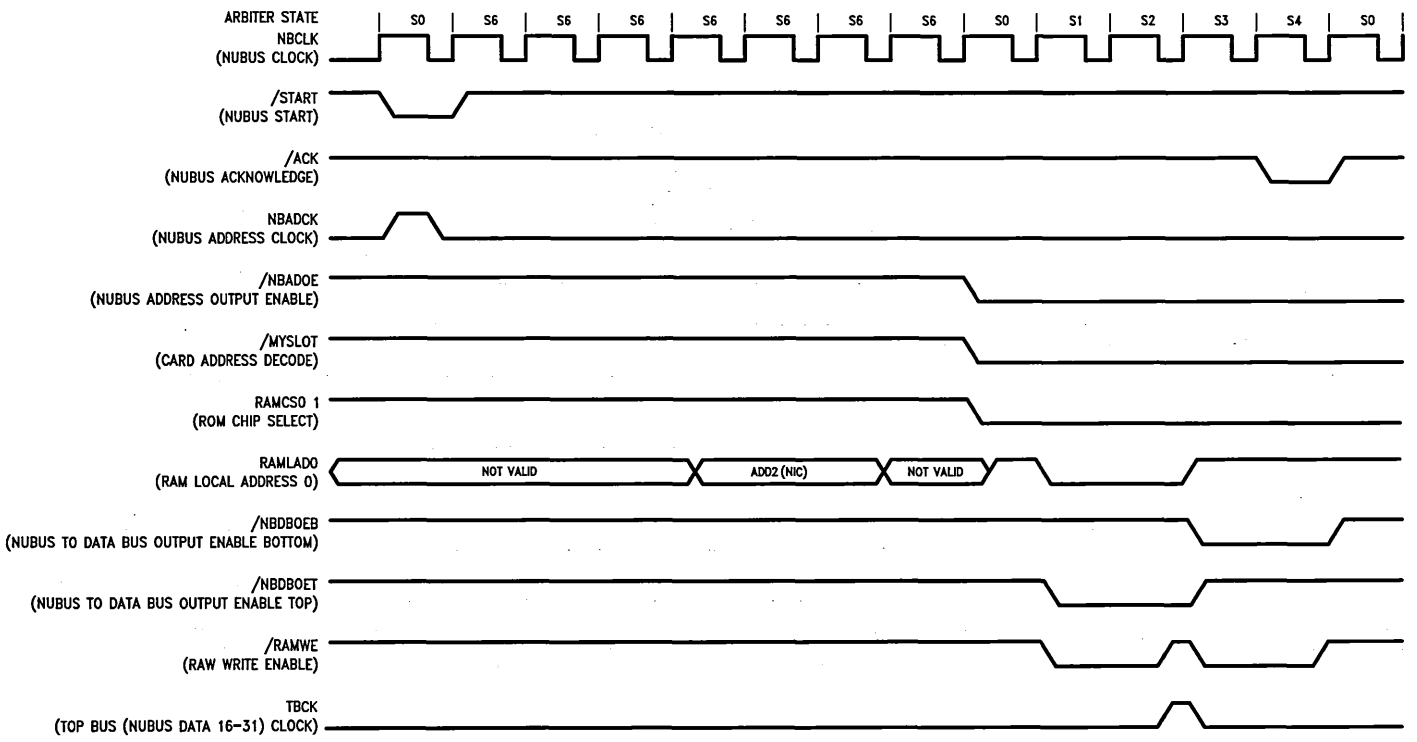
Figure 14 shows an NIC to RAM cycle, its request coinciding with the start of a NuBus cycle, thus illustrating the arbitration process.

Note the NIC runs on a separate 20 MHz clock, asynchronous to the NuBus clock. Therefore NIC signals to the arbiter sequencer are first synchronized to the NuBus clock with a D-type latch (F175) before they are used by the synchronous sequencer PAL running on the NuBus clock. The signals affected are BREQ and NICACK.



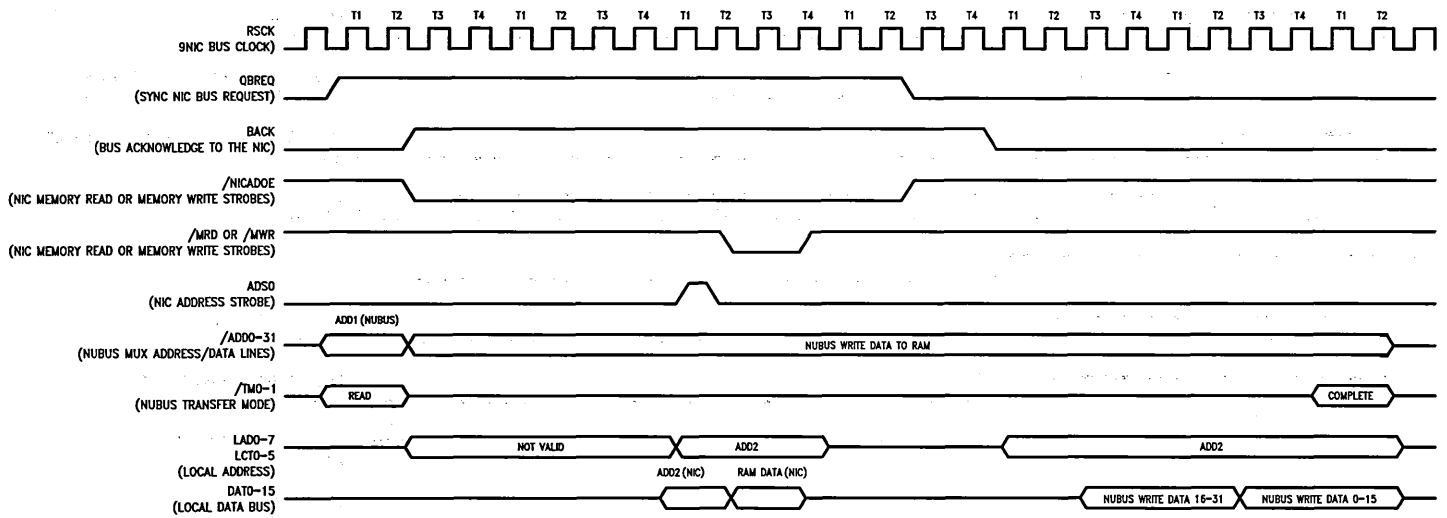
TL/F/10805-12

FIGURE 12. NuBus to RAM Read and Write Cycles



TL/F/10805–13

FIGURE 13. NuBus to RAM—Write Cycles with Arbitration



TL/F/10805–14

FIGURE 14. NIC to RAM—Write Cycles with Arbitration

```
module pal_1B

title 'Bus Controller
Andrew Pagnon  7-3-89';

" Modified for using Abel State_Machine language
"32bit Nubus Version"

"declarations"
      TRUE = 1;
      FALSE = 0;

PAL1B device 'P16R4';

"inputs"
      NBCK        pin 1;
      QBREQ       pin 2;
      QNICACK     pin 3;
      MYSLOT      pin 4;
      LCT4        pin 5;
      BERR        pin 6;
      EBSLOT      pin 7;
      LCT5        pin 8;
      ONE         pin 11;

"outputs"
      TM0         pin 12;
      TM1         pin 13;
      ACK         pin 14;
      DASB        pin 15;
      BACK        pin 16;
      TOP         pin 17;
      NBADOE      pin 18;
      NICADOE     pin 19;

"Declarations
      H,L,CK,XX = 1,0,.C.,.X.;

      input = [QBREQ,QNICACK,MYSLOT,LCT4,BERR,EBSLOT,LCT5];

      s0 = ^b1111;
      s1 = ^b1010;
      s2 = ^b1110;
      s3 = ^b1011;
      s4 = ^b0011;
      s5 = ^b0111;
      s6 = ^b1101;

equations

      enable TM0 = !ACK;
      !TM0 = BERR;
```

TL/F/10805–20

```
      enable TM1 = !ACK;
      !TM1 = !ACK;

      enable NBADOE = TRUE;
      !NBADOE = BACK & !ACK # BACK & !DASB # BACK & !QBREQ # BACK & !TOP;

        " NBADOE = !BACK* + BREQ . (ACK* . DASB* . BACK* . TOP*)"
  "NBADOE IS NOT ACTIVE IF BACK IS ACTIVE OR IF BREQ IS ACTIVE DURING S0"

      enable NICADOE = TRUE;
      !NICADOE = QBREQ & !BACK;

state_diagram [ACK,DASB,BACK,TOP]

      State s0:   case (input == [1,XX,XX,XX,XX,XX,XX])  :s6;
                        (input == [0,XX, 0,XX,XX, 0,XX])  :s1;
                        (input == [0,XX, 1,XX,XX,XX,XX])  :s0; "hold
                  endcase;

      State s1:   case (input == [XX,XX,XX,XX, 0,XX,XX]) :s5;  "BERR
                        (input == [XX, 0, 0, 0, 1,XX,XX]) :s4;  "NIC
                        (input == [XX,XX, 0, 1, 1,XX, 1]) :s4;  "ROM
                        (input == [XX,XX, 0, 1, 1,XX, 0]) :s2;  "RAM
                        (input == [XX, 1, 0, 0, 1,XX,XX]) :s1;  "hold
                  endcase;

      State s2:   goto s3;

      State s3:   goto s4;

      State s4:   goto s0;

      State s5:   goto s0;

      State s6:   case (input == [0,XX,XX,XX,XX,XX,XX])  :s0;
                        (input == [1,XX,XX,XX,XX,XX,XX])  :s6; "hold
                  endcase;
```

TL/F/10805-21

```
test_vectors
([NBCK,QBREQ,QNICACK,MYSLOT,LCT4,BERR,EBSLOT,LCT5,ONE] ->
[TM0,TM1,ACK,DASB,BACK,TOP,NBADOE,NICADOE])
  [.C.,0,1,1,1,1,1,1,0]   -> [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
  [.C.,0,1,1,1,1,1,1,0]   -> [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
  [.C.,0,1,1,1,1,1,1,0]   -> [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
  [.C.,0,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "TEST FOR IDLE          S0"
  [.C.,1,1,0,0,1,0,0,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "SET NIC BREQ           S6"
                                                     "NB TO NIC  START CYCLE  "
  [.C.,1,1,0,0,1,0,0,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "HOLD NIC BREQ          S6"
  [.C.,0,1,0,0,1,0,0,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "CLEAR NIC BREQ         S0"
  [.C.,0,1,0,0,1,0,0,0]   -> [.Z.,.Z.,1,0,1,0,0,1]; "NB TO NIC -            S1"
  [.C.,0,1,0,0,1,0,1,0]   -> [.Z.,.Z.,1,0,1,0,0,1]; "WAIT FOR NICACK        S1"
  [.C.,0,0,0,0,1,0,1,0]   -> [ 0 , 0 ,0,0,1,1,0,1]; "NICACK SETS            S4"
  [.C.,0,1,0,0,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "RETURN TO IDLE         S0"
  [.C.,0,1,0,0,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "IDLE WITH NO EBSLOT    S0"
  [.C.,0,1,0,1,1,0,0,0]   -> [.Z.,.Z.,1,0,1,0,0,1]; "NB TO RAM -            S1"
  [.C.,0,1,0,1,0,0,0,0]   -> [ 1 , 0 ,0,1,1,1,0,1]; "BERR SETS              S5"
  [.C.,1,1,0,1,0,1,1,0]   -> [.Z.,.Z.,1,1,1,1,1,1]; "IDLE NIC BREQ SETS     S0"
  [.C.,1,1,0,1,0,1,1,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "NIC MASTER             S6"
  [.C.,1,1,0,1,1,0,1,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "NB TO MEM -NIC MASTER  S6"
  [.C.,0,1,0,1,1,0,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "RETURN TO IDLE         S0"
  [.C.,0,1,0,1,1,0,1,0]   -> [.Z.,.Z.,1,0,1,0,0,1]; "NB TO ROM -            S1"
  [.C.,1,1,0,1,1,0,1,0]   -> [ 0 , 0 ,0,0,1,1,0,1]; "NIC BREQ - SET NB ACK S4"
  [.C.,1,1,0,1,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,1,1]; "RETURN TO IDLE         S0"
  [.C.,1,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "NIC MASTER             S6"
  [.C.,0,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "RETURN TO IDLE         S0"
  [.C.,0,1,0,1,1,0,0,0]   -> [.Z.,.Z.,1,0,1,0,0,1]; "NB TO RAM              S1"
  [.C.,1,1,0,1,1,0,0,0]   -> [.Z.,.Z.,1,1,1,0,0,1]; "NIC BREQ-LTCH DA16-31 S2"
  [.C.,1,1,0,1,1,0,0,0]   -> [.Z.,.Z.,1,0,1,1,0,1]; "EB DA0-15              S3"
  [.C.,1,1,0,1,1,0,0,0]   -> [ 0 , 0 ,0,0,1,1,0,1]; "ACK to NUBUS           S4"
  [.C.,1,1,0,1,1,0,0,0]   -> [.Z.,.Z.,1,1,1,1,1,1]; "RETURN TO IDLE         S0"
  [.C.,1,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,0,1,1,0]; "NIC MASTER             S6"
  [.C.,0,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "RETURN TO IDLE         S0"
  [.C.,0,1,1,1,1,1,1,0]   -> [.Z.,.Z.,1,1,1,1,0,1]; "STAY IN IDLE           S0"


end  pal_1B
```

TL/F/10805-22

```
module pal_2

title 'Memory decoder
Andrew Pagnon  8-3-89';

 "declarations"
     TRUE = 1;
     FALSE = 0;

PAL2 device 'P16L8';

"inputs"
     LCT0       pin 1;
     LCT1       pin 2;
     LCT2       pin 3;
     LCT3       pin 4;
     LCT4       pin 5;
     LCT5       pin 6;
     MYSLOT     pin 7;
     DASB       pin 8;
     NICADOE    pin 9;
     MSRAMSL    pin 11;

"outputs"
     RAMCS1     pin 12;
     ROMCS      pin 13;
     NICCS      pin 14;
     RAMCS0     pin 15;
     RAMCS3     pin 16;
     RAMCS2     pin 17;
     BERR       pin 18;
     SRD        pin 19;

equations

     enable RAMCS0 = TRUE;

     !RAMCS0 = !NICADOE & !MSRAMSL
     #  LCT0 &  LCT1 & !LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write byte 0 "
     # !LCT0 &  LCT1 &  LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write hw 0 "
     # !LCT0 &  LCT1 & !LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write word "
     # !LCT1 &  LCT4 & !LCT5 & !MYSLOT; " Read "

     enable RAMCS1 = TRUE;

     !RAMCS1 = !NICADOE & !MSRAMSL
     #  LCT0 &  LCT1 &  LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write byte 1 "
     # !LCT0 &  LCT1 &  LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write hw 0 "
     # !LCT0 &  LCT1 & !LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write word "
     # !LCT1 &  LCT4 & !LCT5 & !MYSLOT; " Read "
```

TL/F/10805-23

```
      enable RAMCS2 = TRUE;

      !RAMCS2 = !NICADOE & MSRAMSL
      #  LCT0 &  LCT1 & !LCT2 &  LCT3 & LCT4 & !LCT5 & !MYSLOT " Write byte 2"
      # !LCT0 &  LCT1 &  LCT2 &  LCT3 & LCT4 & !LCT5 & !MYSLOT " Write hw 1 "
      # !LCT0 &  LCT1 & !LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT " Write word "
      # !LCT1 & LCT4 & !LCT5 & !MYSLOT; " Read "

      enable RAMCS3 = TRUE;

      !RAMCS3 = !NICADOE & MSRAMSL
      #  LCT0 &  LCT1 &  LCT2 &  LCT3 & LCT4 & !LCT5 & !MYSLOT  " Write byte 3 "
      # !LCT0 &  LCT1 &  LCT2 &  LCT3 & LCT4 & !LCT5 & !MYSLOT  " Write hw 1 "
      # !LCT0 &  LCT1 & !LCT2 & !LCT3 & LCT4 & !LCT5 & !MYSLOT  " Write word "
      # !LCT1 &  LCT4 & !LCT5 & !MYSLOT; " Read "

      enable ROMCS = TRUE;
      !ROMCS = !LCT1 & LCT4 & LCT5 & !MYSLOT & !DASB; " Read "

      enable NICCS = TRUE;
      !NICCS = !DASB & !MYSLOT & !LCT4 & LCT5; " NIC Register Read or Write "

      enable SRD = TRUE;
      !SRD = !NICCS & !LCT1; " NIC register read "

      enable BERR = TRUE;
      !BERR = LCT1 & LCT4 & LCT5 & !MYSLOT      "ROM WRITE"
            # !LCT4 & !LCT5 & !MYSLOT;          "NOT IN CARD"

test_vectors
([LCT0,LCT1,LCT2,LCT3,LCT4,LCT5,MYSLOT,DASB,NICADOE,MSRAMSL] ->
[RAMCS0,RAMCS1,RAMCS2,RAMCS3,ROMCS,NICCS,BERR,SRD])
[.X.,.X.,.X.,.X.,.X.,.X.,1,.X.,0,0] -> [0,0,1,1,1,1,1,1];"NIC RD/WT RAM HW0"
[.X.,.X.,.X.,.X.,.X.,.X.,1,.X.,0,1] -> [1,1,0,0,1,1,1,1];"NIC RD/WT RAM HW1"
[1,1,0,0,0,1,0,0,1,.X.]              -> [1,1,1,1,1,0,1,1];"NB WT NIC BYTE0"
[0,0,0,0,0,1,0,0,1,.X.]              -> [1,1,1,1,1,0,1,0];"NB RD NIC WD"
[0,0,0,0,1,1,0,.X.,1,.X.]            -> [1,1,1,1,0,1,1,1];"NB RD ROM WD"
[0,0,0,0,1,0,0,.X.,1,.X.]            -> [0,0,0,0,1,1,1,1];"NB RD RAM WD"
[1,1,0,0,1,0,0,.X.,1,.X.]            -> [0,1,1,1,1,1,1,1];"NB WT RAM BYTE0"
[1,1,1,0,1,0,0,.X.,1,.X.]            -> [1,0,1,1,1,1,1,1];"NB WT RAM BYTE1"
[1,1,0,1,1,0,0,.X.,1,.X.]            -> [1,1,0,1,1,1,1,1];"NB WT RAM BYTE2"
[1,1,1,1,1,0,0,.X.,1,.X.]            -> [1,1,1,0,1,1,1,1];"NB WT RAM BYTE3"
[0,1,1,0,1,0,0,.X.,1,.X.]            -> [0,0,1,1,1,1,1,1];"NB WT RAM HW0"
[0,1,1,1,1,0,0,.X.,1,.X.]            -> [1,1,0,0,1,1,1,1];"NB WT RAM HW1"
[0,1,0,0,1,0,0,.X.,1,.X.]            -> [0,0,0,0,1,1,1,1];"NB WT RAM WD"
[1,0,.X.,.X.,1,0,0,.X.,1,.X.]        -> [0,0,0,0,1,1,1,1];"NB RD RAM BY    BERR"
[0,0,.X.,1,1,0,0,.X.,1,.X.]          -> [0,0,0,0,1,1,1,1];"NB RD RAM H1,BL BERR"
[0,0,1,0,1,0,0,.X.,1,.X.]            -> [0,0,0,0,1,1,1,1];"NB RD RAM H0    BERR"
[.X.,1,.X.,.X.,1,1,0,.X.,1,.X.]      -> [1,1,1,1,1,1,0,1];"NB WT ROM       BERR"
[.X.,.X.,.X.,.X.,0,0,0,.X.,1,.X.] -> [1,1,1,1,1,1,0,1];"ADD NOT IN CARD BERR"

end pal_2
```

TL/F/10805–24

```
module pal_3

title 'Memory and buffer control
Andrew Pagnon 23-3-89';

 "declarations"
     TRUE = 1;
     FALSE = 0;

PAL3 device 'P16L8';

"inputs"
     START      pin 1;
     NBCLK      pin 2;
     ACK        pin 3;
     DASB       pin 4;
     MYSLOT     pin 5;
     LCT1       pin 6;
     MRD        pin 7;
     TOP        pin 8;
     LCT5       pin 9;
     RESET      pin 11;

"outputs"
     DBNBOE     pin 12;
     NBADCK     pin 13;
     C          pin 14;
     RAMWEB     pin 15;
     NBDBOEB    pin 16;
     A          pin 17;
     EBSLOT     pin 18;
     ACKN       pin 19;

equations

     enable A = TRUE;
     !A = A & !NBADCK & START & NBCLK # !A & !NBADCK & START
         # !A & !NBADCK & !NBCLK # !A & !NBADCK & !START & NBCLK;

     enable NBADCK = TRUE;
     !NBADCK = A & NBADCK & !START & !NBCLK # A & !NBADCK & !START
             # A & !NBADCK & !NBCLK # A & !NBADCK & START & NBCLK
             # !A & !NBADCK & START # !A & !NBADCK & !NBCLK;

     enable C = TRUE;
     !C = !C & RAMWEB # !C & !RAMWEB & ACK & !DASB
         # C & RAMWEB & NBDBOEB;

     enable RAMWEB = TRUE;
     !RAMWEB = !C & RAMWEB & !NBDBOEB # !C & !RAMWEB & ACK & !DASB
              # !C & !RAMWEB & !ACK # !C & !RAMWEB & DASB
              # C & !RAMWEB & NBCLK;
```

TL/F/10805–25

```
      enable NBDBOEB = TRUE;
      !NBDBOEB = LCT1 & !DASB & !MYSLOT & TOP & !LCT5;

      enable DBNBOE = TRUE;
      !DBNBOE = !LCT1 & !DASB & !MYSLOT
              # !LCT1 & !TOP & !MYSLOT;

      enable EBSLOT = TRUE;
      !EBSLOT = EBSLOT & !START & RESET # !EBSLOT & ACK & RESET;

      enable ACKN = !ACK;
      !ACKN = !ACK;


test_vectors
([START,NBCLK,ACK,DASB,MYSLOT,LCT1,TOP,LCT5,RESET] ->
[A,NBADCK,C,RAMWEB,NBDBOEB,DBNBOE,EBSLOT,ACKN])
[1,1,1,1,1,.X.,1,.X.,0] -> [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[1,0,1,1,1,.X.,1,.X.,1] -> [.X.,.X.,.X.,.X.,.X.,.X.,.X.,.X.];
[1,1,1,1,1,.X.,1,.X.,1] -> [.X.,.X.,.X.,.X.,1,1,1,.Z.];
[1,0,1,1,1,.X.,1,.X.,1] -> [.X.,.X.,.X.,.X.,1,1,1,.Z.];
[0,1,1,1,1,.X.,1,.X.,1] -> [.X.,1,0,1,1,1,0,.Z.];"S0,START LOW NBCKL HIGH"
[0,1,1,1,0,1,1,0,1]     -> [1,1,0,1,1,1,0,.Z.];"S0,MYSLOT* LOW,S0"
[0,0,1,1,0,1,1,0,1]     -> [1,0,0,1,1,1,0,.Z.];"S0,NBCLK1=0,S0"
[1,1,1,0,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S1,NBCLK2=1,START*=1"
[1,1,1,0,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S1"
[1,0,1,0,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S1,NBCLK2=0"
[1,1,1,1,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S2 NBCLK3=1"
[1,1,1,1,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S2"
[1,1,1,1,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S2"
[1,0,1,1,0,1,0,0,1]     -> [0,0,0,1,1,1,0,.Z.];"S2,NBCLK3=0"
[1,1,1,0,0,1,1,0,1]     -> [0,0,0,0,0,1,0,.Z.];"S3,NBCLK4=1"
[1,1,1,0,0,1,1,0,1]     -> [0,0,0,0,0,1,0,.Z.];"S3"
[1,0,1,0,0,1,1,0,1]     -> [0,0,0,0,0,1,0,.Z.];"S3,NBCLK=0"
[1,1,0,0,0,1,1,0,1]     -> [0,0,1,0,0,1,1,0];  "S4"
[1,0,0,0,0,1,1,0,1]     -> [0,0,1,1,0,1,1,0];  "S4"
[1,1,1,1,0,1,1,0,1]     -> [0,0,0,1,1,1,1,.Z.];"S0"
[1,0,1,1,0,1,1,0,1]     -> [0,0,0,1,1,1,1,.Z.];"S0"
[0,1,1,1,0,1,1,1,1]     -> [1,1,0,1,1,1,0,.Z.];"S0,START*=0,NBCLK5=1"
[0,1,1,1,0,1,1,1,1]     -> [1,1,0,1,1,1,0,.Z.];"S0,NIC OR ROM WRITE"
[0,0,1,1,0,1,1,1,1]     -> [1,0,0,1,1,1,0,.Z.];"S0,NBCLK5=0"
[0,0,1,1,0,1,1,1,1]     -> [1,0,0,1,1,1,0,.Z.];"S0,MYSLOT*=1"
[1,1,1,0,0,1,0,1,1]     -> [0,0,0,1,1,1,0,.Z.];"S1,NBCLK6=1"
[1,0,1,0,0,1,0,1,1]     -> [0,0,0,1,1,1,0,.Z.];"S1"
[1,1,0,0,0,1,1,1,1]     -> [0,0,0,1,1,1,1,0];  "S4"
[1,0,0,0,0,1,1,1,1]     -> [0,0,0,1,1,1,1,0];  "S4"
[1,1,1,1,0,1,1,1,1]     -> [0,0,0,1,1,1,1,.Z.];"S0"
[1,0,1,1,.X.,.X.,1,.X.,1]->[0,0,0,1,1,1,1,.Z.];"S0"

end pal_3
```

TL/F/10805–26

```
module pal_4

title '32 bit Nubus control
Andrew Pagnon  11-10-89';

 "declarations"
     TRUE = 1;
     FALSE = 0;

PAL6 device 'P16L8';

"inputs"
     TOP        pin 1;
     NBCLK      pin 2;
     DASB       pin 3;
     BACK       pin 4;
     LAD0       pin 5;
     MYSLOT     pin 6;
     ACK        pin 7;
     BERR       pin 8;
     LCT1       pin 9;
     LCT4       pin 11;
     LCT5       pin 18;

"outputs"
     RAMLAD0    pin 12;
     TBCK       pin 13;
     NBDBOET    pin 14;
     RAMWET     pin 15;
     D          pin 16;
     DBNBOEB    pin 17;

equations

     enable RAMLAD0 = TRUE;
     !RAMLAD0 = !BACK & !LAD0 # BACK & !TOP;

     enable TBCK = TRUE;
     !TBCK = TOP # NBCLK # !DASB;

     enable NBDBOET = TRUE;
     !NBDBOET = LCT1 & !MYSLOT & !TOP "RAM"
            # LCT1 & !MYSLOT & !DASB & BERR & !LCT4; "NIC"

     enable RAMWET = TRUE;
     !RAMWET = !D & RAMWET & !NBDBOET
             # !D & !RAMWET & !DASB & !MYSLOT & ACK
             # !D & !RAMWET & DASB
             # !D & !RAMWET & !ACK
             # !D & !RAMWET & MYSLOT
             # D & !RAMWET & NBCLK;

     enable D = TRUE;
     !D = !D & RAMWET
        # !D & !RAMWET & !DASB & ACK & !MYSLOT
       # D & RAMWET & NBDBOET;
```

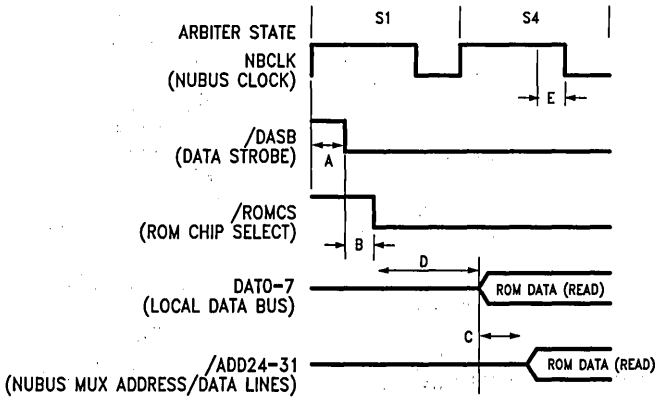
TL/F/10805–27

```
      enable DBNBOEB = TRUE;
      !DBNBOEB = !LCT1 & !DASB & !MYSLOT & !LCT5
               # !LCT1 & !TOP & !MYSLOT & !LCT5;

test_vectors
([TOP,NBCLK,DASB,BACK,LAD0,MYSLOT,ACK,BERR,LCT1,LCT4,LCT5] ->
[RAMLAD0,TBCK,NBDBOET,RAMWET,DBNBOEB])
[1,1,1,0,1,1,1,1,.X.,.X.,.X.] -> [.X.,.X.,.X.,.X.,.X.];"RESET"
[1,0,1,0,1,1,1,1,.X.,.X.,.X.] -> [.X.,.X.,.X.,.X.,.X.];"RESET"
[1,1,1,0,1,1,1,1,.X.,.X.,.X.] -> [1,0,1,1,1];"LAD0=1,BACK*=0"
[1,0,1,0,1,1,1,1,.X.,.X.,.X.] -> [1,0,1,1,1];
[1,1,1,0,0,1,1,1,.X.,.X.,.X.] -> [0,0,1,1,1];"LAD0=0,BACK*=0"
[1,0,1,0,0,1,1,1,.X.,.X.,.X.] -> [0,0,1,1,1];
[1,1,1,1,0,1,1,1,.X.,.X.,.X.] -> [0,0,1,1,1];"LAD0=0,BACK*=1,TOP=1->RAMLAD0=1"
[1,0,1,1,0,1,1,1,.X.,.X.,.X.] -> [0,0,1,1,1];
[0,1,0,1,0,0,1,1,1,0,1]      -> [1,0,0,0,1];"S1,NIC WT"
[0,0,0,1,.X.,0,1,1,1,0,1]    -> [1,0,0,0,1];"S1,NIC WT,NBCLK=0"
[1,1,0,1,.X.,0,0,1,1,0,1]    -> [0,0,0,0,1];"S4,NIC WT,ACK=0"
[1,0,0,1,.X.,0,0,1,1,0,1]    -> [0,0,0,1,1];"S4,NIC WT,ACK=0,NBCLK=0"
[1,1,1,1,.X.,0,1,1,1,0,1]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,1,0,1,1]      -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,1,0,1,1]      -> [1,0,1,1,1];"S1,ROM RD"
[0,0,0,1,.X.,0,1,1,0,1,1]      -> [1,0,1,1,1];"S1,ROM RD,NBCLK=0"
[1,1,0,1,.X.,0,0,1,0,1,1]      -> [0,0,1,1,1];"S4,ROM RD,ACK=0"
[1,0,0,1,.X.,0,0,1,0,1,1]      -> [0,0,1,1,1];"S4,ROM RD,ACK=0,NBCLK=0"
[1,1,1,1,.X.,0,1,1,0,1,1]      -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,1,0,1,0]      -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,1,0,1,0]      -> [1,0,1,1,0];"S1,RAM RD"
[0,0,0,1,.X.,0,1,1,0,1,0]      -> [1,0,1,1,0];"S1,RAM RD,NBCLK=0"
[0,1,1,1,.X.,0,1,.X.,0,1,0]    -> [1,0,1,1,0];"S2,RAM RD"
[0,0,1,1,.X.,0,1,.X.,0,1,0]    -> [1,1,1,1,0];"S2,RAM RD,NBCLK=0,TBCK=1"
[1,1,0,1,.X.,0,1,.X.,0,1,0]    -> [0,0,1,1,0];"S3,RAM RD,TOP=1"
[1,0,0,1,.X.,0,1,.X.,0,1,0]    -> [0,0,1,1,0];"S3,RAM RD,NBCLK=0"
[1,1,0,1,.X.,0,1,.X.,0,1,0]    -> [0,0,1,1,0];"S4,RAM RD,ACK=0"
[1,0,0,1,.X.,0,0,.X.,0,1,0]    -> [0,0,1,1,0];"S4,RAM RD,ACK=0,NBCLK=0"
[1,1,1,1,.X.,0,1,.X.,0,1,0]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,1,1,1,0]      -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,1,1,1,0]      -> [1,0,0,0,1];"S1,RAM WT"
[0,0,0,1,.X.,0,1,1,1,1,0]      -> [1,0,0,0,1];"S1,RAM WT,NBCLK=0"
[1,1,1,1,.X.,0,0,.X.,1,1,0]    -> [0,0,1,0,1];"S5,RAM WT,BERR=0"
[1,0,1,1,.X.,0,0,.X.,1,1,0]    -> [0,0,1,1,1];"S5,RAM WT,BERR=0,NBCLK=0"
[1,1,1,1,.X.,0,1,.X.,1,1,0]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,1,1,1,1]      -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,1,1,1,1]      -> [1,0,0,0,1];"S1,ROM WT"
[0,0,0,1,.X.,0,1,1,1,1,1]      -> [1,0,0,0,1];"S1,ROM WT,NBCLK=0"
[1,1,1,1,.X.,0,0,.X.,1,1,1]    -> [0,0,1,0,1];"S5,ROM WT,BERR=0"
[1,0,1,1,.X.,0,0,.X.,1,1,1]    -> [0,0,1,1,1];"S5,ROM WT,BERR=0,NBCLK=0"
[1,1,1,1,.X.,0,1,.X.,1,1,1]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,1,1,1,0]      -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,1,1,1,0]      -> [1,0,0,0,1];"S1,RAM WT"
[0,0,0,1,.X.,0,1,1,1,1,0]      -> [1,0,0,0,1];"S1,RAM WT,NBCLK=0"
[0,1,1,1,.X.,0,1,.X.,1,1,0]    -> [1,0,0,0,1];"S2,RAM WT"
[0,0,1,1,.X.,0,1,.X.,1,1,0]    -> [1,1,0,1,1];"S2,RAM WT,NBCLK=0,TBCK=1"
[1,1,0,1,.X.,0,1,.X.,1,1,0]    -> [0,0,1,1,1];"S3,RAM WT,TOP=1"
[1,0,0,1,.X.,0,1,.X.,1,1,0]    -> [0,0,1,1,1];"S3,RAM WT,NBCLK=0"
```

TL/F/10805–28

```
[1,1,0,1,.X.,0,1,.X.,1,1,0]    -> [0,0,1,1,1];"S4,RAM WT,ACK=0"
[1,0,0,1,.X.,0,0,.X.,1,1,0]    -> [0,0,1,1,1];"S4,RAM WT,ACK=0,NBCLK=0"
[1,1,1,1,.X.,0,1,.X.,1,1,0]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,0,1,.X.,0,1,1]    -> [0,0,1,1,1];"S0"
[0,1,0,1,.X.,0,1,.X.,0,1,1]    -> [1,0,1,1,1];"S1,ROM RD"
[0,0,0,1,.X.,0,1,1,0,1,1]      -> [1,0,1,1,1];"S1,ROM RD,NBCLK=0"
[1,1,0,1,.X.,0,0,.X.,0,1,1]    -> [0,0,1,1,1];"S4,ROM RD,ACK=0"
[1,0,0,1,.X.,0,0,.X.,0,1,1]    -> [0,0,1,1,1];"S4,ROM RD,ACK=0,NBCLK=0"
[1,1,1,1,.X.,0,1,.X.,0,1,1]    -> [0,0,1,1,1];"S0"
[1,0,1,1,.X.,1,1,.X.,.X.,.X.,.X.] -> [1,0,1,1,1];"S0"
[1,1,1,1,.X.,1,1,.X.,.X.,.X.,.X.] -> [1,0,1,1,1];"S0"
[.X.,.X.,.X.,0,0,.X.,.X.,.X.,.X.,.X.,.X.] ->
[0,.X.,.X.,.X.,.X.];"BACK*=0,LAD&RLAD=0"
[.X.,.X.,.X.,0,1,.X.,.X.,.X.,.X.,.X.,.X.] ->
[1,.X.,.X.,.X.,.X.];"BACK*=0,LAD&RLAD=1"

end pal_6
```

TL/F/10805-29



TL/F/10805–15

FIGURE A-1. Detailed NuBus to ROM Read Timing

TIMINGS

A = 15 ns, 16R4B prop delay, NbClk to DaSb

B = 15 ns, 16L8B prop delay, DaSb to ROMCS

C = 8 ns, F651 prop delay, Data Bus to NuBus

D = 35 ns, Max ROM enable access time, ROMCS to Data Bus

E = 21 ns, NuBus data set up time, NuBus data to NbClk low (S4)

Note: The address to the ROM is valid midway through the S0 state. Therefore the ROM address access time (70 ns) is not in the timing critical path.

ROM READ DATA SET UP TIME

Spec Times To Be Met

Read data set up time to NuBus (NbClk(S4)low) = 21 ns (Tsu in Nubus spec)

Adapter Card Times

Read data set up time

= Tcp(S1) + Tcw(S4) − A − B − C − D

= 100 + 75 + 15 + 15 − 8 − 35

= 102 ns (81 ns to spare)

Data hold times are as per RAM read cycles

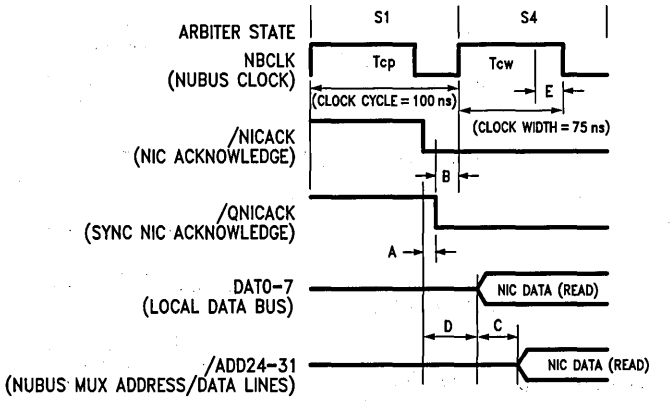


FIGURE A-2. Detailed NuBus to DP8390 Read Timing

TIMINGS

A = 4 ns, Min F175 prop delay, NICACK to QNICACK

B = 10 ns, Typ 16R4B Set Up, QNICACK to NBClk

C = 8 ns, F651 prop delay, Data Bus to NuBus

D = 55 ns, NIC Register access time, NICACK to Data Bus

E = 21 ns, NuBus Data set up time, NuBus data to NbClk

Data hold times are as per RAM read cycles

NIC REGISTER READ DATA SET UP TIME

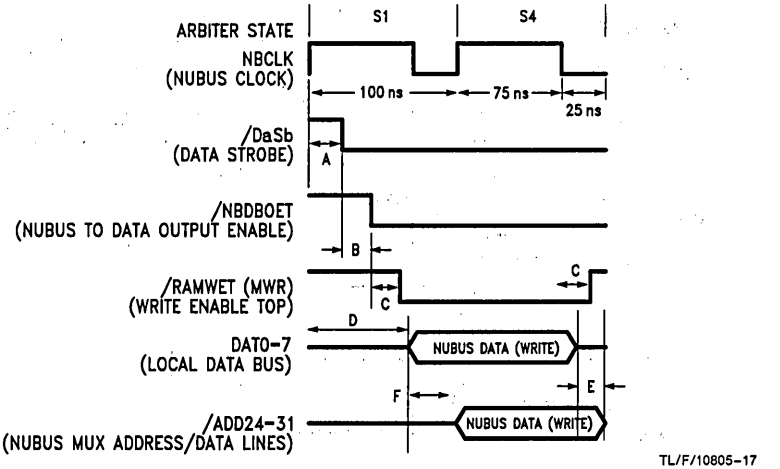
Spec Times To Be Met

Read data set up time to NuBus(NbClk(S4)low) = 21 ns (Tsu in NuBus spec)

Adapter Card Times

Read data set up time =
Tcw(S2) + A + B − C − D = 75 + 4 + 10 − 8 − 55
= 26 ns (5 ns to spare)

(Note B is a typical value)



TL/F/10805–17

FIGURE A-3. NuBus to NIC Write Timing

TIMINGS

A+B+C = 45 ns, 16R4B and 16L8B delay, NbClk to RAMWET

D = 52 ns, NuBus data enable (Ton+2Tpd), DaSb to ROMCS

E = 8 ns, F651 prop delay, Data Bus to NuBus

F = 25 ns, Nubus data hold time (Tcp–Tcw)

G = 2 ns, F651 min prop delay, Local bus to NuBus

For Tcp, Tcw, Ton, Tpd see RAM read timings

NIC REGISTER WRITE DATA SET UP TIME

Spec Times To Be Met

Write data set up time to end of MWR = 20 ns (NIC spec)

Adapter Card Times

Write data set up time to end of MWR = Tcp(S1)+Tcw(S2)−D−E = 115 ns (95 ns to spare).

NIC REGISTER WRITE DATA HOLD TIME

Spec Times To Be Met

Write data hold time after MWR = 17 ns (NIC spec)

Adapter Card Times

Write data hold time after MWR = F+E−C = 17 ns (C = 10 ns (D series PAL))

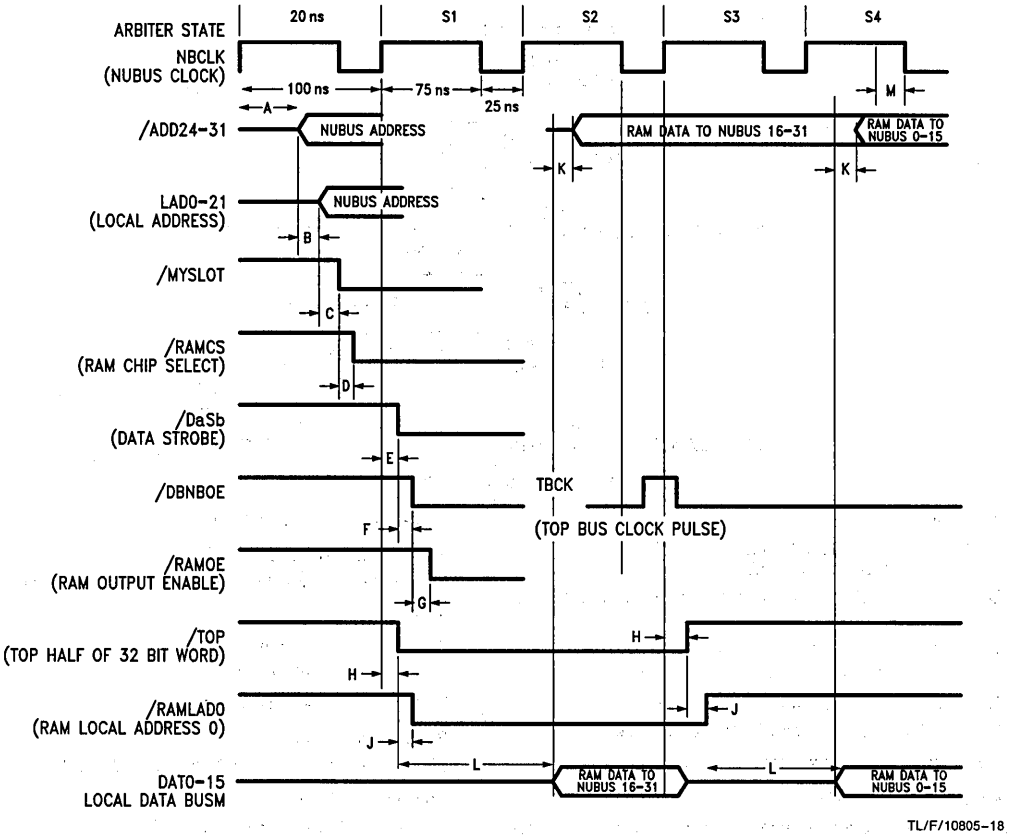
NIC REGISTER WRITE WIDTH FROM ACK

Spec Times To Be Met

Write width from ACK = 50 ns min (NIC spec)

Adapter Card Times

As RAMWET (MWR) is set before S4, ACK to RAMWET > 75 ns (>25 ns to spare).



TL/F/10805-18

FIGURE A-4. NuBus to RAM Read Timing and Address Recognition Timing

K = 8 ns, F651 prop delay, Local data bus to NuBus

P = 15 ns, PAL16L8B prop delay, NbClk(S2) low to TBCK

Spec Times To Be Met

RAM Read data set up time to NuBus (NbClk(S4)low) = M = 21 ns (Tsu in NuBus spec)

RAM Read data set up time in TBCK = 7 ns (F651 spec)

TBCK minimum pulse width = 7 ns (F651 spec)

Adapter Card Times

NbClk(S1) to RAM Read data (bottom 16 bits) on local data bus = H+J+L = 130 ns

Set up time to TBCK = Tcp(S1+S2)−130 = 70 ns (63 ns to spare)

NbClk(S3) to RAM Read data (32 bits) on NuBus = H+J+L+K = 138 ns

Adapter Card Times

Data buffer enabling signals are not cleared until after NbClk(S3) goes high

Hold time on Local data bus after TBCK > Tcp(S2)−Tcw(S2)−P = 10 ns (> 10 ns to spare)

Data must be held on NuBus while NbClk(S4) is low and as buffer enabling signals are not cleared until the next rising edge of NbClk, NuBus hold time is met

Address Recognition

= 52 ns, Nubus address valid (Ton+2Tpd), NbClk to NuBus

= 10 ns, F533 prop delay, Nubus to Local Address

= 11 ns, F521 prop delay, Local address to MySlot

TIMINGS

A = 15 ns, B series PAL prop delay, NbClk to DaSb or TOP

B = 15 ns, PAL16L8B prop delay, DaSb or TOP to NBDBOE

C = 15 ns, PAL16L8B prop delay, NBDBOE to RAMWET/B

D = 6.5 ns, F11 prop delay, RAMWET/B to RAMWE

E = 52 ns, NuBus data turn on time, NbClk(S1) to Local data bus

F = 8 ns, F651 prop delay, Local data bus to NuBus

RAM CE TO END OF WE

Spec Times To Be Met

RAM CE to end of RAMWE = 85 ns (RAM spec)

Adapter Card Times

RAMCE is driven during S0 (See RAM read timing)

RAMCE to end of RAMWE>Tcp(S1)+Tcw(S2) = 175 ns (>90 ns to spare)

RAM WE PULSE WIDTH

Spec Times To Be Met

RAMWE pulse width = 70 ns min (RAM spec)

Adapter Card Times

RAMWE pulse width >Tcp(S1)+Tcw(S2)−A−B−C−D = 123.5 ns (53.5 ns to spare)

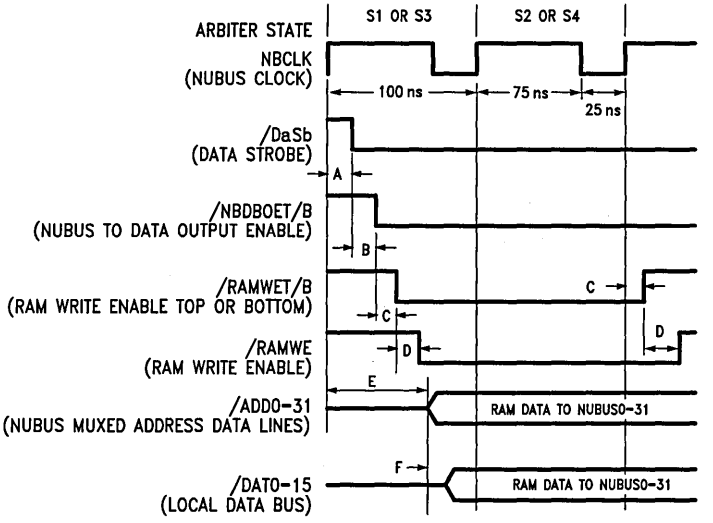
DATA VALID TO END OF WE

Spec Times To Be Met

Data valid to end of RAMWE = 50 ns min (RAM spec)

Adapter Card Times

Data valid to end of RAMWE = Tcp(S1)+Tcw(S2)−E−F = 115 ns (65 ns to spare)



TL/F/10805–19

FIGURE A-5. NuBus to RAM Write Timing

DATA HOLD AFTER END OF WE

Spec Times To Be Met

Data hold after end of RAMWE = 0 ns min (RAM spec)

Adapter Card Times

Data buffer enabling signals are not cleared unitl after NbClk(S3) goes high

Data hold after end of RAMWE >

Tcp(S2 or S4)+Tcw(S2 or S4)−C−D = 3.5 ns
(>3.5 ns to spare)

DP839EB Network Evaluation Board

National Semiconductor
Application Note 479

GENERAL DESCRIPTION

The DP839EB LAN Evaluation Board provides IBM PCs, IBM PC ATs and compatibles with Ethernet, Thin-Ethernet, Twisted Pair Ethernet and StarLAN connections. The DP839EB is compatible with the PC-bus and requires only a ½ size slot for installation.

The LAN Evaluation Board utilizes the National Semiconductor IEEE 802.3 chip set consisting of the DP8390 Network Interface Controller, the DP8391 Serial Network Interface Adapter, and the DP8392 Coaxial Transceiver Interface. The dual DMA capabilities of the DP8390, coupled with 8 kBytes of local buffer RAM allow the evaluation board to appear as an I/O port to the system.

The Evaluation Board provides the designer with a good example of an 8-bit controller implementation, that can be extrapolated to 16-bit buses.

In addition, software is included with the board to facilitate low level functional check out of the board, and as a guide to programming the board. The source code is provided, and can serve as a model for additional software development.

Additionally the DP839EB is compatible with Novell NetWare™, and TCP/IP software support is available through third party developers.

HARDWARE FEATURES

- Half-size IBM PC card form factor
- DP8390 IEEE 802.3 Chip Set utilizing dual DMA controller
- 8 kByte on board packet buffer
- Interfaces to PC using DMA, or to an AT via string I/O instructions
- Ethernet connector (15 pin D), Thin-Ethernet connector (BNC)
- StarLAN with optional daughter card
- Low power operation
- Utilizes DP8390, DP8391, and DP8392

SOFTWARE FEATURES

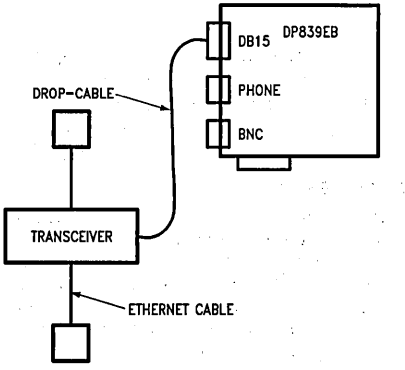
- No Software changes for conversion between Ethernet/ Cheapernet and StarLAN
- Demonstration and diagnostic software available

NETWORK INTERFACE OPTIONS

The evaluation board supports three physical layer options: Ethernet, Cheapernet and StarLAN. When using Ethernet, a drop cable is connected to an external transceiver which is connected to a standard Ethernet network. (See *Figure 1*). When using Cheapernet, a low cost version of Ethernet, a transceiver is available on-board allowing direct connection to the network via the evaluation board. (See *Figure 2*). When using a StarLAN network, an optional daughter card replaces the SNI function and implements the required electronics to interface the DP8390 NIC to StarLAN. This configuration is illustrated in *Figure 3*. No software changes are needed for conversion between any of the described configurations.

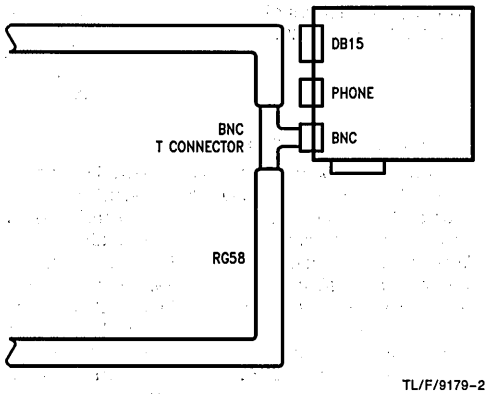
HARDWARE DESCRIPTION

The block diagram shown in *Figure 4* illustrates the architecture of the Network Interface Adapter. The system/network interface is partitioned at the DP8390 Network Interface Controller (NIC). The NIC acts as both a master and a slave on the local bus. During reception or transmission of packets, the NIC is a master. When accessed by the PC, the NIC becomes a slave. The NIC utilizes a local 8-bit data bus connected to an 8k x 8 Static RAM for packet storage. The 8k x 8 RAM is partitioned into a transmit buffer and a receive buffer. All outgoing packets are first assembled in the packet buffer and then transmitted by the NIC. All incoming packets are placed in the packet buffer by the NIC and then transferred to the PC's memory. The transfer of data between the evaluation board and the PC is accomplished using the PC's DMA in conjunction with the NIC's Remote DMA. Two LS374 latches implement a bidirectional I/O port with the PC bus. The 8-bit transceiver (LS245) allows the PC to access to the NIC's internal registers for programming. A 32 x 8 PROM located on the evaluation board contains the unique Physical Address assigned to each board.



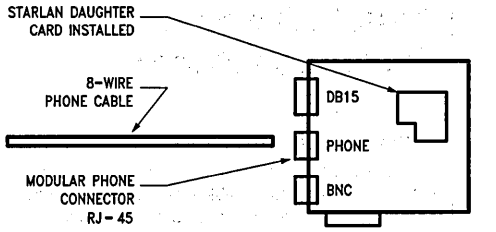
TL/F/9179–1

FIGURE 1. Ethernet Connection



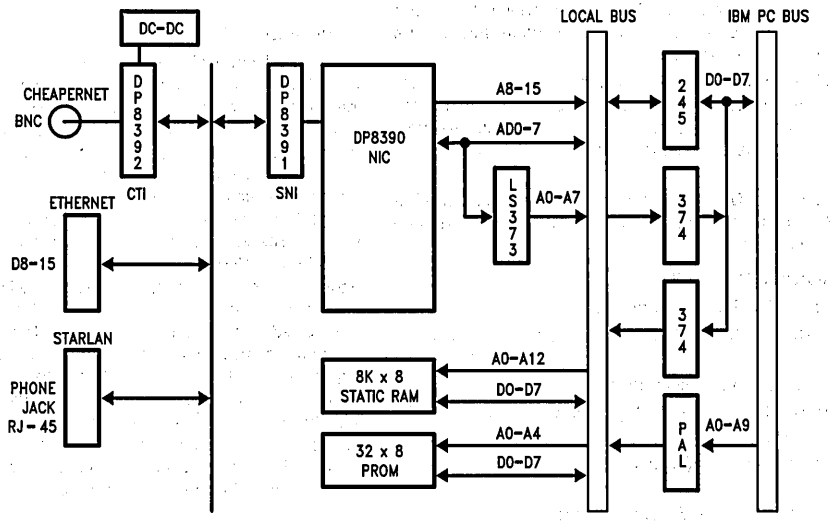
TL/F/9179-2

FIGURE 2. Cheapernet Connector



TL/F/9179-3

FIGURE 3. StarLAN Connector



TL/F/9179-4

FIGURE 4

Since the NIC is accessing 8-bit memory, only a single demultiplexing latch is required for the lower 8-bits of address. An LS373 is provided for this purpose.

A 20L8 PAL provides the address decoding and support for DMA handshaking and wait state generation.

SOFTWARE SUPPORT

The evaluation board provides a simple programming interface for development of software. Several software packages are provided for evaluation and development of networks using the DP8390 Chip set. SDEMO is a demonstration program that provides a low level interface to the DP8390 NIC for transmission and reception of packets. SDEMO supports register dumps and simple register modification. CONF is a conferencing program which supports simple message transfer. WORKSTAT and SERVER support file transfer between two nodes, one configured as a server and a second configured as a workstation. NLS, Network Load Simulator, is a program that simulates network loads based on statistical distributions of packet sizes, bursts and intervals. NLS is useful for performance measurement and debug of software drivers. NES, Network Evaluation Software, consists of sample software drivers implementing a low level interface to the evaluation board.

LOCAL MEMORY MAP

The DP8390 NIC accesses an 8k x 8 buffer RAM located in its 64 kbyte memory space. This buffer RAM is used for temporary storage of receive and transmit packets. Data from this RAM is transferred between the host (the PC) and the evaluation board using the DP8390 NIC's remote DMA channel. An ID address PROM, containing the physical address of the evaluation board is also mapped into the memory space of the NIC.

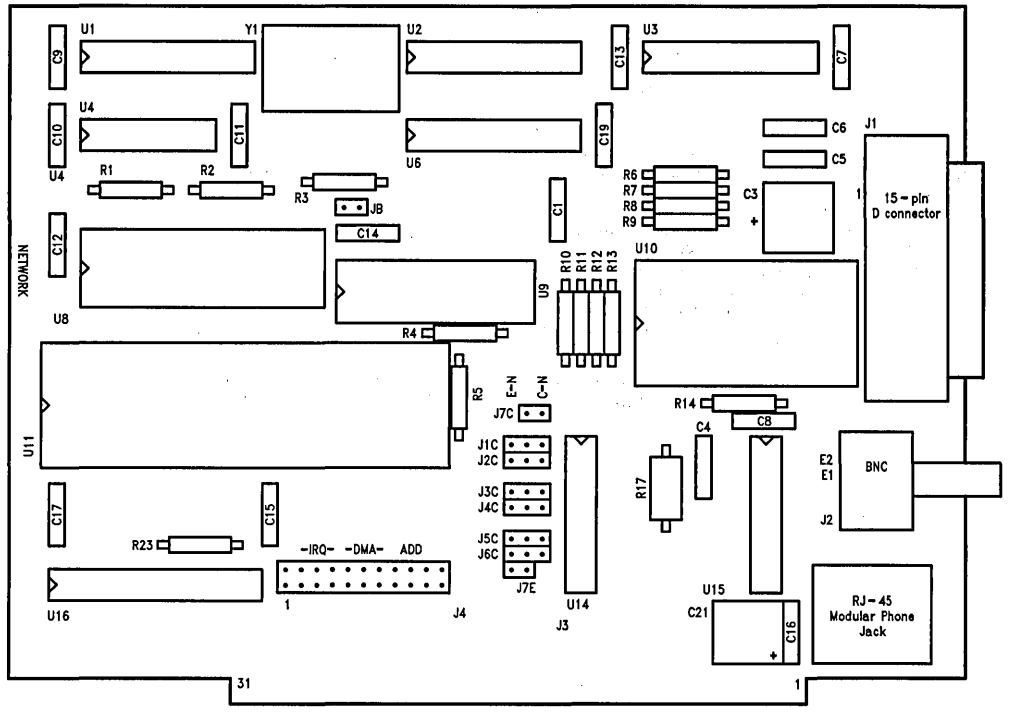
Note: Partial decoding is performed on the PROM and RAM which will result in these devices appearing at other locations in the 64k memory space. The first occurrance of the PROM and RAM are used for programming purposes.

0000h 001fh	PROM
•	•
•	•
2000h 3fffh	8k x 8 BUFFER RAM
•	•
•	•
ffffh	•

PROM FORMAT

Each evaluation board is assigned a unique network (physical) address. This address is stored in a 74S288 32 x 8 PROM. The physical address is followed by a checksum. The checksum is calculated by exclusive OR-ing the 6 address bytes with each other. At initialization the software reads the PROM, verifies the checksum and loads the NIC's physical address registers. The following format is used in the PROM:

Address	Contents
00h	ADDRESS 0 (Physical Address Most Significant Byte)
01h	ADDRESS 1
02h	ADDRESS 2
03h	ADDRESS 3
04h	ADDRESS 4
05h	ADDRESS 5 (Physical Address Least Significant Byte)
06h	CHECKSUM (XOR OF ADDRESS 0-5) OPTIONAL
07h	REV. NUMBER
08h	MANUFACTURE LOT #
09h	MANUFACTURE DATE (MONTH)
10h	MANUFACTURE DATE (YEAR)
11h–1fh	RESERVED



TL/F/9179–5

FIGURE 5

I/O SPACE

The I/O space and Ethernet/Cheapernet configurations are selected using the various I/O jumpers. There are 4 sets of jumpers that should be programmed prior to installation of the evaluation board into the PC environment. There are:

J4 I/O address, interrupt selection, DMA channel assignment

J1C–J7C, J7E Select Ethernet or Cheapernet

Figure 5 depicts the location of the jumpers on the evaluation board.

The Factory Installed Configuration Is:

J4 I/O base = 300h
Interrupt = IRQ3
DMA = DREQ1, DACK1

J1C–J7C, J7E Cheapernet selected

The evaluation board uses 32 I/O locations in the PC's I/O space. The base address is fixed at 300h and is not selectable using jumpers. (See Switch settings section.) The I/O map is shown below:

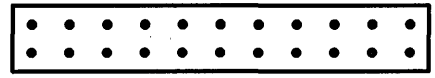
BASE + 00h	COMMAND REGISTER
01h	NIC REGISTER
02h	SPACE
03h	•
04h	•
05h	•
06h	•
07h	•
•	•
0fh	•
10h	I/O PORTS
•	•
1fh	•

NOTES: The NIC's Command Register is always mapped at Base + 0. The NIC registers are Base + 01 to Base + 0f; 0f will contain different registers depending on the value of bits PS0 and PS1 in the Command Register. These two bits select one of three register pages. For additional information consult the DP8390 data sheet.

The NIC uses the remote DMA channel to read/write data from/to the 8k x 8 Buffer RAM on the evaluation board. Typically a DMA channel on the PC is used in conjunction with the NIC's remote DMA. The I/O ports are then serviced by the DMA channel. If a DMA channel on the PC is not available, the NIC's DMA can still be used by accessing the I/O ports using programmed I/O. Reading the I/O port address will result in a $\overline{RACK}$ strobe to the NIC while writing the I/O port address will result in a $\overline{WACK}$ strobe to the NIC.

SWITCH SETTINGS

Jumper J4 allows assignment of I/O Address Bases, DMA channel assignments and Interrupt Request assignments. The jumper configuration is shown below and described in the following sections.



TL/F/9179–6

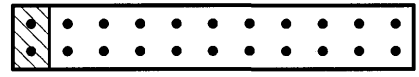
I/O BASE ADDRESS

The I/O Base Address for DP8390B boards is fixed at 300h and is not selectable.

INTERRUPTS

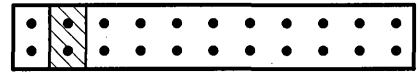
The NIC will generate interrupts based on received and transmitted packets, completion of DMA and other internal events. The interrupt can be connected to Interrupts 2, 3, 4 or 5 (IRQ 2, 3, 4, 5) via Jumper J4. Interrupt 5 is also provided as a software driven DMA Channel. If Interrupt 5 is being used as a DMA channel Interrupt 5 cannot be chosen for the NIC interrupt. The figures below illustrate the jumper positions for the various interrupt levels.

Interrupt 2



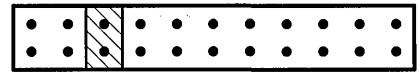
TL/F/9179–9

Interrupt 3
(Factory Installed)



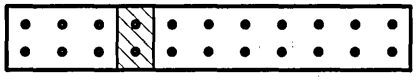
TL/F/9179–10

Interrupt 4



TL/F/9179–11

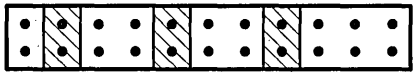
Interrupt 5



TL/F/9179–25

Note: Rev D demo software will not work unless the factory configuration for Jumper Block J4 is used.

FACTORY CONFIGURATION:

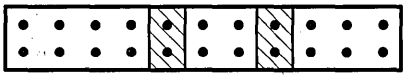


TL/F/9179–26

DMA

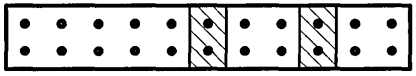
The evaluation board may use 1 DMA channel on the PC expansion bus. DMA channel 1 or 3 can be selected. The corresponding DACK line must also be installed on Jumper J4.

DMA Channel 1
(Factory Installed)



TL/F/9179–15

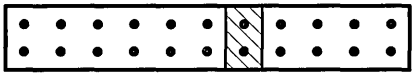
DMA Channel 3



TL/F/9179–16

If a DMA channel is not available an interrupt driven routine can be used to move data between the PC and the buffer memory on the evaluation board. Interrupt 5 is used for this function.

IRQ 5 for DMA

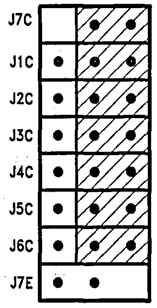


TL/F/9179–17

SELECTING ETHERNET OR CHEAPERNET

Two 10 Mbit/sec Interface options are available, a connection to an external transceiver via the DB-15 connector, or a direct interface to a BNC T-connector. Seven jumpers are used to select the appropriate option. These jumpers are labeled J1C-J7C and J7E.

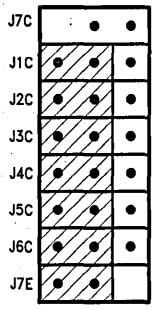
For Cheapernet the following jumpers should be shorted:



TL/F/9179–18

(Factory Installed)

For Ethernet the following jumpers should be shorted.



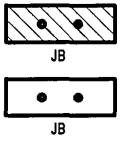
TL/F/9179–19

Double check the jumper positions prior to powering up the board.

OSCILLATOR

When the StarLAN daughter board is used, the 20 MHz oscillator must be disconnected by removing jumper JB. The StarLAN daughter board provides the clock to the NIC.

Ethernet, Cheapernet
(Factory Installed)



StarLAN
(Removed)

TL/F/9179–21

APPENDICES

The remainder of this document contains the evaluation board parts list, schematic and PAL descriptions.

1

PARTS LIST*

Item #	Description	Reference Designator	Qty
1	RES. CC 4.7K Ω 1/4W 5%	R1, R2, R3, R23	4
2	RES. CF 39 Ω 1/4W 5%	R6, R7, R8, R9	4
3	RES. CF 1.5K Ω 1/4W 5%	R10, R11, R12, R13	4
4	RES. CF 1M Ω 1/2W 5%	R17	1
5	RES. CF 270K Ω 1/4W 1%	R4, R5	2
6	RES. MK 1K Ω 1/4W 1%	R14	1
7	CAP. FILM 0.01μF 630V	C4	1
8	CAP. DIP TANT 100 μF 10V RD	C3, C21	2
9	CAP. DIP 0.47 μF 50V 0.3LS	C1, C7–C17, C19	13
10	CAP. CER 0.01 μF 50V 0.2LS	C5, C6	2
11	I.C. 74LS245	U3	1
12	I.C. 74LS374	U2, U6	2
13	I.C. 74LS373	U1	1
14	SRAM HM6264-1OO	U8	1
15	PROM 74S288	U4	1
16	PAL20L8	U16	1
17	TRANSFORMER PE64103	U14	1
18	OSCILLATOR 20.00 MHz	Y1	1
19	JUMPER, 2 POSITION	A/R	13
20	CONN. 15 POS D-SUB	J1	1
21	CONN. MODULAR JACK		1
22	CONN. BNC, R/A PCB MOUNT	J2	1
23	HEADER, 2 PIN SINGLE ROW	JB, J7C, J7E	3
24	HEADER, 3 PIN SINGLE ROW	J1C–J6C	6
25	HEADER, 44 PIN DOUBLE ROW	J4	0.5
26	SOCKET, 24 PIN DIP	U11	2
27	SOCKET, 24 PIN DIP (.300)	U16	1
28	SOCKET, 24 PIN (MACH)	U9	1
29	BRACKET, CNET	J1	1
30	SPACER, D-25 SET	J1	1
31	PCB		
32	DC-DC CONVERTER, 2VP5U9	U10	1
33	I.C. DP8390BN	U11	1
34	I.C. DP8391N	U9	1
35	I.C. DP8392AN	U15	1

*551 A201-01 REV D Board

```
PAL20L8
Decode and DMA Interface Logic for the DP839EB
DECODE1
A9   A8    A7   A6   A5    A4  PCRST /NMRD  NA13 /IORD  /IOWR GND
PRQ /DACK /WAIT AEN /RACK /WACK /CSX /CSN  /NICRST  /CSROM /ACK  VCC

CSN =  /AEN* A9* A8* /A7* /A6* /A5* /A4* IOWR +
       /AEN* A9* A8* /A7* /A6* /A5* /A4* IORD

NICRST = PCRST

RACK = /AEN* A9* A8* /A7* /A6* /A5* A4* PRQ* IORD +
        DACK*  IORD

WACK = /AEN* A9* A8* /A7* /A6* /A5* A4* PRQ* IOWR +
        DACK*  IOWR

CSX =   CSN* IORD +
        CSN* IOWR +
       /AEN* A9* A8* /A7* /A6* /A5* A4* IORD +
       /AEN* A9* A8* /A7* /A6* /A5* A4* IOWR +

IF (CSX)
WAIT = /ACK *  CSN  +
       /PRQ * /CSN

CSROM = /NA13 * NMRD
```

DESCRIPTION

This PAL performs the I/O decodes for selecting the NIC, and the handshake signals for NIC's remote DMA. The PAL supports the DMA channels of the PC for remote DMA transfers with the NIC and also allows the use of string I/O between 80286 PC's and NIC's remote DMA.

DECODE1 fixes the I/O BASE of the card at 300h. NIC registers fall in the space 300h–30fh. To use the string I/O port, reads and writes are done to port 310h.

Wait states are inserted (WAIT) to the PC bus when register accesses are given and the NIC is busy performing DMA operations. When the NIC is ready, /ACK is given and no (more) wait states are inserted.

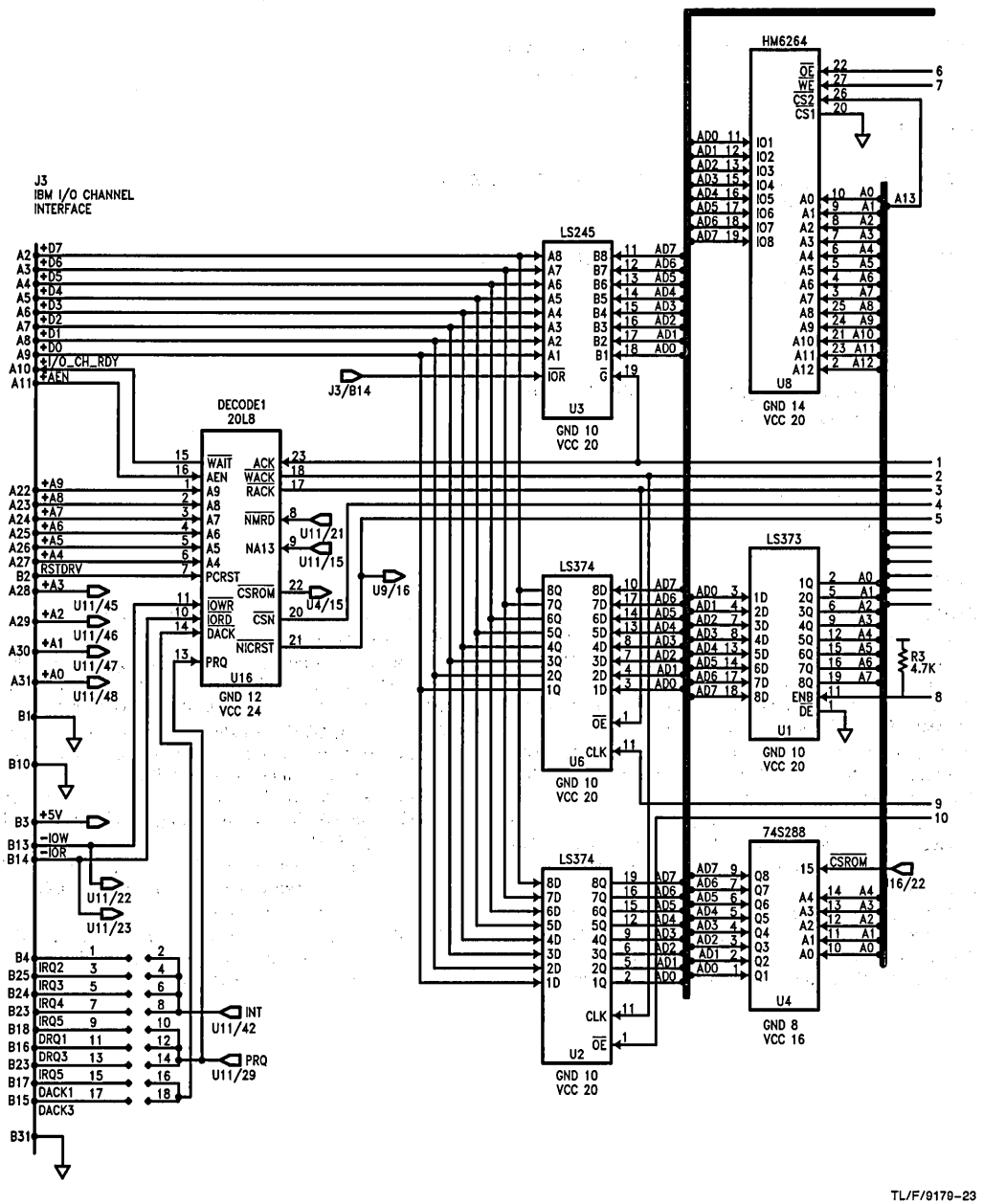
Wait states may also be inserted during remote DMA operations and 80286 machines using string I/O's. WAIT occurs during a remote read if the PC AT's /IORD goes low before the DP8390's PRQ goes high. Similarly, WAIT occurs during a remote write if the PC AT's /IOWR goes low before the NIC's PRQ goes high.

NIC registers are accessed when CSN (Chip Select NIC) is asserted. The IORD and IOWR terms are included to ensure that the address lines are valid when CSN is given.

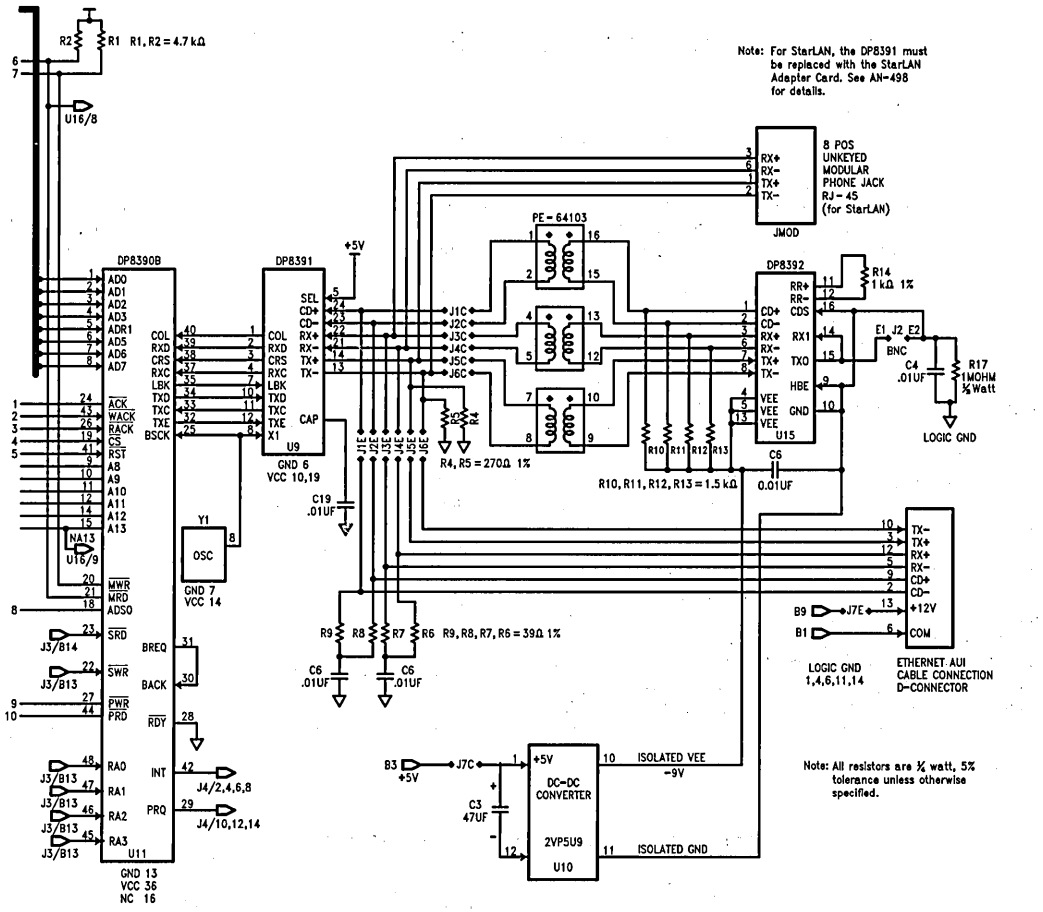
The RACK and WACK signals are used by the NIC's remote DMA channel to acknowledge the end of a single read or write operation through the remote DMA I/O ports. These ports are addressable by the PC DMA channel with DACK and IORD or IOWR, or by addressing the I/O location 310h (with string I/O's).

CSX is used to enable the TRI-STATE output of WAIT during a register access CSN), and during string I/O to the remote DMA's I/O port (CSX).

CSROM provides address decode for the address PROM. The card's unique Ethernet address is transferred to the system using the NIC's remote DMA.


J3
IBM I/O CHANNEL
INTERFACE
+D7
+D6
+D5
+D4
+D3
+D2
+D1
+D0
+I/O_CH_RDY
+AEN
+A9
+A8
+A7
+A6
+A5
+A4
RSTDRV
+A3
+A2
+A1
+A0
-IOW
-IOR
+5V
IRQ2
IRQ3
IRQ4
IRQ5
DRQ1
DRQ3
IRQ5
DACK1
DACK3
INT
PRQ
U11/42
U11/29
U11/45
U11/46
U11/47
U11/48
U11/22
U11/23
U11/21
U11/15
U4/15
U9/16
J3/B14
DECODE1
20L8
U16
GND 12
VCC 24
WAIT
AEN
ACK
WACK
RACK
NMRD
NA13
PCRST
CSROM
IOWR
IORD
CSN
DACK
NICRST
PRQ
LS245
U3
GND 10
VCC 20
IOR
LS374
U6
GND 10
VCC 20
CLK
LS374
U2
GND 10
VCC 20
HM6264
U8
GND 14
VCC 20
LS373
U1
GND 10
VCC 20
ENB
DE
R3
4.7K
74S288
U4
GND 8
VCC 16
CSROM
U16/22

TL/F/9179–23


R1, R2 = 4.7 kΩ
U16/8
DP8390B
ADO
AD1
AD2
AD3
ADR1
AD5
AD6
AD7
ACK
WACK
RACK
CS
RST
A8
A9
A10
A11
A12
A13
NA13
U16/9
MWR
MRD
ADS0
SRD
SWR
PWR
PRD
COL
RXD
CRS
RXC
LBK
TXD
TXC
TXE
BSCK
BREQ
BACK
RDY
INT
PRQ
RA0
RA1
RA2
RA3
U11
GND 13
VCC 36
NC 16
J3/B14
J3/B13
J4/2,4,6,8
J4/10,12,14
Y1
OSC
GND 7
VCC 14
DP8391
SEL
CD+
CD-
RX+
RX-
TX+
TX-
CAP
X1
U9
GND 6
VCC 10,19
+5V
C19
.01UF
J1C
J2C
J3C
J4C
J5C
J6C
J1E
J2E
J3E
J4E
J5E
J6E
R3
R4
R4, R5 = 270Ω 1%
R9
R8
R7
R6
R9, R8, R7, R6 = 39Ω 1%
C6
.01UF
C6
.01UF
PE-64103
Note: For StarLAN, the DP8391 must be replaced with the StarLAN Adapter Card. See AN-498 for details.
RX+
RX-
TX+
TX-
8 POS
UNKEYED
MODULAR
PHONE JACK
RJ-45
(for StarLAN)
JMOD
DP8392
RR+
RR-
CDS
CD+
CD-
RX+
RX-
TX+
TX-
VEE
VEE
VEE
RX1
TXO
HBE
GND
U15
R14
1 kΩ 1%
E1 J2 E2
BNC
C4
.01UF
R17
1MOHM
¼ Watt
LOGIC GND
R10
R11
R12
R13
R10, R11, R12, R13 = 1.5 kΩ
C6
0.01UF
TX-
TX+
RX+
RX-
CD+
CD-
+12V
COM
B9
J7E
B1
LOGIC GND
1,4,6,11,14
ETHERNET AUI
CABLE CONNECTION
D-CONNECTOR
B3
+5V
J7C
C3
47UF
+5V
DC-DC
CONVERTER
2VP5U9
U10
ISOLATED VEE
-9V
ISOLATED GND
Note: All resistors are ¼ watt, 5% tolerance unless otherwise specified.

TL/F/9179-24

National Semiconductor

PRELIMINARY

DP83932B
Systems-Oriented Network Interface Controller

General Description

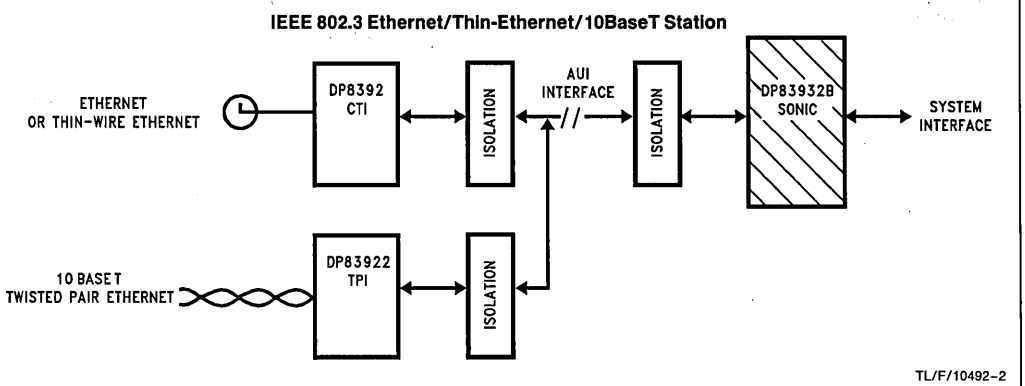
The SONIC™ (Systems-Oriented Network Interface Controller) is a second-generation Ethernet Controller designed to meet the demands of today's high-speed 32- and 16-bit systems. Its system interface operates with a 2-cycle DMA that typically consumes less than 3% of the bus bandwidth. Selectable bus modes provide both big and little endian byte ordering and a clean interface to standard microprocessors. The linked-list buffer management system of SONIC offers maximum flexibility in a variety of environments from PC-oriented adapters to high-speed motherboard designs. Furthermore, the SONIC integrates a fully-compatible IEEE 802.3 Encoder/Decoder (ENDEC) allowing for a simple 2-chip solution for Ethernet when the SONIC is paired with the DP8392 Coaxial Transceiver Interface or the DP83922 Twisted Pair Interface.

For increased performance, the SONIC implements a unique buffer management scheme to efficiently process receive and transmit packets in system memory. No intermediate packet copy is necessary. The receive buffer management uses three areas in memory for (1) allocating additional resources, (2) indicating status information, and (3) buffering packet data. During reception, the SONIC stores packets in the buffer area, then indicates receive status and control information in the descriptor area. The system allocates more memory resources to the SONIC by adding descriptors to the memory resource area. The transmit buffer management uses two areas in memory: one for indicating status and control information and the other for fetching packet data. The system can create a transmit queue allowing multiple packets to be transmitted from a single transmit command. The packet data can reside on any arbitrary byte boundary and can exist in several non-contiguous locations.

Features

- 32-bit non-multiplexed address and data bus
- High-speed, 2-cycle, interruptible DMA
- Linked-list buffer management maximizes flexibility
- Two independent 32-byte transmit and receive FIFOs
- Bus compatibility for all standard microprocessors
- Supports big and little endian formats
- Integrated IEEE 802.3 ENDEC
- Complete address filtering for up to 16 physical and/or multicast addresses
- 32-bit general-purpose timer
- Full-duplex loopback diagnostics
- Fabricated in low-power CMOS
- 132 PQFP package
- Full network management facilities support the 802.3 layer management standard
- Integrated support for bridge and repeater applications

System Diagram



TL/F/10492–2

Table of Contents

1.0 Functional Description

The SONIC *(Figure 1-1)* consists of an encoder/decoder (ENDEC) unit, media access control (MAC) unit, separate receive and transmit FIFOs, a system buffer management engine, and a user programmable system bus interface unit on a single chip. SONIC is highly pipelined providing maximum system level performance. This section provides a functional overview of SONIC.

1.1 IEEE 802.3 ENDEC UNIT

The ENDEC (Encoder/Decoder) unit is the interface between the Ethernet transceiver and the MAC unit. It provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Thin-Ethernet type local area networks. The ENDEC operations of SONIC are identical to the DP83910A CMOS Serial Network Interface device. During transmission, the ENDEC unit combines non-return-zero (NRZ) data from the MAC section and clock pulses into Manchester data and sends the converted data differentially to the transceiver. Conversely, during reception, an analog PLL decodes the Manchester data to NRZ format and receive clock. The ENDEC unit is a functionally complete Manchester encoder/decoder incorporating a balanced driver and receiver, on-board crystal oscillator, collision signal translator, and a diagnostic loopback. The features include:

- Compatible with Ethernet I and II, IEEE 802.3 10base5 and 10base2
- 10Mb/s Manchester encoding/decoding with receive clock recovery
- Requires no precision components
- Decodes Manchester data with up to 18 ns of jitter
- Loopback capability for diagnostics
- Externally selectable half or full step modes of operation at transmit output
- Squelch circuitry at the receive and collision inputs reject noise
- Connects to the transceiver (AUI) cable via external pulse transformer

1.1.1 ENDEC Operation

The primary function of the ENDEC unit *(Figure 1-2)* is to perform the encoding and decoding necessary for compatibility between the differential pair Manchester encoded data of the transceiver and the Non-Return-to-Zero (NRZ) serial data of the MAC unit data line. In addition to encoding and decoding the data stream, the ENDEC also supplies all the necessary special signals (e.g., collision detect, carrier sense, and clocks) to the MAC unit.

Manchester Encoder and Differential Output Driver: During transmission to the network, the ENDEC unit translates the NRZ serial data from the MAC unit into differential pair Manchester encoded data on the Coaxial Transceiver Interface (e.g., National's DP8392) transmit pair. To perform this operation the NRZ bit stream from the MAC unit is passed through the Manchester encoder block of the ENDEC unit. Once the bit stream is encoded, it is transmitted out differentially to the transmit differential pair through the transmit driver.

Manchester Decoder: During reception from the network, the differential receive data from the transceiver (e.g., the DP8392) is converted from Manchester encoded data into NRZ serial data and a receive clock, which are sent to the receive data and clock inputs of the MAC unit. To perform this operation the signal, once received by the differential receiver, is passed to the phase locked loop (PLL) decoder block. The PLL decodes the data and generates a data receive clock and a NRZ serial data stream to the MAC unit.

Special Signals: In addition to performing the Manchester encoding and decoding function, the ENDEC unit provides control and clocking signals to the MAC unit. The ENDEC sends a carrier sense (CRS) signal that indicates to the MAC unit that data is present from the network on the ENDEC's receive differential pair. The MAC unit is also provided with a collision detection signal (COL) that informs the MAC unit that a collision is taking place somewhere on the

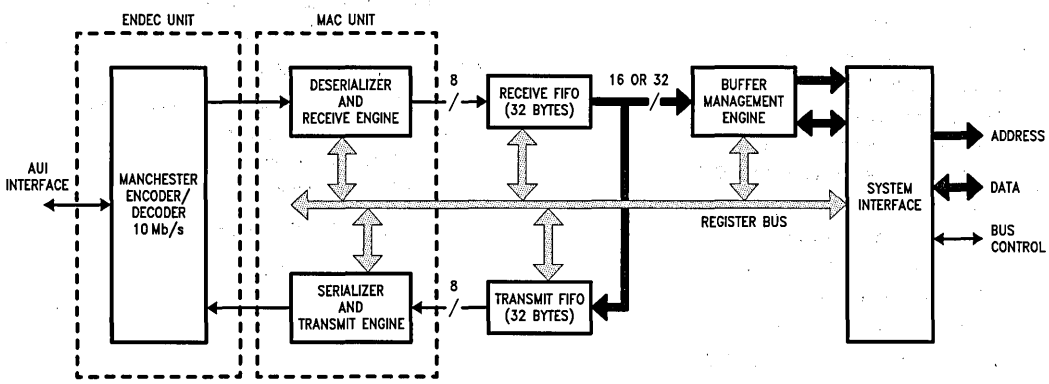


FIGURE 1-1. SONIC Block Diagram

1.0 Functional Description (Continued)

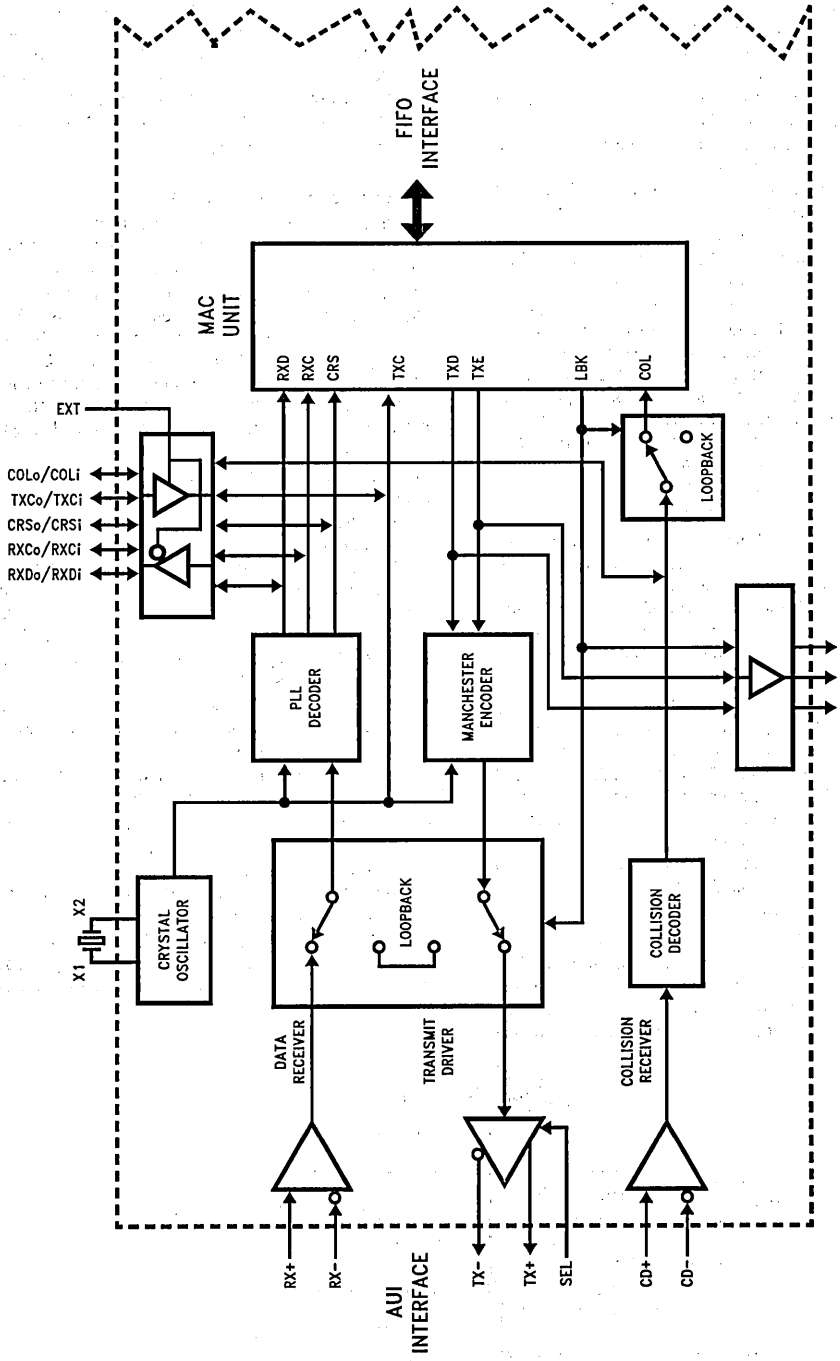


FIGURE 1-2. Block Diagram of Ethernet ENDEC

1.0 Functional Description (Continued)

network. The ENDEC section detects this when its collision receiver detects a 10 MHz signal on the differential collision input pair. The ENDEC also provides both the receive and transmit clocks to the MAC unit. The transmit clock is one half of the oscillator input. The receive clock is extracted from the input data by the PLL.

Oscillator: The oscillator generates the 10 MHz transmit clock signal for network timing. The oscillator is controlled by a parallel resonant crystal or by an external clock (see section 6.1.3). The 20 MHz output of the oscillator is divided by 2 to generate the 10 MHz transmit clock (TXC) for the MAC section. The oscillator provides an internal clock signal for the encoding and decoding circuits.

The signals provided to the MAC unit from the on-chip ENDEC are also provided as outputs to the user.

Loopback Functions: The SONIC provides three loopback modes. These modes allow loopback testing at the MAC, ENDEC and external transceiver level (see section 1.7 for details). It is important to note that when the SONIC is transmitting, the transmitted packet will always be looped back by the external transceiver. The SONIC takes advantage of this to monitor the transmitted packet. See the explanation of the Receive State Machine in section 1.2.1 for more information about monitoring transmitted packets.

1.1.2 Selecting An External ENDEC

An option is provided on SONIC to disable the on-chip ENDEC unit and use an external ENDEC. The internal IEEE 802.3 ENDEC can be bypassed by connecting the EXT pin to V_{CC} (EXT = 1). In this mode the MAC signals are redirected out from the chip, allowing an external ENDEC to be used. See section 5.2 for the alternate pin definitions.

1.2 MAC UNIT

The MAC (Media Access Control) unit performs the media access control functions for transmitting and receiving packets over Ethernet. During transmission, the MAC unit frames information from the transmit FIFO and supplies serialized data to the ENDEC unit. During reception, the incoming information from the ENDEC unit is deserialized, the frame checked for valid reception, and the data is transferred to the receive FIFO. Control and status registers on the SONIC govern the operation of the MAC unit.

1.2.1 MAC Receive Section

The receive section *(Figure 1-3)* controls the MAC receive operations during reception, loopback, and transmission. During reception, the deserializer goes active after detecting the 2-bit SFD (Start of Frame Delimiter) pattern (section 2.1). It then frames the incoming bits into octet boundaries and transfers the data to the 32-byte receive FIFO. Concurrently the address comparator compares the Destination Address Field to the addresses stored in the chip's CAM address registers (Content Addressable Memory cells). If a match occurs, the deserializer passes the remainder of the packet to the receive FIFO. The packet is decapsulated when the carrier sense input pin (CRS) goes inactive. At the end of reception the receive section checks the following:

— Frame alignment errors
— CRC errors
— Length errors (runt packets)

The appropriate status is indicated in the Receive Control register (section 4.3.3). In loopback operations, the receive section operates the same as during normal reception.

During transmission, the receive section remains active to allow monitoring of the self-received packet. The CRC checker operates as normal, and the Source Address field is compared with the CAM address entries. Status of the CRC check and the source address comparison is indicated by the PMB bit in the Transmit Control register (section 4.3.4). No data is written to the receive FIFO during transmit operations.

The receive section consists of the following blocks detailed below.

Receive State Machine (RSM): The RSM insures the proper sequencing for normal reception and self-reception during transmission. When the network is inactive, the RSM remains in an idle state continually monitoring for network activity. If the network becomes active, the RSM allows the deserializer to write data into the receive FIFO. During this state, the following conditions may prevent the complete reception of the packet.

— FIFO Overrun—The receive FIFO has been completely filled before the SONIC could buffer the data to memory.
— CAM Address Mismatch—The packet is rejected because of a mismatch between the destination address of the packet and the address in the CAM.
— Memory Resource Error—There are no more resources (buffers) available for buffering the incoming packets.
— Collision or Other Error—A collision occured on the network or some other error, such as a CRC error, occurred (this is true if the SONIC has been told to reject packets on a collision, or reject packets with errors).

If these conditions do not occur, the RSM processes the packet indicating the appropriate status in the Receive Control register.

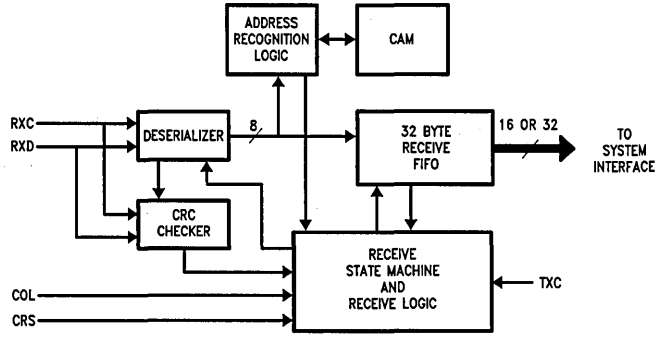


FIGURE 1-3. MAC Receiver

1.0 Functional Description (Continued)

During transmission of a packet from the SONIC, the external transceiver will always loop the packet back to the SONIC. The SONIC will use this to monitor the packet as it is being transmitted. The CRC and source address of the looped back packet are checked with the CRC and source address that were transmitted. If they do not match, an error bit is set in the status of the transmitted packet (see Packet Monitored Bad, PBM, in the Transmit Control Register, section 4.3.4). Data is not written to the receive FIFO during this monitoring process unless Transceiver Loopback mode has been selected (see section 1.7).

Receive Logic: The receive logic contains the command, control, and status registers that govern the operations of the receive section. It generates the control signals for writing data to the receive FIFO, processes error signals obtained from the CRC checker and the deserializer, activates the "packet reject" signal to the RSM for rejecting packets, and posts the applicable status in the Receive Control register.

Deserializer: This section deserializes the serial input data stream and furnishes a byte clock for the address comparator and receive logic. It also synchronizes the CRC checker to begin operation (after SFD is detected), and checks for proper frame alignment with respect to CRS going inactive at the end of reception.

Address Comparator: The address comparator latches the Destination Address (during reception or loopback) or Source Address (during transmission) and determines whether the address matches one of the entries in the CAM (Content Addressable Memory).

CRC Checker: The CRC checker calculates the 4-byte Frame Check Sequence (FCS) field from the incoming data stream and compares it with the last 4-bytes of the received packet. The CRC checker is active for both normal reception and self-reception during transmission.

Content Addressable Memory (CAM): The CAM contains 16 user programmable entries and 1 pre-programmed Broadcast address entry for complete filtering of received packets. The CAM can be loaded with any combination of Physical and Multicast Addresses (section 2.2). See section 4.1 for the procedure on loading the CAM registers.

1.2.2 MAC Transmit Section

The transmit section *(Figure 1-4)* is responsible for reading data from the transmit FIFO and transmitting a serial data stream onto the network in conformance with the IEEE 802.3 CSMA/CD standard. The Transmit Section consists of the following blocks.

Transmit State Machine (TSM): The TSM controls the functions of the serializer, preamble generator, and JAM generator. It determines the proper sequence of events that the transmitter follows under various network conditions. If no collision occurs, the transmitter prefixes a 62-bit preamble and 2-bit Start of Frame Delimiter (SFD) at the beginning of each packet, then sends the serialized data. At the end of the packet, an optional 4-byte CRC pattern is appended. If a collision occurs, the transmitter switches from transmitting data to sending a 4-byte Jam pattern to notify all nodes that a collision has occurred. Should the collision occur during the preamble, the transmitter waits for it to complete before jamming. After the transmission has completed, the transmitter writes status in the Transmit Control register (section 4.3.4).

Protocol State Machine: The protocol state machine assures that the SONIC obeys the CSMA/CD protocol. Before transmitting, this state machine monitors the carrier sense and collision signals for network activity. If another node(s) is currently transmitting, the SONIC defers until the network is quiet, then transmits after its Interframe Gap Timer (9.6 μs) has expired. The Interframe Gap time is divided into two portions. During the first 6.4 μs, network activity restarts the Interframe Gap timer. Beyond this time, however, network activity is ignored and the state machine waits the remaining 3.2 μs before transmitting. If the SONIC experiences a collision during a transmission, the SONIC switches from transmitting data to a 4-byte JAM pattern (4 bytes of all 1's), before ceasing to transmit. The SONIC then waits a random number of slot times (51.2 μs) determined by the *Truncated Binary Exponential Backoff Algorithm* before reattempting another transmission. In this algorithm, the number of slot times to delay before the nth retransmission is chosen to be a random integer r in the range of:

$$0 \leq r \leq 2^k$$

where $k = \min(n,10)$

If a collision occurs on the 16th transmit attempt, the SONIC aborts transmitting the packet and reports an "Excessive Collisions" error in the Transmit Control register.

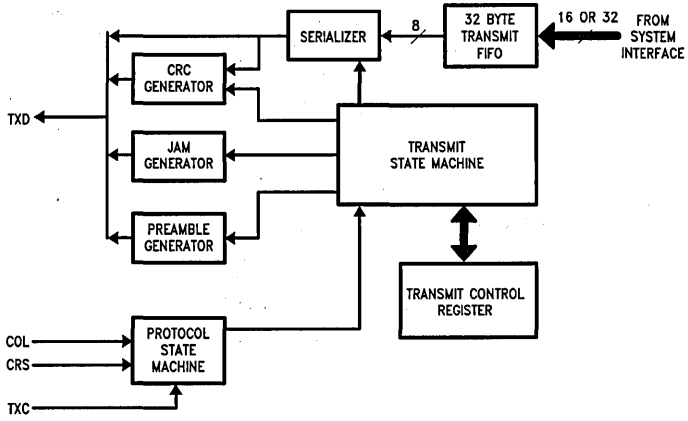


FIGURE 1-4. MAC Transmitter

1.0 Functional Description (Continued)

Serializer: After data has been written into the 32-byte transmit FIFO, the serializer reads byte wide data from the FIFO and sends a NRZ data stream to the Manchester encoder. The rate at which data is transmitted is determined by the transmit clock (TXC). The serialized data is transmitted after the SFD.

Preamble Generator: The preamble generator prefixes a 62-bit alternating "1,0" pattern and a 2-bit "1,1" SFD pattern at the beginning of each packet. This allows receiving nodes to synchronize to the incoming data. The preamble is always transmitted in its entirety even in the event of a collision. This assures that the minimum collision fragment is 96 bits (64 bits of normal preamble, and 4 bytes, or rather 32 bits, of the JAM pattern).

CRC Generator: The CRC generator calculates the 4-byte FCS field from the transmitted serial data stream. If enabled, the 4-byte FCS field is appended to the end of the transmitted packet (section 2.6).

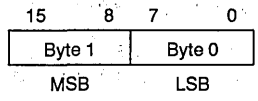
Jam Generator: The Jam generator produces a 4-byte pattern of all 1's to assure that all nodes on the network sense the collision. When a collision occurs, the SONIC stops transmitting data and enables the Jam generator. If a collision occurs during the preamble, the SONIC finishes transmitting the preamble before enabling the Jam generator (see Preamble Generator above).

1.3 DATA WIDTH AND BYTE ORDERING

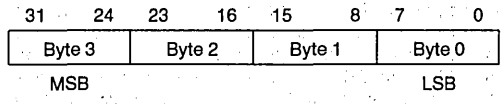
The SONIC can be programmed to operate with either 32-bit or 16-bit wide memory. The data width is configured during initialization by programming the DW bit in the Data Configuration Register (DCR, section 4.3.2). If the 16-bit data path is selected, data is driven on pins D15–D0. The SONIC also provides both Little Endian and Big Endian byte-ordering capability for compatibility with NSC/Intel or Motorola microprocessors respectively by selecting the proper level on the BMODE pin. The byte ordering is depicted at right.

Little Endian mode (BMODE = 0): The byte orientation for received and transmitted data in the Receive Buffer Area (RBA) and Transmit Buffer Area (TBA) of system memory is as follows:

16-Bit Word

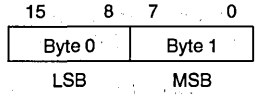


32-Bit Long Word

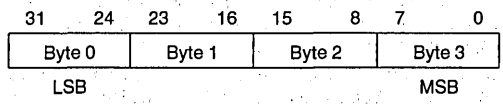


Big Endian mode (BMODE = 1): The byte orientation for received and transmitted data in the RBA and TBA is as follows:

16-Bit Word



32-Bit Long Word



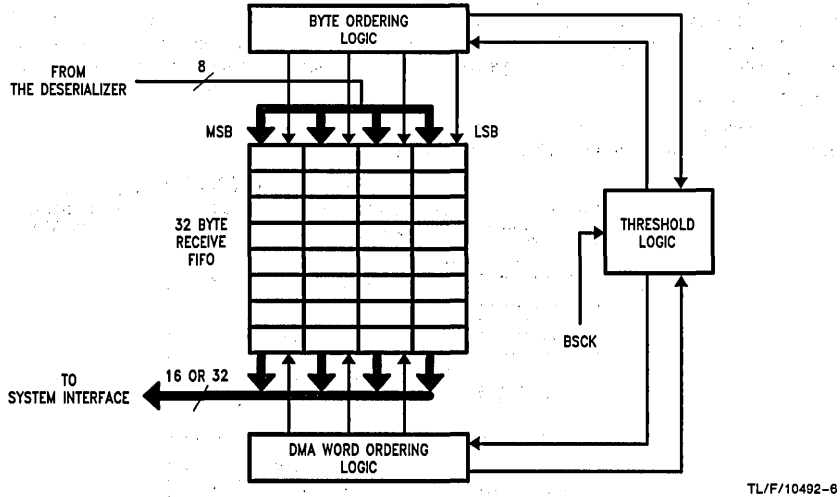


FIGURE 1-5. Receive FIFO

1.0 Functional Description (Continued)

1.4 FIFO AND CONTROL LOGIC

The SONIC incorporates two independent 32-byte FIFOs for transferring data to/from the system interface and from/to the network. The FIFOs, providing temporary storage of data, free the host system from the real-time demands on the network.

The way in which the FIFOS are emptied and filled is controlled by the FIFO threshold values and the Block Mode Select bits (BMS, section 4.3.2). The threshold values determine how full or empty the FIFOs can be before the SONIC will request the bus to get more data from memory or buffer more data to memory. When block mode is set, the number of bytes transferred is set by the threshold value. For example, if the threshold for the receive FIFO is 4 words, then the SONIC will always transfer 4 words from the receive FIFO to memory. If empty/fill mode is set, however, the number of bytes transferred is the number required to fill the transmit FIFO or empty the receive FIFO. More specific information about how the threshold affects reception and transmission of packets is discussed in sections 1.4.1 and 1.4.2 below.

1.4.1 Receive FIFO

To accommodate the different transfer rates, the receive FIFO *(Figure 1-5)* serves as a buffer between the 8-bit network (deserializer) interface and the 16/32-bit system interface. The FIFO is arranged as a 4-byte wide by 8 deep memory array (8-long words, or 32 bytes) controlled by three sections of logic. During reception, the Byte Ordering logic directs the byte stream from the deserializer into the FIFO using one of four write pointers. Depending on the selected byte-ordering mode, data is written either least significant byte first or most significant byte first to accommodate little or big endian byte-ordering formats respectively.

As data enters the FIFO, the Threshold Logic monitors the number of bytes written in from the deserializer. The programmable threshold (RFT1,0 in the Data Configuration Register) determines the number of words (or long words) written into the FIFO from the MAC unit before a DMA request for system memory occurs. When the threshold is reached, the Threshold Logic enables the Buffer Management Engine to read a programmed number of 16- or 32-bit words (depending upon the selected word width) from the FIFO and transfers them to the system interface (the system memory) using DMA. The threshold is reached when the number of bytes in the receive FIFO is greater than the value of the threshold. For example, if the threshold is 4 words (8 bytes), then the Threshold Logic will not cause the Buffer Management Engine to write to memory until there are more than 8 bytes in the FIFO.

The Buffer Management Engine reads either the upper or lower half (16 bits) of the FIFO in 16-bit mode or reads the complete long word (32 bits) in 32-bit mode. If, after the transfer is complete, the number of bytes in the FIFO is less then the threshold, then the SONIC is done. This is always the case when the SONIC is in empty/fill mode. If, however, for some reason (e.g. latency on the bus) the number of bytes in the FIFO is still greater than the threshold value, the Threshold Logic will cause the Buffer Management Engine to do a DMA request to write to memory again. This later case is usually only possible when the SONIC is in block mode.

When in block mode, each time the SONIC requests the bus, only a number of bytes equal to the threshold value will be transferred. The Threshold Logic continues to monitor the number of bytes written in from the deserializer and enables the Buffer Management Engine every time the threshold has been reached. This process continues until the end of the packet.

Once the end of the packet has been reached, the serializer will fill out the last word (16-bit mode) or long word (32-bit mode) if the last byte did not end on a word or long word boundary respectively. The fill byte will be 0FFh. Immediately after the last byte (or fill byte) in the FIFO, the received packets status will be written into the FIFO. The entire packet, including any fill bytes and the received packet status will be buffered to memory. When a packet is buffered to memory by the Buffer Management Engine, it is always taken from the FIFO in words or long words and buffered to memory on word (16-bit mode) or long word (32-bit mode) boundaries. Data from a packet cannot be buffered on odd byte boundaries for 16-bit mode, and odd word boundaries for 32-bit mode (see section 3.3). For more information on the receive packet buffering process, see section 3.4.

1.4.2 Transmit FIFO

Similar to the Receive FIFO, the Transmit FIFO *(Figure 1-6)* serves as a buffer between the 16/32-bit system interface and the network (serializer) interface. The Transmit FIFO is also arranged as a 4 byte by 8 deep memory array (8 long words or 32 bytes) controleld by three sections of logic. Before transmission can begin, the Buffer Management Engine fetches a programmed number of 16- or 32-bit words from memory and transfers them to the FIFO. The Buffer Management Engine writes either the upper or lower half (16 bits) into the FIFO for 16-bit mode or writes the complete long word (32 bits) during 32-bit mode.

The Threshold logic monitors the number of bytes as they are written into the FIFO. When the threshold has been reached, the Transmit Byte Ordering state machine begins reading bytes from the FIFO to produce a continuous byte stream for the serializer. The threshold is met when the number of bytes in the FIFO is greater than the value of the threshold. For example, if the transmit threshold is 4 words (8 bytes), the Transmit Byte Ordering state machine will not begin reading bytes from the FIFO until there are 9 or more bytes in the buffer. The Buffer Management Engine continues replenishing the FIFO until the end of the packet. It does this by making multiple DMA requests to the system interface. Whenever the number of bytes in the FIFO is equal to or less than the threshold value, the Buffer Management Engine will do a DMA request. If block mode is set, then after each request has been granted by the system, the Buffer Management Engine will transfer a number of bytes equal to the threshold value into the FIFO. If empty/fill mode is set, the FIFO will be completely filled in one DMA request.

Since data may be organized in big or little endian byte ordering format, the Transmit Byte Ordering state machine uses one of four read pointers to locate the proper byte within the 4 byte wide FIFO. It also determines the valid number of bytes in the FIFO. For packets which begin or end at odd bytes in the FIFO, the Buffer Management Engine writes extraneous bytes into the FIFO. The Transmit Byte Ordering state machine detects these bytes and only transfers the valid bytes to the serializer. The Buffer Management Engine can read data from memory on any byte boundary (see section 3.3). See section 3.5 for more information on transmit buffering.

1.0 Functional Description (Continued)

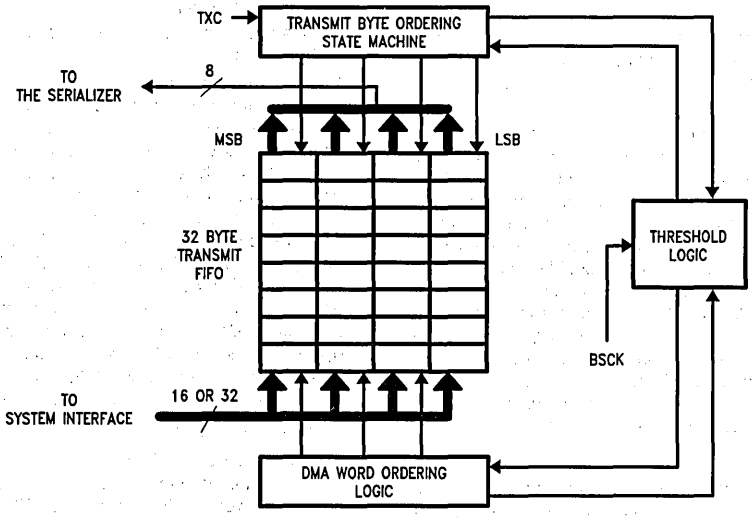


FIGURE 1-6. Transmit FIFO

1.5 STATUS AND CONFIGURATION REGISTERS

The SONIC contains a set of status/control registers for conveying status and control information to/from the host system. The SONIC uses these registers for loading commands generated from the system, indicating transmit and receive status, buffering data to/from memory, and providing interrupt control. Each register is 16 bits in length. See section 4.0 for a description of the registers.

1.6 BUS INTERFACE

The system interface *(Figure 1-7)* consists of the pins necessary for interfacing to a variety of buses. It includes the I/O drivers for the data and address lines, bus access control for standard microprocessors, ready logic for synchronous or asynchronous systems, slave access control, interrupt control, and shared-memory access control. The functional signal groups are shown in *Figure 1-7*. See section 5.0 for a complete description of the SONIC bus interface.

1.7 LOOPBACK AND DIAGNOSTICS

The SONIC furnishes three loopback modes for self-testing from the controller interface to the transceiver interface. The loopback function is provided to allow self-testing of the chip's internal transmit and receive operations. During loopback, transmitted packets are routed back to the receive section of the SONIC where they are filtered by the address recognition logic and buffered to memory if accepted. Transmit and receive status and interrupts remain active during loopback. This means that when using loopback, it is as if the packet was transmitted and received by two separate chips that are connected to the same bus and memory.

MAC Loopback: Transmitted data is looped back at the MAC. Data is not sent from the MAC to either the internal ENDEC or an external ENDEC (the external ENDEC interface pins will not be driven), hence, data is not transmitted from the chip. Even though the ENDEC is not used in MAC loopback, the ENDEC clock (an oscillator or crystal for the internal ENDEC or TXC for an external ENDEC) must be driven. Network activity, such as a collision, does not affect MAC loopback. CSMA/CD MAC protocol is not completely followed in MAC loopback.

ENDEC Loopback: Transmitted data is looped back at the ENDEC. If the internal ENDEC is used, data is switched from the transmit section of the ENDEC to the receive section (*Figure 1-2*). Data is not transmitted from the chip and the collision lines, CD±, are ignored, hence, network activity does not affect ENDEC loopback. The LBK signal from the MAC tells the internal ENDEC to go into loopback mode. If an external ENDEC is used, it should operate in loopback mode when the LBK signal is asserted. CSMA/CD MAC protocol is followed even though data is not transmitted from the chip.

Transceiver Loopback: Transmitted data is looped back at the external transceiver (which is always the case regardless of the SONIC's loopback mode). CSMA/CD MAC protocol is followed since data will be transmitted from the chip. This means that transceiver loopback is affected by network activity. The basic difference between Transceiver Loopback and normal, non-loopback, operations of the SONIC is that in Transceiver Loopback, the SONIC loads the receive FIFO and buffers the packet to memory. In normal operations, the SONIC only monitors the packet that is looped back by the transceiver, but does not fill the receive FIFO and buffer the packet.

1.7.1 Loopback Procedure

The following procedure describes the loopback operation.

1. Initialize the Transmit and Receive Area as described in sections 3.4 and 3.5.
2. Load one of the CAM address registers (see section 4.1), with the Destination Address of the packet if you are verifying the SONIC's address recognition capability.
3. Load one of the CAM address registers with the Source Address of the packet if it is different than the Destination Address to avoid getting a Packet Monitored Bad (PMB) error in the Transmit status (see section 4.3.4).

1.0 Functional Description (Continued)

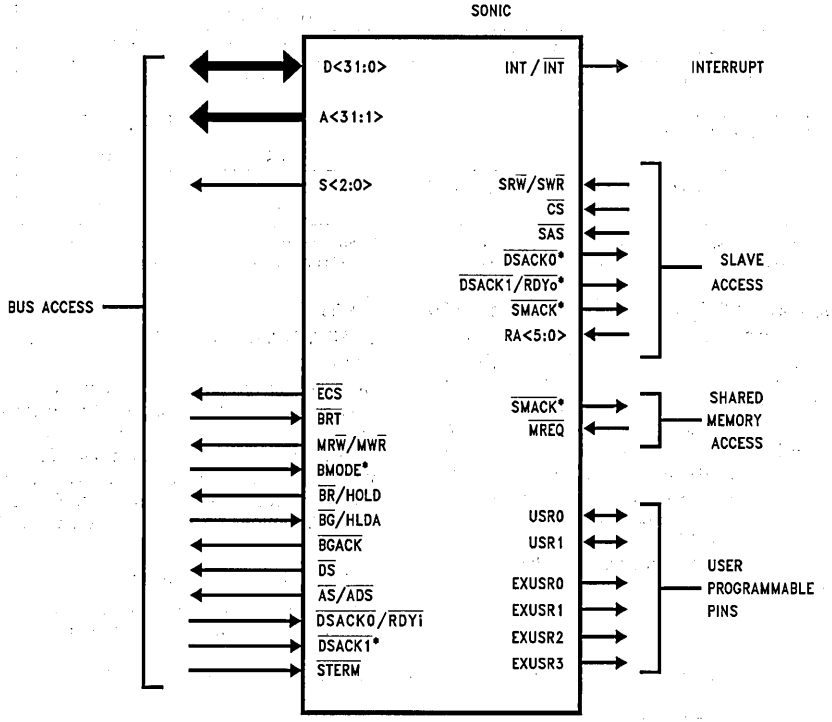
4. Program the Receive Control register with the desired receive filter and the loopback mode (LB1, LB0).
5. Issue the transmit command (TXP) and enable the receiver (RXEN) in the Command register.

The SONIC completes the loopback operation after the packet has been completely received (or rejected if there is an address mismatch). The Transmit Control and Receive Control registers treat the loopback packet as in normal operation and indicate status accordingly. Interrupts are also generated if enabled in the Interrupt Mask register.

Note: For MAC Loopback, only one packet may be queued for proper operation. This restriction occurs because the transmit MAC section, which does not generate an Interframe Gap time (IFG) between transmitted packets, does not allow the receive MAC section to update receive status. There are no restrictions for the other loopback modes.

1.8 NETWORK MANAGEMENT FUNCTIONS

The SONIC fully supports the Layer Management IEEE 802.3 standard to allow a node to monitor the overall performance of the network. These statistics are available on a per packet basis at the end of reception or transmission. In addition, the SONIC provides three tally counters to tabulate CRC errors, Frame Alignment errors, and missed packets. Table 1-1 shows the statistics indicated by the SONIC.



TL/F/10492–8

***Note:** DSACK0,1 are used for both Bus and Slave Access Control and are bidirectional. SMACK is used for both Slave access and shared memory access. The BMODE pin selects between NSC/Intel or Motorola type buses.

FIGURE 1-7. SONIC Bus Interface Signals

1.0 Functional Description (Continued)

TABLE 1-1. Network Management Statistics

Statistic	Register Used	Bits Used
Frames Transmitted OK	TCR (Note)	PTX
Single Collision Frames	(Note)	NC0–NC4
Multiple Collision Frames	(Note)	NC0–NC4
Collision Frames	(Note)	NC0–NC4
Frames with Deferred Transmissions	TCR (Note)	DEF
Late Collisions	TCR (Note)	OWC
Excessive Collisions	TCR (Note)	EXC
Excessive Deferral	TCR (Note)	EXD
Internal MAC Transmit Error	TCR (Note)	BCM,FU
Frames Received OK	RCR (Note)	PRX
Multicast Frames Received OK	RCR (Note)	MC
Broadcast Frames Received OK	RCR (Note)	BC
Frame Check Sequence Errors	CRCT RCR	All CRC
Alignment Errors	FAET RCR	All FAE
Frame Lost due to Internal MAC Receive Error	MPT ISR	All RFO

Note: The number of collisions and the contents of the Transmit Control register are posted in the TXpkt.status field (see section 3.5.1.2). The contents of the Receive Control register are posted in the RXpkt.status field (see section 3.4.3.1).

2.0 Transmit/Receive IEEE 802.3 Frame Format

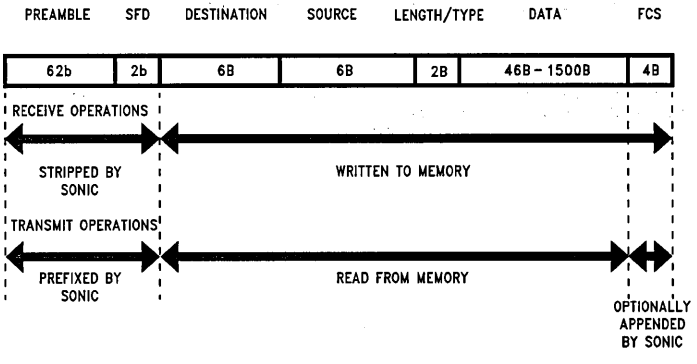
A standard IEEE 802.3 packet *(Figure 2-1)* consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data and Frame Check Sequence (FCS). The typical format is shown in *Figure 2-1*. The packets are Manchester encoded and decoded by the ENDEC unit and transferred serially to/from the MAC unit using NRZ data with a clock. All fields are of fixed length except for the data field. The SONIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

2.1 PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the ENDEC to acquire bit synchronization with an incoming packet. When transmitted, each packet contains 62 bits of an alternating 1,0 preamble. Some of this preamble may be lost as the packet travels through the network. Byte alignment is performed when the Start of Frame Delimiter (SFD) pattern, consisting of two consecutive 1's, is detected.

2.2 DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted pack-



Note: B = bytes
b = bits

TL/F/10492–9

FIGURE 2-1. IEEE 802.3 Packet Structure

2.0 Transmit/Receive IEEE 802.3 Frame Format (Continued)

ets from reaching a node. There are three types of address formats supported by the SONIC: Physical, Multicast, and Broadcast.

Physical Address: The physical address is a unique address that corresponds only to a single node. All physical addresses have the LSB of the first byte of the address set to "0". These addresses are compared to the internally stored CAM (Content Addressable Memory) address entries. All bits in the destination address must match an entry in the CAM in order for the SONIC to accept the packet.

Multicast Address: Multicast addresses, which have the LSB of the first byte of the address set to "1", are treated similarly as Physical addresses, i.e., they must match an entry in the CAM. This allows perfect filtering of Multicast packet's and eliminates the need for a hashing algorithm for mapping Multicast packets.

Broadcast Address: If the address consists of all 1's, it is a Broadcast address, indicating that the packet is intended for all nodes.

The SONIC also provides a promiscuous mode which allows reception of all physical address packets. Physical, Multicast, Broadcast, and promiscuous address modes can be selected via the Receive Control register.

2.3 SOURCE ADDRESS

The source address is the physical address of the sending node. Source addresses cannot be multicast or broadcast addresses. This field must be passed to the SONIC's transmit buffer from the system software. During transmission, the SONIC compares the Source address with its internal CAM address entries before monitoring the CRC of the self-received packet. If the source address of the packet transmitted does not match a value in the CAM, the packet monitored bad flag (PMB) will be set in the transmit status field of the transmit descriptor (see sections 3.5.1.2 and 4.3.4). The SONIC does not provide Source Address insertion. However, a transmit descriptor fragment, containing only the Source Address, may be created for each packet. (See section 3.5.1.)

2.4 LENGTH/TYPE FIELD

For IEEE 802.3 type packets, this field indicates the number of bytes that are contained in the data field of the packet. For Ethernet I and II networks, this field indicates the type of packet. The SONIC does not operate on this field.

2.5 DATA FIELD

The data field has a variable octet length ranging from 46 to 1500 bytes as defined by the Ethernet specification. Messages longer than 1500 bytes need to be broken into multiple packets for IEEE 802.3 networks. Data fields shorter than 46 bytes require appending a pad to bring the complete frame length to 64 bytes. If the data field is padded, the number of valid bytes are indicated in the length field. The SONIC does not append pad bytes for short packets during transmission, nor check for oversize packets during reception. However, the user's driver software can easily append the pad by lengthening the TXpkt.pkt_size field and TXpkt.frag_size field(s) to at least 64 bytes (see section 3.5.1). While the Ethernet specification defines the maximum number of bytes in the data field the SONIC can transmit and receive packets up to 64k bytes.

2.6 FCS FIELD

The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of error-free packets. During reception, an error-free packet results in a specific pattern in the CRC generator. The AUTODIN II ($X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 + 1$) polynomial is used for the CRC calculations. The SONIC may optionally append the CRC sequence during transmission, and checks the CRC both during normal reception and self-reception during a transmission (see section 1.2.1).

2.7 MAC (MEDIA ACCESS CONTROL) CONFORMANCE

The SONIC is designed to be compliant to the IEEE 802.3 MAC Conformance specification. The SONIC implements most of the MAC functions in silicon and provides hooks for the user software to handle the remaining functions. The MAC Conformance specifications are summarized in Table 2-1.

TABLE 2-1. MAC Conformance Specifications

Conformance Test Name	Support By		
	SONIC	User Driver Software	Notes
Minimum Frame Size	X		
Maximum Frame Size	X	X	1
Address Generation	X	X	2
Address Recognition	X		
Pad Length Generation	X	X	3
Start Of Frame Delimiter	X		
Length Field	X		
Preamble Generation	X		
Order of Bit Transmission	X		
Inconsistent Frame Length	X	X	1
Non-Integral Octet Count	X		
Incorrect Frame Check Sequence	X		
Frame Assembly	X		
FCS Generation and Insertion	X		
Carrier Deference	X		
Interframe Spacing	X		
Collision Detection	X		
Collision Handling	X		
Collision Backoff and Retransmission	X		
FCS Validation	X		
Frame Disassembly	X		
Back-to-Back Frames	X		
Flow Control	X		
Attempt Limit	X		
Jam Size (after SFD)	X		
Jam Size (in Preamble)	X		

Note 1: The SONIC provides the byte count of the entire packet in the RXpkt.byte_count (see section 3.4.3). The user's driver software may perform further filtering of the packet based upon the byte count.

Note 2: The SONIC does not provide Source Address insertion; however, a transmit descriptor fragment, containing only the Source Address, may be created for each packet. See section 3.5.1.

Note 3: The SONIC does not provide Pad generation; however, the user's driver software can easily append the Pad by lengthening the TXpkt.pkt_size field and TXpkt.frag_size field(s) to at least 64 bytes. See section 3.5.1.

3.0 Buffer Management

3.1 BUFFER MANAGEMENT OVERVIEW

The SONIC's buffer management scheme is based on separate buffers and descriptors (*Figures 3-2* and *3-11*). Packets that are received or transmitted are placed in buffers called the Receive Buffer Area (RBA) and the Transmit Buffer Area (TBA). The system keeps track of packets in these buffers using the information in the Receive Descriptor Area (RDA) and the Transmit Descriptor Area (TDA). A single (TDA) points to a single TBA, but multiple RDAs can point to a single RBA (one RDA per packet in the buffer). The Receive Resource Area (RRA), which is another form of descriptor, is used to keep track of the actual buffer.

When packets are transmitted, the system sets up the packets in one or more TBAs with a TDA pointing to each TBA. There can only be one packet per TBA/TDA pair. A single TBA, however, may be made up of several fragments of data dispersed in memory. There is one TDA pointing to each TBA which specifies information about the buffer's size, location in memory, number of fragments and status after transmission. The TDAs are linked together in a linked list. The system causes the SONIC to transmit the packets by passing the first TDA to the SONIC and issuing the transmit command.

Before a packet can be received, an RBA and RDA must be set up by the system. RDAs are made up as a linked list similar to TDAs. An RDA is not linked to a particular RBA, though. Instead, an RDA is linked specifically to a packet after it has been buffered into an RBA. More than one packet can be buffered into the same RBA, but each packet gets its own RDA. A received packet can not be scattered into fragments. The system only needs to tell the SONIC where the first RDA and where the RBAs are. Since an RDA never specifically points to an RBA, the RRA is used to keep track of the RBAs. The RRA is a circular queue of pointers and buffer sizes (not a linked list). When the SONIC receives a packet, it is buffered into a RBA and a RDA is written to so that it points to and describes the new packet. If the RBA does not have enough space to buffer the next packet, a new RBA is obtained from the RRA.

3.2 DESCRIPTOR AREAS

Descriptors are the basis of the buffer management scheme used by the SONIC. A RDA points to a received packet within a RBA, a RRA points to a RBA and a TDA points to a TBA which contains a packet to be transmitted. The conventions and registers used to describe these descriptors are discussed in the next three sections.

3.2.1 Naming Convention for Descriptors

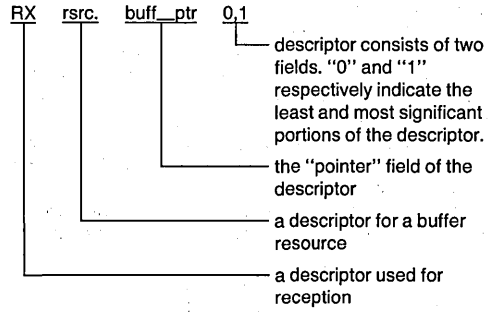
The fields which make up the descriptors are named in a consistent manner to assist in remembering the usage of each descriptor. Each descriptor name consists of three components in the following format.

[RX/TX][descriptor name].[field]

The first two capital letters indicate whether the descriptor is used for transmission (TX) or reception (RX), and is then followed by the descriptor name having one of two names.

rsrc = Resource descriptor
pkt = Packet descriptor

The last component consists of a field name to distinguish it from the other fields of a descriptor. The field name is separated from the descriptor name by a period. (".") An example of a descriptor is shown below.



3.2.2 Abbreviations

The abbreviations at right are used to describe the SONIC registers and data structures in memory. The "0" and "1" in the abbreviations indicate the least and most significant portions of the registers or descriptors. Table 3-1 lists the naming convention abbreviations for descriptors.

3.2.3 Buffer Management Base Addresses

The SONIC uses three areas in memory to store descriptor information: the Transmit Descriptor Area (TDA), Receive Descriptor Area (RDA), and the Receive Resource Area (RRA). The SONIC accesses these areas by concatenating a 16-bit base address register with a 16-bit offset register. The base address register supplies a fixed upper 16 bits of address and the offset registers provide the lower 16 bits of address. The base address registers are the Upper Transmit Descriptor Address (UTDA), Upper Receive Descriptor Address (URDA), and the Upper Receive Resource Address (URRA) registers. The corresponding offset registers are shown below.

Upper Address Registers	Offset Registers
URRA	RSA,REA,RWP,RRP
URDA	CRDA
UTDA	CTDA

See Table 3-1 for definition of register mnemonics.

Figure 3-1 shows an example of the Transmit Descriptor Area and the Receive Descriptor Area being located by the UTDA and URDA registers. The descriptor areas, RDA, TDA, and RRA are allowed to have the same base address. i.e., URRA = URDA = UTDA. Care, however, must be taken to prevent these areas from overwriting each other.

3.0 Buffer Management (Continued)

TABLE 3-1. Descriptor Abbreviations

TRANSMIT AND RECEIVE AREAS	
RRA	Receive Resource Area
RDA	Receive Descriptor Area
RBA	Receive Buffer Area
TDA	Transmit Descriptor Area
TBA	Transmit Buffer Area
BUFFER MANAGEMENT REGISTERS	
RSA	Resource Start Area Register
REA	Resource End Area Register
RRP	Resource Read Pointer Register
RWP	Resource Write Pointer Register
CRDA	Current Receive Descriptor Address Register
CRBA0,1	Current Receive Buffer Address Register
TCBA0,1	Temporary Current Buffer Address Register
RBWC0,1	Remaining Buffer Word Count Register
TRBWC0,1	Temporary Remaining Buffer Word Count Register
EOBC	End of Buffer Count Register
TPS	Transmit Packet Size Register
TSA0,1	Transmit Start Address Register
CTDA	Current Transmit Descriptor Address Register

BUFFER MANAGEMENT REGISTERS (Continued)	
TFC	Transmit Fragment Count Register
TFS	Transmit Fragment Size Register
UTDA	Upper Transmit Descriptor Address Register
URRA	Upper Receive Resource Address Register
URDA	Upper Receive Descriptor Address Register
TRANSMIT AND RECEIVE DESCRIPTORS	
RXrsrc.buff_ptr0,1	Buffer Pointer Field in the RRA
RXrsrc.buff_wc0,1	Buffer Word Count Fields in the RRA
RXpkt.status	Receive Status Field in the RDA
RXpkt.byte_count	Packet Byte Count Field in the RDA
RXpkt.buff_ptr0,1	Buffer Pointer Fields in the RDA
RXpkt.link	Receive Descriptor Link Field in RDA
RXpkt.in_use	"In Use" Field in RDA
TXpkt.frag_count	Fragment Count Field in TDA
TXpkt.pkt_size	Packet Size Field in TDA
TXpkt.pkt_ptr0,1	Packet Pointer Fields in TDA
TXpkt.frag_size	Fragment Size Field in TDA
TXpkt.link	Transmit Descriptor Link Field in TDA

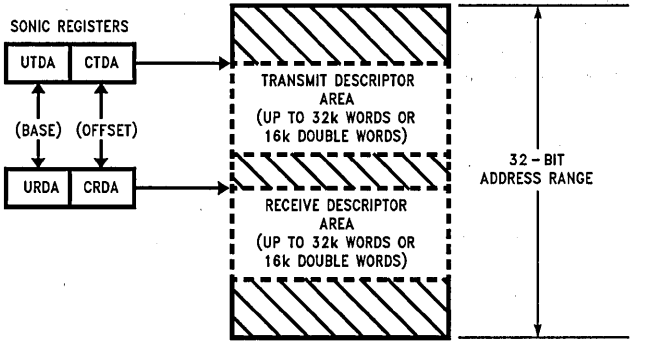


FIGURE 3-1. Transmit and Receive Descriptor Area Pointers

3.0 Buffer Management (Continued)

3.3 DESCRIPTOR DATA ALIGNMENT

All fields used by descriptors (RXpkt.xxx, RXrsrc.xxx, and TXpkt.xxx) are word quantities (16-bit) and must be aligned to word boundaries (A0=0) for 16-bit memory and to long word boundaries (A1,A0=0,0) for 32-bit memory. The Receive Buffer Area (RBA) must also be aligned to a word boundary in 16-bit mode and a long word boundary in 32-bit mode. The fragments in the Transmit Buffer Area (TBA), however, may be aligned on any arbitrary byte boundary.

3.4 RECEIVE BUFFER MANAGEMENT

The Receive Buffer Management operates on three areas in memory into which data, status, and control information are written during reception *(Figure 3-2)*. These three areas must be initialized (section 3.4.4) before enabling the receiver (setting the RXEN bit in the Command register). The receive resource area (RRA) contains descriptors that locate receive buffer areas in system memory. These descriptors are denoted by R1, R2, etc. in *Figure 3-2*. Packets (denoted by P1, P2, etc.) can then be buffered into the corresponding RBAs. Depending on the size of each buffer area and the size of the packet(s), multiple or single packets are buffered into each RBA. The receive descriptor area (RDA) contains status and control information for each packet (D1, D2, etc. in *Figure 3-2*) corresponding to each received packet (D1 goes with P1, D2 with P2, etc.).

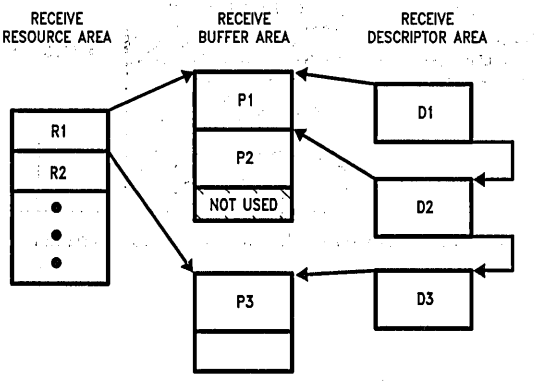
When a packet arrives, the address recognition logic checks the address for a Physical, Multicast, or Broadcast match and if the packet is accepted, the SONIC buffers the packet contiguously into the selected Receive Buffer Area (RBA). Because of the previous end-of-packet processing, the SONIC assures that the complete packet is written into a single contiguous block. When the packet ends, the SONIC writes the receive status, byte count, and location of the packet into the Receive Descriptor Area (RDA). The SONIC then updates its pointers to locate the next available descriptor and checks the remaining words available in the RBA. If sufficient space remains, the SONIC buffers the next packet immediately after the previous packet. If the current buffer is out of space the SONIC fetches a Resource descriptor from the Receive Resource Area (RRA) acquiring an additional buffer that has been previously allocated by the system.

3.4.1 Receive Resource Area (RRA)

As buffer memory is consumed by the SONIC for storing data, the Receive Resource Area (RRA) provides a mechanism that allows the system to allocate additional buffer space for the SONIC. The system loads this area with resource descriptors that the SONIC, in turn, reads as its current buffer space is used up. Each resource descriptor consists of a 32-bit buffer pointer locating the starting point of the RBA and a 32-bit Word Count that indicates the size of the buffer in words (2 bytes per word). The buffer pointer and word count are contiguously located using the format shown in *Figure 3-3* with each component composed of 16-bit fields. The SONIC stores this information internally and concatenates the corresponding fields to create 32-bit long words for the buffer pointer and word count. Note that in 32-bit mode the upper word (D<31:16>) is not used by the SONIC. This area may be used for other purposes since the SONIC never writes into the RRA.

The SONIC organizes the RRA as a circular queue for efficient processing of descriptors. Four registers define the RRA. The first two, the Resource Start Area (RSA) and the Resource End Area (REA) registers, determine the starting and ending locations of the RRA, and the other two registers update the RRA. The system adds descriptors at the address specified by the Resource Write Pointer (RWP), and the SONIC reads the next descriptor designated by the Resource Read Pointer (RRP). The RRP is advanced 4 words in 16-bit mode (4 long words in 32-bit mode) after the SONIC finishes reading the RRA and automatically wraps around to the beginning of the RRA once the end has been reached. When a descriptor in the RRA is read, the RXrsc.buff_pt0,1 is loaded into the CRBA0,1 registers and the RXrsc.buff_wc0,1 is loaded into the RBWC0,1 registers.

The alignment of the RRA is confined to either word or long word boundaries, depending upon the data width mode. In 16-bit mode, the RRA must be aligned to a word boundary (A0 is always zero) and in 32-bit mode, the RRA is aligned to a long word boundary (A0 and A1 are always zero).



TL/F/10492–11

FIGURE 3-2. Overview of Receive Buffer Management

3.0 Buffer Management (Continued)

3.4.2 Receive Buffer Area (RBA)

The SONIC stores the actual data of a received packet in the RBA. The RBAs are designated by the resource descriptors in the RRA as described above. The RXrsrc.buff_wc0,1 fields of the RRA indicate the length of the RBA. When the SONIC gets a RBA from the RRA, the RXrsrc.buff_wc0,1 values are loaded into the Remaining Buffer Word Count registers (RBWC0,1). These registers keep track of how much space (in words) is left in the buffer. When a packet is buffered in a RBA, it is buffered contiguously (the SONIC will not scatter a packet into multiple buffers or fragments). Therefore, if there is not enough space left in a RBA after buffering a packet to buffer at least one more maximum sized packet (the maximum legal sized packet expected to be received from the network), a new buffer must be acquired. The End of Buffer Count (EOBC) register is used to tell the SONIC the maximum packet size that the SONIC will need to buffer.

3.4.2.1 End of Buffer Count (EOBC)

The EOBC is a boundary in the RBA based from the bottom of the buffer. The value written into the EOBC is the maximum expected size (in words) of the network packet that the SONIC will have to buffer. This word count creates a line in the RBA that, when crossed, causes the SONIC to fetch a new RBA resource from the RRA.

Note: The EOBC is a word count, not a byte count. Also, the value programmed into EOBC must be a double word (32-bit) quantity when the SONIC is in 32-bit mode (e.g. in 32-bit mode, EOBC should be set to 760 words, not 759 words even though the maximum size of an IEEE 802.3 packet is 759 words).

3.4.2.2 Buffering the Last Packet in an RBA

At the start of reception, the SONIC stores the packet beginning at the Current Receive Buffer Address (CRBA0,1) and continues until the reception is complete. Concurrent with reception, the SONIC decrements the Remaining Buffer Word Count (RBWC0,1) by one in 16-bit mode or by two in 32-bit mode. At the end of reception, if the packet has crossed the EOBC boundary, the SONIC knows that the next packet might not fit in the RBA. This check is done by comparing the RBWC0,1 registers with the EOBC. If RBWC0,1 is less than the EOBC (the last packet buffered has crossed the EOBC boundary), the SONIC fetches the next resource descriptor in the RRA. If RBWC0,1 is greater than or equal to the EOBC (the EOBC boundary has not been crossed) the next packet reception continues at the present location pointed to by CRBA0,1 in the same RBA. *Figure 3-4* illustrates the SONIC's actions for (1) RBWC0,1 ≥ EOBC and (2) RBWC0,1 < EOBC. See section 3.4.4.4 for specific information about setting the EOBC.

Note: It is important that the EOBC boundary be "crossed." In other words, case #1 in *Figure 3-4* must exist before case #2 exists. If case #2 occurs without case #1 having occurred first, the test for RBWC0,1 < EOBC will not work properly and the SONIC will not fetch a new buffer. The result of this will be a buffer overflow (RBAE in the Interrupt Status Register, section 4.3.6).

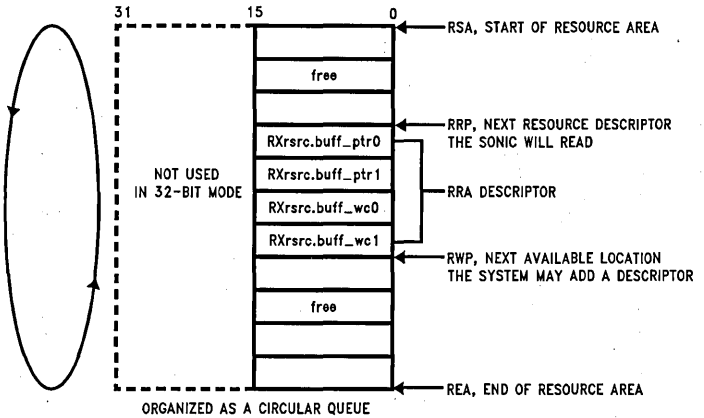
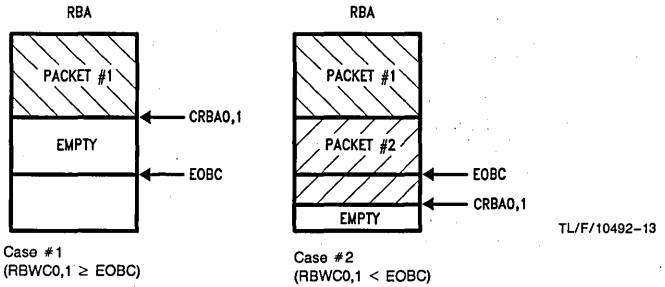


FIGURE 3-3. Receive Resource Area Format



Case #1: SONIC buffers next packet in same RBA.
Case #2: SONIC detects an exhausted RBA and will buffer the next packet in another RBA.

FIGURE 3-4. Receive Buffer Area

3.0 Buffer Management (Continued)

3.4.3 Receive Descriptor Area (RDA)

After the SONIC buffers a packet to memory, it writes 6 words of status and control information into the RDA and then reads the link field to proceed to the next receive descriptor. In 32-bit mode the upper word, D<31:16>, is not used. This unused area in memory should not be used for other purposes since the SONIC may still write into these locations. Each receive descriptor consists of the following sections (*Figure 3-5*).

receive status: indicates status of the received packet. The SONIC writes the Receive Control register into this field. *Figure 3-6* shows the receive status format. This field is loaded from the contents of the Receive Control register. Note that ERR, RNT, BRD, PRO, and AMC are configuration bits and are programmed during initialization. See section 4.3.3 for the description of the Receive Control register.

15	14	13	12	11	10	9	8
ERR	RNT	BRD	PRO	AMC	LB1	LB0	MC

7	6	5	4	3	2	1	0
BC	LPKT	CRS	COL	CRCR	FAER	LBK	PRX

FIGURE 3-6. Receive Status Format

byte count: gives the length of the complete packet from the start of Destination Address to the end of FCS.

packet pointer: a 32-bit pointer that locates the packet in the RBA. The SONIC writes the contents of the CRBA0,1 registers into this field.

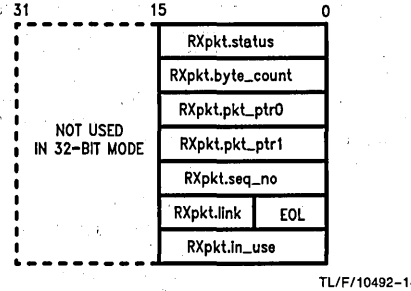
sequence numbers: this field displays the contents of two 8-bit counters (modulo 256) that sequence the RBAs used and the packets buffered. These counters assist the system in determining when an RBA has been completely processed. The sequence numbers allow the system to tally the packets that have been processed within a particular RBA. There are two sequence numbers that describe a packet: the RBA Sequence Number and the Packet Sequence Number. When a packet is buffered to memory, the SONIC maintains a single RBA Sequence Number for all packets in an RBA and sequences the Packet Number for succeeding packets in the RBA. When the SONIC uses the next RBA, it increments the RBA Sequence Number and clears the Packet Sequence Number. The RBA's sequence counter is not incremented when the read RRA command is issued in the Command register. The format of the Receive Sequence Numbers are shown in *Figure 3-7*. These counters are reset during hardware reset or by writing zero to them.

15 — 8	7 — 0
RBA Sequence Number (modulo 256)	Packet Sequence Number (Modulo 256)

FIGURE 3-7. Receive Sequence Number Format

receive link field: a 15-bit pointer (A15–A1) that locates the next receive descriptor. The LSB of this field is the End Of List (EOL) bit, and indicates the last descriptor in the list. (Initialized by the system.)

in use field: this field provides a handshake between the system and the SONIC to indicate the ownership of the descriptor. When the system avails a descriptor to the SONIC, it writes a non-zero value into this field. The SONIC, in turn, sets this field to all "0's" when it has finished processing the descriptor. (That is, when the CRDA register has advanced to the next receive descriptor.) Generally, the SONIC releases control after writing the status and control information into the RDA. If, however, the SONIC has reached the last descriptor in the list, it maintains ownership of the descriptor until the system has appended additional descriptors to the list. The SONIC then relinquishes control after receiving the next packet. (See section 3.4.6.1 for details on when the SONIC writes to this field). The receive packet descriptor format is shown in *Figure 3-5*.



TL/F/10492–14

FIGURE 3-5. Receive Descriptor Format

3.4.4 Receive Buffer Management Initialization

The Receive Resource, Descriptor, and Buffer areas (RRA, RDA, RBA) in memory and the appropriate SONIC registers must be properly initialized before the SONIC begins buffering packets. This section describes the initialization process.

3.4.4.1 Initializing The Descriptor Page

All descriptor areas (RRA, RDA, and TDA) used by the SONIC reside within areas up to 32k (word) or 16k (long word) pages. This page may be placed anywhere within the 32-bit address range by loading the upper 16 address lines into the UTDA, URDA, and URRA registers.

3.4.4.2 Initializing The RRA

The initialization of the RRA consists of loading the four SONIC RRA registers and writing the resource descriptor information to memory.

The RRA registers are loaded with the following values.

Resource Start Area (RSA) register: The RSA is loaded with the lower 16-bit address of the beginning of the RRA.

Resource End Area (REA) register: The REA is loaded with the lower 16-bit address of the end of the RRA. The end of the RRA is defined as the address of the last RXrsrc.ptr0 field in the RRA plus 4 words in 16-bit mode or 4 long words in 32-bit mode (*Figure 3-3*).

Resource Read Pointer (RRP) register: The RRP is loaded with the lower 16-bit address of the first resource descriptor the SONIC reads.

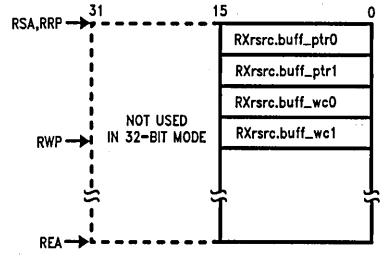
Resource Write Pointer (RWP) register: The RWP is loaded with the lower 16-bit address of the next vacant location where a resource descriptor will be placed by the system.

Note: The RWP register must only point to either (1) the RXrsrc.ptr0 field of one of the RRA Descriptors, (2) the memory address that the RSA points to (the start of the RRA), or (3) the memory address that the REA points to (the end of the RRA). When the RWP = RRP comparison is made, it is performed after the complete RRA descriptor has been read and not during the fetch. Failure to set the RWP to any of the above values prevents the RWP = RRP comparison from ever becoming true.

3.0 Buffer Management (Continued)

All RRA registers are concatenated with the URRA register for generating the full 32-bit address.

The resource descriptors that the system writes to the RRA consists of four fields: (1) RXrsrc.buff_ptr0, (2) RXrsrc.buff_ptr1, (3) RXrsrc.buff_wc0, and (4) RXrsrc.buff_wc1. The fields must be contiguous (they cannot straddle the end points) and are written in the order shown in *Figure 3-8*. The "0" and "1" in the descriptors denote the least and most significant portions for the Buffer Pointer and Word Count. The first two fields supply the 32-bit starting location of the Receive Buffer Area (RBA), and the second two define the number of 16-bit words that the RBA occupies. Note that two restrictions apply to the Buffer Pointer and Word Count. First, in 32-bit mode, since the SONIC always writes long words, an even count must be written to RXrsrc.buff_wc0. Second, the Buffer Pointer must either be pointing to a word boundary in 16-bit mode (A0=0) or a long word boundary in 32-bit mode (A0,A1=0,0). Note also that the descriptors must be properly aligned in the RRA as discussed in section 3.3.



TL/F/10492–15

FIGURE 3-8. RRA Initialization

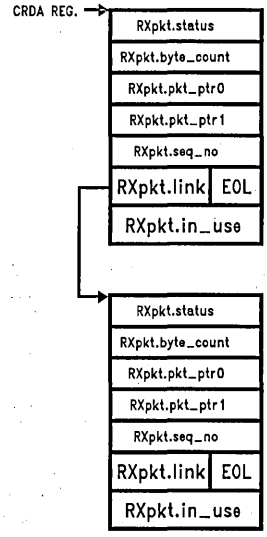
After configuring the RRA, the RRA Read command (setting RRRA bit in the Command register) may be given. This command causes the SONIC to read the RRA descriptor in a single block operation, and load the following registers (see section 4.2 for register mnemonics):

CRBA0 register ← RXrsrc.buff_ptr0
CRBA1 register ← RXrsrc.buff_ptr1
RBWC0 register ← RXrsrc.buff_wc0
RBWC1 register ← RXrsrc.buff_wc1

When the command has completed, the RRRA bit in the Command register is reset to "0". Generally this command is only issued during initialization. At all other times, the RRA is automatically read as the SONIC finishes using an RBA.

3.4.4.3 Initializing The RDA

To accept multiple packets from the network, the receive packet descriptors must be linked together via the RXpkt.link fields. Each link field must be written with a 15-bit (A15–A1) pointer to locate the beginning of the next descriptor in the list. The LSB of the RXpkt.link field is the End of List (EOL) bit and is used to indicate the end of the descriptor list. EOL = 1 for the last descriptor and EOL = 0 for the first or middle descriptors. The RXpkt.in_use field indicates whether the descriptor is owned by the SONIC. The system writes a non-zero value to this field when the descriptor is available, and the SONIC writes all "0's" when it finishes using the descriptor. At startup, the Current Receive Descriptor Address (CRDA) register must be loaded with the address of the first RXpkt.status field in order for the SONIC to begin receive processing at the first descriptor. An example of two descriptors linked together is shown in *Figure 3-9*. The fields initialized by the system are displayed in **bold type.** The other fields are written by the SONIC after a packet is accepted. The RXpkt.in_use field is first written by the system, and then by the SONIC. Note that the descriptors must be aligned properly as discussed in section 3.3. Also note that the URDA register is concatenated with the CRDA register to generate the full 32-bit address.



TL/F/10492–16

FIGURE 3-9. RDA Initialization Example

3.4.4.4 Initializing the Lower Boundary of the RBA

A "false bottom" is set in the RBA by loading the End Of Buffer Count (EOBC) register with a value equal to the maximum size packet in words (16 bits) that may be received. This creates a lower boundary in the RBA. Whenever the Remaining Buffer Word Count (RBWC0,1) registers decrement below the EOBC register, the SONIC buffers the next packet into another RBA. This also guarantees that a packet is always contiguously buffered into a single Receive Buffer Area (RBA). The SONIC does not buffer a packet into multiple RBAs. Note that in 32-bit mode, the SONIC holds the LSB always low so that it properly compares with the RBWC0,1 registers.

After a hardware reset, the EOBC register is automatically initialized to 2F8h (760 words or 1520 bytes). For 32-bit applications this is the suggested value for EOBC. EOBC defaults to 760 words (1520 bytes) instead of 759 words (1518 bytes) because 1518 is not a double word (32-bit) boundary (see section 3.4.2.1). If the SONIC is used in 16-bit mode, then EOBC should be set to 759 words (1518 bytes) because 1518 is a word (16-bit) boundary.

Sometimes it may be desired to buffer a single packet per RBA. When doing this, it is important to set EOBC and the buffer size correctly. The suggested practice is to set EOBC to a value that is at least 4 bytes, in 32-bit mode, or 2 bytes, in 16-bit mode, less than the buffer size. An example of this for 32-bit mode is to set EOBC to 760 words (1520 bytes)

3.0 Buffer Management (Continued)

and the buffer size to 762 words (1524 bytes). A similar example for 16-bit mode would be EOBC = 759 words (1518 bytes) and the buffer size set to 760 words (1520 bytes). The buffer can be any size, but as long as the EOBC is 2 words, for 32-bit mode, or 1 word, for 16-bit mode, less than the buffer size, only one packet will be buffered in that RBA.

Note 1: It is possible to filter out most oversized packets by setting the buffer size to 760 words (1520 bytes) in 32-bit mode or 759 words (1518 bytes) in 16-bit mode. EOBC would be set to 758 words (1516 bytes) for both cases. With this configuration, any packet over 1520 bytes, in 32-bit mode, or 1518 bytes, in 16-bit mode, will not be completely buffered because the packet will overflow the buffer. When a packet overflow occurs, a Receive Buffer Area Exceeded interrupt (RBAE in the Interrupt Status Register, section 4.3.6) will occur.

Note 2: When buffering one packet per buffer, it is suggested that the values in Note 1 above be used. Since the minimum legal sized Ethernet packet is 64 bytes, however, it is possible to set EOBC as much as 64 bytes less than the buffer size and still end up with one packet per buffer. *Figure 3-10* shows this "range."

3.4.5 Beginning Of Reception

At the beginning of reception, the SONIC checks its internally stored EOL bit from the previous RXpkt.link field for a "1". If the SONIC finds EOL=1, it recognizes that after the previous reception, there were no more remaining receive packet descriptors. It re-reads the same RXpkt.link field to check if the system has updated this field since the last reception. If the SONIC still finds EOL=1, reception ceases. (See section 3.5 for adding descriptors to the list.) Otherwise, the SONIC begins storing the packet in the RBA starting at the Current Receive Buffer Address (CRBA0,1) registers and continues until the packet has completed. Concurrent with the packet reception, the Remaining Buffer Word Count (RBWC0,1) registers are decremented after each word is written to memory. This register determines the remaining words in the RBA at the end of reception.

3.4.6 End Of Packet Processing

At the end of a reception, the SONIC enters its end of packet processing sequence to determine whether to accept or reject the packet based on receive errors and packet size. At the end of reception the SONIC enters one of the following two sequences:

— Successful reception sequence

— Buffer recovery for runt packets or packets with errors

3.4.6.1 Successful Reception

If the SONIC accepts the packet, it first writes 5 words of descriptor information in the RDA beginning at the address pointed to by the Current Receive Descriptor Address (CRDA) register. It then reads the RXpkt.link field to advance the CRDA register to the next receive descriptor. The SONIC also checks the EOL bit for a "1" in this field. If EOL=1, no more descriptors are available for the SONIC. The SONIC recovers the address of the current RXpkt.link field (from a temporary register) and generates a "Receive Descriptors Exhausted" indication in the Interrupt Status register. (See section 3.4.7 on how to add descriptors.) The SONIC maintains ownership of the descriptor by *not* writing to the RXpkt.in_use field. Otherwise, if EOL=0, the SONIC advances the CRDA register to the next descriptor and resets the RXpkt.in_use field to all "0's".

The SONIC accesses the complete 7 word RDA descriptor in a single block operation.

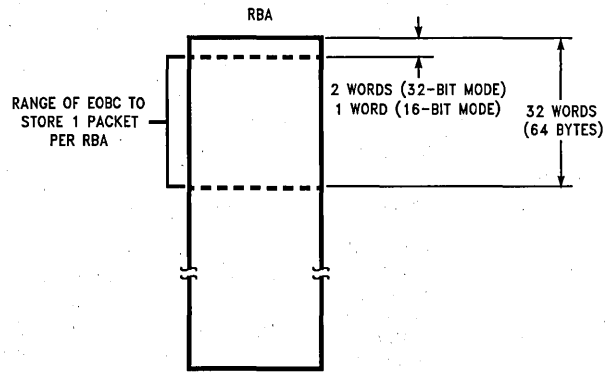
The SONIC also checks if there is remaining space in the RBA. The SONIC compares the Remaining Buffer Word Count (RBWC0,1) registers with the static End Of Buffer Count (EOBC). If the RBWC is less than the EOBC, a maximum sized packet will no longer fit in the remaining space in the RBA; hence, the SONIC fetches a resource descriptor from the RRA and loads its registers with the pointer and word count of the next available RBA.

3.4.6.2 Buffer Recovery For Runt Packets Or Packets With Errors

If a runt packet (less than 64 bytes) or packet with errors arrives and the Receive Control register has been configured to not accept these packets, the SONIC recovers its pointers back to the original positions. The CRBA0,1 registers are not advanced and the RBWC0,1 registers are not decremented. The SONIC recovers its pointers by maintaining a copy of the buffer address in the Temporary Receive Buffer Address registers (TRBA0,1). The SONIC recovers the value in the RBWC0,1 registers from the Temporary Buffer Word Count registers (TBWC0,1).

3.4.7 Overflow Conditions

When an overflow condition occurs, the SONIC halts its DMA operations to prevent writing into unauthorized memory. The SONIC uses the Interrupt Status register (ISR) to indicate three possible overflow conditions that can occur



TL/F/10492–17

Range of EOBC = (RXrsrc.wc0,1 −2 to RXrsrc.wc0,1 −32)

FIGURE 3-10. Setting EOBC for Single Packet RBA

3.0 Buffer Management (Continued)

when its receive resources have been exhausted. The system should respond by replenishing the resources that have been exhausted. These overflow conditions (Descriptor Resources Exhausted, Buffer Resources Exhausted, and RBA Limit Exceeded) are indicated in the Interrupt Status register and are detailed as follows:

Descriptor Resources Exhausted: This occurs when the SONIC has reached the last receive descriptor in the list, meaning that the SONIC has detected EOL = 1. The system must supply additional descriptors for continued reception. The system can do this in one of two ways: 1) appending descriptors to the existing list, or 2) creating a separate list.

1) Appending descriptors to the existing list. This is the easiest and preferred way. To do this, the system, after creating the new list, joins the new list to the existing list by simply writing the beginning address of the new list into the RXpkt.link field and setting EOL = 0. At the next reception, the SONIC re-reads the last RXpkt.link field, and updates its CRDA register to point to the next descriptor.
2) Creating a separate list. This requires an additional step because the lists are not joined together and requires that the CRDA register be loaded with the address of the RXpkt.link field in the new list.

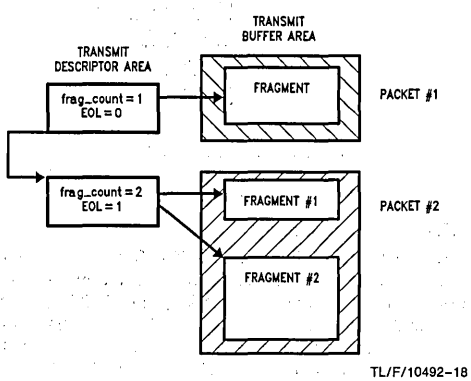
During this overflow condition, the SONIC maintains ownership of the descriptor (RXpkt.in_use ≠ 00h) and waits for the system to add additional descriptors to the list. When the system appends more descriptors, the SONIC releases ownership of the descriptor after writing 0000h to the RXpkt.in_use field.

Buffer Resources Exhausted: This occurs when the SONIC has detected that the Resource Read Pointer (RRP) and Resource Write Pointer (RWP) registers are equal (i.e., all RRA descriptors have been exhausted). The RBE bit in the Interrupt Status register is set when the SONIC finishes using the second to last receive buffer and reads the last RRA descriptor. Actually, the SONIC is not truly out of resources, but gives the system an early warning of an impending out of resources condition. To continue reception after the last RBA is used, the system must supply additional RRA descriptor(s), update the RWP register, and clear the RBE bit in the ISR. The SONIC rereads the RRA after this bit is cleared.

RBA Limit Exceeded: This occurs when a packet does not completely fit within the remaining space of the RBA. This can occur if the EOBC register is not programmed to a value greater than the largest packet that can be received. When this situation occurs, the packet is truncated and the SONIC reads the RRA to obtain another RBA. Indication of an RBA limit being exceeded is signified by the Receive Buffer Area Exceeded (RBAE) interrrupt being set (see section 4.3.6). An RDA will not be set up for the truncated packet and the buffer space will not be re-used. To rectify this potential overflow condition, the EOBC register must be loaded with a value equal to or greater than the largest packet that can be accepted. See section 3.4.2.

3.5 TRANSMIT BUFFER MANAGEMENT

To begin transmission, the system software issues the Transmit command (TXP = 1 in the CR). The Transmit Buffer Management uses two areas in memory for transmitting packets *(Figure 3-11)*, the Transmit Descriptor Area (TDA) and the Transmit Buffer Area (TBA). During transmission, the SONIC fetches control information from the TDA, loads its appropriate registers, and then transmits the data from the TBA. When the transmission is complete, the SONIC writes the status information in the TDA. From a single transmit command, packets can either be transmitted singly or in groups if several descriptors have been linked together.



TL/F/10492–18

FIGURE 3-11. Overview of Transmit Buffer Management

3.5.1 Transmit Descriptor Area (TDA)

The TDA contains descriptors that the system has generated to exchange status and control information. Each descriptor corresponds to a single packet and consists of the following 16-bit fields.

TXpkt.status: This field is written by the SONIC and provides status of the transmitted packet. See section 3.5.1.2 for more details.

TXpkt.config: This field allows programming the SONIC to one of the various transmit modes. The SONIC reads this field and loads the corresponding configuration bits (PINTR, POWC, CRCI, and EXDIS) into the Transmit Control register. See section 3.5.1.1 for more details.

TXpkt.pkt_size: This field contains the byte count of the entire packet

TXpkt.frag_count: This field contains the number of fragments the packet is segmented into.

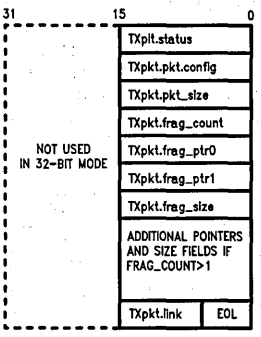
TXpkt.frag_ptr0,1: This field contains a 32-bit pointer which locates the packet fragment to be transmitted in the Transmit Buffer Area (TBA). This pointer is not restricted to any byte alignment.

TXpkt.frag_size: This field contains the byte count of the packet fragment. The minimum fragment size is 1 byte.

TXpkt.link: This field contains a 15-bit pointer (A15–A1) to the next TDA descriptor. The LSB, the End Of List (EOL) bit, indicates the last descriptor in the list when set to a "1". When descriptors have been linked together, the SONIC transmits back-to-back packets from a single transmit command.

The data of the packet does not need to be contiguous, but can exist in several locations (fragments) in memory. In this case, the TXpkt.frag_count field is greater than one, and additional TXpkt.frag_ptr0,1 and TXpkt.frag_size fields corresponding to each fragment are used. The descriptor format is shown in *Figure 3-12*. Note that in 32-bit mode the upper word, D<31:16>, is not used.

3.0 Buffer Management (Continued)



TL/F/10492–19

FIGURE 3-12. Transmit Descriptor Area

3.5.1.1 Transmit Configuration

The TXpkt.config field allows the SONIC to be programmed into one of the transmit modes before each transmission. At the beginning of each transmission, the SONIC reads this field and loads the PINTR, POWC, CRCI, and EXDIS bits into the Transmit Control register (TCR). The configuration bits in the TCR correspond directly with the bits in the TXpkt.config field as shown in *Figure 3-13.* See section 4.3.4 for the description on the TCR.

15	14	13	12	11	10	9	8
PINTR	POWC	CRCI	EXDIS	X	X	X	X

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

Note: x = don't care

FIGURE 3-13. TXpkt.config Field

3.5.1.2 Transmit Status

At the end of each transmission the SONIC writes the status bits (<10:0>) of the Transmit Control Register (TCR) and the number of collisions experienced during the transmission into the TXpkt.status field *(Figure 3-14,* res = reserved). Bits NC4-NC0 indicate the number of collisions where NC4 is the MSB. See section 4.3.4 for the description of the TCR.

15	14	13	12	11	10	9	8
NC4	NC3	NC2	NC1	NC0	EXD	DEF	NCRS

7	6	5	4	3	2	1	0
CRSL	EXC	OWC	res	PMB	FU	BCM	PTX

FIGURE 3-14. TXpkt.status Field

3.5.2 Transmit Buffer Area (TBA)

The TBA contains the fragments of packets that are defined by the descriptors in the TDA. A packet can consist of a single fragment or several fragments, depending upon the fragment count in the TDA descriptor. The fragments also can reside anywhere within the full 32-bit address range, and be aligned to any byte boundary. When an odd byte boundary is given, the SONIC automatically begins reading data at the corresponding word boundary in 16-bit mode or a long word boundary in 32-bit mode. The SONIC ignores the extraneous bytes which are written into the FIFO during odd byte alignment fragments. The minimum allowed fragment size is 1 byte. *Figure* 3-11 shows the relationship between the TDA and the TBA for single and multi-fragmented packets.

3.5.3 Preparing To Transmit

All fields in the TDA descriptor and the Current Transmit Descriptor Address (CTDA) register of the SONIC must be initialized before the Transmit Command (setting the TXP bit in the Command register) can be issued. If more than one packet is queued, the descriptors must be linked together with the TXpkt.link field. The last descriptor must have EOL=1 and all other descriptors must have EOL=0. To begin transmission, the system loads the address of the first TXpkt.status field into the CTDA register. Note that the upper 16-bits of address are loaded in the Upper Transmit Descriptor (UTDA) register. The user performs the following transmit initialization.

1) Initialize the TDA
2) Load the CTDA register with the address of the first transmit descriptor
3) Issue the transmit command

Note that if the Source Address of the packet being transmitted is not in the CAM, the Packet Monitored Bad (PMB) bit in the TXpxt.status field will be set (see section 4.3.4).

3.5.3.1 Transmit Process

When the Transmit Command (TXP = 1 in the Command register) is issued, the SONIC fetches the control information in the TDA descriptor, loads its appropriate registers (shown below) and begins transmission. (See section 4.2 for register mnemonics.)

TCR ← TXpkt.config
TPS ← TXpkt.pkt_size
TFC ← TXpkt.frag_count
TSA0 ← TXpkt.frag_ptr0
TSA1 ← TXpkt.frag_ptr1
TFS ← TXpkt.frag_size
CTDA ← TXpkt.link

(CTDA is loaded after all fragments have been read and successfully transmitted. If the halt transmit command is issued (HTX bit in the Command register is set) the CTDA register is not loaded.)

During transmission, the SONIC reads the packet descriptor in the TDA and transmits the data from the TBA. If TXpkt.frag_count is greater than one, the SONIC, after finishing transmission of the fragment, fetches the next TXpkt.frag_ptr0,1 and TXpkt.frag_size fields and transmits the next fragment. This process continues until all fragments of a packet are transmitted. At the end of packet transmission, status is written in to the TXpkt.status field. The SONIC then reads the TXpkt.link field and checks if EOL = 0. If it is "0", the SONIC fetches the next descriptor and transmits the next packet. If EOL = 1 the SONIC generates a "Transmission Done" indication in the Interrupt Status register and resets the TXP bit in the Command register.

In the event of a collision, the SONIC recovers its pointer in the TDA and retransmits the packet up to 15 times. The SONIC maintains a copy of the CTDA register in the Temporary Transmit Descriptor Address (TTDA) register.

The SONIC performs a block operation of 6, 3, or 2 accesses in the TDA, depending on where the SONIC is in the transmit process. For the first fragment, it reads the

3.0 Buffer Management (Continued)

TXpkt.config to TXpkt.frag_size (6 accesses). For the next fragment, if any, it reads the next 3 fields from TXpkt.frag_ptr0 to TXpkt.frag_size (3 accesses). At the end of transmission it writes the status information to TXpkt.status and reads the TXpkt.link field (2 accesses).

3.5.3.2 Transmit Completion

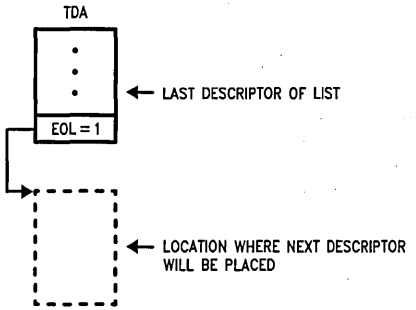
The SONIC stops transmitting under two conditions. In the normal case, the SONIC transmits the complete list of descriptors in the TDA and stops after it detects EOL = 1. In the second case, certain transmit errors cause the SONIC to abort transmission. If *FIFO Underrun, Byte Count Mismatch, Excessive Collision,* or *Excessive Deferral* (if enabled) errors occur, transmission ceases. The CTDA register points to the last packet transmitted. The system can also halt transmission under software control by setting the HTX bit in the Command register. Transmission halts after the SONIC writes to the TXpkt.status field.

3.5.4 Dynamically Adding TDA Descriptors

Descriptors can be dynamically added during transmission without halting the SONIC. The SONIC can also be guaranteed to transmit the complete list including newly appended descriptors (barring any transmit abort conditions) by observing the following rule: The last TXpkt.link field must point to the next location where a descriptor will be added (see step 3 below and *Figure 3-15*). The procedure for appending descriptors consists of:

1. Creating a new descriptor with its TXpkt.link pointing to the next vacant descriptor location and its EOL bit set to a "1".
2. Resetting the EOL bit to a "0" of the previously last descriptor.
3. Re-issuing the Transmit command (setting the TXP bit in the Command register).

Step 3 assures that the SONIC will transmit all the packets in the list. If the SONIC is currently transmitting, the Transmit command has no effect and continues transmitting until it detects EOL = 1. If the SONIC had just finished transmitting, it continues transmitting from where it had previously stopped.



TL/F/10492–20

FIGURE 3-15. Initializing Last Link Field

4.0 SONIC Registers

The SONIC contains two sets of registers: The status/control registers and the CAM memory cells. The status/control registers are used to configure, control, and monitor SONIC operation. They are directly addressable registers and occupy 64 consecutive address locations in the system memory space (selected by the RA5–RA0 address pins). There are a total of 64 status/control registers divided into the following categories:

User Registers: These registers are accessed by the user to configure, control, and monitor SONIC operation. These are the only SONIC registers the user needs to access. *Figure 4-3* shows the programmer's model and Table 4-1 lists the attributes of each register.

Internal Use Registers: These registers (Table 4-2) are used by the SONIC during normal operation and are not intended to be accessed by the user.

National Factory Test Registers: These registers (Table 4-3) are for National factory use only and should never be accessed by the user. Accessing these registers during normal operation can cause improper functioning of the SONIC.

4.1 THE CAM UNIT

The CAM unit memory cells are indirectly accessed by programming the CAM descriptor area in system memory and issuing the LCAM command (setting the LCAM bit in the Control register). The CAM cells do not occupy address locations in register space and, thus, are not accessible through the RA5–RA0 address pins. The CAM control registers, however, are part of the user register set and must be initialized before issuing the LCAM command (see section 4.3.10).

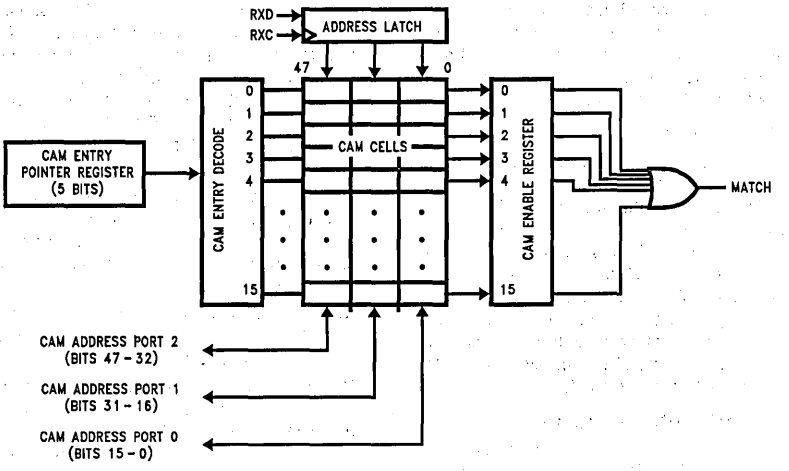
The Content Addressable Memory (CAM) consists of sixteen 48-bit entries for complete address filtering *(Figure 4-1)* of network packets. Each entry corresponds to a 48-bit destination address that is user programmable and can contain any combination of Multicast or Physical addresses. Each entry is partitioned into three 16-bit CAM cells accessible through CAM Address Ports (CAP 2, CAP 1 and CAP 0) with CAP0 corresponding to the least significant 16 bits of the Destination Address and CAP2 corresponding to the most significant bits. The CAM is accessed in a two step process. First, the CAM Entry Pointer is loaded to point to one of the 16 entries. Then, each of the CAM Address Ports is accessed to select the CAM cell. The 16 user programmable CAM entries can be masked out with the CAM Enable register (see section 4.3.10).

Note: It is not necessary to program a broadcast address into the CAM when it is desired to accept broadcast packets. Instead, to accept broadcast packets, set the BRD bit in the Receive Control register. If the BRD bit has been set, the CAM is still active. This means that it is possible to accept broadcast packets at the same time as accepting packets that match physical addresses in the CAM.

4.1.1 The Load CAM Command

Because the SONIC uses the CAM for a relatively long period of time during reception, it can only be written to via the CAM Descriptor Area (CDA) and is only readable when the

4.0 SONIC Registers (Continued)



TL/F/10492–21

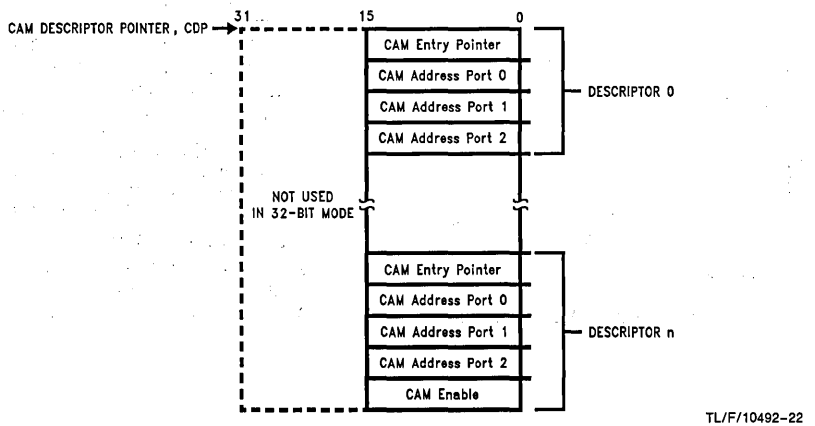
FIGURE 4-1. CAM Organization

SONIC is in software reset. The CDA resides in the same 64k byte block of memory as the Receive Resource Area (RRA) and contains descriptors for loading the CAM registers. These descriptors are contiguous and each descriptor consists of four 16-bit fields *(Figure 4-2)*. In 32-bit mode the upper word, D<31:16>, is not used. The first field contains the value to be loaded into the CAM Entry Pointer and the remaining fields are for the three CAM Address Ports (see section 4.3.10). In addition, there is one more field after the last descriptor containing the mask for the CAM Enable register. Each of the CAM descriptors are addressed by the CAM Descriptor Pointer (CDP) register.

After the system has initialized the CDA, it can issue the Load CAM command to program the SONIC to read the CDA and load the CAM. The procedure for issuing the Load CAM command is as follows.

1. Initialize the Upper Receive Resource Address (URRA) register. Note that the CAM Descriptor Area must reside within the same 64k page as the Receive Resource Area. (See section 4.3.9).
2. Initialize the CDA as described above.
3. Initialize the CAM Descriptor Count with the number of CAM descriptors. Note, only the lower 5 bits are used in this register. The other bits are don't cares. (See section 4.3.10).
4. Initialize the CAM Descriptor Pointer to locate the first descriptor in the CDA. This register must be reloaded each time a new Load CAM command is issued.
5. Issue the Load CAM command (LCAM) in the Command register. (See section 4.3.1).

If a transmission or reception is in progress, the CAM DMA function will not occur until these operations are complete. When the SONIC completes the Load CAM command, the CDP register points to the next location after the CAM Enable field and the CDC equals zero. The SONIC resets the LCAM bit in the Command register and sets the Load CAM Done (LCD) bit in the ISR.



TL/F/10492–22

FIGURE 4-2. CAM Descriptor Area Format

4.0 SONIC Registers (Continued)

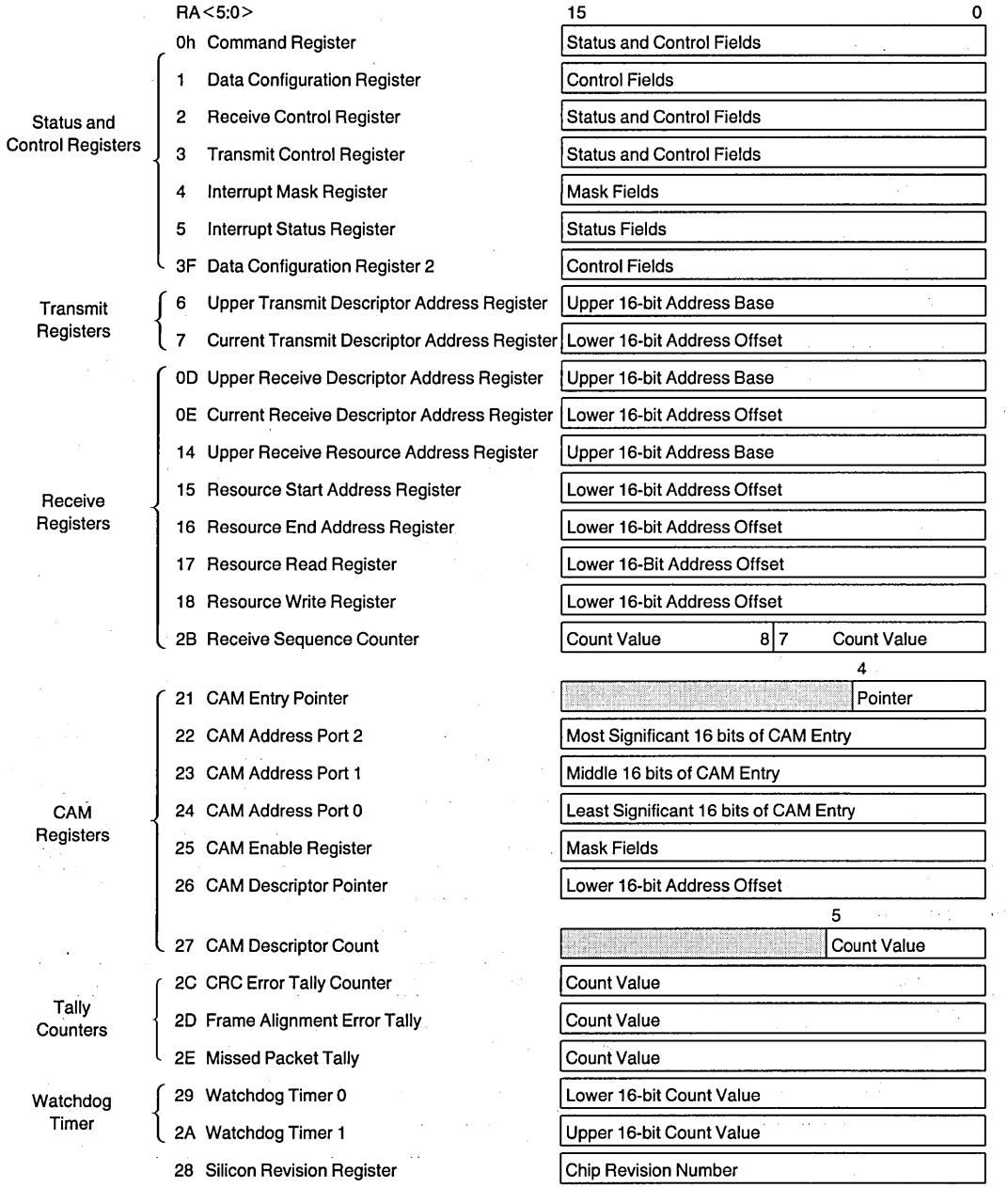


FIGURE 4-3. Register Programming Model

4.0 SONIC Registers (Continued)

4.2 STATUS/CONTROL REGISTERS

This set of registers is used to convey status/control information to/from the host system and to control the operation of the SONIC. These registers are used for loading commands generated from the system, indicating transmit and receive status, buffering data to/from memory, and providing interrupt control. The registers are selected by asserting chip select to the SONIC and providing the necessary address on register address pins RA5–RA0. Tables 4-1, 4-2, and 4-3 show the locations of all SONIC registers and where information on the registers can be found in the data sheet.

TABLE 4-1. User Registers

RA5–RA0	Access	Register	Symbol	Description (section)
COMMAND AND STATUS REGISTERS				
00h	R/W	Command	CR	4.3.1
01 (Note 3)	R/W	Data Configuration	DCR	4.3.2
02	R/W	Receive Control	RCR	4.3.3
03	R/W	Transmit Control	TCR	4.3.4
04	R/W	Interrupt Mask	IMR	4.3.5
05	R/W	Interrupt Status	ISR	4.3.6
3F (Note 3)	R/W	Data Configuration 2	DCR2	4.3.7
TRANSMIT REGISTERS				
06	R/W	Upper Transmit Descriptor Address	UTDA	4.3.8, 3.4.4.1
07	R/W	Current Transmit Descriptor Address	CTDA	4.3.8, 3.5.3
RECEIVE REGISTERS				
0D	R/W	Upper Receive Descriptor Address	URDA	4.3.9, 3.4.4.1
0E	R/W	Current Receive Descriptor Address	CRDA	4.3.9, 3.4.4.3
13	R/W	End of Buffer Word Count	EOBC	4.3.9, 3.4.2
14	R/W	Upper Receive Resource Address	URRA	4.3.9, 3.4.4.1
15	R/W	Resource Start Address	RSA	4.3.9, 3.4.1
16	R/W	Resource End Address	REA	4.3.9, 3.4.1
17	R/W	Resource Read Pointer	RRP	4.3.9, 3.4.1
18	R/W	Resource Write Pointer	RWP	4.3.9, 3.4.1
2B	R/W	Receive Sequence Counter	RSC	4.3.9, 3.4.3.2
CAM REGISTERS				
21	R/W	CAM Entry Pointer	CEP	4.1, 4.3.10
22 (Note 1)	R	CAM Address Port 2	CAP2	4.1, 4.3.10
23 (Note 1)	R	CAM Address Port1	CAP1	4.1, 4.3.10
24 (Note 1)	R	CAM Address Port 0	CAP0	4.1, 4.3.10
25 (Note 2)	R/W	CAM Enable	CE	4.1, 4.3.10
26	R/W	CAM Descriptor Pointer	CDP	4.1, 4.3.10
27	R/W	CAM Descriptor Count	CDC	4.1, 4.3.10
TALLY COUNTERS				
2C (Note 4)	R/W	CRC Error Tally	CRCT	4.3.11
2D (Note 4)	R/W	FAE Tally	FAET	4.3.11
2E (Note 4)	R/W	Missed Packet Tally	MPT	4.3.11

4.0 SONIC Registers (Continued)

TABLE 4-1. User Registers (Continued)

RA5–RA0	Access	Register	Symbol	Description (section)
WATCHDOG COUNTERS				
29	R/W	Watchdog Timer 0	WT0	4.3.12
2A	R/W	Watchdog Timer 1	WT1	4.3.12
SILICON REVISION				
28	R	Silicon Revision	SR	4.3.13

Note 1: These registers can only be read when the SONIC is in reset mode (RST bit in the CR is set). The SONIC gives invalid data when these registers are read in non-reset mode.

Note 2: This register can only be written to when the SONIC is in reset mode. This register is normally only loaded by the Load CAM command.

Note 3: The Data Configuration registers, DCR and DCR2, can only be written to when the SONIC is in reset mode (RST bit in CR is set). Writing to these registers while not in reset mode does not alter the registers.

Note 4: The data written to these registers is inverted before being latched. That is, if a value of FFFFh is written, these registers will contain and read back the value of 0000h. Data is not inverted during a read operation.

TABLE 4-2. Internal Use Registers (Users should not write to these registers)

(RA5–RA0)	Access	Register	Symbol	Description (section)
TRANSMIT REGISTERS				
08 (Note 1)	R/W	Transmit Packet Size	TPS	3.5
09	R/W	Transmit Fragment Count	TFC	3.5
0A	R/W	Transmit Start Address 0	TSA0	3.5
0B	R/W	Transmit Start Address 1	TSA1	3.5
0C (Note 2)	R/W	Transmit Fragment Size	TFS	3.5
20	R/W	Temporary Transmit Descriptor Address	TTDA	3.5.4
2F	R	Maximum Deferral Timer	MDT	4.3.4
RECEIVE REGISTERS				
0F	R/W	Current Receive Buffer Address 0	CRBA0	3.4.2, 3.4.4.2
10	R/W	Current Receive Buffer Address 1	CRBA1	3.4.2, 3.4.4.2
11	R/W	Remaining Buffer Word Count 0	RBWC0	3.4.2, 3.4.4.2
12	R/W	Remaining Buffer Word Count 1	RBWC1	3.4.2, 3.4.4.2
19	R/W	Temporary Receive Buffer Address 0	TRBA0	3.4.6.2
1A	R/W	Temporary Receive Buffer Address 1	TRBA1	3.4.6.2
1B	R/W	Temporary Buffer Word Count 0	TBWC0	3.4.6.2
1C	R/W	Temporary Buffer Word Count 1	TBWC1	3.4.6.2
1F	R/W	Last Link Field Address	LLFA	none
ADDRESS GENERATORS				
1D	R/W	Address Generator 0	ADDR0	none
1E	R/W	Address Generator 1	ADDR1	none

Note 1: The data that is read from these registers is the inversion of what has been written to them.

Note 2: The value that is written to this register is shifted once in 16-bit mode and shifted twice in 32-bit mode.

TABLE 4-3. National Factory Test Registers

(RA5–RA0)	Access	Register	Symbol	Description (section)
30 • 3E	R/W	These registers are for factory use only. Users must not address these registers or improper SONIC operation can occur	none	none

4.0 SONIC Registers (Continued)

4.3 REGISTER DESCRIPTION

4.3.1 Command Register

(RA<5:0> = 0h)

This register *(Figure 4-4)* is used for issuing commands to the SONIC. These commands are issued by setting the corresponding bits for the function. For all bits, except for the RST bit, the SONIC resets the bit after the command is completed. With the exception of RST, writing a "0" to any bit has no effect. Before any commands can be issued, the RST bit must first be reset to "0". This means that, if the RST bit is set, two writes to the Command Register are required to issue a command to the SONIC; one to clear the RST bit, and one to issue the command.

This register also controls the general purpose 32-bit Watchdog Timer. After the Watchdog Timer register has been loaded, it begins to decrement once the ST bit has been set to "1". An interrupt is issued when the count reaches zero if the Timer Complete interrupt is enabled in the IMR.

During hardware reset, bits 7, 4, and 2 are set to a "1"; all others are cleared. During software reset bits 9, 8, 1, and 0 are cleared and bits 7 and 2 are set to a "1"; all others are unaffected.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LCAM	RRRA	RST	0	ST	STP	RXEN	RXDIS	TXP	HTX
						r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w

r = read only, r/w = read/write

FIGURE 4-4. Command Register

Field	Meaning
LCAM	LOAD CAM
RRRA	READ RRA
RST	SOFTWARE RESET
ST	START TIMER
STP	STOP TIMER
RXEN	RECEIVER ENABLE
RXDIS	RECEIVER DISABLE
TXP	TRANSMIT PACKET(S)
HTX	HALT TRANSMISSION

Bit	Description
15–10	Must be 0
9	**LCAM: LOAD CAM** Setting this bit causes the SONIC to load the CAM with the descriptor that is pointed to by the CAM Descriptor Pointer register. **Note:** This bit must not be set during transmission (TXP is set). The SONIC will lock up if both bits are set simultaneously.
8	**RRRA: READ RRA** Setting this bit causes the SONIC to read the next RRA descriptor pointed to by the Resource Read Pointer (RRP) register. Generally this bit is only set during initialization. Setting this bit during normal operation can cause improper receive operation.
7	**RST: SOFTWARE RESET** Setting this bit resets all internal state machines. The CRC generator is disabled and the Tally counters are halted, but not cleared. The SONIC becomes operational when this bit is reset to "0". A hardware reset sets this bit to a "1". It must be reset to "0" before the SONIC becomes operational.
6	Must be 0.
5	**ST: START TIMER** Setting this bit enables the general-purpose watchdog timer to begin counting or to resume counting after it has been halted. This bit is reset when the timer is halted (i.e., STP is set). Setting this bit resets STP.
4	**STP: STOP TIMER** Setting this bit halts the general-purpose watchdog timer and resets the ST bit. The timer resumes when the ST bit is set. This bit powers up as a "1". Note: Simultaneously setting bits ST and STP stops the timer.

4.0 SONIC Registers (Continued)

4.3 REGISTER DESCRIPTION

4.3.1 Command Register (Continued)

(RA<5:0> = 0h)

Bit	Description
3	**RXEN: RECEIVER ENABLE** Setting this bit enables the receive buffer management engine to begin buffering data to memory. Setting this bit resets the RXDIS bit. Note: If this bit is set while the MAC unit is currently receiving a packet, both RXEN and RXDIS are set until the network goes inactive (i.e., the SONIC will not start buffering in the middle of a packet being received).
2	**RXDIS: RECEIVER DISABLE** Setting this bit disables the receiver from buffering data to memory or the Receive FIFO. If this bit is set during the reception of a packet, the receiver is disabled only after the packet is processed. The RXEN bit is reset when the receiver is disabled. Tally counters remain active regardless of the state of this bit. Note: If this bit is set while the SONIC is currently receiving a packet, both RXEN and RXDIS are set until the packet is fully received.
1	**TXP: TRANSMIT PACKET(S)** Setting this bit causes the SONIC to transmit packets which have been set up in the Transmit Descriptor Area (TDA). The SONIC loads its appropriate registers from the TDA, then begins transmission. The SONIC clears this bit after any of the following conditions have occurred: (1) transmission had completed (i.e., after the SONIC has detected EOL = 1), (2) the Halt Transmission command (HTX) has taken effect, or (3) a transmit abort condition has occurred. This condition occurs when any of the following bits in the TCR have been set: EXC, EXD, FU, or BCM. **Note:** This bit must not be set if a Load CAM operation is in progress (LCAM is set). The SONIC will lock up if both bits are set simultaneously.
0	**HTX: HALT TRANSMISSION** Setting this bit halts the transmit command after the current transmission has completed. TXP is reset after transmission has halted. The Current Transmit Descriptor Address (CTDA) register points to the last descriptor transmitted. The SONIC samples this bit after writing to the TXpkt.status field.

4.0 SONIC Registers (Continued)

4.3.2 Data Configuration Register

(RA<5:0> = 1h)

This register *(Figure 4-5)* establishes the bus cycle options for reading/writing data to/from 16- or 32-bit memory systems.

During a hardware reset, bits 15 and 13 are cleared; all other bits are unaffected. (Because of this, the first thing the driver software does to the SONIC should be to set up this register.) All bits are unaffected by a software reset. This register must only be accessed when the SONIC is in reset mode (i.e., the RST bit is set in the Command register).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXBUS	0	LBR	PO1	PO0	SBUS	USR1	USR0	WC1	WC0	DW	BMS	RFT1	RFT0	TFT1	TFT0
r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

r/w = read/write

FIGURE 4-5. Data Configuration Register

Field	Meaning
EXBUS	EXTENDED BUS MODE
LBR	LATCHED BUS RETRY
PO0,PO1	PROGRAMMABLE OUTPUTS
SBUS	SYNCHRONOUS BUS MODE
USR0, USR1	USER DEFINABLE PINS
WC0, WC1	WAIT STATE CONTROL
DW	DATA WIDTH SELECT
BMS	BLOCK MODE SELECT FOR DMA
RFT0, RFT1	RECEIVE FIFO THRESHOLD
TFT0, TFT1	TRANSMIT FIFO THRESHOLD

Bit	Description
15	**EXBUS: EXTENDED BUS MODE** Setting this bit enables the Extended Bus mode which enables the following: 1)Extended Programmable Outputs, EXUSR <3:0>: This changes the TXD, LBK, RXC and RXD pins from the external ENDEC interface into four programmable user outputs, EXUSR <3:0> respectively, which are similar to USR <1:0>. These outputs are programed with bits 15-12 in the DCR2 (see section 4.3.7). On hardware reset, these four pins will be TRI-STATE and will remain that way until the DCR is changed. If EXBUS is enabled, then these pins will remain TRI-STATE until the SONIC becomes a bus master, at which time they will be driven according to the DCR2. If EXBUS is disabled, then these four pins work normally as external ENDEC interface pins. 2)Synchronous Termination, $\overline{\text{STERM}}$: This changes the TXC pin from the External ENDEC interface into a synchronous memory termination input for compatibility with Motorola style processors. This input is only useful when Asynchronous Bus mode is selected (bit 10 below is set to "0") and BMODE = 1 (Motorola mode). On hardware reset, this pin will be TRI-STATE and will remain that way until the DCR is changed. If EXBUS is enabled, this pin will remain TRI-STATE until the SONIC becomes a bus master, at which time it will become the $\overline{\text{STERM}}$ input. If EXBUS is disabled, then this pin works normally as the TXC pin for the external ENDEC interface. 3)Asynchronous Bus Retry: Causes $\overline{\text{BRT}}$ to be clocked in asynchronously off the falling edge of bus clock. This only applies, however, when the SONIC is operating in asynchronous mode (bit 10 below is set to "0"). If EXBUS is not set, $\overline{\text{BRT}}$ is sampled synchronously off the rising edge of bus clock. (See section 5.4.6.)
14	Must be 0.
13	**LBR: LATCHED BUS RETRY** The LBR bit controls the mode of operation of the $\overline{\text{BRT}}$ signal (see pin description). It allows the BUS Retry operation to be latched or unlatched. 0:Unlatched mode: The assertion of $\overline{\text{BRT}}$ forces the SONIC to finish the current DMA operation and get off the bus. The SONIC will retry the operation when $\overline{\text{BRT}}$ is deserted. 1:Latched mode: The assertion of $\overline{\text{BRT}}$ forces the SONIC to finish the current DMA operation as above, however, the SONIC will not retry until $\overline{\text{BRT}}$ is deasserted and the BR bit in the ISR (see section 4.3.6) has been reset. Hence, the mode has been latched on until the BR bit is cleared. **Note:** Unless LBR is set to a "1", $\overline{\text{BRT}}$ must remain asserted at least until the SONIC has gone idle. See section 5.4.6 and the timing for Bus Retry in section 7.0.
12, 11	**PO1, PO0: PROGRAMMABLE OUTPUTS** The PO1,PO0 bits individually control the USR1,0 pins respectively when SONIC is a bus master (HLDA or $\overline{\text{BGACK}}$ is active). When PO1/PO0 are set to a 1 the USR1/USR0 pins are high during bus master operations and when these bits are set to a 0 the USR1/USR0 pins are low during bus master operations.

4.0 SONIC Registers (Continued)

4.3.2 Data Configuration Register (Continued)
(RA<5:0> = 1h)

Bit	Description
10	**SBUS: SYNCHRONOUS BUS MODE** The SBUS bit is used to select the mode of system bus operation when SONIC is a bus master. This bit selects the internal ready line to be either a synchronous or asynchronous input to SONIC during block transfer DMA operations. 0: Asynchronous mode. $\overline{RDYi}$ (BMODE = 0) or $\overline{DSACK0,1}$ (BMODE = 1) are respectively internally synchronized at the falling edge of the bus clock (T2 of the DMA cycle). No setup or hold times need to be met with respect to this edge to guarantee proper bus operation. 1: Synchronous mode. $\overline{RDYi}$ (BMODE = 0) and $\overline{DSACK0,1}$ (BMODE = 1) must respectively meet the setup and hold times with respect to the rising edge of T1 or T2 to guarantee proper bus operation.
9, 8	**USR1,0: USER DEFINABLE PINS** The USR1,0 bits report the level of the USR1,0 signal pins, respectively, after a chip hardware reset. If the USR1,0 signal pins are at a logical 1 (tied to V_{CC}) during a hardware reset the USR1,0 bits are set to a 1. If the USR1,0 pins are at a logical 0 (tied to ground) during a hardware reset the USR1,0 bits are set to a 0. These bits are latched on the rising edge of $\overline{RST}$. Once set they remain set/reset until the next hardware reset.
7, 6	**WC1,0: WAIT STATE CONTROL** These encoded bits determine the number of additional bus cycles (T2 states) that are added during each DMA cycle. **WC1** **WC0** **Bus Cycles Added** 0 0 0 0 1 1 1 0 2 1 1 3
5	**DW: DATA WIDTH SELECT** These bits select the data path width for DMA operations. **DW** **Data Width** 0 16-bit 1 32-bit
4	**BMS: BLOCK MODE SELECT FOR DMA** Determines how data is emptied or filled into the Receive or Transmit FIFO. 0: Empty/fill mode: All DMA transfers continue until either the Receive FIFO has emptied or the Transmit FIFO has filled completely. 1: Block mode: All DMA transfers continue until the programmed number of bytes (RFT0, RFT1 during reception or TF0, TF1 during transmission) have been transferred. (See note for TFT0, TFT1.)
3, 2	**RFT1,RFT0: RECEIVE FIFO THRESHOLD** These encoded bits determine the number of words (or long words) that are written into the receive FIFO from the MAC unit before a receive DMA request occurs. (See section 1.4.) **RFT1** **RFT0** **Threshold** 0 0 2 words or 1 long word (4 bytes) 0 1 4 words or 2 long words (8 bytes) 1 0 8 words or 4 long words (16 bytes) 1 1 12 words or 6 long words (24 bytes) **Note:** In block mode (BMS bit = 1), the receive FIFO threshold sets the number of words (or long words) written to memory during a receive DMA block cycle.
1, 0	**TFT1,TFT0: TRANSMIT FIFO THRESHOLD** These encoded bits determine the minimum number of words (or long words) the DMA section maintains in the transmit FIFO. A bus request occurs when the number of words drops below the transmit FIFO threshold. (See section 1.4.) **TFT1** **TFT0** **Threshold** 0 0 4 words or 2 long words (8 bytes) 0 1 8 words or 4 long words (16 bytes) 1 0 12 words or 6 long words (24 bytes) 1 1 14 words or 7 long words (28 bytes) **Note:** In block mode (BMS = 1), the number of bytes the SONIC reads in a single DMA burst equals the transmit FIFO threshold value. If the number of words or long words needed to fill the FIFO is less than the threshold value, then only the number of reads required to fill the FIFO in a single DMA burst will be made. Typically, with the FIFO threshold value set to 12 or 14 words, the number of memory reads needed is less than the FIFO threshold value.

4.0 SONIC Registers (Continued)

4.3.3 Receive Control Register

(RA<5:0> = 2h)

This register is used to filter incoming packets and provide status information of accepted packets *(Figure 4-6)*. Setting any of bits 15–11 to a "1" enables the corresponding receive filter. If none of these bits are set, only packets which match the CAM Address registers are accepted. Bits 10 and 9 control the loopback operations.

After reception, bits 8–0 indicate status information about the accepted packet and are set to "1" when the corresponding condition is true. If the packet is accepted, all bits in the RCR are written into the RXpkt.status field. Bits 8–6 and 3–0 are cleared at the reception of the next packet.

This register is unaffected by a software reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR	RNT	BRD	PRO	AMC	LB1	LB0	MC	BC	LPKT	CRS	COL	CRCR	FAER	LBK	PRX
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r	r	r	r	r	r	r	r

r=read only, r/w=read/write

FIGURE 4-6. Receive Control Register

Field	Meaning
ERR	ACCEPT PACKET WITH ERRORS
RNT	ACCEPT RUNT PACKETS
BRD	ACCEPT BROADCAST PACKETS
PRO	PHYSICAL PROMISCUOUS PACKETS
AMC	ACCEPT ALL MULTICAST PACKETS
LB0,LB1	LOOPBACK CONTROL
MC	MULTICAST PACKET RECEIVED
BC	BROADCAST PACKET RECEIVED
LPKT	LAST PACKET IN RBA
CRS	CARRIER SENSE ACTIVITY
COL	COLLISION ACTIVITY
CRCR	CRC ERROR
FAER	FRAME ALIGNMENT ERROR
LBK	LOOPBACK PACKET RECEIVED
PRX	PACKET RECEIVED OK

Bit	Description
15	**ERR: ACCEPT PACKET WITH CRC ERRORS OR COLLISIONS** 0: Reject all packets with CRC errors or when a collision occurs. 1: Accept packets with CRC errors and ignore collisions.
14	**RNT: ACCEPT RUNT PACKETS** 0: Normal address match mode. 1: Accept runt packets (packets less than 64 bytes in length). **Note:** A hardware reset clears this bit.
13	**BRD: ACCEPT BROADCAST PACKETS** 0: Normal address match mode. 1: Accept broadcast packets (packets with addresses that match the CAM are also accepted). **Note:** This bit is cleared upon hardware reset.
12	**PRO: PHYSICAL PROMISCUOUS MODE** Enable all Physical Address packets to be accepted. 0: normal address match mode. 1: promiscuous mode.
11	**AMC: ACCEPT ALL MULTICAST PACKETS** 0: normal address match mode. 1: enables all multicast packets to be accepted. Broadcast packets are also accepted regardless of the BRD bit. (Broadcast packets are a subset of multicast packets.)

4.0 SONIC Registers (Continued)

4.3.3 Receive Control Register (Continued)
(RA<5:0> = 2h)

Bit	Description
10, 9	**LB1,LB0: LOOPBACK CONTROL** These encoded bits control loopback operations for MAC loopback, ENDEC loopback and Transceiver lookback. For proper operation, the CAM Address registers and Receive Control register must be initialized to accept the Destination address of the loopback packet (see section 1.7). **Note:** A hardware reset clears these bits. **LB1 LB0 Function** 0 0 no loopback, normal operation 0 1 MAC loopback 1 0 ENDEC loopback 1 1 Transceiver loopback
8	**MC: MULTICAST PACKET RECEIVED** This bit is set when a packet is received with a Multicast Address.
7	**BC: BROADCAST PACKET RECEIVED** This bit is set when a packet is received with a Broadcast Address.
6	**LPKT: LAST PACKET IN RBA** This bit is set when the last packet is buffered into a Receive Buffer Area (RBA). The SONIC detects this condition when its Remaining Buffer Word Count (RBWC0,1) register is less than the End Of Buffer Count (EOBC) register. (See section 3.4.2.)
5	**CRS: CARRIER SENSE ACTIVITY** Set when CRS is active. Indicates the presence of network activity.
4	**COL: COLLISION ACTIVITY** Indicates that the packet received had a collision occur during reception.
3	**CRCR: CRC ERROR** Indicates the packet contains a CRC error. If the packet also contains a Frame Alignment error, FAER will be set instead (see below).
2	**FAER: FRAME ALIGNMENT ERROR** Indicates that the incoming packet was not correctly framed on an 8-bit boundary. Note: if no CRC errors have occurred, this bit is not set (i.e., this bit is only set when both a frame alignment and CRC error occurs).
1	**LBK: LOOPBACK PACKET RECEIVED** Indicates that the SONIC has successfully received a loopback packet.
0	**PRX: PACKET RECEIVED OK** Indicates that a packet has been received without CRC, frame alignment, length (runt packet) errors or collisions.

4.0 SONIC Registers (Continued)

4.3.4 Transmit Control Register

(RA<5:0> = 3h)

This register is used to program the SONIC's transmit actions and provide status information after a packet has been transmitted *(Figure 4-7)*. At the beginning of transmission, bits 15, 14, 13 and 12 from the TXpkt.config field are loaded into the TCR to configure the various transmit modes (see section 3.5.1.1). When the transmission ends, bits 10–0 indicate status information and are set to a "1" when the corresponding condition is true. These bits, along with the number of collisions information, are written into the TXpkt.status field at the end of transmission (see section 3.5.1.2). Bits 9 and 5 are cleared after the TXpkt.status field has been written. Bits 10, 7, 6, and 1 are cleared at the commencement of the next transmission while bit 8 is set at this time.

A hardware reset sets bits 8 and 1 to a "1". This register is unaffected by a software reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINTR	POWC	CRCI	EXDIS	0	EXD	DEF	NCRS	CRSL	EXC	OWC	0	PMB	FU	BCM	PTX
r/w	r/w	r/w	r/w		r	r	r	r	r	r		r	r	r	r

r=read only, r/w=read/write

FIGURE 4-7. Transmit Control Register

Field	Meaning
PINTR	PROGRAMMABLE INTERRUPT
POWC	PROGRAMMED OUT OF WINDOW COLLISION TIMER
CRCI	CRC INHIBIT
EXDIS	DISABLE EXCESSIVE DEFERAL TIMER
EXD	EXCESSIVE DEFERRAL
DEF	DEFERRED TRANSMISSION
NCRS	NO CRS
CRSL	CRS LOST
EXC	EXCESSIVE COLLISIONS
OWC	OUT OF WINDOW COLLISION
PMB	PACKET MONITORED BAD
FU	FIFO UNDERRUN
BCM	BYTE COUNT MISMATCH
PTX	PACKET TRANSMITTED OK

Bit	Description
15	**PINTR: PROGRAMMABLE INTERRUPT** This bit allows transmit interrupts to be generated under software control. The SONIC will issue an interrupt (PINT in the Interrupt Status Register) immediately after reading a TDA and detecting that PINTR is set in the TXpkt.config field. **Note:** In order for PINTR to operate properly, it must be set and reset in the TXpkt.config field by alternating TDAs. This is necessary because after PINT has been issued in the ISR, PINTR in the Transmit Control Register must be cleared before it is set again in order to have the interrupt issued for another packet. The only effective way to do this is to set PINTR to a 1 no more often than every other packet.
14	**POWC: PROGRAM "OUT OF WINDOW COLLISION" TIMER** This bit programs when the out of window collision timer begins. 0: timer begins after the Start of Frame Delimiter (SFD). 1: timer begins after the first bit of preamble.
13	**CRCI: CRC INHIBIT** 0: transmit packet with 4-byte FCS field 1: transmit packet without 4-byte FCS field
12	**EXDIS: DISABLE EXCESSIVE DEFERRAL TIMER:** 0: excessive deferral timer enabled 1: excessive deferral timer disabled
11	Must be 0.
10	**EXD: EXCESSIVE DEFERRAL** Indicates that the SONIC has been deferring for 3.2 ms. The transmission is aborted if the excessive deferral timer is enabled (i.e. EXDIS is reset). This bit can only be set if the excessive deferral timer is enabled.

4.0 SONIC Registers (Continued)

4.3.4 Transmit Control Register (Continued)
(RA<5:0> = 3h)

Bit	Description
9	**DEF: DEFERRED TRANSMISSION** Indicates that the SONIC has deferred its transmission during the first attempt. If subsequent collisions occur, this bit is reset. This bit is cleared after the TXpkt.status field is written in the TDA.
8	**NCRS: NO CRS** Indicates that Carrier Sense (CRS) was not present during transmission. CRS is monitored from the beginning of the Start of Frame Delimiter to the last byte transmitted. The transmission will not be aborted. This bit is set at the start of preamble and is reset if CRS is detected. Hence, if CRS is never detected throughout the entire transmission of the packet, this bit will remain set. **Note:** NCRS will always remain set in MAC loopback.
7	**CRSL: CRS LOST** Indicates that CRS has gone low or has not been present during transmission. CRS is monitored from the beginning of the Start of Frame Delimiter to the last byte transmitted. The transmission will not be aborted. **Note:** If CRS was never present, both NCRS and CRSL will be set simultaneously. Also, CRSL will always be set in MAC loopback.
6	**EXC: EXCESSIVE COLLISIONS** Indicates that 16 collisions have occurred. The transmission is aborted.
5	**OWC: OUT OF WINDOW COLLISION** Indicates that an illegal collision has occurred after 51.2 μs (one slot time) from either the first bit of preamble or from SFD depending upon the POWC bit. The transmission backs off as in a normal transmission. This bit is cleared after the TXpkt.status field is written in the TDA.
4	Must be 0.
3	**PMB: PACKET MONITORED BAD** This bit is set, if after the receive unit has monitored the transmitted packet, the CRC has been calculated as invalid, a frame alignment error occurred or the Source Address does not match any of the CAM address registers. **Note 1:** The SONIC's CRC checker is active during transmission. **Note 2:** If CRC has been inhibited for transmissions (CRCI is set), this bit will always be low. This is true regardless of Frame Alignment or Source Address mismatch errors. **Note 3:** If a Receive FIFO overrun has occurred, the transmitted packet is not monitored completely. Thus, if PMB is set along with the RFO bit in the ISR, then PMB has no meaning. The packet must be completely received before PMB has meaning.
2	**FU: FIFO UNDERRUN** Indicates that the SONIC has not been able to access the bus before the FIFO has emptied. This condition occurs from excessive bus latency and/or slow bus clock. The transmission is aborted. (See section 1.4.2.)
1	**BCM: BYTE COUNT MISMATCH** This bit is set when the SONIC detects that the TXpkt.pkt_size field is not equal to the sum of the TXpkt.frag_size field(s). Transmission is aborted.
0	**PTX: PACKET TRANSMITTED OK** Indicates that a packet has been transmitted without the following errors: —Excessive Collisions (EXC) —Excessive Deferral (EXD) —FIFO Underrun (FU) —Byte Count Mismatch (BCM)

4.0 SONIC Registers (Continued)

4.3.5 Interrupt Mask Register

(RA<5:0> = 4h)

This register masks the interrupts that can be generated from the ISR *(Figure 4-8)*. Writing a "1" to the bit enables the corresponding interrupt. During a hardware reset, all mask bits are cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	BREN	HBLEN	LCDEN	PINTEN	PRXEN	PTXEN	TXEREN	TCEN	RDEEN	RBEEN	RBAEEN	CRCEN	FAEEN	MPEN	RFOEN
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

r/w=read/write

FIGURE 4-8. Interrupt Mask Register

Field	Meaning
BREN	BUS RETRY OCCURRED ENABLE
HBLEN	HEARTBEAT LOST ENABLE
LCDEN	LOAD CAM DONE INTERRUPT ENABLE
PINTEN	PROGRAMMABLE INTERRUPT ENABLE
PRXEN	PACKET RECEIVED ENABLE
PTXEN	PACKET TRANSMITTED OK ENABLE
TXEREN	TRANSMIT ERROR ENABLE
TCEN	TIMER COMPLETE ENABLE
RDEEN	RECEIVE DESCRIPTORS ENABLE
RBEEN	RECEIVE BUFFERS EXHAUSTED ENABLE
RBAEEN	RECEIVE BUFFER AREA EXCEEDED ENABLE
CRCEN	CRC TALLY COUNTER WARNING ENABLE
FAEEN	FAE TALLY COUNTER WARNING ENABLE
MPEN	MP TALLY COUNTER WARNING ENABLE
RFOEN	RECEIVE FIFO OVERRUN ENABLE

Bit	Description
15	Must be 0.
14	**BREN: BUS RETRY OCCURRED enabled:** 0: disable 1: enables interrupts when a Bus Retry operation is requested.
13	**HBLEN: HEARTBEAT LOST enable:** 0: disable 1: enables interrupts when a heartbeat lost condition occurs
12	**LCDEN: LOAD CAM DONE INTERRUPT enable:** 0: disable 1: enables interrupts when the Load CAM command has finished
11	**PINTEN: PROGRAMMABLE INTERRUPT enable:** 0: disable 1: enables programmable interrupts to occur when the PINTR bit the TXpkt.config field is set to a "1".
10	**PRXEN: PACKET RECEIVED enable:** 0: disable 1: enables interrupts for packets accepted.
9	**PTXEN: PACKET TRANSMITTED OK enable:** 0: disable 1: enables interrupts for transmit completions
8	**TXEREN: TRANSMIT ERROR enable:** 0: disable 1: enables interrupts for packets transmitted with error.

4.0 SONIC Registers (Continued)

4.3.5 Interrupt Mask Register (Continued)
(RA<5:0> = 4h)

Bit	Description
7	**TCEN: GENERAL PURPOSE TIMER COMPLETE enable:** 0: disable 1: enables interrupts when the general purpose timer has rolled over from 0000 0000h to FFFF FFFFh.
6	**RDEEN: RECEIVE DESCRIPTORS EXHAUSTED enable:** 0: disable 1: enables interrupts when all receive descriptors in the RDA have been exhausted.
5	**RBEEN: RECEIVE BUFFERS EXHAUSTED enable:** 0: disable 1: enables interrupts when all resource descriptors in the RRA have been exhausted.
4	**RBAEEN: RECEIVE BUFFER AREA EXCEEDED enable:** 0: disable 1: enables interrupts when the SONIC attempts to buffer data beyond the end of the Receive Buffer Area.
3	**CRCEN: CRC TALLY COUNTER WARNING enable:** 0: disable 1: enables interrupts when the CRC tally counter has rolled over from FFFFh to 0000h.
2	**FAEEN: FRAME ALIGNMENT ERROR (FAE) TALLY COUNTER WARNING enable:** 0: disable 1: enables interrupts when the FAE tally counter rolled over from FFFFh to 0000h.
1	**MPEN: MISSED PACKET (MP) TALLY COUNTER WARNING enable:** 0: disable 1: enables interrupts when the MP tally counter has rolled over from FFFFh to 0000h.
0	**RFOEN: RECEIVE FIFO OVERRUN enable:** 0: disable 1: enables interrupts when the receive FIFO has overrun.

4.0 SONIC Registers (Continued)

4.3.6 Interrupt Status Register

(RA<5:0> = 5h)

This register *(Figure 4-9)* indicates the source of an interrupt when the INT pin goes active. Enabling the corresponding bits in the IMR allows bits in this register to produce an interrupt. When an interrupt is active, one or more bits in this register are set to a "1". A bit is cleared by writing "1" to it. Writing a "0" to any bit has no effect.

This register is cleared by a hardware reset and unaffected by a software reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	BR	HBL	LCD	PINT	PKTRX	PTDN	TXER	TC	RDE	RBE	RBAE	CRC	FAE	MP	RFO
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

r/w = read/write

FIGURE 4-9. Interrupt Status Register

Field	Meaning
BR	BUS RETRY OCCURRED
HBL	CD HEARTBEAT LOST
LCD	LOAD CAM DONE
PINT	PROGRAMMABLE INTERRUPT
PKTRX	PACKET RECEIVED
TXDN	TRANSMISSION DONE
TXER	TRANSMIT ERROR
TC	TIMER COMPLETE
RDE	RECEIVE DISCRIPTORS EXHAUSTED
RBE	RECEIVE BUFFERS EXHAUSTED
RBAE	RECEIVE BUFFER AREA EXCEEDED
CRC	CRC TALLY COUNTER ROLLOVER
FAE	FRAME ALIGNMENT ERROR
MP	MISSED PACKET COUNTER ROLLOVER
RFO	RECEIVE FIFO OVERRUN

Bit	Description
15	Must be 0.
14	**BR: BUS RETRY OCCURRED** Indicates that a Bus Retry ($\overline{BRT}$) operation has occurred. In Latched Bus Retry mode (LBR in the DCR), BR will only be set when the SONIC is a bus master. Before the SONIC will continue any DMA operations, BR must be cleared. In Unlatched mode, the BR bit should be cleared also, but the SONIC will not wait for BR to be cleared before requesting the bus again and continuing its DMA operations. (See sections 4.3.2 and 5.4.6 for more information on Bus Retry).
13	**HBL: CD HEARTBEAT LOST** If the transceiver fails to provide a collision pulse (heart beat) during the first 6.4 μs of the Interframe Gap after transmission, this bit is set.
12	**LCD: LOAD CAM DONE** Indicates that the Load CAM command has finished writing to all programmed locations in the CAM. (See section 4.1.1.)
11	**PINT: PROGRAMMED INTERRUPT** Indicates that upon reading the TXpkt.config field, the SONIC has detected the PINTR bit to be set. (See section 4.3.4.)
10	**PKTRX: PACKET RECEIVED** Indicates that a packet has been received and been buffered to memory. This bit is set after the RXpkt.seq_no field is written to memory.
9	**TXDN: TRANSMISSION DONE** Indicates that either (1) there are no remaining packets to be transmitted in the Transmit Descriptor Area (i.e., the EOL bit has been detected as a "1"), (2) the Halt Transmit command has been given (HTX bit in CR is set to a "1"), or (3) a transmit abort condition has occurred. This condition occurs when any of following bits in the TCR are set: BCM, EXC, FU, or EXD. This bit is set after the TXpkt.status field has been written to.

4.0 SONIC Registers (Continued)

4.3.6 Interrupt Status Register (Continued)
(RA<5:0> = 5h)

Bit	Description
8	**TXER: TRANSMIT ERROR** Indicates that a packet has been transmitted with at least one of the following errors. —Byte count mismatch (BCM) —Excessive collisions (EXC) —FIFO underrun (FU) —Excessive deferral (EXD) The TXpkt.status field reveals the cause of the error(s).
7	**TC: GENERAL PURPOSE TIMER COMPLETE** Indicates that the timer has rolled over from 0000 0000h to FFFF FFFFh. (See section 4.3.12.)
6	**RDE: RECEIVE DESCRIPTORS EXHAUSTED** Indicates that all receive packet descriptors in the RDA have been exhausted. This bit is set when the SONIC detects EOL = 1. (See section 3.4.7.)
5	**RBE: RECEIVE BUFFER EXHAUSTED** Indicates that the SONIC has detected the Resource Read Pointer (RRP) is equal to the Resource Write Pointer (RWP). This bit is set after the last field is read from the resource area. (See section 3.4.7.) **Note 1:** This bit will be set as the SONIC finishes using the second to last receive buffer and reads the last RRA descriptor. This gives the system an early warning of impending no resources. **Note 2:** The SONIC will stop reception of packets when the last RBA has been used and will not continue reception until additional receive buffers have been added (i.e., RWP is incremented beyond RRP) and this bit has been reset. **Note 3:** If additional buffers have been added, resetting this bit causes the SONIC to read the next resource descriptor pointed to by the RRP in the Receive Resource Area. Note that resetting this bit under this condition is similar to issuing the Read RRA command (setting the RRRA bit in the Command Register). This bit should never be reset until after the addtional resources have been added to the RRA.
4	**RBAE: RECEIVE BUFFER AREA EXCEEDED** Indicates that during reception, the SONIC has reached the end of the Receive Buffer Area. Reception is aborted and the SONIC fetches the next available resource descriptors in the RRA. The buffer space is not re-used and an RDA is not set up for the truncated packet (see section 3.4.7).
3	**CRC: CRC TALLY COUNTER ROLLOVER** Indicates that the tally counter has rolled over from FFFFh to 0000h. (See section 4.3.11.)
2	**FAE: FRAME ALIGNMENT ERROR (FAE) TALLY COUNTER ROLLOVER** Indicates that the FAE tally counter has rolled over from FFFFh to 0000h. (See section 4.3.11.)
1	**MP: MISSED PACKET (MP) COUNTER ROLLOVER** Indicates that the MP tally counter has rolled over from FFFFh to 0000h. (See section 4.3.11.)
0	**RFO: RECEIVE FIFO OVERRUN** Indicates that the SONIC has been unable to access the bus before the receive FIFO has filled from the network. This condition is due to excessively long bus latency and/or slow bus clock. Note that FIFO underruns are indicated in the TCR. (See section 1.4.1.)

4.0 SONIC Registers (Continued)

4.3.7 Data Configuration Register 2

(RA<5:0> = 3Fh)

This register *(Figure 4-10)* is for enabling the extended bus interface options.

A hardware reset will set all bits in this register to "0" except for the Extended Programmable Outputs which are unknown until written to. A software reset will not affect any bits in this register. This register should only be written to when the SONIC is in software reset (the RST bit in the Command Register is set).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXPO3	EXPO2	EXPO1	EXPO0	0	0	0	0	0	0	0	PH	LRDY	PCM	PCNM	RJCM
r/w	r/w	r/w	r/w								r/w	r/w	r/w	r/w	r/w

FIGURE 4-10. Data Configuration Register 2

Field	Meaning
EXPO3..0	EXTENDED PROGRAMMABLE OUTPUTS
PH	PROGRAM HOLD
LRDY	LATCHED READY
PCM	PACKET COMPRESS WHEN MATCHED
PCNM	PACKET COMPRESS WHEN NOT MATCHED
RJCM	REJECT ON CAM MATCH

Bit	Description
15–12	**EXPO<3:0> EXTENDED PROGRAMMABLE OUTPUTS** These bits program the level of the Extended User outputs (EXUSR<3:0>) when the SONIC is a bus master. Writing a "1" to any of these bits programs a high level to the corresponding output. Writing a "0" to any of these bits programs a low level to the corresponding output. EXUSR<3:0> are similiar to USR<1:0> except that EXUSR<3:0> are only available when the Extended Bus mode is selected (bit 15 in the DCR is set to "1", see section 4.3.2).
11–5	Must be zero.
4	**PH: PROGRAM HOLD** When this bit is set to "0", the HOLD request output is asserted/deasserted from the falling edge of bus clock. If this bit is set to "1", HOLD will be asserted/deasserted ½ clock later on the rising edge of bus clock.
3	**LRDY: LATCHED READY** When this bit is set to a "1", the ready input ($\overline{RDYi}$) is latched to provide a fast setup time (<10 ns) with respect to the rising edge of bus clock. This provides better compatibility with NSC/Intel mode designs. This mode is only useful in synchronous mode when BMODE = 0. **Note:** When this mode is selected, the SONIC will go into a Tx state before entering the next T1 state, hence, there will be an extra clock cycle before the T1 state.
2	**PCM: PACKET COMPRESS WHEN MATCHED** When this bit is set to a "1" (and the PCNM bit is reset to a "0"), the $\overline{PCOMP}$ output will be asserted if the destination address of the packet being received matches one of the entries in the CAM (Content Addressable Memory). This bit, along with PCNM, is used with the Management Bus of the DP83950, Repeater Interface Controller (RIC). See the DP83950 datasheet for more details on the RIC Management Bus. This mode is also called the Managed Bridge Mode. **Note 1:** Setting PCNM and PCM to "1" at the same time is not allowed. **Note 2:** If PCNM and PCM are both "0", the $\overline{PCOMP}$ output will remain TRI-STATE until PCNM or PCM are changed.
1	**PCNM: COMPRESS WHEN NOT MATCHED** When this bit is set to a "1" (and the PCM bit is set to "0"), the $\overline{PCOMP}$ output will be asserted if the destination address of the packet does not match one of the entries in the CAM. See the PCM bit above. This mode is also called the Managed Hub Mode. **Note:** $\overline{PCOMP}$ will not be asserted if the destination address is a broadcast address. This is true regardless of the state of the BRD bit in the Receive Control Register.
0	**RJCM: REJECT ON CAM MATCH** When this bit is set to "1", the SONIC will reject a packet on a CAM match. Setting RJCM to "0" causes the SONIC to operate normally by accepting packets on a CAM match. Setting this mode is useful for a small bridge with a limited number of nodes attached to it. RJCM only affects the CAM, though. Setting RJCM will not invert the function of the BRD, PRO or AMC bits (to accept broadcast, all physical or multicast packets respectively) in the Receive Control Register (see section 4.3.3). This means, for example, that it is not possible to set RJCM and BRD to reject all broadcast packets. If RJCM and BRD are set at the same time, however, all broadcast packets will be accepted, but any packets that have a destination address that matches an address in the CAM will be rejected.

4.0 SONIC Registers (Continued)

4.3.8 Transmit Registers

The transmit registers described in this section are part of the User Register set. The UTDA and CTDA must be initialized prior to issuing the transmit command (setting the TXP bit) in the Command register.

Upper Transmit Descriptor Address Register (UTDA): This register contains the upper address bits (A<31:16>) for accessing the transmit descriptor area (TDA) and is concatenated with the contents of the CTDA when the SONIC accesses the TDA in system memory. The TDA can be as large as 32k words or 16k long words and can be located anywhere in system memory. This register is unaffected by a hardware or software reset.

Current Transmit Descriptor Address Register (CTDA): The 16-bit CTDA register contains the lower address bits (A<15:1>) of the 32-bit transmit descriptor address. During initialization this register must be programmed with the lower address bits of the transmit descriptor. The SONIC concatenates the contents of this register with the contents of the UTDA to point to the transmit descriptor. For 32-bit memory systems, bit 1, corresponding to address signal A1, must be set to "0" for alignment to long-word boundaries. Bit 0 of this register is the End of List (EOL) bit and is used to denote the end of the list. This register is unaffected by a hardware or software reset.

4.3.9 Receive Registers

The receive registers described in this section are part of the User Register set. A software reset has no effect on these registers and a hardware reset only affects the EOBC and RSC registers. The receive registers must be initialized prior to issuing the receive command (setting the RXEN bit) in the Command register.

Upper Receive Descriptor Address Register (URDA): This register contains the upper address bits (A<31:16>) for accessing the receive descriptor area (RDA) and is concatenated with the contents of the CRDA when the SONIC accesses the RDA in system memory. The RDA can be as large as 32k words or 16k long words and can be located anywhere in system memory. This register is unaffected by a hardware or software reset.

Current Receive Descriptor Address Register (CRDA): The CRDA is a 16-bit read/write register used to locate the received packet descriptor block within the RDA. It contains the lower address bits (A<15:1>). The SONIC concatenates the contents of the CRDA with the contents of the URDA to form the complete 32-bit address. The resulting 32-bit address points to the first field of the descriptor block. For 32-bit memory systems, bit 1, corresponding to address signal A1, must be set to "0" for alignment to long-word boundaries. Bit 0 of this register is the End of List (EOL) bit and is used to denote the end of the list. This register is unaffected by a hardware or software reset.

End of Buffer Word Count Register (EOBC): The SONIC uses the contents of this register to determine where to place the next packet. At the end of packet reception, the SONIC compares the contents of the EOBC register with the contents of the Remaining Buffer Word Count registers (RBWC0,1) to determine whether: (1) to place the next packet in the same RBA or (2) to place the next packet in another RBA. If the EOBC is less than or equal to the remaining number of words in the RBA after a packet is received (i.e., EOBC $\leq$ RBWC0,1), the SONIC buffers the next packet in the same RBA. If the EOBC is greater than the remaining number of words in the RBA after a packet is received (i.e., EOBC > RBWC0,1), the Last Packet in RBA bit, LPKT in the Receive Control Register, section 4.3.3, is set and the SONIC fetches the next resource descriptor. Hence, the next packet received will be buffered in a new RBA. A hardware reset sets this register to 02F8H (760 words or 1520 bytes). See sections 3.4.2 and 3.4.4.4 for more information about using EOBC.

Upper Receive Resource Address Register (URRA): The URRA is a 16-bit read/write register. It is programmed with the base address of the receive resource area (RRA). This 16-bit upper address value (A<31:16>) locates the receive resource area in system memory. SONIC uses the URRA register when accessing the receive descriptors within the RRA by concatenating the lower address value from one of four receive resource registers (RSA, REA, RWP, or RRP).

Resource Start Address Register (RSA): The RSA is a 15-bit read/write register. The LSB is not used and always reads back as a 0. The RSA is programmed with the lower 15-bit address (A<15:1>) of the starting address of the receive resource area. SONIC concatenates the contents of this register with the contents of the URRA to form the complete 32-bit address.

Resource End Address Register (REA): The REA is a 15-bit read/write register. The LSB is not used and always reads back as a 0. The REA is programmed with the lower 15-bit address (A<15:1>) of the ending address of the receive resource area. SONIC concatenates the contents of this register with the contents of the URRA to form the complete 32-bit address.

Resource Read Pointer Register (RRP): The RRP is a 15-bit read/write register. The LSB is not used and always reads back as a 0. The RRP is programmed with the lower 15-bit address (A<15:1>) of the first field of the next descriptor the SONIC will read. SONIC concatenates the contents of this register with the contents of the URRA to form the complete 32-bit address.

Resource Write Pointer Register (RWP): The RWP is a 15-bit read/write register. The LSB is not used and always reads back as a 0. The RWP is programmed with the lower 15-bit address (A<15:1>) of the next available location the system can add a descriptor. SONIC concatenates the contents of this register with the contents of the URRA to form the complete 32-bit address. In 32-bit mode, bit 1, corresponding to address signal A1, must be zero to insure the proper equality comparison between this register and the RRP register.

Receive Sequence Counter Register (RSC): This is a 16-bit read/write register containing two fields. The SONIC uses this register to provide status information on the number of packets within a RBA and the number of RBAs. The RSC register contains two 8-bit (modulo 256) counters. After each packet is received the packet sequence number is incremented. The SONIC maintains a single sequence number for each RBA. When the SONIC uses the next RBA, the packet sequence number is reset to zero and the RBA sequence number is incremented. This register is reset to 0 by a hardware reset or by writing zero to it. A software reset has no affect.

15 8	7 0
RBA Sequence Number (modulo 256)	Packet Sequence Number (Modulo 256)

4.0 SONIC Registers (Continued)

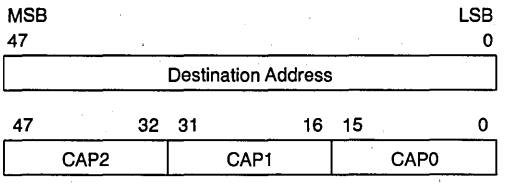
4.3.10 CAM Registers

The CAM registers described in this section are part of the User Register set. They are used to program the Content Addressable Memory (CAM) entries that provide address filtering of packets. These registers, except for the CAM Enable register, are unaffected by a hardware or software reset.

CAM Entry Pointer Register (CEP): The CEP is a 4-bit register used by SONIC to select one of the sixteen CAM entries. SONIC uses the least significant 4-bits of this register. The value of 0h points to the first CAM entry and the value of Fh points to the last entry.

CAM Address Port 2, 1, 0 Registers (CAP2, CAP1, CAP0): Each CAP is a 16-bit read-only register used to access the CAM cells. Each CAM cell is 16-bits wide and contains one third of the 48-bit CAM entry which is used by the SONIC for address filtering. The CAP2 register is used to access the upper bits (<47:32>), CAP1 the middle bits (<31:16>) and CAP0 the lower bits (<15:0>) of the CAM entry. Given the physical address 10:20:30:40:50:60, which is made up of 6 octets or bytes, where 10h is the least significant byte and 60h is the most significant byte (10h would be the first byte received from the network and 60h would be the last), CAP0 would be loaded with 2010h, CAP1 with 4030h and CAP2 with 6050h.

To read a CAM entry, the user first places the SONIC in software reset (set the RST bit in the Command register), programs the CEP register to select one of sixteen CAM entries, then reads CAP2, CAP1, and CAP0 to obtain the complete 48-bit entry. The user can not write to the CAM entries directly. Instead, the user programs the CAM descriptor area in system memory (see section 4.1.1), then issues the Load CAM command (setting LCAM bit in the Command register). This causes the SONIC to read the descriptors from memory and loads the corresponding CAM entry through CAP2-0.



CAM Enable Register (CE): The CE is a 16-bit read/write register used to mask out or enable individual CAM entries. Each register bit position corresponds to a CAM entry. When a register bit is set to a "1" the corresponding CAM entry is enabled. When "0" the entry is disabled. This register is unaffected by a software reset and cleared to zero (disabling all entries) during a hardware reset. Under normal operations the user does not access this register. Instead the user sets up this register through the last entry in the CAM descriptor area. The SONIC loads the CE register during execution of the LCAM Command.

CAM Descriptor Pointer Register (CDP): The CDP is a 15-bit read/write register. The LSB is unused and always reads back as 0. The CDP is programmed with the lower address (A<15:1>) of the first field of the CAM descriptor block in the CAM descriptor area (CDA) of system memory. SONIC uses the contents of the CDP register when accessing the CAM descriptors. This register must be programmed by the user before issuing the LCAM command. During execution of the LCAM Command SONIC concatenates the contents of this register with the contents of the URRA register to form the complete 32-bit address. During the Load CAM operation this register is incremented to address the fields in the CDA. After the Load Command completes this register points to the next location after the CAM Descriptor Area.

CAM Descriptor Count Register (CDC): The CDC is a 5-bit read/write register. It is programmed with the number of CAM descriptor blocks in the CAM descriptor area. This register must be programmed by the user before issuing the LCAM command. SONIC uses the value in this register to determine how many entries to place in the CAM during execution of the LCAM command. During LCAM execution SONIC decrements this register each time it reads a descriptor block. When the CDC decrements to zero SONIC terminates the LCAM execution. Since the CDC register is programmed with the number of CAM descriptor blocks in the CAM Descriptor Area, the value programmed into the CDC register ranges 1 to 16 (1h to 10h).

4.3.11 Tally Counters

The SONIC provides three 16-bit counters used for monitoring network statistics on the number of CRC errors, Frame Alignment errors, and missed packets. These registers rollover after the count of FFFFh is reached and produce an interrupt if enabled in the Interrupt Mask Register (IMR). These counters are unaffected by the RXEN bit in the CR, but are halted when the RST bit in the CR is set. The data written to these registers is inverted before being latched. This means that if a value of FFFFh is written to these registers by the system, they will contain and read back the value 0000h. Data is not inverted during a read operation. The Tally registers, therefore, are cleared by writing all "1's" to them. A software or hardware reset does not affect the tally counters.

CRC Tally Counter Register (CRCT): The CRCT is a 16-bit read/write register. This register is used to keep track of the number of packets received with CRC errors. After a packet is accepted by the address recognition logic, this register is incremented if a CRC error is detected. If the packet also contains a Frame Alignment error, this counter is not incremented.

FAE Tally Counter Register (FAET): The FAET is a 16-bit read/write register. This register is used to keep track of the number of packets received with frame alignment errors. After a packet is accepted by the address recognition logic, this register is incremented if a FAE error is detected.

Missed Packet Tally Counter Register (MPT): The MPT is a 16-bit read/write register. After a packet is received, this counter is incremented if there is: (1) lack of memory resources to buffer the packet, (2) a FIFO overrun, or (3) a valid packet has been received, but the receiver is disabled (RXDIS is set in the command register).

4.0 SONIC Registers (Continued)

4.3.12 General Purpose Timer

The SONIC contains a 32-bit general-purpose watchdog timer for timing user-definable events. This timer is accessed by the user through two 16-bit read/write registers (WT1 and WT0). The lower count value is programmed through the WT0 register and the upper count value is programmed through the WT1 register.

These two registers are concatenated together to form the complete 32-bit timer. This timer, clocked at ½ the Transmit Clock (TXC) frequency, counts down from its programmed value and generates an interrupt, if enabled (Interrupt Mask register), when it rolls over from 0000 0000h to FFFF FFFFh. When the counter rolls over it continues decrementing unless explicitly stopped (setting the STP bit). The timer is controlled by the ST (Start Timer) and STP (Stop Timer) bits in the Command register. A hardware or software reset halts, but does not clear, the General Purpose timer.

31 16	15 0
WT1 (Upper Count Value)	WT0 (Lower Count Value)

4.3.13 Silicon Revision Register

This is a 16-bit read only register. It contains information on the current revision of the SONIC. The initial silicon begins at 0000h and subsequent revision will be incremented by one.

5.0 Bus Interface

SONIC features a high speed non-multiplexed address and data bus designed for a wide range of system environments. The data bus can be programmed (via the Data Configuration Register) to a width of either 32- or 16-bits. SONIC contains an on-chip DMA and supplies all the necessary signals for DMA operation. With 31 address lines SONIC can access a full 2 G-word address space. To accommodate different memory speeds wait states can be added to the bus cycle by two methods. The memory subsystem can add wait states by simply withholding the appropriate handshake signals. In addition, the SONIC can be programmed (via the Data Configuration Register) to add wait states.

The SONIC is designed to interface to both the NSC/Intel and Motorola style buses. To facilitate minimum chip count designs and complete bus compatibility the user can program the SONIC for the following bus modes:

— NSC/Intel bus operating in synchronous mode

— NSC/Intel bus operating in asynchronous mode

— Motorola bus operating in synchronous mode

— Motorola bus operating in asynchronous mode

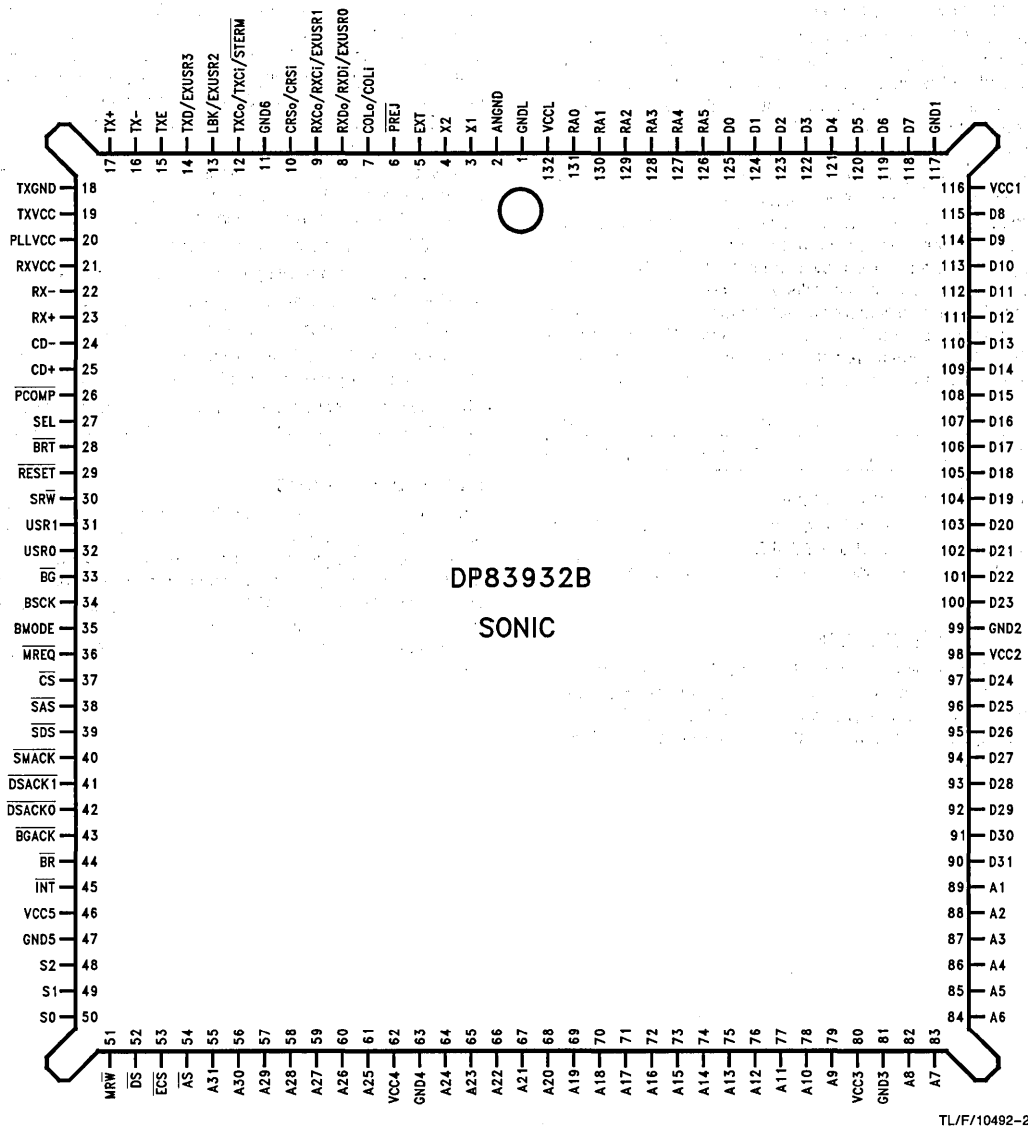
The mode pin (BMODE) along with the SBUS bit in the Data Configuration Register are used to select the bus mode.

This section describes the SONIC's pin signals, provides system interface examples, and describes the various SONIC bus operations.

5.1 PIN CONFIGURATIONS

There are two user selectable pin configurations for SONIC to provide the proper interface signals for either the NSC/Intel or Motorola style buses. The state of the BMODE pin is used to define the pin configuration. Figure 5-1 shows the pin configuration when BMODE = 1 (tied to V_{CC}) for the Motorola style bus. Figure 5-2 shows the pin configuration when BMODE = 0 (tied to ground) for the NSC/Intel style bus.

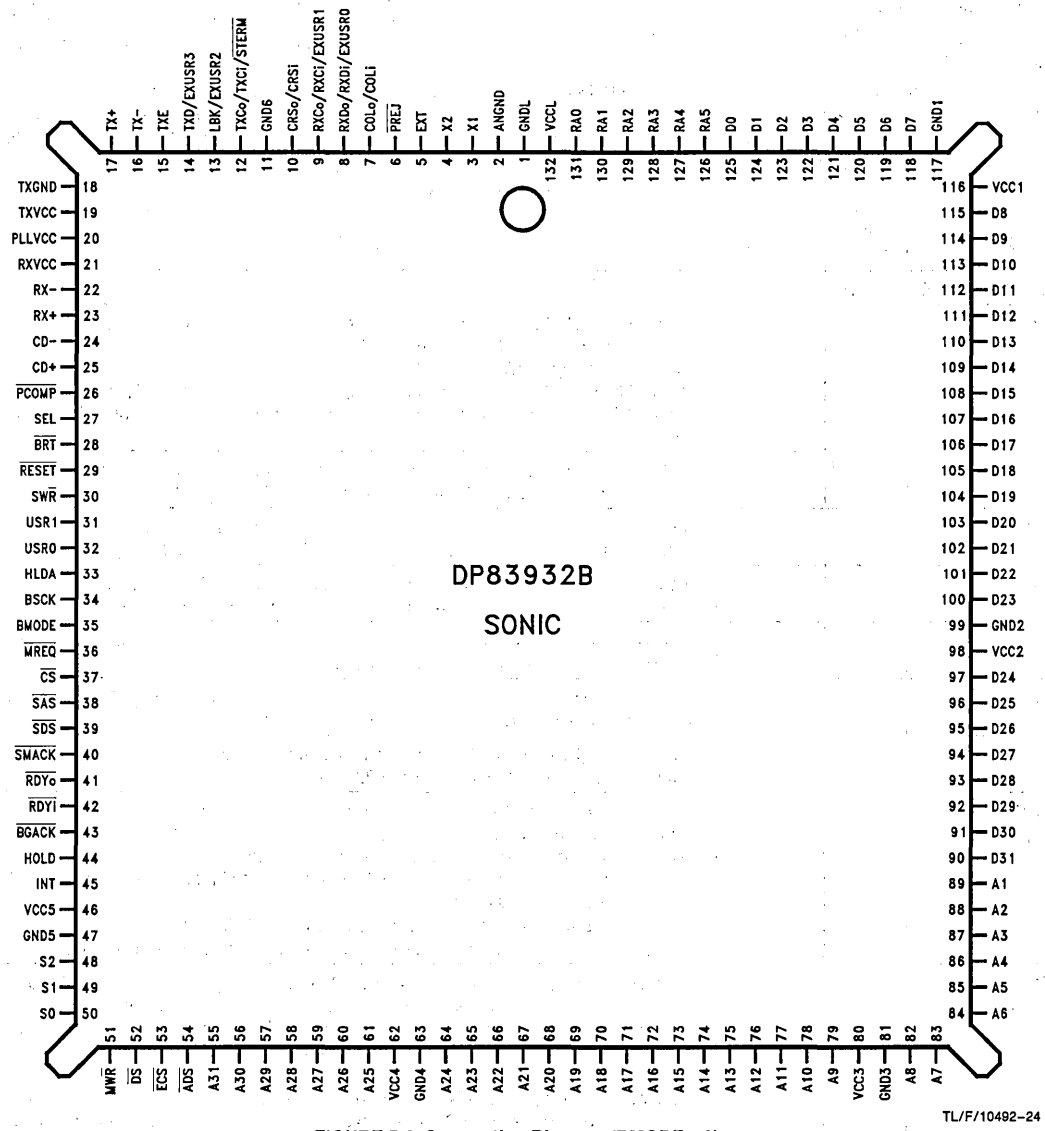
5.0 Bus Interface (Continued)



TL/F/10492–23

FIGURE 5-1. Connection Diagram (BMODE = 1)

5.0 Bus Interface (Continued)



TL/F/10492–24

FIGURE 5-2. Connection Diagram (BMODE = 0)

5.0 Bus Interface (Continued)

5.2 PIN DESCRIPTION

I = input, O = output, and Z = TRI-STATE

Inputs are TTL compatible

ECL = ECL-like drivers for interfacing to the AUI interface.

TP = Totem pole like drivers. These drivers are driven either high or low and are always driven. Drive levels are CMOS compatible.

TRI = TRI-STATE drivers. These pins are driven high, low or TRI-STATE. Drive levels are CMOS compatible. These pins may also be inputs (depending on the pin).

OC = Open Collector type drivers. These drivers are TRI-STATE when inactive and are driven low when active. These pins may also be inputs (depending on the pin).

TABLE 5-1. Pin Description

Symbol	Driver Type	Direction	Description
NETWORK INTERFACE PINS			
EXT		I	**External ENDEC Select:** Tying this pin to V_{CC} (EXT = 1) disables the internal ENDEC and allows an external ENDEC to be used. Tying this pin to ground (EXT = 0) enables the internal ENDEC. This pin must be tied either to V_{CC} or ground. Note the alternate pin definitions for CRSo/CRSi, COLo/COLi, RXDo/RXDi, RXCo/RXCi, and TXCo/TXCi. When EXT = 0 the first pin definition is used and when EXT = 1 the second pin definition is used.
CD+		I	**Collision +:** The positive differential collision input from the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1).
CD−		I	**Collision −:** The negative differential collision input from the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1).
RX+		I	**Receive +:** The positive differential receive data input from the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1)
RX−		I	**Receive −:** The negative differential receive data input from the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1)
TX+	ECL	O	**Transmit +:** The positive differential transmit output to the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1).
TX−	ECL	O	**Transmit −:** The negative differential transmit output to the transceiver. This pin should be unconnected when an external ENDEC is selected (EXT = 1).
CRSo CRSi	TP	O I	**Carrier Sense Output (CRSo)** from the internal ENDEC (EXT = 0): When EXT = 0 the CRSo signal is internally connected between the ENDEC and MAC units. It is asserted on the first valid high-to-low transition in the receive data (RX+/−). This signal remains active 1.5 bit times after the last bit of data. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Carrier Sense Input (CRSi)** from an external ENDEC (EXT = 1): The CRSi signal is activated high when the external ENDEC detects valid data at its receive inputs.
COLo COLi	TP	O I	**Collision Output (COLo)** from the internal ENDEC (EXT = 0): When EXT = 0 the COLo signal is internally connected between the ENDEC and MAC units. This signal generates an active high signal when the 10 MHz collision signal from the transceiver is detected. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Collision Detect Input (COLi)** from an external ENDEC (EXT = 1): The COLi signal is activated from an external ENDEC when a collision is detected. This pin is monitored during transmissions from the beginning of the Start Of Frame Delimiter (SFD) to the end of the packet. At the end of transmission, this signal is monitored by the SONIC for CD heartbeat.

5.0 Bus Interface (Continued)

TABLE 5-1. Pin Description (Continued)

Symbol	Driver Type	Direction	Description
NETWORK INTERFACE PINS (Continued)			
RXDo RXDi EXUSR0	TP TRI	O I O, Z	This pin will be TRI-STATE until the DCR has been written to. (See section 4.3.2, EXBUS, for more information.) **Receive Data Output (RXDo)** from the internal ENDEC (EXT = 0): NRZ data output. When EXT = 0 the RXDOUT signal is internally connected between the ENDEC and MAC units. This signal must be sampled on the rising edge of the receive clock output (RXCo). Although this signal is used internally by the SONIC it is also provided as an output to the user. **Receive Data Input (RXDi)** from an external ENDEC (EXT = 1): The NRZ data decoded from the external ENDEC. This data is clocked in on the rising edge of RXCi. **Extended User Output (EXUSR0):** When EXBUS has been set (see section 4.3.2), this pin becomes a programmable output. It will remain TRI-STATE until the SONIC becomes a bus master, at which time it will be driven according to the value programmed in the DCR2 (Section 4.3.7).
RXCo RXCi EXUSR1	TP TRI	O I O, Z	This pin will be TRI-STATE until the DCR has been written to. (See section 4.3.2, EXBUS, for more information.) **Receive Clock Output (RXCo)** from the internal ENDEC (EXT = 0): When EXT = 0 the RXCo signal is internally connected between the ENDEC and MAC units. This signal is the separated receive clock from the Manchester data stream. It remains active 5-bit times after the deassertion of CRSo. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Receive Clock Input (RXCI)** from an external ENDEC (EXT = 1): The separated received clock from the Manchester data stream. This signal is generated from an external ENDEC. **Extended User Output (EXUSR1):** When EXBUS has been set (see section 4.3.2), this pin becomes a programmable output. It will remain TRI-STATE until the SONIC becomes a bus master, at which time it will be driven according to the value programmed in the DCR2 (Section 4.3.7).
TXD EXUSR3	TP TRI	O O, Z	This pin will be TRI-STATE until the DCR has been written to. (See section 4.3.2, EXBUS, for more information.) **Transmit Data (TXD):** The serial NRZ data from the MAC unit which is to be decoded by an external ENDEC. Data is valid on the rising edge of TXC. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Extended User Output (EXUSR3):** When EXBUS has been set (see section 4.3.2), this pin becomes a programmable output. It will remain TRI-STATE until the SONIC becomes a bus master, at which time it will be driven according to the value programmed in the DCR2 (Section 4.3.7).
TXE	TP	O	**Transmit Enable:** This pin is driven high when the SONIC begins transmission and remains active until the last byte is transmitted. Although this signal is used internally by the SONIC it is also provided as an output to the user.
TXCo TXCi $\overline{\text{STERM}}$	TRI	O, Z I I	This pin will be TRI-STATE until the DCR has been written to. (See section 4.3.2, EXBUS, for more information.) **Transmit Clock Output (TXCo)** from the internal ENDEC (EXT = 0): This 10 MHz clock transmit clock output is derived from the 20 MHz oscillator. When EXT = 0 the TXCOUT signal is internally connected between the ENDEC and MAC units. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Transmit Clock Input (TXCi)** (EXT = 1): This input clock from an external ENDEC is used for shifting data out of the MAC unit serializer. This clock is nominally 10 MHz. **Synchronous Termination ($\overline{\text{STERM}}$):** When the SONIC is a bus master, it samples this pin before terminating its memory cycle. This pin is sampled synchronously and may only be used in asynchronous bus mode when BMODE = 1. See section 5.4.5 for more details.

5.0 Bus Interface (Continued)

TABLE 5-1. Pin Description (Continued)

Symbol	Driver Type	Direction	Description
NETWORK INTERFACE PINS (Continued)			
LBK EXUSR2	TP TRI	O O, Z	This pin will be TRI-STATE until the DCR has been written to. (See section 4.3.2, EXBUS, for more information.) **Loopback (LBK):** When ENDEC loopback is programmed, this pin is asserted high. Although this signal is used internally by the SONIC it is also provided as an output to the user. **Extended User Output (EXUSR2):** When EXBUS has been set (see section 4.3.2), this pin becomes a programmable output. It will remain TRI-STATE until the SONIC becomes a bus master, at which time it will be driven according to the value programmed in the DCR2 (Section 4.3.7).
$\overline{PCOMP}$	TRI	O, Z	**Packet Compression:** This pin is used with the Management Bus of the DP83950, Repeater Interface Controller (RIC). The SONIC can be programmed to assert $\overline{PCOMP}$ whenever there is a CAM match, or when there is not a match. The RIC uses this signal to compress (shorten) a received packet for management purposes and to reduce memory usage. (See the DP83950 datasheet for more details on the RIC Management Bus.) The operation of this pin is controlled by bits 1 and 2 in the DCR2 register. $\overline{PCOMP}$ will remain TRI-STATE until these bits are written to.
SEL		I	**Mode Select** (EXT = 0): This pin is used to determine the voltage relationship between TX+ and TX− during idle at the primary of the isolation transformer on the network interface. When tied to V_{CC}, TX+ and TX− are at equal voltages during idle. When tied to ground, the voltage at TX+ is positive with respect to TX− during idle on the primary side of the isolation transformer (*Figure 6-2*).
$\overline{PREJ}$		I	**Packet Reject:** This signal is used to reject received packets. When asserted low for at least two receive clocks (RXC), the SONIC will reject the incoming packet. This pin can be asserted up to the 2nd to the last bit of reception to reject a packet.
X1		I	**Crystal or External Oscillator Input:** This signal is used to provide clocking signals for the internal ENDEC. A crystal can be connected to this pin along with X2, or an oscillator module may be used. Typically the output of an oscillator module is connected to this pin. See section 6.1.3 for more information about using oscillators or crystals.
X2	TP	I, O	**Crystal Feedback Output:** This signal is used to provide clocking signals for the internal ENDEC. A crystal may be connected to this pin along with X1, or an oscillator module may be used. See section 6.1.3 for more information about using oscillator modules or crystals.
BUS INTERFACE PINS			
BMODE		I	**Bus Mode:** This input enables the SONIC to be compatible with standard microprocessor buses. The level of this pin affects byte ordering (little or big endian) and controls the operation of the bus interface control signals. A high level (tied to V_{CC}) selects Motorola mode (big endian) and a low level (tied to ground) selects NSC/Intel mode (little endian). Note the alternate pin definitions for $\overline{AS}/\overline{ADS}$, $MR\overline{W}/MW\overline{R}$, $\overline{INT}/INT$, $B\overline{R}/HOLD$, $B\overline{G}/HLDA$, $SR\overline{W}/SW\overline{R}$, $\overline{DSACK0}/RDYi$, and $\overline{DSACK1}/RDYo$. When BMODE = 1 the first pin definition is used and when BMODE = 0 the second pin definition is used. See sections 5.4.1, 5.4.4, and 5.4.5.

5.0 Bus Interface (Continued)

TABLE 5-1. Pin Description (Continued)

Symbol	Driver Type	Direction	Description
BUS INTERFACE PINS (Continued)			
D31–D0	TRI	I, O, Z	**Data Bus:** These bidirectional lines are used to transfer data on the system bus. When the SONIC is a bus master, 16-bit data is transferred on D15–D0 and 32-bit data is transferred on D31–D0. When the SONIC is accessed as a slave, register data is driven onto lines D15–D0 and D31–D16 are held TRI-STATE.
A31–A1	TRI	O, Z	**Address Bus:** These signals are used by the SONIC to drive the DMA address after the SONIC has acquired the bus. Since the SONIC aligns data to word boundaries, only 31 address lines are needed.
RA5–RA0		I	**Register Address Bus:** These signals are used to access SONIC's internal registers. When the SONIC is accessed, the CPU drives these lines to select the desired SONIC register.
$\overline{AS}$ $\overline{ADS}$	TRI TRI	O, Z O, Z	**Address Strobe ($\overline{AS}$):** When BMODE = 1, the falling edge indicates valid status and address. The rising edge indicates the termination of the memory cycle. **Address Strobe ($\overline{ADS}$):** When BMODE = 0, the rising edge indicates valid status and address.
MR$\overline{W}$ MW$\overline{R}$	TRI TRI	O, Z O, Z	When the SONIC has acquired the bus, this signal indicates the direction of data. **Memory Read/Write Strobe (MR$\overline{W}$):** When BMODE = 1, this signal is high during a read cycle and low during a write cycle. **Memory Read/Write Strobe (MW$\overline{R}$):** When BMODE = 0, the signal is low during a read cycle and high during a write cycle.
$\overline{INT}$ INT	OC TP	O, Z O	Indicates that an interrupt (if enabled) is pending from one of the sources indicated by the Interrupt Status register. Interrupts that are disabled in the Interrupt Mask register will not activate this signal. **Interrupt ($\overline{INT}$):** This signal is active low when BMODE = 1. **Interrupt (INT):** This signal is active high when BMODE = 0.
$\overline{RESET}$		I	**Reset:** This signal is used to hardware reset the SONIC. When asserted low, the SONIC transitions into the reset state after 10 transmit clocks or 10 bus clocks if the bus clock period is greater than the transmit clock period.
S2–S0	TP	O	**Bus Status:** These three signals provide a continuous status of the current SONIC bus operations. See section 5.4.3 for status definitions.
BSCK		I	**Bus Clock:** This clock provides the timing for the SONIC DMA engine.
$\overline{BR}$ HOLD	OC TP	O, Z O	**Bus Request ($\overline{BR}$):** When BMODE = 1, the SONIC asserts this pin low when it attempts to gain access to the bus. When inactive this signal is tri-stated. **Hold Request (HOLD):** When BMODE = 0, the SONIC drives this pin high when it intends to use the bus and is driven low when inactive.
$\overline{BG}$ HLDA		I I	**Bus Grant (BG):** When BMODE = 1 this signal is a bus grant. The system asserts this pin low to indicate potential mastership of the bus. **Hold Acknowledge (HLDA):** When BMODE = 0 this signal is used to inform the SONIC that it has attained the bus. When the system asserts this pin high, the SONIC has gained ownership of the bus.
$\overline{BGACK}$	TRI	O, Z	**Bus Grant Acknowledge:** When BMODE = 1, the SONIC asserts this pin low when it has determined that it can gain ownership of the bus. The SONIC checks the following signal before driving $\overline{BGACK}$. 1) $\overline{BG}$ has been received through the bus arbitration process. 2) $\overline{AS}$ is deasserted, indicating that the CPU has finished using the bus. 3) $\overline{DSACK0}$ and $\overline{DSACK1}$ are deasserted, indicating that the previous slave device is off the bus. 4) $\overline{BGACK}$ is deasserted, indicating that the previous master is off the bus. This pin is only used when BMODE = 1.

5.0 Bus Interface (Continued)

TABLE 5-1. Pin Description (Continued)

Symbol	Driver Type	Direction	Description
BUS INTERFACE PINS (Continued)			
$\overline{CS}$		I	**Chip Select:** The system asserts this pin low to access the SONIC's registers. The registers are selected by placing an address on lines RA5–RA0. **Note:** Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently. If these signals are successively asserted, there must be at least two bus clocks between the deasserting edge of the first signal and the asserting edge of the second signal.
$\overline{SAS}$		I	**Slave Address Strobe:** The system asserts this pin to latch the register address on lines RA0–RA5. When BMODE = 1, the address is latched on the falling edge of $\overline{SAS}$. When BMODE = 0 the address is latched on the rising edge of $\overline{SAS}$.
$\overline{SDS}$		I	**Slave Data Strobe:** The system asserts this pin to indicate valid data is on the bus during a register write operation or when data may be driven onto the bus during a register read operation. **Note:** In the DP83932, $\overline{SDS}$ was used only in Motorola mode slave accesses to end the bus cycle by causing the deassertion of $\overline{DSACK0,1}$, $\overline{SMACK}$ and the data, D<15:0>. It served no other function. In the DP83932B (and the DP83932A), however, $\overline{SAS}$ now accomplishes the same function, hence, $\overline{SDS}$ is no longer needed, and does not have to be driven ($\overline{SAS}$ must be driven instead). This change should not cause any compatibility problems with older versions of the SONIC.
SR$\overline{W}$ SW$\overline{R}$		I I	The system asserts this pin to indicate whether it will read from or write to the SONIC's registers. **Slave Read/Write (SR$\overline{W}$):** When BMODE = 1, this signal is asserted high during a read and low during a write. **Slave Read/Write Strobe (SW$\overline{R}$):** when BMODE = 0, this signal is asserted low during a read and high during a write.
$\overline{DS}$	TRI	O, Z	**Data Strobe:** When the SONIC is bus master, it drives this pin low during a read cycle to indicate that the slave device may drive data onto the bus; in a write cycle, this pin indicates that the SONIC has placed valid data onto the bus.
$\overline{DSACK0}$ $\overline{RDYi}$ $\overline{DSACK1}$ $\overline{RDYo}$	TRI TRI TRI	I, O, Z I I, O, Z O, Z	**Data and Size Acknowledge 0 and 1** ($\overline{DSACK0,1}$ BMODE = 1): These pins are the output slave acknowledge to the system when the SONIC registers have been accessed and the input slave acknowledgement when the SONIC is busmaster. When a register has been accessed, the SONIC drives the $\overline{DSACK0,1}$ pins low to terminate the slave cycle. (Note that the SONIC responds as a 32-bit peripheral, but drives data only on lines D0–D15). When the SONIC is bus master, it samples these pins before terminating its memory cycle. These pins are sampled synchronously or asynchronously depending on the state of the SBUS bit in the Data Configuration register. See section 5.4.5 for details. Note that the SONIC does not allow dynamic bus sizing. Bus size is statically defined in the Data Configuration register (see section 4.3.2). **Ready Input** ($\overline{RDYi}$, BMODE = 0): When the SONIC is a bus master, the system asserts this signal high to insert wait-states and low to terminate the memory cycle. This signal is sampled synchronously or asynchronously depending on the state of the SBUS bit. See section 5.4.5 and 4.3.2 for details. **Ready Output** ($\overline{RDYo}$, BMODE = 0): When a register is accessed, the SONIC asserts this signal to terminate the slave cycle.
$\overline{BRT}$		I	**Bus Retry:** When the SONIC is bus master, the system asserts this signal to rectify a potentially correctable bus error. This pin has 2 modes. Mode 1 (the LBR in the Data Configuration register is set to 0): Assertion of this pin forces the SONIC to terminate the current bus cycle and will repeat the same cycle after $\overline{BRT}$ has been deasserted. Mode 2 (the LBR bit in the Data Configuration register is set to 1): Assertion of this signal forces the SONIC to retry the bus operation as in Mode 1. However, the SONIC will not continue DMA operations until the BR bit in the ISR is reset.
$\overline{ECS}$	TRI	O	**Early Cycle Start:** This output gives the system earliest indication that a memory operation is occurring. This signal is driven low at the rising edge of T1 and high at the falling edge of T1.

5.0 Bus Interface (Continued)

TABLE 5-1. Pin Description (Continued)

Symbol	Driver Type	Direction	Description
SHARED-MEMORY ACCESS PINS			
$\overline{MREQ}$		I	**Memory Request:** The system asserts this signal low when it attempts to access the shared-buffer RAM. The on-chip arbiter resolves accesses between the system and the SONIC. **Note:** Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently. If these signals are successively asserted, there must be at least two bus clocks between the deasserting edge of the first signal and the asserting edge of the second signal.
$\overline{SMACK}$	TP	O	**Slave and Memory Acknowledge:** SONIC asserts this dual function pin low in response to either a Chip Select ($\overline{CS}$) or a Memory Request ($\overline{MREQ}$) when the SONIC's registers or it's buffer memory is available for accessing. This pin can be used for enabling bus drivers for dual-bus systems.
USER DEFINABLE PINS			
USR0,1	TRI	I, O	**User Define 0,1:** These signals are inputs when SONIC is hardware reset and are outputs when SONIC is a bus master (HLDA or $\overline{BGACK}$). When hard reset ($\overline{RST}$) is low, these signals input directly into bits 8 and 9 of the Data Configuration register (DCR) respectively. The levels on these pins are latched on the rising edge of $\overline{RST}$. During busmaster operations (HLDA or $\overline{BGACK}$ is active), these pins are outputs whose levels are programmable through bits 11 and 12 of the DCR respectively. The USR0,1 pins should be pulled up to V_{CC} or pulled down to ground. A 4.7 kΩ pull-up resistor is recommended.
POWER AND GROUND PINS			
VCC1–5			**Power:** The +5V power supply for the digital portions of the SONIC.
TXVCC RXVCC PLLVCC VCCL			**Power:** These pins are the +5V power supply for the SONIC ENDEC unit. These pins must be tied to V_{CC} even if the internal ENDEC is not used.
GND1–6			**Ground:** The ground reference for the digital portions of the SONIC.
TXGND ANGND GNDL			**Ground:** These pins are the ground references for the SONIC ENDEC unit. These pins must be tied to ground even if the internal ENDEC is not used.

5.3 SYSTEM CONFIGURATION

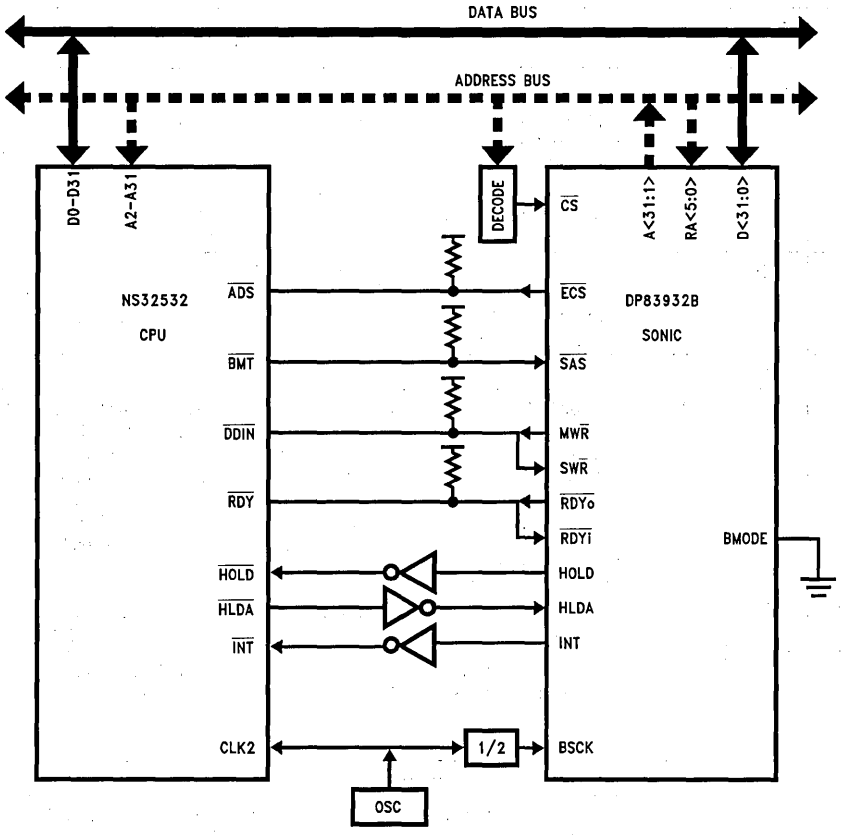
Any device that meets the SONIC interface protocol and electrical requirements (timing, threshold, and loading) can be interfaced to SONIC. Since two bus protocols are provided, via the BMODE pin, the SONIC can interface directly to most microprocessors. *Figure 5-3* shows a typical interface to the NSC/Intel style bus (BMODE=0) and *Figure 5-4* shows a typical interface to the Motorola style bus (BMODE=1).

The BMODE pin also controls byte ordering. When BMODE=1 big endian byte ordering is selected and when BMODE=0 little endian byte ordering is selected.

5.4 BUS OPERATIONS

There are two types of system bus operations: 1) SONIC as a slave, and 2) SONIC as a bus master. When SONIC is a slave (e.g., a CPU accessing SONIC registers) all transfers are non-DMA. When SONIC is a bus master (e.g., SONIC accessing receive or transmit buffer/descriptor areas) all transfers are block transfers using SONIC's on-chip DMA. This section describes the SONIC bus operations. Pay special attention to all sections labeled as **"Note"**. These conditions must be met for proper bus operation.

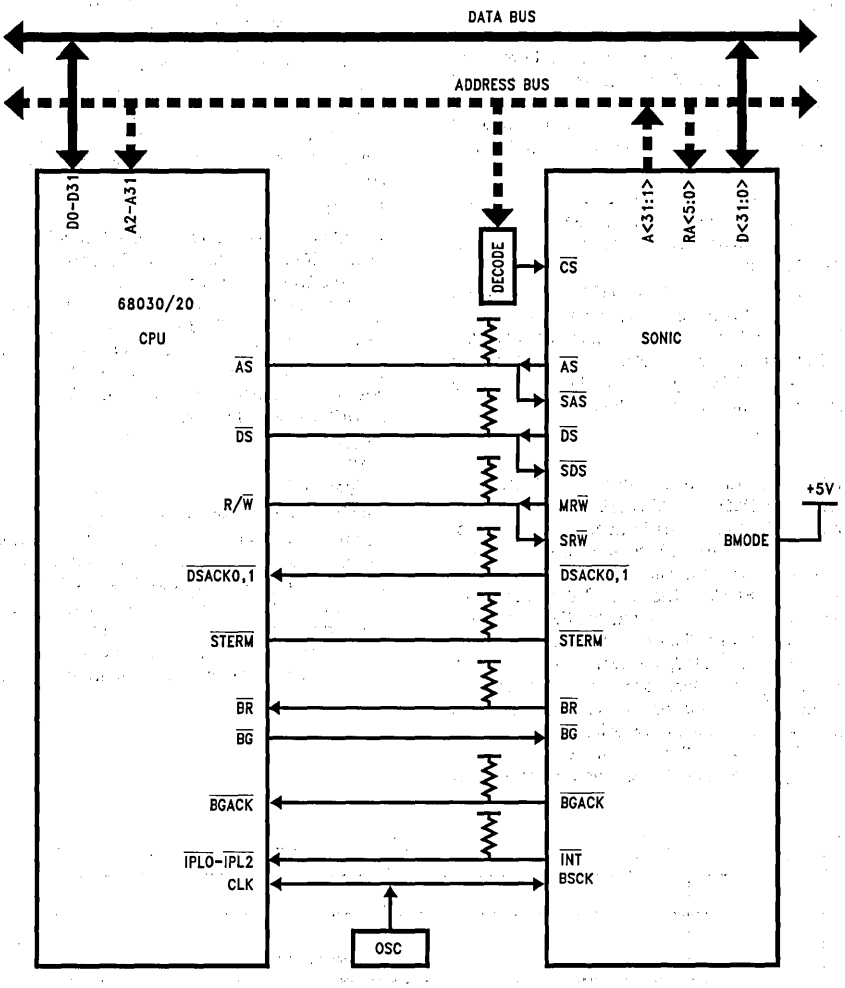
5.0 Bus Interface (Continued)



TL/F/10492–25

FIGURE 5-3. SONIC to NS32532 Interface Example

5.0 Bus Interface (Continued)



TL/F/10492–26

FIGURE 5-4. SONIC to Motorola 68030/20 Interface Example

5.0 Bus Interface (Continued)

5.4.1 Acquiring The Bus

The SONIC requests the bus when 1) its FIFO threshold has been reached or 2) when the descriptor areas in memory (i.e., RRA, RDA, CDA, and TDA) are accessed. Note that when the SONIC moves from one area in memory to another (e.g., RBA to RDA), it always deasserts its bus request and then requests the bus again when accessing the next area in memory.

The SONIC provides two methods to acquire the bus for compatibility with NSC/Intel or Motorola type microprocessors. These two methods are selected by setting the proper level on the BMODE pin.

Figures 5-5 and *5-6* show the NSC/Intel (BMODE = 0) and Motorola (BMODE = 1) bus request timing. Descriptions of each mode follows. For both modes, when the SONIC relinquishes the bus, there is an extra holding state (Th) for one bus cycle after the last DMA cycle (T2). This assures that the SONIC does not contend with another bus master after it has released the bus.

BMODE = 0

The NSC/Intel processors require a 2-way handshake using a HOLD REQUEST/HOLD ACKNOWLEDGE protocol (*Figure 5-5*). When the SONIC needs to access the bus, it issues a HOLD REQUEST (HOLD) to the microprocessor. The microprocessor, responds with a HOLD ACKNOWLEDGE (HLDA) to the SONIC. The SONIC then begins its memory transfers on the bus. As long as the CPU maintains HLDA active, the SONIC continues until it has finished its memory block transfer. The CPU, however, can preempt the SONIC from finishing the block transfer by deasserting HLDA before the SONIC deasserts HOLD. This allows a higher priority device to preempt the SONIC from continuing to use the bus. The SONIC will request the bus again later to complete any operation that it was doing at the time of preemption.

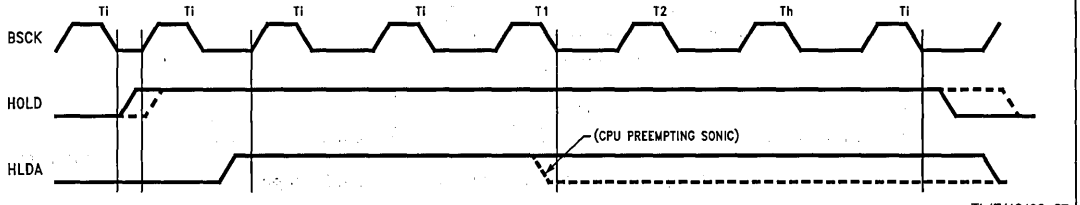
As shown in *Figure 5-5*, the SONIC will assert HOLD to either the falling or rising edge of the bus clock (BSCK). The default is for HOLD to be asserted on the falling edge. Setting the PH bit in the DCR2 (see section 4.3.7) causes HOLD to be asserted ½ bus clock later on the rising edge (shown by the dotted line). Before HOLD is asserted, the SONIC checks the HLDA line. If HLDA is asserted, HOLD will not be asserted until after HLDA has been deasserted first.

BMODE = 1

The Motorola protocol requires a 3-way handshake using a BUS REQUEST, BUS GRANT, and BUS GRANT ACKNOWLEDGE handshake (*Figure 5-6*). When using this protocol, the SONIC requests the bus by lowering BUS REQUEST ($\overline{BR}$). The CPU responds by issuing BUS GRANT ($\overline{BG}$). Upon receiving $\overline{BG}$, the SONIC assures that all devices have relinquished control of the bus before using the bus. The following signals must be deasserted before the SONIC acquires the bus:

$\overline{BGACK}$
$\overline{AS}$
$\overline{DSACK0,1}$
$\overline{STERM}$ (Asynchronous Mode Only)

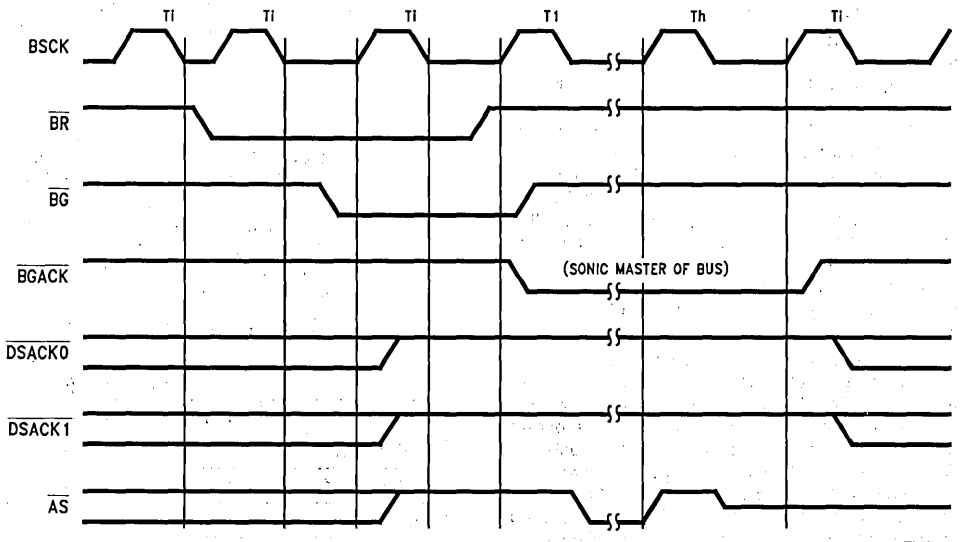
Deasserting $\overline{BGACK}$ indicates that the previous master has released the bus. Deasserting $\overline{AS}$ indicates that the previous master has completed its cycle and deasserting $\overline{DSACK0,1}$ and $\overline{STERM}$ indicates that the previous slave has terminated its connection to the previous master. The SONIC maintains its mastership of the bus until it deasserts $\overline{BGACK}$. It can not be preempted from the bus.



TL/F/10492–27

FIGURE 5-5. Bus Request Timing, BMODE = 0

5.0 Bus Interface (Continued)



TL/F/10492–28

FIGURE 5-6. Bus Request Timing, BMODE = 1

5.4.2 Block Transfers

The SONIC performs block operations during all bus actions, thereby providing efficient transfers to memory. The block cycle consists of three parts. The first part is the bus acquisition phase, as discussed above, in which the SONIC gains access to the bus. Once it has access of the bus, the SONIC enters the second phase by transferring data to/from its internal FIFOs or registers from/to memory. The SONIC transfers data from its FIFOs in either EXACT BLOCK mode or EMPTY/FILL.

EXACT BLOCK mode: In this mode the number of words (or long words) transferred during a block transfer is determined by either the Transmit or Receive FIFO thresholds programmed in the Data Configuration Register.

EMPTY/FILL mode: In this mode the DMA completely fills the Transmit FIFO during transmission, or completely empties the Receive FIFO during reception. This allows for greater bus latency.

When the SONIC accesses the Descriptor Areas (i.e., RRA, RDA, CDA, and TDA), it transfers data between its registers and memory. All fields which need to be used are accessed in one block operation. Thus, the SONIC performs 4 accesses in the RRA (see section 3.4.4.2), 7 accesses in the RDA (see section 3.4.6.1), 2, 3, or 6 accesses in the TDA (see section 3.5.4) and 4 accesses in the CDA.

5.4.3 Bus Status

The SONIC presents three bits of status information on pins S2–S0 which indicate the type of bus operation the SONIC is currently performing (Table 5-2). Bus status is valid when at the falling edge of $\overline{AS}$ or the rising edge of $\overline{ADS}$.

TABLE 5-2. Bus Status

S2	S1	S0	Status
1	1	1	The bus is idle. The SONIC is not performing any transfers on the bus.
1	0	1	The Transmit Descriptor Area (TDA) is currently being accessed.
0	0	1	The Transmit Buffer Area (TBA) is currently being read.
0	1	1	The Receive Buffer Area (RBA) is currently being written to. Only data is being written, though, not a Source or Destination address.
0	1	0	The Receive Buffer Area (RBA) is currently being written to. Only the Source or Destination address is being written, though.
1	1	0	The Receive Resource Area (RRA) is currently being read.
1	0	0	The Receive Descriptor Area (RDA) is currently being accessed.
0	0	0	The CAM Descriptor Area (CDA) is currently being accessed.

5.0 Bus Interface (Continued)

5.4.3.1 Bus Status Transitions

When the SONIC acquires the bus, it only transfers data to/from a single area in memory (i.e., TDA, TBA, RDA, RBA, RRA, or CDA). Thus, the bus status pins remain stable for the duration of the block transfer cycle with the following three exceptions: 1) If the SONIC is accessed during a block transfer, S2–S0 indicates bus idle during the register access, then returns to the previous status. 2) If the SONIC finishes writing the Source Address during a block transfer S2–S0 changes from [0,1,0] to [0,1,1]. 3) During an RDA access between the RXpkt.seq_no and RXpkt.link access, and between the RXpkt.link and RXpkt.in_use access, S2–S0 will respectively indicate idle [1,1,1] for 2 or 1 bus clocks. Status will be valid on the falling edge of $\overline{AS}$ or rising edge of $\overline{ADS}$.

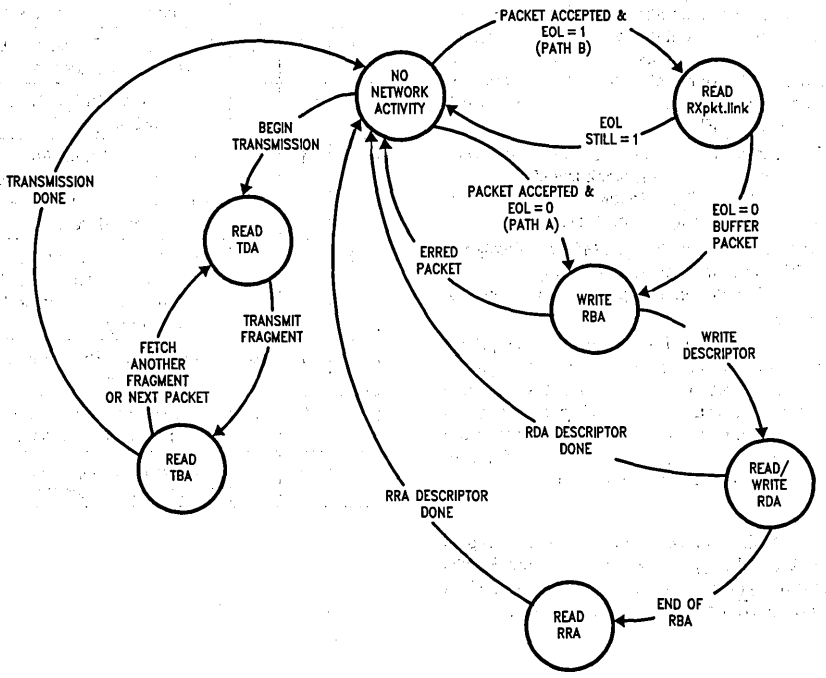
Figure 5-7 illustrates the SONIC's transitions through memory during the process of transmission and reception. During transmission, the SONIC reads the descriptor information from the TDA and then transmits data of the packet from the TBA. The SONIC moves back and forth between the TDA and TBA until all fragments and packets are transmitted. During reception, the SONIC takes one of two paths. In the first case (path A), when the SONIC detects EOL=0 from the previous reception, it buffers the accepted packet into the RBA, and then writes the descriptor information to the RDA. If the RBA becomes depleted (i.e., RBWC0,1 < EOBC), it moves to the RRA to read a resource descriptor. In the second case (path B), when the SONIC detects EOL=1 from the previous reception, it rereads the RXpkt.link field to determine if the system has reset the EOL bit since the last reception. If it has, the SONIC buffers the packet as in the first case. Otherwise, it rejects the packet and returns to idle.

5.4.4 Bus Mode Compatibility

For compatibility with different microprocessor and bus architectures, the SONIC operates in one of two modes (set by the BMODE pin) called the NSC/Intel or little endian mode (BMODE tied low) and the Motorola or big endian mode (BMODE tied high). The definitions for several pins change depending on the mode the SONIC is in. Table 5-3 shows these changes. These modes affect both master and slave bus operations with the SONIC.

TABLE 5-3. Bus Mode Compatibility

Pin Name	BMODE=0 (NSC/Intel)	BMODE=1 (Motorola)
$\overline{BR}$/HOLD	HOLD	$\overline{BR}$
$\overline{BG}$/HLDA	HLDA	$\overline{BG}$
MR$\overline{W}$/MW$\overline{R}$	MW$\overline{R}$	MR$\overline{W}$
SR$\overline{W}$/SW$\overline{R}$	SW$\overline{R}$	SR$\overline{W}$
$\overline{DSACK0}$/$\overline{RDYi}$	$\overline{RDYi}$	$\overline{DSACK0}$
$\overline{DSACK1}$/$\overline{RDYo}$	$\overline{RDYo}$	$\overline{DSACK1}$
$\overline{AS}$/$\overline{ADS}$	$\overline{ADS}$	$\overline{AS}$
$\overline{INT}$/INT	INT	$\overline{INT}$



TL/F/10492–29

FIGURE 5-7. Bus Status Transitions

5.0 Bus Interface (Continued)

5.4.5 Master Mode Bus Cycles

In order to add additional compatibility with different bus architectures, there are two other modes that affect the operation of the bus. These modes are called the synchronous and asynchronous modes and are programmed by setting or resetting the SBUS bit in the Data Configuration Register (DCR). The synchronous and asynchronous modes do not have an effect on slave accesses to the SONIC but they do affect the master mode operation. Within the particular bus/processor mode, synchronous and asynchronous modes are very similar. This section discusses all four modes of operation of the SONIC (NSC/Intel vs. Motorola, synchronous vs. asynchronous) when it is a bus master.

In this section, the rising edge of T1 and T2 means the beginning of these states, and the falling edge of T1 and T2 means the middle of these states.

5.4.5.1 Adding Wait States

To accommodate different memory speeds, the SONIC provides two methods for adding wait states for its bus operations. Both of these methods can be used singly or in conjunction with each other. A memory cycle is extended by adding additional T2 states. The first method inserts wait-states by withholding the assertion of $\overline{\text{DSACK0,1}}$/$\overline{\text{STERM}}$ or $\overline{\text{RDYi}}$. The other method allows software to program wait-states. Programming the WC0, WC1 bits in the Data Configuration Register allows 1 to 3 wait-states to be added on each memory cycle. These wait states are inserted between the T1 and T2 bus states and are called T2(wait) bus states. The SONIC will not look at the $\overline{\text{DSACK0,1}}$, $\overline{\text{STERM}}$ or $\overline{\text{RDYi}}$ lines until the programmed wait states have passed. Hence, in order to complete a bus operation that includes programmed wait states, the $\overline{\text{DSACK0,1}}$, $\overline{\text{STERM}}$ or $\overline{\text{RDYi}}$ lines must be asserted at their proper times at the end of the cycle during the last T2, not during a programmed wait state. The only exception to this is asynchronous mode where $\overline{\text{DSACK0,1}}$ or $\overline{\text{RDYi}}$ would be asserted during the last programmed wait state, T2 (wait). See the timing for these signals in the timing diagrams for more specific information. Programmed wait states do not affect Slave Mode bus cycles.

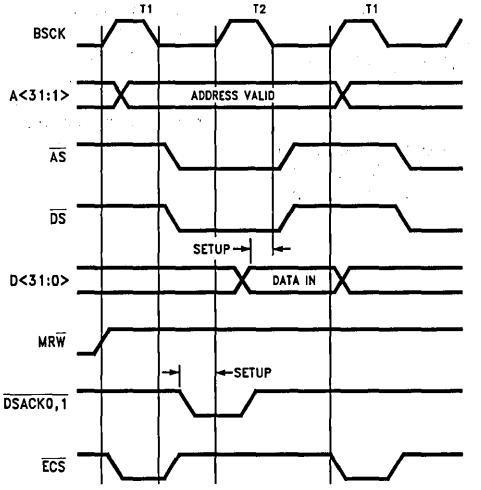
5.0 Bus Interface (Continued)

5.4.5.2 Memory Cycle for BMODE = 1, Synchronous Mode

On the rising edge of T1, the SONIC asserts $\overline{ECS}$ to indicate that the memory cycle is starting. The address (A31-A1), bus status (S2-S0) and the direction strobe (MR$\overline{W}$) are driven and do not change for the remainder of the memory cycle. On the falling edge of T1, the SONIC deasserts $\overline{ECS}$ and asserts $\overline{AS}$.

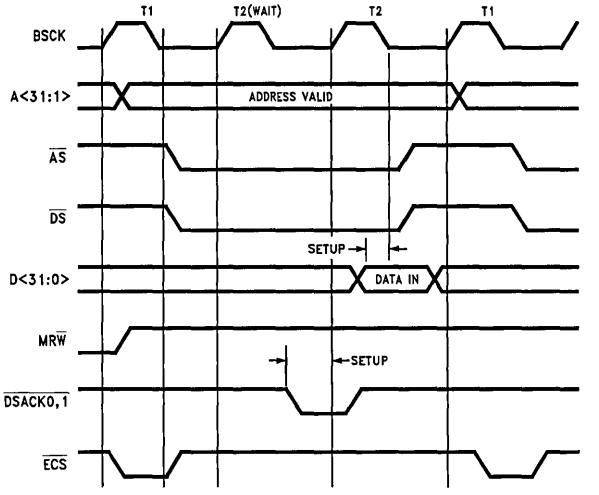
In synchronous mode, $\overline{DSACK0,1}$ are sampled on the rising edge of T2. T2 states will be repeated until $\overline{DSACK0,1}$ are sampled properly in a low state. $\overline{DSACK0,1}$ must meet the setup and hold times with respect to the rising edge of bus clock for proper operation.

During read cycles (*Figures 5-8* and *5-9*) data (D31-D0) is latched at the falling edge of T2 and $\overline{DS}$ is asserted at the falling edge of T1. For write cycles (*Figures 5-10* and *5-11*) data is driven on the falling edge of T1. If there are wait states inserted, $\overline{DS}$ is asserted on the falling edge of T2. $\overline{DS}$ is not asserted for zero wait state write cycles. The SONIC terminates the memory cycle by deasserting $\overline{AS}$ and $\overline{DS}$ at the falling edge of T2.



TL/F/10492–30

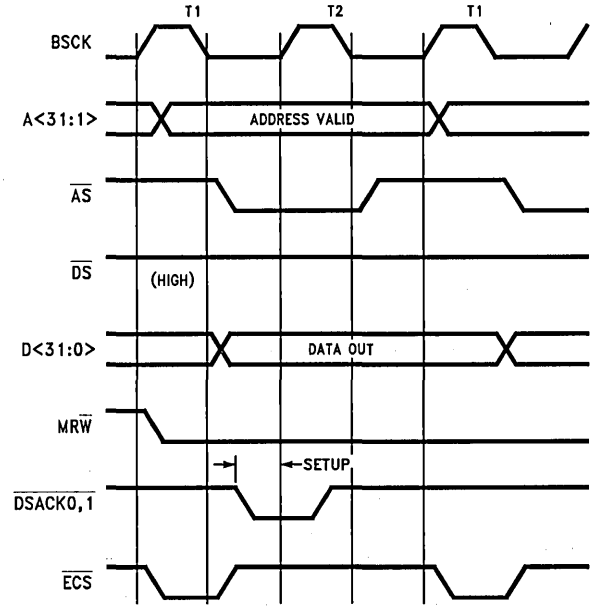
FIGURE 5-8. Memory Read, BMODE = 1, Synchronous (0 Wait-State)



TL/F/10492–31

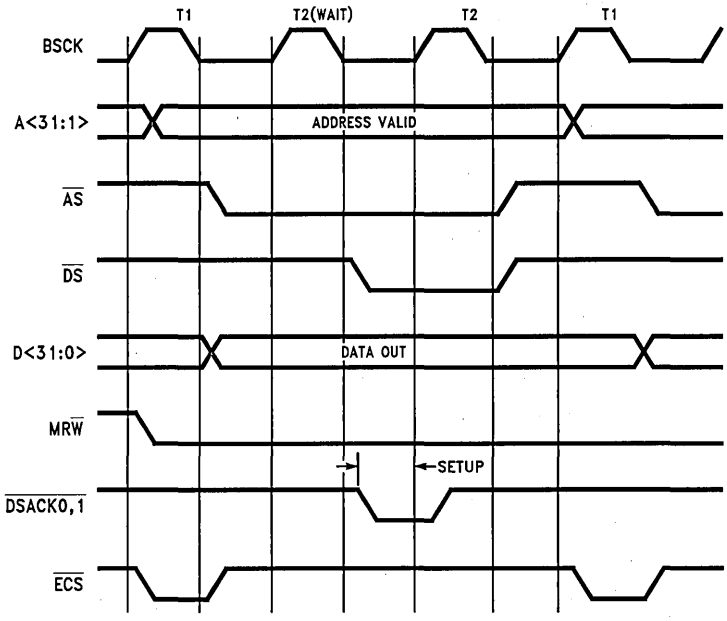
FIGURE 5-9. Memory Read, BMODE = 1, Synchronous (1 Wait-State)

5.0 Bus Interface (Continued)



TL/F/10492–32

FIGURE 5-10. Memory Write, BMODE = 1, Synchronous (0 Wait-State)



TL/F/10492–33

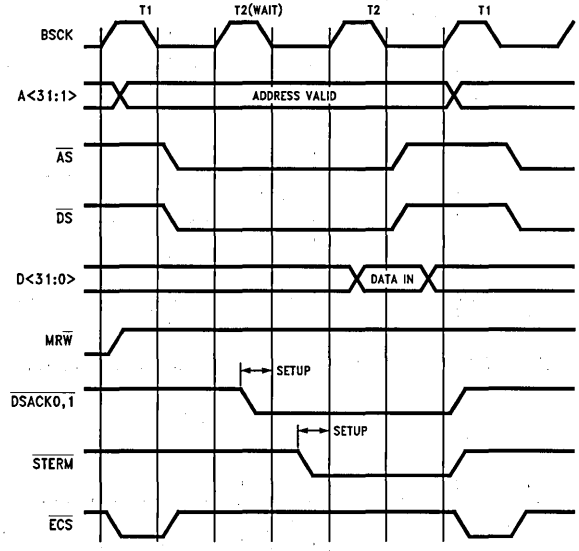
FIGURE 5-11. Memory Write, BMODE = 1, Synchronous (1 Wait-State)

5.0 Bus Interface (Continued)

5.4.5.3 Memory Cycle for BMODE = 1, Asynchronous Mode

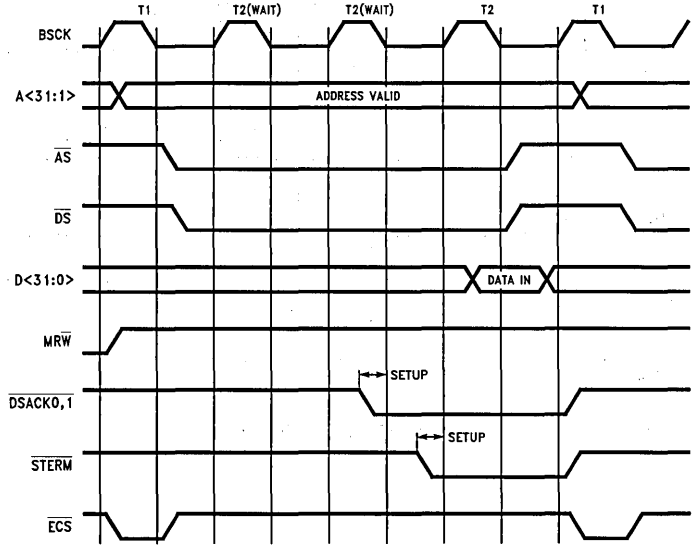
On the rising edge of T1, the SONIC asserts $\overline{ECS}$ to indicate that the memory cycle is starting. The address (A31-A1), bus status (S2-S0) and the direction strobe (MR$\overline{W}$) are driven and do not change for the remainder of the memory cycle. On the falling edge of T1, the SONIC deasserts $\overline{ECS}$ and asserts $\overline{AS}$.

In asynchronous mode, $\overline{DSACK0,1}$ are asynchronously sampled on the falling edge of both T1 and T2. $\overline{DSACK0,1}$ do not need to be synchronized to the bus clock because the chip always resolves these signals to either a high or low state. If a synchronous termination of the bus cycle is required, however, $\overline{STERM}$ may be used. $\overline{STERM}$ is sampled on the rising edge of T2 and must meet the setup and hold times with respect to that edge for proper operation. Meeting the setup time for $\overline{DSACK0,1}$ or $\overline{STERM}$ guarantees that the SONIC will terminate the memory cycle 1½



TL/F/10492–36

FIGURE 5-12. Memory Read, BMODE = 1, Asynchronous (1 Wait-State)



TL/F/10492–37

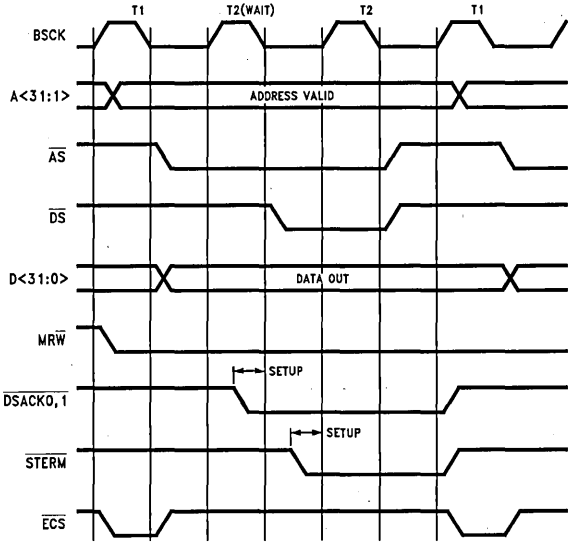
FIGURE 5-13. Memory Read, BMODE = 1, Asynchronous (2 Wait-State)

5.0 Bus Interface (Continued)

bus clocks after $\overline{DSACK0,1}$ were sampled, or 1 cycle after $\overline{STERM}$ was sampled. T2 states will be repeated until $\overline{DSACK0,1}$ or $\overline{STERM}$ are sampled properly in a low state. (see note below).

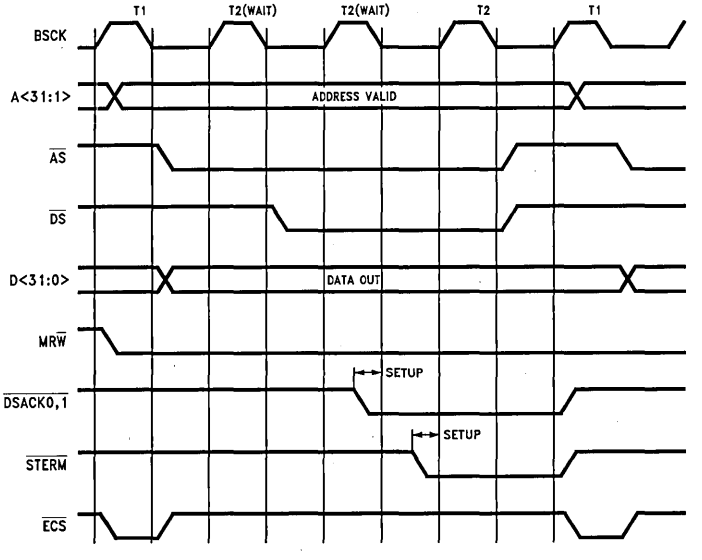
During read cycles (*Figures 5-12* and *5-13*), data (D31-D0) is latched at the falling edge of T2 and $\overline{DS}$ is asserted at the falling edge of T1 . For write cycles (*Figures 5-14* and *5-15*) data is driven on the falling edge of T1. If there are wait states inserted, $\overline{DS}$ is asserted on the falling edge of the first T2(wait). $\overline{DS}$ is not asserted for zero wait state write cycles. The SONIC terminates the memory cycle by deasserting $\overline{AS}$ and $\overline{DS}$ at the falling edge of T2.

Note: If the setup time for $\overline{DSACK0,1}$ is met during T1, or the setup time for $\overline{STERM}$ is met during the first T2, the full asynchronous bus cycle will take only 2 bus clocks. This may be an unwanted situation. If so, $\overline{DSACK0,1}$ and $\overline{STERM}$ should be deasserted during T1 and the start of T2 respectively.



TL/F/10492–34

FIGURE 5-14. Memory Write, BMODE = 1, Asynchronous (1 Wait-State)



TL/F/10492–35

FIGURE 5-15. Memory Write, BMODE = 1, Asynchronous (2 Wait-State)

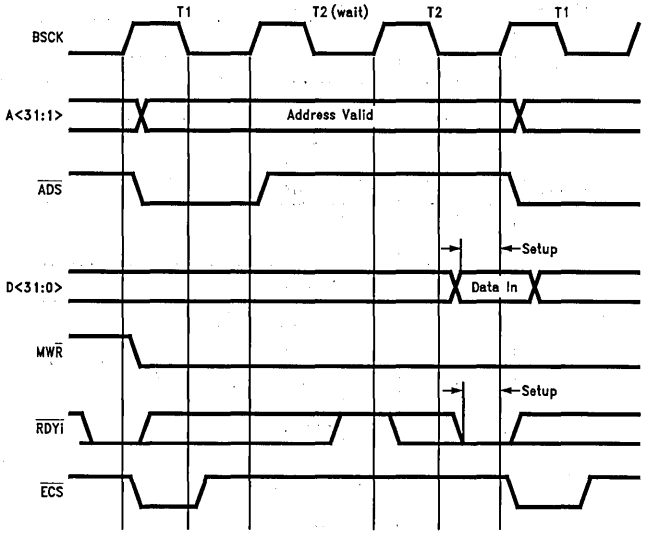
5.0 Bus Interface (Continued)

5.4.5.4 Memory Cycle for BMODE = 0, Synchronous Mode

On the rising edge of T1, the SONIC asserts $\overline{ADS}$ and $\overline{ECS}$ to indicate that the memory cycle is starting. The address (A31-A1), bus status (S2-S0) and the direction strobe ($MW\overline{R}$) are driven and do not change for the remainder of the memory cycle. On the falling edge of T1, the SONIC deasserts $\overline{ECS}$. $\overline{ADS}$ is deasserted on the rising edge of T2.

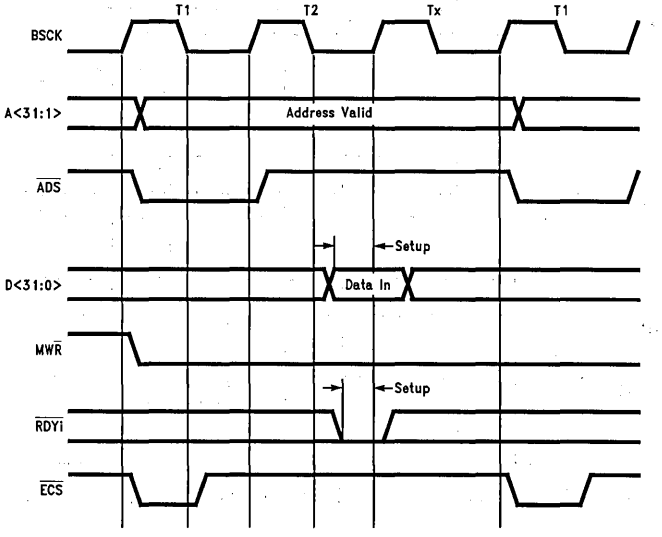
In Synchronous mode, $\overline{RDYi}$ is sampled on the rising edge at the end of T2 (the rising edge of the next T1 or Tx). T2 states will be repeated until $\overline{RDYi}$ is sampled properly in a low state. $\overline{RDYi}$ must meet the setup and hold times with respect to the rising edge of bus clock for proper operation. During read cycles (*Figures 5-16* and *5-17*), data (D31-D0) is latched at the rising edge at the end of T2. For write cycles (*Figures 5-18* and *5-19*) data is driven on the falling edge of T1 and stays driven until the end of the cycle.

If Latched Ready mode is used (see section 4.3.7) there will be a Tx state between the last T2 of the bus cycle and the T1 of the next bus cycle. $\overline{RDYi}$ and the data are still latched in on the rising edge at the end of T2 (rising edge of Tx). All the other signals are carried out to the end of Tx (*Figures 5-17* and *5-19*).



TL/F/10492–38

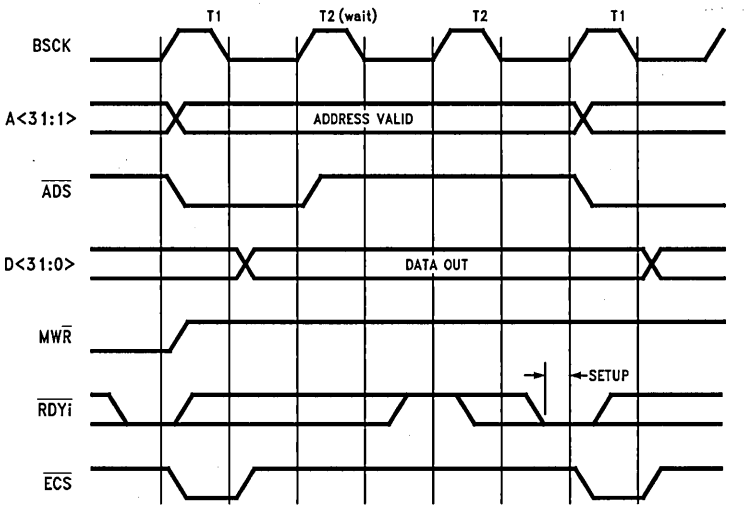
FIGURE 5-16. Memory Read, BMODE = 0, Synchronous (1 Wait-State)



TL/F/10492–39

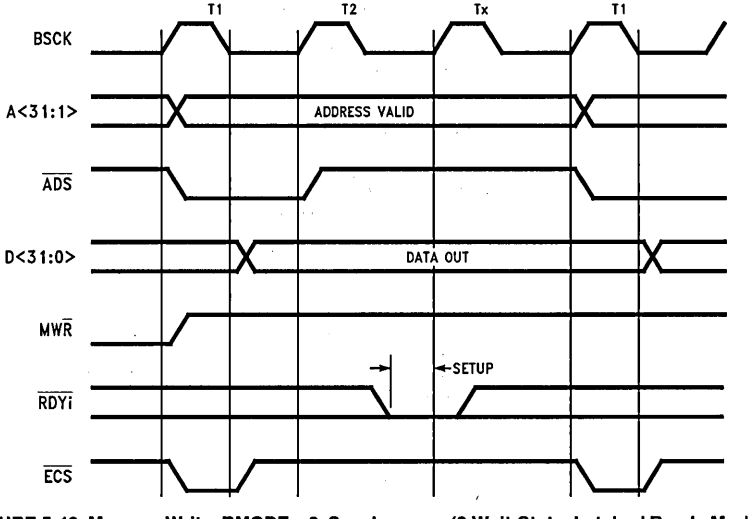
FIGURE 5-17. Memory Read, BMODE = 0, Synchronous (0 Wait-State, Latched Ready Mode)

5.0 Bus Interface (Continued)



TL/F/10492–40

FIGURE 5-18. Memory Write, BMODE = 0, Synchronous (1 Wait-State)



TL/F/10492–41

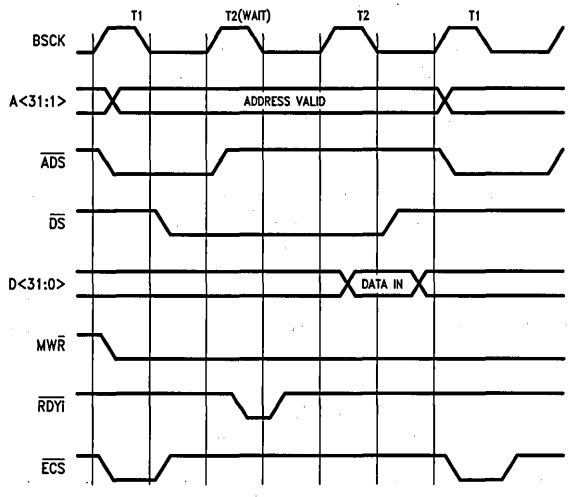
FIGURE 5-19. Memory Write, BMODE = 0, Synchronous (0 Wait-State, Latched Ready Mode)

5.0 Bus Interface (Continued)

5.4.5.5 Memory Cycle for BMODE = 0, Asynchronous Mode

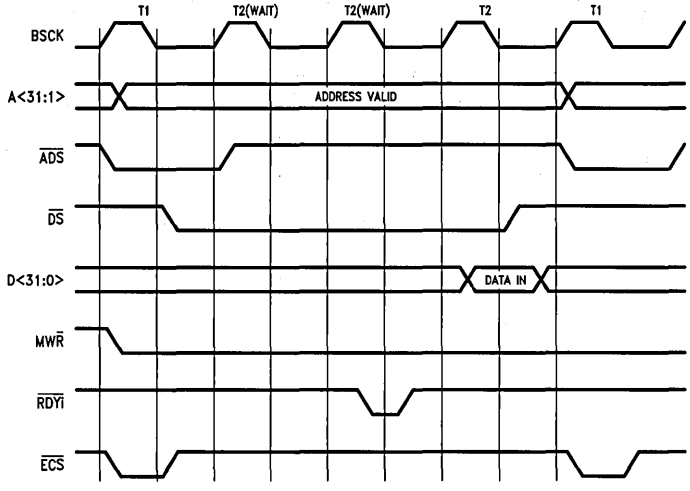
On the rising edge of T1, the SONIC asserts $\overline{ADS}$ and $\overline{ECS}$ to indicate that the memory cycle is starting. The address (A31-A1), bus status (S2-S0) and the direction strobe ($MW\overline{R}$) are driven and do not change for the remainder of the memory cycle. On the falling edge of T1, the SONIC deasserts $\overline{ECS}$. $\overline{ADS}$ is deasserted on the rising edge of T2.

In Asynchronous mode, $\overline{RDYi}$ is asynchronously sampled on the falling edge of both T1 and T2. $\overline{RDYi}$ does not need to be synchronized to the bus clock because the chip always resolves these signals to either a high or low state. Meeting the setup time for $\overline{RDYi}$ guarantees that the SONIC will terminate the memory cycle 1½ bus clocks after $\overline{RDYi}$ was sampled. T2 states will be repeated until $\overline{RDYi}$ is sampled properly in a low state (see note below).



TL/F/10492-44

FIGURE 5-20. Memory Read, BMODE = 0, Asynchronous (1 Wait-State)



TL/F/10492-45

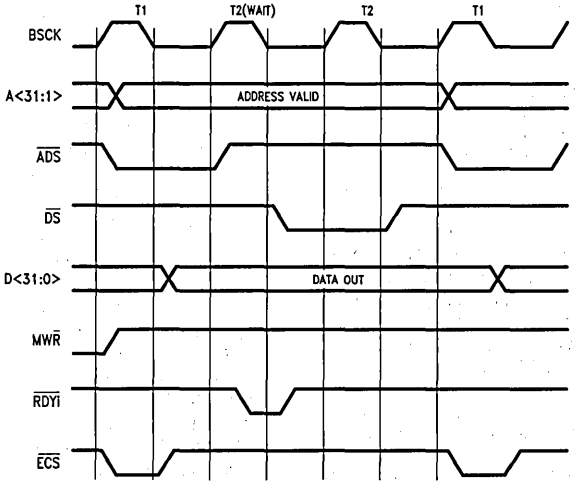
FIGURE 5-21. Memory Read, BMODE = 0, Asynchronous (2 Wait-State)

5.0 Bus Interface (Continued)

During read cycles (*Figures 5-20* and *5-21*), data (D31-D0) is latched at the falling edge of T2 and $\overline{DS}$ is asserted at the falling edge of T1. For write cycles (*Figures 5-22* and *5-23*) data is driven on the falling edge of T1. If there are wait states inserted, $\overline{DS}$ is asserted on the falling edge of the first T2(wait). $\overline{DS}$ is not asserted for zero wait state write cycles.

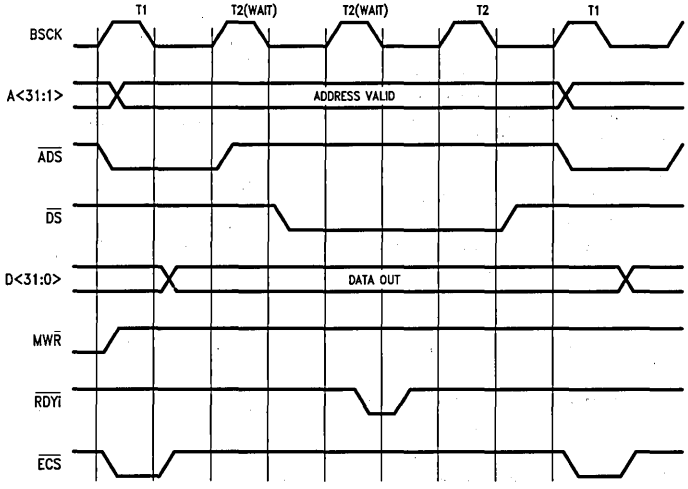
The SONIC terminates the memory cycle by deasserting $\overline{DS}$ at the falling edge of T2.

Note: If the setup time for $\overline{RDYi}$ is met during T1, the full asynchronous bus cycle will take only 2 bus clocks. This may be an unwanted situation. If so, $\overline{RDYi}$ should be deasserted during T1.



TL/F/10492–42

FIGURE 5-22. Memory Write, BMODE = 0, Asynchronous (1 Wait-State)



TL/F/10492–43

FIGURE 5-23. Memory Write, BMODE = 0, Asynchronous (2 Wait-State)

5.0 Bus Interface (Continued)

5.4.6 Bus Exceptions (Bus Retry)

The SONIC provides the capability of handling errors during the execution of the bus cycle (*Figure 5-24*).

The system asserts $\overline{BRT}$ (bus retry) to force the SONIC to repeat the current memory cycle. When the SONIC detects the assertion of $\overline{BRT}$, it completes the memory cycle at the end of T2 and gets off the bus by deasserting $\overline{BGACK}$ or HOLD. Then, if Latched Bus Retry mode is not set (LBR in the Data Configuration Register, section 4.3.2), the SONIC requests the bus again to retry the same memory cycle. If Latched Bus Retry is set, though, the SONIC will not retry until the BR bit in the ISR (see section 4.3.6) has been reset and $\overline{BRT}$ is deasserted. $\overline{BRT}$ has precedence of terminating a memory cycle over $\overline{DSACK0,1}$, $\overline{STERM}$ or $\overline{RDYi}$.

$\overline{BRT}$ may be sampled synchronously or asynchronously by setting the EXBUS bit in the DCR (see section 4.3.2). If synchronous Bus Retry is set, $\overline{BRT}$ is sampled on the rising edge of T2. If asynchronous Bus Retry is set, $\overline{BRT}$ is double synchronized from the falling edge of T1. The asynchronous setup time does not need to be met, but doing so will guarantee that the bus exception will occur in the current bus cycle instead of the next bus cycle. Asynchronous Bus Retry may only be used when the SONIC is set to asynchronous mode.

Note 1: The deassertion edge of HOLD is dependent on the PH bit in the DCR2 (see section 4.3.7). Also, $\overline{BGACK}$ is driven high for about ½ bus clock before going TRI-STATE.

Note 2: If Latched Bus retry is set, $\overline{BRT}$ need only satisfy its setup time (the hold time is not important). Otherwise, $\overline{BRT}$ must remain asserted until after the Th state.

Note 3: If $\overline{DSACK0,1}$, $\overline{STERM}$ or $\overline{RDYi}$ remain asserted after $\overline{BRT}$, the next memory cycle, may be adversely affected.

5.4.7 Slave Mode Bus Cycle

The SONIC's internal registers can be accessed by one of two methods (BMODE = 1 or BMODE = 0). In both methods, the SONIC is a slave on the bus. This section describes the SONIC's slave mode bus operations.

5.4.7.1 Slave Cycle for BMODE = 1

The system accesses the SONIC by driving $\overline{SAS}$, SR$\overline{W}$ and RA<5:0>. These signals will be sampled each bus cycle, but the SONIC will not actually start a slave cycle until $\overline{CS}$ has also been asserted. $\overline{CS}$ should not be asserted before $\overline{SAS}$ is driven low as this will cause improper slave operation. Once $\overline{SAS}$ has been driven low, between one and two bus clocks after the assertion of $\overline{CS}$, $\overline{SMACK}$ will be asserted to signify that the SONIC has started the slave cycle. Although $\overline{CS}$ is an asynchronous input, meeting its setup time (as shown in *Figures 5-25* and *5-26*) will guarantee that $\overline{SMACK}$, which is asserted off of a falling edge, will be asserted 1 bus clock after the falling edge that $\overline{CS}$ is clocked in on. This is assuming that the SONIC is not a bus master when $\overline{CS}$ was asserted. If the SONIC is a bus master, then, when $\overline{CS}$ is asserted, the SONIC will complete its current master bus cycle and get off the bus temporarily (see section 5.4.8). In this case, $\overline{SMACK}$ will be asserted 5 bus clocks after the falling edge that $\overline{CS}$ was clocked in on. This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.

If the slave access is a read cycle (*Figure 5-25*), then the data will be driven off the same edge as $\overline{SMACK}$. If it is a write cycle (*Figure 5-26*), then the data will be latched in exactly 2 bus clocks after the assertion of $\overline{SMACK}$. In either case, $\overline{DSACK0,1}$ are driven low 2 bus clocks after $\overline{SMACK}$ to terminate the slave cycle. For a read cycle, the assertion of $\overline{DSACK0,1}$ indicates valid register data and for a write cycle, the assertion indicates that the SONIC has latched the data. The SONIC deasserts $\overline{DSACK0,1}$, $\overline{SMACK}$ and the data if the cycle is a read cycle at the rising edge of $\overline{SAS}$ or $\overline{CS}$ depending on which is deasserted first.

Note 1: Although the SONIC responds as a 32-bit peripheral when it drives $\overline{DSACK0,1}$ low, it transfers data only on lines D<15:0>.

Note 2: For multiple register accesses, $\overline{CS}$ can be held low and $\overline{SAS}$ can be used to delimit the slave cycle (this is the only case where $\overline{CS}$ may be asserted before $\overline{SAS}$). In this case, $\overline{SMACK}$ will be driven low due to $\overline{SAS}$ going low since $\overline{CS}$ has already been asserted. Notice that this means $\overline{SMACK}$ will not stay asserted low during the entire time $\overline{CS}$ is low (as is the case for $\overline{MREQ}$, section 5.4.8).

Note 3: If memory request ($\overline{MREQ}$) follows a chip select ($\overline{CS}$), it must be asserted at least 2 bus clocks after $\overline{CS}$ is deasserted. Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently.

Note 4: When $\overline{CS}$ is deasserted, it must remain deasserted for at least one bus clock.

Note 5: The way in which $\overline{SMACK}$ is asserted due to $\overline{CS}$ is not the same as the way in which $\overline{SMACK}$ is asserted due to $\overline{MREQ}$. The assertion of $\overline{SMACK}$ is dependent upon both $\overline{CS}$ and $\overline{SAS}$ being low, not just $\overline{CS}$. This is not the same as the case for $\overline{MREQ}$ (see section 5.4.8). The assertion of $\overline{SMACK}$ in these two cases should not be confused.

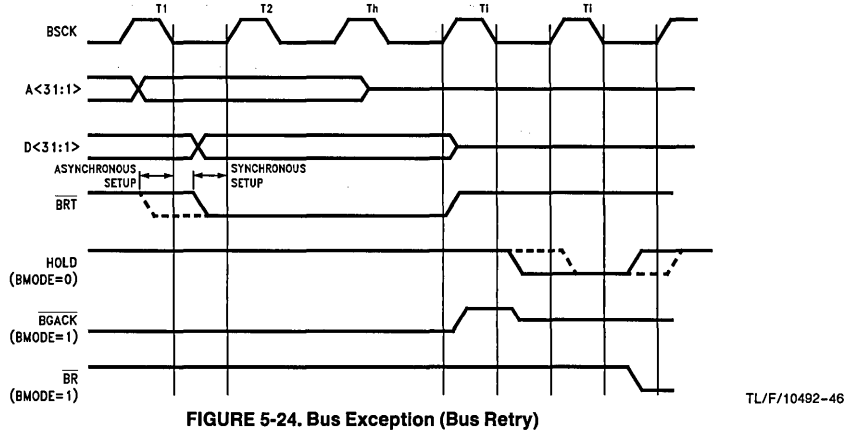
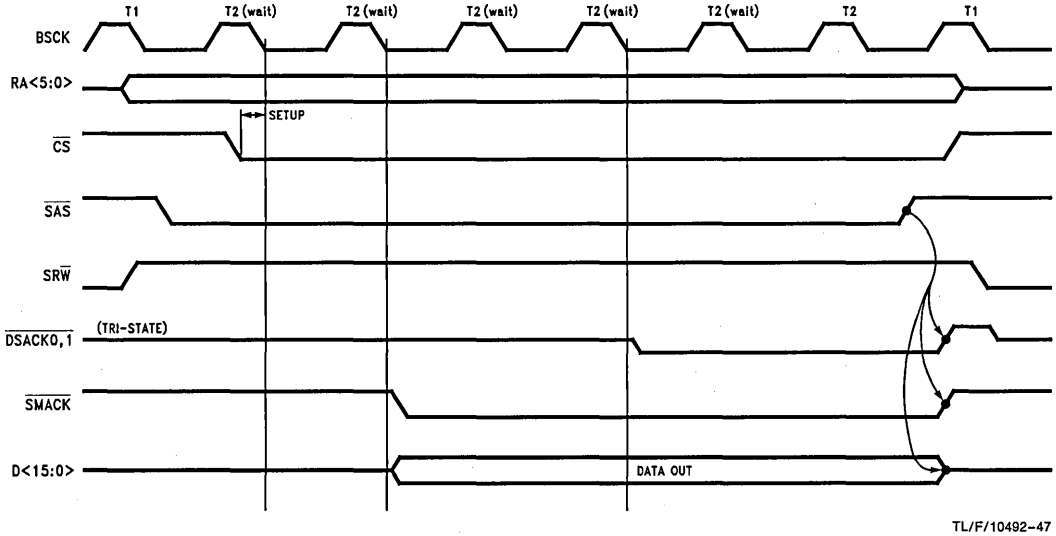


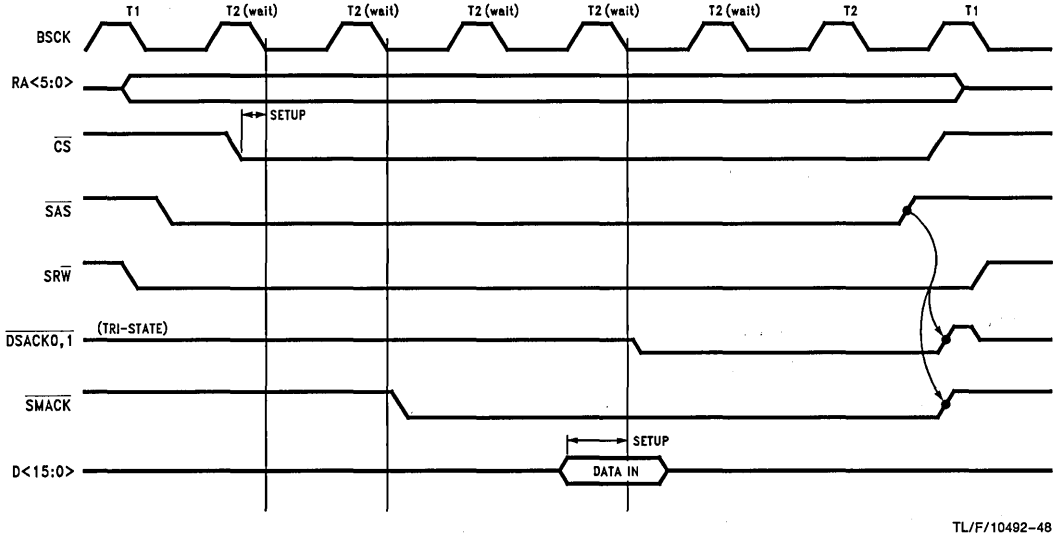
FIGURE 5-24. Bus Exception (Bus Retry)

5.0 Bus Interface (Continued)



TL/F/10492–47

FIGURE 5-25. Register Read, BMODE = 1



TL/F/10492–48

FIGURE 5-26. Register Write, BMODE = 1

5.0 Bus Interface (Continued)

5.4.7.2 Slave Cycle for BMODE = 0

The system accesses the SONIC by driving $\overline{SAS}$, $\overline{CS}$, $\overline{SWR}$ and RA<5:0>. These signals will be sampled each bus cycle, but the SONIC will not actually start a slave cycle until $\overline{CS}$ has been sampled low and $\overline{SAS}$ has been sampled high. $\overline{CS}$ should not be asserted low before the falling edge of $\overline{SAS}$ as this will cause improper slave operation. $\overline{CS}$ may be asserted low, however, before the rising edge of $\overline{SAS}$. In this case, it is suggested that $\overline{SAS}$ be driven high within one bus clock after the falling edge of $\overline{CS}$. Once $\overline{SAS}$ has been driven high, between one and two bus clocks after the assertion of $\overline{CS}$, $\overline{SMACK}$ will be driven low to signify that the SONIC has started the slave cycle. Although $\overline{CS}$ is an asynchronous input, meeting its setup time (as shown in *Figures 5-27* and *5-28*) will guarantee that $\overline{SMACK}$, which is asserted off a falling edge, will be asserted 1 bus clock after the falling edge that $\overline{CS}$ was clocked in on. This is assuming that the SONIC is not a bus master when $\overline{CS}$ is asserted. If the SONIC is a bus master, then, when $\overline{CS}$ is asserted, the SONIC will complete its current master bus cycle and get off the bus temporarily (see section 5.4.8). In this case, $\overline{SMACK}$ will be asserted 5 bus clocks after the falling edge that $\overline{CS}$ was clocked in on. This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.

If the slave access is a read cycle (*Figure 5-27*), then the data will be driven off the same edge as $\overline{SMACK}$. If it is a write cycle (*Figure 5-28*), then the data will be latched in exactly 2 bus clocks after the assertion of $\overline{SMACK}$. In either case, $\overline{RDYo}$ is driven low 2½ bus clocks after $\overline{SMACK}$ to terminate the slave cycle. For a read cycle, the assertion of $\overline{RDYo}$ indicates valid register data and for a write cycle, the assertion indicates that the SONIC has latched the data. The SONIC deasserts $\overline{RDYo}$, $\overline{SMACK}$ and the data if the cycle is a read cycle at the falling edge of $\overline{SAS}$ or the rising edge of $\overline{CS}$ depending on which is first.

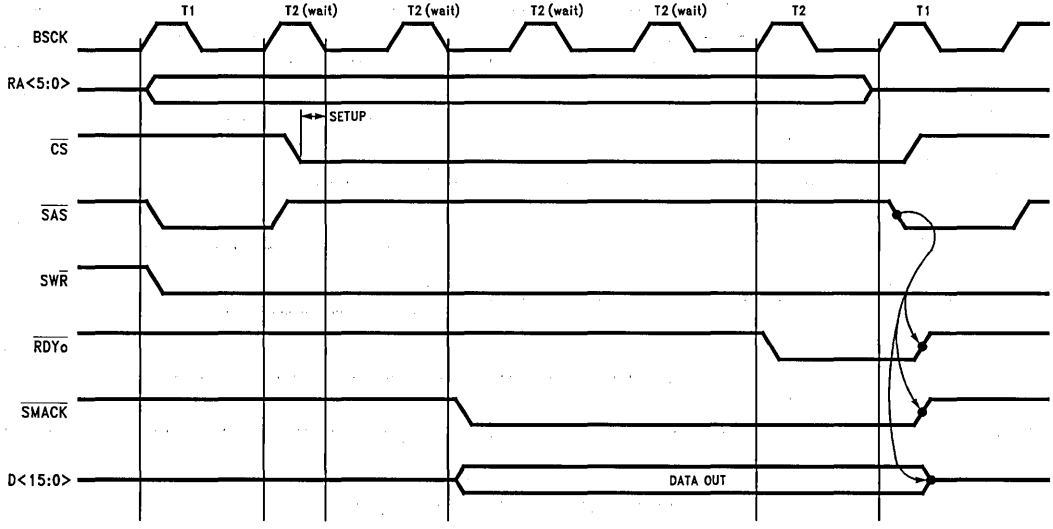
Note 1: The SONIC transfers data only on lines D<15:0> during slave mode accesses.

Note 2: For multiple register accesses, $\overline{CS}$ can be held low and $\overline{SAS}$ can be used to delimit the slave cycle (this is the only case where $\overline{CS}$ may be asserted before $\overline{SAS}$). In this case, $\overline{SMACK}$ will be driven low due to $\overline{SAS}$ going high since $\overline{CS}$ has already been asserted. Notice that this means $\overline{SMACK}$ will not stay asserted low during the entire time $\overline{CS}$ is low (as is the case for $\overline{MREQ}$, section 5.4.8).

Note 3: If memory request ($\overline{MREQ}$) follows a chip select ($\overline{CS}$), it must be asserted at least 2 bus clocks after $\overline{CS}$ is deasserted. Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently.

Note 4: When $\overline{CS}$ is deasserted, it must remain deasserted for at least one bus clock.

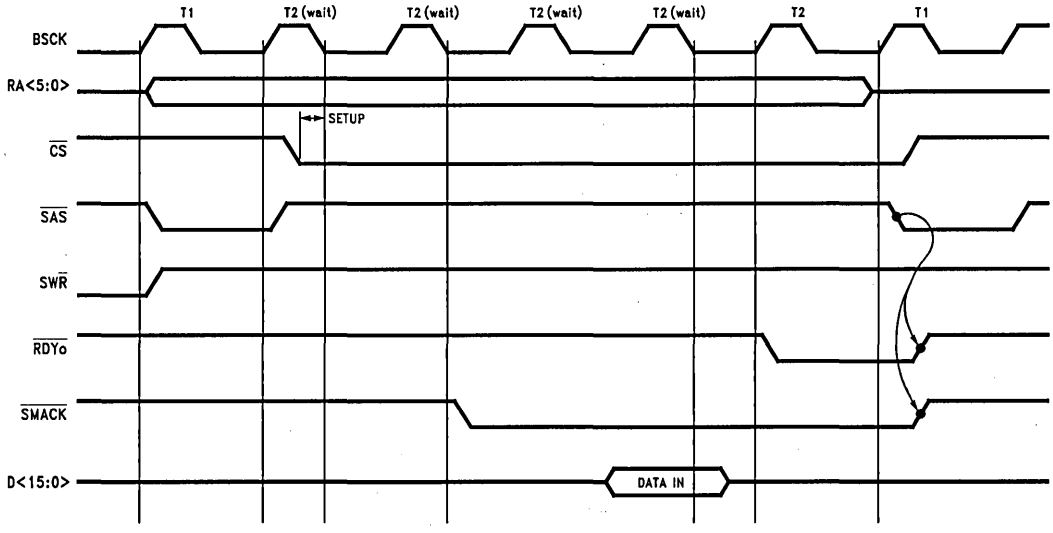
Note 5: The way in which $\overline{SMACK}$ is asserted due to $\overline{CS}$ is not the same as the way in which $\overline{SMACK}$ is asserted due to $\overline{MREQ}$. The assertion of $\overline{SMACK}$ is dependent upon both $\overline{CS}$ and $\overline{SAS}$ being low, not just $\overline{CS}$. This is not the same as the case for $\overline{MREQ}$ (see section 5.4.8). The assertion of $\overline{SMACK}$ in these two cases should not be confused.



TL/F/10492–49

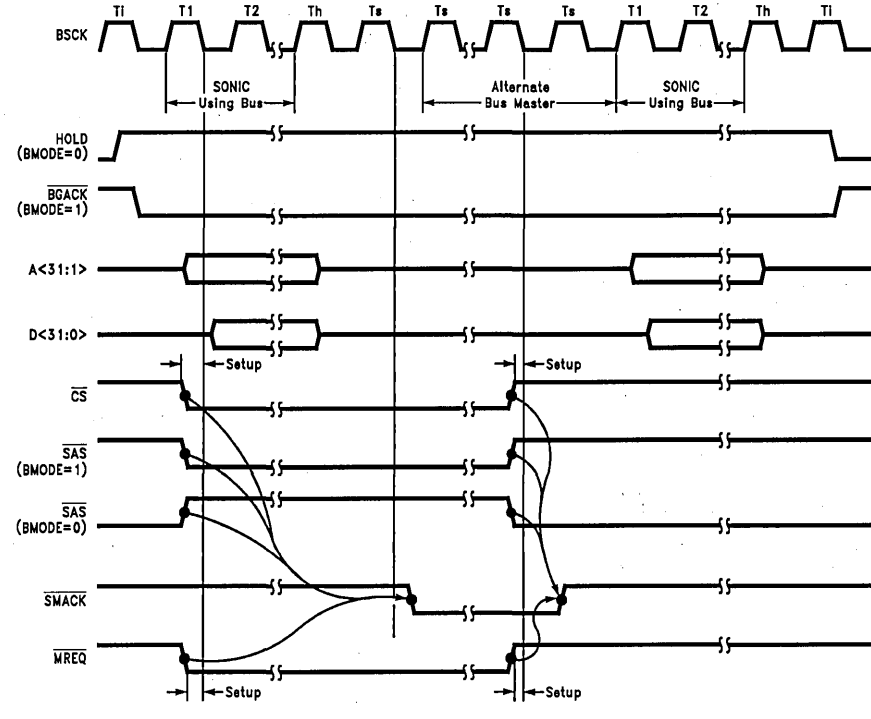
FIGURE 5-27. Register Read, BMODE = 0

5.0 Bus Interface (Continued)



TL/F/10492–50

FIGURE 5-28. Register Write, BMODE = 0



TL/F/10492–51

FIGURE 5-29. On-Chip Memory Arbiter

5.0 Bus Interface (Continued)

5.4.8 On-Chip Memory Arbiter

For applications which share the buffer memory area with the host system (shared-memory applications), the SONIC provides a fast on-chip memory arbiter for efficiently resolving accesses between the SONIC and the host system (*Figure 5-29*). The host system indicates its intentions to use the shared-memory by asserting Memory Request ($\overline{MREQ}$). The SONIC will allow the host system to use the shared memory by acknowledging the host system's request with Slave and Memory Acknowledge ($\overline{SMACK}$). Once $\overline{SMACK}$ is asserted, the host system may use the shared memory freely. The host system gives up the shared memory by deasserting $\overline{MREQ}$.

$\overline{MREQ}$ is clocked in on the falling edge of bus clock and is double synchronized internally to the rising edge. $\overline{SMACK}$ is asserted on the falling edge of a Ts bus cycle. If the SONIC is not currently accessing the memory, $\overline{SMACK}$ is asserted immediately after $\overline{MREQ}$ was clocked in. If, however, the SONIC is accessing the shared memory, it finishes its current memory transfer and then issues $\overline{SMACK}$. $\overline{SMACK}$ will be asserted 1 or 5 (see Note 2 below) bus clocks, respectively, after $\overline{MREQ}$ is clocked in. Since $\overline{MREQ}$ is double synchronized, it is not necessary to meet its setup time. Meeting the setup time for $\overline{MREQ}$ will, however, guarantee that $\overline{SMACK}$ is asserted in the next or fifth bus clock after the current bus clock. $\overline{SMACK}$ will deassert within one bus clock after $\overline{MREQ}$ is deasserted. The SONIC will then finish its master operation if it was using the bus previously.

If the host system needs to access the SONIC's registers instead of shared memory, $\overline{CS}$ would be asserted instead of $\overline{MREQ}$. Accessing the SONIC's registers works almost exactly the same as accessing the shared memory except that the SONIC goes into a slave cycle instead of going idle. See section 5.4.7 for more information about how register accesses work.

Note 1: The successive assertion of $\overline{CS}$ and $\overline{MREQ}$ must be separated by at least two bus clocks. Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently.

Note 2: The number of bus clocks between $\overline{MREQ}$ being asserted and the assertion of $\overline{SMACK}$ when the SONIC is in Master Mode is 5 bus clocks assuming there were no wait states in the Master Mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle (the time will be 5 + the number of wait states).

Note 3: The way in which $\overline{SMACK}$ is asserted to due to $\overline{CS}$ is not the same as the way in which $\overline{SMACK}$ is asserted due to $\overline{MREQ}$. $\overline{SMACK}$ goes low as a direct result of the assertion of $\overline{MREQ}$, whereas, for $\overline{CS}$, $\overline{SAS}$ must also be driven low (BMODE = 1) or high (BMODE = 0) before $\overline{SMACK}$ will be asserted. This means that when $\overline{SMACK}$ is asserted due to $\overline{MREQ}$, $\overline{SMACK}$ will remain asserted until $\overline{MREQ}$ is deasserted. Multiple memory accesses can be made to the shared memory without $\overline{SMACK}$ ever going high. When $\overline{SMACK}$ is asserted due to $\overline{CS}$, however, $\overline{SMACK}$ will only remain low as long as $\overline{SAS}$ is also low (BMODE = 1) or high (BMODE = 0). $\overline{SMACK}$ will not remain low throughout multiple register accesses to the SONIC because $\overline{SAS}$ must toggle for each register access. This is an important difference to consider when designing shared memory designs.

TABLE 5-4. Internal Register Content after Reset

Register	Contents after Reset	
	Hardware Reset	Software Reset
Command	0094h	0094h/00A4h
Data Configuration (DCR and DCR2)	*	unchanged
Interrupt Mask	0000h	unchanged
Interrupt Status	0000h	unchanged
Transmit Control	0101h	unchanged
Receive Control	**	unchanged
End Of Buffer Count	02F8h	unchanged
Sequence Counters	0000h	unchanged
CAM Enable	0000h	unchanged

*Bits 15 and 13 of the DCR and bits 4 through 0 of the DCR2 are reset to a 0 during a hardware reset. Bits 15-12 of the DCR2 are unknown until written to. All other bits in these two registers are unchanged.

**Bits LB1, LB0 and BRD are reset to a 0 during hardware reset. All other bits are unchanged.

5.4.9 Chip Reset

The SONIC has two reset modes; a hardware reset and a software reset. The SONIC can be hardware reset by asserting the $\overline{RESET}$ pin or software reset by setting the RST bit in the Command Register (section 4.3.1). The two reset modes are not interchangeable since each mode performs a different function.

After power-on, the SONIC must be hardware reset before it will become operational. This is done by asserting $\overline{RESET}$ for a minimum of 10 transmit clocks (10 ethernet transmit clock periods, TXC). If the bus clock (BSCK) period is greater than the transmit clock period, $\overline{RESET}$ should be asserted for 10 bus clocks instead of 10 transmit clocks. A hardware reset places the SONIC in the following state. (The registers affected are listed in parentheses. See Table 5-4 and section 4.3 for more specific information about the registers and how they are affected by a hardware reset. Only those registers listed below and in Table 5-4 are affected by a hardware reset.)

1. Receiver and Transmitter are disabled (CR).
2. The General Purpose timer is halted (CR).
3. All interrupts are masked out (IMR).
4. The NCRS and PTX status bits in the Transmit Control Register (TCR) are set.
5. The End Of Byte Count (EOBC) register is set to 02F8h (760 words).
6. Packet and buffer sequence number counters are set to zero.
7. All CAM entries are disabled. The broadcast address is also disabled (CAM Enable Register and the RCR).
8. Loopback operation is disabled (RCR).
9. The latched bus retry is set to the unlatched mode (DCR).
10. All interrupt status bits are reset (ISR).
11. The Extended Bus Mode is disabled (DCR).
12. HOLD will be asserted/deasserted from the falling clock edge (DCR2).

5.0 Bus Interface (Continued)

13. Latched Ready Mode is disabled (DCR2).
14. $\overline{\text{PCOMP}}$ will not be asserted (DCR2).
15. Packets will be accepted (not rejected) on CAM match (DCR2).

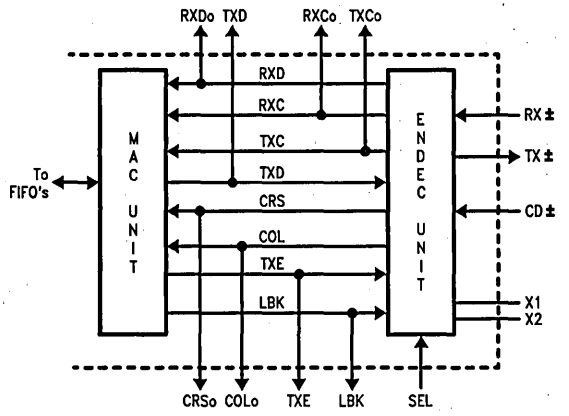
A software reset immediately terminates DMA operations and future interrupts. The chip is put into an idle state where registers can be accessed, but the SONIC will not be active in any other way. The registers are affected by a software reset as shown in Table 5-4 (only the Command Register is changed).

6.0 Network Interfacing

The SONIC contains an on-chip ENDEC that performs the network interfacing between the AUI (Attachment Unit Interface) and the SONIC's MAC unit. A pin selectable option allows the internal ENDEC to be disabled and the MAC/ENDEC signals to be supplied to the user for connection to an external ENDEC. If the EXT pin is tied to ground (EXT = 0) the internal ENDEC is selected and if EXT is tied to V_{CC} (EXT = 1) the external ENDEC option is selected.

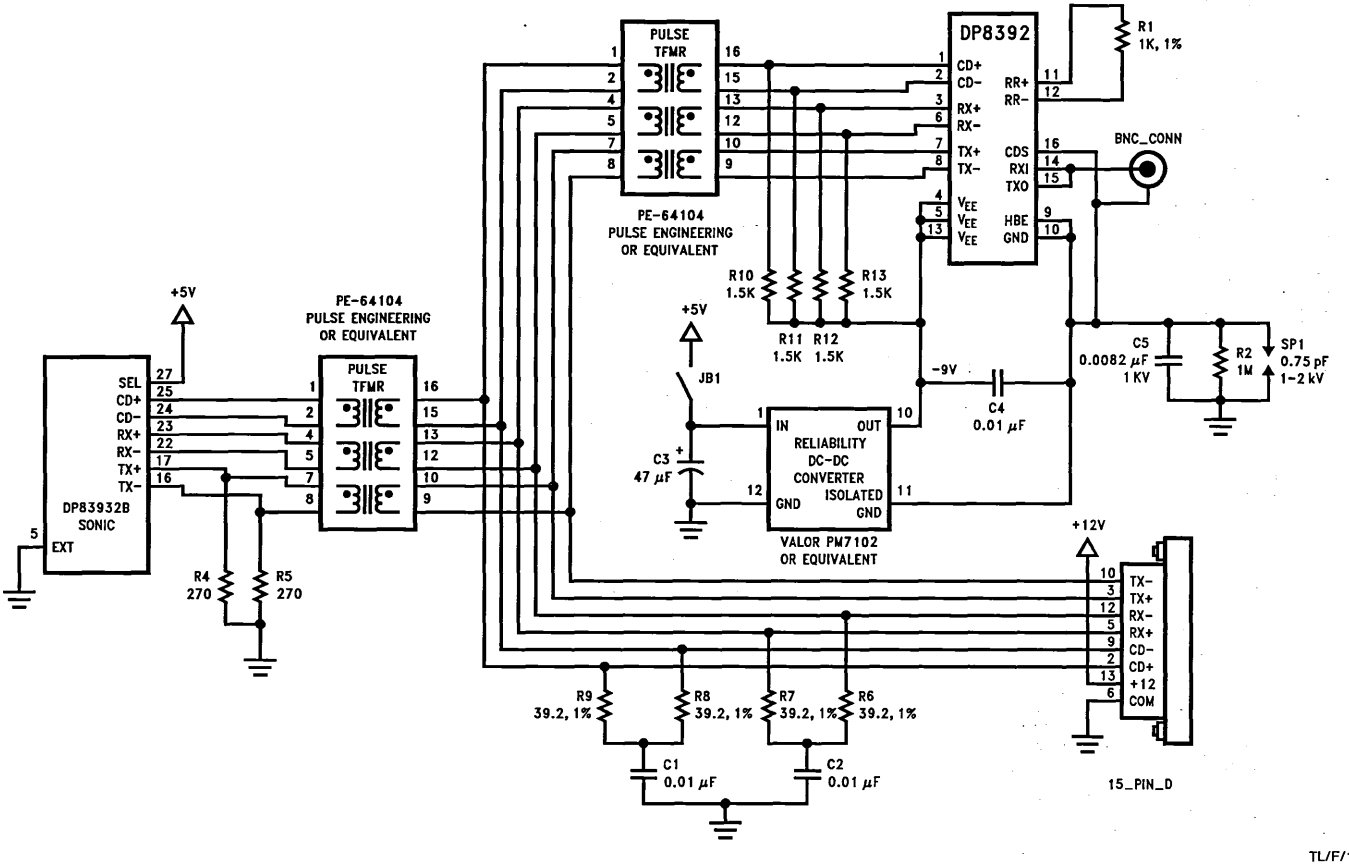
Internal ENDEC: When the internal ENDEC is used (EXT = 0) the interface signals between the ENDEC and MAC unit are internally connected. While these signals are used internally by the SONIC they are also provided as an output to the user *(Figure 6-1)*.

The internal ENDEC allows for a 2-chip solution for the complete Ethernet interface. *Figure 6-2* shows a typical diagram of the network interface.



TL/F/10492–52

FIGURE 6-1. MAC and Internal ENDEC Interface Signals



TL/F/10492–53

FIGURE 6-2. Network Interface Example (EXT = 0, Using a Single Jumper, JB1, for Network Interface Selection)

6.0 Network Interfacing (Continued)

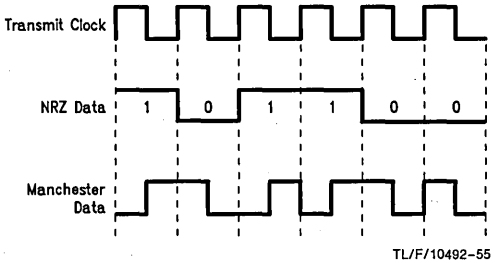
External ENDEC: When EXT = 1 the internal ENDEC is bypassed and the signals are provided directly to the user. Since SONIC's on-chip ENDEC is the same as National's DP83910 Serial Network Interface (SNI) the interface considerations discussed in this section would also apply to using this device in the external ENDEC mode.

6.1 MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The ENDEC unit's encoder begins operation when the MAC section begins sending the serial data stream. It converts NRZ data from the MAC section to Manchester data for the differential drivers (TX+/−). In Manchester encoding, the first half of the bit cell contains the complementary data and the second half contains the true data *(Figure 6-3)*. A transition always occurs at the middle of the bit cell. As long as the MAC continues sending data, the ENDEC section remains in operation. At the end of transmission, the last transition is always positive, occurring at the center of the bit cell if the last bit is a one, or at the end of the bit cell if the last bit is a zero.

The differential transmit pair drives up to 50 meters of twisted pair AUI cable. These outputs are source followers which require two 270Ω pull-down resistors to ground. In addition, a pulse transformer is required between the transmit pair output and the AUI interface.

The driver allows both half-step and full-step modes for compatibility with Ethernet I and IEEE 802.3. When the SEL pin is tied to ground (for Ethernet I), TX+ is positive with respect to TX− during idle on the primary side of the isolation transformer (*Figure 6-2*). When SEL is tied to V_{CC} (for IEEE 802.3), TX+ and TX− are equal in the idle state.



TL/F/10492–55

FIGURE 6.3. Manchester Encoded Data Stream

6.1.1 Manchester Decoder

The decoder consists of a differential receiver and a phase lock loop (PLL) to separate the Manchester encoded data stream into clock signals and NRZ data. The differential input must be externally terminated with two 39Ω resistors connected in series. In addition, *a pulse transformer is required between the receive input pair and the AUI interface.*

To prevent noise from falsely triggering the decoder, a squelch circuit at the input rejects signals with a magnitude less than −175 mV. Signals more negative than −300 mV are decoded.

Once the input exceeds the squelch requirements, the decoder begins operation. The decoder may tolerate bit jitter up to 18 ns in the received data. The decoder detects the end of a frame within one and a half bit times after the last bit of data.

6.1.2 Collision Translator

When the Ethernet transceiver (DP8392 CTI) detects a collision, it generates a 10 MHz signal to the differential collision inputs (CD+ and CD−) of the SONIC. When SONIC detects these inputs active, its Collision translator converts the 10 MHz signal to an active collision signal to the MAC section. This signal causes SONIC to abort its current transmission and reschedule another transmission attempt.

The collision differential inputs are terminated the same way as the differential receive inputs and a pulse transformer is required between the collision input pair and the AUI interface. The squelch circuitry is also similar, rejecting pulses with magnitudes less than −175 mV.

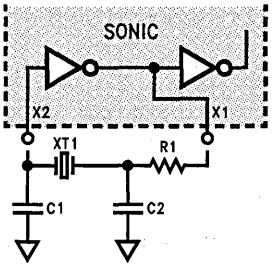
6.1.3 Oscillator Inputs

The oscillator inputs to the SONIC (X1 and X2) can be driven with a parallel resonant crystal or an external clock. In either case the oscillator inputs must be driven with a 20 MHZ signal. The signal is divided by 2 to generate the 10 MHz transmit clock (TXC) for the MAC unit. The oscillator also provides internal clock signals for the encoding and decoding circuits.

6.1.3.1 External Crystal

According to the IEEE 802.3 standard, the transmit clock (TXC) must be accurate to 0.01%. This means that the oscillator circuit, which includes the crystal and other parts involved must be accurate to 0.01% after the clock has been divided in half. Hence, when using a crystal, it is necessary to consider all aspects of the crystal circuit. An example of a recommended crystal circuit is shown in *Figure 6-4* and suggested oscillator specifications are shown in Table 6-1. The load capacitors in *Figure 6-4*, C1 and C2, should be no greater than 36 pF each, including all stray capacitance (see note 2 below). The resistor, R1, may be required in order to minimize frequency drift due to changes in V_{CC}. If R1 is required, its value must be carefully selected since R1 decreases the loop gain. If R1 is made too large, the loop gain will be greatly reduced and the crystal will not oscillate. If R1 is made too small, normal variations in V_{CC} may cause the oscillation frequency to drift out of specification. As a first rule of thumb, the value of R1 should be made equal to five times the motional resistance of the crystal. The motional resistance of 20 MHz crystals is usually in the range of 10Ω to 30Ω. This implies that reasonable values for R1 should be in the range of 50Ω to 150Ω. The decision of whether or not to include R1 should be based upon measured variations of crystal frequency as each of the circuit parameters are varied.

6.0 Network Interfacing (Continued)



TL/F/10492–81

FIGURE 6.4. Crystal Connection to the SONIC (see text)

Note 1: The X1 pin is not guaranteed to provide a TTL compatible logic output, and should not be used to drive any external logic. If additional logic needs to be driven, then an external oscillator should be used as described in the following section.

Note 2: The frequency marked on the crystal is usually measured with a fixed load capacitance specified in the crystal's data sheet. The actual load capacitance used should be the specified value minus the stray capacitance.

TABLE 6-1. Crystal Specifications

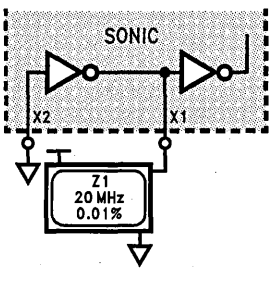
Resonant frequency	20 MHz
Tolerance (see text)	±0.01% at 25°C
Accuracy	±0.005% (50 ppm) at 0 to 70°C
Fundamental Mode Series Resistance	≤25Ω
Specified Load Capacitance	≤18 pF
Type	AT cut
Circuit	Parallel Resonance

6.1.3.2 Clock Oscillator Module

If an external clock oscillator is used, the SONIC can be connected to the external oscillator in one of two ways. The first configuration is shown in *Figure 6-5*. In this case, an oscillator that provides the following should be used:

1. TTL or CMOS output with a 0.01% frequency tolerance
2. 40%–60% duty cycle
3. ≥5 TTL loads output drive (I_{OL} = 8 mA) (Additional output drive may be necessary if the oscillator must also drive other components.)

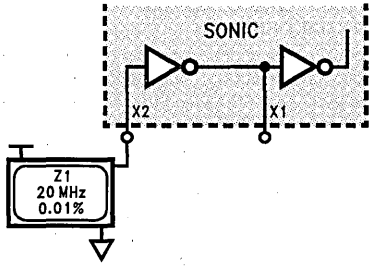
Again, the above assumes no other circuitry is driven.



TL/F/10492–82

FIGURE 6.5. Oscillator Module Connection to the SONIC

The second configuration, shown in *Figure 6-6*, connects to the X2 input. This connection requires an oscillator with the same specifications as the previous circuit except that the output drive specification need only be one CMOS load. This circuit configuration also offers the advantage of slightly lower power consumption. In this configuration, the X1 pin must be left open and should not drive external circuitry. Also, as shown by *Figure 6-6*, there is a 180° phase difference between connecting an oscillator to X1 compared to X2. This difference only affects the relationship between TXC and the oscillator module output. The operation of the SONIC is not affected by this phase change.



TL/F/10492–83

FIGURE 6.6. Alternate Oscillator Module Connection to the SONIC

6.1.3.3 PCB Layout Considerations

Care should be taken when connecting a crystal. Stray capacitance (e.g., from PC board traces and plated through holes around the X1 and X2 pins) can shift the crystal's frequency out of range, causing the transmitted frequency to exceed the 0.01% tolerance specified by IEEE. The layout considerations for using an external crystal are rather straightforward. The oscillator layout should locate all components close to the X1 and X2 pins and should use short traces that avoid excess capacitance and inductance. A solid ground should be used to connect the ground legs of the two capacitors.

When connecting an external oscillator, the only considerations are to keep the oscillator module as close to the SONIC as possible to reduce stray capacitance and inductance and to give the module a clean V_{CC} and a solid ground.

6.1.4 Power Supply Considerations

In general, power supply routing and design for the SONIC need only follow standard practices. In some situations, however, additional care may be necessary in the layout of the analog supply. Specifically special care may be needed for the TXVCC, RXVCC and PLLVCC power supplies and the TXGND and ANGND. In most cases the analog and digital power supplies can be interconnected. However, to ensure optimum performance of the SONIC's analog functions, power supply noise should be minimized. To reduce analog supply noise, any of several techniques can be used.

1. Route analog supplies as a separate set of traces or planes from the digital supplies with their own decoupling capacitors.
2. Provide noise filtering on the analog supply pins by inserting a low pass filter. Alternatively, a ferrite bead could be used to reduce high frequency power supply noise.
3. Utilize a separate regulator to generate the analog supply.

7.0 AC and DC Specifications

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to 7.0V
DC Input Voltage (V_{IN}	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to 150°C
Power Dissipation (PD)	500 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C
ESD Rating (R_{ZAP} = 1.5k, C_{ZAP} = 120 pF)	TBD

PARAMETRICS DISCLAIMER

The current AC and DC specifications contained in this document are considered target design specifications and may not represent actual guaranteed tested timing parameters. This information represents simulated, as well as, limited sampled empirical data. Guaranteed specifications will be provided after full device characterization.

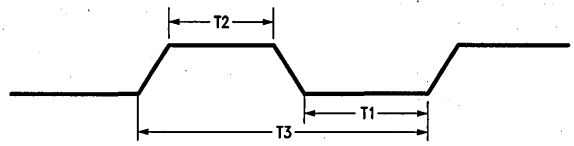
DC Specifications T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

Symbol	Parameter	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage	I_{OH} = −20 μA I_{OH} = −8 mA	V_{CC} − 0.1 3.0		V
V_{OL}	Maximum Low Level Output Voltage	I_{OL} = 20 μA I_{OL} = 8 mA		0.1 0.4	V
V_{IH}	Minimum High Level Input Voltage		2.0		V
V_{IL}	Maximum Low Level Input Voltage			0.8	V
I_{IN}	Input Current	V_{IN} = V_{CC} or GND	−1.0	1.0	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND	−10	10	μA
I_{CC}	Average Operating Supply Current	I_{OUT} = 0 mA, Freq = f_{max}		80	mA
AUI INTERFACE PINS (TX±, RX±, and CD±)					
V_{OD}	Diff. Output Voltage (TX±)	78Ω Termination, and 270Ω from Each to GND	±550	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±)	78Ω Termination, and 270Ω from Each to GND	Typical: 40 mV		
V_U	Undershoot Voltage (TX±)	78Ω Termination, and 270Ω from Each to GND	Typical: 80 mV		
V_{DS}	Diff. Squelch Threshold (RX± and CD±)		−175	−300	mV
OSCILLATOR PINS (X1 AND X2)					
V_{IH}	X1 Input High Voltage	X1 is Connected to an Oscillator and X2 is Grounded	2.0		V
V_{IL}	X1 Input Low Voltage	X1 is Connected to an Oscillator and X2 is Grounded		0.8	V
I_{OSC1}	X1 Input Current	X1 is Connected to an Oscillator and X2 is Grounded V_{IN} = V_{CC} or GND		8	mA
V_{IH}	X2 Input High Voltage	X2 is Connected to an Oscillator and X1 is Open	2.0		V
V_{IL}	X2 Input Low Voltage	X2 is Connected to an Oscillator and X1 is Open		0.8	V
I_{OSC2}	X2 Input Leakage Current	X2 is Connected to an Oscillator and X1 is Open V_{IN} = V_{CC} or GND	−10	10	μA

7.0 AC and DC Specifications (Continued)

AC Specifications

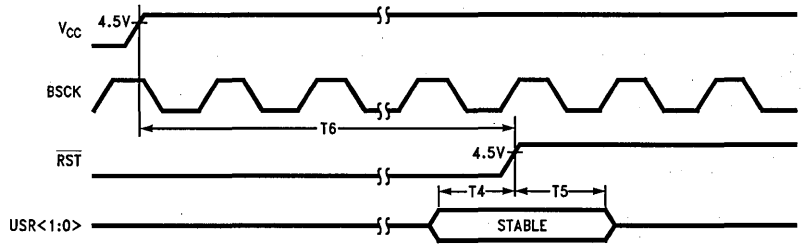
BUS CLOCK TIMING



TL/F/10492–56

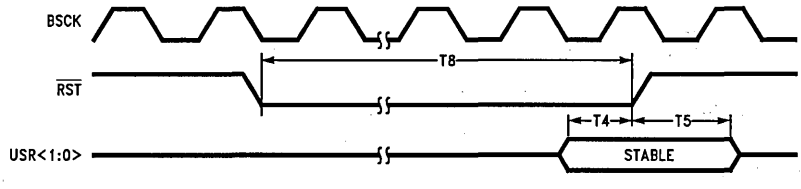
Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T1	bcl	Bus Clock Low Time	22.5		18		ns
T2	bch	Bus Clock High Time	22.5		18		ns
T3	bcyc	Bus Clock Cycle Time (Note 1)	50	100	40	100	ns

POWER-ON RESET



TL/F/10492–57

NON POWER-ON RESET



TL/F/10492–58

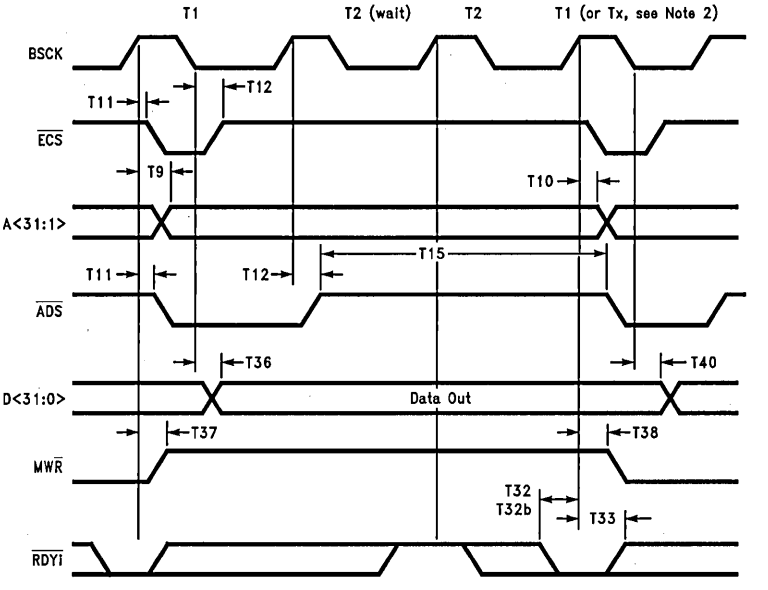
Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T4	usrs	USR<1:0> Setup to $\overline{RST}$	10		8		ns
T5	usrh	USR<1:0> Hold from $\overline{RST}$	10		8		ns
T6	pwr	Power-On Reset High	10		10		TXC (Note 2)
T8	rstw	Reset Pulse Width	10		10		TXC (Note 2)

Note 1: The maximum Bus Clock Cycle Time is untested, but is guaranteed by design.

Note 2: The reset time is determined by the slower of BSCK or TXC. If BSCK > TXC, T6 and T8 equal 10 TXCs. If BSCK < TXC, T6 and T8 equal 10 BSCKs (T3).

7.0 AC and DC Specifications (Continued)

MEMORY WRITE, BMODE = 0, SYNCHRONOUS MODE (one wait-state shown)



TL/F/10492–59

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11	bcsl	BSCK to $\overline{ADS}$, $\overline{ECS}$ Low		30		28	ns
T12	bcsh	BSCK to $\overline{ADS}$, $\overline{ECS}$ High		25		23	ns
T15	ashw	$\overline{ADS}$ High Width	bcyc-5 (Note 3)		bcyc-5		ns
T32	rdys	$\overline{RDYi}$ Setup to BSCK	25		25		ns
T32b	xrdys	$\overline{RDYi}$ Setup Time to BSCK (Latched Ready Mode, Note 2)	10		8		ns
T33	rdyh	$\overline{RDYi}$ Hold from BSCK	5		3		ns
T36	bcdv	BSCK to Memory Write Data Valid		50		45	ns
T37	bcwsv	BSCK to MW$\overline{R}$ (Write) Valid		30		28	ns
T38	bcwsinv	BSCK to MW$\overline{R}$ (Write) Invalid (Note 1)		0		0	ns
T40	wdh	Write Data Hold Time from BSCK	12		8		ns

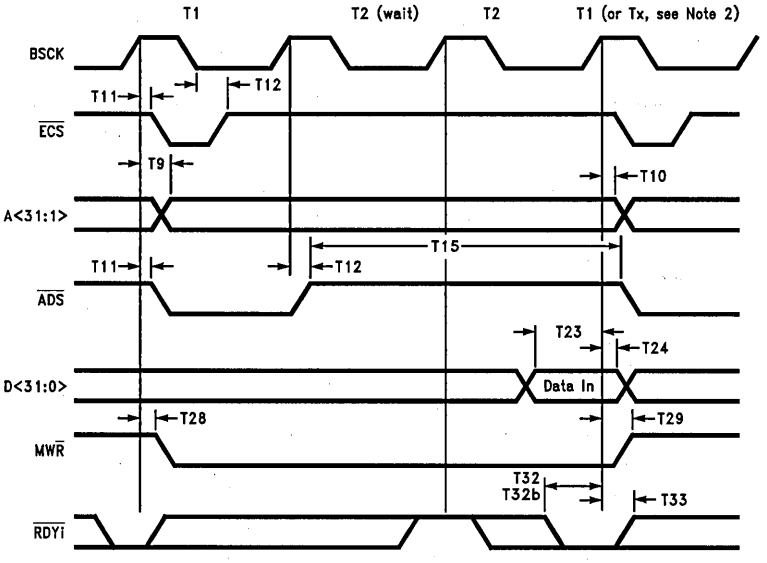
Note 1: For successive write operations, MW$\overline{R}$ remains high.

Note 2: If in "Latched Ready" mode (see Section 4.3.7), the SONIC will go into the Tx state before entering the T1 state, hence, there will be an extra clock cycle before the T1 state. $\overline{ECS}$, A<31:1>, MW$\overline{R}$, and $\overline{ADS}$ will, however, always be asserted off of the rising edge of T1, never Tx. The data will be held until T1.

Note 3: bcyc = bus clock cycle time (T3).

7.0 AC and DC Specifications (Continued)

MEMORY READ, BMODE = 0, SYNCHRONOUS MODE (one wait-state shown)



TL/F/10492–60

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Max	Min	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11	bcsl	BSCK to $\overline{ADS}$, $\overline{ECS}$ Low		30		28	ns
T12	bcsh	BSCK to $\overline{ADS}$, $\overline{ECS}$ High		25		23	ns
T15	ashw	$\overline{ADS}$ High Width	bcyc − 5 (Note 3)		bcyc − 5		ns
T23	rdsbc	Read Data Setup Time to BSCK	11		5		ns
T24	rdhbc	Read Data Hold Time from BSCK	6		3		ns
T28	bcrsv	BSCK to $\overline{MWR}$ (Read) Valid		30		28	ns
T29	bcrsinv	BSCK to $\overline{MWR}$ (Read) Invalid (Note 1)		0		0	ns
T32	rdys	$\overline{RDYi}$ Setup Time to BSCK	25		25		ns
T32b	xrdys	$\overline{RDYi}$ Setup Time to BSCK (Latched Ready Mode, Note 2)	10		8		ns
T33	rdyh	$\overline{RDYi}$ Hold Time to BSCK	5		3		ns

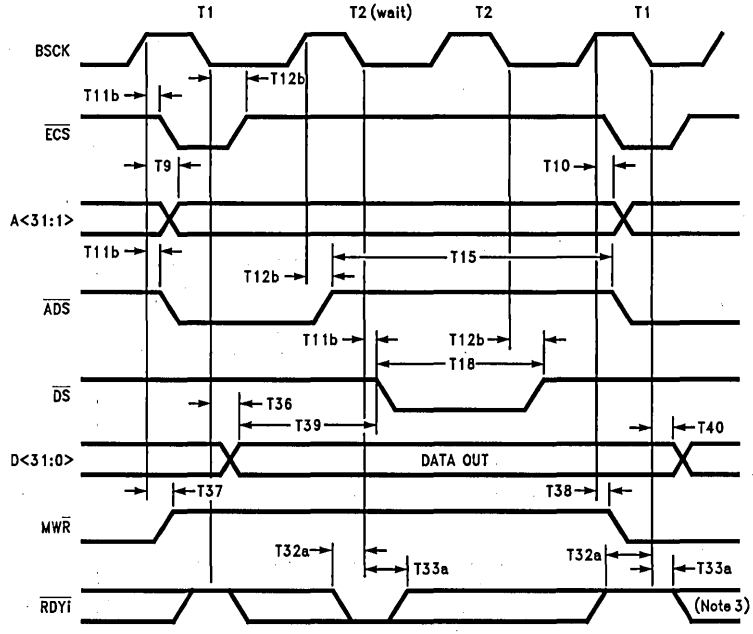
Note 1: For successive read operations, $\overline{MWR}$ remains low.

Note 2: If in "Latched Ready" mode (see Section 4.3.7), the SONIC will go into the Tx state before entering the T1 state, hence, there will be an extra clock cycle before the T1 state. $\overline{ECS}$, A<31:1>, $\overline{MWR}$, and $\overline{ADS}$ will, however, always be asserted off of the rising edge of T1, never Tx. The data will be latched in on the rising edge of Tx in this mode.

Note 3: bcyc = bus clock cycle time (T3).

7.0 AC and DC Specifications (Continued)

MEMORY WRITE, BMODE = 0, ASYNCHRONOUS MODE



TL/F/10492-61

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11b	bcsl	BSCK to $\overline{ADS}$, $\overline{DS}$, $\overline{ECS}$ Low		30		28	ns
T12b	bcsh	BSCK to $\overline{ADS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T15	ashw	$\overline{ADS}$ High Width	bcyc − 5 (Note 2)		bcyc − 5		ns
T18	wdslw	Write Data Strobe Low Width (Note 4)	bcyc − 5		bcyc − 5		ns
T32a	rdys	Ready Asynch. Setup to BSCK (Note 3)	8		5		ns
T33a	rdyh	Ready Asynch. Hold from BSCK	5		3		ns
T36	bcdv	BSCK to Memory Write Data Valid		50		45	ns
T37	bcwsv	BSCK to MW$\overline{R}$ (Write) Valid		30		28	ns
T38	bcwsinv	BSCK to MW$\overline{R}$ (Write) Invalid (Note 1)		0		0	ns
T39	ddsl	Write Data Valid to Data Strobe Low	10		7		ns
T40	wdh	Write Data Hold Time from BSCK	12		8		ns

Note 1: For successive write operations, MW$\overline{R}$ remains high.

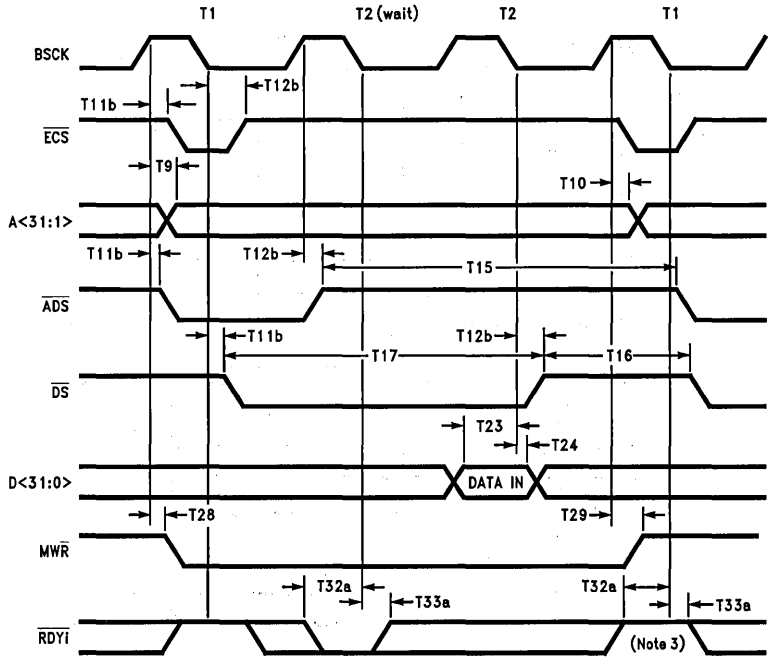
Note 2: bcyc = bus clock cycle time (T3)

Note 3: This setup time assures that the SONIC terminates the memory cycle on the next bus clock (BSCK). $\overline{RDYi}$ does not need to be synchronized to the bus clock, though, since it is an asynchronous input in this case. $\overline{RDYi}$ is sampled during the falling edge of BSCK. If the SONIC samples $\overline{RDYi}$ low during the T1 cycle, the SONIC will finish the current access in a total of two bus clocks instead of three, which would be the case if $\overline{RDYi}$ had been sampled low during T2(wait). (This is assuming that programmable wait states are set to 0).

Note 4: $\overline{DS}$ will only be asserted if the bus cycle has at least one wait state inserted..

7.0 AC and DC Specifications (Continued)

MEMORY READ, BMODE = 0, ASYNCHRONOUS MODE



TL/F/10492-62

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11b	bcsl	BSCK to $\overline{ADS}$, $\overline{DS}$, $\overline{ECS}$ Low		30		28	ns
T12b	bcsh	BSCK to $\overline{ADS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T15	ashw	$\overline{ADS}$ High Width	bcyc − 5 (Note 2)		bcyc − 5		ns
T16	rdshw	Read Data Strobe High Width	bcyc − 10		bcyc − 10		ns
T17	rdslw	Read Data Strobe Low Width	bcyc − 5		bcyc − 5		ns
T23	rdsbc	Read Data Setup Time to BSCK	11		5		ns
T24	rdhbc	Read Data Hold Time from BSCK	6		3		ns
T28	bcrsv	BSCK to $\overline{MWR}$ (Read) Valid		30		28	ns
T29	bcrsinv	BSCK to $\overline{MWR}$ (Read) Invalid (Note 1)		0		0	ns
T32a	rdys	Ready Asynch. Setup Time to BSCK (Note 3)	8		5		ns
T33a	rdyh	Ready Asynch. Hold Time to BSCK	5		3		ns

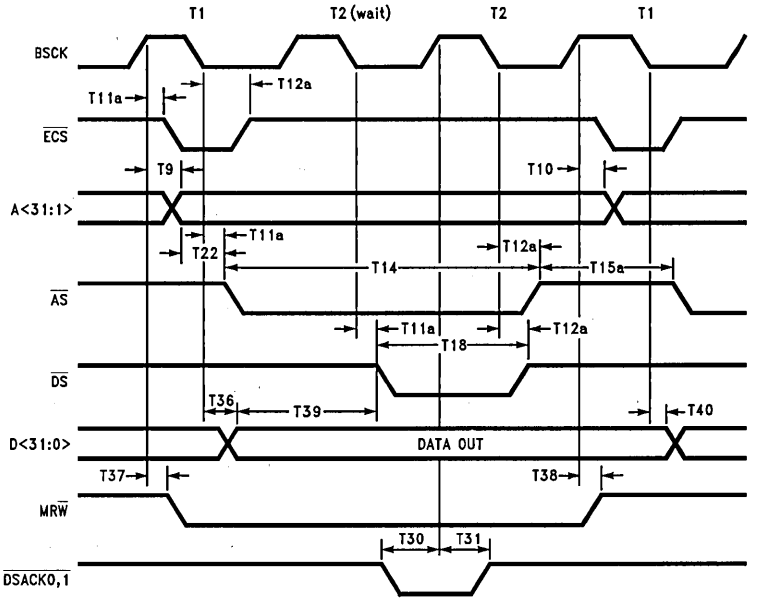
Note 1: For successive read operations, $\overline{MWR}$ remains low.

Note 2: bcyc = bus clock cycle time (T3)

Note 3: This setup time assures that the SONIC terminates the memory cycle on the next bus clock (BSCK). $\overline{RDYi}$ does not need to be synchronized to the bus clock, though, since it is an asynchronous input in this case. $\overline{RDYi}$ is sampled during the falling edge of BSCK. If the SONIC samples $\overline{RDYi}$ low during the T1 cycle, the SONIC will finish the current access in a total of two bus clocks instead of three, which would be the case if $\overline{RDYi}$ had been sampled low during T2(wait). (This is assuming that programmable wait states are set to 0).

7.0 AC and DC Specifications (Continued)

MEMORY WRITE, BMODE = 1, SYNCHRONOUS MODE (one wait-state shown)



TL/F/10492–63

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11a	bcsl	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ Low		20		18	ns
T12a	bcsh	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T14	aslw	$\overline{AS}$ Strobe Low Width	bcyc − 5 (Note 3)		bcyc − 5 (Note 3)		ns
T15a	ashw1	$\overline{AS}$ Strobe High Width	bcyc − 5		bcyc − 5		ns
T18	wdslw	Write Data Strobe Low Width (Note 1)	bcyc − 5		bcyc − 5		ns
T22	advas1	Address Valid to $\overline{AS}$	10		7		ns
T30	acks	$\overline{DSACK0,1}$ Setup to BSCK (Note 4)	6		4		ns
T31	ackh	$\overline{DSACK0,1}$ Hold from BSCK	12		8		ns
T36	bcdv	BSCK to Memory Write Data Valid		50		45	ns
T37	bcwsv	BSCK to MR$\overline{W}$ (Write) Valid		30		28	ns
T38	bcwinv	BSCK to MR$\overline{W}$ (Write) Invalid (Note 2)		0		0	ns
T39	ddsl	Write Data Valid to Data Strobe Low	10		7		ns
T40	wdh	Memory Write Data Hold Time from BSCK	12		8		ns

Note 1: $\overline{DS}$ will only be asserted if the bus cycle has at least one wait state inserted.

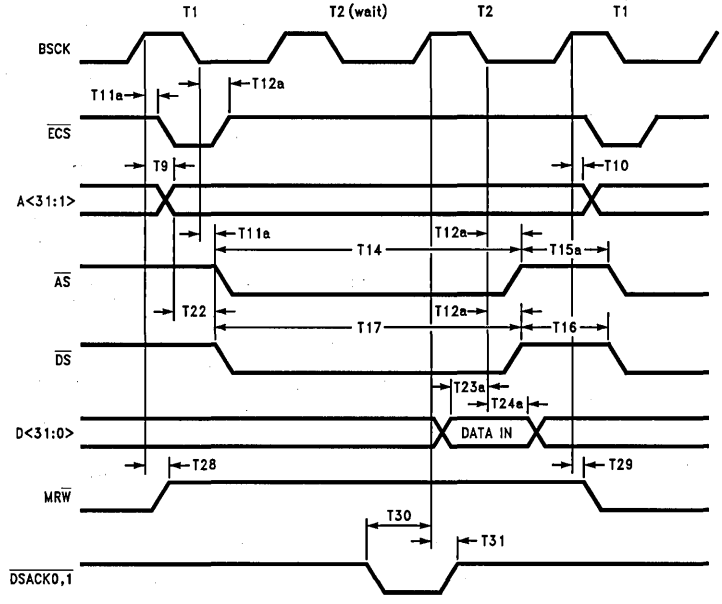
Note 2: For successive write operations, MR$\overline{W}$ remains low.

Note 3: bcyc = bus clock cycle time (T3).

Note 4: $\overline{DSACK0,1}$ must be synchronized to the bus clock (BSCK) during synchronous mode.

7.0 AC and DC Specifications (Continued)

MEMORY READ, BMODE = 1, SYNCHRONOUS MODE (one wait-state shown)



TL/F/10492-64

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11a	bcsl	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ Low		20		18	ns
T12a	bcsh	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T14	aslw1	$\overline{AS}$ Strobe Low Width	bcyc − 5 (Note 3)		bcyc − 5 (Note 3)		ns
T15a	ashw1	$\overline{AS}$ Strobe High Width	bcyc − 5		bcyc − 5		ns
T16	rdshw	Read Data Strobe High Width	bcyc − 10		bcyc − 10		ns
T17	rdslw	Read Data Strobe Low Width	bcyc − 5		bcyc − 5		ns
T22	advas1	Address Valid to $\overline{AS}$	10		7		ns
T23a	rdsbca	Read Data Setup Time to BSCK	8		6		ns
T24a	rdhbca	Read Data Hold Time from BSCK	5		5		ns
T28	bcrsv	BSCK to MR$\overline{W}$ (Read) Valid		30		28	ns
T29	bcrsinv	BSCK to MR$\overline{W}$ (Read) Invalid (Note 1)		0		0	ns
T30	acks	$\overline{DSACK0,1}$ Setup to BSCK (Note 2)	6		4		ns
T31	ackh	$\overline{DSACK0,1}$ Hold from BSCK	12		8		ns

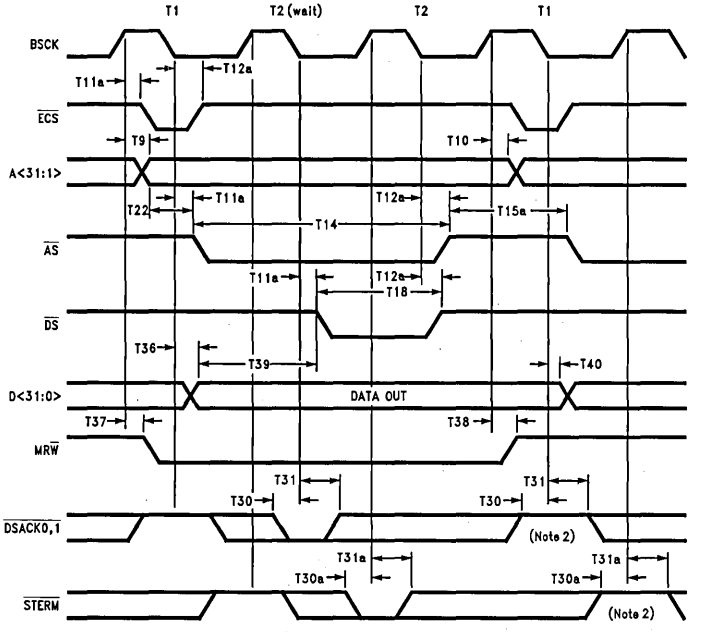
Note 1: For successive write operations, MR$\overline{W}$ remains low.

Note 2: $\overline{DSACK0,1}$ must be synchronized to the bus clock (BSCK) during synchronous mode.

Note 3: bcyc = bus clock cycle time (T3).

7.0 AC and DC Specifications (Continued)

MEMORY WRITE, BMODE = 1, ASYNCHRONOUS MODE



TL/F/10492–65

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11a	bcsl	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ Low		20		18	ns
T12a	bcsh	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T14	aslw	$\overline{AS}$ Strobe Low Width	bcyc − 5 (Note 3)		bcyc − 5		ns
T15a	ashw1	$\overline{AS}$ Strobe High Width	bcyc − 5		bcyc − 5		ns
T18	wdslw	Write Data Strobe Low Width (Note 4)	bcyc − 5		bcyc − 5		ns
T22	advas1	Address Valid to $\overline{AS}$	10		7		ns
T30	acks	$\overline{DSACK0,1}$ Setup to BSCK (Note 2)	6		4		ns
T30a	acks	$\overline{STERM}$ Setup to BSCK (Note 2)	6		4		ns
T31	ackh	$\overline{DSACK0,1}$ Hold from BSCK	12		8		ns
T31a	ackh	$\overline{STERM}$ Hold from BSCK	12		8		ns
T36	bcdv	BSCK to Memory Write Data Valid		50		45	ns
T37	bcwsv	BSCK to MR$\overline{W}$ (Write) Valid		30		28	ns
T38	bcwinv	BSCK to MR$\overline{W}$ (Write) Invalid (Note 1)		0		0	ns
T39	ddsl	Write Data Valid to Data Strobe Low	10		7		ns
T40	wdh	Memory Write Data Hold from BSCK	12		8		ns

Note 1: For successive write operations, MR$\overline{W}$ remains low.

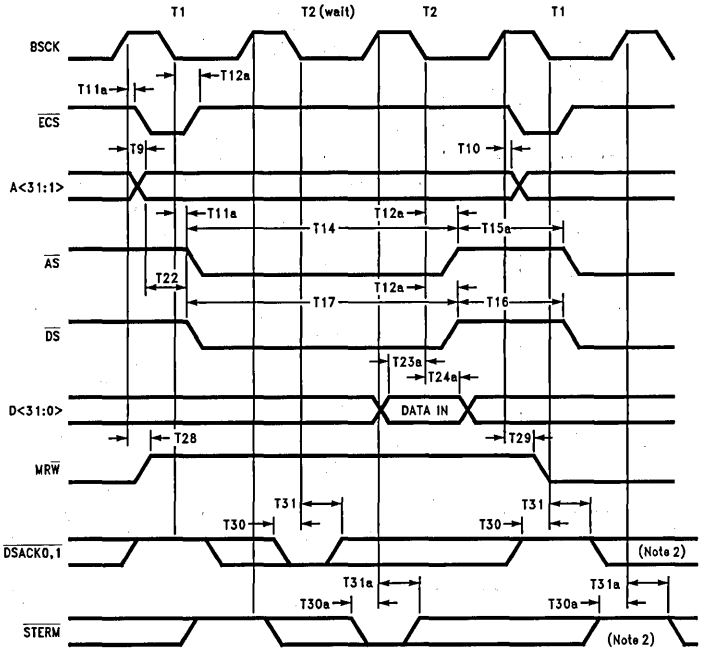
Note 2: Meeting the setup time for $\overline{DSACK0,1}$ or $\overline{STERM}$ guarantees that the SONIC will terminate the memory cycle 1½ bus clocks after $\overline{DSACK0,1}$ were sampled, or 1 cycle after $\overline{STERM}$ was sampled. T2 states will be repeated until $\overline{DSACK0,1}$ or $\overline{STERM}$ are sampled properly in a low state. If the SONIC samples $\overline{DSACK0,1}$ or $\overline{STERM}$ low during the T1 or first T2 state respectively, the SONIC will finish the current access in a total of two bus clocks instead of three (assuming that programmable wait states are set to 0). $\overline{DSACK0,1}$ are asynchronously sampled and $\overline{STERM}$ is synchronously sampled.

Note 3: bcyc = bus clock cycle time (T3).

Note 4: $\overline{DS}$ will only be asserted if the bus cycle has at least one wait state inserted.

7.0 AC and DC Specifications (Continued)

MEMORY READ, BMODE = 1, ASYNCHRONOUS MODE



TL/F/10492–66

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T9	bcadv	BSCK to Address Valid		30		28	ns
T10	ah	Address Hold Time from BSCK	0		0		ns
T11a	bcsl	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ Low		20		18	ns
T12a	bcsh	BSCK to $\overline{AS}$, $\overline{DS}$, $\overline{ECS}$ High		25		23	ns
T14	aslw	$\overline{AS}$ Strobe Low Width	bcyc − 5 (Note 3)		bcyc − 5 (Note 3)		ns
T15a	ashw1	$\overline{AS}$ Strobe High Width	bcyc − 5		bcyc − 5		ns
T16	rdshw	Read Data Strobe High Width	bcyc − 10		bcyc − 10		ns
T17	rdslw	Read Data Strobe Low Width	bcyc − 5		bcyc − 5		ns
T22	advas1	Address Valid to $\overline{AS}$	10		7		ns
T23a	rdsbca	Read Data Setup Time to BSCK	8		6		ns
T24a	rdhbca	Read Data Hold Time from BSCK	5		5		ns
T28	bcrsv	BSCK to MR$\overline{W}$ (Read) Valid		30		28	ns
T29	bcsinv	BSCK to MR$\overline{W}$ (Read) Invalid (Note 1)		0		0	ns
T30	acks	$\overline{DSACK0,1}$ Setup to BSCK (Note 2)	6		4		ns
T30a	acks	$\overline{STERM}$ Setup to BSCK (Note 2)	6		4		ns
T31	ackh	$\overline{DSACK0,1}$ Hold from BSCK	12		8		ns
T31a	ackh	$\overline{STERM}$ Hold from BSCK	12		8		ns

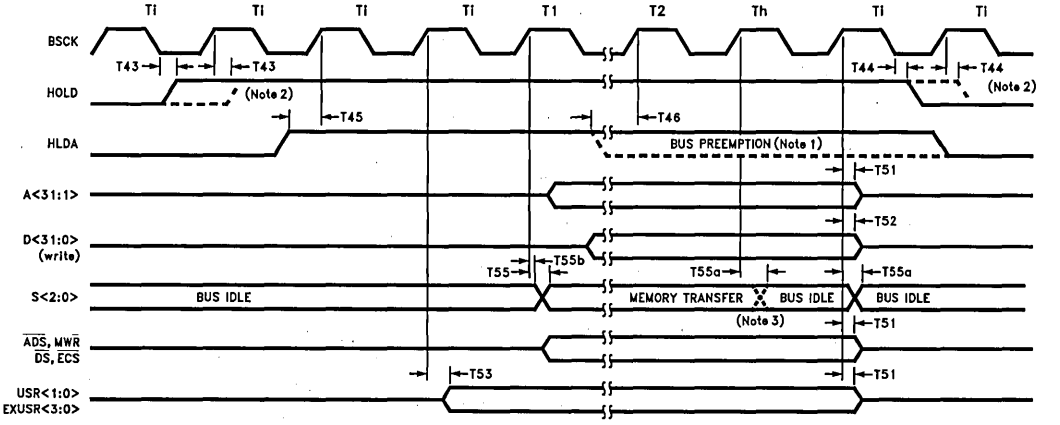
Note 1: For successive write operations, MR$\overline{W}$ remains low.

Note 2: Meeting the setup time for $\overline{DSACK0,1}$ or $\overline{STERM}$ guarantees that the SONIC will terminate the memory cycle 1½ bus clocks after $\overline{DSACK0,1}$ were sampled, or 1 cycle after $\overline{STERM}$ was sampled. T2 states will be repeated until $\overline{DSACK0,1}$ or $\overline{STERM}$ are sampled properly in a low state. If the SONIC samples $\overline{DSACK0,1}$ or $\overline{STERM}$ low during the T1 or first T2 state respectively, the SONIC will finish the current access in a total of two bus clocks instead of three (assuming that programmable wait states are set to 0). $\overline{DSACK0,1}$ are asynchronously sampled and $\overline{STERM}$ is synchronously sampled.

Note 3: bcyc = bus clock cycle time (T3).

7.0 AC and DC Specifications (Continued)

BUS REQUEST TIMING, BMODE = 0



TL/F/10492–67

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T43	bchldh	BSCK to HOLD High (Note 2)		22		20	ns
T44	bchldl	BSCK to HOLD Low (Note 2)		22		20	ns
T45	hldas	HLDA Asynchronous Setup Time to BSCK	10		5		ns
T46	hldads	HLDA Deassert Setup Time (Note 1)	10		5		ns
T51	bcaz	BSCK to Address, $\overline{ADS}$, MWR, $\overline{DS}$, $\overline{ECS}$, USR<1:0> and EXUSR<3:0> TRI-STATE (Note 4)		40		35	ns
T52	bcdz	BSCK to Data TRI-STATE (Note 4)		50		40	ns
T53	bcstrv	BSCK to USR<1:0> Valid		30		28	ns
T55	bcsvo	BSCK to Bus Status Idle to Non-Idle		38		35	ns
T55a	bcsv1	BSCK to Bus Status Non-Idle to Idle (Note 3)		25		22	ns
T55b	bcshd	S<2:0> Hold from BSCK	10		10		ns

Note 1: A block transfer by the SONIC can be pre-empted from the bus by deasserting HLDA provided HLDA is asserted T46 before the rising edge of the last T2 in the current access.

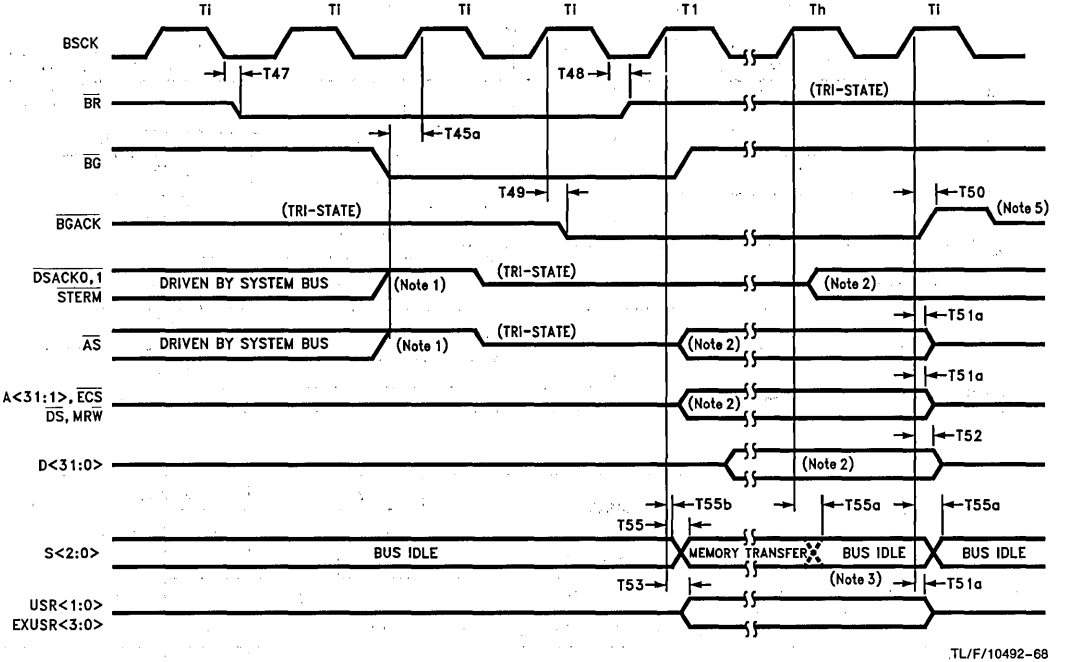
Note 2: The assertion edge for HOLD is dependent upon the PH bit in the DCR2. The default situation is shown wih a solid line in the timing diagram. T43 and T44 apply for both modes. Also, if HLDA is asserted when the SONIC wants to acquire the bus, HOLD will not be asserted until HLDA has been deasserted first.

Note 3: S<2:0> will indicate IDLE at the end of T2 if the last operation is a read operation, or at the end of Th if the last operation is a write operation.

Note 4: This timing value includes an RC delay inherent in the test measurement. These signals typically TRI-STATE ½ bus clock earlier, enabling other devices to drive these lines without contention.

7.0 AC and DC Specifications (Continued)

BUS REQUEST TIMING, BMODE = 1



TL/F/10492–68

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T45a	brs	$\overline{BG}$ Asynchronous Setup Time to BSCK	10		5		ns
T47	bcbrl	BSCK Low to $\overline{BR}$ Low		20		17	ns
T48	bcbrh	BSCK Low to $\overline{BR}$ TRI-STATE		30		28	ns
T49	bcbgackl	BSCK High to $\overline{BGACK}$ Low (Note 1)		20		15	ns
T50	bcgbgackh	BSCK High to $\overline{BGACK}$ High (Note 5)		20		15	ns
T51a	bcaz	BSCK to Address, $\overline{AS}$, MR$\overline{W}$, $\overline{DS}$, $\overline{ECS}$, USR<1:0> and EXUSR<3:0> TRI-STATE (Note 4)		40		35	ns
T52	bcdz	BSCK to Data TRI-STATE (Note 4)		50		40	ns
T53	bcstrv	BSCK to USR<1:0> Valid		30		28	ns
T55	bcsvo	BSCK to Bus Status Idle to Non-Idle		38		35	ns
T55a	bcsv1	BSCK to Bus Status Non-Idle to Idle (Note 3)		25		22	ns
T55b	bcsvd	S<2:0> Hold from BSCK	10		10		ns

Note 1: $\overline{BGACK}$ is only issued if $\overline{BG}$ is low and $\overline{AS}$, $\overline{DSACK0,1}$, $\overline{STERM}$ and $\overline{BGACK}$ are deasserted.

Note 2: For specific timing on these signals, see the memory read and memory write timing diagrams on previous pages.

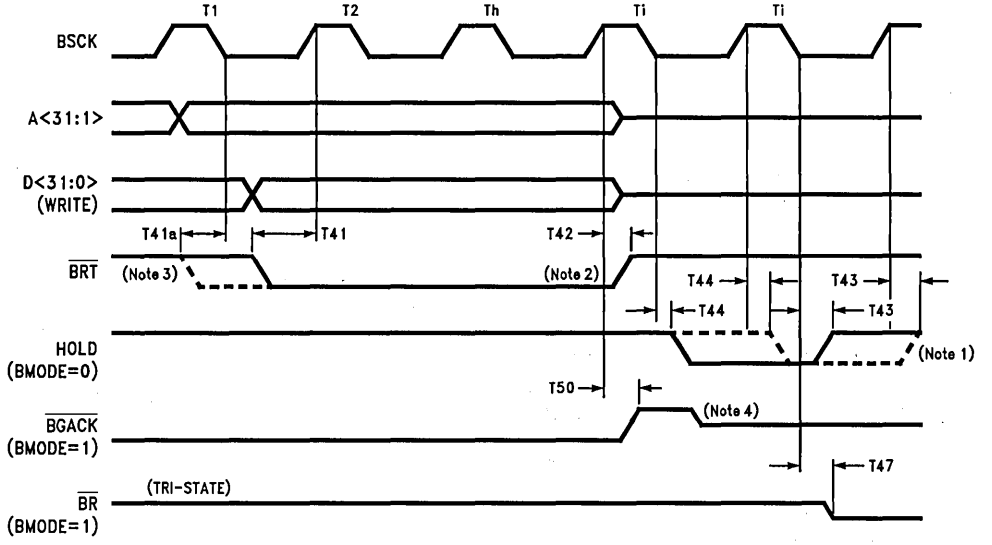
Note 3: S<2:0> will inidicate IDLE at the end of T2 if the last operation is a read operation or at the end of Th if the last operation is a write operation.

Note 4: This timing value includes an RC delay inherent in our test measurement. These signals typically TRI-STATE ½ bus clock eariler, enabling other devices to drive these lines without contention.

Note 5: $\overline{BGACK}$ is driven high for approximately ½ BSCK before going TRI-STATE.

7.0 AC and DC Specifications (Continued)

BUS RETRY



TL/F/10492–69

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T41	brts0	Bus Retry Synchronous Setup Time to BSCK (Note 3)	5		3		ns
T41a	brts1	Bus Retry Asynchronous Setup Time to BSCK (Note 3)	5		3		ns
T42	brth	Bus Retry Hold Time from BSCK (Note 2)	5		3		ns
T43	bchldh	BSCK to HOLD High (Note 1)		22		20	ns
T44	bchldl	BSCK to HOLD Low (Note 1)		22		20	ns
T47	bcbrl	BSCK to $\overline{BR}$ Low		20		17	ns
T50	bcbgackh	BSCK to $\overline{BGACK}$ High (Note 4)		20		15	ns

Note 1: Depending upon the mode, the SONIC will assert and deassert HOLD from the rising or falling edge of BSCK.

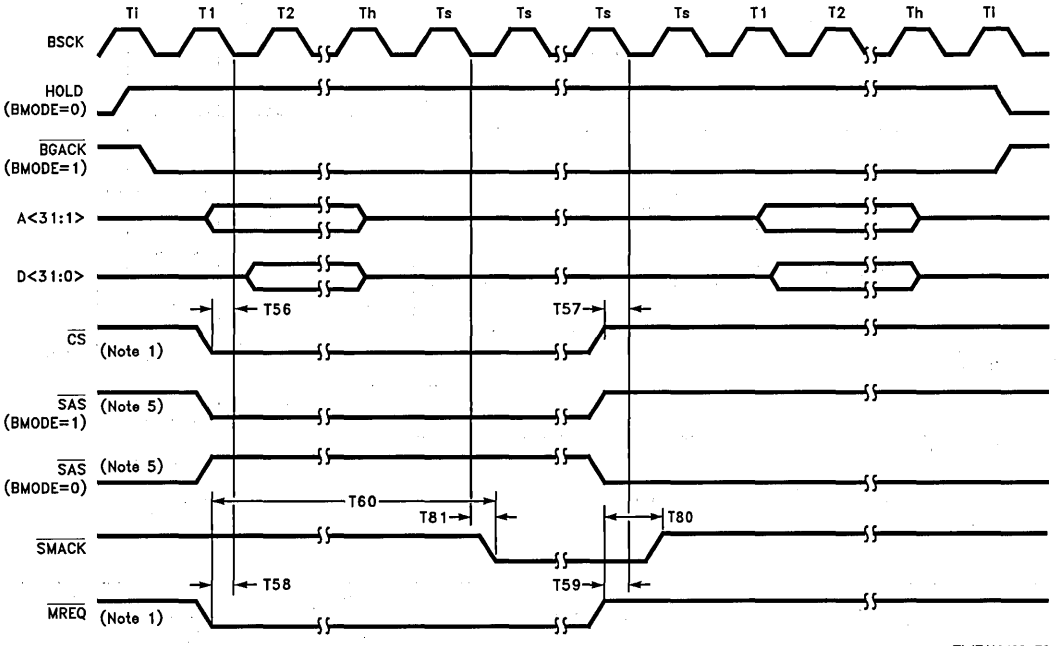
Note 2: Unless Latched Bus Retry mode is set (LBR in the Data Configuration Register, Section 4.3.2), $\overline{BRT}$ must remain asserted until after the Th state. If Latched Bus Retry mode is used, $\overline{BRT}$ does not need to satisfy T42.

Note 3: T41 is for synchronous bus retry and T41a is for asynchronous bus retry (see Section 4.3.2, bit 15, Extended Bus Mode). Since T41a is an asynchronous setup time, it is not necessary to meet it, but doing so will guarantee that the bus exception occurs in the current memory transfer, not the next.

Note 4: $\overline{BGACK}$ is driven high for approximately ½ BSCK before going TRI-STATE.

7.0 AC and DC Specifications (Continued)

MEMORY ARBITRATION/SLAVE ACCESS



TL/F/10492–70

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T56	cssbc	$\overline{CS}$ Low Asynch. Setup to BSCK (Note 2)	12		10		ns
T57	csdsbc	$\overline{CS}$ High Asynch. Setup to BSCK	12		10		ns
T58	mrsbc	$\overline{MREQ}$ Low Asynch. Setup to BSCK (Note 2)	12		10		ns
T59	mrdsbc	$\overline{MREQ}$ High Asynch. Setup to BSCK	12		10		ns
T60	mrmackl	$\overline{MREQ}$ or $\overline{CS}$ to $\overline{SMACK}$ Low (Note 3)		1.5 5.5		1.5 5.5	bcyc (Note 4)
T80	assackh	$\overline{CS}$ or $\overline{MREQ}$ to $\overline{SMACK}$ High		15		12	ns
T81	bcsackl	BSCK to $\overline{SMACK}$ Low		25		20	ns

Note 1: Both $\overline{CS}$ and $\overline{MREQ}$ must not be asserted concurrently. If these signals are successively asserted, there must be at least two bus clocks between the deasserting and asserting edges of these signals.

Note 2: It is not necessary to meet the setup times for $\overline{MREQ}$ or $\overline{CS}$ since these signals are asynchronously sampled. Meeting the setup time for these signals, however, makes it possible to use T60 to determine exactly when $\overline{SMACK}$ will be assorted.

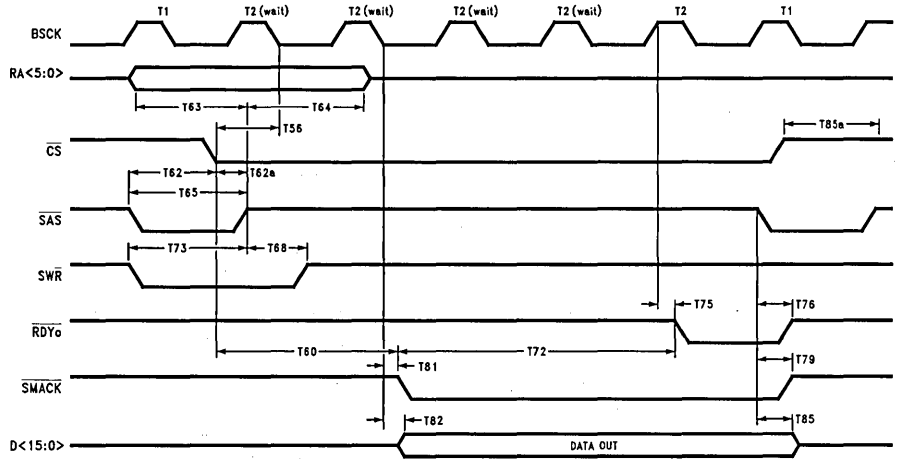
Note 3: The smaller value for T60 refers to when the SONIC is accessed during an Idle condition and the other value refers to when the SONIC is accessed during non-idle conditions. These values are not tested, but are guaranteed by design. This specification assumes that $\overline{CS}$ or $\overline{MREQ}$ is asserted ½ bus clock before the falling edge that these signals are asynchronously clocked in on (see T56 and T58). If T56 is met for $\overline{CS}$ or T58 is met for $\overline{MREQ}$, then $\overline{SMACK}$ will be asserted exactly 1 bus clock, when the SONIC was idle, or 5 bus clocks, when the SONIC was in master mode, after the edge that T56 and T58 refer to. (This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.)

Note 4: bcyc = bus clock cycle time (T3).

Note 5: The way in which $\overline{SMACK}$ is asserted is due to $\overline{CS}$ is not the same as the way in which $\overline{SMACK}$ is asserted due to $\overline{MREQ}$. $\overline{SMACK}$ goes low as a direct result of the assertion of $\overline{MREQ}$, whereas, for $\overline{CS}$, $\overline{SAS}$ must also be driven low (BMODE = 1) or high (BMODE = 0) before $\overline{SMACK}$ will be asserted. This means that when $\overline{SMACK}$ is asserted due to $\overline{MREQ}$, $\overline{SMACK}$ will remain asserted until $\overline{MREQ}$ is deasserted. Multiple memory accesses can be made to the shared memory without $\overline{SMACK}$ ever going high. When $\overline{SMACK}$ is asserted due to $\overline{CS}$, however, $\overline{SMACK}$ will only remain low as long as $\overline{SAS}$ is also low (BMODE = 1) or high (BMODE = 0). $\overline{SMACK}$ will not remain low throughout multiple register accesses to the SONIC because $\overline{SAS}$ must toggle for each register access. This in an important difference to consider when designing shared memory designs.

7.0 AC and DC Specifications (Continued)

REGISTER READ, BMODE = 0 (Note 1)



TL/F/10492–71

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T56	cssbc	$\overline{CS}$ Asynch. Setup to BSCK (Note 4)	12		10		ns
T60	mrmackl	$\overline{MREQ}$ or $\overline{CS}$ to $\overline{SMACK}$ Low (Notes 5, 8)		1.5 5.5		1.5 5.5	bcyc (Note 3)
T62	asscs0	$\overline{SAS}$ Assertion before $\overline{CS}$ (Note 6)	0		0		ns
T62a	asscs1	$\overline{SAS}$ Deassertion after $\overline{CS}$ (Note 6)		bcyc		bcyc	ns
T63	rsas	Register Address Setup to $\overline{SAS}$	10		7		ns
T64	rhas	Register Address Hold Time from $\overline{SAS}$	10		8		ns
T65	sasw	$\overline{SAS}$ Pulse Width	bcyc − 10		bcyc − 10		ns
T68	rshas	$\overline{SWR}$ (Read) Hold from $\overline{SAS}$	10		8		ns
T72	csrdyol	$\overline{SMACK}$ to $\overline{RDYo}$ Low (Note 8)	2.5		2.5		bcyc
T73	rdsas	$\overline{SWR}$ (Read) Setup to $\overline{SAS}$	5		2		ns
T75	bcrdyol	BSCK to $\overline{RDYo}$ Low		35		30	ns
T76	asrdyoh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{RDYo}$ TRI-STATE (Note 2)		20		20	ns
T79	dssackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{SMACK}$ High (Note 2)		15		12	ns
T81	bcsackl	BSCK to $\overline{SMACK}$ Low		25		20	ns
T82	bcrdv	BSCK to Register Data Valid		70		70	ns
T85	asdz	$\overline{SAS}$ or $\overline{CS}$ to Data TRI-STATE (Notes 2, 7)		50		40	ns
T85a	csds	Min. $\overline{CS}$ Deassert Time	1		1		bcyc

Note 1: This figure shows a slave access to the SONIC when the SONIC is idle, or rather not in master mode. If the SONIC is a bus master, there will be some differences as noted in the Memory Arbitration/Slave Access diagram. The BSCK states (T1, T2, etc.) are the equivalent processor states during a slave access.

Note 2: If $\overline{CS}$ is deasserted before the falling edge of $\overline{SAS}$, T76, T79 and T85 are referenced from the rising edge of $\overline{CS}$.

Note 3: bcyc = bus clock cycle time (T3).

Note 4: It is not necessary to meet the setup time for $\overline{CS}$ since this signal is asynchronously sampled. Meeting the setup time for this signal, however, makes it possible to use T60 to determine exactly when $\overline{SMACK}$ will be asserted.

Note 5: The smaller value for T60 refers to when the SONIC is accessed during an Idle condition and the other value refers to when the SONIC is accessed during non-idle conditions. These values are not tested, but are guaranteed by design. This specification assumes that $\overline{CS}$ is asserted ½ bus clock before the falling edge that $\overline{CS}$ is asynchronously clocked in on (see T56). If T56 is met for $\overline{CS}$, then $\overline{SMACK}$ will be asserted exactly 1 bus clock, when the SONIC was idle, or 5 bus clocks, when the SONIC was in master mode, after the edge that T56 refers to. (This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.)

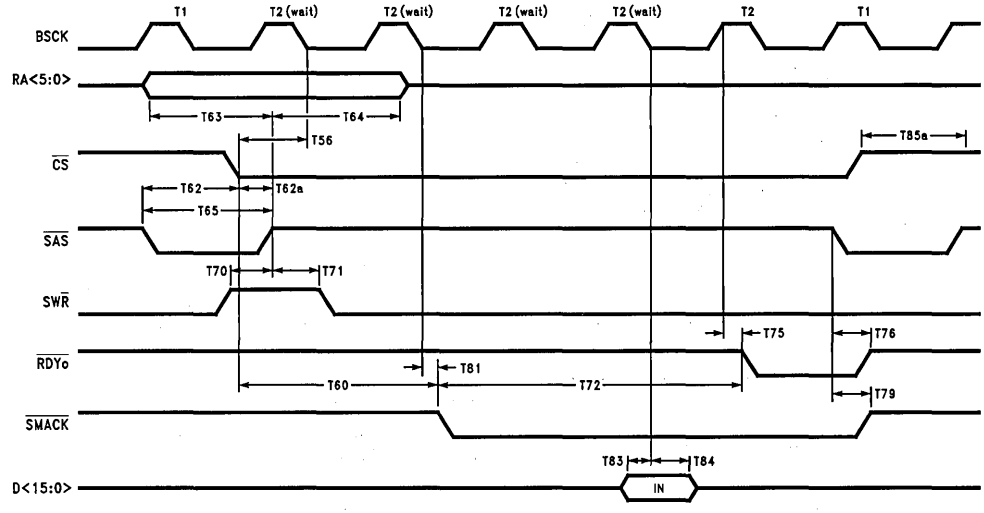
Note 6: $\overline{SAS}$ may be asserted low anytime before or simultaneous to the falling edge of $\overline{CS}$, but not after. It is suggested that $\overline{SAS}$ be driven high no later than 1 bus clock after $\overline{CS}$ is asserted low. $\overline{SAS}$ may be driven high anytime before the falling edge of $\overline{CS}$, however.

Note 7: This timing value includes an RC delay inherent in the test measurement. These signals typically TRI-STATE ½ bus clock earlier, enabling other devices to drive these lines without contention.

Note 8: These values are not tested, but are guaranteed by design. They are provided as a design guideline only.

7.0 AC and DC Specifications (Continued)

REGISTER WRITE, BMODE = 0 (Note 1)



TL/F/10492–72

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T56	cssbc	$\overline{CS}$ Asynch. Setup to BSCK (Note 4)	12		10		ns
T60	mrmackl	$\overline{MREQ}$ or $\overline{CS}$ to $\overline{SMACK}$ Low (Notes 5, 7)		1.5 5.5		1.5 5.5	bcyc (Note 3)
T62	asscs0	$\overline{SAS}$ Assertion before $\overline{CS}$ (Note 6)	0		0		ns
T62a	asscs1	$\overline{SAS}$ Deassertion after $\overline{CS}$ (Note 6)		bcyc		bcyc	ns
T63	rsas	Register Address Setup to $\overline{SAS}$	10		7		ns
T64	rhas	Register Address Hold Time from $\overline{SAS}$	10		8		ns
T65	sasw	$\overline{SAS}$ Pulse Width	bcyc − 10		bcyc − 10		ns
T70	wssas	$\overline{SWR}$ (Write) Setup to $\overline{SAS}$	5		2		ns
T71	wshas	$\overline{SWR}$ (Write) Hold from $\overline{SAS}$	10		8		ns
T72	csrdyol	$\overline{SMACK}$ to $\overline{RDYo}$ Low (Note 7)	2.5		2.5		bcyc
T75	bcrdyol	BSCK to $\overline{RDYo}$ Low		35		30	ns
T76	asrdyoh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{RDYo}$ TRI-STATE (Note 2)		20		20	ns
T79	dssackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{SMACK}$ High (Note 2)		15		12	ns
T81	bcsackl	BSCK to $\overline{SMACK}$ Low		25		20	ns
T83	rwds	Register Write Data Setup to BSCK	45		45		ns
T84	rwdh	Register Write Data Hold from BSCK	0		0		ns
T85a	csds	Min. $\overline{CS}$ Deassert Time	1		1		bcyc

Note 1: This figure shows a slave access to the SONIC when the SONIC is idle, or rather not in master mode. If the SONIC is a bus master, there will be some differences as noted in the Memory Arbitration/Slave Access diagram. The BSCK states (T1, T2, etc.) are the equivalent processor states during a slave access.

Note 2: If $\overline{CS}$ is deasserted before the falling edge of $\overline{SAS}$, T76 and T79 are referenced from the rising edge of $\overline{CS}$.

Note 3: bcyc = bus clock cycle time (T3).

Note 4: It is not necessary to meet the setup time for $\overline{CS}$ since this signal is asynchronously sampled. Meeting the setup time for this signal, however, makes it possible to use T60 to determine exactly when $\overline{SMACK}$ will be asserted.

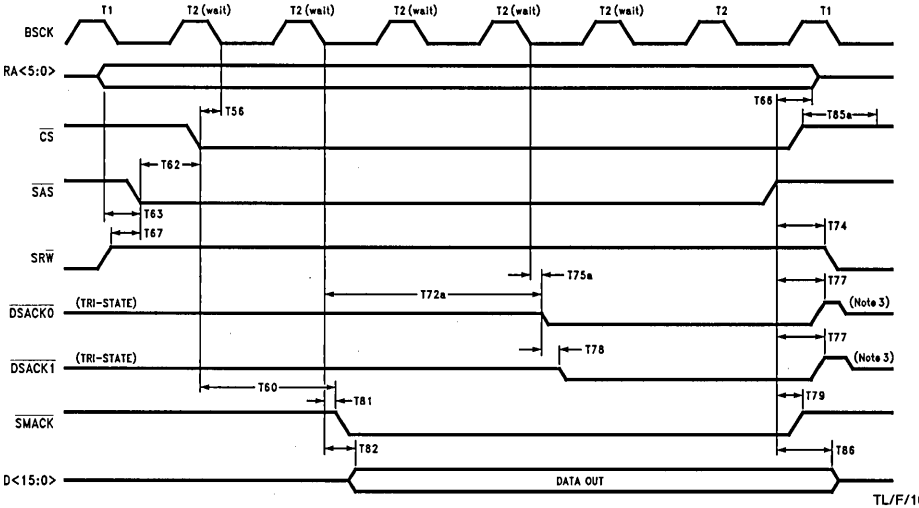
Note 5: The smaller value for T60 refers to when the SONIC is accessed during an Idle condition and the other value refers to when the SONIC is accessed during non-idle conditions. These values are not tested, but are guaranteed by design. This specification assumes that $\overline{CS}$ is asserted ½ bus clock before the falling edge that $\overline{CS}$ is asynchronously clocked in on (see T56). If T56 is met for $\overline{CS}$, then $\overline{SMACK}$ will be asserted exactly 1 bus clock, when the SONIC was idle, or 5 bus clocks, when the SONIC was in master mode, after the edge that T56 refers to. (This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.)

Note 6: $\overline{SAS}$ may be asserted low anytime before or simultaneous to the falling edge of $\overline{CS}$, but not after. It is suggested that $\overline{SAS}$ be driven high no later than 1 bus clock after $\overline{CS}$ is asserted low. $\overline{SAS}$ may be driven high anytime before the falling edge of $\overline{CS}$, however.

Note 7: These values are not tested, but are guaranteed by design. They are provided as a design guideline only.

7.0 AC and DC Specifications (Continued)

REGISTER READ, BMODE = 1 (Note 1)



TL/F/10492-73

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T56	cssbc	$\overline{CS}$ Asynch. Setup to BSCK (Note 5)	12		10		ns
T60	mrmackl	$\overline{MREQ}$ or $\overline{CS}$ to $\overline{SMACK}$ Low (Notes 6, 9)		1.5 5.5		1.5 5.5	bcyc (Note 4)
T62	asscs0	$\overline{SAS}$ Assertion before $\overline{CS}$ (Note 7)	0		0		ns
T63	rsas	Register Address Setup to $\overline{SAS}$	10		7		ns
T66	rhas1	Register Address Hold from $\overline{SAS}$ (Note 2)	0		0		ns
T67	rdsas	SR$\overline{W}$ (Read) Setup to $\overline{SAS}$	5		2		ns
T72a	csrdyol	$\overline{SMACK}$ to $\overline{DSACK0,1}$ Low (Note 9)	2		2		bcyc
T74	rshds	SR$\overline{W}$ (Read) Hold from $\overline{SAS}$ (Note 2)	0		0		ns
T75a	bcrdyol	BSCK to $\overline{DSACK0,1}$ Low		35		30	ns
T77	dsackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{DSACK0,1}$ High (Notes 2, 3)		25		20	ns
T78	skw	Skew between $\overline{DSACK0,1}$		10		7	ns
T79	dssackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{SMACK}$ High (Note 2)		15		12	ns
T81	bcsackl	BSCK to $\overline{SMACK}$ Low		25		20	ns
T82	bcrdv	BSCK to Register Data Valid		70		70	ns
T85a	csds	Min. $\overline{CS}$ Deassert Time	1		1		bcyc
T86	dsdz	$\overline{SAS}$ or $\overline{CS}$ to Register Data TRI-STATE (Notes 2, 8)		50		40	ns

Note 1: This figure shows a slave access to the SONIC when the SONIC is idle, or rather not in master mode. If the SONIC is a bus master, there will be some differences as noted in the Memory Arbitration/Slave Access diagram. The BSCK states (T1, T2, etc.) are the equivalent processor states during a slave access.

Note 2: If $\overline{CS}$ is deasserted before the rising edge of $\overline{SAS}$, T66, T74, T77, T79 and T86 are referenced off the rising edge of $\overline{CS}$ instead of $\overline{SAS}$.

Note 3: $\overline{DSACK0,1}$ are driven high for about ½ bus clock before going TRI-STATE.

Note 4: bcyc = bus clock cycle time (T3).

Note 5: It is not necessary to meet the setup time for $\overline{CS}$ since this signal is asynchronously sampled. Meeting the setup time for this signal, however, makes it possible to use T60 to determine exactly when $\overline{SMACK}$ will be asserted.

Note 6: The smaller value for T60 refers to when the SONIC is accessed during an Idle condition and the other value refers to when the SONIC is accessed during non-idle conditions. These values are not tested, but are guaranteed by design. This specification assumes that $\overline{CS}$ is asserted ½ bus clock before the falling edge that $\overline{CS}$ is asynchronously clocked in on (see T56). If T56 is met for $\overline{CS}$, then $\overline{SMACK}$ will be asserted exactly 1 bus clock, when the SONIC was idle, or 5 bus clocks, when the SONIC was in master mode, after the edge that T56 refers to. (This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.)

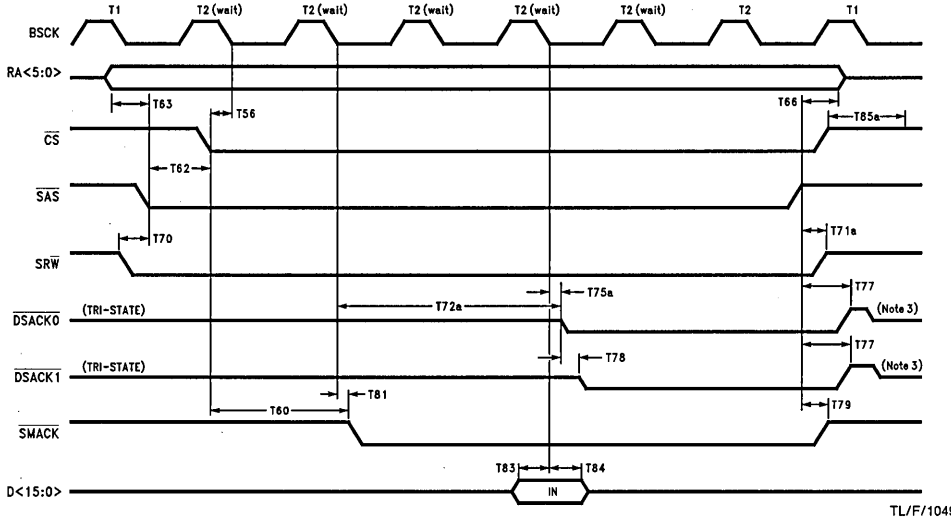
Note 7: $\overline{SAS}$ may be asserted at anytime before or simultaneous to the falling edge of $\overline{CS}$, but not after.

Note 8: This timing value includes an RC delay inherent in the test measurement. These signals typically TRI-STATE ½ bus clock eariler, enabling other devices to drive these lines without contention.

Note 9: These values are not tested, but are guaranteed by design. They are provided as a design guideline only.

7.0 AC and DC Specifications (Continued)

REGISTER WRITE, BMODE = 1 (Note 1)



TL/F/10492–74

Number	Symbol	Parameter	20 MHz		25 MHz		Units
			Min	Max	Min	Max	
T56	cssbc	$\overline{CS}$ Asynch. Setup to BSCK (Note 5)	12		10		ns
T60	mrmackl	$\overline{MREQ}$ or $\overline{CS}$ to $\overline{SMACK}$ Low (Notes 6, 9)		1.5 5.5		1.5 5.5	bcyc (Note 4)
T62	asscs0	$\overline{SAS}$ Assertion before $\overline{CS}$ (Note 7)	0		0		ns
T63	rsas	Register Address Setup to $\overline{SAS}$	10		7		ns
T66	rhas1	Register Address Hold from $\overline{SAS}$ (Note 2)	0		0		ns
T70a	wssas	$\overline{SRW}$ (Write) Setup to $\overline{SAS}$	5		2		ns
T71a	wshas	$\overline{SRW}$ (Write) Hold from $\overline{SDS}$ (Note 2)	10		8		ns
T72a	csrdyol	$\overline{SMACK}$ to $\overline{DSACK0,1}$ Low (Note 9)	2		2		bcyc
T75	bcrdyol	BSCK to $\overline{DSACK0,1}$ Low		35		30	ns
T77	dsackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{DSACK0,1}$ High (Notes 2, 3)		25		20	ns
T78	skw	Skew between $\overline{DSACK0,1}$		10		7	ns
T79	dssackh	$\overline{SAS}$ or $\overline{CS}$ to $\overline{SMACK}$ High (Note 2)		15		12	ns
T81	bcsackl	BSCK to $\overline{SMACK}$ Low		25		20	ns
T83	rwds	Register Write Data Setup to BSCK	45		45		ns
T84	rwdh	Register Write Data Hold from BSCK	0		0		ns
T85a	csds	Min. $\overline{CS}$ Deassert Time	1		1		bcyc

Note 1: This figure shows a slave access to the SONIC when the SONIC is idle, or rather not in master mode. If the SONIC is a bus master, there will be some differences as noted in the Memory Arbitration/Slave Access diagram. The BSCK states (T1, T2, etc.) are the equivalent processor states during a slave access.

Note 2: If $\overline{CS}$ is deasserted before the rising edge of $\overline{SAS}$, then T66, T71a, T77 and T79 are referenced off the rising edge of $\overline{CS}$ instead of $\overline{SAS}$.

Note 3: $\overline{DSACK0,1}$ are driven high for about ½ bus clock before going TRI-STATE.

Note 4: bcyc = bus clock cycle time (T3).

Note 5: It is not necessary to meet the setup time for $\overline{CS}$ since this signal is asynchronously sampled. Meeting the setup time for this signal, however, makes it possible to use T60 to determine exactly when $\overline{SMACK}$ will be asserted.

Note 6: The smaller value for T60 refers to when the SONIC is accessed during an Idle condition and the other value refers to when the SONIC is accessed during non-idle conditions. These values are not tested, but are guaranteed by design. This specification assumes that $\overline{CS}$ is asserted ½ bus clock before the falling edge that $\overline{CS}$ is asynchronously clocked in on (see T56). If T56 is met for $\overline{CS}$, then $\overline{SMACK}$ will be asserted exactly 1 bus clock, when the SONIC was idle, or 5 bus clocks, when the SONIC was in master mode, after the edge that T56 refers to. (This is assuming that there were no wait states in the current master mode access. Wait states will increase the time for $\overline{SMACK}$ to go low by the number of wait states in the cycle.)

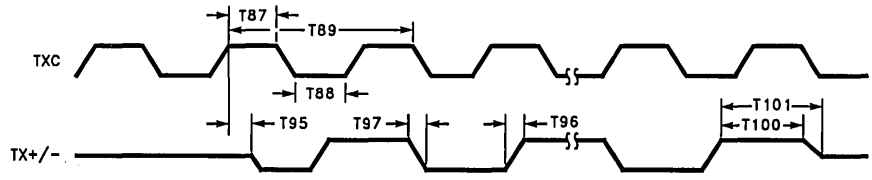
Note 7: $\overline{SAS}$ may be asserted at anytime before or simultaneous to the falling edge of $\overline{CS}$, but not after.

Note 8: This timing value includes an RC delay inherent in the test measurement. These signals typically TRI-STATE ½ bus clock eariler, enabling other devices to drive these lines without contention.

Note 9: These values are not tested, but are guaranteed by design. They are provided as a design guideline only.

7.0 AC and DC Specifications (Continued)

ENDEC TRANSMIT TIMING (INTERNAL ENDEC MODE)



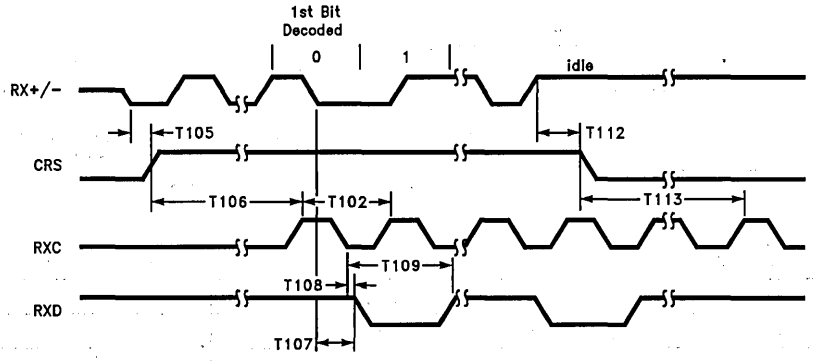
TL/F/10492–75

Number	Symbol	Parameter	Min	Max	Units
T87	txch	Transmit Clock High Time (Note 1)	40		ns
T88	txcl	Transmit Clock Low Time (Note 1)	40		ns
T89	txcc	Transmit Clock Cycle Time (Note 1)	99.99	100.01	ns
T95	tod	Transmit Output Delay (Note 1)		55	ns
T96	tof	Transmit Output Fall Time (80% to 20%, Note 1)		7	ns
T97	tor	Transmit Output Rise Time (20% to 80%, Note 1)		7	ns
T98	toj	Transmit Output Jitter (Not Shown)	0.5 Typ		ns
T100	toh	Transmit Output High before Idle (Half Step)	200		ns
T101	toi	Transmit Output Idle Time (Half Step)	8000		ns

Note 1: This specification is provided for information only and is not tested.

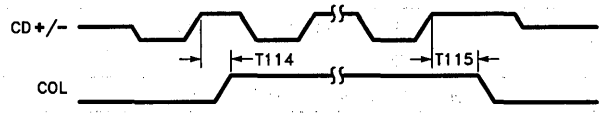
7.0 AC and DC Specifications (Continued)

ENDEC RECEIVE TIMING (INTERNAL ENDEC MODE)



TL/F/10492–76

ENDEC COLLISION TIMING



TL/F/10492–77

Number	Symbol	Parameter	Min	Max	Units
T102	rxcd	Receive Clock Duty Cycle Time (Note 1)	40	60	ns
T105	crson	Carrier Sense on Time		70	ns
T106	dat	Data Acquisition Time		700	ns
T107	rdd	Receive Data Output Delay		150	ns
T108	rdv	Receive Data Valid from RXC		10	ns
T109	rds	Receive Data Stable Valid Time	90		ns
T112	crsoff	Carrier Sense Off Delay (Note 2)		155	ns
T113	clks	Minimum Number of RXCs after CRS Low	5		rcyc (Note 3)
T114	colon	Collision Turn On Time		55	ns
T115	coloff	Collision Turn Off Time		250	ns

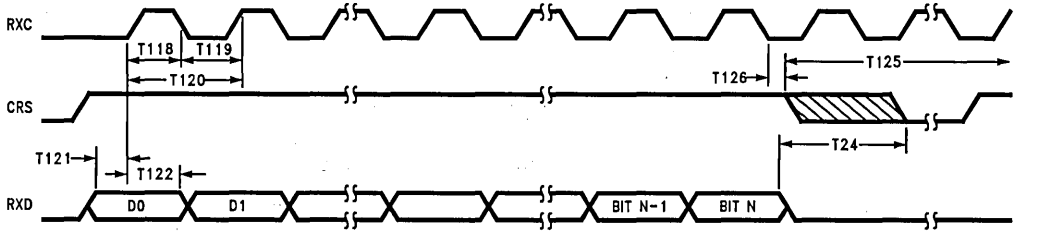
Note 1: This parameter is measured at the 50% point of each clock edge.

Note 2: When CRSi goes low, it remains low for a minimum of 2 receive clocks (RXCs).

Note 3: rcyc = receive clocks.

7.0 AC and DC Specifications (Continued)

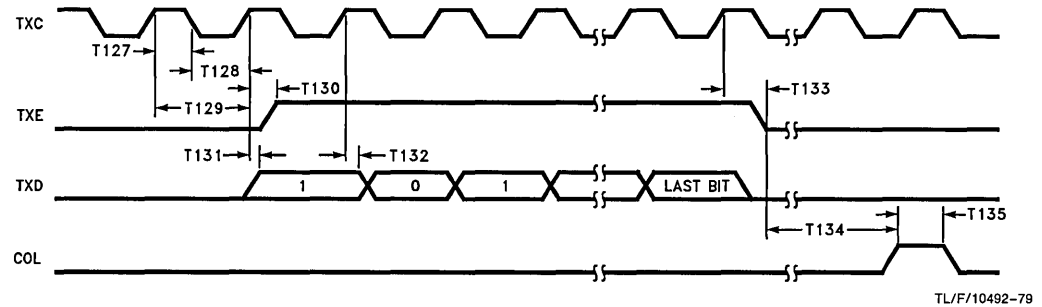
ENDEC-MAC SERIAL TIMING FOR RECEPTION (EXTERNAL ENDEC MODE)



TL/F/10492–78

Number	Symbol	Parameter	Min	Max	Units
T118	rch	Receive Clock High Time	35		ns
T119	rcl	Receive Clock Low Time	35		ns
T120	rcyc	Receive Clock Cycle Time	90	110	ns
T121	rds	RXD Setup to RXC	20		ns
T122	rdh	RXD Hold from RXC	15		ns
T124	drb	Maximum Allowed Dribble Bits		6	Bits
T125	rxrvy	Receive Recovery Time		(Note 2)	bcyc (Note 1)
T126	crsl	RXC to Carrier Sense Low		1	rcyc (Note 1)

ENDEC-MAC SERIAL TIMING FOR TRANSMIT (NO COLLISION)



TL/F/10492–79

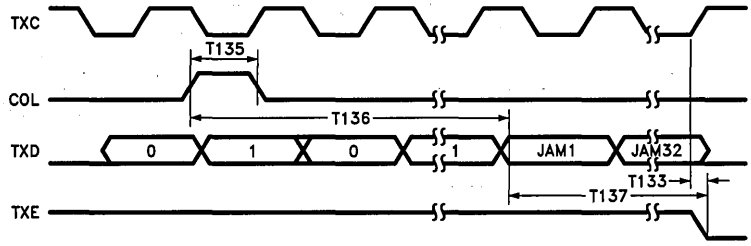
Number	Symbol	Parameter	Min	Max	Units
T127	txch	Transmit Clock High Time	40		ns
T128	txcl	Transmit Clock Low Time	40		ns
T129	tcyc	Transmit Clock Cycle Time	90	110	ns
T130	txcenh	TXC to TXE High		40	ns
T131	txds	TXC to TXD Valid		15	ns
T132	txdh	TXD Hold Time from TXC	5		ns
T133	txaenl	TXC to TXE Low		40	ns
T134	tdcdh	TXE Low to Start of CD Heartbeat		64	tcyc (Note 1)
T135	cdw	Collision Detect Width	2		tcyc (Note 1)

Note 1: tcyc = transmit clocks, rcyc = receive clocks, bcyc = T3.

Note 2: This parameter refers to longest time (not including wait-states) the SONIC requires to perform its end of receive processing and be ready for the next start of frame delimiter. This time is 4 tcyc + 36 bcyc. This is guaranteed by design and is not tested.

7.0 AC and DC Specifications (Continued)

ENDEC-MAC SERIAL TIMING FOR TRANSMISSION (COLLISION)



TL/F/10492–80

Number	Symbol	Parameter	Min	Max	Units
T135	cdw	Collision Detect Width	2		tcyc (Note 1)
T136	tcdj	Delay from Collision		8	tcyc
T137	tjam	Jam Period		32	tcyc

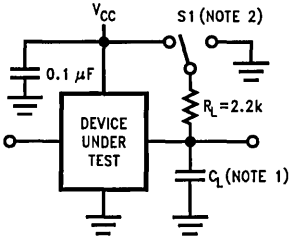
Note 1: tcyc = transmit clock.

8.0 AC Timing Test Conditions

All specifications are valid only if the mandatory isolation is employed and all differential signals are taken to be at the AUI side of the pulse transformer.

Input Pulse Levels (TTL/CMOS)	GND to 3.0V
Input Rise and Fall Times (TTL/CMOS)	5 ns
Input and Output Reference Levels (TTL/CMOS)	1.5V
Input Pulse Levels (Diff.)	−350 mV to −1315 mV
Input and Output Reference Levels (Diff.)	50% Point of the Differential
TRI-STATE Reference Levels	Float (ΔV) ±0.5V

Output Load (See Figure below)



TL/F/10492–84

Note 1: 50 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.

S1 = V_{CC} for V_{OL} test.

S1 = GND for V_{OH} test.

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active High to High Impedance measurements.

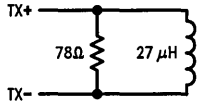
Capacitance T_A = 25°C, f = MHz

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	7	pF
C_{OUT}	Output Capacitance	7	pF

DERATING FACTOR

Output timing is measured with a purely capacitive load of 50 pF. The following correction factor can be used for other loads: $C_L \geq 50$ pF + 0.3 ns/pF.

AUI Transmit Test Load



TL/F/10492–85

Note: In the above diagram, the TX+ and TX− signals are taken from the AUI side of the isolation (pulse transformer). The pulse transformer used for all testing is a selected 100 μH ±0.1% Pulse Engineering PE64103.

DP83932 SONIC™ Bus Operations Guide

National Semiconductor
Application Note 745
Wesley Lee

This application note is intended to be a supplementary document for the DP83932 SONIC datasheet, expanding upon the bus functional descriptions found in the datasheet. It is recommended that you are familiar with the bus operations of the SONIC before reading this document.

This application note gives additional examples of the SONIC's bus operations to illustrate a broader picture of receptions, transmissions, etc. Where possible, special conditions are included to show all conceivable bus operations performed by the SONIC. Detailed figures are shown to enhance clarity. This document is divided into two sections for bus master and slave operations. The bus master section details bus operations during transmission, receptions and load CAM operations, and the slave access section describes SONIC register accesses during idle and non-idle conditions.

TERMS AND ABBREVIATIONS

In this document certain terms and abbreviations will be used to describe the bus operations of the SONIC. These words are defined as follows:

Block Transfer: A multiple transfer bus operation in which the address increments for each transfer.

Word: Refers to a 16-bit quantity; a double word is a 32-bit quantity.

Memory Cycle: The basic cycle which the SONIC reads from or writes to memory. The minimum cycle time is two bus clocks in synchronous mode and three bus clocks in asynchronous mode.

Bus Tenure: The complete time the SONIC uses the bus during a block transfer.

Bus Latency: This is the time from when the SONIC requests for the bus to when the SONIC is granted the bus.

CAM: Content Addressable Memory

TDA: Transmit Descriptor Area

TBA: Transmit Buffer Area

RRA: Receive Resource Area

RDA: Receive Descriptor Area

RBA: Receive Buffer Area

CDA: CAM Descriptor Area

1.0 BUS MASTER OPERATIONS

1.1 The Basic Block Transfer Cycle

The basic transfer cycle of the SONIC is composed of three basic operations: (1) acquiring the bus, (2) transferring data onto/from the bus, and (3) relinquishing the bus. Operations (1) and (3) are described in Section 5.4 of the datasheet and will not be discussed here. Operation (2), however, will be more fully explained.

When the SONIC uses the bus, it transfers data to/from one specific area in memory (i.e., RBA, RDA, RRA, TDA, or TBA) as indicated by the bus status pins S<2:0>. If the SONIC needs to transfer the data to multiple areas in memory, it deasserts its bus request (HOLD or $\overline{\text{BGACK}}$), then request for the bus again (HOLD or $\overline{\text{BR}}$). During its tenure on the bus, the SONIC transfers a programmed number of words to memory, depending on where data is placed. The number of transfers to the descriptor areas (TDA, RDA, RRA, and CDA), are shown in the following table. Note that since the upper word (D<31:16>) is not used in 32-bit mode, the number of transfer are the same for both 16-bit and 32-bit modes.

TABLE 1-1. Number of Memory Transfers to the Descriptor Areas

Area	Number	R/W	When
CDA	4	R	All bus tenures except the last one
	5	R	Last bus tenure. The additional access is to load the CAM Enable register.
TDA	6	R	First descriptor fetch
	3	R	Additional fragment pointer and size fetches, if any
	2	R/W	Status and link access
RDA	7	R/W	Updating receive descriptor information
	6	R/W	Updating receive descriptor information but SONIC has read EOL = 1
	2	R/W	Re-reading RXpkt.link and writing to RXpkt.in_use when EOL has previously been detected as 1. The SONIC writes to the in_use field when EOL now reads 0.
	1	R	Re-reading the RXpkt.link as above but the SONIC still reads EOL = 1.
RRA	4	R	All bus tenures

For buffer area transfers (TBA and RBA), the number of memory transfers is determined by the FIFO threshold and whether the SONIC is in "empty/fill' or "exact block" transfer modes, programmed in the Data Configuration register. For "exact block" transfer mode, the SONIC transfers the same number of words (or double words) as are programmed for the FIFO threshold. For example, if you programmed a 4 double words for the receive FIFO, the SONIC will transfer this amount of data to memory per bus tenure. There are two exceptions to this rule, however. First, during transmission or reception, if the packet is not a multiple of the FIFO threshold, the last bus tenure will contain less transfers than the FIFO threshold. Second, for high transmit FIFO thresholds (12 words or 14 words), the SONIC will only fill the transmit FIFO only as much as needed to completely fill it (and not overfill it). Thus, if you choose a 12 word transmit FIFO threshold, the first bus tenure will transfer 12 words, but the second tenure will only transfer 4 words (12 words + 4 words = 32 bytes). This last example assumes that the bus latencies are zero.

For "empty/fill" mode, the number of transfers is also dependent on the bus latency. When the FIFO threshold has been reached, the SONIC will either completely empty the FIFO during reception or completely fill the FIFO during transmission. At the time of the bus request, the FIFO threshold equals the number of words in the FIFO, but due to bus latencies, additional bytes may have entered the FIFO (during reception). Thus, the number of words transfer during a bus tenure in this mode is the FIFO threshold plus the additional bytes that have entered the FIFO during reception or minus the bytes that have been serialized during transmission.

1.2 Packet Reception

This section gives a step-by-step description of the SONIC receiving a 68-byte packet. The initial conditions are shown below.

Initial conditions:

(a) The incoming packet is one that the SONIC will accept.

(b) The SONIC has detected that EOL = 1 from the previous reception, but the software has subsequently appended another receive descriptor before receiving this packet.

(c) The packet begins on a double word boundary.

(d) The Data Configuration register has been configured for:

- 32-bit data width mode (DB5 = DW = 1)
- 4 double word Receive FIFO threshold (DB3,2 = RF1,0 = 1,0)
- Exact Block Transfer mode (DB4 = BMS = 1)

(e) The packet has crossed over the End of Buffer Count (EOBC) register during this reception; hence the SONIC will need to use another Receive Buffer Area (RBA).

The reception is described as follows.

Note: The numbers in this section correspond to the numbers in *Figure 1-1.*

(1) Because of condition (a), the SONIC reads the RXpkt.link again to see if the software has subsequently reset the EOL bit to zero. Since it has (condition [b]), the SONIC writes to the RXpkt.in_use field and buffers the packet to the RBA. Note that this step is skipped if the SONIC has sufficient descriptors.

(2) Once the receive FIFO has reached its threshold (4 double words), the SONIC will write 4 double words during its bus tenure. For a 68-byte packet, the SONIC will perform this operation 4 times.

(3) During the last RBA bus tenure, the packet has ended (CRS goes low). The SONIC, again, requests for the bus once again (in 3 bus clocks) and flushes the remaining bytes in the FIFO (8 bytes). The first 4 bytes are the remainder of the packet and the last 4 bytes are the receive status that is automatically written into the FIFO by the SONIC. These last 4 bytes are extraneous to the RBA and are overwritten during the next reception. The usable receive status is written to the Receive Descriptor Area.

Note 1: If the packet size is not a multiple of the memory transfer size (16 bits or 32 bits), the SONIC will pad the last memory transfer with all 1's.

Note 2: If any of the last 4 bytes exceeds the length of the Receive Buffer Area, these bytes will not be written to memory.

(4) The SONIC writes the status information in the Receive Descriptor Area. The SONIC performs 7 consecutive memory transfers during its bus tenure (5 writes to the RXpkt.status, RXpkt.byte_count, RXpkt.pkt_ptr0, RXpkt.pkt_ptr1, and RXpkt.seq_no. fields, 1 read to the RXpkt.link field, and 1 write to the RXpkt.in_ use field). See *Figure 1-3* and Section 1.2.2 for further details.

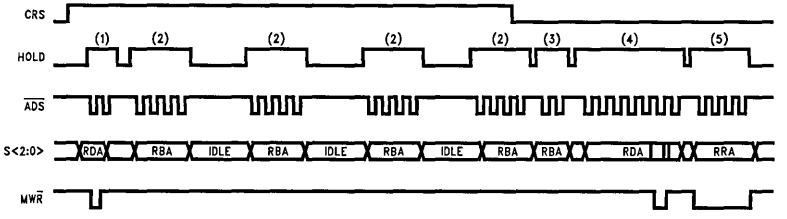
(5) Because of condition (e), the SONIC requests the bus again (in 3 bus clocks) and fetches a resource descriptor from the Receive Resource Area. The SONIC reads this area in 4 consecutive memory read operations.

1.2.1 Detail of Access to the RBA

Figure 1-2a and *1-2b* show the first SONIC access to the RBA for BMODE = 0 and 1, Note that the status pins S<2:0> change from 0,1,0 to 0,1,1 as the SONIC finishes writing the Source Address of the packet and continues buffering the rest of the packet.

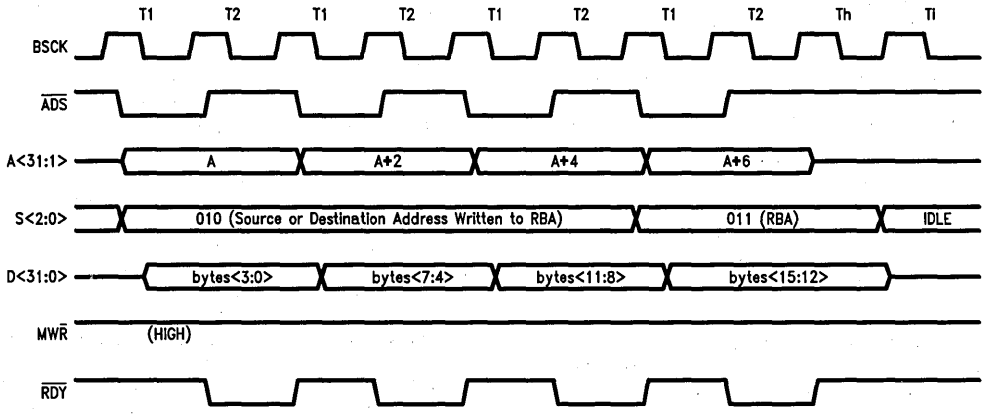
1.2.2 Detail of Access to the RDA

Figure 1-3a and *1-3b* shows the SONIC accessing the RDA for both BMODE = 0 and 1. Note that this block transfer contains both read and write accesses. Also note that the status pins S<2:0> briefly change from RDA (1,0,0) to Idle (1,1,1) between the read and write operations. This occurs between the RXpkt.seq_no and RXpkt.link accesses, and between the RXpkt.link and RXpkt.in_use accesses.



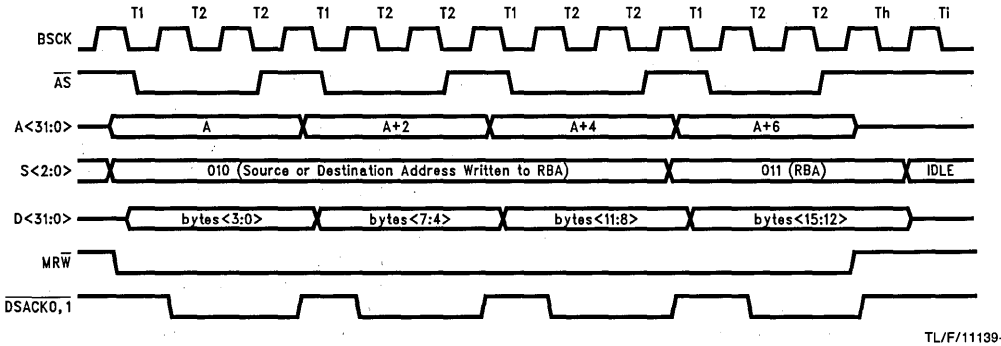
TL/F/11139–1

FIGURE 1-1. Complete Reception of a 68-Byte Packet



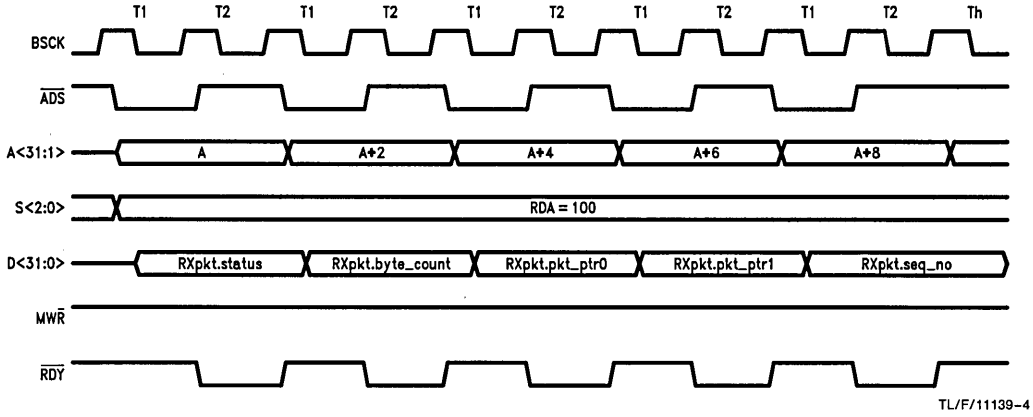
TL/F/11139-2

FIGURE 1-2a. First RBA Access for Storing Packet (BMODE = 0, Synchronous Mode)



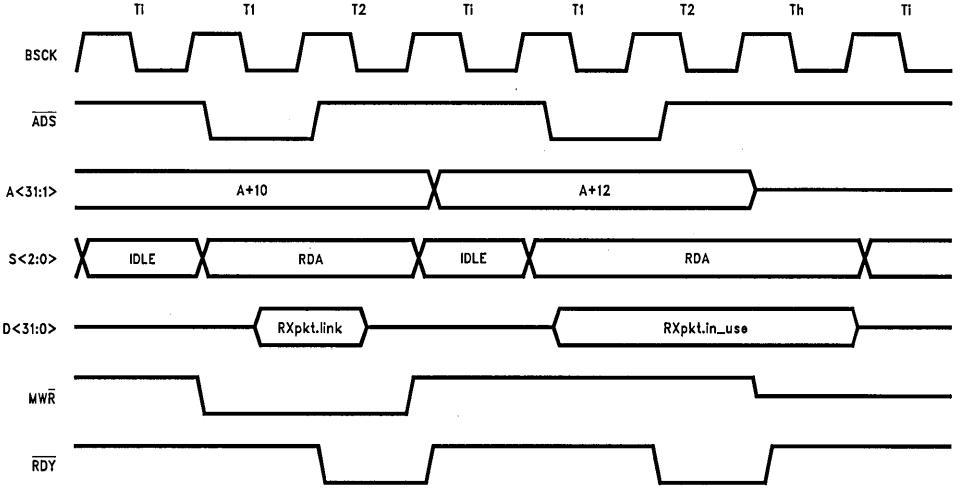
TL/F/11139-3

FIGURE 1-2b. First RBA Access for Storing Packet (BMODE = 1, Asynchronous Mode)



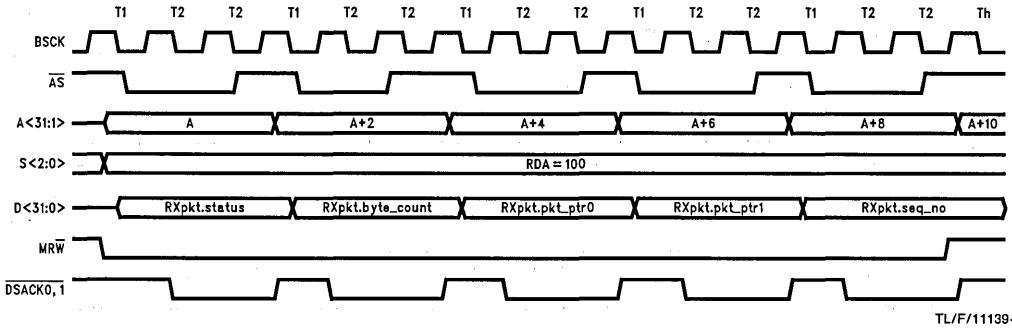
TL/F/11139–4

FIGURE 1-3a. RDA Access for Storing Descriptor Information (BMODE = 0, Synchronous Mode)



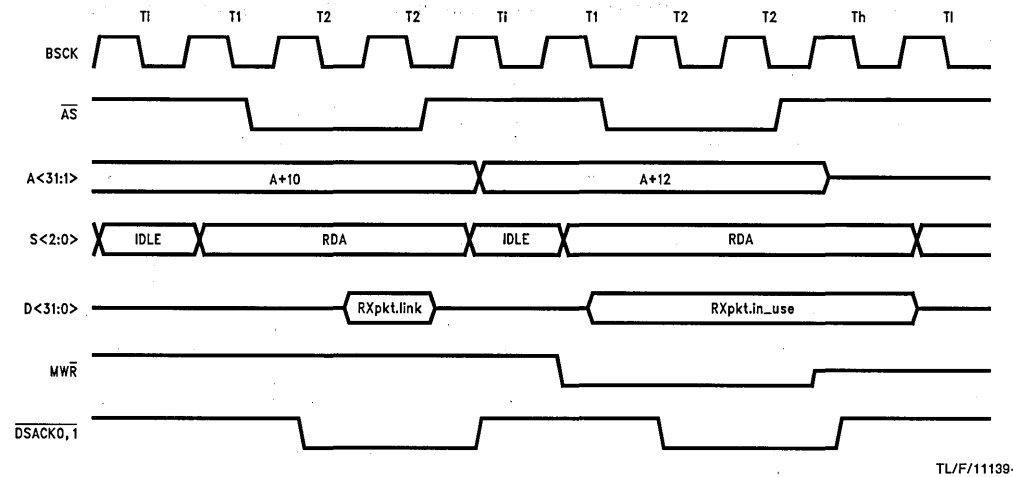
TL/F/11139–5

FIGURE 1-3a. RDA Access for Storing Descriptor Information (BMODE = 0) (Continued)



TL/F/11139-6

FIGURE 1-3b. RDA Access for Storing Descriptor Information (BMODE = 1, Asynchronous Mode)



TL/F/11139-7

FIGURE 1-3b. RDA Access for Storing Descriptor Information (BMODE = 1) (Continued)

1.3 Packet Transmission

This section gives a step-by-step analysis of a complete transmission using the initial conditions below.

Initial conditions:

(a) The Data Configuration register has been configured for:
- 32-bit data wide mode (DB5 = DW = 1)
- 16-byte Transmit FIFO threshold (BD1, 0 = TF1,0 = 0,1)
- Exact Block Transfer mode (DB4 = BMS = 1)

(b) The packet consists of 2 fragments. The first one is 48 bytes long and the last one is 20 bytes long.

(c) Both fragments are double-word aligned.

The transmit operation is described as follows: (the numbers in this section correspond to the numbers in *Figure 1-4*)

(1) Before the SONIC transmits, it fetches a descriptor from the Transmit Descriptor Area (TDA) to load its transmit registers. In 6 consecutive memory read operations, the SONIC reads the TXpkt.config, TXpkt.pkt_size, TXpkt.frag_count, TXpkt.frag_ptr0, TKpkt.frag_ptr1, and TXpkt.frag_size fields. Note that the TXpkt.status field is skipped during the first TDA access. Note also that if a collision occurs, forcing the SONIC to retransmit, the SONIC will once again fetch the descriptor from the beginning (i.e., starting at TXpkt.config).

(2) After fetching the descriptor, the SONIC begins loading the FIFO to its transmit threshold. The SONIC performs 4 consecutive memory operations in the Transmit Buffer Area (TBA) per bus tenure. Note that the fragment may begin on any byte boundary; if this is the case, the SONIC reads the corresponding double word which contains the beginning of the packet.

(3) The SONIC immediately requests for the bus again (in 3 bus clocks) because the number of words in the FIFO is equal to or less than the Transmit FIFO threshold. When the threshold has been exceeded, the SONIC commences transmission (TXE goes high). Subsequent requests for the TBA will not occur until the serializer has removed enough bytes from the FIFO to lower it below its threshold.

(4) Because of condition (b), the SONIC goes back to the Transmit Descriptor Area to obtain the pointer and length count of the next fragment. In three consecutive read operations, the SONIC will read the next TXpkt.frag_ptr0, TXpkt.frag_ptr1, and TXpkt.frag_size fields.

(5) At the end of transmission, the SONIC will write the status of the TXpkt.status field, then read the TXpkt.link field to locate the next descriptor.

1.3.1 Detail of Access to the TDA

Figure 1-5a and *1-5b* shows the SONIC accessing the TDA at the end of transmission. The SONIC writes the status information at the beginning of the descriptor and reads the link field at the end of descriptor. (Note n = the number of fragments.)

1.3.2 Detail of Access to the TBA

Figure 1-6a and *1-6b* shows the SONIC accessing the TBA when the transmit FIFO has been programmed for (1) 32-bit mode (2) exact block transfer mode, and (3) a 4 double word threshold.

1.4 Loading the CAM (Content Addressable Memory)

After the CAM descriptor Area has been initialized and the Load CAM command issued to the SONIC, the SONIC will read the CAM Descriptor Area (CDA) and load its CAM. The SONIC, in 4 memory read cycles, accesses memory and loads one CAM entry per bus tenure. During the last block transfer, the SONIC reads one additional word to load its CAM Enable register. In the example illustrated in *Figure 1-7*, the SONIC has been programmed to load 4 CAM locations.

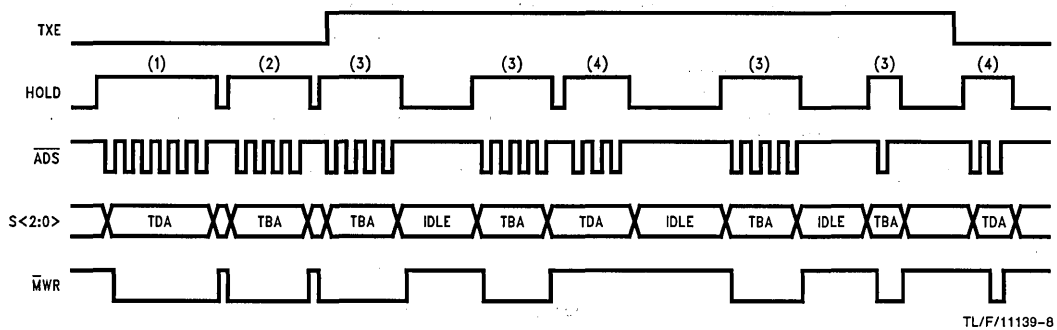


FIGURE 1-4. Complete Transmission of a 68-Byte Packet

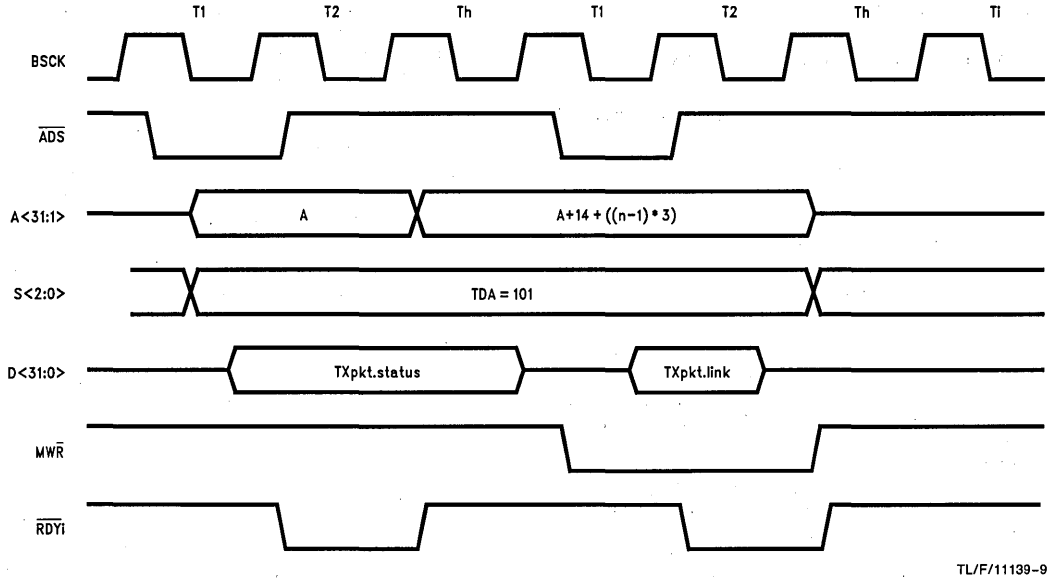


FIGURE 1-5a. Last TDA Access (BMODE = 0, Synchronous Mode)

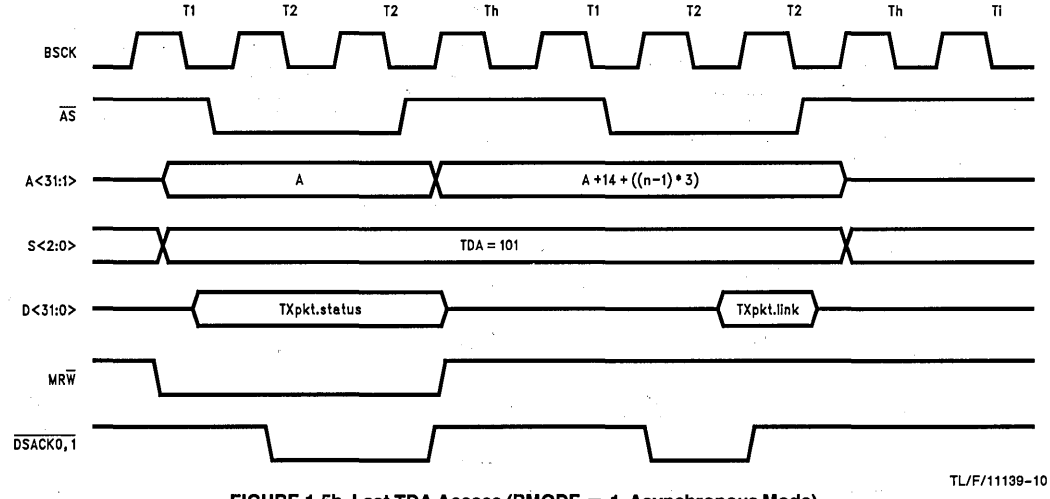


FIGURE 1-5b. Last TDA Access (BMODE = 1, Asynchronous Mode)

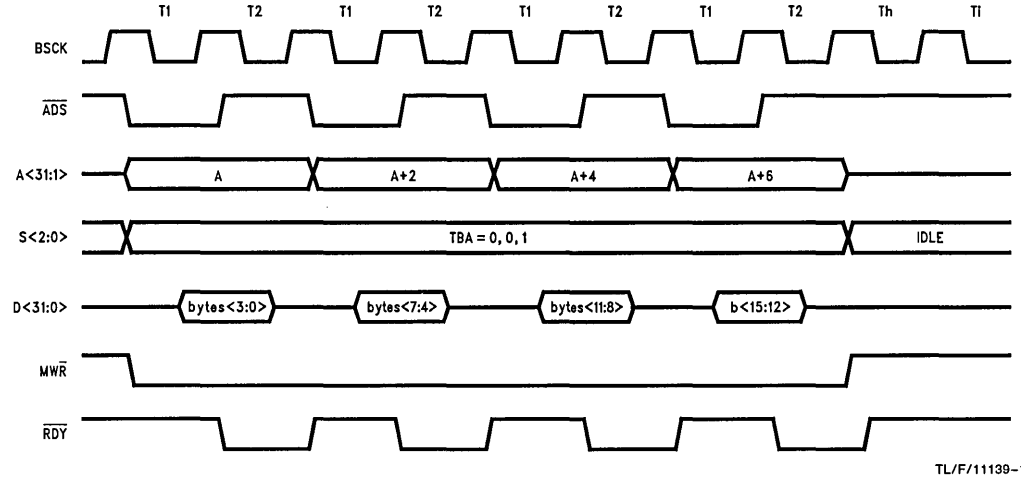


FIGURE 1-6a. Typical TBA Access (BMODE = 0, Synchronous Mode)

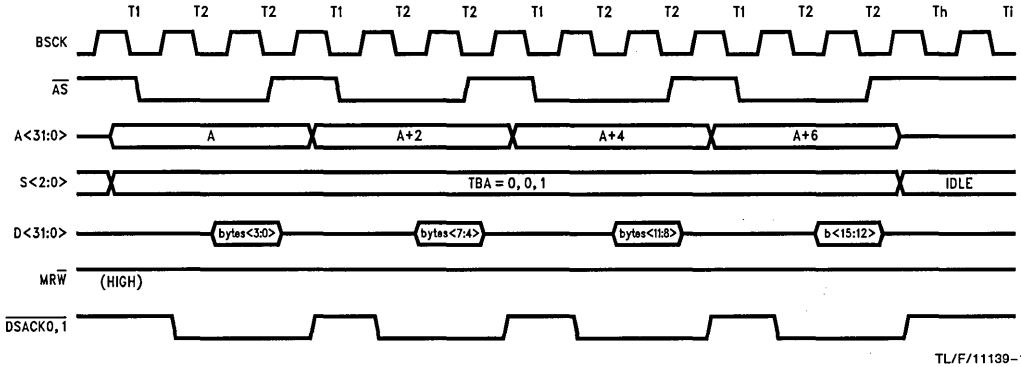


FIGURE 1-6b. Typical TBA Access (BMODE = 1, Asynchronous Mode)

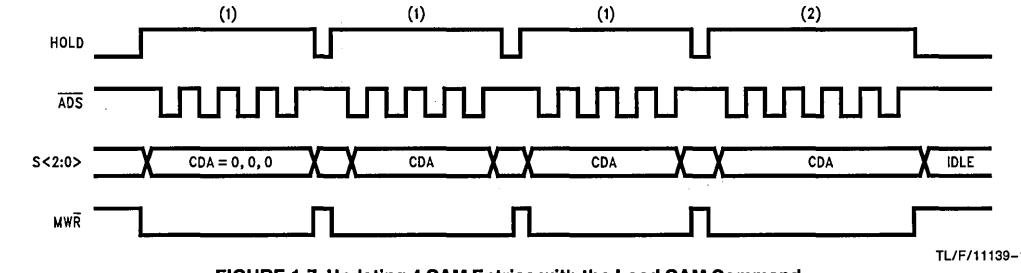


FIGURE 1-7. Updating 4 CAM Entries with the Load CAM Command

2.0 SLAVE OPERATIONS

The slave operations of the SONIC can be classified into two cases, (1) registers access while the SONIC is idle, and (2) register accesses while the SONIC is not idle. The first case always occurs in a single bus systems where the CPU and SONIC reside on the same bus. Only one bus master is allowed on the bus in such a system. The second case may occur in dual bus systems where the CPU and SONIC lie on different busses. In this case, the CPU may access the SONIC while it is currently using the bus (such as during transmission or reception). The SONIC does not respond immediately in this case, but finishes off its current bus master operation before responding to the register access. The following two sections give a step-by-step description of the slave operations.

2.1 Register Access During Idle

(Refer to *Figures 2-1a* and *2-1b*)

(1) The CPU presents the register address, address strobe, chip select, and slave direction strobe to the SONIC.

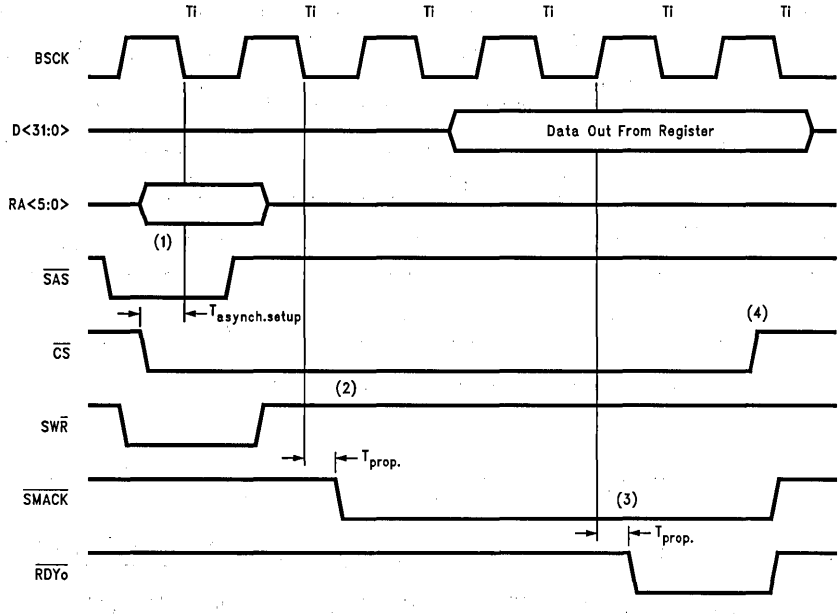
Note: For BMODE = 0, $\overline{SAS}$ must be asserted low before or at the same time that $\overline{CS}$ is asserted. The rising edge of $\overline{SAS}$ latches the register address, RA<5:0> and slave direction strobe, $SW\overline{R}$.

(2) The SONIC synchronizes chip select to the falling edge of bus clock and responds with slave and memory acknowledge ($\overline{SMACK}$) at the next falling edge of bus clock.

Note: BMODE = 0, if $\overline{SAS}$ remains asserted (low) beyond the falling edge of $\overline{CS}$, $\overline{SMACK}$ will not be assert until $\overline{SAS}$ is deasserted high.

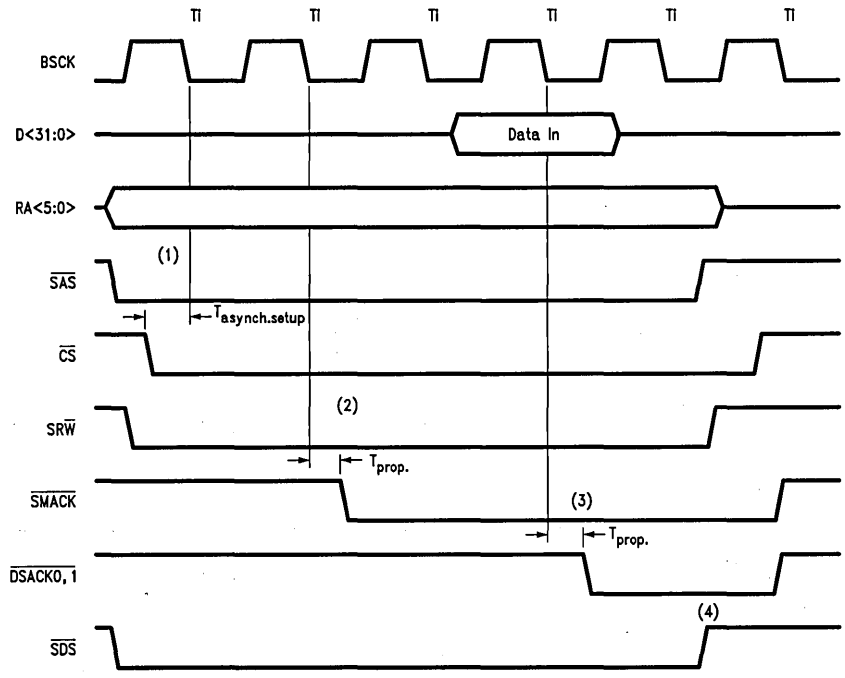
(3) For BMODE = 1, $\overline{DSACK0,1}$ is generated 2 bus clocks after $\overline{SMACK}$ is asserted, and for BMODE = 0, $\overline{RDYo}$ is generated 2.5 bus clocks after $\overline{SMACK}$. These outputs indicate to the CPU that the slave access is over. For read cycles, register data is valid at the falling edge of the ready signal ($\overline{RDYo}$ or $\overline{DSACK0,1}$); for write cycles, the SONIC has latched the register data.

(4) The CPU completes the slave cycle by deasserting $\overline{CS}$ or reasserting $\overline{SAS}$ low for BMODE = 0, or deasserting $\overline{CS}$ or $\overline{SAS}$ for BMODE = 1. The earliest of these signals will terminate the slave cycle.



TL/F/11139–14

FIGURE 2-1a. Register Read While SONIC is Idle (BMODE = 0)



TL/F/11139–15

FIGURE 2-1b. Register Read While SONIC is Idle (BMODE = 1)

2.2 Register Access While SONIC is a Bus Master

(Refer to *Figures 2-2a* and *2-2b*)

(1) During bus master operations, the SONIC samples chip select, $\overline{CS}$, at the falling edge of T1. If it samples $\overline{CS}$ to be active, the SONIC finishes the current bus cycle, then TRI-STATES off the bus.

(2) During falling edge the first Ts state (slave), the SONIC asserts slave and memory acknowledge ($\overline{SMACK}$).

(3) For BMODE = 1, $\overline{DSACK0,1}$ is generated 2 bus clocks after $\overline{SMACK}$ is asserted, and for BMODE = 0, $\overline{RDY0}$ is generated 2.5 bus clocks after $\overline{SMACK}$. These outputs indicate to the CPU that the slave access is over. For read cycles, register data is valid at the falling edge of the ready signal ($\overline{RDY0}$ or $\overline{DSACK0,1}$); for write cycles, the SONIC has latched the register data at the falling edge of the ready signal.

(4) The CPU terminates the slave cycle by the deasserting chip select. The SONIC samples the rising edge of $\overline{CS}$ at the rising edge of bus clock.

(5) After the SONIC samples chip select to be inactive, the SONIC continues its bus master operations from where it left off.

2.3 Asserting $\overline{MREQ}$ to Access Shared Memory

Asserting $\overline{MREQ}$ to the SONIC has nearly the same effect as asserting $\overline{CS}$. $\overline{SMACK}$ is generated identically as before, but the ready signal ($\overline{RDY0}$ or $\overline{DSACK0,1}$) is not asserted. The ready signal must be asserted by the memory control logic.

Note: Both $\overline{MREQ}$ and $\overline{CS}$ must not be asserted simultaneously. This will cause spurious accesses to the SONIC's registers. Note also that the SONIC requires a recovery time of 1.5 bus clocks between the deassertion edge of one signal to the assertion edge of the other.

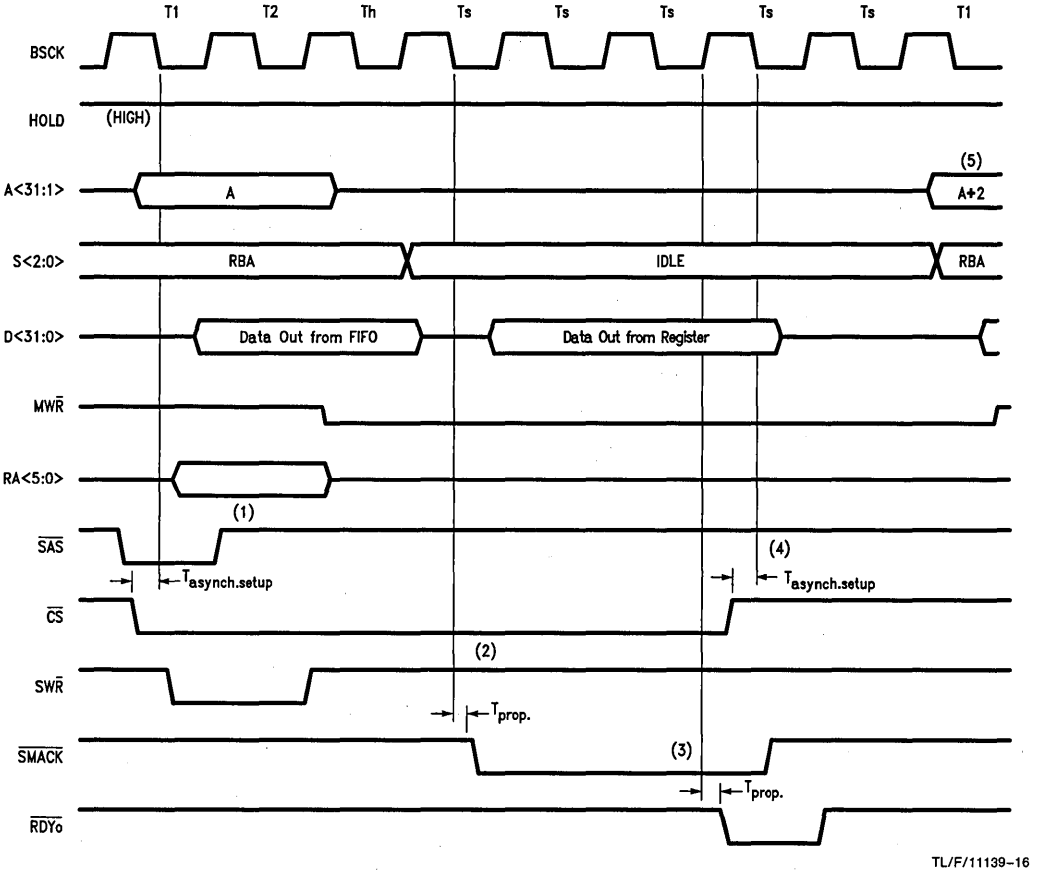
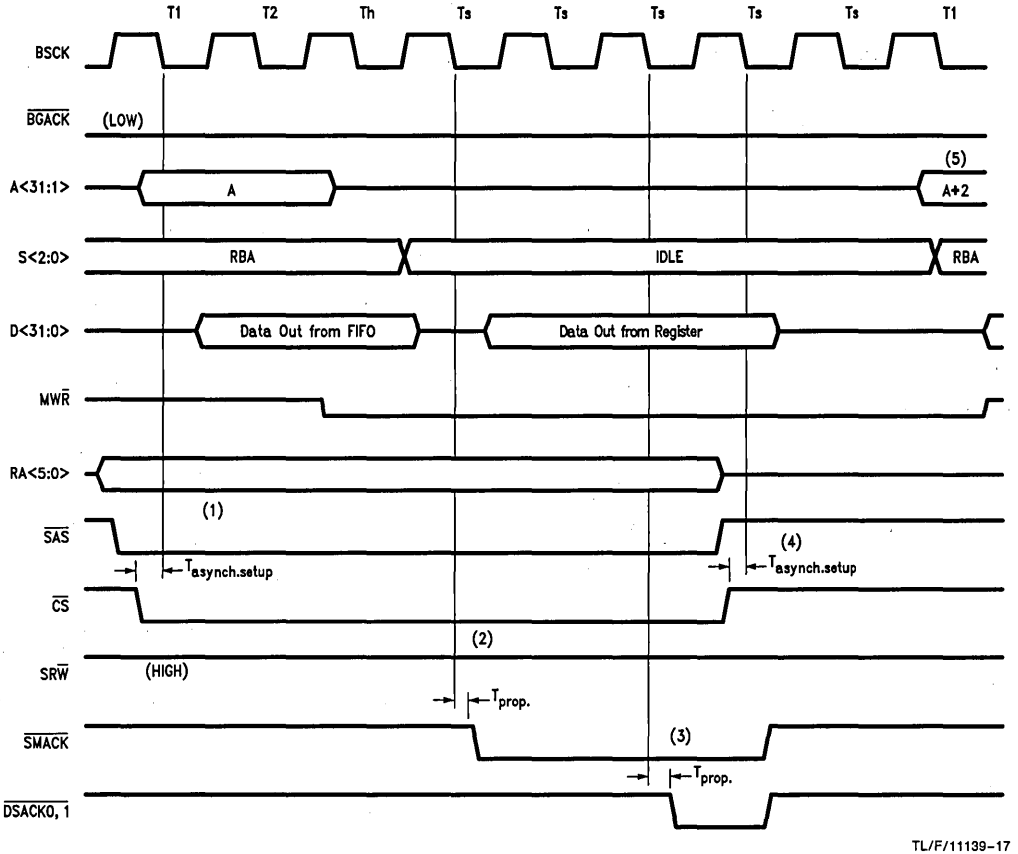


FIGURE 2-2a. Register Read While SONIC is Currently Using the Bus (BMODE = 0)



TL/F/11139–17

FIGURE 2-2b. Register Write While SONIC Is Currently Using the Bus (BMODE = 1)

Software Driver Programmer's Guide for the DP83932 SONIC™

National Semiconductor
Application Note 746
Wesley Lee
Mike Lui

INTRODUCTION

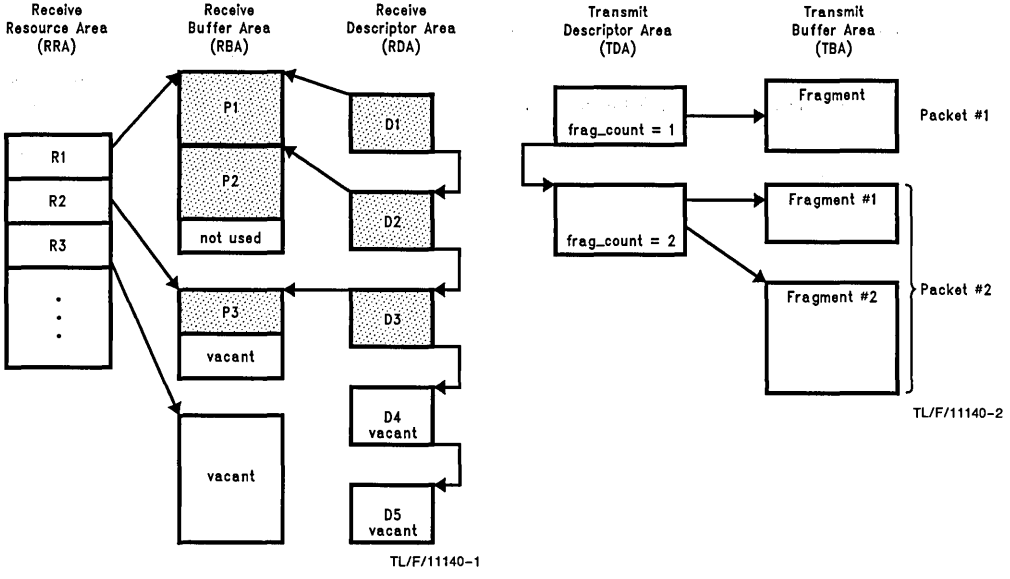
In the past, Ethernet chips have concentrated on interfacing well with the hardware, but have given the software interface only passing notice. While hardware designers may have been satisfied, the software developers were forced to write drivers for unwieldy silicon. Recently, with companies looking for ways to increase performance, they have found that the software interface is crucial and has been one of the bottlenecks in the system. A chip with an over constraining buffer management slows down the system by introducing more levels of indirection (pointers) than are truly needed by the system software. In view of these shortcomings, National surveyed a number of software developers to define a buffer management system which operates efficiently with the driver. Their basic response was *Keep it Simple.* The reasons were twofold. First, a simple software interface engenders a driver which is easy to write and secondly, a simpler, thus shorter, driver leads to a faster driver. The SONIC's buffer management epitomizes this with three salient features. First, only one level of indirection is used to reference data in memory; secondly, link-lists are chosen to endow the software developer with the flexibility to easily manipulate descriptors, and thirdly, a register-based command interface is provided to make commands fast and immediate.

ABOUT THIS GUIDE

This guide will provide you the information needed to write a driver for the DP83932 System-Oriented Network Interface Controller (SONIC). You will first be introduced to basic algorithms using the SONIC's buffer management, then be shown actual implementation examples. It is recommended that you are familiar with the DP83932 SONIC datasheet before reading this document.

1.0 THE DRIVER SOFTWARE—SONIC INTERACTION

The key to making a Driver and all upper levels of the network software efficient, is to ensure that they must be capable of referencing received or transmitted packets via pointers and then conveying these pointers up to the next level of software. By employing pointers in this manner, needless packet copying from one area in memory to another is eliminated. As shown in *Figure 1-1*, the SONIC's descriptor areas, the RDA and TDA reference the received and transmitted packet and the RRA references the buffers for the received packets. The actual received and transmitted packets remain in their original locations in the RBA and TBA and are not copied elsewhere. In this section the basic algorithms are given to illustrate the usage of the RDA, RRA and TDA. Section 4.0 describes the implementation examples.



R = resource descriptor
D = packet descriptor
P = packet

FIGURE 1-1. Overview of the SONIC's Buffer Management

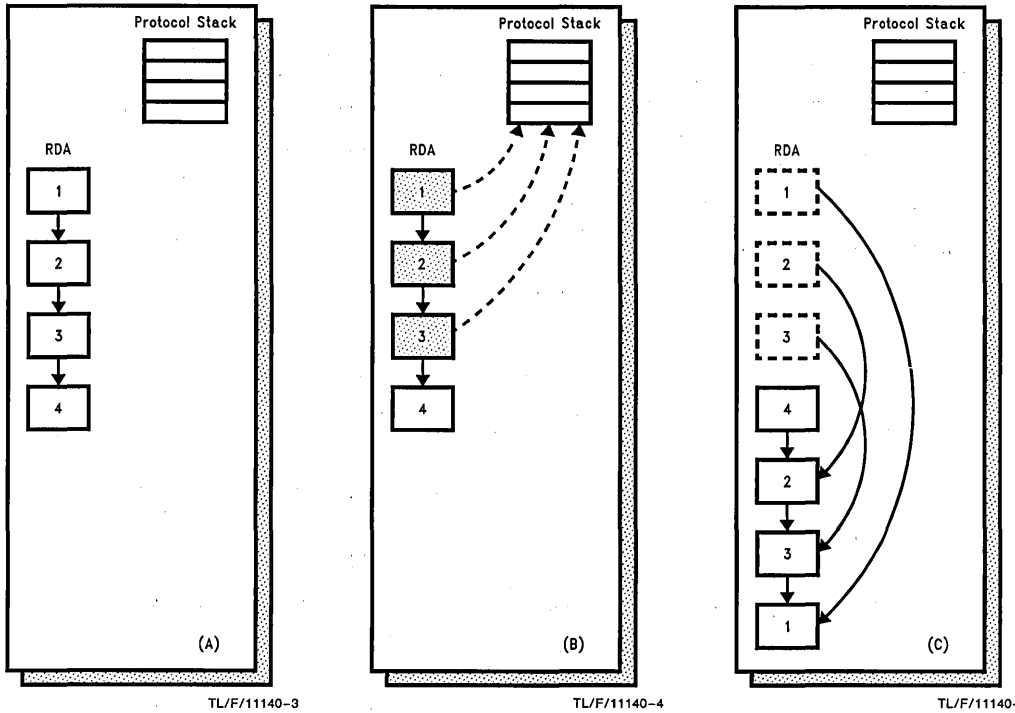
1.1 Processing Packets in the Receive Descriptor Area (RDA)

After the SONIC has received the packet, it places the packet in the RBA and the packet information in the RDA. The Driver, in turn, processes this packet by locating the packet from the packet pointer (RXpkt.ptr0,1) fields in the RDA and then delivering the pointer up to the next level software for further processing. The Driver then returns the descriptor to the front of the list for reuse. This process is illustrated in *Figures 1-2 (a), (b),* and *(c).* Note that the link-list allows descriptors to be appended to the front of the list in any order. Note also that no special considerations are required to append receive descriptors. The pseudo code below illustrates the simplicity in appending descriptors.

```
append_descr( )
      new_RXpkt.link = 1;
      old_RXpkt.link = new_RXpkt.status
      /* Old link field points to
      address of new status field */
```

1.2 Recycling Buffers in the Receiver Resource Area (RRA)

Intermixed with processing of packets, the Driver must also replenish the receive buffer pool by adding resources descriptors to the RRA. A suggested method for replenishing receive buffers is given in the following example. (This method assumes that more than one packet is stored in an RBA.)



TL/F/11140–3 TL/F/11140–4 TL/F/11140–5

FIGURE 1-2 a, b, c. Processing Descriptors in the RDA

(a) Initial condition: four descriptors are available for use
(b) Packets received: three packets having been received are then passed up to the upper level software for further processing.
(c) Packets processed: the upper level software having finished processing the packets, the Driver returns the descriptors to the front of the list.

The Driver allocates a fixed number of receive buffers (RBAs), determined at initialization, and recycles them as they are used. When the upper level software receives a packet from the driver (via pointers), it processes the packet at the location received (in the RBA), and when done, notifies the Driver of the freed memory space. The Driver, then, records this event by tallying the packet in a "scoreboard" corresponding to the RBA (see *Figure 1-3*). When the number of packets processed equals the total number of packets in an RBA, then RBA is free and may be returned to the RRA ring.

RBA#	Packets Processed	Total Packets in RBA
0	5	5
1	2	4
2	3	6
3	3	Unknown

FIGURE 1-3: RBA Scoreboard Example

RBA#0 is now free since packets processed equals total packets in RBA. For RBA #3, the software does not yet know how many packets reside in an RBA since the SONIC has not finished using this RBA. When the software detects the LPKT bit set, the packet sequence number reveals the total number of packets (see below).

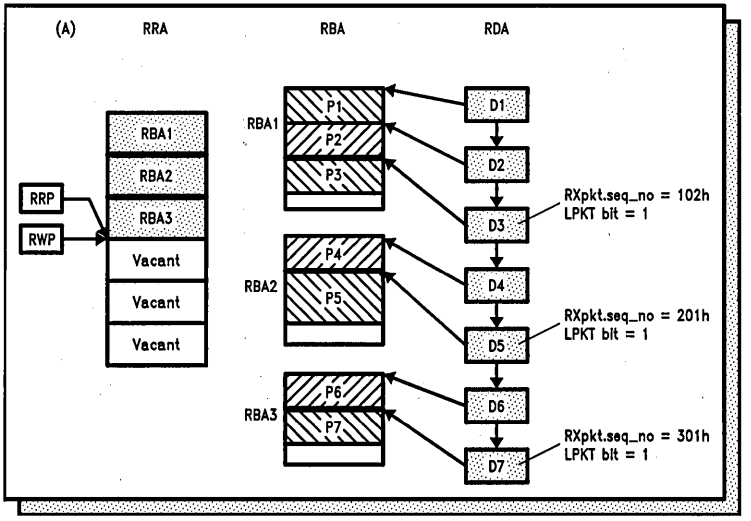
Because packets may be processed in any order (thus, packets may be freed up out of sequence), freeing up an RBA is not a straight forward. However, the SONIC reduces this task to a simple tallying procedure with its Receive Sequence Numbers (RXpkt.seq_no). When the Driver detects the LPKT (last packet) bit set to a 1, the sequence numbers indicate how many packets are in a given RBA. Thus, the Driver simply tallies the number of packets processed for a given RBA and when this is equal to the total number of packets, the RBA is free. The sequence numbers are shown below.

15 — 8	7 — 0
RBA Sequence Number (Modulo 256)	Packet Sequence Number (Modulo 256)

If LPKT = 1

packet sequence number equals total number of packets minus one in the RBA (packet sequence number starts at zero)

The following three figures (*Figures 1-4a*, *1-4b*, and *1-4c*) show a scenario depicting the Driver using 3 RBAs and updating the RBA "scoreboard". The flowchart in section 4.2 (*Figure 4-3*) illustrates the recycling of RBAs during receive processing.



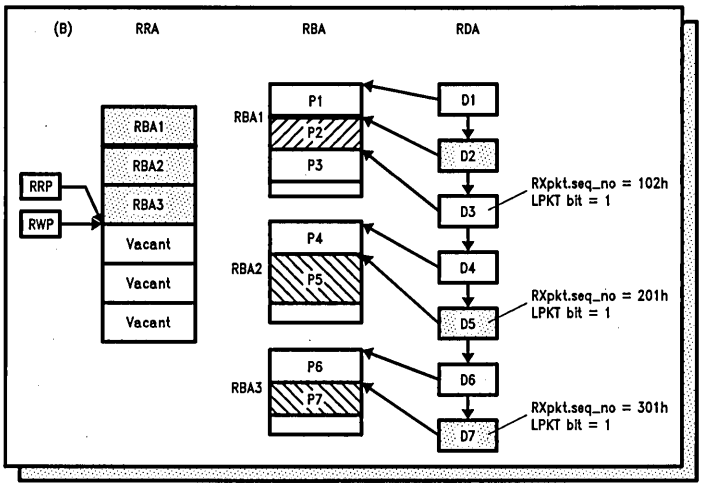
TL/F/11140–6

RBA Scoreboard

RBA#	Processed Packets	Total Packets
1	0	3
2	0	2
3	0	2

FIGURE 1-4 (a). Recycling Buffers in the RRA

(a) This figure shows the SONIC, having stored seven packets (P1–P7) in the RBA, has exhausted all its receive buffers (RRP = RWP). The RBA scoreboard indicates that there are 3 unprocessed packets in RBA #1, 2 in RBA #2 and 2 in RBA #3. These numbers are determined by the RXpkt.seq_no field.



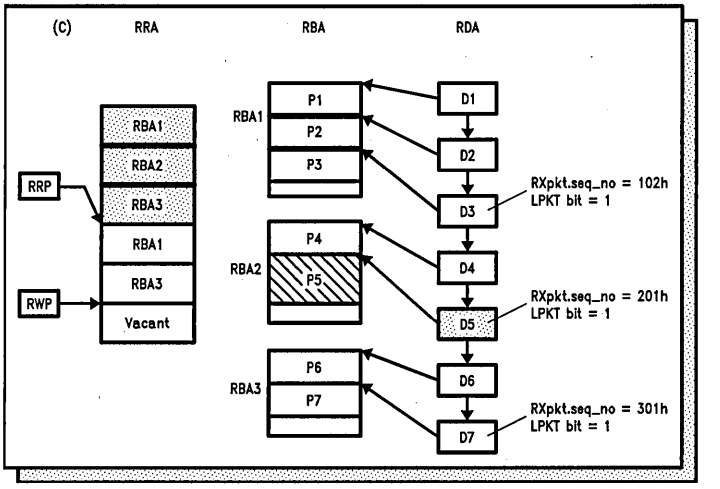
TL/F/11140–7

RBA Scoreboard

RBA#	Processed Packets	Total Packets
1	2	3
2	1	2
3	1	2

FIGURE 1-4(b). Recycling Buffers in the RRA

(b) The upper level software has finished processing four packets (P1, P3, P4, and P6) and has notified the Driver of this action. The RBA scoreboard now indicates that there is 1 unprocessed packet in RBA #1, 1 in RBA #2 and 1 in RBA #3.



TL/F/11140–8

RBA Scoreboard

RBA#	Processed Packets	Total Packets	
1	3	3	→ RBA 1 may be recycled
2	1	2	
3	2	2	→ RBA 3 may be recycled

FIGURE 1-4 (c). Recycling Buffers in the RRA

(c) The upper level has now finished processing 6 packets (P1, P2, P3, P4, P6, and P7). The RBA scoreboard now indicates that RBA #1 and RBA #3 are freed up. The Driver returns these buffers back to the RRA and increments the RWP register accordingly.

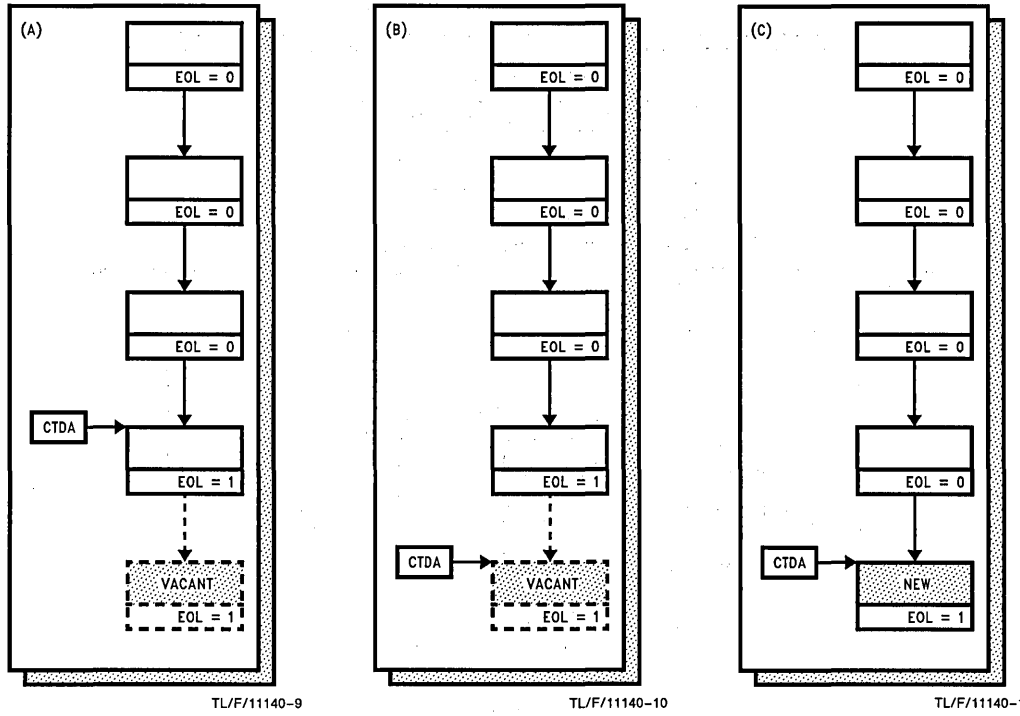
1.3 Transmitting Packets from the Transmit Descriptor Area (TDA)

For transmit operation, the Driver uses the TDA to enqueue packets for transmission. Multiple packets may be sent from a single command with each packet allowed to be fragmented (reside in different areas in memory). The fragments themselves may be as small as 1 byte and begin on any byte boundary. Furthermore, particular attention has been made to allow the Driver to append descriptors "on the fly".

To send packets, the driver first creates a list of descriptors in the TDA, then issues the transmit command. The SONIC then reads the TDA and transmits the packets. Once a list is created, the Driver can add to this list "on the fly" without the SONIC stopping. The following rule, however, must be followed: *the last TXpkt.link field must point to the next location where a descriptor will be added* as illustrated in *Figure 1-5 (a)*. The procedure for appending descriptors is outlined as follows:

1. Create a new descriptor with its TXpkt.link pointing to the next vacant descriptor location and its EOL bit set to a "1".
2. Reset the EOL bit to a "0" of the previously last descriptor.
3. Re-issue the Transmit command (setting the TXP bit in the Command register).

Re-issuing the Transmit command assures that the SONIC will continue to send all packets in the list. If the SONIC is currently transmitting, the Transmit command has no effect. If the SONIC has stopped transmitting (which occurs if the SONIC has reached the last descriptor before the Driver has had a chance to append to it) it continues transmitting from where it had previously stopped. The rule, as stated above, guarantees that the Current Transmit Descriptor (CTDA) register points to a valid descriptor after the SONIC has stopped transmitting (see *Figures 1-5 (a), (b)* and *(c)*).



TL/F/11140–9 TL/F/11140–10 TL/F/11140–11

FIGURE 1-5 a, b, c. Appending Descriptors "On the Fly" in the TDA

These series of figures shows a scenario whereby the SONIC has reached the end of the descriptor list before the Driver has appended a new descriptor.

(a)This figure shows the Driver has created a list of four descriptors with the last descriptor pointing to the next location where a descriptor will be added. The transmit command has subsequently been issued and the SONIC has reached the last descriptor.

(b) The SONIC has finished transmitting the last descriptor. It reads the last link field and updates the CTDA register to point to the vacant descriptor location. Note that the CTDA register is already prepared for the next transmission.

(c) The Driver has appended a descriptor at the vacant location and reissues the transmit command. Note that the CTDA register is pointing to the proper location.

2.0 REGISTER MODEL OF THE SONIC

As a brief review, this section gives a short description of the SONIC User registers. This section is similar to section 4.0 of the SONIC datasheet. It may be skipped without loss of continuity.

2.1 Register Layout

The SONIC contains 64 16-bit registers used for conveying status and control information. Not all registers, however, are needed by the system since some registers are used for internal operations of the SONIC and others used for in-house factory testing. The registers are categorized as follows:

User Registers: The registers are accessed by the user to status, control and monitor SONIC operations. These are the only registers you need to access.

Internal Use Registers: These registers are used by the SONIC during the course of operation and are not intended to be accessed by you.

Factory Test Registers: These registers are used by National Semiconductor for production testing of the SONIC and should not be accessed. Accessing these registers during SONIC operations may cause erratic behavior.

	RA<5:0>		15 ... 0
Status and Control Registers	0h	Command Register	Status and Control Fields
	1	Data Configuration Register	Status and Control Fields
	2	Receive Control Register	Status and Control Fields
	3	Transmit Control Register	Status and Control Fields
	4	Interrupt Mask Register	Mask Fields
	5	Interrupt Status Register	Status Fields
Transmit Registers	6	Upper Transmit Descriptor Address Register	Upper 16-Bit Address Base
	7	Current Transmit Address Register	Lower 16-Bit Address Offset
	2F	Maximum Deferral Timer	Count Value
Receive Registers	0D	Upper Receive Descriptor Address Register	Upper 16-Bit Address Base
	0E	Current Receive Address Register	Lower 16-Bit Address Offset
	14	Upper Receive Resource Address Register	Lower 16-Bit Address Offset
	15	Resource Start Address Register	Lower 16-Bit Address Offset
	16	Resource End Address Register	Lower 16-Bit Address Offset
	17	Resource Read Register	Lower 16-Bit Address Offset
	18	Resource Write Register	Lower 16-Bit Address Offset
	2B	Receive Sequence Counter	Count Value (8) | (7) Count Value
CAM Registers	21	CAM Entry Pointer	(4) Pointer
	22	CAM Address Port 2	Most Signif. 16 Bits of CAM Entry
	23	CAM Address Port 1	Middle 16 Bits of CAM Entry
	24	CAM Address Port 0	Least Signif. 16 Bits of CAM Entry
	25	CAM Enable Register	Mask Fields
	26	CAM Descriptor Pointer	Lower 16-Bit Address Offset
	27	CAM Descriptor Count	(5) Count Value
Tally Counters	2C	CRC Error Tally Counter	Count Value
	2D	Frame Alignment Error Tally	Count Value
	2E	Missed Packet Tally	Count Value
Watchdog Timer	29	Watchdog Timer 0	Lower 16-Bit Count Value
	2A	Watchdog Timer 1	Upper 16-Bit Count Value
	28	Silicon Revision Register	Chip Revision Number

FIGURE 2-1. User Register Grouping

2.2 User Register Grouping

The User register may be further categorized into 6 groups (*Figure 2-1*) based upon their functionality, i.e., Status and Control, Transmit, Receive, Content Addressable Memory (CAM), Tally counters, and General-Purpose timer. These groups are described as follows:

2.2.1 Status and Control Registers

These registers, controlling the transmit, receive, bus, and interrupt operations of the SONIC, consist of the Command, Data Configuration, Receive Control, Transmit Control, Interrupt Mask, and Interrupt Status registers. Of these registers, only the Command and Interrupt Status register are accessed frequently during operation; all others are generally accessed only once during initialization (see section 3.0). These registers are briefly described below.

Command register. This register is used for issuing the commands to the SONIC such as transmitting packets, enabling the receiver, and software reset. Commands may be issued by setting the corresponding bit to a "1". During normal operation, the transmit command is the only command that is generally used.

Data Configuration register: This register configures the bus interface circuitry, programming the data width size (16 or 32 bits), wait-state insertion (if any), and FIFO threshold. This register may only be written to while the SONIC is in software reset.

Receive Control register. This register contains two type of bits, configuration and status. The configuration bits program the SONIC to accept the different classes of packets which may be received such as Physical, Multicast, Broadcast packets, and Runt and Errored packets. The SONIC can also accept all packets from the network for network management and Bridge applications. The Receive Control register also reports the status of the received packet. The software should not read this register directly since status is updated from the next incoming packet and the previous status is overwritten. Instead, the software obtains the status in the status field (RXpkt.status) of the Receive Descriptor Area.

Transmit Control register. This register also contains two types of bits, configuration and status. The configuration bits program the various transmit options for (1) generating and interrupts after selected packets have been transmitted, (2) enabling when the "Out of Window" collision timer begins (either at the beginning of the packet or at the State of Frame Delimiter), (3) inhibiting the CRC from being appended to the packet, and (4) enabling the excessive deferral timer (3.2 μs). The software should not load this register directly; instead, it writes to the configuration field (TXpkt.config) of the Transmit Descriptor Area (TDA) which the SONIC reads before transmission. The status bits post status of the transmitted packet. Again, this register is not directly read since the SONIC clears the status after it reads the TXpkt.link field. Instead, the software acquires status from the state field (TXpkt.status) in the TDA.

Interrupt Mask register. This register enables the various interrupts that the SONIC may generate. Writing a "1" to the bit enables the corresponding interrupt.

Interrupt Status register. This register reports interrupts which the SONIC has generated. Interrupts are indicated by a "1" and are cleared when a "1" has been written to it. Since writing a "0" to any bit has no effect, only the specified bits are cleared during the write operation.

2.2.2 Transmit Register

The Transmit registers, the Upper Transmit Descriptor Address (UTDA) and the Current Transmit Descriptor Address (CTDA) registers, locate the active descriptor in the Transmit Descriptor Area. The UTDA register, containing a fixed upper 16 bits of address, A<31:16> and CTDA register, containing an active lower 15 bits of address, A<15:1> are concatenated together to form a complete 31-bit address. (The SONIC only provides word or double word addressing.) The LSB of the CTDA register is the End of List (EOL) bit and is used by the SONIC to determine the last descriptor in the list.

2.2.3 Receive Registers

The receive registers consist of the Receive Sequence Counter, the End of Buffer Count (EOBC) register, and two groups of registers, the descriptor registers and the resource registers. These registers are briefly described as follows:

The Receive Sequence Counter: This counter indicates the number of packets that reside in a particular Receive Buffer Area (RBA). See section 1.1 for an explanation on how to use this register.

EOBC register. This register defines the lower boundary in the RBA. If after reception, the remaining numbers words in the RBA are equal to or greater than the EOBC register, reception continues within the same RBA; otherwise, the SONIC stores the packet in another RBA.

Descriptor registers: These registers locate the active descriptor in the Receive Descriptor Area (RDA) and are composed of the Upper Receive Descriptor (URDA) and the Current Receive Descriptor (CRDA) registers. These registers are concatenated similarly as the Transmit registers (UTDA and CTDA) above where the URDA contains a fixed upper 16 address bits, A<31:16> and the CRDA contains the lower 15 address bits, A<15:1>. The LSB of the CRDA register is used by the SONIC to determine the last descriptor in the receive list.

Resource registers: These registers, used to define the Receive Resource Area (RRA), composed of the Resource Start Area (RSA), the Resource End Area (REA), Resource Write Pointer (RWP), Resource Read Pointer (RRP) and the Upper Receive Resource Address (URRA) registers. The first two registers are static and define the starting and ending points of the RRA. The second two are active and respectively point to the next location where the software places a new descriptor and where the SONIC reads the next descriptor. The SONIC concatenates the last register, the URRA with the other registers to provide a full 31-bit address. The URRA register contains a fixed upper address, A<31:16> and the other four contain an active lower address, A<15:1>. The LSB of these registers is not used since the SONIC only provide word or double word addressability.

2.2.4 CAM Registers

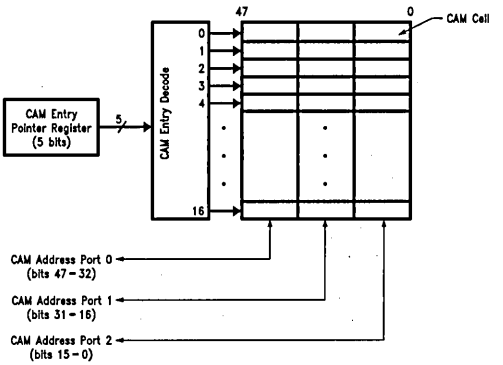
The CAM registers are used to access the 16 48-bit CAM entries. Because random accessibility to all CAM entries would consume too much register space (16 x 3 = 48 locations), the CAM entries are accessed via a 4-bit pointer register (CAM Entry Pointer) and 3 16-bit ports (CAM Access Ports 0 to 2). The CAM Entry Pointer selects 1 of 16 entries and the CAM Access Ports 0 to 2, respectively access the least through the most significant portions of the 48-bit entry *(Figure 2-2)*.

Note: The least significant byte of the address is the *first* byte received/transmitted from the network.

Reading the CAM

The CAM is accessed in the following manner:

1) Place the SONIC in software reset by setting the RST bit in the Command register. This condition must be met before reading the CAM.
2) Select the CAM entry by writing the corresponding value in the CAM Entry Pointer.
3) Read the CAM Address Ports 0 to 2 to obtain the complete 48-bit entry.



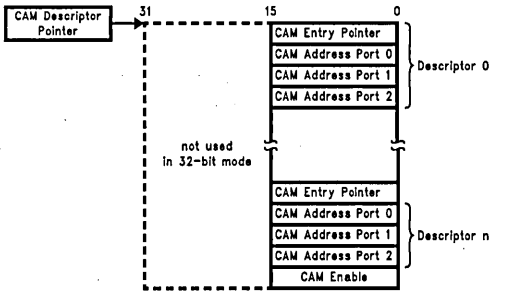
TL/F/11140–12

FIGURE 2-2. CAM Organization

Writing to the CAM

To avoid internal conflicts with the CAM entries when receiving packets, the SONIC does not allow the entries to be written to directly. Instead, the entries are written to indirectly via the CAM Descriptor Area (CDA). This area, maintained in memory, contains the data to be written into the CAM and upon command, the SONIC reads this area and load its CAM. The CDA is composed of *n* number descriptors (*Figure 2-3*) which are used to load the CAM Entry Pointer, the CAM Access Ports, and the CAM Enable register. To program the CAM, you first initialize the CDA, load the CAM Descriptor Count register with the number of descriptors and the CAM Descriptor Pointer register with the starting address of the CDA, then issue the Load CAM command to the SONIC. This operation is summarized below:

1) Load the CDA as specified in *Figure 2-3*.
2) Load the CAM Descriptor Count register with the number of descriptors.
3) Load the CAM Descriptor Pointer register with the starting address of the CDA.
4) Issue the Load CAM command (setting the LCAM bit in the Command register). The SONIC finishes this command when the LCAM bit is reset.



TL/F/11140–13

FIGURE 2-3. CAM Descriptor Area Format

2.2.5 Tally Counters

The Tally counters maintain the network management events which occur too frequently for the software to maintain. These events, CRC errors, frame alignment errors, and missed packets are tallied by the CRC, FAE and Missed Packets Tally counters. these counters are 16-bit counters and can generate an interrupt when a rollover occurs. These registers are generally used in conjunction with software to maintain a 32-bit counter. These counters maintain the time-sensitive lower 16 bits of the count while software maintains the upper 16 bits.

2.2.6 General-Purpose Timer

This 32-bit timer, clocked at one half the 10 MHz transmit clock frequency, is used for timing user definable events. The timer measures events ranging from microseconds up to minutes. The time can be calculated by multiplying the count value by 200 ns (½ the transmit clock period). Table 2-1 gives some example values. To use the timer, you first load the timer with a count value, then start the timer by setting the ST bit in the Command register. The SONIC then begins decrementing the timer. When the rollover is reached (0000 0000h to FFFF FFFFh), the Timer Complete (TC) bit in the Interrupt Status register is set. Note that the timer does not stop when the rollover occurs, but continues to decrement (from FFFF FFFFh). It must be explicitly stopped by setting the STP bit in the Command register.

Table 2-1. Example Timer Values

Timer	WT1	WT0
0.1 sec	7	A120
0.5 sec	26	25A0
1.0 sec	4C	4B4
10 sec	2FA	F080
30 sec	8F0	D180
1 min	11E1	A300
5 min	5968	2F00
10 min	B2D0	2E00

2.2.7 Silicon Revision Register

This register supplies information on the revision stepping of the SONIC. This register begins at zero and counts upward. Contact National Semiconductor for latest information on this register.

3.0 INITIALIZING THE SONIC

Initializing the SONIC is the crucial first step before any SONIC operations can commence. This step involves setting up the SONIC's registers for reception and transmission and initializing the memory structures for the Buffer Management. This section describes the initialization process by introducing what information is needed, then discussing an example initialization routine.

Getting Started

Before initializing the SONIC, a few details regarding the hardware and network operating system must be obtained. By answering the questions below, the required information can be gathered.

1) What is the bus size?

   The SONIC supports bus sizes of 16 or 32 bits.

2) Does the system operate in a synchronous or asynchronous manner?

   This question refers to how the $\overline{\text{RDYi}}$ (or $\overline{\text{DASCK0,1}}$) input is issued to the SONIC. If this line is asserted with guaranteed setup and hold times by the hardware, use synchronous mode; otherwise, use asynchronous mode. Synchronous mode has the advantage of having a minimum memory cycle of 2 bus clocks as opposed to 3 bus clocks for asynchronous mode.

3) What is the maximum bus latency does the SONIC expect?

   The bus latency is the time from when the SONIC requests for the bus (by asserting the HOLD or $\overline{\text{BR}}$ pin) to when the SONIC begins using the bus. The bus latency tolerance can be increased by programming the transmit FIFO threshold higher and the receive FIFO lower. The bus latency tolerance is calculated by the following equations:

   TX FIFO Tolerance = (FIFO threshold) * (0.8 μs)

   RXFIFO Tolerance = (32 − FIFO threshold) * (0.8 μs)

4) Do wait-states need to be added into the memory cycle?

   The SONIC can operate up to a 2 bus clock memory cycle. If this is too fast, you can program the SONIC to insert 1 to 3 wait-states for each memory cycle. A two clock memory cycle requires a memory access time of approximately 40 ns–50 ns. (Note that wait state can also be inserted by hardware using the $\overline{\text{RDYi}}$ or $\overline{\text{DSACK0}}$, $\overline{1}$ inputs.)

5) What type of packets do you want to accept?

   The SONIC is generally programmed to accept its own physical address and the Broadcast address. In some applications, however, the SONIC may be programmed to accept multiple physical/multicast addresses (up to 16), and errored and runt packets.

6) What is the maximum number of consecutive packets that you expect to receive?

   This question is perhaps the most difficult to answer since it deals with the upper level protocols. In many transport protocols, flow control is used by the receiving node to limit the number of consecutive packets the transmitting node may send unacknowledged. This is generally called the "window size". Ideally, the software provides the SONIC with the memory resources it needs to completely buffer a complete "window".

7) What types of interrupts do you want the system to respond to?

   The SONIC can generate a variety of interrupts. Not all interrupts, however, need be (or should be) used to generate interrupts to the system. For maximum performance, you want as few interrupts as possible. A typical system allows interrupts occurring from good receptions and transmissions, and errored transmissions.

Initialization Example

Once the above questions have been answered, you can begin coding the initialization routine. This routine has been divided into 9 steps, but, only steps 1 and 9 need to be followed in the order presented. Example code is provided in the appendix.

1) Reset the SONIC: When the SONIC is powered-on, the hardware generally resets the SONIC by pulsing the $\overline{\text{RESET}}$ pin low. Thus, software does nothing to reset the SONIC. Once reset, the SONIC remains in reset mode until the RST bit in the Command register is cleared. If the hardware does not provide the reset, the software can perform the functional equivalent by simply setting the RST bit. All initialization should be done in reset mode to prevent spurious actions by the SONIC.

2) Configure the System Interface: This step writes to the Data Configuration Register (DCR) to configure the SONIC's bus interface circuitry. The configuration information is found by answering questions 1 through 4, discussed above. Note that the DCR can only be written to in reset mode.

3) Set Up the Receive Filters: This step determines what types of packets to accept (i.e., Physical, Multicast, Broadcast, Runt, and Errored packets) and what addresses to accept. The type of packet to accept is programmed in the Receive Configuration register and the addresses to accept are programmed into the Content Address Memory (CAM). See section 2.2.4 for loading the CAM.

4) Enable the Interrupts: This step enables the interrupts by writing to the Interrupt Mask register (IMR). Note that the interrupting condition is indicated by the Interrupt Status Register (ISR), but will not generate an interrupt unless the corresponding IMR bit is set. Note also that if the SONIC is initialized in reset mode, no interrupts can be generated.

5) Initialize Memory: This step initializes the three memory structures used by the SONIC for transmission and reception and allocates the memory blocks for storing received packets. An initialization example is illustrated in *Figures 3-1* and *3-2*. The *non-shaded* areas indicate fields which must be initialized and *shaded* areas indicate fields which are written to by the SONIC.

There are a few caveats discussed below:

All Descriptor Areas:

- Descriptor must be aligned to word (16-bit) boundaries in 16-bit mode and aligned to double word (32-bit) boundaries in 32-bit mode.
- The Descriptor Areas must not cross over a 32k word boundary since it only operates within this range.
- In 32-bit mode, the upper 16 data bits, D<31:16> are not used.

Transmit Descriptor Area:

- The transmit buffers (Transmit Buffer Area) may be aligned to any boundary; that is, the TXpkt.ptr0, 1 fields may contain any value.
- The packet and fragment size may be as low as 1 byte; that is, the TXpkt.pkt_size and TXpkt.frag_size may contain the value of one.

Receive Resource Area

- The resource descriptors must be contiguous and can not straddle the endpoints.
- In the lower buffer pointer field, RXrsrc.ptr0, the SONIC ignores least significant bit in 16-bit mode and the 2 least significant bits in 32-bit mode. This forces receive buffers to always align to either word or double word boundaries.

6) Initialize the Buffer Management Registers: This step initializes the buffer management registers to the starting positions in the buffer management (see *Figures 3-1* and *3-2*). These initialized registers are shown in Table 3-1.

7) Issue RRA command: By setting the RRRA bit in the Command register, you force the SONIC to read the RRA. The SONIC reads the RRA beginning at the RRP location and loads the following registers. (For mnemonics description, see appendix.)

   CRBA0 ← RXrsrc.ptr0

   CRBA1 ← RXrsrc.ptr1

   RBWC0 ← RXrsrc.wc0

   RBWC1 ← RXrsrc.wc1

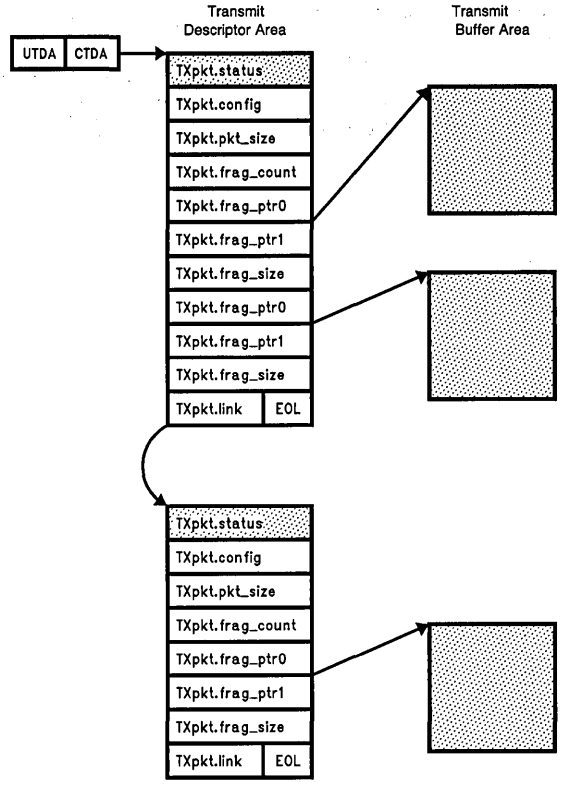
   After this command has executed (RRRA bit resets), the SONIC is ready to store the next packet in the first RBA allocated to it.

8) Clear and Tally Counters (optional): The tally counters (CRC, Frame Alignment, and Missed Packets) may be cleared by writing FFFFh to these registers. These counters will rollover after FFFFh is reached.

9) Bring the SONIC On-line: This last step commissions the SONIC to receive, transmit, and generate interrupts. The software enables the SONIC by setting the RXEN bit and clearing the RST bit in the Command register.

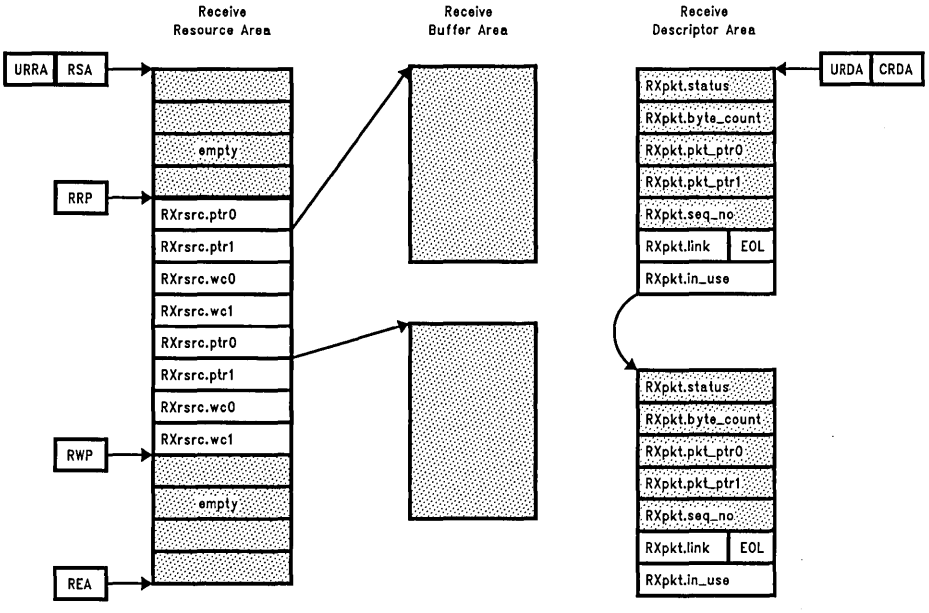
TABLE 3-1. Initialization of Buffer Management Registers

Reg.	Initialized with
URDA	A<31:16> of starting location of RDA
CRDA	A<15:1> of starting location of RDA
UTDA	A<31:16> of starting location of TDA
CTDA	A<15:1> of starting location of TDA
URRA	A<31:16> of starting location of RRA
RSA	A<15:1> of starting location of RRA
REA	A<15:1> of ending location of RRA
RRP	Points to first descriptor the SONIC reads
RWP	Points to next location where the software will place a descriptor



TL/F/11140–14

FIGURE 3-1. Initialization Example for Transmit Buffer Management (shaded areas not initialized)

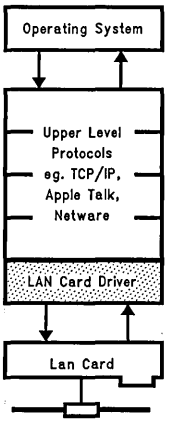


TL/F/11140–15

FIGURE 3-2. Initialization Example for Receive Buffer Management (shaded areas not initialized)

4.0 WRITING DRIVERS FOR THE SONIC

The Driver (see *Figure 4-1*), being the lowest level of software, shields the upper software levels from the details of the hardware. The Driver performs the required low-level transmit and receive functions such as passing packet up to the upper level software, recycling receive buffers, and enqueuing packets for transmission. The Driver performance is important since it may potentially receive packets at the full network rate. Any packet losses at this level can severely affect the overall performance of the network. This section describes the basic algorithms for writing a Driver for the SONIC. Example code is provided in the appendix.



TL/F/11140–16

FIGURE 4-1. Relationship of Driver of Upper Level Software

Overview

The Driver for the SONIC consist of two procedures, INITIATE_TX *(Figure 4-2)* and SONIC_ISR *(Figure 4-3)* for transmit and receive operations. During transmit operations, the upper level software first assembles packets for transmission by gathering the pointers to the fragments and then calling INITIATE_TX to begin the transmission. When the SONIC finishes transmission, it interrupts the system. The system then enters the interrupt service routine, SONIC_ISR, where it reports the status of the packets transmitted. During received operations, the SONIC also interrupts the system upon receiving a packet. The system enters SONIC_ISR to post status and then to pass the packet up to the upper level software via pointers.

4.1 INITIATE_TX

This procedure requires that all pointers to the fragments and the sizes of these fragments are passed down to it by the upper level software. It only initiates a packet for transmission; it does not report status. This action is performed by SONIC_ISR after the packet has been transmitted. INITIATE_TX operates as follows:

1) Obtains the pointers delivered by the upper level software and fills out a descriptor in the Transmit Descriptor area (TDA).
2) If the packet is less than 64 bytes, it pads it out to this length.
3) Issue the transmit command to the SONIC and return.

It is important that descriptors are appended in the manner prescribed in section 1.3. This algorithm improves performance by guaranteeing that the SONIC continues to transmit all packets in the descriptor list.

4.2 SONIC_ISR

This procedure is the interrupt service routine which responds to three interrupts generated by the SONIC: PACKET RECEIVED, TRANSMISSION DONE, and TRANSMIT ERROR. Interrupts occurring before and during the interrupt service routine are serviced before SONIC_ISR exits. SONIC_ISR is broken down into three main sections: (1) reading the cause of the interrupt, (2) processing received packets, and (3) posting status of transmitted packets. The first action performed is finding the cause of the interrupt. For receive interrupts, SONIC_ISR jumps to the receive routine, and for transmit interrupts (good and errored transmissions), it jumps to the transmit routine. The receive routine examines the first descriptor in the RDA, then passes the pointer of the packet up to the upper level software for further processing. It continues reading the RDA until it reaches the end of the descriptor list. The receive routine also recycles receive buffers as necessary. The transmit routine reads the first descriptor in the TDA and reports the status of the transmitted packet to the upper level software. If more than one packet has been enqueued, the transmit routine examines the complete list in the TDA. SONIC_ISR is summarized below.

Reading the Interrupt

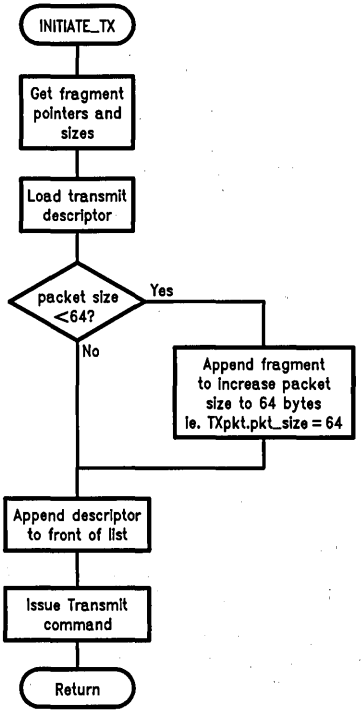
1) Read the Interrupt Status register for the cause of interrupt. If a transmit interrupt has occurred, go to step 2; if a receive interrupt has occurred, go to step 4; or if no more interrupts are present, return.

Transmit Routine

2) Read the next TXpkt.status in the Transmit Descriptor Area and post status to the upper level software.
3) Read the End of List (EOL) bit in the TXpkt.link field to determine if the current descriptor is the last descriptor. If it is not, go back to step 2 to post status of the remaining packets; otherwise go back to step 1.

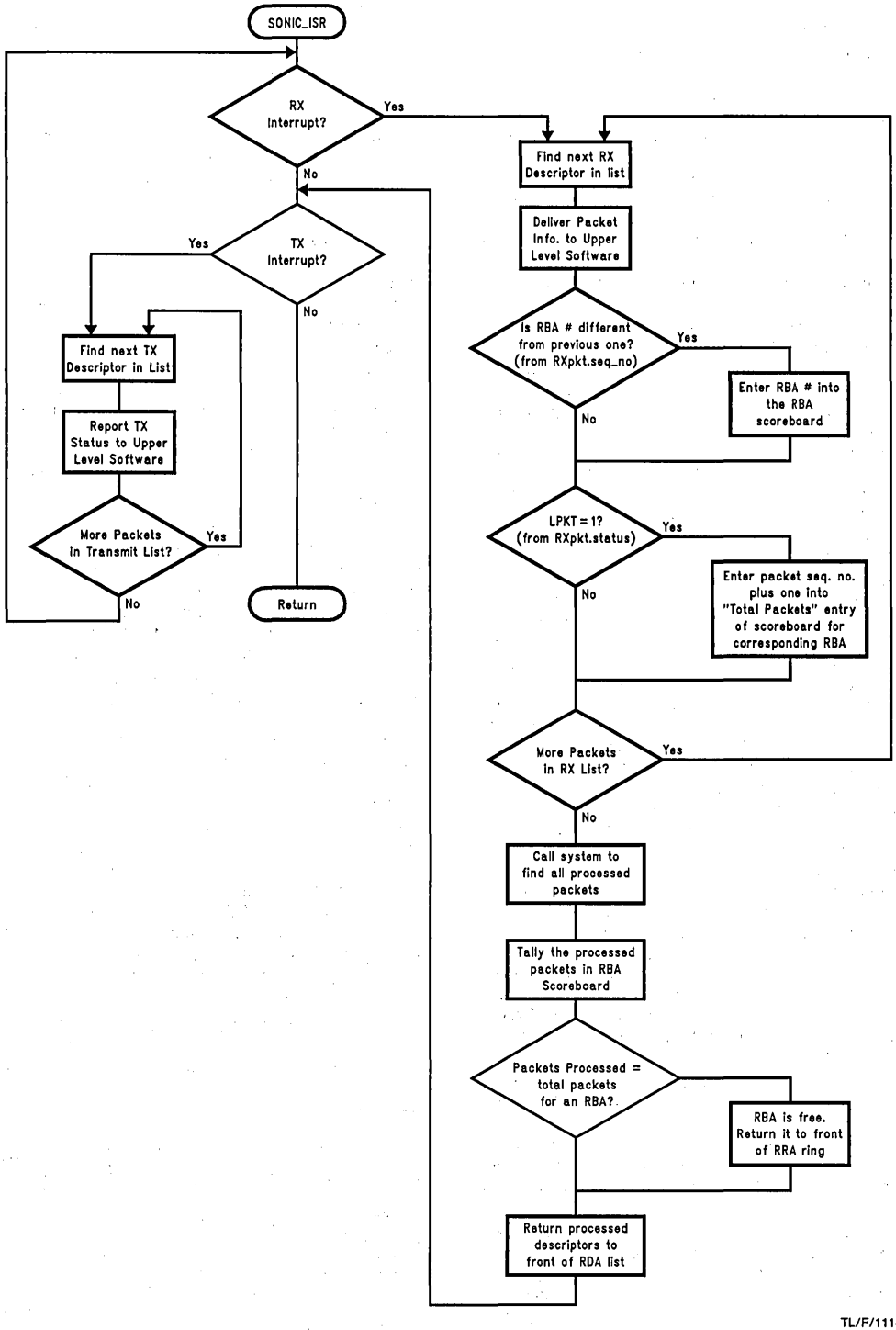
Receive Routine

4) Read the next RXpkt.status field in the Receive Descriptor Area and pass the pointer and status of the packet up to the upper level software.
5) Read the RXpkt.seq_no field. If the RBA number is different from the previous one, enter the RBA number into the RBA "scoreboard". For more information, see section 1.2.
6) Check the LPKT bit from the RXpkt.status field. If set to "1", enter the packet sequence number (from the RXpkt.seq_no) into the RBA scoreboard.
7) Read the RXpkt.in_use field, if the field is cleared to all zeros, go back to step 4 to process the remaining packets; otherwise if RXpkt.in_use is not equal to zero, the end of the list has been reached; proceed to step 7.
8) Call the system to determine which packets have been processed by the upper level software. Tally the processed packets in he RBA scoreboard.
9) Find freed up RBAs and return them to the front of Receive Resource Area (RRA).
10) Find the freed up receive descriptors and return them to the front of the descriptor list; then go to step 1.



TL/F/11140–17

FIGURE 4-2. INITIATE_TX Routine



TL/F/11140–18

FIGURE 4-3. SONIC__ISR Routine

5.0 STRATEGIES FOR IMPROVING DRIVER PERFORMANCE

Making the Driver as efficient as possible is crucial for the overall performance of the network. Empirical results have shown that the difference between a poor and a good Driver can vary as much as 10% to 20%. The Driver is particularly vulnerable to becoming a bottleneck since it may, at times, be receiving data at the full network bandwidth (10 Mb/s). Any packets that are lost at the Driver level impacts all levels. While upper level protocols provide packet recovery mechanisms, these tend to be quite slow (on the order of seconds). Typically, software timers must time out before the upper level software retransmits an unacknowledged packet. In this section, some hints are discussed to make a fast Driver.

1) Write the Driver in assembly code: The fastest code is generally written in assembly code since people write more efficient code than a compiler. Writing your own assembly code also gives you the option to use some "tricks" which are not normally accepted as "good" programming practice. One such example is using a JUMP statement instead of a CALL statement. The JUMP statement, by nature, is quite messy, but is considerably faster since it involves less CPU cycles. Of course, the disadvantage in using assembly code is that it is less readable and portable. As a compromise, you may consider a good optimizing compiler.
2) Reduce the Number of Interrupts: Interrupts to the system inherently make it less efficient since the CPU must make a context switch between what it was currently doing to the interrupt service routine. This switch involves pushing the CPU registers onto the stack, jumping to an interrupt vector table, issuing an interrupt acknowledge to the interrupt controller, then executing the interrupt service routine. The overhead associated with each interrupt makes the CPU less efficient. The example interrupt service routine discussed in section 4.0, responded to interrupts generated from good transmission and receptions, and errored transmission. It is possible, however, to reduce the source of interrupts to just two, allowing only interrupts to occur from good receptions and errored transmissions. The reason good transmission interrupts may be eliminated is because the upper level software generally does nothing for these events. Only for an errored transmission must the upper level software intervene such as to retransmit the packet. Good transmissions, while they still need to be reported, can be status on a less timely basis such as after processing receive interrupts or after a specified time period. The SONIC's General Purpose timer can be used to generate such a time period.
3) Append Transmit Descriptors as described in section 1.0: The algorithm described guarantees that the SONIC continues to transmit all packets in the list, even if it has reached the point where the new descriptor(s) have been appended to the end of the list. If the algorithm is not followed, the SONIC may stop at the enjoining point and this forces the Driver to intervene.
4) Supply Sufficient Number of Receive Packet Descriptors: Since the receive descriptor uses a relatively small amount of memory (7 words or double words, depending on the data size mode), allocate sufficient number of them such that the SONIC never (or at least rarely) runs out of them. If the SONIC ever runs out of them, reception ceases, resulting in packet losses. The number of descriptors to allocate can be determined by answering question 6 of section 3.0.
5) Make the Receive Resource Area (RRA) Sufficiently Large: Since the RRA does not take up much memory (4 words or double words per descriptor), make it larger than the total number descriptors you expect to put into it. For example, if you expect you will need 10 resource descriptors, make the RRA large enough to accommodate 15 descriptors. Making the RRA larger than you will need, prevents the RRA from becoming a bottleneck in adding more resources.
6) Optimize the Size of the Receive Buffer Areas (RBAs): Generally speaking, the larger the RBAs, the more efficient the Driver. This is because the Driver handles fewer number of receive buffers and, thus, less processing time is dedicated to managing the buffers. There is a tradeoff, however. If the buffers are very large, the entire buffer areas are locked out for recycling so that large buffers become less space efficient in memory. As a guideline, 4k to 8k byte RBAs are good starting points for experimentation. Use larger buffers, if memory is plentiful.

6.0 SELF-TEST DIAGNOSTICS

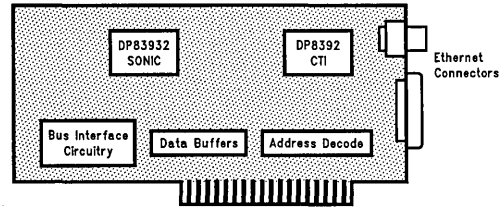
After the hardware has been designed and the Drivers written, there is still a need to verify that the hardware is still functioning. Rough shipping or improper handling (without static protection) can produce innumerable problems. Some boards which work fine in the lab invariably fail in the field. Thus, self-test diagnostics are used to determine the health of the boards and diagnose problems if something is amiss.

Figure 6-1 shows the basic components of the Ethernet system: address decode circuitry, data buffers, bus interface logic, Ethernet chipset (SONIC and transceiver) and the Ethernet connectors (BNC and 15-pin D). The Ethernet hardware can be fully tested by using the SONIC's three loopback modes. Each loopback mode is full-duplex, transmitting data as well as receiving it and are summarized below. An example routine is given in the appendix.

Mode 1: Data is routed back through the SONIC's MAC Unit. Both the transmit and receive Buffer Management operations are active and must be initialized accordingly. Verifies the MAC Unit, Bus interface logic, address decode circuitry and data buffers.

Mode 2: Similar to above, but data is routed back through the SONIC's ENDEC Unit. Verifies the SONIC's ENDEC unit.

Mode 3: Similar to above, but data is routed back at the transceiver. Verifies the Ethernet connectors (BNC and 15-pin D) and Ethernet transceiver (DP8392 CTI).



TL/F/11140–19

FIGURE 6-1. Basic Components of Ethernet Hardware

Appendix

A. Initialization Routine

```
/*****************************************************************/
/*                                                               */
/* Initialization Routine for SONIC                                  */
/*                                                               */
/*****************************************************************/

sonic_init ()
      {
      unsigned short init_RRA[512];   /* memory for RRA */

      /* initialize some registers */
      set_reg_value();

      /* allocate memory for TX descriptors and init UTDA and CTDA */
      init_tda()

     /* Init receive buffer area and RX registers  */
      Init_Des_page();
      Initial_RRA(RRA_NUM);
      Init_RDA(RDA_NUM);

      /* Issue Read RRA command */
      /* Must first bring SONIC out of reset before issuing any
         commands */
      REG_WRITE(card.crd_iobase+SONIC_cr*2, 0x0);
      REG_WRITE(card.crd_iobase+SONIC_cr*2, 0x0100);

      /* Bring SONIC on-line by enabling MAC receiver */
      REG_WRITE(card.crd_iobase+SONIC_cr*2, 0x0008);
      }
/*****************************************************************/
/* This routine initializes some of the SONIC's registers.     */
/*   ie., CR, DCR, RCR, IMR, ISR, CRCT, FAET, and MPT          */
/*****************************************************************/
set_reg_value()
{

      /* Put SONIC is reset */
      REG_WRITE(card.crd_iobase+SONIC_cr*2, 0x0080);

/* dcr value depends upon data width (16 or 32 bits) */
   #ifdef BIT32
      REG_WRITE(card.crd_iobase + SONIC_dcr*2, 0x00f9);
   #else
      REG_WRITE(card.crd_iobase + SONIC_dcr*2, 0x00d9);
   #endif

      REG_WRITE(card.crd_iobase + SONIC_rcr*2, 0x0000);
      REG_WRITE(card.crd_iobase + SONIC_imr*2, 0x3fff);
   /* Clear ISR  */
      REG_WRITE(card.crd_iobase + SONIC_isr*2, 0xffff);
   /* Clear Tally counters by writing FFFFh to them */
      REG_WRITE(card.crd_iobase + SONIC_crct*2, 0xffff);
      REG_WRITE(card.crd_iobase + SONIC_faet*2, 0xffff);
      REG_WRITE(card.crd_iobase + SONIC_mpt*2, 0xffff);
}
```

TL/F/11140–20

```
/*************************************************************
 *
 * Allocate memory for TDA and initialize UTDA and CTDA registers.
 *
 *************************************************************/

init_tda()
{
      short i;
      unsigned long addr;
      unsigned long tda1_start, tda2_start, tda3_start;
      unsigned short saddr;
      unsigned short u16, l16;

      /* Allocate memory for TDAs */
      tda1=(ONE_FRAG_TDA *) malloc(sizeof(ONE_FRAG_TDA) + 2);
      tda1_start = (unsigned long) tda1;
      tda2=(TWO_FRAG_TDA *) malloc(sizeof(TWO_FRAG_TDA) + 2);
      tda2_start = (unsigned long) tda2;
      tda3=(TWO_FRAG_TDA *) malloc(sizeof(TWO_FRAG_TDA) + 2);
      tda3_start = (unsigned long) tda3;

      /* Force TX descriptors to double word alignment */
   #ifdef BIT32
      if ( (tda1_start & 0x00000003) == 0)
            ;
      else
            tda1_start += 2;
      if ( (tda2_start & 0x00000003) == 0)
            ;
      else
            tda2_start += 2;
      if ( (tda3_start & 0x00000003) == 0)
            ;
      else
            tda3_start += 2;
   #endif
      /* Convert the double word alignment address to pointer */
      tda1=(ONE_FRAG_TDA *) tda1_start;
      tda2=(TWO_FRAG_TDA *) tda2_start;
      tda3=(TWO_FRAG_TDA *) tda3_start;

      /* Finding effective address of TDA1 to load UTDA and CTDA regs.*/
      addr=(unsigned long) tda1;     /* Using large mem. model..*/
                  /* addr is the address in 8086 format */
                  /* upper 16 bits = BASE, lower 16 bits = OFFSET */
      u16 = addr >> 16;
      l16 = addr;
      addr=(unsigned long) u16 * 16 + l16;
      u16 =addr >> 16;
      REG_WRITE(card.crd_iobase+SONIC_utda*2, u16);
      REG_WRITE(card.crd_iobase+SONIC_ctda*2, addr);
      }
```

TL/F/11140–21

```
/*************************************************************
      Name
            Initialize Descriptor Page
      Syntax
            Init_Des_Page();
      Description
            This function  gets 3 4K consecutive bytes of memory
            from  the host for the RBA. Also initializes the URRA
            and CDP registers.

      Input
            None.

      Author
            Michael Lui

*************************************************************/

short Init_Des_Page()

{
      unsigned short urra;     /* upper 16 bits of the beginning
                                  addr of RRA */
      unsigned short urda;     /* upper 16 bits of the beginning
                                  addr of RDA */
      unsigned short cdp;           /* beginning address of the cdp */
      R_DESCRIPTOR *temp_RDA;
      unsigned short i;             /* index */
      unsigned long laddr;
      unsigned long addr;
      long EA();
      unsigned long rba1_start, rba2_start, rba3_start;
      unsigned short u16, l16;

     /* allocate memory to RDAs */
      L_RDA=F_RDA=NULL;
      for (i=0; i<RDA_NUM; i++) {
      temp_RDA=(R_DESCRIPTOR *) malloc(sizeof(R_DESCRIPTOR) + 2);

      /* force double word alignment for RX descriptor */
#ifdef BIT32
      addr = (unsigned long) temp_RDA;
      if ((addr & 0x00000003) == 0)
            ;
      else
            addr += 2;
      temp_RDA = (R_DESCRIPTOR *) addr;
#endif
      temp_RDA->next=NULL;
      if (F_RDA == NULL)
            L_RDA=F_RDA=temp_RDA;
      else {
            L_RDA->next=temp_RDA;
            L_RDA=temp_RDA;
      }
      }
      /* allocate memory for RBA */
      init_RBA1=(unsigned char *) malloc(4100);
```

TL/F/11140–22

```
        init_RBA2=(unsigned char *) malloc(4100);
        init_RBA3=(unsigned char *) malloc(4100);

        rba1_start=(unsigned long) init_RBA1;
        rba2_start=(unsigned long) init_RBA2;
        rba3_start=(unsigned long) init_RBA3;

        /* forcing double word alignment for RBAs.  */
ifdef BIT32
        if ( (rba1_start & 0x00000003) == 0)
        ;
        else
        rba1_start+=2;
        if ( (rba2_start & 0x00000003) == 0)
        ;
        else
        rba2_start+=2;
        if ( (rba3_start & 0x00000003) == 0)
        ;
        else
        rba3_start+=2;
#endif
        /* Convert double word alignment address to pointer */
        RBA1 = (unsigned char*) rba1_start;
        RBA2 = (unsigned char*) rba2_start;
        RBA3 = (unsigned char*) rba3_start;

        /* initialize  URRA and CDP registers */
        RRA_start = (unsigned long) init_RRA;
        /* check RRA is aligned on double word boundary */
ifdef BIT32
        if ( (RRA_start & 0x00000003) == 0)
        ;
        else
        RRA_start+=2;
#endif

        /* Assign urra */
        laddr = (unsigned long) RRA_start;
        u16=laddr >> 16;
        l16=laddr;
        laddr = (unsigned long) u16 * 16 + l16;
        urra = laddr >> 16;
        /* Load the URRA register */
        REG_WRITE(card.crd_iobase+SONIC_urra*2, urra);

        /* load the CDA descriptor pointer */
        laddr = (unsigned long)u16 * 16 +l16 +CAM_OFFSET;
        cdp=laddr;
        /* load the CDP register */
        REG_WRITE(card.crd_iobase+SONIC_cdp*2,cdp);
}
```

TL/F/11140–23

```
/*****************************************************************
      Name
      Initialize RRA

      Syntax
      flag=Init_RRA(n);

      Description
      This function will create a circular queue with n
      number of RRA descriptors in it. The RRA descriptors
      are pointing to the     corresponding RBA blocks. It will
      also load the RSA, REA, RRP, and RWP registers.

      Returned Value
      1 = Success
      0 = Failed

      Author
      Michael Lui

*****************************************************************/

short Initial_RRA()
{
      struct sonicreg *sonic=0;
unsigned short rsa;           /* Resource Start Area */
unsigned short rea;           /* Resource End Area */
unsigned short rrp;           /* Resource Read Pointer */
unsigned short rwp;      /* Resource Write Pointer */
unsigned short urba;     /* Upper 16 bit of the RBA starting
                                  address */
unsigned short lrba;     /* Lower 16 bit of the RBA starting
                                  address */
unsigned short i;        /* for loop index */
      unsigned short low_addr;
      unsigned short high_addr;
      unsigned long addr,laddr;
short inc;               /* RRA increment */
     unsigned short u16, l16;

      addr = (unsigned long) RRA_start;
      u16=addr >> 16;
      l16=addr;
addr = (unsigned long) u16 * 16 + l16;

      /* Lower 16 bit of the RRA */
rsa = (unsigned short) addr;
      /* Load the RSA Register */
      REG_WRITE(card.crd_iobase+SONIC_rsa*2, rsa);

laddr=addr + RWP_OFFSET;
rea = (unsigned short) laddr; /* Ending address of RRA */
      /* Load the REA Register */
      REG_WRITE(card.crd_iobase+SONIC_rea*2, rea);

rrp = rsa;  /* Read Pointer starts at the beginning
                        address */

      /* Load the RRP Register */
```

TL/F/11140–24

```
      REG_WRITE(card.crd_iobase+SONIC_rrp*2, rrp);

laddr = addr + RWP_OFFSET/2;
rwp = (unsigned short) laddr; /* Only 3 descriptors
                                    initially */
      /* Load the RWP Register */
      REG_WRITE(card.crd_iobase+SONIC_rwp*2, rwp);

      /* Initialize the RRA descriptors */
      RRA=RRA_start;
      /* for 32-bit memory each descriptor uses a double word, for
         16-bit memory, each descr. uses a word. */
#ifdef BIT32
      inc=4;
#else
      inc=2;
#endif
      /* Load RBA1 address */
     addr=(unsigned long) RBA1;
     u16=addr >> 16;
     l16=addr;
      addr=(unsigned long)u16 * 16 + l16;
      low_addr = addr & 0x0000ffff;
      * (unsigned long *)RRA = low_addr;
      RRA +=inc;
      * (unsigned long *)RRA = addr >> 16;
      RRA +=inc;
      /* Load RXrsrc.buff_wc0 */
      * (unsigned short *)RRA = 0x0800;
      RRA +=inc;
      /* Load RXrsrc.buff_wc1 */
      * (unsigned short *)RRA = 0;
      RRA +=inc;

      /* Load RBA2 address */
     addr=(unsigned long) RBA2;
     u16=addr >> 16;
     l16=addr;
      addr=(unsigned long) u16 * 16 + l16;
      low_addr = addr & 0x0000ffff;
      * (unsigned short *)RRA = low_addr;
      RRA +=inc;
      * (unsigned short *)RRA = addr >> 16;
      RRA +=inc;
      /* Load RXrsrc.buff_wc0 */
      * (unsigned short *)RRA = 0x0800;
      RRA+=inc;
      /* Load RXrsrc.buff_wc1 */
      * (unsigned short *)RRA = 0;
      RRA +=inc;

      /* Load RBA3 address */
     addr=(unsigned long) RBA3;
     u16=addr >> 16;
     l16=addr;
      addr=(unsigned long)u16 * 16 + l16;
      low_addr = addr & 0x0000ffff;
      *(unsigned short *)RRA = low_addr;
      RRA+=inc;
      * (unsigned short *)RRA = addr >> 16;
```

TL/F/11140–25

```
        RRA+=inc;
        /* Load RXrsrc.buff_wc0 */
        * (unsigned short *)RRA = 0x0800;
        RRA+=inc;
        /* Load RXrsrc.buff_wc1 */
        * (unsigned short *)RRA = 0;
          RRA+=inc;
}
```

TL/F/11140-26

```
/*************************************************************
      Name
      Initialize RDA

      Syntax
      flag = Init_RDA(n);

      Description
      This function will create a linked list of some
      arbitrary number of packet descriptors. The EOL bit for
      the last    descriptor should set to 1 while the others
      should set to 0.  The in_use field should set to a
      non-zero value for all descriptors. The CRDA register
      should loaded with the address of the first descriptor.

      Returned Value
      1 = Success
      0 = Failed

*************************************************************/
short Init_RDA()
{
unsigned long crda;      /* Current CRDA Register */
      unsigned char *RDA; /* RDA address */
      R_DESCRIPTOR *cur_RDA;  /* current RDA */
unsigned short n_RDA_addr;    /* next RDA address */
      unsigned long addr;
      short i;
      unsigned u16, l16;

crda = (unsigned long)  F_RDA;
      u16 = crda >> 16;
      l16 = crda;
      crda = (unsigned long)u16 * 16 + l16;

      /* Load the CRDA Register */
REG_WRITE(card.crd_iobase+SONIC_crda*2, crda);

cur_RDA=F_RDA;
      while (cur_RDA->next != NULL)
      {
          addr = (unsigned long) cur_RDA->next;
          u16 = addr >> 16;
          l16 = addr;
      addr=(unsigned long) u16 * 16 + l16;
      n_RDA_addr=(unsigned short) addr;
      cur_RDA->pkt_link=n_RDA_addr;
          cur_RDA->status=0;
          cur_RDA->byte_count=0;
          cur_RDA->pkt_ptr0=0;
          cur_RDA->pkt_ptr1=0;
          cur_RDA->seq_no=0;
      cur_RDA->in_use=0xffff;
      cur_RDA=cur_RDA->next;
      }
      /* last descriptor */
cur_RDA->pkt_link=0x0001;     /* last desr. has EOL = 1 */
      cur_RDA->in_use=0xffff;
      lrda = cur_RDA;
}
```

TL/F/11140–27

B. Initiate Transmission Routine

```
/*********************************************************
*
Driver_send().  This routine, called by the upper level
software, gets the byte count, pointers to fragments and
the fragment sizes, enters these parameters into the TDA,
then initiates a transmission.

*********************************************************/
driver_send(ptr)
pktstruc *ptr     /*pointer to structure which gives
                    pkt_size, frag_count, frag_size */
{

    /* Fill out TDA */
    tda->pkt_size=packet_size;
    tda->frag_count=fragment_count;
    for (i=0; i<fragment_count; i++)
        Fill_fragment_ptr_size();

    /* Check packet length; if less than 46 bytes, add pad */
    Check_pkt_length();


    /* Get address of next TX descriptor to use */
    tda->link = get_next(); /* returns addr. of descr. */
    /* Set EOL.to 1.  */
    tda->link |= 0x1;


    /* ISR will Set this flag to 1 */
    xmit_interrupt=0;

    /* Issue transmit command */
    REG_WRITE(card.crd_iobase+SONIC_cr*2, CMD_TXP);


}
```

TL/F/11140–28

C. Interrupt Service Routine

```
/*******************************************
 Interrupt Service Routine

 (For simplicity the code for recycling RBAs
 has heen removed.)
 *******************************************/

interrupt _sonic_isr()
{
    unsigned short imr, isr, mask;
    unsigned int status, byte_count;
    int oldinterrupts; long temp_ptr, ptr;

    mask=0;
    /* mask the imr */
    REG_WRITE(card.crd_iobase+SONIC_imr*2, mask);

    while (isr=REG_READ(card.crd_iobase+SONIC_isr*2)) {
        if (isr & ISR_PKTRX) {
            /* reset PKTRX bit */
            REG_WRITE(card.crd_iobase+SONIC_isr*2,ISR_PKTRX);

            /* Process receive packets */

            while (cur_rda->in_use == 0) {
                    TotalRxPacketCount++;
                    status = cur_rda->status;
                    byte_count = cur_rda->byte_count;
                    temp_ptr = cur_rda->ptr1;
                    temp_ptr = temp_ptr<<16;
                    ptr = temp_ptr | cur_rda->ptr0;
                    /* Report packet to upper level software */
                    packet_received(status,byte_count,ptr);

                    /* Processing packets in order, when LPKT is 1,
                       update the RWP register */
                    if (cur_rda->status==RCR_LPKT) {
                            cur_rwp=cur_rwp->next;
                            /* advance rwp */
                            REG_WRITE(card.crd_iobase+SONIC_rwp*2,
                                                cur_rwp->loc);
                    }

                    /* finish up receive */
                    if (cur_rda->in_use == 0) {
                        cur_rda->in_use=0x0ffff;
                        cur_rda->pkt_link |= 0x1;
                        lrda->pkt_link &= 0x0fffffffe;
                        lrda=cur_rda;
                        cur_rda=cur_rda->next;
                    }
            }

            /* check for RBE overflow (required) */
             isr=REG_READ(card.crd_iobase+SONIC_isr*2);
             if (isr & ISR_RBE) {
                   /* Increment buffer overflow counter */
```

TL/F/11140–29

```
                RRAExhaustCount++;
                /* reset RBE, this also causes the SONIC to read
                   the RRA */
                REG_WRITE(card.crd_iobase+SONIC_isr*2,R_RBE);
            }

            /* check for RDE overflow (optional) */
            if (isr & ISR_RDE) {
                RDAExhaustCount++;
                REG_WRITE(card.crd_iobase+SONIC_isr*2, ISR_RDE);
            }
        }

        /* Process transmitted packets */
        else if (isr & (ISR_TXER|ISR_TXND)) {
            xmit_interrupt=1;
            REG_WRITE(card.crd_iobase+SONIC_isr*2, ISR_TXER|ISR_TXDN);
            while (1) {
                if (tda->status & TCR_PTX) {/* Successful TX occurred */
                    TotalTxPacketCount++;

                    /* Post status of transmitted packet to
                       upper level software */
                    packet_tx(TX_status);
                  /* Increment counters for net. management. */
                    if (tda->status & TCR_DEF)
                        DeferXmissionCount++;
                    if (tda->status & TCR_NCRS)
                        NoCRSCount++;
                    if (tda->status & TCR_CRSL)
                        CRSLostCount++;
                    if (tda->status & TCR_OWC)
                        OutOfWindowCollisionCount++;
                    if (tda->status & TCR_PMB)
                        PacketMonitorBadCount++;
                }
                /* TX abort condition occured. CTDA register points to
                   last descriptor attempted. */
                else {
                    /* Increment counters for net. management. */
                    if (tda->status & TCR_EXD)
                        ExcessDeferalCount++;
                    if (tda->status & TCR_EXC)
                        ExcessCollisionsCount++;
                    if (tda->status & TCR_FU)
                        FIFOUnderRunCount++;
                    if (tda->status & TCR_BCM) {
                        ByteCountMismatchCount++;
                        tda->pkt_size=Total_fragment_size(tda);
                    }
                    if (--RetryCounter == 0)
                        HardTransmitErrorCount++;

                        /* Post status of transmitted packet to
                           upper level software that packet was undeliverable
*/
                            packet_tx(TX_status);
                    else {
                        /* resend the same packet again up to RetryCounter */
                        REG_WRITE(card.crd_iobase+SONIC_cr*2, CMD_TXP);
```

TL/F/11140–30

```
                    }
                }
                /* look for last descriptor in TX list */
                if (tda->link & 0x1)
                    break;
                else
                    tda=tda->next;
            }
        }
    }
    pic_eoi(card.crd_interrupt);
    REG_WRITE(card.crd_iobase+SONIC_imr*2, card.crd_intmask);
}
```

TL/F/11140–31

D. Diagnostic Routine

```
sonic_diag ()
      {
      int oldinterrupt;
      struct aclock *clk, *alock_alarm();
      long timeout=0;
      unsigned short temp, u16, l16, addr;
      unsigned long laddr;
        extern int timeout_func();
        short result;

      /* Before loopback test can commence SONIC needs to be
         initialized */

        /* check BNC cable connection:  If transmission does not
           finish after specified time period (~1sec), the BNC
           connector is not connected.  If excessive collisions
           occur, the cable is not terminated */

      clk=aclock_alarm(50,50,timeout_func, &timeout);
        REG_WRITE(card.crd_iobase+SONIC_isr*2, 0xffff);
        /* Get the 1st tda */
        laddr=(unsigned long) tda1;
        u16=laddr >> 16;
        l16=laddr;
        laddr=(unsigned long)u16 * 16 + l16;
        addr=(unsigned short) laddr;
        REG_WRITE(card.crd_iobase+SONIC_ctda*2, addr);
        tda1->link=0x0001;

      /* Issue transmit command */
      REG_WRITE(card.crd_iobase+SONIC_cr*2, CMD_TXP);
      for (timeout_value=0; timeout_value < 2; ) {
              temp=REG_READ(card.crd_iobase+SONIC_isr*2);
                if (temp & (ISR_TXDN | ISR_TXER))
                    break;
      }
      clock_kill(clk);
      if (timeout_value) {
          check_cable=2;      /*Timeout occurred, BNC not connected*/
          goto final;
      }
      else if (tda1->status & TCR_EXC) {
              check_cable = 3;
              goto final;     /* Exc. Coll. occurred, cable not
                                 terminated */
      }
      else
              check_cable = 1;

        /* MAC loopback */
      laddr = (unsigned long) F_RDA;
      u16=laddr >> 16;
      l16=laddr;
      laddr = (unsigned long)u16 * 16 + l16;
      addr = (unsigned short) laddr;
      REG_WRITE(card.crd_iobase+SONIC_crda*2, addr);
        mac_loopback=loopback(0x0200);
```

TL/F/11140–32

```
        if (mac_loopback != 1)
            goto final;

        /* ENDEC loopback */
      laddr = (unsigned long) F_RDA;
      u16=laddr >> 16;
      l16=laddr;
      laddr = (unsigned long)u16 * 16 + l16;
      addr = (unsigned short) laddr;
      REG_WRITE(card.crd_iobase+SONIC_crda*2, addr);
        endec_loopback=loopback(0x0400);
        if (endec_loopback != 1)
            goto final;

        /* transceiver loopback */
      laddr = (unsigned long) F_RDA;
      u16=laddr >> 16;
      l16=laddr;
      laddr = (unsigned long)u16 * 16 + l16;
      addr = (unsigned short) laddr;
      REG_WRITE(card.crd_iobase+SONIC_crda*2, addr);
        trans_loopback=loopback(0x0600);
        if (trans_loopback != 1)
            goto final;

      return(ok);

final:
      return(error);    /* one of the loopback test failed/*
}


/* This routine is to perform the loopback tests */

loopback( rcr_mode)
unsigned short  rcr_mode;
{
        struct aclock *clk;
        unsigned short temp, u16, l16, addr, rcr_value;
        unsigned long laddr;
        long timeout=0;
        short i;
        struct aphys *phys;

        /* Set up the clock to measure timeout */
      clk=aclock_alarm(50,50,timeout_func, &timeout);
        REG_WRITE(card.crd_iobase+SONIC_isr*2, 0xffff);
        /* Get the 1st tda */
        laddr=(unsigned long) tda1;
        u16=laddr >> 16;
        l16=laddr;
        laddr=(unsigned long)u16 * 16 + l16;
        addr=(unsigned short) laddr;
        tda1->link=0x0001;
        rcr_value=rcr_mode|0x3800;
        REG_WRITE(card.crd_iobase+SONIC_rcr*2, rcr_value);

        /* Out of reset mode */
        REG_WRITE(card.crd_iobase+SONIC_cr*2, 0);
```

TL/F/11140–33

```
        REG_WRITE(card.crd_iobase+SONIC_ctda*2, addr);
        REG_WRITE(card.crd_iobase+SONIC_cr*2, CMD_RXEN);
        /* Issue transmit command */
        REG_WRITE(card.crd_iobase+SONIC_cr*2, CMD_TXP);
      for (timeout_value=0; timeout_value < 2; ) {
              temp=REG_READ(card.crd_iobase+SONIC_isr*2);
                if (temp & (ISR_TXDN | ISR_TXER))
                    break;
      }
      clock_kill(clk);
      if (timeout_value)
          return(2);                    /* timeout error */
      else if (tda1->status & TCR_PTX) {
                if (F_RDA->status & RCR_LBK) {
                    F_RDA->in_use=0xffff;
                    return(1);          /* good TX and RX status */
                }                       /*   loopback OK */
                else {
                    F_RDA->in_use=0xffff;
                    return(3);          /* Bad RX status error */
                }
        }
        else {
            F_RDA->in_use=0xffff;
            return(4);                  /* Bad TX status error */
        }
}
```

TL/F/11140–34

Determining Arbitration and Threshold Levels in a SONIC™ Based Micro Channel Adapter

National Semiconductor
Application Note 747
Bill Carlson, FAE

ABSTRACT

With the number of bus master adapter boards increasing in Micro Channel based systems, many issues arise. This is especially true regarding Bus Master Ethernet LAN controllers such as the DP839EB-MCS. As such, the entire MCA environment needs to be considered so that critical settings for arbitration levels, threshold levels, and fairness options can be chosen. This paper describes these issues as they relate to National Semiconductor's DP839EB-MCS 32-bit Ethernet LAN controller board, which utilizes the DP83932 (SONIC).

The major issues include bus latency, bus efficiency and the contributing factors affecting these critical system level parameters. Factors such as bus occupancy times, DRAM refresh rates, floppy controller accesses, CPU accesses, mass storage transfer rates, latency tolerances, and priority levels all contribute to latency and efficiency. Within this environment, the high performance levels of the SONIC are achieved, even in worst-case scenarios in heavily loaded file servers with multiple bus masters.

It is also important to note that many of the basic concepts and considerations required in this application will also apply to other buses. Although the detailed analysis will differ.

OVERVIEW

The DP83932 (SONIC) is a high performance, 32-bit, bus mastering Ethernet controller designed for a wide variety of applications. These applications include motherboards, routers, bridges and gateways, buffered and intelligent adapter boards, and bus master adapter boards. In each of these applications, determining the optimum thresholds and arbitration levels are key parameters to choose to ensure optimum performance. In determining these parameters, the anticipated system configuration needs to be understood. Specifically, the number and type of bus mastering devices in a system needs to be determined. Once these bus masters have been identified, the device thresholds and board arbitration levels can be determined.

Determining the anticipated number and type of bus masters directly affects a bus specification known as *Bus Latency*. Bus latency is defined as the time between when a bus master requests the bus to when it actually gets it.

Bus latency is a critical systems level specification because if it is too long, a bus master who doesn't get the bus when it needs it could suffer performance degradations or even more severe conditions such as a lost Ethernet packet or missed "sector" in a streaming tape drive. As such the Ethernet controller subsystem needs to have enough tolerance to handle large latencies to guarantee it's access to the bus and avoid this missed packet condition. The SONIC was specifically designed to perform in these applications. By having a high speed, 66 MB/s, DMA host interface the SONIC maximizes bus bandwidth and minimizes time on the bus. Coupled with two efficient, 32 byte receive and transmit FIFOs, the SONIC will tolerate most latencies found in many applications.

Determining bus latencies is easy in many applications. Bridges and gateways, motherboards, intelligent and/or buffered adapter boards are systems in which the anticipated bus masters are known. In these systems it would be common to have the host CPU, a DMA controller, and peripheral devices (SCSI, FDDI, . . .) all known by the system designer before the product is shipped out the door.

It is the designer who has to design a bus master adapter board or motherboard for a target bus (be in Micro Channel, EISA, VME, etc.) with expansion slots who has a tougher problem. He doesn't know what the end system configuration will be so he has to design to what is anticipated to be a worst case system configuration. The adapter board designer's customers would be the systems integrators who need to make sure that his board is designed properly so it will operate in fully loaded systems and still attain the high performance that he expects from this type of bus-mastering device.

Towards this end, this paper is written to assist the SONIC adapter board designer in choosing the correct arbitration and threshold levels for an IBM PS/2 Model 80 application, most probably operating as a file server having multiple LAN and mass storage devices on the MCA bus. For designers of other systems, this paper should help in understanding many of the issues that arise in a bus master LAN environment.

Before discussing this, a few MCA specifics need to be addressed. First off is the arbitration scheme. There can be up to 8 bus master expansion boards on the Model 80 MCA bus, including 8 DMA channels, the system CPU, refresh, and NMI which are on the system motherboard. Most have their own arbitration level as programmed via a POS register. When a device wants ownership of the bus, it asserts the PREEMPT* signal and will then monitor the ARB/GNT* signal, and when high (as controlled by the central arbitration logic on the system board) will place it's arbitration vector on the bus. If it's vector has the highest value, it wins the bus, ARB/GNT* goes low, PREEMPT* is de-asserted, and it can now do data transfers. If other devices want the bus they can asynchronously assert PREEMPT*. The first device has 7.8 μs to get off the bus and then all requesting devices, including the first if it wants to, compete for the bus and the arbitration process starts over again. When determining system characteristics, this 7.8 μs is often used as it dictates the maximum amount of time that a device can own the bus if others are requesting it.

Another aspect of the MCA architecture is a feature called Fairness. Fairness allows all devices access to the bus in a round-robin fashion as determined by pre-assigned priority levels. Carefully choosing which devices are fair or not allows proper performance levels for the various devices on the bus. If fairness is enabled for a device and it currently owns the bus and another device(s) wants it, it will wait to re-arbitrate until all other requesting devices have had a chance on the bus themselves (this is noticed by the absence of an active PREEMPT* signal). In this way no device will hog the bus and prevent others from accessing it. If fairness is disabled for a device, it will arbitrate for the bus any chance a valid arbitration cycle is available, regardless whether other devices are waiting to arbitrate also. Even with fairness enabled, the winner of the bus still needs the highest arbitration level, however, properly setting the fairness option will determine who will do the arbitrating.

In determining the arbitration levels and thresholds the designer of the SONIC bus master adapter board needs to account for a worst case bus situations. This would most likely be a high performance file server with multiple adapter boards. These could include an ESDI disk controller, an SCSI controller for additional disk and tape backup facilities and from 1 to 4 LAN boards to handle a heavily loaded network. Other anticipated bus master boards could also be included in this scenario (e.g., FDDI) but our discussion will be limited to the aforementioned configuration. (This is indeed a worst case scenario. A more typical case for a file server would have 1 or 2 LAN boards and both a SCSI and ESDI controller).

To summarize our worst case scenario for this analysis, we will assume the Micro Channel PS/2 has these adapter boards installed:

- 4 SONIC Bus Master Adapter Boards
- 1 Bus Master SCSI Controller
- 1 Bus Master ESDI Controller

DETERMINING ARBITRATION LEVELS AND THE FAIRNESS OPTION

When determining these it must be understood that the mass storage devices and the LAN controllers have different goals when it comes to bus utilization. The mass storage devices will have large blocks of data to transfer that are typically already stored in a local buffer on the adapter board or on the drive itself. All ESDI disk controllers have a local buffer, some with megabytes of storage. Most SCSI host adapters have buffering as well, although a trend is to use a bus-mastering SCSI controller IC that can gain the bus similar to the way the SONIC does. These don't have local buffering outside of their internal FIFO, but have the data storage on the disk drive itself. The main priority for the storage devices is to transfer as much data as possible for as long as it has the bus. Of second priority is latency toleration. These devices can wait a reasonable amount of time before they get the bus. Because they already have a large amount of data buffered, no data should be lost if it isn't granted the bus immediately. However, when it does get the bus, it needs to transfer as much as possible.

The Bus Master LAN controllers, on the other hand, need to have quicker access than the mass storage devices and within their latency period. This is especially true when receiving a packet, for to get a FIFO overrun error would cause upper protocol layers to initiate long and time consuming recovery procedures. Once they are on the bus, however, they are on for a relatively short period of time. This is due to the fast 20 MB/s MCA transfer rate and the smaller amount of data that is to be transferred at one time. (A disk or tape cache can have many Kbytes available for transfer, the \ 32 byte FIFO will transfer at the most that amount.)

With this in mind, the LAN controllers should be configured to have near immediate access to the bus. As such, each should be set to have a priority level higher than the storage devices. Thus whenever an arbitration takes place, a LAN controller should always participate and win so it can attain bus ownership as soon as possible. The setting of the fairness option should also be chosen to allow the LAN boards immediate bus access. If all devices had enabled the fairness option it is possible for the LAN board to be off the bus for a longer period of time than it's latency tolerance allows, for example as shown in Table I.

TABLE I. Possible (but Not Optimum) Priority Settings for Adapters, but Not the Optimum Solution

Device	Priority	Fairness
LAN0	0	Yes
LAN1	1	Yes
LAN2	3	Yes
LAN3	4	Yes
SCSI	6	Yes
ESDI	7	Yes

In this scenario all devices have fairness enabled and the LAN boards have the higher priority. If a LAN board is awaiting arbitration it will win vs. the ESDI and SCSI boards. However, since fairness is enabled for the LAN boards it means that they must defer arbitrating until all other devices have been on the bus. These boards should participate in every arbitration cycle and by enabling fairness for them, this is prevented. Specifically in this example, the SCSI and ESDI boards will be on the bus consecutively for 7.8 μs each (for 16.2 μs total, including arbitration time) and the LAN boards would miss the intermediary arbitration cycle; this might exceed the boards latency toleration. By disabling fairness on the LAN boards, each is guaranteed to participate in every arbitration cycle and not have to wait for other device's arbitrations and bus occupancy times. Because of this and their higher priority levels, a LAN board will **always** arbitrate and win when an arbitration cycle occurs. We now have this:

TABLE II. Priority Settings for Adapters with Correct Fairness Setting

Device	Priority	Fairness
LAN0	0	No
LAN1	1	No
LAN2	3	No
LAN3	4	No
SCSI	6	Yes
ESDI	7	Yes

What about the storage devices? Fairness should be enabled for them. Due to the large amounts of data available for them to transfer in their respective caches, they will always have a need to own the bus and so they will always be requesting it. If fairness were disabled, the higher priority device (the SCSI controller in this case) would hog the bus and prevent the ESDI controller from accessing it. Thus fairness should be enabled for them.

To summarize, the above configuration will give each LAN board immediate access to the bus. The SCSI and ESDI boards would each have accessibility to the bus and although delayed due to the higher priority LAN boards, their latency tolerances are much higher and would incur only a minor, yet expected loss in bus acquisition time. The settings for the DMA slave ESDI controller that is configured with the Model 80, does indeed default to these settings. Fairness is enabled for it and it occupies DMA channel 7, the lowest priority DMA Channel.

The following *Figure 1* illustrates the sequence of events in a fully loaded, extreme worst case situation by properly setting the arbitration levels and fairness. Other devices such as refresh and the floppy controller will be included later when FIFO thresholds are discussed.

It should be remembered that the system CPU, the floppy controller, refresh, and other devices will be on the bus as well. These, along with the adapter boards all contribute to bus latency. Because of this latency the SONIC's FIFO threshold must be set properly to tolerate the expected latencies and avoid overrun/underrun errors. When set properly the SONIC will achieve the high performance the designer wants and the system's integrator expects.

DETERMINING THRESHOLD LEVELS

The FIFO threshold is an option that is programmed in the SONICs Data Configuration Register and both the receive and transmit FIFOs can be programmed for different values. What is the FIFO threshold? The threshold is simply the point in time that the DMA engine requests the bus after a certain amount of data has filled the FIFOs. For example, a threshold of 1 long word for the receive FIFO would mean that after 4 received bytes from the network have filled the receive FIFO the DMA engine will request the bus. For the transmit FIFO, a threshold of 4 long words would cause the DMA engine to request the bus when the number of bytes in the FIFO falls below 16.

When determining the threshold levels, we need to first explore the specific latencies expected in our worst case scenario. The latency calculation is done by adding together the bus occupancy times of the various bus masters, their priority levels, and the fairness option. We will assume the following:

- All adapter boards have 32-bit MCA bus master interfaces
- The SONIC board transfer rate will be at 250 ns (although MCA will operate @ 200 ns and the SONIC can do synchronous transfers on other buses @ 100 ns)
- Arbitration time will be 300 ns (0.3 μs)
- EMPTY/FILL Mode is enabled for FIFO buffering
- The Floppy controller will request service from DMA Channel 2 every 12 μs and will remain on the bus for 500 ns.
- Refresh occurs every 15.1 μs and inserts itself in the middle of an arbitration cycle, extending it 200 ns for a total arbitration time of 500 ns.

In this example we will assume that the SCSI controller just got on the bus and then immediately afterwards all four LAN boards and the ESDI controller request the bus by asserting PREEMPT*. This example takes a worst case latency and will show how the chosen threshold and arbitration levels and fairness options will guarantee proper system performance by showing how all four LAN boards will be able to access the MCA bus. When these devices request the bus it is to be understood that their FIFO thresholds have been reached. The LAN controllers will be buffering a received packet, a very critical bus access.

What should the threshold levels be for the 4 LAN controllers? Choosing the proper threshold involves trade-offs between a number of systems level specifications. By having a low threshold, maximum latency is assured. However, fewer bytes will transfer so the arbitration percentage will be higher, reducing efficiency. Also, the controller will request the bus more often causing bursty traffic across the bus. A larger threshold on the other hand, solves these problems at the expense of lower bus latency tolerance. In light of this, the thresholds of LAN0:1 should be higher than LAN2:3. LAN0:1 won't see larger latencies due to their higher priorities. However, they shouldn't request the bus again before LAN2:3 get a chance, increasing the latency they already incur. LAN2:3, however, need to tolerate longer latencies than LAN0:1 because, due to their priorities, they will be off the bus for longer periods of time. They will request the bus sooner and more often, however, this shouldn't impact system performance due to the short bus duration. By choosing a threshold of 16 bytes for LAN0:1 and 8 bytes for LAN2:3, as summarized in Table III, a good balance between these issues is achieved.

SCSI	LAN0	LAN1	LAN2	LAN3	ESDI	LAN0	LAN1	LAN2	LAN3	SCSI

FIGURE 1. Bus Ownership In Example PS/2 Under Worst Case Bus Request

Table III shows the arbitration bus priority assignments that show proper settings for the IBM PS/2 Model 80 devices. It should be remembered that these device assignments are determined by the MCA specification. Some of the assignments are pre-set, while others can be occupied by installable adapter boards. For example, refresh and NMI are preset to arbitration levels −2 and −1. The Floppy controller occupies DMA channel 2. The other DMA channels are available for adapter boards.

TABLE III. Arbitration, Fairness, and FIFO Threshold Settings

Device	Priority	Fairness	Threshold	Latency	Latency μs
Refresh	−2				
NMI	−1				
LMA0	0	No	16 Bytes	16 Bytes	12.8
LAN1	1	No	16	16	12.8
Floppy	2				
LAN2	3	No	8	24	19.2
LAN3	4	No	8	24	19.2
Available (Note 1)	5				
SCSI	6	Yes			
ESDI (Note 2)	7	Yes			
Available	8–E				
CPU	F				

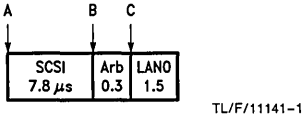
Note 1: An IBM ST-506 disk controller will default to an arbitration level of 5 with fairness enabled.

Note 2: An IBM ESDI controller will default to arbitration level of 7 with fairness enabled.

Devices 8–E are available for bus masters. In our example, DMA channels 0, 1, 3, and 4 are masked out and are used to hold the bus mastering LAN controllers. The bus master SCSI host adapter is put at ARB 6 with DMA channel 6 masked out. A standard PS/2 Model 80 comes with an ESDI disk controller operating as a DMA slave at ARB 7. This is the default setting for this controller. Because of this, the LAN designer doesn't have to worry about the arbitration level and fairness options for this controller. It can be assumed that the SCSI host adapter will be configured in the same way: with a low priority and with fairness enabled. In our example we have assumed a bus mastering ESDI controller; however, the standard one is a DMA slave device. For our discussion, though, we will assume it is a bus master for clarity's sake.

Once the arbitration levels and thresholds are determined for the LAN boards, they must be set when installed. IBM automatically sets the default values for the ESDI controller, but what about the LAN boards. How should they be set? Does the end user have to be aware of all these issues just to install a board? A simple solution would be for the driver to call a BIOS routine that would poll all the MCA slots to determine how many LAN boards are installed. The driver would then set the threshold and arbitration levels appropriately for each board. Using this method the user would be far removed from the details of these specifics and a smooth installation would be insured.

At point "A" in *Figure 2* below, LAN0:3 and the ESDI controller request the bus. At point "B", 7.8 μs later the SCSI controller removes itself and an arbitration cycle begins with the other devices participating. It should be noted that if the bus-mastering SCSI controller IC is in the middle of a block transfer when it gets off, it will need to tell the target so it won't request more data transfers of it and the system any more. It does this by simply refusing to issue more acknowledges to the target after the REQ/ACK offset has been met (in synchronous mode). In this way the target won't be requesting the initiator until it has access to the system bus again. The effect is that the SCSI controller can be off the bus even during the middle of a block transfer. After the arbitration following this SCSI transfer, LAN0 will win due to it's higher priority. To determine system latency we will need to calculate the sum total of the occupancy times of all devices. If this latency is less than the maximum latency tolerance of all the LAN devices, proper bus access and performance levels can be expected. If not, FIFO overruns would occur, the situation we are trying to prevent and will show won't happen.



TL/F/11141–1

FIGURE 2. Initial DMA Sequence

With that, how long will LAN0 be on the bus? Since LAN0 didn't get the bus until point "C", 8.1 μs later, and the controller has been programmed for EMPTY/FILL mode, it will transfer the sum of the number of bytes determined by the FIFO threshold and the number of bytes accumulated from the network since the request was made. Let's call the "threshold" transfer time T_T and the transfer time for the accumulated bytes T_A. We will call the number of accumulated bytes simply "#". Since our threshold for LAN0 is 16 bytes, T_T will be the time it takes to transfer 16 bytes. T_A will be the time it takes to transfer the number of bytes accumulated since the request was made (8.1 μs), as well as T_T. So we have:

$$T_{TOT} = T_T + T_A$$

$$T_T = 16 \text{ Bytes} \left(\frac{1 \text{ Transfer}}{4 \text{ Bytes}}\right) 0.25\ \mu\text{s/Transfer} = 1.0\ \mu\text{s}.$$

$$\# = (8.1\ \mu\text{s} + 1.0\ \mu\text{s})/(0.8\ \mu\text{s/Byte})$$
$$= 11.375 \text{ Bytes Accumulated.}$$

8 bytes (two long words) will transfer with 3 bytes left in FIFO and 3 bits in serial/parallel converter. (The SONIC will transfer only long-word values to/from the FIFO).

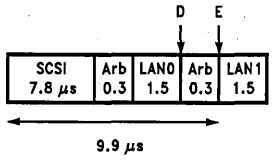
$$T_A = 8 \text{ Bytes} \left(\frac{1 \text{ Transfer}}{4 \text{ Bytes}}\right) 0.25\ \mu\text{s/Transfer} = 0.5\ \mu\text{s}.$$

$$T_{TOT} = 1.0\ \mu\text{s} + 0.5\ \mu\text{s} = 1.5\ \mu\text{s}.$$

Therefore the total transfer time for LAN0 is 1.5 μs. LAN0 will then request the bus again when it's FIFO threshold has been reached. Since there are 3 bytes left in FIFO and 3 bits in the serial/parallel converter,

$$T_{REQ} = (16 - 3 - 3/8 \text{ Bytes})\ (0.8\ \mu\text{s/Byte}) = 10.1\ \mu\text{s}.$$

So LAN0 will request the bus 10.1 μs later. It should be noticed that LAN0 (and LAN1 also) have a latency tolerance of 12.8 μs. This latency is more than adequate for the current latency of 8.1 μs.



TL/F/11141–2

FIGURE 3. Initial Latency for LAN1 Card

At point "D" LAN0 finished it's transfer and LAN1:3 and the ESDI controller arbitrate with LAN1 winning due to it's higher priority. Total bus occupancy for LAN1 will again be $T_{TOT} = T_T + T_A$.

T_T = 1.0 μs (because of the 16 byte transfer as calculated above).

$$\# = \frac{9.9\ \mu s + 1.0\ \mu s}{0.8\ \mu s/\text{Byte}} = 13.625\ \text{Bytes Accumulated.}$$

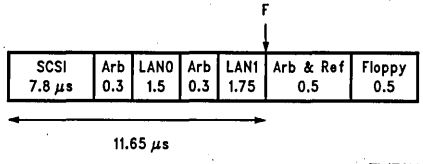
12 additional bytes (3 long words) will transfer with 1 byte remaining in the FIFO and 5 bits in serial/parallel converter.

$$T_A = 12\ \text{Bytes}\frac{0.25\ \mu s}{4\ \text{Bytes}} = 0.75\ \mu s$$

$$T_{TOT} = 1.0\ \mu s + 0.75\ \mu s = 1.75\ \mu s.$$

Therefore LAN1 will own the bus for 1.75 μs. Since LAN1's latency tolerance of 12.8 μs is greater than the current latency of 9.9 μs, it will be guaranteed access and no FIFO overruns will occur. LAN1 will then request the bus when it's FIFO threshold has again been reached. Since there is 1 byte left in the FIFO and 5 bits in the serial/parallel converter, the request time will be:

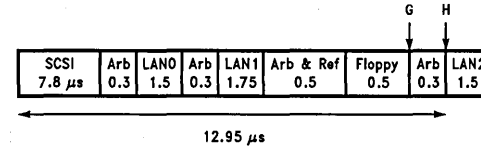
$$T_{REQ} = (16 - 1 - 5/8\ \text{Bytes})\ (0.8\ \mu s/\text{Byte}) = 11.5\ \mu s$$



TL/F/11141–3

FIGURE 4. Latency Till End of LAN1 Card Bus Occupancy Followed by Arbitration and Floppy Disk Access

At point "F" the SCSI controller, LAN0 and LAN1 have had their turn on the bus. At this point another arbitration will take place. Since the system needs to refresh memory, we will put in a refresh cycle now. This refresh will extend the arbitration by 200 ns, to a total of 500 ns. We also need to account for a floppy controller access. It is important for the floppy controller to gain access to the bus because if one of it's drives is a "floppy tape" and a byte was lost, the tape would have to stop, rewind, and re-read/write to that logical sector, taking a very bad performance hit. This situation needs to be prevented. We will assume that DMA channel 2 will win this arbitration and the floppy controller will transfer one byte, staying on the bus for approximately 500 ns. We now have:



TL/F/11141–4

FIGURE 5. Bus Latency Time for LAN2 Card

After the floppy access, LAN2:3 and the ESDI controller will arbitrate at point "G", with LAN2 winning and beginning to transfer at point "H". Since LAN2's latency tolerance is 19.2 μs and 12.95 μs is the current latency, there is 6.25 μs of margin left to guarantee proper access. How long will LAN2 stay on the bus?

$$T_{TOT} = T_T + T_A.$$

T_T = 0.5 μs (for any 8 Byte Transfer)

$\#$ = (12.95 μs + 0.5 μs) (1 Byte/0.8 μs)

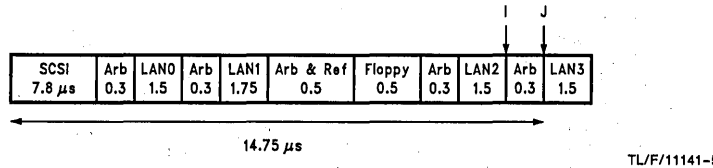
= 16.8125 Accumulated Bytes.

The SONIC will then transfer the additional 16 bytes (4 long words) that were accumulated in the FIFO and keep the remaining 6.5 bits in the serial/parallel converter.

T_A = 1.0 μs (from a previous calculation for a 16 byte transfer)

$$T_{TOT} = 0.5\ \mu s + 1.0\ \mu s = 1.5\ \mu s.$$

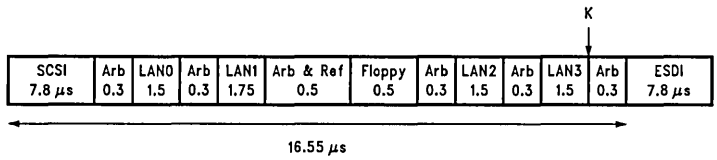
LAN2 will then re-arbitrate when it's FIFO has reached 8 bytes. This will be as shown in *Figure 6.*



TL/F/11141–5

FIGURE 6. Bus Latency Time for LAN3 Card

T_{REQ} = (8 − 6.5/8 Bytes) (0.8 μs/Bytes) = 5.75 μs later.



TL/F/11141–6

FIGURE 7. Total Bus Latency Time until Beginning of ESDI Drive Bus Access

At point "I" LAN2 is finished and LAN3 and the ESDI board will arbitrate with LAN3 winning. Since LAN3 has a latency tolerance of 19.2 μs and only 14.75 μs have occurred since LAN3 could have owned the bus, the latency margin of 4.45 μs is left over and a proper bus access has been guaranteed. LAN3 will then occupy the bus for:

$$T_{TOT} = T_T + T_A$$

T_T = 0.5 μs (from before for an 8 byte threshold)

= (14.75 μs + 0.5 μs) (1 Byte/0.8 μs)

= 19.0625 Accumulated Bytes.

The SONIC will transfer 16 bytes (4 long words) with 3 bytes remaining in the FIFO and 0.5 bits in the serial to parallel converter.

T_A = 1.0 μs for a 16 byte transfer so we have

T_{TOT} = 0.5 μs + 1.0 μs = 1.5 μs.

LAN3 will then arbitrate again when its FIFO threshold of 8 bytes has been reached. This will be:

$$T_{REQ} = \left(\frac{8 - 3 - 0.5}{8}\right) (0.8\ \mu s/Byte) = 3.95\ \mu s$$

So LAN3 will request the bus again in 3.95 μs. At this point we have the following sequence of events:

At point "K", the ESDI controller will arbitrate and win and will stay on the bus for 7.8 μs. After winning the bus, the ESDI controller will deassert PREEMPT*. The SCSI controller can now assert PREEMPT* (because fairness has been enabled for it) to request the bus again since it has still more data to transfer.

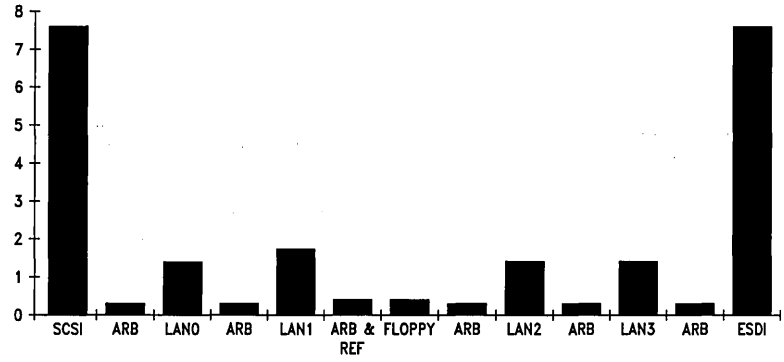
In all of the previous illustrations we showed all devices and their respective occupancy times and their relative sequence. The following graph visually shows how long all devices will own the bus relative to each other. It is quite apparent that due to the SONIC's and MCA's high speed DMA, the LAN controllers are on for a minimal amount of time. Streaming Mode MCA adapters would be on for half the time.

In this example we have taken a worst case scenario by assuming all the LAN boards and the ESDI board will request the bus simultaneously at the very beginning of the SCSI transfer period. We have shown that even in this situation all devices have accessed the MCA bus without error and with plenty of latency margin left over. Table IV summarizes these results.

TABLE IV. Accrued Latency

Device	Accrued System Latency (μs)	Device Latency Tolerance (μs)	Latency Margin (μs)
SCSI	0	(Note)	
LAN0	8.1	12.8	4.7
LAN1	9.9	12.8	2.9
REFRESH	11.65		
FLOPPY	12.15	(Note)	
LAN2	12.95	19.2	6.25
LAN3	14.75	19.2	4.45
ESDI	16.55	(Note)	

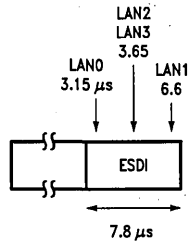
Note: These latencies are particular to the device in question.



TL/F/11141–8

FIGURE 8. Individual Bus Usage Times for All Bus Masters, and Arbitration Cycles

Since we are basing our calculation on this simultaneous request, what will happen when these LAN boards arbitrate again? Will this worst case scenario happen again? Based on our previous calculations the LAN boards will request again at different times. The following diagram shows when the LAN boards will arbitrate once more:



TL/F/11141–7

FIGURE 9

It can be seen now that starting with a worst case scenario as described above, the next set of LAN requests will be staggered apart throughout the ESDI transfer and our worst case scenario has all but disappeared, even after starting with it in the beginning. The LAN boards will still request and occupy the bus consecutively, however, they will now be on the bus for a shorter period of time. This is because the controller will get the bus sooner than in our worst case scenario; fewer bytes would have been accumulated in the FIFO since its threshold was reached hence a shorter transfer period. This means that other devices such as the CPU and mass storage controllers can have the bus sooner and occupy it longer than before. This equates to overall faster data throughput and more processing time for the CPU. It is up to the designer to determine when this worst case scenario will occur again, but it can be seen that the probabilities are exceptionally low that it will ever be repeated; however, if it did by properly setting arbitration and threshold levels and fairness options, the high performance of the SONIC can be readily achieved.

Since all devices have had a chance on the bus, what happens to the CPU during this worst case scenario? It has duties of its own such as protocol processing, updating descriptor lists, managing packets, etc. In the rare instance of this worst case scenario it wouldn't have immediate access to the bus. However, in nearly all the following accesses where the LAN accesses are staggered apart, there would be plenty of time for the CPU to access system memory.

One of the assumptions of this example is that no two consecutive transfers of 7.8 μs will occur in a row on the MCA bus when the LAN controllers are requesting it. The only way for this to happen was if there was a board which needed the bus immediately, and had a higher priority than the LAN boards and also would own the bus for a long period of time. However, a long bus occupancy time suggests a large buffer to hold all that data that is being transferred. A large buffer means it can tolerate longer latencies which means it can be set to a lower priority level, which effectively means this situation is avoided. Thus the LAN boards can effectively remain at the highest priority level and not be potentially locked out due to multiple, consecutive, 7.8 μs transfers, which won't happen.

Throughout this analysis a concern of the designer may be bus efficiency. Since the SONIC transfers just a few bytes at a time, it will request the bus often causing the arbitration time to be a significant portion of the transfer cycle. However, because of the Ethernet transfer rate of 1.25 MB/s these requests won't be often. When compared to the transfer times of the SCSI and ESDI boards, these arbitration times are not too significant (see Figure 8) and won't occupy much bus bandwidth. With these lower thresholds and bursty transfers, these inefficiencies become apparent. However, the SONIC more than compensates in other areas. The 20 MB/s transfer rate of the DMA allows for minimal time on the bus. With Streaming Mode Micro Channel, the bus occupancy can be further lowered by having a 40 MB/s data rate. By keeping the FIFO down to 32 bytes, the buffering of runt packets is eliminated. A larger FIFO may buffer many of these unwanted packets in a heavily loaded network and wastes valuable bandwidth. Also, the SONIC's buffer management structure has been designed for simplicity and performance. With much of the performance bottleneck happening in the upper protocol layers, a very fast and efficient driver becomes a necessity. The SONIC's register oriented buffer management scheme makes this possible. Upating descriptor lists is simple and doesn't take much processor overhead. It is very efficient. The on-board CAM can hold up to 16 different physical and multicast addresses. This allows supporting multiple protocols at the MAC level. By assigning a different physical address to each of the different protocols supported by the file server, protocol filtering can be done at a very low level, where it is much more efficient. To implement this without a CAM with a controller that supports only one physical address would necessitate it to enter promiscuous mode, meaning that it would have to buffer every packet on the network-this would be a very great waste of system bandwidth. Another way to improve efficiency would be to tie multiple SONICs together while maintaining a single MCA bus interface. The MREQ* and SMACK* pins on the SONIC allow it to be a slave to other devices, even other SONICs. By tieing multiple SONICs together, they could be time multiplexed into one MCA time slot; this would have the advantage of requiring only one arbitration cycle for multiple controllers. Not only would the efficiency go up but costs would come down as multiple SONICs would share just one bus interface. In short, the SONIC provides an optimal balance to achieve exceptional performance at all levels where system performance is measured.

32-Bit Bus Master Ethernet Interface for the 68030 (Using the Macintosh SE/30)

National Semiconductor
Application Note 691
William Harmon

OVERVIEW

National Semiconductor's SE/30 Ethernet adapter provides a high performance, 32-bit, bus master network connection for Apple's 68030 based compact Macintosh computer. This design is based around National Semiconductor's Systems Oriented Network Interface Controller (SONIC™, DP83932), which interfaces directly to the extension slot of the SE/30. The SE/30 design also serves as a model for designing the SONIC onto the mother board of a 68030 based system, since the SE/30's one expansion slot is essentially a direct connection to the Motorola 68030.

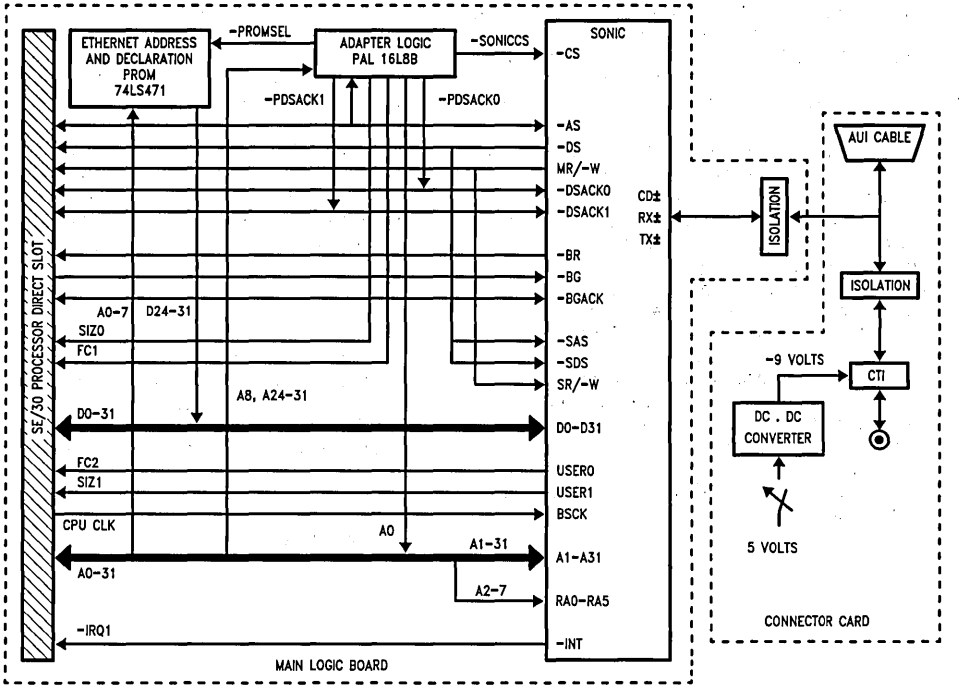
A block diagram of the adapter can be seen in *Figure 1*. The SE/30 Ethernet adapter operates synchronously with the 16 MHz SE/30 mother board and accesses the necessary transmit and receive buffers directly in the system's main memory, via 16 MHz 3 cycle asynchronous DMA operations. At this rate, the bus utilization for the buffering of a single packet is approximately 6% of the total bus bandwidth.

In addition to it's high performance DMA, the SONIC also has an on board Ethernet Manchester Encoder/Decoder (ENDEC), which allows the SONIC to communicate directly with any AUI interface. In fact, the SE/30 board has the capability to be connected to a network through either thin wire (10Base2) or AUI drop cable (10Base5) Ethernet.

It is also worth mentioning that the SE/30 adapter supports the use of Macintosh Nubus Slot Manager features, such as interrupt handling, with an on board Slot Manager PROM. This does not cause the board to incur any extra cost, since some type of PROM must already be used to store the adapter's Ethernet address.

FEATURES

- 32-bit bus master system interface
- Asynchronous high speed 3 cycle DMA
- 100% on card address filtering, via the SONIC's on board Content Addressable Memory (CAM)
- Minimal number of components
- Supports both AUI cable and thin wire Ethernet
- Optimal placement of receive and transmit data and descriptors in system memory
- Supports Macintosh Slot Manager
- Portable to 68030 mother board designs



TL/F/10848–1

FIGURE 1. Adapter Block Diagram

FUNCTIONAL OVERVIEW

System Interface

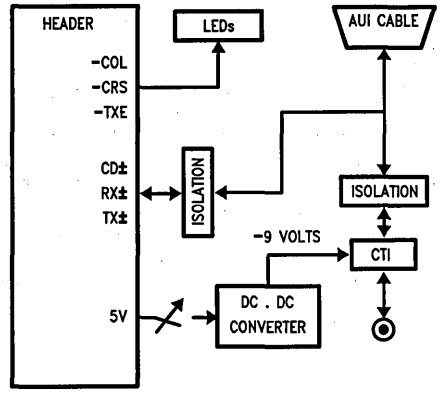
The Macintosh SE/30 provides a single 32-bit expansion slot, which basically consists of the control, data, and address signals of the mother board's CPU, the 68030. In addition to these signals, the expansion slot also provides Nubus compatible interrupt lines, so that Slot Manager software can be incorporated, if a Slot Manager PROM is placed on the card. Hence, the hardware interface between the SE/30 and the Ethernet adapter is essentially a 68030 bus interface, while the driver software interface is similar to that of a Macintosh II Nubus adapter. This solution is optimally achieved through the use of National Semiconductor's SONIC, whose 32-bit data and address buses and control signals interface directly to those of the Motorola 68030.

The SE/30 Ethernet adapter operates in both a slave and master mode. When operating in a bus master mode, the SE/30 Ethernet adapter arbitrates with the host system for control of the SE/30 bus and proceeds to operate as a 32-bit DMA engine between the system memory and the network. A block diagram of this interface can be found in *Figure 1*. The bus master mode of operation allows for the use of system memory, instead of on card RAM, for the buffering of transmit and receive data and their descriptors. Master operation is facilitated by the SONIC, which is at the heart of this adapter's design. The SONIC provides the complete implementation of the IEEE 802.3 specification from the AUI interface through the MAC layer, as well as performs a direct system interface to the 68030. In fact, when interfacing to the SE/30 backplane, the SONIC carries out 16 MHz 3 cycle asynchronous DMA, which is fully synchronous with the 16 MHz mother board of the SE/30. This enables the SE/30 adapter to operate on the bus in the same fashion as the 68030 and utilize only 6% of the bus bandwidth, during an Ethernet reception or transmission. The bus master design provides for the highest possible throughput between the system and the network, while at the same time requiring only a minimum of parts to implement.

When the adapter is a slave, the host system accesses either the Slot Manager PROM or the SONIC's registers. All slave operations are done via memory reads and writes, since both the PROM and the SONIC registers are mapped into system memory. The slave architecture is depicted in the adapter block diagram *(Figure 1)*. While in the slave mode, the SONIC once again provides a direct interface to the SE/30. The only necessary interface logic is the address decode for the SONIC chip select (-SONICCS). At this point, it is worth noting that the slave address strobe (-SAS) of the SONIC is connected to the data strobe (-DS) of the SE/30 instead of the SE/30's address strobe (-AS). This is due to the operation of the SE/30 backplane and will be further discussed in the design section of this document. However, it is important to remember that in interfacing directly to a 68030 CPU the SONIC's -SAS would be connected directly to the 68030's -AS.

Network Interface

With respect to the adapter's physical layer design, both AUI drop cable Ethernet and thin wire Ethernet are supported. The SE/30 adapter consists of two boards, the main logic board, which contains the SONIC, and the connector card which provides for the AUI and thin wire network connections. The connector card, whose block diagram is shown in *Figure 2*, contains a 15-pin AUI drop cable connector for standard drop cable Ethernet implementations, as well as a thin wire Ethernet connection via the National Semiconductor coaxial transceiver interface (CTI, DP8392).



TL/F/10848–2

FIGURE 2. Connector Card Block Diagram

Either of these network connections can be chosen through the use of a single jumper. In either case, the AUI signals (RX±, TX±, and CD±) are sent back to the main logic board, where the SONIC resides. These signals are interfaced to the ENDEC portion of the SONIC, which provides for communication between the AUI interface and the non-return to zero (NRZ) signals (RXD, TXD, and COL) of the Media Access Control (MAC) module of the SONIC. It should be noted that the integrated ENDEC module of the SONIC alleviates the need for an external Ethernet Manchester encoder/docoder, such as National's CMOS Serial Network Interface (CMOS SNI, DP83910).

BOARD ARCHITECTURE AND DESIGN

Memory Map

As stated previously, the SE/30 adapter is completely mapped into the addressable memory space of the SE/30. A diagram of the memory map can be found in *Figure 3*. The board is mapped into the memory locations F9FFFFFF through F9000000. Locations F9FFFFFF through F9FFFF00 contain the Ethernet address and declaration PROM. This region contains the adapter's Ethernet address as well as the declaration data that is necessary for the adapter's interrupt service routine to take advantage of the Slot Manager features, which are provided by the SE/30 operating system.

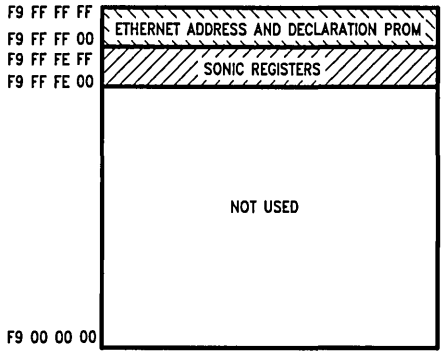


FIGURE 3. SE/30 Adapter's Memory Map

The SE/30 specification states that this declaration data begin at memory location F9FFFFFF, if the adapter's interrupt is generated on the system's IRQ 1 interrupt line. This is the case with the SE/30 adapter. The six byte Ethernet address immediately follows the Slot Manager software in the PROM. It should be noted that the data in the PROM is all byte addressable. In addition to the declaration information, the SONIC registers are also mapped into memory. These registers are mapped into locations F9FFFFE0 through F9FFFEFF. The SONIC's registers are mapped as 32-bit addressable quantities, in spite of the fact that internally all SONIC registers are only 16 bits wide. This is due to the fact that the SONIC will always respond as a 32-bit port, since it is programmed to operate in 32-bit mode.

Slave Operation

When operating as a slave, the Ethernet adapter appears as a block of memory to the host system. In slave mode, either the SONIC or PROM can be accessed. Timing diagrams for slave accesses appear in *Figures 4–6*. In the case of accessing the PROM *(Figure 4)*, the 68030 will issue a byte read command. The adapter logic will decode address lines 8 and 24 through 31 and recognize the fact that the SE/30 adapter's PROM is being selected. Once the 68030 asserts its address strobe, the logic will issue an enable signal to the PROM (-PROMSEL) and assert the cycle acknowledge signals (-PDSACK0 and -PDSACK1) back to the 68030, in order to indicate the adapter's acknowledgement of a byte access. In parallel with the logic's operation, the PROM will decode the address it is being given (A0–A7) and begin to source data after receiving the -PROMSEL signal. Finally, the 68030 will then finish the read cycle, at which point the adapter logic will then deassert -PROMSEL, -PDSACK0, and -PDSACK1.

As seen in *Figures 5* and *6*, slave accesses to the SONIC are completely compatible with the bus of the 68030 processor. The only deviation is the connection of -SAS to the 68030's -DS instead of -AS. This is due to the fact that during slave writes a glitch may occur on the memory read/write line (MR/-W) of the SE/30 backplane, while -AS is being asserted. This is fatal, since the SONIC latches the value of the slave read/write line (SR/-W, which is connected to MR/-W) with the falling edge of -SAS. The connection of the 68030's -DS to -SAS solves this problem. It should also be noted that the SONIC is mapped into memory as a 32-bit peripheral and will respond accordingly. However, only the lower 16 data lines (D0–D15) will be valid inputs and outputs during slave accesses.

A 32-bit mapping was selected, since the SONIC is programmed to operate in 32-bit mode, which causes the SONIC to respond with the acknowledge signals of a 32-bit port (-DSACK0 = 0 and -DSACK1 = 0). The only adapter logic necessary to facilitate this interface is the decode of address lines 8 and 24 through 31, along with the address strobe (-AS), to generate a chip select signal to the SONIC (-SONICCS). When accessing the SONIC registers, the 68030 will perform either a 32-bit read or a 32-bit write. Once -SONICCS is asserted the SONIC will respond with the acknowledge signals (-DSACK0 and -DSACK1) and appropriately source or sink data. The deassertion of -DS by the 68030 signals the end of the cycle and causes the SONIC to deassert -DSACK0 and -DSACK1 and terminate the slave cycle.

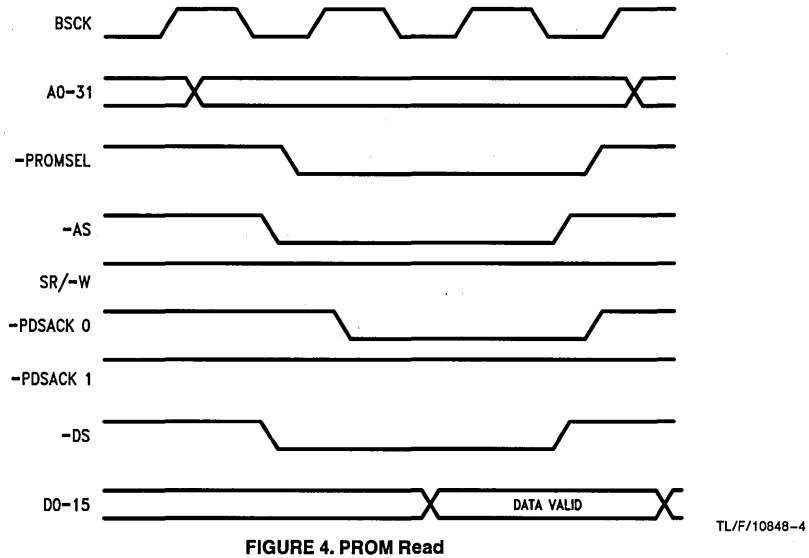


FIGURE 4. PROM Read

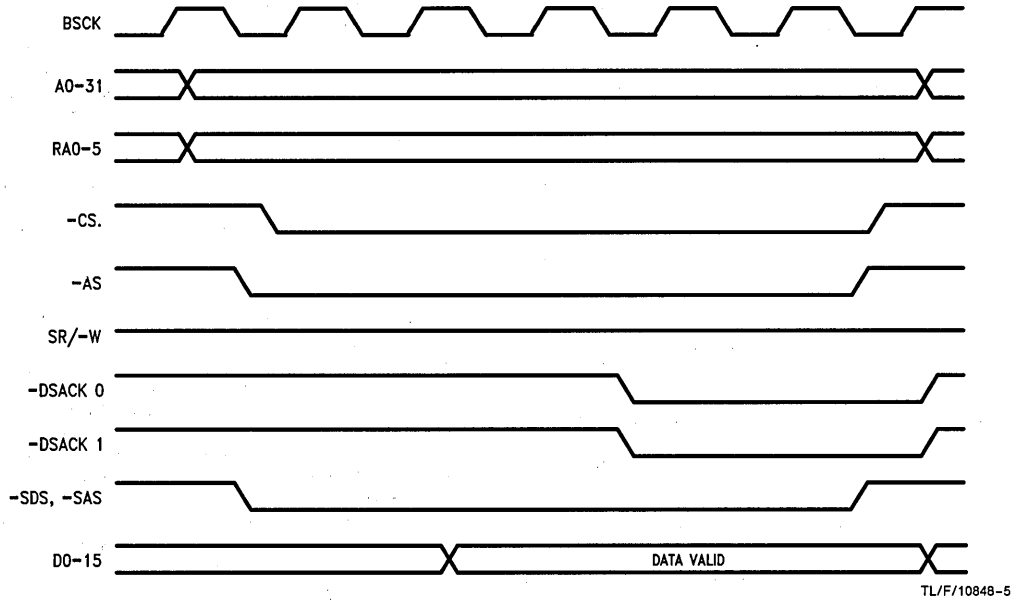


FIGURE 5. SONIC Slave Read

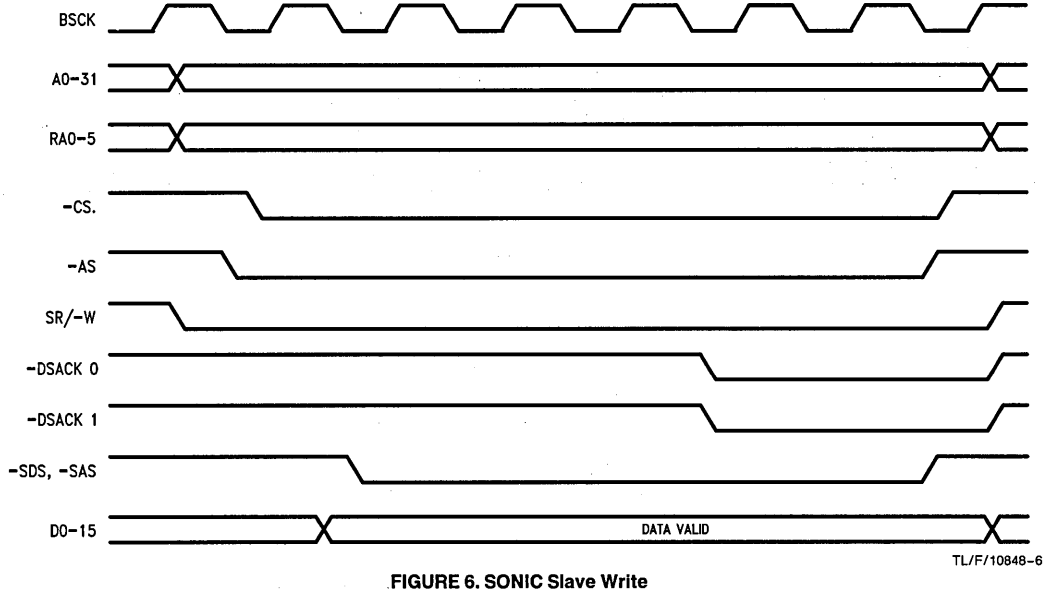


FIGURE 6. SONIC Slave Write

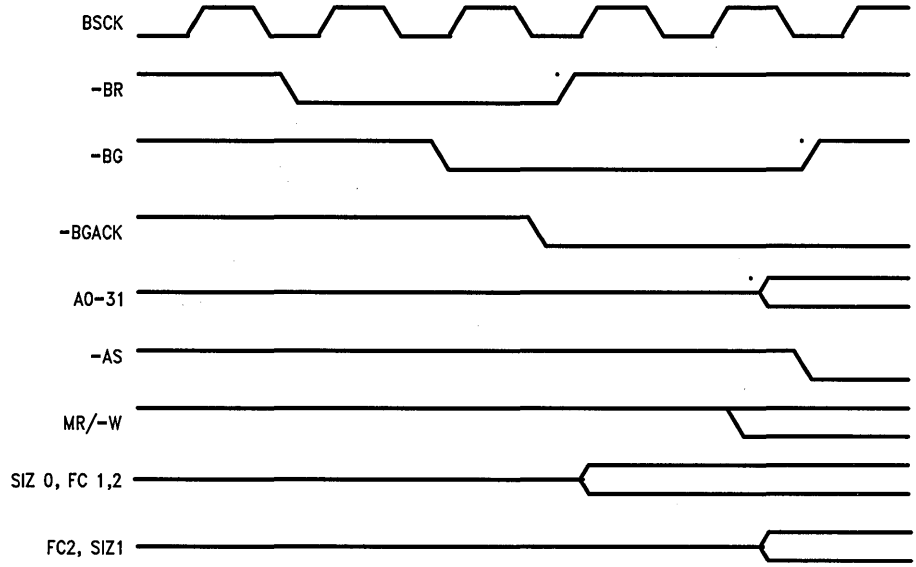
Master Operation

As stated previously, the SE/30 Ethernet adapter contains no local memory. All Ethernet data and descriptors are stored in system memory, which is accessed directly by the adapter. More specifically, the system memory is accessed directly by the SONIC, which interfaces directly to the 68030 mother board bus of the SE/30. When a master DMA access is required, the SE/30 adapter will arbitrate for the SE/30 system bus. This arbitration is performed by the SONIC, which connects directly to the bus arbitration signals of the 68030. This is depicted in the adapter block diagram *(Figure 1)*. A timing diagram of the arbitration cycle can be found in *Figure 7*. When the SONIC initiates a request for the system bus, it asserts the bus request signal (-BR) and waits for the bus grant signal (-BG) to be returned by the system. Once the bus grant signal is received, the SONIC will take ownership of the bus by asserting the open collector bus grant acknowledge signal (-BGACK), when the host system's -AS, -DSACK0, -DSACK1, and -BGACK are all deasserted. Once -BGACK is asserted, the SONIC removes the bus request signal.

After acquiring the bus the SE/30 adapter will begin to perform 16 MHz 3 cycle asynchronous DMA on the system bus. This function is also facilitated by the SONIC, whose direct 68030 interface allows the SE/30 adapter to operate on this bus with virtually no interface logic. As seen in *Figure 1*, the bus interface signals on the SONIC are attached directly to those of the SE/30 backplane.

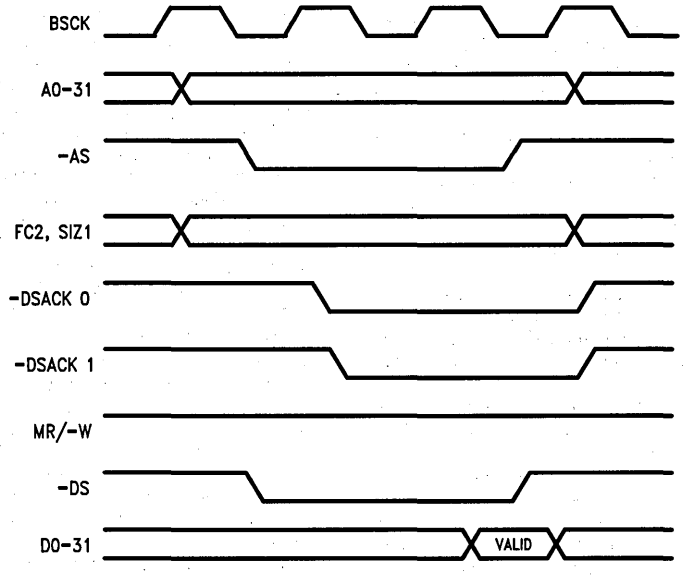
Timing diagrams of the adapter's master read and master write cycles appear in *Figures 8* and *9*. The only external interface logic required is for the generation of Function Code bit 1 (FC1), SIZ0, and address line 0 (A0). These lines are not provided directly by the SONIC, but are formulated in the adapter logic. All three signals are driven low upon the SONIC's assertion of -BGACK. It should also be noted that Function Code bits 0 and 2 (FC0 and FC2) and the SIZ1 signal are also not provided directly by the SONIC. However, these signals require no extra logic. The FC0 signal is tied high through a backplane resistor and requires no board connection, since the SONIC should only access memory areas which correspond to function codes with the least significant bit set high. These areas are user data space (FC2, FC1, FC0 = 001) and supervisor data space (FC2, FC1, FC0 = 101).

The FC2 and SIZ1 signals are not defined on the SONIC, but they can be generated by using the user 0 and user 1 pins. The user 0 and 1 outputs can be programmed by the programmable output bits (PO0 and PO1) in the SONIC's data configuration register (DCR). The output timing for these signals corresponds to the timing for the SONIC's address lines, which is the correct timing for both FC2 and SIZ1. By programming PO0 with a 0 or 1, the adapter can be made to access either the user data space or supervisor data space of the SE/30's system memory. In order to provide the correct SIZ1 signal for 32-bit operation the PO1 bit should be programmed to a 0.



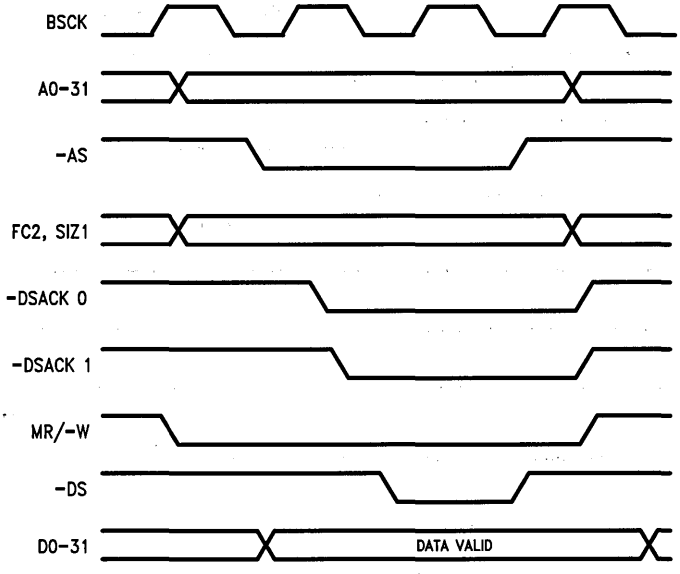
TL/F/10848–7

FIGURE 7. SONIC Master Arbitration



TL/F/10848–8

FIGURE 8. Master Read



TL/F/10848–9

FIGURE 9. Master Write

Physical Layer

The physical layer interface for the SE/30 adapter resides on the connector card, which attaches to the back of the SE/30. The connector board is linked to the main logic board through a ribbon cable that attaches to a 20-pin header on the main logic board (J2) and another 20-pin header on the connector board (J3). These two headers can be seen on the adapter schematics, which appear at the end of the manual of this application note. The adapter can be used in either a thin wire or standard drop cable Ethernet environment. When the adapter is used in a thin wire Ethernet application, jumper 3 (JB3) must have the jumper covering both posts. This enables the DC-to-DC converter to receive a 5V input from the SE/30 backplane and convert this to a −9V output, which is required by National Semiconductor's Coaxial Transceiver Interface (CTI, DP8392). The CTI provides an interface between the 10 MHz Manchester encoded coax cable and the 10 MHz Manchester encoded differential signals of the SONIC's ENDEC. In the case of a standard drop cable Ethernet application, JB3 is left uncovered so that the CTI will not receive power. This allows the signals of the SONIC's ENDEC to pass directly to the AUI cable, via the 15-pin AUI connector. In examining the schematic of the physical layer design, it can be seen that there is a pulse transformer at the AUI side of the CTI. This is placed here to isolate the CTI from the SONIC's ENDEC signals, when the AUI drop cable connection is being employed. This transformer also provides the IEEE 802.3 specified isolation between the coax and the differential AUI signals, when thin wire Ethernet is being used. It is also necessary to provide a termination for the 78Ω AUI cable's differential receive and collision pair (RX± and CD±). This is the reason for the 39Ω -1% resistors and 0.01 μF capacitors that are shown in the schematic.

Since the ENDEC resides within the SONIC, two components of the physical layer design are located on the main logic board, which can be seen on the schematics. First, each one of the transmit pairs (TX+ and TX−) requires a 270Ω non-precision pull down resistor (R1 and R2) to complete the internal source follower amplifiers that drive these signals. Second, there is an isolation transformer (T1) placed between the differential signals of the SONIC's ENDEC and the header for the ribbon cable. This isolation is necessary to guarantee that the SONIC meets the IEEE 802.3 fail safe specification of a 16V DC level appearing on the AUI cable's differential signals. The external isolation is necessary, due to the fact that in the powered down state the CMOS process, in which the SONIC is manufactured, may not be able to withstand this voltage.

The final feature of the physical layer design is the diagnostic LEDs. The yellow LED indicates that the ENDEC carrier sense signal (CRS) is asserted. An inverted version of CRS drives this LED. The green LED indicates that a transmission is in progress, and the red LED indicates the presence of a collision. The transmission LED and collision LED are driven by inversions of the SONIC's transmit enable (TXE) and collision output (COL) signals, respectively. The signals for the LEDs are supplied from the main logic board, via the ribbon cable that connects the two boards.

ADAPTER LOGIC EQUATIONS

The following is the set of logic equations that are necessary to implement the adapter logic block found in *Figure 1.* As shown in the schematics, this logic can be implemented in a single 16L8B PAL.

Inputs

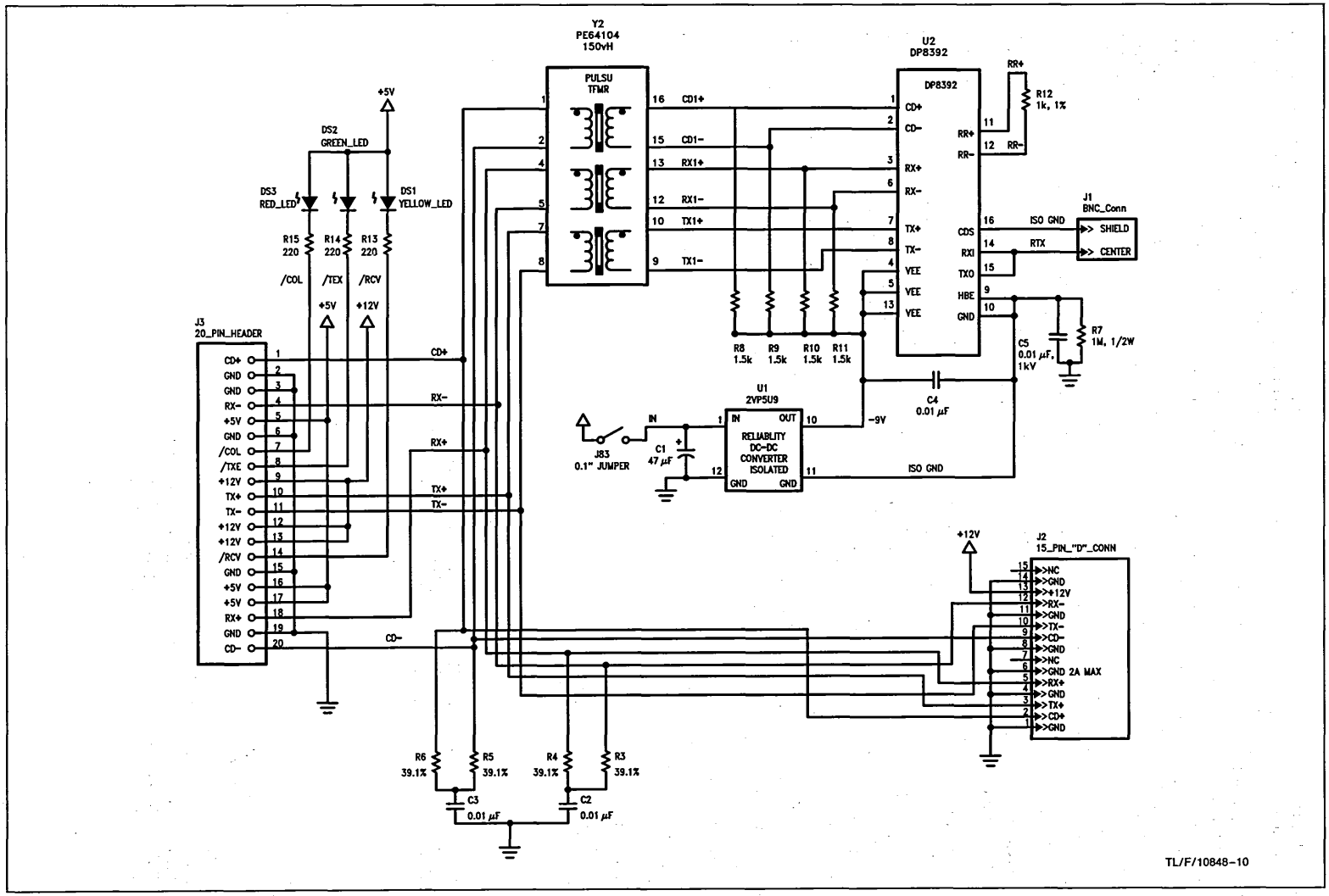
A31, A30, A29	Pin	1, 2, 3
A28, A27, A26	Pin	4, 5, 6
A25, A24, A8	Pin	7, 8, 9
AS, BGACK	Pin	11, 13

Outputs

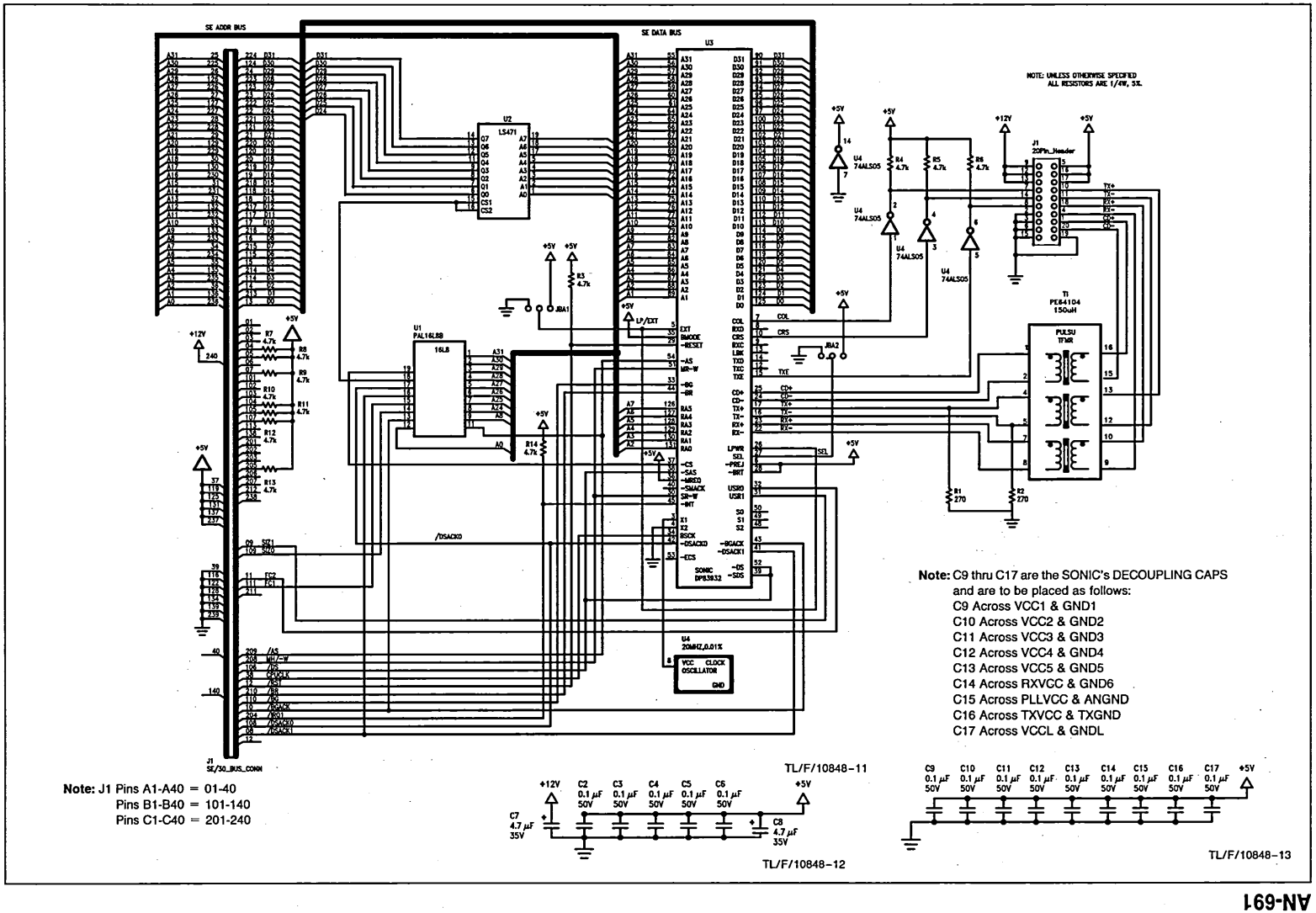
A0	Pin	12; Address line 0 (TRI-STATE)
SIZ0	Pin	14; 68030 SIZ0 signal (TRI-STATE)
FC1	Pin	15; 68030 Function Code 1 signal (TRI-STATE)
PDSACK1	Pin	16; PROM cycle acknowledgement (TRI-STATE)
PDSACK0	Pin	17; PROM cycle acknowledgement (TRI-STATE)
-PROMSEL	Pin	18; PROM chip select
-SONICCS	Pin	19; SONIC chip select

Equation:

```
A0 = 0
SIZ0 = 0
FC1 = 0
PDSACK0 = 0
PDSACK1 = 1
ENABLE A0 = -BGACK
ENABLE SIZ0 = -BGACK
ENABLE FC1 = -BGACK
ENABLE PDSACK0 = A31*A30*A29*A28*A27*-A26*-A25*A24*A8*-AS
ENABLE PDSACK1 = A31*A30*A29*A28*A27*-A26*-A25*A24*A8*-AS
SONICCS = A31*A30*A29*A28*A27*-A26*-A25*A24*-A8*-AS
PROMSEL = A31*A30*A29*A28*A27*-A26*-A25*A24*A8*-AS
```


Y2
PE64104
150vH
PULSU
TFMR
U2
DP8392
DP8392
+5V
DS2
GREEN_LED
DS3
RED_LED
DS1
YELLOW_LED
R15
220
R14
220
R13
220
/COL
/TEX
/RCV
+5V
+12V
J3
20_PIN_HEADER
CD+
GND
GND
RX-
+5V
GND
/COL
/TXE
+12V
TX+
TX-
+12V
+12V
/RCV
GND
+5V
+5V
RX+
GND
CD-
CD+
RX-
RX+
TX+
TX-
CD-
CD1+
CD1-
RX1+
RX1-
TX1+
TX1-
RR+
R12
1k, 1%
RR-
J1
BNC_Conn
ISO GND
SHIELD
RTX
CENTER
R8
1.5k
R9
1.5k
R10
1.5k
R11
1.5k
C5
0.01 µF,
1kV
R7
1M, 1/2W
C4
0.01 µF
U1
2VP5U9
IN
OUT
RELIABILITY
DC-DC
CONVERTER
ISOLATED
GND
GND
-9V
ISO GND
J83
0.1" JUMPER
C1
47 µF
+12V
J2
15_PIN_"D"_CONN
NC
GND
+12V
RX-
GND
TX-
CD-
GND
NC
GND 2A MAX
RX+
GND
TX+
CD+
GND
R6
39.1%
R5
39.1%
R4
39.1%
R3
39.1%
C3
0.01 µF
C2
0.01 µF

TL/F/10848–10



Note: C9 thru C17 are the SONIC's DECOUPLING CAPS and are to be placed as follows:

C9 Across VCC1 & GND1
C10 Across VCC2 & GND2
C11 Across VCC3 & GND3
C12 Across VCC4 & GND4
C13 Across VCC5 & GND5
C14 Across RXVCC & GND6
C15 Across PLLVCC & ANGND
C16 Across TXVCC & TXGND
C17 Across VCCL & GNDL

Note: J1 Pins A1-A40 = 01-40
Pins B1-B40 = 101-140
Pins C1-C40 = 201-240

TL/F/10848–12

TL/F/10848–13

TL/F/10848–11

DP839EB-MCS SONIC™ MICROCHANNEL® Ethernet Adapter

National Semiconductor
Application Note 732
Wesley Lee
Richard Bowers

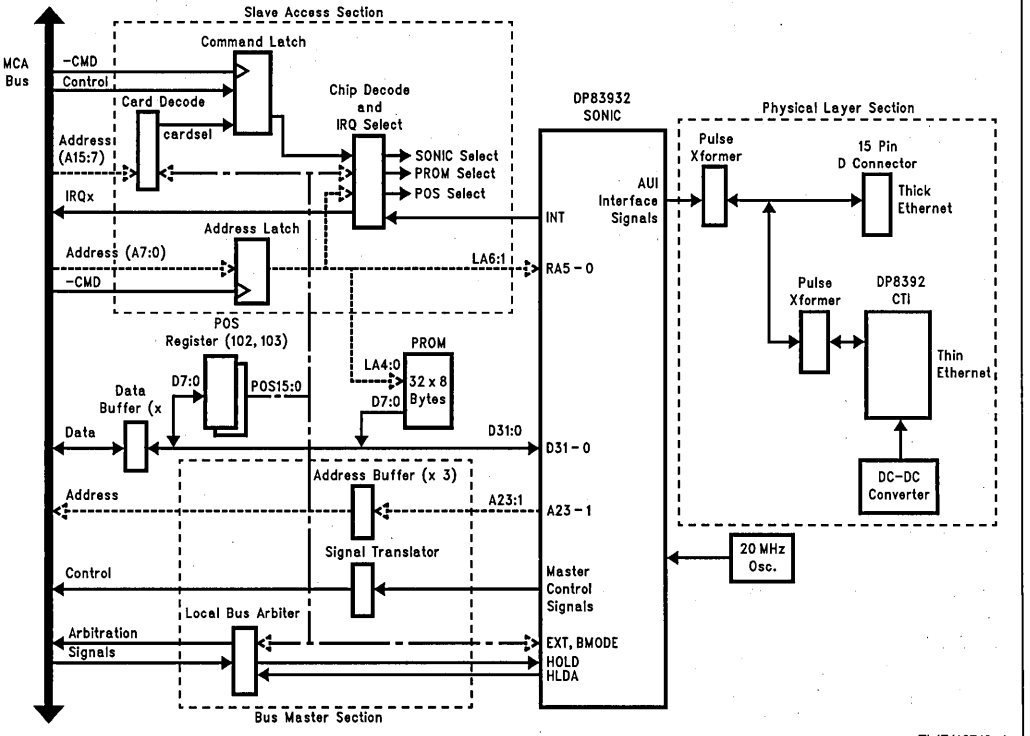
GENERAL DESCRIPTION

The Microchannel Evaluation board is a high-performance Ethernet Adapter card designed to provide significant throughput increases over other Ethernet adapter cards currently available. This board employs a bus master architecture for directly storing/retrieving data into system memory; thus, eliminating the need for intermediate packet copying. Using the DP83932's high-speed DMA capabilities, this board achieves a 16 Mbytes/sec transfer rate across the bus, utilizing a 32-bit data and 24-bit address path. The board also features extensive evaluation options to choose between Ethernet/Thin-Ethernet, 16/32-bit data path, and internal/external Ethernet ENDEC. The Microchannel Ethernet Adapter couples National's DP83932 Systems-Oriented Network Interface Controller (SONIC) with the DP8392 Coaxial Transceiver Interface (CTI) to form a simple two chipset solution for IEEE 802.3 networks.

FEATURES

- Utilizes the DP83932 and DP8392 chipset
- 16/32-bit data path
- 16 Mbytes/sec DMA throughput
- Extensive SONIC evaluation options
- 14 selectable I/O address ranges
- Compatible with PS/2® models 50, 60, 70, 80
- 4 selectable interrupts
- Bus master/burst capability
- Ethernet/thin-ethernet selectable

BLOCK DIAGRAM



TL/F/10748–1

1.0 BOARD DESCRIPTION

The Microchannel Ethernet Adapter is a high-performance, 16/32-bit busmaster board designed to demonstrate the advanced features of the DP83932. It consists of three main sections: (1) the physical layer, (2) the slave access section, and (3) the bus master section. The Physical Layer section consists of the internal ENDEC (encoder/decoder) of the SONIC, the coaxial transceiver (DP8392) and isolation transformers and is responsible for driving and receiving the IEEE 802.3 networks signals. The Slave Access section consists of the address decode circuitry necessary to access the SONIC's internal registers, the POS registers, and the PROM. Finally, the Bus Master section composes of the interface logic which permits the SONIC to gain access of the Microchannel bus.

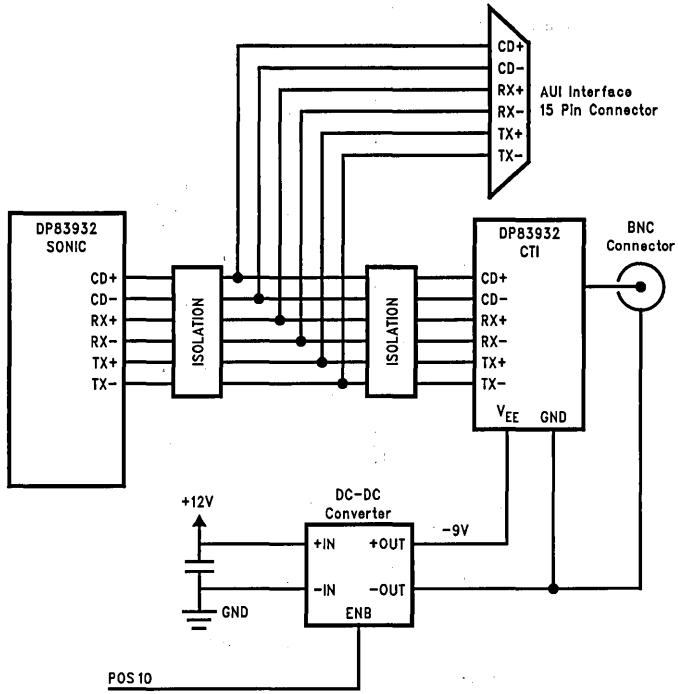
2.0 PHYSICAL LAYER SECTION

The physical layer section drives and encodes data onto the network during transmission, and decodes the data during reception. The Ethernet Adapter uses the on-chip ENDEC (encoder/decoder) from the SONIC and the coaxial transceiver, the DP8392, for these purposes. This physical layer section is illustrated in *Figure 1*.

The Ethernet Adapter provides connections to both thick-wire and thin-wire Ethernet. The thick-wire (AUI) connection is made via the TX±, RX±, and CD± signals from the SONIC, an external pulse transformer and a 15-pin D connector. The external pulse transfer is required to meet the IEEE 802.3 high voltage (16V) specification at AUI interface. The thin-wire connection is made via another external pulse transformer and the on-board coaxial transceiver, the DP8392. This external pulse transformer is necessary to completely isolate the TX±, RX±, and CD± pins when the DP8392 is powered down.

2.1 Switching between Thin or Thick-Wire Ethernet

To swap between thin and thick-wire Ethernet, POS 10-bit turns on/off the DC-DC converter. In the thin-wire configuration, the DC-DC converter is turned on, powering up the DP8392 to receive/transmit data onto the thin-wire cable. In the thick-wire configuration, the DC-DC converter is turned off, forcing the DP8392 to shut down. The pulse trans-



TL/F/10748–2

FIGURE 1. Simplified Physical Layer Section

former B isolates the TX±, RX± and CD± signals of the DP8392 and allows data to pass through the 15-pin D connector unloaded by the DP8392.

3.0 SLAVE ACCESS SECTION

During initialization and status updates, the SONIC register may be accessed to provide configuration and status information. The CPU accesses these registers by driving the proper address on the register address lines (RA5–RA0) and chip selecting the SONIC. The SONIC responds with -RDYO (ready out) when its registers are available for accessing. The CPU may also access the ID PROM and POS registers to respectively read the board's Ethernet Physical Address and configuration information. The system accesses the SONIC, PROM, and POS register via the Address Decode Logic described below. (The slave access section is shown on page 1 of the schematic.)

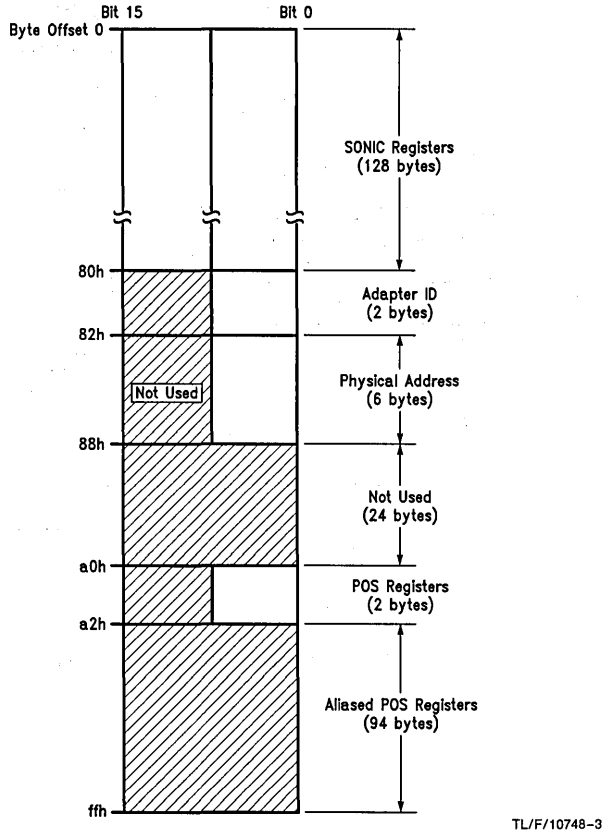
3.1 Address Decode Logic

The Address Decode Logic provides the decoding for the SONIC, POS registers, and PROM. This decoding is user programmable with the Address Select bits (ADDR0–3) from POS register 102. The Decode Logic decodes these bits along with the address and control signals from the Microchannel bus to determine the selected address base (see POS register 102 description in Section 5.0). The decoding is a two step process. First, the unlatched address and control signals are decoded to determine if the card is selected; then further decoding is performed to select the proper component on the board. Because the address and control signal on Microchannel bus are not valid for the duration of the bus cycle, these signals must be latched.

The Address Decode Logic, implemented with two PALs and two latches, operates in the following manner. The CARD DECODE PAL first decodes the unlatched address (A15–A7) and control signal (M/-IO) from the Microchannel bus along with the POS address select bits (ADDR0–3) to generate the CARDSEL signal. This signal is then latched and further decoding is performed by the CHIP DECODE PAL. This PAL decodes the latched -LCARDSEL signal, the latched lower address LA0–2, LA5–7 and the latched control signals (-LS0, -LS1, and -LCDSETUP) to provide chip selects the SONIC, POS registers, or the PROM. The address and control signals are latched on the leading edge of -CMD with a ACT573 transparent latch.

3.2 I/O ADDRESS MAP

The Microchannel Ethernet Adapter's address base is specified by bits ADDR3–0 in POS register 102. The board occupies 256 bytes as shown in *Figure 2*.



TL/F/10748–3

FIGURE 2. Ethernet Adapter's I/O Address Map

4.0 BUS MASTER SECTION

The Microchannel evaluation board employs a bus master architecture to quickly transfer data from/to system memory to/from the internal FIFOs of the DP83932. The DP83932's FIFOs are sufficiently large (32 bytes) to absorb all reasonable bus latencies which may occur on the Microchannel bus. When accessing the bus, the evaluation board follows the required Microchannel protocol using the arbitration level programmed in the POS 103 register. The bus master logic is partitioned into two sections, the Local Bus Arbiter which implements the microchannel bus access protocol and the Signal Translator which interfaces the SONIC control signals to the Microchannel bus. (The bus master section is shown on page 2 of the schematic.)

4.1 Local Bus Arbiter

The local bus arbiter allows the Ethernet Adapter to compete for the Microchannel bus when an arbitration cycle is in progress. The arbitration cycle begins after the Ethernet Adapter asserts -PREMPT and the ARB/-GNT signal has subsequently gone high. The Ethernet adapter participates in the arbitration cycle by gating its arbitration vector onto the bus and simultaneously comparing it with all other vectors appearing on the bus. If the Local Arbiter detects an arbitration vector lower than its own, it removes its vector and waits for the next arbitration cycle; otherwise, if the Ethernet Adapter has won, it may begin transferring data onto the bus.

The Local Arbiter is implemented with two PALs, ARBVEC, and ARBMAC (support logic is also in the Logic PAL). The ARBVEC PAL implements the arbitration vector function which drives and simultaneously reads the vector on lines ARB3–ARB0 when an arbitration cycle is in progress. This PAL indicates whether it has won the bus by asserting -BUSWIN low. The arbitration vector is user programmable via bits SELARB0–3 of POS register 103. The second PAL, ARBMAC, controls the enabling of the arbitration vector. This PAL consists of an asynchronous state machine to monitor bus activity from the Microchannel bus and the SONIC. This state machine shown in *Figure 3* contains the following states.

IDLE: No bus activity by the SONIC; HOLD request is not asserted

REQ: The SONIC is requesting usage of the bus but another device may be using the bus; -PREMPT is asserted

ARB: An arbitration cycle is occurring on the bus. The arbitration vector is enabled; -PREMPT is still asserted.

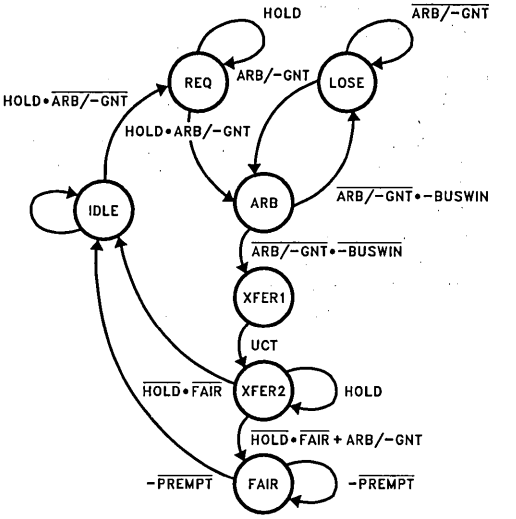
LOSE: The SONIC has lost the arbitration cycle. It degates its vector and waits for the next cycle. -PREMPT is still asserted.

XFER1: The SONIC has won; detects -BUSWIN low from ARBVEC PAL. An intermediate state to XFER2. (This state is needed so that only one output changes between states.)

PREMPT is still asserted.

XFER2: The SONIC performs its data transfer. -BURST is asserted and -HLDA is issued to the SONIC. -PREMPT is deasserted.

FAIR: The fairness algorithm has been enabled.



TL/F/10748–4

Note: UCT = Unconditional Transfer

FIGURE 3. Local Bus Arbiter State Machine (ARBMAC)

The Local Arbiter state machine defaults to IDLE when the SONIC is idle. When HOLD is asserted, the state machine transitions to REQ and then to ARB when ARB/-GNT goes high. During this state, ARBMAC enables ARBVEC to begin driving its vector onto lines ARB3–0 and monitors -BUSWIN when ARB/-GNT has subsequently gone low. If -BUSWIN is high, the Ethernet adapter has lost and the state machine proceeds to LOSE and waits for the next arbitration cycle. Otherwise, the Ethernet adapter has won and the state machine proceeds to XFER2 via XFER1. During XFER2, Hold Acknowledge (HLDA) is issued to the SONIC, allowing it to transfer data onto the bus. (For robust asynchronous state machine design, XFER1 is used to insure that only one output changes between states.) When the SONIC finishes its block transfer, the state machine finally transitions either to FAIR or IDLE, depending upon the FAIR bit in the POS 103 register.

The state machine obeys the fairness algorithm when the FAIR bit in POS register 103 has been set. This algorithm insures that all competing devices will eventually gain access to the bus. If the FAIR bit is set, the state machine will stay in FAIR until all other devices have completed their bus transfers (i.e., -PREMPT is no longer asserted) before returning to IDLE.

4.2 Signal Translator

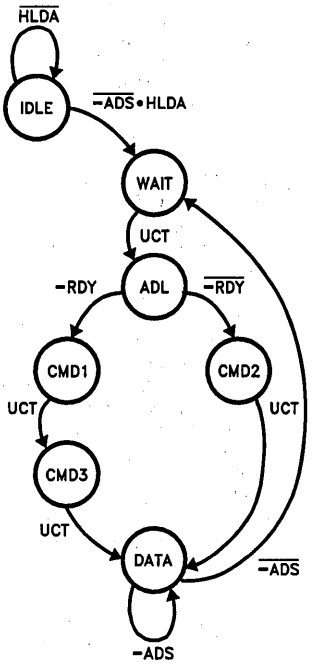
The Signal Translator converts the control signals from the SONIC into the signals required by the Microchannel bus and matches the timing of these signals to be consistent with the 3 primary modes of the Microchannel specification, *Default* mode, *Synchronous Extended* mode, and *Asynchronous Extended* mode. The Signal Translator matches the

timing by using a synchronous state machine to convert the SONIC's control signals, MW-R, -ADS, and -DS to -S0, -S1, -ADL, and -CMD of the Microchannel bus. Note that since the SONIC is capable of operating considerably faster than the Microchannel bus, *it is required that the SONIC be programmed in asynchronous mode and inserts 3 wait-states for each memory cycle.* Configuring asynchronous mode and the 3 wait-states are programmed by resetting the STERM bit and setting bits WC1 and WC0 and in the Data Configuration register. The state machine, shown in *Figure 4*, contains the following states.

IDLE: No bus activity by the SONIC.

WAIT: SONIC has asserted -ADS; wait for one bus clock to provide address setup time to leading edge of -ADL.

ADL: Assert -ADL for one clock cycle.



Note: UCT = Unconditional Transfer TL/F/10748–5

Note: The state machine is reset when HLDA is deasserted.

FIGURE 4. Signal Translator State Machine

CMD1: *Synchronous* or *Asynchronous Extended* mode requested. CDCHRDY has been deasserted by memory. Begin asserting -CMD and SYNWAIT.

CMD2: *Default* mode requested. Begin asserting CMD.

CMD3: Continuing to assert -CMD and SYNWAIT.

DATA: SONIC finishes up memory transfer. -CMD is deasserted when -DS goes low. Transition back to IDLE when -ADS goes low.

The Signal Translator may take one of two paths during a SONIC bus operation. In the first path, the Signal Translator goes through states, WAIT, ADL, CMD2, and finally DATA. This path is taken if the memory does not require any wait-states and allows the SONIC to operate as fast as permitted on the Microchannel bus (*Default* mode, minimum cycle time = 200 ns). To satisfy all the timing requirements of Microchannel, the SONIC must use 5 bus clocks at 20 MHz for the memory cycle. This path is illustrated in *Figure 5*. The second path goes through states: WAIT, ADL, CMD1, CMD3, and DATA. This path gives at least an additional 100 ns to the memory cycle for compatibility with the *Synchronous Extended* mode (300 ns) or the *Asynchronous Extended* mode (≥300 ns).

The memory may deassert CDCHRDY in two ways. In the first manner, the memory requests a *Synchronous Extended* mode by pulsing CDCHRDY within 60 ns after the address goes valid then driving it active after -CMD has subsequently gone low. The timing is illustrated in *Figure 6*. (Note that the Ethernet adapter accesses the ready signal via the return signal, CHRDYRTN.) The additional 100 ns is added by deasserting the RDYi input of the SONIC during CMD1 and CMD3. The RDYi input is deasserted by NANDing (in the IRQSEL PAL) the SYNWAIT output generated by the Signal Translator with CHRDYRTN. Note that in asynchronous mode, the SONIC terminates the memory cycle 1 bus after clock after -RDYi is asserted.) In the second manner, the memory requests the *Asynchronous* mode by deasserting CDCHRDY as before, but does not assert CDCHRDY until it is ready to be accessed.

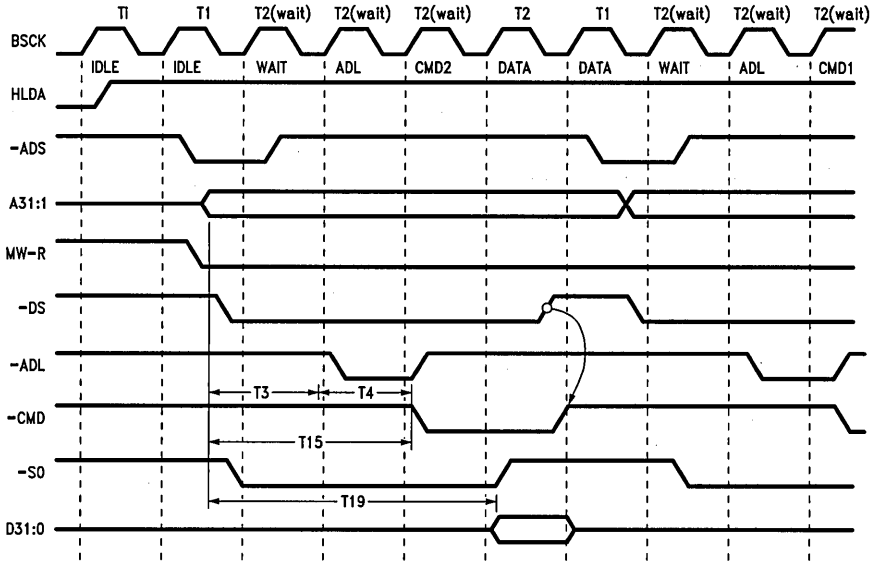
In addition to the signals generated by the Signal Translator (-S0, S1, -ADL, -CMD), there are 8 other signals which must be generated each time the SONIC gains access to the bus. These signals are MADE24, M/-IO, -SHBE, TR32, and BE0–3. The first 4 signals are enabled whenever the SONIC is bus master, and the latter 4 are enabled when the SONIC is configured for 32-bit data mode. Since these signals remain constant for the duration of the SONIC's transfer cycle, they are driven to their proper levels using a ACT244.

4.2.1 SIGNAL TRANSLATOR TIMING

Figures 5, 6, and *7* show the timing generated by the Signal translator for Default Mode (minimum cycle time = 200 ns), Synchronous Extended Mode (minimum cycle time = 300 ns) and Asynchronous mode (cycle time ≥ 300 ns). States Ti, T1 and T2 indicate the DMA states of the SONIC while states IDLE, WAIT, ADL, CMD1, CMD2, CMD3, and DATA indicate the corresponding states of the Signal Translator. Note that the SONIC must be configured in asynchronous mode and be inserting 3 wait-states. Also note that CHRDYRTN, shown in *Figures 6* and *7,* is the ready return signal provided by the Microchannel bus.

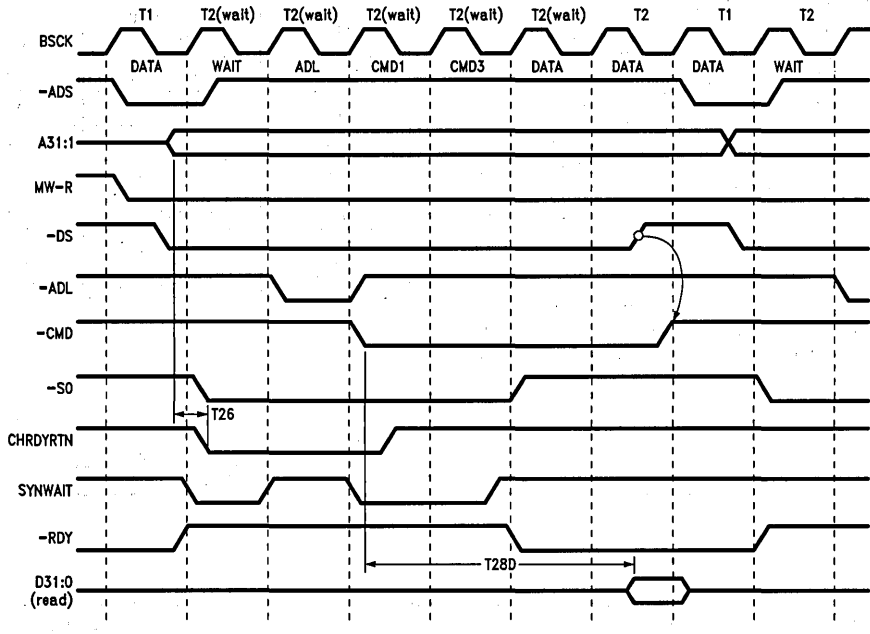
The timing parameters, shown as "T#," indicate the critical timing constraints which the Signal Translator and the SONIC must meet in order to be compatible with the Microchannel bus. These parameters and the corresponding Ethernet adapter parameters are tabulated below.

Parameter	Microchannel Spec.	Ethernet Adapter
T3	45 ns (min)	75 ns (min)
T4	40 ns (min)	50 ns (min)
T15	85 ns (min)	125 ns (min)
T19	125 ns (max)	200 ns (max)
T26	60 ns (max)	100 ns (max)
T28D	160 ns (max)	175 ns (max)
T29S	60 ns (max)	60 ns (max)



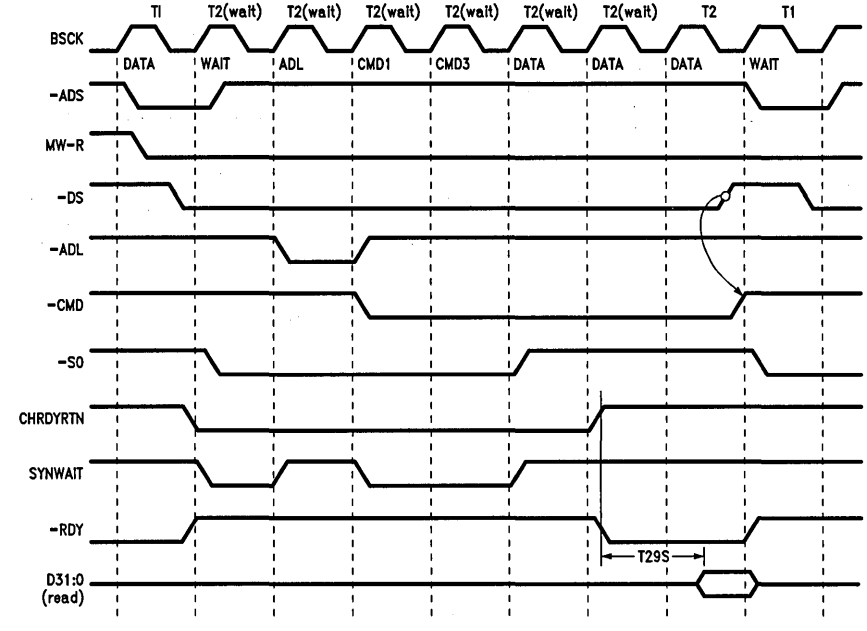
TL/F/10748–6

FIGURE 5. Default Mode, Memory Read



TL/F/10748–7

FIGURE 6. Synchronous Extended Mode, Memory Read



TL/F/10748–8

FIGURE 7. Asynchronous Extended Mode, Memory Read

5.0 POS REGISTERS

The POS registers provide the required identification (ID) bytes (2) and configuration information needed by the Ethernet Adapter. The Microchannel bus during power-on reads the ID bytes, located at addresses 100 and 101, and compares these bytes with values saved in battery back-up RAM. If the comparison is true, it will proceed to write the configuration information stored in RAM into the remaining POS registers residing at addresses 102 to 103. The configuration information, guarantees that no I/O, memory, or interrupts conflict with other boards using the Microchannel bus. For this implementation, only two POS registers are needed to provide the required configuration parameters. The following description gives the bit definitions for POS registers 102 and 103.

POS 102 BIT DEFINITIONS

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
ADDR3	ADDR2	ADDR1	ADDR0	res	INT1	INT0	CARDEN

Bit	Name	Function
7–4	ADDR3–0	**Address Select:** These bits select which base address the Ethernet Adapter Card will reside. The address selections are shown below: **ADDR3–0** — **I/O Address** 1111 — Not Used 1110 — Not Used 1101 — 1e00–1effh 1100 — 1c00–1cffh 1011 — 1a00–1affh 1010 — 1800–18ffh 1001 — 1600–16ffh 1000 — 1400–14ffh 0111 — 1200–12ffh 0110 — 1000–10ffh 0101 — 0e00–0effh 0100 — 0c00–0cffh 0011 — 0a00–0affh 0010 — 0800–08ffh 0001 — 0600–06ffh 0000 — 0400–04ffh
3	res	**Reserved**
2–1	INT1,0	**Interrupt Select:** These bits select the Interrupt Request lines: The selections are shown below: **INT1,0** — **IRQ line** 11 — IRQ9 10 — IRQ7 01 — IRQ6 00 — IRQ3
0	CARDEN	**Card Enable:** When this bit is set to a "0", the card is disabled and responds only to the setup read and write commands. When set to a "1" the card is enabled.

POS 103 BIT DEFINITIONS

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
ARB3	ARB2	ARB1	ARB0	FAIR	E/T	EXT	res

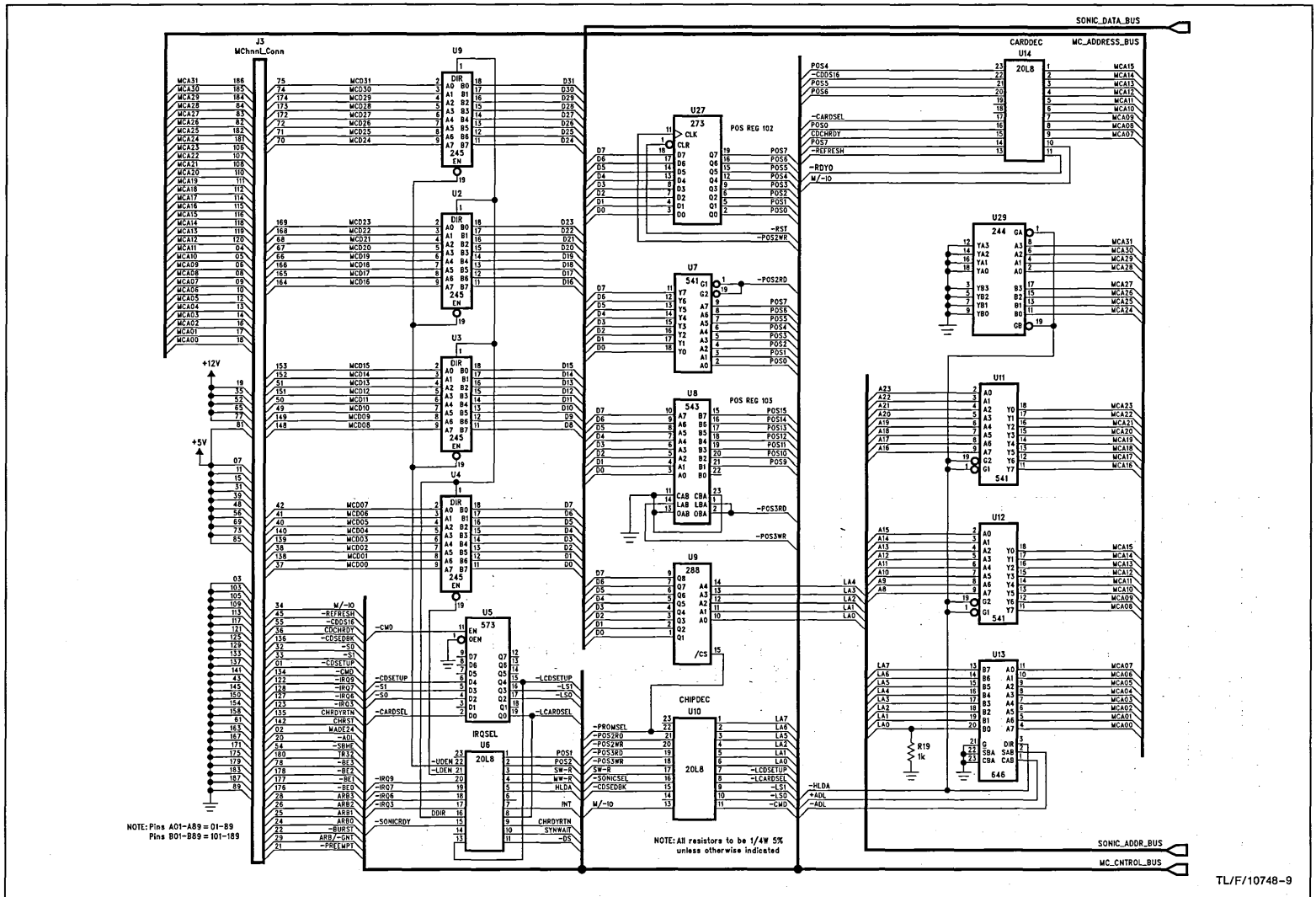
Bit	Name	Function
7–4	SELARB3–0	**Arbitration Vector:** These bits select the arbitration vector when SONIC contends for the bus. The selections are shown below: **SELARB3–0** **Arbitration Vector** 1111 Not Used 1110 ARB14 Lowest Priority 1101 ARB13 1100 ARB12 1011 ARB11 1010 ARB10 1001 ARB9 1000 ARB8 0111 ARB7 0110 ARB6 0101 ARB5 0100 ARB4 0011 ARB3 0010 ARB2 0001 ARB1 0000 ARB0 Highest Priority
3	FAIREN	**Fairness Enable:** When this bit is set to a "1", the Microchannel fairness algorithm is used. When set to a "0", fairness is diabled.
2	E/T	**Ethernet/Thin-Ethernet Select:** This bit selects between the Ethernet and Thin-Ethernet options. This pin is directly connected to the DC-DC Converter. 0: Thin-Ethernet selected (Thin cable) 1: Ethernet selected (Thick cable)
1	EXT	**External ENDEC Select:** This bit selects between the internal and external ENDEC options. This bit is directly connected to the EXT pin of the SONIC. 0: The SONIC's internal ENDEC is enabled. 1: The internal ENDEC of the SONIC is disabled. An external ENDEC is to be used.
0	res	**Reserved**

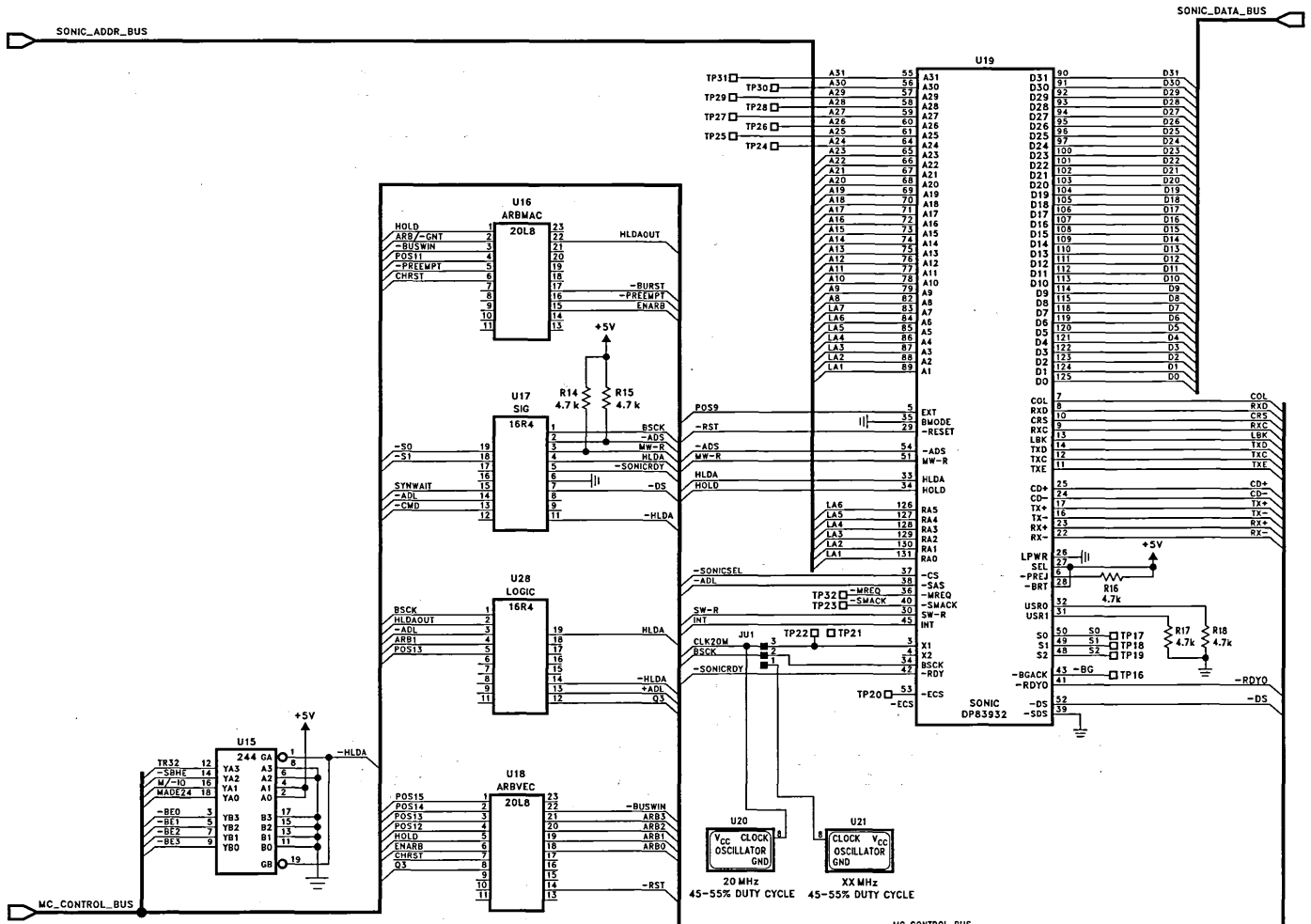
PARTS LIST

Qty.	Part	Number
1	0.01 μF/50V	C2
1	0.01 μF/500V	C1
2	0.01 μF/50V	C4, C5
1	47 μF	C3
1	0.001 μF	C6
36	0.1 μF	C10–C46
2	4.7 μF	C47, C48
1	IN4150	D1
2	1K, 1%	R1, R19
1	1M, ½w	R2
4	1.5K	R3, R4, R5, R6
4	39.2, 1%	R7, R8, R9, R10
2	270	R11, R12
5	4.7K	R14, R15, R16, R17, R18
2	PE64104	T1, T2
1	DP83932 SONIC	U19
1*	DP83910 SNI	U23
1	DP8392 CTI	U25
2	74ACT244	U15, U29
4	74ACT245	U1, U2, U3, U4
1	74ACT273	U27
1	74S288	U9
3	74ACT541	U7, U11, U12
1	74ACT573	U5
1	74ACT646	U13
2	PAL16R4D	U17, U28
5	PAL20L8D	U6, U10, U14, U16, U18
1	20 MHz, 45%–55%	U20
1*	xx MHz, 45%–55%	U21
1	PM9005	U26
1	Spark Gap	C49
1	15 Pin D Connector	J1
1	BNC Connector	J2
1	MChannel Conn.	J3
1	3 Pin Jumper	JU1
1	Dip Switch	SW1
17	Test Points	TP16–TP32

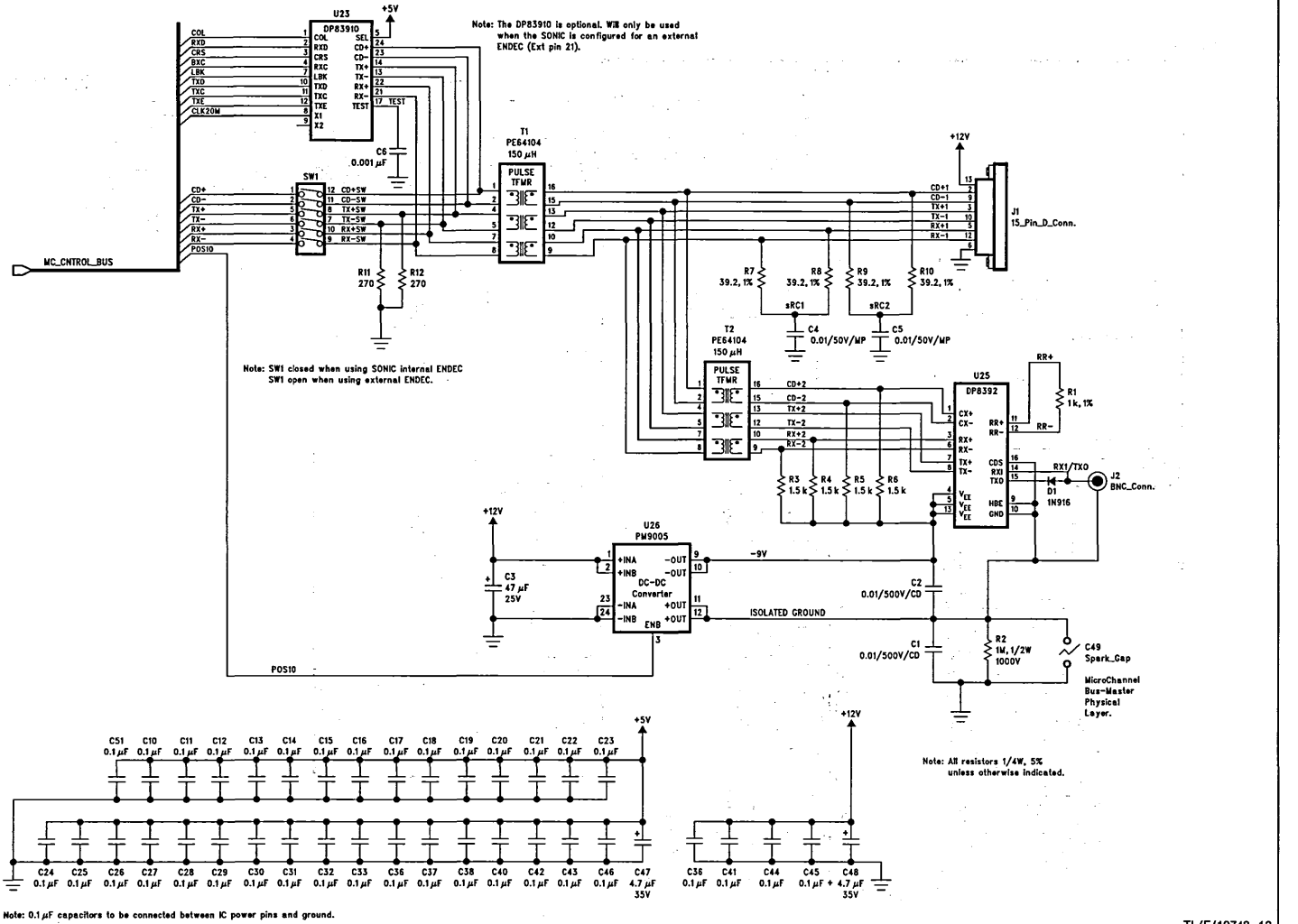
*Optional

All resistors are 5% unless otherwise specified


SONIC_DATA_BUS
J3
MChnnl_Conn
CARDDEC
MC_ADDRESS_BUS
U14
20L8
U9
U2
U3
U4
U5
U6
U7
U8
U27
273
POS REG 102
POS REG 103
U29
244
U11
U12
U13
541
543
573
288
646
CHIPDEC
U10
IRQSEL
+12V
+5V
R19
1k
NOTE: Pins A01-A89 = 01-89
Pins B01-B89 = 101-189
NOTE: All resistors to be 1/4W 5%
unless otherwise indicated
SONIC_ADDR_BUS
MC_CNTROL_BUS
TL/F/10748-9


SONIC_ADDR_BUS
SONIC_DATA_BUS
MC_CONTROL_BUS
U16
ARBMAC
20L8
HOLD
ARB/-CNT
-BUSWIN
POS11
-PREEMPT
CHRST
HLDAOUT
-BURST
-PREEMPT
ENARB
+5V
R14
4.7 k
R15
4.7 k
U17
SIG
16R4
-S0
-S1
SYNWAIT
-ADL
-CWD
BSCK
-ADS
MW-R
HLDA
-SONICRDY
-DS
-HLDA
U28
LOGIC
16R4
BSCK
HLDAOUT
-ADL
ARB1
POS13
HLDA
-HLDA
+ADL
Q3
U15
244
TR32
-SBHE
W/-IO
MADE24
-BE0
-BE1
-BE2
-BE3
GA
YA3
YA2
YA1
YA0
YB3
YB2
YB1
YB0
GB
A3
A2
A1
A0
B3
B2
B1
B0
U18
ARBVEC
20L8
POS15
POS14
POS13
POS12
HOLD
ENARB
CHRST
Q3
-BUSWIN
ARB3
ARB2
ARB1
ARB0
-RST
U19
SONIC
DP83932
TP31
TP30
TP29
TP28
TP27
TP26
TP25
TP24
POS9
-RST
-ADS
MW-R
HLDA
HOLD
-SONICSEL
-ADL
TP32
TP23
-MREQ
-SMACK
SW-R
INT
JU1
TP22
TP21
CLK20M
BSCK
-SONICRDY
TP20
-ECS
EXT
BMODE
-RESET
-CS
-SAS
X1
X2
-RDY
A31
A30
A29
A28
A27
A26
A25
A24
A23
A22
A21
A20
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
LA6
LA5
LA4
LA3
LA2
LA1
A6
A5
A4
A3
A2
A1
A0
D31
D30
D29
D28
D27
D26
D25
D24
D23
D22
D21
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
COL
RXD
CRS
RXC
LBK
TXD
TXC
TXE
CD+
CD-
TX+
TX-
RX+
RX-
LPWR
SEL
-PREJ
-BRT
R16
4.7k
USR0
USR1
S0
S1
S2
TP17
TP18
TP19
R17
4.7k
R18
4.7k
-BGACK
-RDYO
-BG
TP16
-DS
-SDS
U20
CLOCK
OSCILLATOR
20 MHz
45-55% DUTY CYCLE
U21
CLOCK
OSCILLATOR
XX MHz
45-55% DUTY CYCLE

TL/F/10748–10


U23
DP83910
+5V
Note: The DP83910 is optional. Will only be used when the SONIC is configured for an external ENDEC (Ext pin 21).
COL
RXD
CRS
BXC
LBK
TXD
TXC
TXE
CLK20M
SEL
CD+
CD-
TX+
TX-
RX+
RX-
TEST
X1
X2
C6
0.001 µF
T1
PE64104
150 µH
PULSE
TFMR
SW1
CD+SW
CD-SW
TX+SW
TX-SW
RX+SW
RX-SW
MC_CNTROL_BUS
POS10
R11
270
R12
270
Note: SW1 closed when using SONIC internal ENDEC
SW1 open when using external ENDEC.
+12V
J1
15_Pin_D_Conn.
R7
39.2, 1%
R8
39.2, 1%
R9
39.2, 1%
R10
39.2, 1%
sRC1
sRC2
C4
0.01/50V/MP
C5
0.01/50V/MP
T2
PE64104
150 µH
U25
DP8392
RR+
RR-
R1
1k, 1%
CX+
CX-
CDS
RXI
TXO
HBE
GND
VEE
RX1/TXO
D1
1N916
J2
BNC_Conn.
R3
1.5 k
R4
1.5 k
R5
1.5 k
R6
1.5 k
U26
PM9005
DC-DC
Converter
+INA
+INB
-INA
-INB
-OUT
+OUT
ENB
-9V
C3
47 µF
25V
C2
0.01/500V/CD
ISOLATED GROUND
C1
0.01/500V/CD
R2
1W, 1/2W
1000V
C49
Spark_Gap
MicroChannel
Bus-Master
Physical
Layer.
Note: All resistors 1/4W, 5%
unless otherwise indicated.
C51 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23
0.1 µF
C24 C25 C26 C27 C28 C29 C30 C31 C32 C33 C36 C37 C38 C40 C42 C43 C46
0.1 µF
C47
4.7 µF
35V
C36 C41 C44 C45 C48
0.1 µF
4.7 µF
35V
Note: 0.1 µF capacitors to be connected between IC power pins and ground.

TL/F/10748–12

PAL EQUATIONS

The DP82932 SONIC Microchannel adapter contains 7 PALs to implement the bus interface. This section provides a listing of the PALs used. All listings use the abel™ design format.

```
module carddec;          flag '-r2','-t2';

title
'PAL20L8                                    National Semiconductor
MicroChannel Card Decoder                          1/1/90
file name: CARDDEC.ABL                                  wl'

CARDDEC device 'P20L8';
" PAL DESCRIPTION

" This pal determines whether the Micro Channel Ethernet adapter
" is selected.  The address base is determined by bits 7 - 4
" in POS register 102.  The output signals are following:
"      !cardsel = active when base address matches
"      !cdds16 = active when the SONIC register are being accessed
"      !cdchrdy = ready signal

" Equations written in ABEL™ design format.
" declarations

"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      x,z,c = .X.,.Z.,.C.;

      GND,VCC
            pin  12,24;

"outputs
      halfsel1,halfsel2,cardsel,cdds16,cdchrdy
      pin 19,18,17,22,15;


"inputs
      a15,a14,a13,a12,a11,a10,a9,a8,a7,mio,sonicsel,
      addr0,addr1,addr2,addr3,pos0,rdyo
      pin 1,2,3,4,5,6,7,8,9,10,13,23,21,20,14,16,11;

"equates
      addr = [a15,a14,a13,a12,a11,a10,a9,a8];
      possel = [addr3,addr2,addr1,addr0];

equations

!halfsel1 = (possel == ^h0) & (addr == ^h4) #
            (possel == ^h1) & (addr == ^h6) #
            (possel == ^h2) & (addr == ^h8) #
            (possel == ^h3) & (addr == ^ha) #
            (possel == ^h4) & (addr == ^hc) #
            (possel == ^h5) & (addr == ^he) #
            (possel == ^h6) & (addr == ^h10);

!halfsel2 = (possel == ^h7) & (addr == ^h12) #
            (possel == ^h8) & (addr == ^h14) #
```

TL/F/10748–11

PAL EQUATIONS (Continued)

```
          (possel == ^h9) & (addr == ^h16) #
          (possel == ^ha) & (addr == ^h18) #
          (possel == ^hb) & (addr == ^h1a) #
          (possel == ^hc) & (addr == ^h1c) #
          (possel == ^hd) & (addr == ^h1e);

!cardsel = !halfsel1 & !mio & pos0 #
          !halfsel2 & !mio & pos0;

!cdds16 = !cardsel & !a7;

!cdchrdy = !halfsel1 & !mio & pos0 & !a7 & rdyo #
          !halfsel2 & !mio & pos0 & !a7 & rdyo #
         !sonicsel & rdyo;


test_vectors
([addr,possel,mio,pos0] -> [cardsel,cdds16])

[^h4  , ^h0  , 1     ,1] -> [ 0     ,  x   ]; "cardsel
[^h6  , ^h1  , 1     ,1] -> [ 0     ,  x   ];
[^h8  , ^h2  , 1     ,1] -> [ 0     ,  x   ];
[^ha  , ^h3  , 1     ,1] -> [ 0     ,  x   ];
[^hc  , ^h4  , 1     ,1] -> [ 0     ,  x   ];
[^he  , ^h5  , 1     ,1] -> [ 0     ,  x   ];
[^h10 , ^h6  , 1     ,1] -> [ 0     ,  x   ];

[^h12 , ^h7  , 1     ,1] -> [ 0     ,  x   ]; "cardsel
[^h14 , ^h8  , 1     ,1] -> [ 0     ,  x   ];
[^h16 , ^h9  , 1     ,1] -> [ 0     ,  x   ];
[^h18 , ^ha  , 1     ,1] -> [ 0     ,  x   ];
[^h1a , ^hb  , 1     ,1] -> [ 0     ,  x   ];
[^h1c , ^hc  , 1     ,1] -> [ 0     ,  x   ];
[^h1e , ^hd  , 1     ,1] -> [ 0     ,  x   ];

[^h1e , ^hd  , 1     ,0] -> [ 1     ,  x   ];
[^h1c , ^hd  , 1     ,x] -> [ 1     ,  x   ];

test_vectors
([addr,possel,mio,a7,pos0] -> [cardsel,cdds16])
[^h1e , ^hd  , 1     , 1,1] -> [ 0     ,  1   ]; "cdds16
[^h1e , ^hd  , 1     , 0,1] -> [ 0     ,  0   ]; "cdds16
[^h1e , ^hd  , 1     , 0,0] -> [ 1     ,  1   ]; "cdds16

test_vectors
([addr,possel,mio,pos0,rdyo,a7] -> [cardsel,cdchrdy])

[^h4  , ^h0  , 0 , 1   ,  1 ,0 ] -> [ 0     ,  0   ]; "cdchrdy
[^h4  , ^h0  , 0 , 1   ,  0 ,0 ] -> [ 0     ,  1   ]; "cdchrdy
[^h6  , ^h0  , 0 , 1   ,  1 ,0 ] -> [ 1     ,  1   ]; "cdchrdy
[^h4  , ^h0  , 1 , 1   ,  1 ,0 ] -> [ 1     ,  1   ]; "cdchrdy
[^h4  , ^h0  , 0 , 1   ,  1 ,0 ] -> [ 1     ,  1   ]; "cdchrdy
[^h4  , ^h0  , 0 , 0   ,  1 ,0 ] -> [ 1     ,  1   ]; "cdchrdy
[^h4  , ^h0  , 0 , 1   ,  1 ,1 ] -> [ 0     ,  1   ]; "cdchrdy


end carddec;
```

TL/F/10748–13

PAL EQUATIONS (Continued)

```
Module chipdec;          flag '-r2','-t2';

title
PAL20L8                                        National Semiconductor
MicroChannel Slave Chip Decoder                          1/1/90
file name: CHIPDEC.ABL                                        wl'

CHIPDEC device 'P20L8';

" PAL DESCRIPTION

" This pal selects which component on the board is accessed.  See
" DP83932 SONIC Micro Channel application note for I/O mapping.
"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      x,z,c = .X.,.Z.,.C.;

      GND,VCC
            pin  12,24;

"outputs
      sonicsel,promsel,pos2rd,pos2wr,pos3rd,pos3wr,swr,cdsfdbk
  pin     16    ,   22  ,   21 ,  20  ,  19  ,  18  ,17 ,  15;

"inputs
      la7,la6,la5,la2,la1,la0,lchsetup,lcardsel,ls1,ls0,mio,cmd
  pin   1  , 2 , 3 , 4 , 5 , 6 ,   7    ,    8   , 9 ,10 ,13, 11;

"equates
      addr = [la7,la6,la5,la2,la1,la0];

equations

!sonicsel = !lcardsel & !la7 & !cmd;

!promsel  = !lcardsel & la7 & !la6 & !la5 & ls0 & !ls1 & !cmd  #
            !lchsetup & !la2 & !la1 & ls0 & !ls1  & !cmd;

!pos2rd   = !lcardsel & la7 & !la6 & la5 & !la0 & ls0 & !ls1  & !cmd#
            !lchsetup & !la2 & la1 & !la0 & ls0 & !ls1  & !cmd;

!pos2wr   = !lcardsel & la7 & !la6 & la5 & !la0 & !ls0 & ls1  & !cmd#
            !lchsetup & !la2 & la1 & !la0 & !ls0 & ls1  & !cmd;

!pos3rd   = !lcardsel & la7 & !la6 & la5 & la0 & ls0 & !ls1  & !cmd #
            !lchsetup & !la2 & la1 & la0 & ls0 & !ls1  & !cmd;

!pos3wr   = !lcardsel & la7 & !la6 & la5 & la0 & !ls0 & ls1  & !cmd #
            !lchsetup & !la2 & la1 & la0 & !ls0 & ls1  & !cmd;

!swr      = ls0 & !ls1;

!cdsfdbk  = !lcardsel & lchsetup & !mio;


test_vectors
([addr   ,lchsetup,lcardsel,ls0,ls1,cmd] -> [sonicsel,promsel,swr])
```

TL/F/10748–14

PAL EQUATIONS (Continued)

```
[^b011111,  x     , 0     , x , x ,0  ] -> [   0    ,  x    , x  ]; "sonicsel
[^b111111,  x     , 0     , x , x ,x  ] -> [   1    ,  x    , x ]; "mreq
[^b111111,  x     , 0     , 1 , 0 ,x  ] -> [   x    ,  x    , 0 ]; "swr
[^b111111,  x     , 0     , 0 , 1 ,x  ] -> [   x    ,  x    , 1 ]; "swr

[^b100111,  x     , 0     , 1 , 0 ,0  ] -> [   1    ,  0    , x ]; "promsel
[^b101111,  x     , 0     , 1 , 0 ,0  ] -> [   1    ,  1    , x ]; "promsel
[^b111001,  0     , x     , 1 , 0 ,0  ] -> [   1    ,  0    , x ];
[^b111000,  0     , x     , 1 , 0 ,0  ] -> [   1    ,  0    , x ];

test_vectors
([addr   ,lchsetup,lcardsel,ls0,ls1,cmd] -> [pos2rd,pos2wr,pos3rd,pos3wr])

[^b101000,  x     , 0     , 1 , 0,0 ] -> [   0  ,  1   , 1    ,  1   ];
"pos2rd
[^b111010,  0     , x     , 1 , 0,0 ] -> [   0  ,  1   , 1    ,  1   ];

[^b101000,  x     , 0     , 0 , 1,0 ] -> [   1  ,  0   , 1    ,  1   ];
"pos2wr
[^b111010,  0     , x     , 0 , 1,0 ] -> [   1  ,  0   , 1    ,  1   ];

[^b101001,  x     , 0     , 1 , 0,0 ] -> [   1  ,  1   , 0    ,  1   ];
"pos3rd
[^b111011,  0     , x     , 1 , 0,0 ] -> [   1  ,  1   , 0    ,  1   ];

[^b101001,  x     , 0     , 0 , 1,0 ] -> [   1  ,  1   , 1    ,  0   ];
"pos3wr
[^b111011,  0     , x     , 0 , 1,0 ] -> [   1  ,  1   , 1    ,  0   ];


test_vectors
([lcardsel,lchsetup,mio] -> [cdsfdbk])

[   1     ,   x    ,  x ] -> [  1  ];
[   x     ,   0    ,  x ] -> [  1  ];
[   0     ,   1    ,  0 ] -> [  0  ];

end chipdec;
```

TL/F/10748–15

PAL EQUATIONS (Continued)

```
module irqsel;          flag '-r2','-t2';

title
'PAL20L8                                        National Semiconductor
MicroChannel IRQ Selector and Buffer Enable                    1/1/90
file name: IRQSEL.ABL                                           wl'

IRQSEL device 'P20L8';

" This pal selects which interrupt line to use (IRQ9, IRQ7, IRQ6, or
" IRQ3) when the SONIC asserts its interrupt. Each IRQ line is an open
" collector type output (only asserted low).  This pal also enables the
" data buffers with !ddir, !lden, and !uden, and produces the ready
" signal for the SONIC with !sonicrdy. Wait states are inserted by
" memory (cdchrdy not asserted) and by the signal translator(SIG5) pal
" (synwait not asserted).
"declarations

"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      x,z,c = .X.,.Z.,.C.;

      GND,VCC
            pin  12,24;

"outputs
            ddir,uden,lden,irq9,irq7,irq6,irq3,sonicrdy
  pin    16 ,22 , 21  , 20 , 19 , 18 , 17 ,   15;

"inputs
      int0,int1,swr,mwr,hlda,cmd,int,lcardsel,cdchrdy,synwait,ds,lchsetup,
smack
  pin    1  , 2  , 3 , 4 , 5  , 6,  7  ,  8     ,  9    ,  10 , 11,  13,
14  ;

"equates
      sel = [int0,int1];

equations

      irq9 = 0;
      irq7 = 0;
      irq6 = 0;
      irq3 = 0;

enable irq9 = int & (sel == 3);
enable irq7 = int & (sel == 2);
enable irq6 = int & (sel == 1);
enable irq3 = int & (sel == 0);

!ddir = !swr & !hlda #          "Data buffer direction
        mwr & hlda;

!lden  = hlda & !ds #           "Address and Lower Data buffer enable (D7-D0)
       !lchsetup  #
       !lcardsel & !cmd;
```

TL/F/10748–27

PAL EQUATIONS (Continued)

```
!uden  = hlda & !ds #
       !smack & !cmd;          "Upper data buffer enable (D16 - D31)

!sonicrdy = cdchrdy & synwait;      "ready signal for the SONIC and sig.
trans.


end irqsel;

" PAL DESCRIPTION
```

TL/F/10748-28

PAL EQUATIONS (Continued)

```
module arbmac2;          flag '-r2','-t2';

title
'PAL20L8                                        National Semiconductor
Micro Channel Bus Arbiter State Machine                    1/8/89
file name: ARBMAC.ABL                                          wl'

ARBMAC2 device 'P20L8';

" PAL DESCRIPTION

" This pal controls the ARBVEC pal when it may drives the arbitration
" vector.  This pal consists on an aysnchronous state machine.  These
" state machine equations (!q0 - !q3) are not reduced (ABEL™ will do this).
" The outputs are described as follows:
"     enarb = enables the ARBVEC pal to drive the arb. vector
"     hlda = hold acknowledge to the SONIC
"     burst = BURST signal on the microchannel bus (open collector
"                 type output).
"     preout = PREMPT signal on the microchannel bus (open collector
"                 type output).



"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      X,Z,C = .X.,.Z.,.C.;

      GND,VCC
            pin  12,24;

      "outputs
      q3,q2,q1,q0,burst,preout,hlda,enarb
            pin  21,20,19,18,17,16,22,15;

      "inputs
      hold,arbgnt,buswin,fair,prein,chrst
            pin  1,2,3,4,5,6;

" States of Arbiter

      st = [q3,q2,q1,q0];

      idle = st ==  [1,1,1,1];
      req  = st ==  [1,1,1,0]; "request uchannel bus; preout (q0) active
      arb  = st ==  [1,0,1,0]; "vectoring arb priority; preout(q0), enarb
active
      lose = st ==  [1,0,0,0]; "lost arb battle; preout (q0) active
      xfer1 = st == [0,0,1,0]; "intermdiate state to xfer2
      xfer2 = st == [0,0,1,1]; "xfering data on bus; burst, hlda active
      xfer3 = st == [1,0,1,1]; "intermediate state to idle
      pen   = st == [0,1,1,1]; "holding pen when fairness is enabled
      esc1  = st == [1,0,0,1]; "intermediate state to esc2
      esc2  = st == [1,1,0,1]; "intermediate state to idle

equations
```

TL/F/10748–30

PAL EQUATIONS (Continued)

```
!q0 = idle & hold & !arbgnt & !chrst #   " This also -PREEMPT on the MCA bus
      req  & hold & !arbgnt & !chrst #
      req  & hold & arbgnt & !chrst #
      arb  & arbgnt & !chrst #
      arb  & !arbgnt & buswin & !chrst #
      arb  & !arbgnt & !buswin & !chrst #
      lose & hold & !arbgnt & !chrst #
      lose & hold & arbgnt & !chrst;

!q1 = arb  & !arbgnt & buswin & !chrst #
      lose & hold & !arbgnt & !chrst #
      lose & !hold & !chrst #
      esc1 & !chrst;

!q2 = req   & hold & arbgnt & !chrst #
      arb   & arbgnt  & !chrst #
      arb   & !arbgnt & buswin & !chrst #
      arb   & !arbgnt & !buswin & !chrst #
      lose  & hold & !arbgnt & !chrst #
      lose  & hold & arbgnt & !chrst #
      lose  & !hold & !chrst #
      xfer1 & !chrst #
      xfer2 & hold & !arbgnt & !chrst #
      xfer2 & !hold & !arbgnt & !fair & !chrst;


!q3 = arb   & !arbgnt & !buswin & !chrst #
      xfer1 & !chrst #
      xfer2 & hold & !arbgnt & !chrst #
      xfer2 & !hold & !arbgnt & fair & !chrst #
      xfer2 & arbgnt & !chrst #
      pen   & !prein & !chrst;

enarb  = arb # xfer1 # xfer2; "enables arb vector (ARB0 - 3) on bus
hlda    = xfer2;        "HOLD ACK to SONIC
enable burst   = xfer2;       "-BURST on MCA bus (tri-state output)
enable preout  = !q0;         "-PREMPT on MCA bus (tri-state output)

burst   = 0;
preout  = 0;

end arbmac2;
```

TL/F/10748–31

PAL EQUATIONS (Continued)

```
module arb;       flag '-r2','-t2';

title
'PAL20L8                                        National Semiconductor
MicroChannel Bus Arbiter Vector                          1/9/90
ARBVEC4                                                       wl'

ARBVEC4 device 'P20L8';

" PAL DESCRIPTION

" This pal dumps the arbitration vector onto the Micro Channel bus
" (ARB0 - 3) when the ARBMAC pal enables it (by ENARB). ARB3 - 0
" are open collector type outputs (only driven low).  This pal
" indicates it has won the bus by driving -BUSWIN low.

"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      X,Z,C = .X.,.Z.,.C.;

      GND,VCC
            pin  12,24;
" Outputs
            BUSWIN,ARB3,ARB2,ARB1,ARB0,Q,Q2,_RST
            pin  22,21,20,19,18,17,16,15;
" Inputs
      SEL3,SEL2,SEL1,SEL0,HOLD,ENARB,CHRST,Q3
            pin  1,2,3,4,5,6,7,8;

equations

      ARB3 = 0;          "When enabled, these tri-state outputs
      ARB2 = 0;          "will pull these arbitration lines low.
      ARB1 = 0;
      ARB0 = 0;

Q           = (ARB3 # !SEL3);
Q2          = (ARB3 # !SEL3) & (ARB2 # !SEL2);

!BUSWIN     = Q2 & Q3 & (ARB0 # !SEL0) & ENARB & HOLD;

enable ARB3 = !SEL3 & ENARB & HOLD;
enable ARB2 = !SEL2 & Q & ENARB & HOLD;
enable ARB1 = !SEL1 & Q2 & ENARB & HOLD;
enable ARB0 = !SEL0 & Q2 & Q3 & ENARB & HOLD;

_RST = !CHRST;                  " reset for the adapter

end arb;
```

TL/F/10748–32

PAL EQUATIONS (Continued)

```
module sig;       flag '-r2','-t2';

title
'PAL16R4                                        National Semiconductor
MicroChannel Signal Translator for the SONIC                    1/1/90
file name: SIG5.ABL                                              wl'

SIG5 device 'P16R4';


" PAL DESCRIPTION

" This pal provides the signal conversion from the SONIC to the
" Micro Channel bus.  The state machine used is written in the
" ABEL™ design format.


"declarations

      TRUE,FALSE = 1,0;
      H,L = 1,0;
      x,z,c = .X.,.Z.,.C.;

      GND,VCC
            pin  10,20;

      "outputs
      q3,q2,q1,q0,s0,s1,cmd,delayads
            pin  17,16,15,14,19,18,13,12;
      "q0 is used for -ADL on MCA bus


      "inputs
      bsck,ads,mwr,hlda,rdy,fast,enb,ds
            pin  1,2,3,4,5,6,11,7;

      input = [ads,hlda,rdy,fast];

" States of Translator

      idle    = ^b1111;
      data    = ^b1011;
      wait    = ^b1101; "wait state to delay assertion of -ADL
      addlcht  = ^b1110; "asserting -ADL on MCA bus
      cmd1    = ^b0101; "beginning to assert -CMD;
      cmd2    = ^b0011; "still asserting -CMD; return to DATA state
                        " on next clock
      cmd3    = ^b0001;  "still asserting -CMD; inserting 1st wait-state
                        "for MCA's synchronous mode

      data_st  = [q3,q2,q1,q0] == [1,0,1,1];
      wait_st  = [q3,q2,q1,q0] == [1,1,0,1];
      addl_st  = [q3,q2,q1,q0] == [1,1,1,0];
      cmd1_st  = [q3,q2,q1,q0] == [0,1,0,1];
      cmd3_st  = [q3,q2,q1,q0] == [0,0,0,1];
      idle_st  = [q3,q2,q1,q0] == [1,1,1,1];


state_diagram [q3,q2,q1,q0]
```

TL/F/10748–33

PAL EQUATIONS (Continued)

```
State idle: case (input == [0,1,x,0])     :wait;
                 (input == [1,1,x,x])     :idle;
                 (input == [x,0,x,x])     :idle;
                 (input == [0,1,x,1])     :addlcht;  "fast mode
            endcase;


State data: case (input == [1,1,x,x])     :data;
                 (input == [0,1,x,0])     :wait;
                 (input == [x,0,x,x])     :idle;
                 (input == [0,1,x,1])     :addlcht;  "fast mode
            endcase;

State wait: case (input == [x,1,x,x])  :addlcht;
                 (input == [x,0,x,x])  :idle;
            endcase;

State addlcht:    case (input == [x,1,0,x])  :cmd2;
                     (input == [x,1,1,x])  :cmd1;
                     (input == [x,0,x,x])  :idle;
                endcase;

State cmd1: case (input == [x,1,x,x])  :cmd3;
                     (input == [x,0,x,x])  :idle;
                endcase;

State cmd2: case (input == [x,1,x,x])  :data;
                 (input == [x,0,x,x])  :idle;
                endcase;

State cmd3: case (input == [x,1,x,x])  :data;
                 (input == [x,0,x,x])  :idle;
                endcase;

equations

enable s0  = hlda;
enable s1  = hlda;
enable cmd = hlda;

" MCA Signals

!s0  = mwr & !data_st & !idle_st;   "-S0 for MCA bus

!s1  = !mwr & !data_st & !idle_st;  "-S1 on MCA bus

!cmd = !q3 & q2 & delayads & !ds & hlda #
       !q3 & !q2 & delayads & !ds & hlda #      "-CMD for MCA bus
       q3 & !q2 & delayads & !ds & hlda;

delayads = ads;                        "delaying ADS for the -CMD term to
                                "prevents glitches from occuring
                                "during the transitions from the DATA
                                "to the WAIT state


end sig;
```

TL/F/10748–34

PAL EQUATIONS (Continued)

```
module logic;

title
'PAL16R4 Microchannel Logic                          National Semiconductor
file name: logic.abl                                       1/9/90'

LOGIC device 'P16R4';

"declarations

      TRUE, FALSE = 1,0;
      H,L = 1,0;
      X,Z,C = .X.,.Z.,.C.;

      GND, VCC
            PIN 10, 20;

"inputs
      bsck,hldaout,_adl,arbl,sell,enb
      pin 1,2,3,4,5,11;
"outputs
      q3,adl,hlda,_hlda,hlda1
      pin 12,13,15,14,19;

equations

      hlda := hldaout;
      _hlda := !hldaout;
      hlda1 = hlda;
      adl = !_adl;
      q3 = arbl # !sell;

test_vectors ([bsck,hldaout,_adl,arbl,sell,enb] -> [hlda,_hlda,hlda1,adl,q3])
[C,1,1,0,0,0] -> [1,0,1,0,1];
[C,0,0,0,1,0] -> [0,1,0,1,0];
[C,1,X,1,1,0] -> [1,0,1,X,1];
end log;
```

TL/F/10748–35

High Performance AT Compatible Bus Master Ethernet Adapter

National Semiconductor
Application Note 760

INTRODUCTION

The DP839EB-ATS/DP83932EB-AT is a high performance 16-bit Ethernet adapter card for IBM® PC-AT® and compatible computers. The main function of this adapter card is to transfer Ethernet packet data to/from the AT computer's memory via the AT system bus during LAN transmission and reception. This board employs National Semiconductor's newest Ethernet Controller: the DP83932 System Oriented Network Interface Controller (SONIC™).

Note: This design has been tested on a number of AT compatible machines. See Table X.

Using the latest generation 16/32-bit Ethernet controller, this design enables the implementation of a high performance bus master Ethernet card by utilizing the SONIC's capabilities to enhance performance over today's I/O mapped or dual ported adapter RAM designs. Additionally, since this utilizes the highly integrated SONIC controller and does not require on-card buffer RAM, the cost of the board is roughly equivalent to many current 16-bit RAM based implementations.

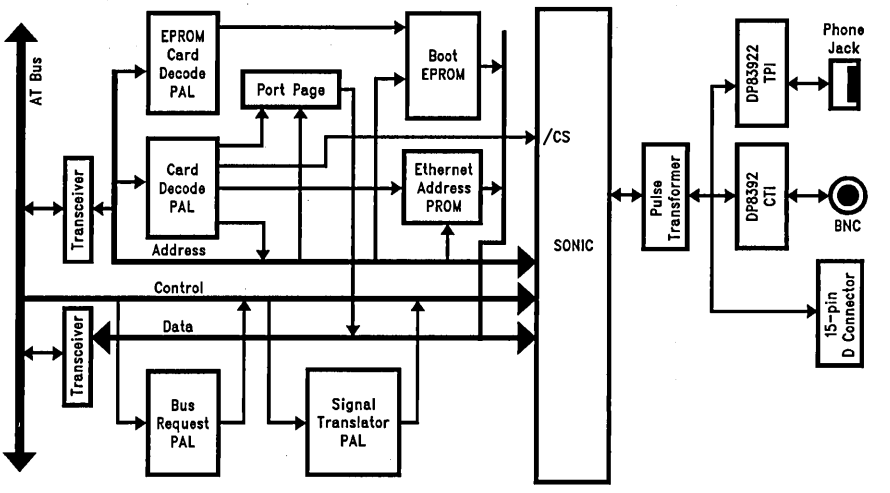
Since the SONIC can become an efficient bus master, it can access system memory directly. This architecture enables network data to be directly placed into main system memory. The SONIC is ideally suited for this architecture because of it's bus latency, it's full 32-bit memory addressing ability, and it's sophisticated link-listed buffer management scheme. The buffer management of the SONIC allows the Network OS to process receive and transmit packets without extraneous copying of data.

Also unique to this adapter design is its ability to support any one of the three IEEE 802.3 cable interfaces: Thin Ethernet (RG58), Thick Ethernet (via AUI cable and external transceiver), and the new twisted pair cable, 10BASE-T, standard. This design includes the DP8392 Coax Transceiver to drive Thin wire Ethernet, and the DP83922 Twisted Pair Interface. The DP83932 Ethernet Controller includes a PLL encoder/decoder with the capability of driving the AUI cable itself. These chips enable the implementation of a very low parts count adapter that easily provides all three standard cable interface options by merely altering the jumper block configuration on the board.

This paper will first discuss the overview of the hardware design, then the details of the design. Finally the PAL® equations, and detailed schematics of this Ethernet Adapter are described.

HARDWARE OVERVIEW

The block diagram for the board is shown in *Figure 1*. The design can be broken into 3 sections: the slave logic section, the bus master logic section, and the physical interface.



TL/F/11180–1

FIGURE 1. SONIC AT Board Block Diagram

The slave interface allows the AT's processor to access the SONIC and other registers on the board. Included on this board, besides SONIC, are a Network Boot EPROM (socket), Ethernet Address ID PROM, and a paging register. Since the SONIC registers are 16-bit wide, the board ensures 16-bit transfers by asserting the IO16 signal back to the system board. The slave circuitry decode and control logic is implemented by: the CARDEC PAL, EPROMDEC PAL, COMMAND PAL and CONTROL PAL.

The bus master section contains the bus request logic, which is in charge of converting the SONIC's bus request and acknowledge signal into the signals required by the AT bus. Since the system board of the AT uses the 8237 DMA controller to arbitrate between bus requestors, the adapter itself does not perform any arbitration. While the SONIC is the bus master, a second function of the bus master logic is to convert the SONIC's bus signals to signals compatible with the AT bus. The bus master logic is implemented by several PALs on the board: the BUSREQ PAL, SIGTRAN PAL, and CONTROL PAL.

The third section, the physical interface, is responsible for connecting the SONIC to the three supported network media. Thick Ethernet, Thin Ethernet and 10BASE-T twisted pair cabling. These are selectable by changing a set of jumpers.

All three blocks are described in detail in the following sections, starting with the slave logic.

SLAVE LOGIC SECTION

The PC-AT's processor can access one of the four slave devices on the board, the SONIC's 64 internal registers, the Ethernet address PROM, the Port Page register or the boot-up EPROM. The first three reside in the PC's I/O space.

The slave logic supports 16-bit transfers to the SONIC registers by driving the IO16 signal back to the AT. The PROM and Port Page registers are 8-bit devices. The optional boot-up EPROM, which resides in the PC's memory address space as an 8-bit device, contains code to enable the PC to boot up remotely from the network when powered up. The slave logic blocks described are shown in *Figure 2*.

Slave Memory and I/O Space Map

There are 7 32-byte I/O address blocks available in the PC-AT's I/O space. The board can be mapped into any of these blocks by means of three on-board user selectable jumpers.

The Boot EPROM can also be memory mapped into one of seven locations above 640K within the first megabyte of memory. The jumpers used for I/O address space configuration are also used to select the EPROM's base address as shown in Table I. Note that a 0 selection in the jumper column represent a jumper link shorted to ground.

TABLE I. Seven I/O Page and EPROM Page Locations

JP0/1/2	I/O Address (Hex)	Memory Address for EPROM (Hex)
000	100–11F	C8000–CBFFF
100	120–13F	CC000–CFFFF
010	140–15F	D0000–D3FFF
110	160–17F	D4000–D7FFF
001	**300–31F***	**D8000–DBFFF**
101	320–33F	DC000–DFFFF
011	340–35F	E0000–E3FFF

***Note:** This address is the default.

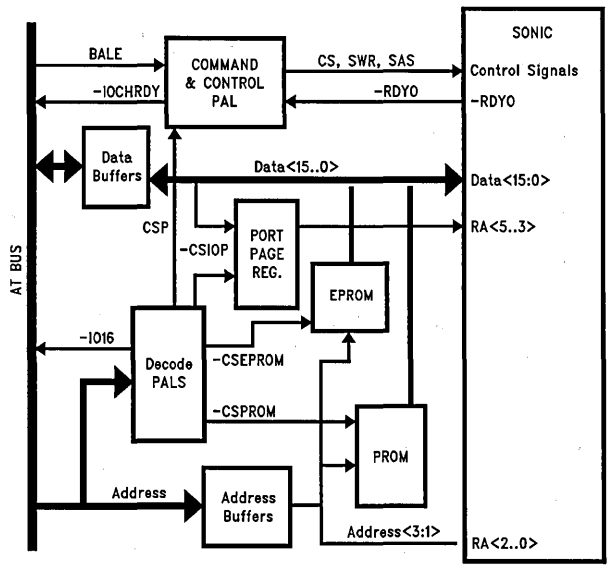


FIGURE 2. Slave Cycle Block Diagram

The detailed mapping of the SONIC, PROM and paging register (called the Port Page) into one of these 32-byte blocks of PC-ATs I/O space map is shown in *Figure 3*. The lower 6 bytes of I/O space contain the Ethernet Address PROM, the 16th location contains the Port Page Register and the upper 16 bytes (8 words) enable access to the 64 SONIC registers by means of a simple mapping scheme explained below.

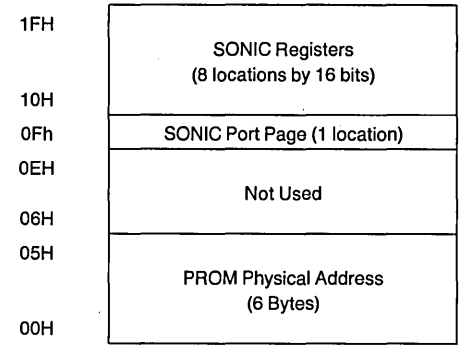


FIGURE 3. SONIC-AT Board's 32-Byte I/O Port Map

SONIC Register Slave Access

Due to the limited availability of I/O space on PC-ATs (7 32-byte I/O address blocks) the adapter card's slave control logic maps the 64 SONIC registers into eight 16-bit locations in the PC-AT I/O address space by means of a 3-bit mapping register called the Port Page register.

This Port Page register is implemented on the adapter board as a D-type latch occupying one 8-bit I/O space location on the PC-AT (see *Figure 4*). The PC-AT's data bits SD3–SD5 containing the SONIC register page number are written to the Port Page by accessing this I/O space location. Its contents are subsequently enabled onto the SONIC's address bus whenever a SONIC register address is decoded on the AT bus.

SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
0	0	RA5	RA4	RA3	0	0	0

FIGURE 4. Port Page Register Definition

The Port Page register divides the SONIC registers into 8 pages of 8 registers each, so that before one of the 64 SONIC registers can be accessed by addressing one of the 8 PC-AT 16-bit I/O locations allocated to the SONIC, the page number (0 to 7) must be written into the Port Page register by means of a separate I/O write cycle. Therefore a two-step process is required to access the SONIC's registers:

STEP 1: Setting up the Port Page register—The CPU executes an 8-bit I/O write cycle to the Port Page Register with SD5–SD3 (PC-AT's Data Bus) containing the page number of the register bank to be accessed. These 3 bits will be used to drive the three most significant SONIC register address bits RA5–RA3 in STEP 2.

STEP 2: Accessing a SONIC register—The CPU executes a normal 16-bit I/O cycle to the SONIC. The PC-AT's lower address lines SA3–SA1 are used to drive the SONIC's least significant register address bits RA2–RA0 while the 3 page number bits from the Port Page register are enabled onto the SONIC's three most significant register address bits RA5–RA3 making up the 6-bit (64 16-bit locations) SONIC register address RA5–0. Note that as SONIC registers are 16 bits wide, the PC-AT's byte addressing bit SA0 is not used and the IO16 signal is driven back to the AT system to identify the SONIC as a 16-bit I/O slave.

Note that, as a mapping register, the Port Page only needs to be set up during initial power up and whenever a SONIC register from a different page is accessed. As page 0 contains the most frequently used SONIC registers (the lower 8 registers), most SONIC register cycles will not require the separate access to the Port Page register described in STEP 1.

Slave Logic Detailed Description

The Slave Logic decodes AT memory and I/O cycles to the adapter card, and implements the address decoding as discussed previously, and conversion of the AT's read/write strobes to the SONIC's. This section discusses the signals provided by the AT to the board, and the signals generated to enable access to the SONIC, PROM, EPROM and Port Page Register.

The AT's address lines SA<9..5> are first decoded with AEN and the SONIC's HLDA for accesses to the adapter card's I/O space and CARDSEL is generated. AEN is a slot specific signal asserted to all slots during DMA cycles to prevent I/O slaves from misinterpreting DMA cycles as valid I/O cycles. Also, during accesses to a particular I/O slot, it is asserted to all other slots to stop them from responding to the cycle. HLDA is generated by the adapter card's master logic when it has been granted ownership of the bus and it is used by the slave decoding logic to prevent the master logic from accessing its own card slave address space.

CARDSEL, together with the bottom SA lines, is further decoded for access to the SONIC registers, the SONIC port page and the Ethernet address PROM, to generate the relevant address decode signals -CSP, -CSIOP and -CSPROM. These signals need to be further decoded with the IOR and IOW signals to disable memory cycle decoding to generate the relevant chip select and chip enable signals.

-CSP is gated with IOR, and IOW to generate -CS, the chip select to the SONIC. -CSIOP is gated with the AT's -IOW strobe to generate EN_IOP, the Port Page register's latch input. The Port Page is a write only register. -CSPROM is gated with the AT's IOR signal to generate EN_PROM, the chip enable to the PROM.

SONIC registers are 16 bits wide, therefore the SONIC register address decode -CSP is used to drive IO16 back to the AT to indicate a 16-bit slave device. Note that as IO16 must be generated before IOR, IOW are asserted, -CSP rather than -CS is used to generate IO16. Therefore IO16 will be asserted during some memory cycles. The SONIC Port Page and Ethernet address PROM are 8-bit slave devices.

A separate PAL, EPROMDEC, is used to decode SA<19..14>, HLDA, AEN and MEMR to generate a chip select for the boot EPROM, which occupies 32k of memory address space.

The signals from the AT Bus and the corresponding signals that are generated for the SONIC PROM, EPROM, and Port Page Register are shown in Table II. This table also describes how the board signals are generated.

In addition to generating the chip enables for the SONIC, the control signals from AT are converted to the control signals required by SONIC. *Figure 5* is a timing diagram that shows the AT bus signal timing and the resultant timing generated by the slave interface to the SONIC.

The AT bus initiates an I/O or boot EPROM cycle by asserting BALE active high, generating IOR/IOW or SMEMR and driving the address onto the bus. The card's logic generates the SONIC's -SAS (Slave Address Strobe) and SWR (Slave Write Read) signal which is derived directly from IOR and IOW. The asynchronous state machine is shown in *Figure 6*. This state machine asserts SAS in state S1 once BALE is active and the address has been decoded. It remains in state S1 till IOR or IOW have generated SWR to the SONIC.

Once the SONIC has latched the write data or driven valid read data onto the bus it generates RDYO to terminate the cycle. This is used to generate IOCHRDY to the AT bus.

The exact equations for this state machine can be found in the PAL equations at the end of this note. Note that only BALE and -CSP are used to assert SAS. As IOR or IOW are not used to generate -CSP, some AT memory cycles will assert -SAS but this will have no effect as -CS (Chip select to the SONIC) will not be asserted.

The descriptions of each state are detailed in Table III.

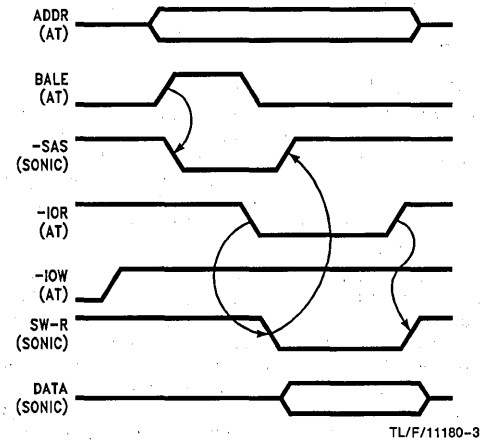


FIGURE 5. -SAS, BALE, IOR and -SWR Timing

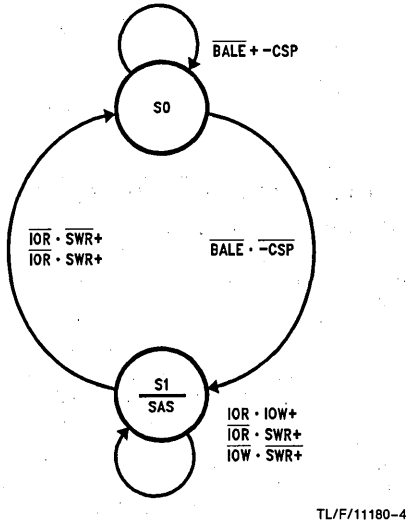
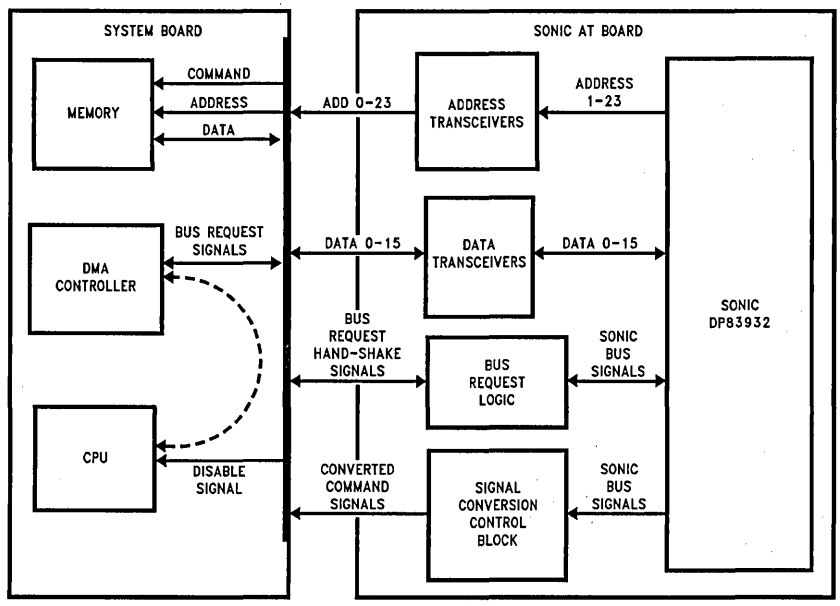


FIGURE 6. Asynchronous State Machine for -SAS

TABLE II. Definition of Slave Signals Generated on the Adapter

Signals	Inputs	Destination	Description
RA<5-3>	Latched SD<5-3>	SONIC	SONIC Register Address
RA<2-0>	SA<3-1>	SONIC	SONIC Register Address
-CARDSEL	SA<9-5>, AEN, HLDA	CARDDEC PAL	SONIC, Port Page and PROM Address Decode
-CSP	CARDSEL, SA4	CARDDEC PAL	SONIC Address Decode
-CSIOP	CARDSEL, SA<4-0>	COMMAND PAL	Port Page Address Decode
-CSPROM	CARDSEL, SA<4-1>	COMMAND PAL	Ethernet PROM Address Decode
-CS	CSP, IOR, IOW	SONIC	SONIC Chip Select
-EN_IOP	CSIOP, IOW	PORT PAGE	Port Page Register Clock Input
-EN_PROM	CSPROM, IORD	PROM	Ethernet PROM Chip Enable
-EPROMRD	SA<19-14>, AEN, HLDA, SMEMRD	EPROM	Boot EPROM Chip Enable
-SAS	!BALE, CSP, IOR, IOW and SWR	SONIC	SONIC Slave Address Strobe
SW-R	IOR, IOW	SONIC	SONIC Slave Read Write
-IO16	CSP	AT BUS	AT 16-Bit I/O Transfer
IOCHRDY	RDYO	AT BUS	AT I/O Channel Ready Signal



TL/F/11180–5

FIGURE 7. Master Logic Block Diagram

TABLE III. The Description for -SAS State Machine's States

State	Description
S0	This is the idle state; the state machine is in this state if BALE is low or -CSP is high.
S1	The state machine enters this state if BALE is asserted high and a SONIC register address is decoded. This causes -SAS to be asserted low. It remains in this state until IOR or IOW are set and they in turn have set SWR to the SONIC

BUS MASTER SECTION

Overview

The AT allows other bus master devices to take over the system bus and use peripherals and memory directly. The SONIC bus master interface to the AT bus consists of two main logic blocks, the Bus Request logic, and the Signal Conversion logic, see *Figure 7*. The Bus Request logic requests access to the bus, while the Signal Conversion logic generates AT compatible read/write signals when the SONIC is a bus master.

When the SONIC requires access to system memory to read/write packet data or descriptor information, the adapter board requests the AT bus through the system DMA controller by setting one of the DMA request lines active. Once the system board issues a DMA acknowledge granting the DMA controller ownership of the AT bus, the adapter board disables the DMA controller off the bus by asserting the MASTER16 signal gaining ownership of the AT bus. This signal handshake is handled by the Bus Request logic.

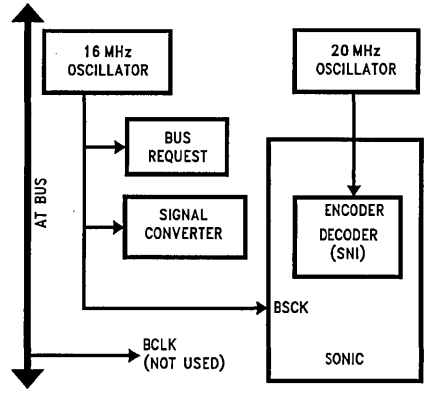
Note: MASTER16 is sometimes also referred to as GRAB.

Once the adapter gains control of the AT bus, the adapter card signal translation logic converts and drives the SONIC cycle command signals onto the AT bus.

Each of the bus master function blocks are described in detail in the following section. First, the adapter card's system clock choice is discussed, followed by a detailed description of the Bus Request and Signal Conversion logic blocks.

Bus Master State Machine Clock Selection

Even though there are two clock signals available on the adapter board, the 20 MHz SNI encoder decoder clock and the AT's system clock (see *Figure 8*), a separate 16 MHz oscillator was added to drive the two synchronous master state machines. The AT's system clock is not reliable enough for critical on-card timing, and can only be used for synchronization. Also, a 16 MHz clock provides a closer and more efficient signal match to the AT timing specifications than the 20 MHz oscillator used for the network interface.



TL/F/11180–6

FIGURE 8. SONIC AT Bus Master Clock Options

Bus Request Logic

The bus request logic enables the adapter card to gain ownership of the AT bus in response to a DMA request from the SONIC.

The AT bus allows devices other than the main CPU to become masters by using the system's DMA controller to assist in the arbitration process. The adapter card user must first choose a spare DMA channel. PC AT's have two 8237A DMA controllers with 4 channels each. Controller 1 contains channels 0–3 supporting byte transfers and these are normally reserved for diskette, SDLC, etc. Controller 2 contains channels 4–7 supporting word transfers. Channel 4 is used to cascade controller 1 but channels 5–7 are normally spare. The adapter card enables the user to select DMA channels 5, 6 or 7 by means of a jumper block which routes the SONIC's DMA request and Acknowledge signals to the appropriate DMA controller channel line (see *Figure 9*).

The Adapter card device driver must program the selected DMA channel by writing to three of the DMA controller 2 internal registers which reside in the PC's I/O space as shown in Table IV. The channel must be programmed for cascade mode. This mode is normally used to enable another DMA controller to be connected to that channel and forces the cascading controller to simply arbitrate for the bus without executing DMA memory or I/O cycles. It must also be programmed to define the sense of the DRQ and DAK lines, the arbitration priority algorithm (fixed or rotating), to enable the particular channel used by clearing its mask bit, and to enable the controller.

TABLE IV. DMA Controller Programming

DMA Register	I/O Addr	Data	Description
Command	D1H	10H	DRQ Active High DAK Active Low Mem to Mem Disable Rotating Priority Controller Enabled
Mode	D6H	D1H D2H D3H	Channel 5 Cascade Channel 6 Cascade Channel 7 Cascade
Write Single Mask Bit	D4H	01H 02H 03H	DRQ5 Enabled DRQ6 Enabled DRQ7 Enabled

The request process is initiated by the SONIC driving its DMA request signal HOLD to one of the system's DMA controller DRQ lines via the on-board jumper block as shown in *Figure 9*. The system board performs the bus arbitration and asserts DAK, granting control of the bus to the adapter card and disabling the system board address data and control lines.

The adapter board routes the DAK signal through the jumper block into the Control PAL which asserts MASTER16 to the AT bus.

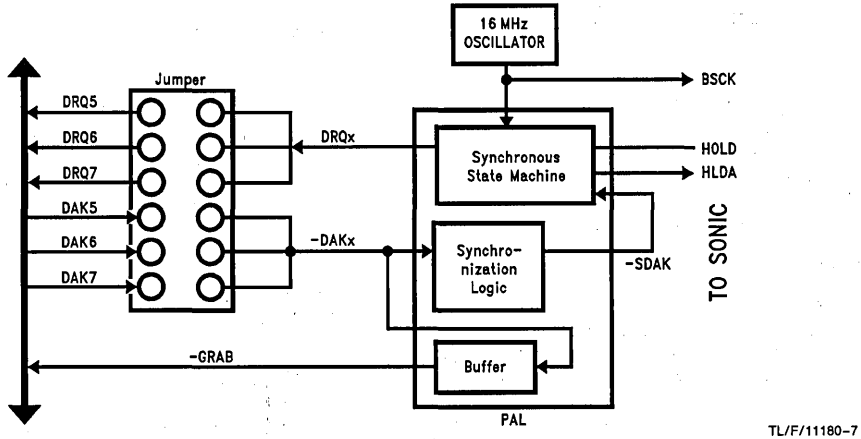
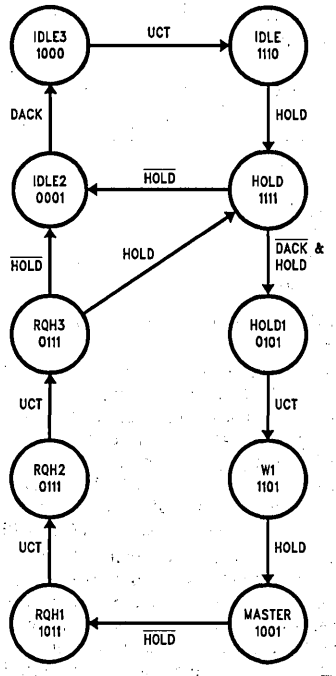


FIGURE 9. Bus Request Logic Block Diagram

The adapter card has been designed to support AT bus clock speeds of 8 MHz and upwards. One important timing constraint is that the bus request logic is required to wait one AT clock after asserting MASTER16 before driving the AT bus. This is done by also inputting the DAK signal into the Bus Request PAL where it is synchronized to the SONIC clock during the state of HOLD1 the Bus Request synchronous state machine, whose state diagram is given in *Figure 10*. *Figure 11* shows the resultant timing for the Bus Request State machine.



TL/F/11180–8

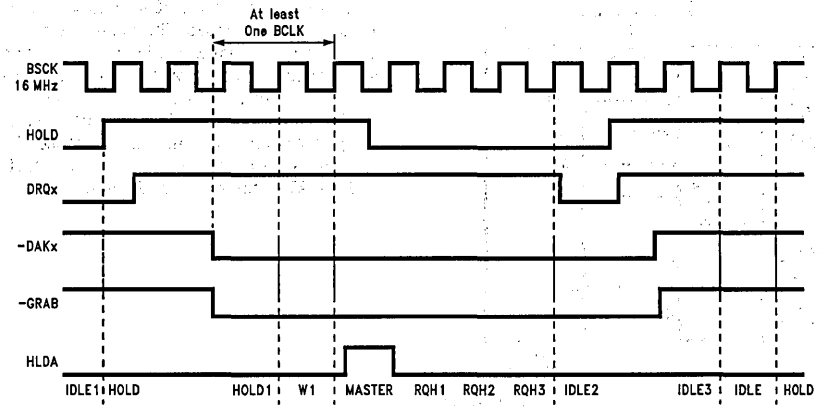
FIGURE 10. Bus Request Synchronous State Machine

The Bus Request state machine initially transitions from the IDLE to the HOLD state when the SONIC asserts its DMA request. Once the system asserts DAK in response to DREQ, MASTER16 is asserted and the state machine transitions to the master state through two wait states, HOLD1 and W1. These provide the minimum delay of two 16 MHz clock beats (equivalent to one 8 MHz AT clock) required by the AT bus logic to disable its drivers following the assertion of MASTER16 before the bus can be driven by the new master. Note that, as already pointed out in the clock selection section, slower AT machines can be supported by adding wait states to the W1 state.

Once in the MASTER state the Bus Request PAL asserts the SONIC DMA acknowledge signal HLDA, enabling the SONIC to initiate its first memory cycle.

The Bus Request state machine will remain in the MASTER state while the SONIC executes its memory cycles. Once the SONIC completes its DMA burst it releases its DMA request HOLD. The state machine will transition through a further 3 wait states, maintaining ownership of the bus by keeping DRQ asserted before resampling the SONIC HOLD signal. If it is deasserted, the state machine transitions to the IDLE2 state. The state machine will remain in this state until the system board clears the DAK signal before transitioning back to the IDLE state. This ensures the state machine is not able to reach the HOLD state again following a quick reassertion of HOLD by the SONIC before the system board has had time to deassert DAK for the previous DMA burst. Once DAK is deasserted, the state machine returns to the IDLE state through a further wait state, IDLE3. Note that both HOLD1 and IDLE3 states result from the assertion/deassertion of DAK, which is an asynchronous input to the state machine. Therefore, state variables were chosen to ensure that any metastability on the changing variables would not cause any temporary erroneous signal decode.

If the HOLD signal is found to have been reasserted by the SONIC when sampled on the RQH3 state, the state machine transitions directly to the HOLD state without relinquishing the bus. This enables the SONIC to execute back-to-back DMA bursts during one continuous AT bus tenure.



TL/F/11180–9

Note: BCLK is the 8 MHz AT bus clock which is equivalent to 2 SONIC Bus Clocks, BSCK.

FIGURE 11. Bus Request Signal Conversion Timing Diagram

This is particularly important during transmission of fragmented packets where accesses to each fragment and descriptor is done in separate DMA bursts.

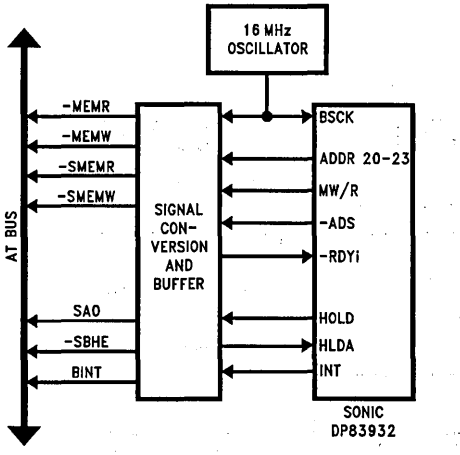
A description of the states that compose the Bus Request State Machine is shown in Table V.

TABLE V. Bus Request State Machine State Descriptions

State	Description
IDLE	SONIC has not requested the bus.
HOLD	When the HOLD signal is asserted, DRQx is generated and the state machine waits for DAK to be asserted. MASTER16 is driven low as soon as DAK is asserted.
HOLD1	This state synchronizes the DAK input.
W1	This state delays the generation of the HLDA signal to the SONIC from MASTER16 to enable the system board to float its drivers.
MASTER	HLDA is asserted during this state. SONIC is the AT bus master and stays in this state until HOLD is deasserted.
RQH1/2/3	These states provide a 3 clock delay before the HOLD signal is sampled.
IDLE2	This state enables the DMA controller to deassert DAK before a new HOLD is able to reassert DRQx.
IDLE3	This state synchronizes the DAK input.

Signal Conversion Logic

Once the SONIC gains ownership of the AT bus it generates read and write cycles to system memory. The Signal Conversion logic is implemented as a synchronous state machine driven by the SONIC control signals and running off the 16 MHz clock. It generates the AT command signals -MEMR, -MEMW, SMEMR and SMEMW.



TL/F/11160–10

FIGURE 12. SONIC AT Bus Signal Conversion

Also, a number of AT's static signals, -M16, -SBHE, BALE, BINT are buffered onto the bus while the SONIC is the bus master. A block diagram showing the basic connection of these functions is shown in *Figure 12*. Both the conversion signals and the static signals are listed in Table VI.

TABLE VI. SONIC Command and Control Signals Conversion

AT BUS	SONIC
COMMAND SIGNAL CONVERSIONS	
-MEMR -MEMW -SMEMR -SMEMW	MW/R, -ADS, IOCHRDY, ADD 20 to 23, HOLD & HLDA
AT STATIC SIGNAL GENERATION	
BALE	HIGH
-M16	LOW
-SBHE	LOW
BINT	INT
BCLK	Not Used
SA0	LOW

SIGNAL CONVERSION SYNCHRONOUS STATE MACHINE

This state machine, running on a 16 MHz clock synchronous to the SONIC, is driven by the SONIC cycle control signals once it has gained ownership of the bus, and the AT IOCHRDY signal indicating the end of the AT cycle. It generates the AT memory cycle signals MEMR, MEMW, SMEMR and SMEMW. The -SMEMR, -SMEMW signals are active only when the first megabyte of memory is accessed. This is determined by decoding address lines 20 thru 23.

The adapter card AT cycle generation logic was originally designed to meet the EISA/ISA bus specification published by the EISA consortium. During prototype testing some AT compatibles were found not to meet one of the timings, namely the MEM asserted to data valid, and the logic was modified to cater to this anomally. This resulted in a typical AT cycle time of six 16 MHz clocks with 3 clocks between cycles. However, prototype testing also showed that for most AT machines the ISA specification timings quoted were over-conservative. Therefore, a jumper selectable option (fast AT speed) was added to the design which allows the user to eliminate some of the original design inherent wait states, reducing the AT cycle times to three 16 MHz clock beats per cycle with 1 clock between cycles. Later in this paper before the PAL equation section is a section that contains a list of the machines used to test the prototype and the AT cycle speed mode required.

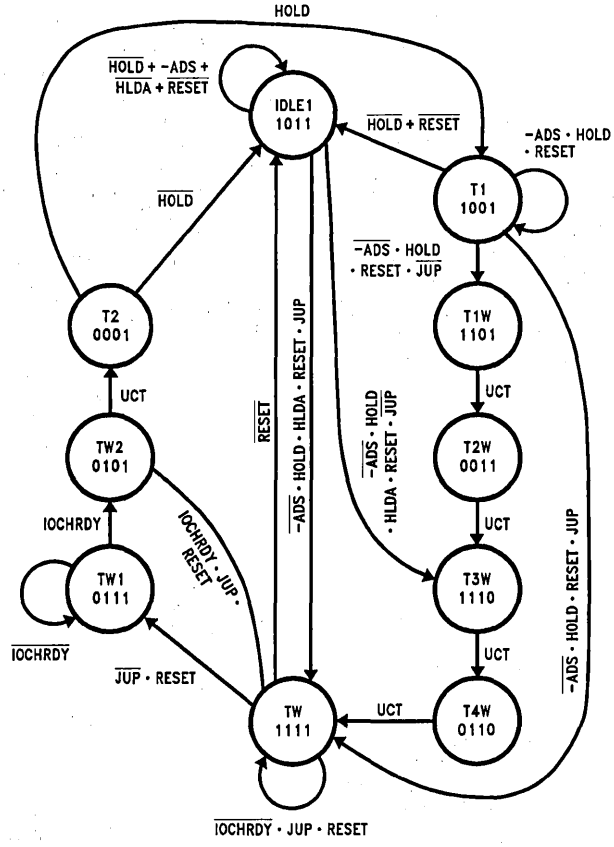
SIGNAL CONVERSION DETAILED DESCRIPTION

This section describes the signal conversion logic when used with the slow AT machine option disabled, showing the operation and timing of the synchronous state machine by running through the first two SONIC/AT DMA cycles in a burst describing the function and timing requirements of each state. This will be followed by a description of the fast AT machine option. *Figure 13* shows the state diagram for both the slow and fast mode. Table VII describes each of the states, and *Figures 15* and *16* later in this document show the resultant timing diagrams for the conversion logic.

The state machine resets into the IDLE state. Once HLDA is asserted by the Bus Request control logic, the SONIC initiates its first cycle, driving its address strobe signal ADS.

The state machine proceeds to states T3W, T4W and TW, driving the corresponding memory control signals to the AT bus and deasserts the SONIC's ready signal RDY to wait state the SONIC.

It then proceeds to state TW1 on the next clock, where the signal CHRDY is sampled. This signal cannot be sampled during TW as the ISA spec quotes a minimum time between MEMR, which is set at the beginning of the TW state, and CHRDY valid of 80 ns.



TL/F/11180–11

FIGURE 13. Signal Conversion Synchronous State Machine

TABLE VII. Signal Conversion State Description

State	Description
IDLE	The state machine waits for the SONIC to initiate the first memory cycle of a DMA burst to proceed to T3W.
T1	This is the first state of all but the first SONIC AT cycles in a DMA burst. ADS from the SONIC is sampled.
T1W	This wait state is required to meet the AT control signals deassertion time between cycles.
T2W	As T1w. These two wait states are skipped for the first AT cycle of a SONIC DMA burst.
T3W	This wait state is required for slow AT memories data enable time.
T4W	As T3w.
TW	This wait state enables the AT slave to drive the CHRDY signal.
TW1	The CHRDY signal is sampled to enable the AT slave to insert wait states.
TW2	This wait state enables the AT slave to drive read data and provides it with write data set up time. The RDY to the SONIC is asserted.
T2	SONIC completes its transfer. It goes to IDLE state, if HOLD is deasserted or to T1 state to initiate another AT cycle.

Note the CHRDY signal is AT generated and asynchronous to the 16 MHz clock. That is the TW1 to TW2 state transition, which occurs once CHRDY is asserted by the AT bus, could generate metastability in the one output that changes, Q2. The TW2 outputs are decoded to generate the AT MEM signals and the SONIC RDY signal. As the MEM signal is active in both the TW1 and TW2 states, switching the Q2 output will not have any effect. The RDY signal is asserted between TW1 and TW2 states but it is not sampled by the SONIC until the next falling edge of the clock by which time the output should have stabilized.

Once CHRDY is sampled high on the TW1 state the state machine proceeds to state TW2 and RDY is asserted to the SONIC.

The SONIC, which is set to run in asynchronous mode, samples RDY on the TW2 state falling edge of the clock, and enters its T2 state on the next rising clock edge at the same time as the state machine transitions to the T2 state.

At the end of the T2 state the state machine samples the SONIC's DMA request signal HOLD. If it is deasserted the state machine will proceed to the IDLE1 state ready for the next DMA cycle burst. If HOLD is still asserted the state machine transitions to the T1 state where the HOLD signal is again sampled for the end of the DMA burst.

If HOLD is active in state T1, the SONIC's address strobe ADS is sampled. When asserted the state machine will proceed through two wait states, T1W and T2W, holding the AT MEM signals deasserted and through three further wait states T3W, T4W, and TW, holding the relevant AT MEM signals asserted onto the TW1 state where again CHRDY from the AT bus is sampled as described above.

The first AT cycle in a SONIC DMA burst will cycle through the following states: IDLE1 - TW3 - TW4 - TW - TW1 - TW2 - T2 (seven 16 MHz clock beats). All other cycles in that burst will cycle through the following states: T1 - T1W - T2W - T3W - T4W - TW - TW1 - TW2 - T2 (nine 16 MHz clock cycles).

SIGNAL CONVERSION TIMING REQUIREMENTS

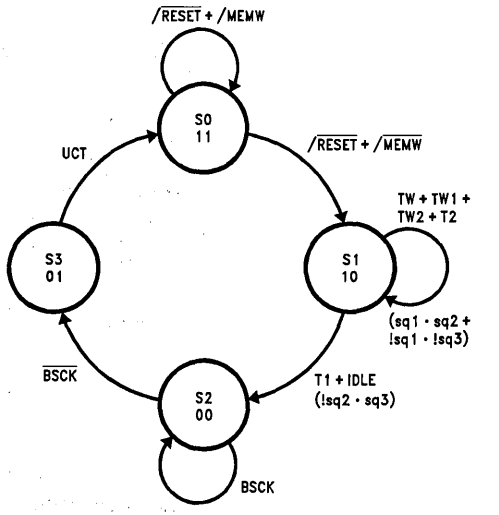
States T1W and T2W are required to meet the ISA specification minimum time for MEM signals deassertion of 170 ns MEM. All AT MEM signals are deasserted during states T1, T1W and T2W.

States T3W and T4W were added to cater to some AT machines which did not meet the ISA specification MEM active to data valid of 194 ns, but required almost 300 ns and did not make use of the CHRDY signal to insert extra wait states.

The ATs MEM signals are asserted during states T3W, T4W, TW, TW1, TW2 and T2.

For read cycles the SONIC latches valid read data on T2's falling clock edge. AT data is available 80 ns maximum from CHRDY assertion during TW1 or 194 ns (300 ns for some AT machines) after MEMRD is asserted in state T3W.

For write cycles the AT requires valid data 212 ns before MEMWT is deasserted with a 25 ns hold time to that trailing edge. As MEMWT is deasserted at the end of T2 the adapter card ensures the write data hold time is met by maintaining the data buffers enabled onto the AT bus for half a 16 MHz clock after the T2 state with the ENW signal asserted during states S1 and S2 in the asynchronous state machine shown in *Figure 14.*



TL/F/11180–12

Note: TW, TW1, TW2, T2, T1 and idle are states of signal converter. The q0 (right bit of state assignment) is ENW signal.

FIGURE 14. Asynchronous State Diagram for ENW Signal

The data buffers direction signal are set to enable data into the adapter board by default and only drive onto the AT bus during adapter card slave read cycles and SONIC master write cycles. Careful consideration was given to ensuring the data direction signals are in a defined state when the data enable signals are active with sufficient set up and hold time to the enabling signals to prevent bus contention on the AT or SONIC buses.

FAST AT MODE OPERATION

In fast AT mode the states T1W, T2W, T3W, T4W and TW1 are eliminated and the AT CHRDY signal is sampled during state TW. A number of AT timings as specified in the EISA/ ISA specification are violated when in this mode, the main ones being MEM asserted to deasserted, MEM deasserted to reasserted, MEMRD asserted to read data valid, write data valid to MEMWT deasserted and MEM asserted to CHRDY valid. This mode, which is user selectable by means of a jumper was found to work in most AT compatible machines and reduces AT cycle times by over 50%.

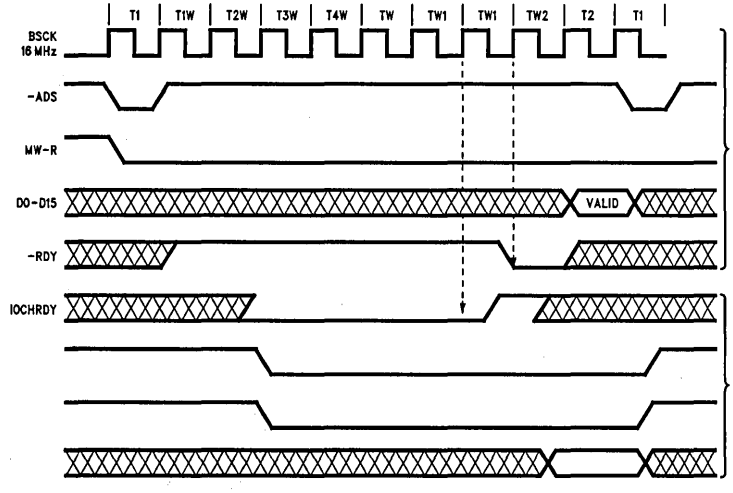
The first AT cycle in a SONIC DMA burst in fast AT mode will cycle through the following states IDLE1 - TW - TW2 - T2 (four 16 MHz clock beats). All other cycles in that burst will cycle through the following states T1 - TW - TW2 - T2 (four 16 MHz clock cycles).

MISCELLANEOUS CONTROL SIGNALS

When the AT card is a bus master, the control signals -SBHE and SA0 are driven low by the Control PAL, see *Figure 12*. The SONIC AT cycles generated are always word wide so SA0 is always driven low and the lines -SBHE and SA0 are used to determine which bytes are being sent. Since SONIC is programmed for 16-bit transfers, it will only transfer 16-bit quantities. It does not perform 8-bit transfers at all. Therefore the address signals SA0 and -SHBE are driven low by a PAL during the master bus cycle.

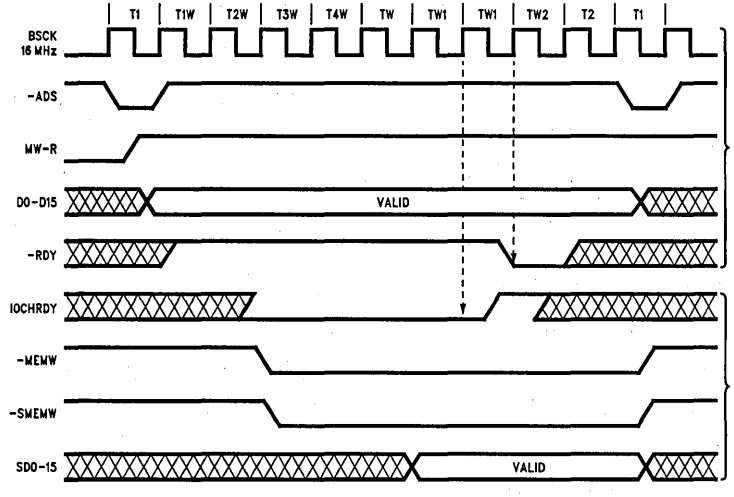
SONIC generates an interrupt to the CPU when one or more unmasked bits are set in ISR. The SONIC asserts the INT signal high. This is buffered through a PAL onto a four bit jumper block which enables the user to select one of four AT interrupt lines, IRQ3 to 5 or IRQ9.

The controls signals for data and address transceivers were designed so that the default direction for the data and address signals is from the AT bus to the SONIC.



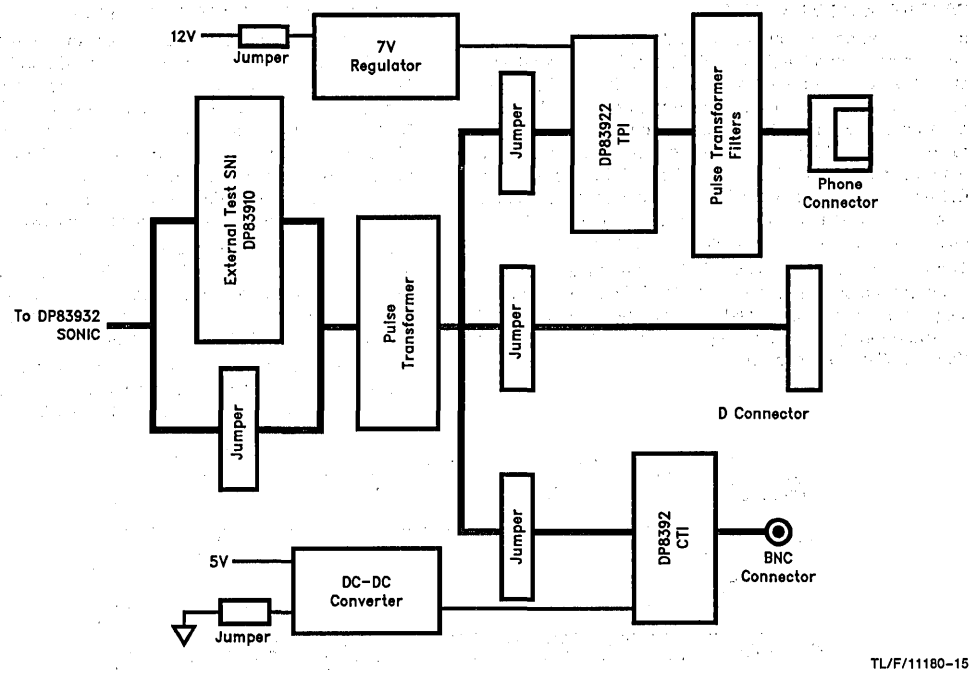
TL/F/11180–13

FIGURE 15. SONIC and AT Bus Signal Timings: Memory Read



TL/F/11180–14

FIGURE 16. SONIC and AT Bus Signal Timings: Memory Write



TL/F/11180–15

FIGURE 17. Detailed Block Diagram of the Physical Interface

NETWORK INTERFACE

Three network interface options are available on the SONIC AT evaluation board. They are the thick-Ethernet, Thin-Ethernet, and the Twisted-Pair, *Figure 17* shows the block diagram for this section. Any one of these three options may be selected by setting the appropriate jumpers and jumper blocks. Opening JPER1 or JPER2 disables the power supply for the Thin Ethernet or Twisted-Pair interfaces respectively. In addition, three jumper blocks, JB5–JB7, select which of the three interfaces is connected. To select an interface, one jumper block is shorted. Only one of the three blocks may be selected at a time. Table VIII shows how each type of interface is selected.

TABLE VIII. Physical Interface Jumpers

JPER1	JPER2	Block Shorted	Interface Type
open	short	JB5	twisted-pair
short**	**open****	**JB6****	**thin-ethernet****
open	open	JB7	thick-ethernet

(**This is the default setting)

In addition to the three interface options, there is a choice of using an external Encoder/Decoder or using the SONIC's internal Encoder/Decoder. To use the SONIC's internal Encoder/Decoder, the external encoder/decoder must be removed from its socket, and JB4 must be shorted. To use the external encoder/decoder, install the DP8391 or the DP83910, and disconnect the JB4 jumpers.

Transceivers are needed to change the differential ECL signals to non-differential ECL signals before entering the cable. The CTI transceiver is provided to do this task for the thin-ethernet, while the TPI is provided for the twisted-pair interface.

Finally, a DC-to-DC converter is needed for the CTI transceiver, and a set of voltage-regulator and heat sink is needed for the TPI transceiver.

PC-AT COMPATIBILITY RESULTS

Table IX lists the PC-AT machines used to test the prototype design. It describes the adapter card AT speed mode jumper option (JUP) required for each machine and the number of AT clock beats per memory cycle.

TABLE IX. Compatibility Testing for DP839EB-ATS

Computer	Fast Cycles (JUP = 1)	Slow Cycles (JUP = 0)
COMPAQ 386/20	works	works
COMPAQ 286/12	works	works
COMPAQ Portable III	works	works
NEC 386SX/16	fails	works
AST386/33	fails	works
IBM AT 6 MHz	works	fails
Tulip 386SX	works	works
EVEREX 386is/16	works	works
EVEREX 386/20	works	works
First Cycle (Note 1)	4 AT clocks	7 AT clocks
Other Cycles (Note 2)	4 AT clocks	9 AT clocks

Note 1: Number of AT bus clocks per cycle for the first cycle in a SONIC DMA burst.

Note 2: Number of AT bus clocks per cycle all other cycles in a SONIC DMA burst.

D. PAL EQUATIONS

D.1 BUS MASTER PAL'S

Three PAL's are used to support the AT bus master operation. These perform the functions of synchronization of asynchronous signals, bus request signal generation, and command signal generation from the synchronous state machine.

PAL U9, BUSREQ:

The BUSREQ PAL is a registered P16R4 PAL. It contains the synchronous state machine for bus request logic, hlda and deco signals generation logic. The registers of the synchronous state machine are driven by the 16Mhz oscillator.

hlda : is an acknowledge signal to inform SONIC that the adapter board has the control of the AT bus.It is asserted high only during the MASTER state of the synchronous state machine. It is asserted two 16MHz clock periods after the system board responds with -DAKx.

enw: It is an output of the asynchronous state machine. It is fed to the Command PAL to generate the appropriate enable signals for the data transceivers during a memory write cycle.

dumw: It is the dummy variable for the enw generation asynchronous state machine.

_rdy: SONIC bus cycles can be lengthen by asserting this signal high during the wait state. According to the Signal Conversion state diagram, this signal has to stay high for T1w, T2w, T3w, T4w, Tw,Tw1 but not Tw2.

```
module bus;
flag '-r3';
title
      'SONIC MASTER BUS REQUEST 1/November/90
      filename: Bus request Pal'

BUS device 'p16R4';

"declarations
      x,z,c = .X.,.Z.,.C.;

      gnd, vcc
      pin 10,20;

"outputs
      q2,q1,q0,q3,dumw,_enw,_rdy,hlda
      pin 17,16,15,14,18,19,13,12;

"inputs to pal
      bsck,Hold,_dakx,sq3,sq2,sq1,bck,_reset,_memw,enb
      pin 1,2,3,4,5,6,7,8,9,11;

"inputs to state machine
input = [_dakx,Hold,_reset];

"states of signal translator:

      idle        =^b1110;
      hold        =^b1111;
      w1          =^b1101;
      master      =^b1001;
      rqh1            =^b1011;
      rqh2            =^b0111;
      rqh3            =^b0001;
      idle2       =^b1000;
```

TL/F/11180–16

```
        idle3           =^b1100;
        hold1           =^b0101;

Idle_st   = [q3,q2,q1,q0] == [1,1,1,0];
Master_st = [q3,q2,q1,q0] == [1,0,0,1];
W1_st = [q3,q2,q1,q0] == [1,1,0,1];

state_diagram [q3,q2,q1,q0]

state idle: case  (input == [x,0,x]) :idle;
                  (input == [x,x,0]) :idle;
                  (input == [x,1,1]) :hold;
                 endcase;

state hold: case  (input == [x,x,0]) :idle;
                  (input == [1,1,1]) :hold;
                  (input == [0,1,1]) :hold1;
                  (input == [x,0,1]) :idle2;
                 endcase;

state hold1:    case    (input == [x,x,0]) :idle;
                       (input == [x,x,1]) :w1;
                       endcase;

state w1:   case    (input == [x,x,0]) :idle;
                  (input == [x,1,1]) :master;
                  (input == [x,0,1]) :idle2;
                 endcase;

state master:   case    (input == [x,x,0]) : idle;
                  (input == [x,1,1]) :master;
                  (input == [x,0,1]) :rqh1;
                 endcase;

state idle3:    case    (input == [x,x,x]) :idle;
                  endcase;

state rqh1:     case    (input == [x,x,0]) :idle;
                       (input == [x,x,1]) :rqh2;
                  endcase;

state rqh2:     case    (input == [x,x,0]) :idle;
                        (input == [x,x,1]) :rqh3;
                  endcase;

state rqh3:     case    (input == [x,x,0]) :idle;
                        (input == [x,1,1]) :hold;
                        (input == [x,0,x]) :idle2;
                  endcase;

state idle2:    case    (input == [x,x,0]) :idle;
                       (input == [1,x,1]) :idle2;
                       (input == [0,x,1]) :idle3;
                        endcase;
equations

      _rdy = (sq3 & sq2 & !sq1) #
            (sq3 & sq2 & sq1) #
            (!sq3 & sq2 & sq1) #
            (!sq3 & !sq2 & sq1);

      hlda = Master_st;
```

TL/F/11180–17

```
            "is asserted to SONIC two BCLK after _grab

       !dumw =!_enw & dumw & !sq2 & sq3 #
              !_enw & dumw & !_reset #
              !_enw & !dumw ;

       !_enw = hlda & _enw & dumw & !_memw & _reset #
              !_enw & dumw & sq1 & sq2 & _reset #
              !_enw & dumw & !sq1 & !sq3 & _reset #
              !_enw & dumw & !_reset #
              !_enw & dumw & !sq2 & sq3 #
              !_enw & !dumw & bck ;
              " _sbhe is implemented in control2 PAL

test_vectors
([bsck,_dakx,Hold,enb,_reset] -> [q3,q2,q1,q0,hlda])

[c,x,x,x,0] -> [1,1,1,0,x];    "IDLE
[c,x,0,0,x] -> [1,1,1,0,0];    "IDLE STATE
"SONIC is in slave cycle
"_aen is active
[c,1,1,0,1] -> [1,1,1,1,0];    "HOLD STATE,_aen is 1
[c,0,1,0,1] -> [0,1,0,1,0];
[c,x,1,0,1] -> [1,1,0,1,0];    "sampled _sdack5 low,_aen is 1
[c,x,1,0,1] -> [1,0,0,1,1];    "W1 STATE,_aen is 1
[c,x,1,0,1] -> [1,0,0,1,1];    "MASTER SATE
[c,0,1,0,1] -> [1,0,0,1,1];    "stay in MASTER STATE
[c,0,x,0,1] -> [1,0,1,1,0];
[c,x,x,0,1] -> [0,1,1,1,0];
[c,x,x,0,1] -> [0,0,0,1,0];
[c,x,0,0,1] -> [1,0,0,0,0];
[c,1,x,0,1] -> [1,0,0,0,0];
[c,0,x,0,1] -> [1,1,0,0,0];
[c,x,x,0,1] -> [1,1,1,0,0];    "IDLE STATE
[c,x,0,0,1] -> [1,1,1,0,0];    "IDLE, eprom is selected
[c,x,1,0,1] -> [1,1,1,1,0];
[c,0,1,0,1] -> [0,1,0,1,0];
[c,x,1,0,1] -> [1,1,0,1,0];
[c,x,1,0,1] -> [1,0,0,1,1];
[c,x,0,0,1] -> [1,0,1,1,0];
[c,x,x,0,1] -> [0,1,1,1,0];
[c,x,x,0,1] -> [0,0,0,1,0];
[c,x,1,0,1] -> [1,1,1,1,0];

test_vectors
([bsck,_dakx,Hold,enb,sq1,sq2,sq3,bck,_memw,dumw,_enw,_reset] ->
[hlda,q2,q1,q0,dumw,_enw])

[c,x,x,x,x,x,x,x,x,x,x,0] -> [0,1,1,0,1,1];
[c,x,0,0,x,x,x,x,x,1,1,1] -> [0,1,1,0,1,1];
[c,1,1,0,x,x,x,x,x,1,1,1] -> [0,1,1,1,1,1];
[c,0,1,0,x,x,x,x,x,1,1,1] -> [0,1,0,1,1,1];
[c,x,1,0,x,x,x,x,x,1,1,1] -> [0,1,0,1,1,1];
[c,x,1,0,x,x,x,x,1,1,1,1] -> [1,0,0,1,1,1];
[c,0,1,0,1,1,x,x,0,1,1,1] -> [1,0,0,1,1,0];
[c,0,1,0,0,x,0,x,x,1,0,1] -> [1,0,0,1,1,0];
[c,0,1,0,1,1,x,x,x,1,0,1] -> [1,0,0,1,1,0];
[c,0,1,0,x,0,1,1,x,1,0,1] -> [1,0,0,1,0,0];
[c,0,1,0,x,x,x,1,x,0,0,1] -> [1,0,0,1,0,0];
[c,0,1,0,x,x,x,0,1,0,0,1] -> [1,0,0,1,1,1];
end bus;
```

TL/F/11180–18

PAL U8, SIGCONV:

The master signal translator is optimized for different AT speed. It is implemented with a 16R4 PAL and the primary outputs are _memr, _memw, _smemr, _smemw. These command outputs are only enabled during the bus master operation.

The registers of the signal translator state machine are clocked with a fixed frequency oscillator (16Mhz). During the SONIC MEMORY access, different number of wait-states are inserted in the state machine for various types of ATs by sampling the IOCHRDY signal.

_memr,_memw: They are active for a certain number of states for a certain AT speed in order to match to the AT timing specification. They are decoded from the states of the signal translator synchronous state machine.

_smemr, _smemw: They are only active when the first megabyte of memory is being accessed. It is decoded with the deco signal from the control2 PAL. They are only active when address 20 thru 23 are all 0.

```
module si3; flag `-r3`; title 'SONIC AT BUS MASTER SIGNAL TRANSLATOR PAL
19/April/90
file name: si3.abl' SI3 device 'p16r4';

"declarations
 c,z,x = .C.,.Z.,.X.;

          gnd,vcc pin 10,20;

"outputs

          q3,q2,q1,q0,_memr,_memw,_smemr,_smemw pin
          17,16,15,14,12,18,13,19;

"inputs to PAL
          bsck,_ads,mwr,hlda,hold,iochrdy,_reset,deco,jup,enb pin
          1,2,3,4,5,6,7,8,9,11;

"inputs to state machine
          input = [_ads,hlda,hold,iochrdy,jup,_reset];

"states of signal translator
          idle      =         ^b1011;
          t1        =         ^b1001;
          t1w       =         ^b1101;
          t2w       =         ^b0011;
          t3w       =         ^b1110;
          t4w       =         ^b0110;
          tw        =         ^b1111;
          tw1       =         ^b0111;
          tw2       =         ^b0101;
          t2        =         ^b0001;

Idle      =         [q3,q2,q1,q0]       == [1,0,1,1];
T1        =         [q3,q2,q1,q0]       == [1,0,0,1];
T1W       =         [q3,q2,q1,q0]       == [1,1,0,1];
T3W       =         [q3,q2,q1,q0]       == [1,1,1,0];
T4W       =         [q3,q2,q1,q0]       == [0,1,1,0];
TW        =         [q3,q2,q1,q0]       == [1,1,1,1];
TW1       =         [q3,q2,q1,q0]       == [0,1,1,1];
TW2       =         [q3,q2,q1,q0]       == [0,1,0,1];
```

TL/F/11180–19

```
T2        =          [q3,q2,q1,q0]       == [0,0,0,1];

state_diagram [q3,q2,q1,q0]

state idle: case                          (input == [x,0,x,x,x,1]) :idle;
                                              (input == [1,x,x,x,x,x]) :idle;
                                              (input == [x,x,0,x,x,x]) :idle;
                                              (input == [0,1,1,x,0,1]) :t3w;
                                              (input == [0,1,1,x,1,1]) :tw;
                                              (input == [x,x,x,x,x,0]) :idle;
                                              endcase;

state t1: case                            (input == [x,x,x,x,x,0]) :idle;
                                              (input == [x,x,0,x,x,x]) :idle;
                                              (input == [0,x,1,x,0,1]) :t1w;
                                              (input == [1,x,1,x,x,1]) :t1;
                                              (input == [0,x,1,x,1,1]) :tw;
                                           endcase;

state t1w: case                           (input == [x,x,x,x,x,x]) :t2w;
                                              endcase;

state t2w: case                           (input == [x,x,x,x,x,x]) :t3w;
                                              endcase;

state t3w: case                           (input == [x,x,x,x,x,x]) :t4w;
                                              endcase;

state t4w: case                           (input == [x,x,x,x,x,x]) :tw;
                                              endcase;

state tw: case                            (input == [x,x,x,x,0,1]) :tw1;
                                              (input == [x,x,x,1,1,1]) :tw2;
                                              (input == [x,x,x,0,1,1]) :tw;
                                              (input == [x,x,x,x,x,0]) :idle;
                                              endcase;

state tw1: case                           (input == [x,x,x,0,x,x]) :tw1;
                                              (input == [x,x,x,1,x,x]) :tw2;
                                              endcase;

state tw2: case                           (input == [x,x,x,x,x,x]) :t2;
                                              endcase;

state t2: case                            (input == [x,x,1,x,x,x]) :t1;
                                              (input == [x,x,0,x,x,x]) :idle;
                                              endcase;

equations

          enable _memr = hold & hlda & _reset;

          _memr = !(!mwr & T3W #
                      !mwr & T4W # !mwr & TW #
                      !mwr & TW1 # !mwr & TW2
                      # !mwr & T2);
```

TL/F/11180–20

```
        enable _memw = hold & hlda & _reset;

        _memw = !(mwr & T3W #
                  mwr & T4W # mwr & TW #
                  mwr & TW1 # mwr & TW2
                  # mwr & T2);

        enable _smemr = hold & hlda &  _reset;

        _smemr = !(!mwr & T3W & deco #
                   !mwr & T4W & deco # !mwr & TW &
                   deco # !mwr & TW1 & deco # !mwr
                   & TW2 & deco # !mwr & T2 &
                   deco);

        enable _smemw = hold & hlda & _reset;

        _smemw = !(mwr & T3W & deco #
                   mwr & T4W & deco # mwr & TW &
                   deco # mwr & TW1 & deco # mwr
                   & TW2 & deco # mwr & T2 &
                   deco);


test_vectors

([bsck,mwr,deco,_ads,hlda,hold,iochrdy,enb,jup,_reset] ->
[q3,q2,q1,q0,_memr,_memw,_smemr,_smemw]) [c,x,x,x,x,x,x,0,x,x] ->
[1,0,1,1,z,z,z,z];"idle [c,x,x,x,x,0,x,0,x,x] -> [1,0,1,1,z,z,z,z];"idle
[c,x,x,x,0,x,x,0,x,x] -> [1,0,1,1,z,z,z,z];"idle [c,x,x,1,1,1,x,0,x,1] ->
[1,0,1,1,1,1,1,1];"idle [c,1,1,0,1,1,x,0,0,1] -> [1,1,1,0,1,0,1,0];"t3w _ads=0
[c,1,1,x,1,1,x,0,x,1] -> [0,1,1,0,1,0,1,0];"t4w [c,1,1,x,1,1,x,0,0,1] ->
[1,1,1,1,1,0,1,0];"tw jup=0 [c,1,1,x,1,1,x,0,x,1] -> [0,1,1,1,1,0,1,0];"tw1
[c,1,1,x,1,1,0,0,x,1] -> [0,1,1,1,1,0,1,0];"tw1 [c,1,1,x,1,1,1,0,x,1] ->
[0,1,0,1,1,0,1,0];"tw2 [c,1,1,x,1,1,x,0,x,1] -> [0,0,0,1,1,0,1,0];"t2
[c,x,x,x,1,1,x,0,x,1] -> [1,0,0,1,1,1,1,1];"t1 [c,x,x,1,1,1,x,0,x,1] ->
[1,0,0,1,1,1,1,1];"t1 _ads=1 [c,0,0,0,1,1,x,0,1,1] -> [1,1,1,1,0,1,1,1];"tw
[c,0,0,x,1,1,x,0,x,1] -> [0,1,1,1,0,1,1,1];"tw1 [c,0,0,x,1,1,1,0,x,1] ->
[0,1,0,1,0,1,1,1];"tw2 [c,0,0,x,1,1,x,0,x,1] -> [0,0,0,1,0,1,1,1];"t2
[c,x,x,x,x,0,x,0,x,x] -> [1,0,1,1,z,z,z,z];"idle

end si3;
```

TL/F/11180–21

PAL U11, CONTROL2:

_grab: This signal is also know as MASTER16. It is pulled low once SONIC becomes the bus master, to disable the DMA from driving signals onto the AT bus.

_adden: It is the enable signal of the transceiver for the upper 8-bit address signal. It is also the direction control signal of the lower 16-bit address transceivers.

_aden: The transceivers for the lower 16 address bits are bidirectional (74AS245). The outputs are controlled by _aden pin. During the master cycle, the addresses go from SONIC to AT bus, and conversely during the slave cycle. The direction control signal for the thransceivers is generated in CONTROL PAL.

deco: Is generated for decoding the command signals: _smemr, _smemw. They are active only when SONIC is addressing the first megabyte of memory. Deco is active when sonic address signals a20..a23 are all zero.

iochrdy: Generated when the CPU is accessing a SONIC register. It is driven inactive until the SONIC drives RDYO active.

sbhe: This signal is driven active whenever the SONIC executes a master cycle on the AT bus as all SONIC master memory cycles are 16 bits wide. It indicates to the slave that there is valid data SD15..8.

```
module con2;
flag '-r2';
title
'PAL20L8, for master control signals.      12/December/90
 filename: con2.abl'
CON2 device 'P20L8';

"declarations
x,z = .X.,.Z.;

gnd,vcc
pin 12, 24;

"outputs
      _aden,deco,_sbhe,_grab,_adden,iochrdy,bint
      pin 15,16,17,18,20,21,22;

"inputs
      _cardsel,_epromsel,_cs,hlda,_dakx,hold,_reset,
      int,a20,a21,a22,a23,_rdyo
      pin 1,2,3,4,5,6,7,8,9,10,11,14,13;

equations

      enable iochrdy = _reset & !_cs;
            iochrdy = !_rdyo;

      enable _grab = hold & _reset;
            !_grab = hold & hlda & _reset #
                  hold & !_dakx & _reset;

      _aden = !((hlda # !_cardsel & _grab # !_epromsel & _grab) & _reset);

      enable _sbhe = hold & hlda & _reset;
            _sbhe = 0;

      _adden = !(hold & hlda & _reset);
            bint = int;
```

TL/F/11180-22

```
      deco = !(a20 # a21 # a22 # a23);

      "control signals in sigtran pal are enabled by memen signal

test_vectors
([_cardsel,_epromsel,hlda,hold,_dakx,_reset,a20,a21,a22,a23]
->[_aden,_sbhe,_grab,_adden,deco])

[x,x,0,0,1,x,0,0,0,0] -> [x,z,z,1,1];
[x,1,x,x,x,0,0,1,1,0] -> [1,z,z,1,0];
[x,x,0,1,1,1,x,x,x,x] -> [x,z,1,1,x];
[x,x,1,1,1,1,x,x,x,x] -> [0,0,0,0,x];
[0,0,1,1,0,1,x,x,x,x] -> [0,0,0,0,x];
[x,x,1,1,0,1,x,x,x,x] -> [x,0,0,0,x];

test_vectors
([_reset,_cs,_rdyo] -> [iochrdy])

[0,x,x] -> [z];
[1,1,x] -> [z];
[1,0,1] -> [0];
[1,0,0] -> [1];

end con2;
```

TL/F/11180-23

D.2 SLAVE PALS

PAL U6, CARDDEC:

This PAL is the main slave interface between the AT and the adapter board. It decodes the address lines from AT and jumpers the user has set up to determine whether the board is selected. If so, further decoding is done to determine if the SONIC, Port Page, or the PROM is selected. When SONIC is selected, the signal -IO16 will also be driven to inform the CPU that all SONIC register accesses will be 16-bit operations.

Description of the output signals:

-cardsel : This output signal determines whether the SONIC AT board is selected by the CPU. It is asserted when the address inputs from the AT and the jumpers on the board correspond to the correct location in the IO space.

-CSP: This output signal is generated from the second round of decoding, which will determine if the SONIC register space is selected.

-CSIOP : This output signal determines if the SONIC Port Page is selected by the CPU.

-CSPROM: This signal is asserted to chip-select the PROM for physical address of the board.

-IO16: This signal is asserted when the SONIC is selected. This is to inform the CPU that the SONIC's 64 internal registers are 16 bits wide, while if either the port page or the the PROM is selected, this signal would not be asserted. This is a tri-state signal.

LA17: This is one of the address lines on the bus: Latchable Address<17>. This PAL acts as an address buffer to this signal when the SONIC is the bus master.

```
module carddec;
flag '-r0';

title
'PAL20L8D
SONIC slave card decode 15/March/89
file name: carddec.abl'

CARDDEC device 'P20L8';

"declarations
          x,z = .X.,.Z.;

          GND,VCC
                    pin 12,24;

          "outputs
          _IO16,_cardsel,_CSP,_CSIOP,_CSPROM,LA17
          pin 15,16,17,18,22,20;

          "inputs
          AEN,SA9,SA8,SA7,SA6,SA5,SA4,SA3,SA2,SA1,SA0
          pin 1,2,3,4,5,6,7,8,9,10,11;
          JP2,JP1,JP0,HLDA,A17
          pin 13,14,19,23,21;

"equates
          addr = [SA9,SA8,SA7,SA6,SA5];
          jpsel = [JP2,JP1,JP0];
```

TL/F/11180–24

```
equations
!_cardsel = !AEN & !HLDA & (jpsel == ^h0) & (addr == ^h8) #
                    !AEN & !HLDA & (jpsel == ^h1) & (addr == ^h9) #
                    !AEN & !HLDA & (jpsel == ^h2) & (addr == ^ha) #
                    !AEN & !HLDA & (jpsel == ^h3) & (addr == ^hb) #
                    !AEN & !HLDA & (jpsel == ^h4) & (addr == ^h18) #
                    !AEN & !HLDA & (jpsel == ^h5) & (addr == ^h19) #
                    !AEN & !HLDA & (jpsel == ^h6) & (addr == ^h1a);

_IO16 = 0;

enable _IO16 = !_cardsel & SA4;

!_CSP = !_cardsel & SA4;

!_CSIOP = !_cardsel & !SA4 & SA3 & SA2 & SA1 & SA0;

!_CSPROM = !_cardsel & !SA4 & !SA3 & !SA2 #
                   !_cardsel & !SA4 & !SA3 & !SA1;

LA17 = A17;

enable LA17 = HLDA;

test_vectors
([AEN,HLDA,JP2,JP1,JP0,SA9,SA8,SA7,SA6,SA5] -> [_cardsel])

[1,0,0,0,0,0,0,0,0,0] -> [1];
[1,1,1,1,1,1,1,1,1,1] -> [1];
[0,0,0,0,0,0,0,0,0,0] -> [1];
[0,1,0,1,0,1,0,1,0,1] -> [1];
[0,0,0,0,0,0,1,0,0,0] -> [0];
[0,0,0,0,1,0,1,0,0,1] -> [0];
[0,0,0,1,0,0,1,0,1,0] -> [0];
[0,0,0,1,1,0,1,0,1,1] -> [0];
[0,0,1,0,0,1,1,0,0,0] -> [0];
[0,0,1,0,1,1,1,0,0,1] -> [0];
[0,0,1,1,0,1,1,0,1,0] -> [0];

test_vectors
([AEN,HLDA,JP2,JP1,JP0,SA9,SA8,SA7,SA6,SA5,SA4,SA3,SA2,SA1,SA0] ->
[_CSP,_CSIOP,_CSPROM,_IO16])

^h4000 -> [1,1,1,z];
^h011a -> [0,1,1,0];
^h0111 -> [0,1,1,0];
^h010f -> [1,0,1,z];
^h010e -> [1,1,1,z];
^h0100 -> [1,1,0,z];
^h0103 -> [1,1,0,z];
^h0107 -> [1,1,1,z];

end carddec;
```

TL/F/11180–25

PAL U7, EPROMDEC:

This PAL decodes the address lines and jumpers the user has set up to determine whether the EPROM is selected. The EPROM is memory-mapped onto the first megabyte of the system memory. If the EPROM is selected, the line _epromrd enables the EPROM to be read.

The -RESET signal for the SONIC is also generated from RESDRV in this PAL, as well as the upper address lines LA18 and LA19 onto the AT bus when SONIC is the bus master.

Description of the output signals:

-EPROMRD: This output signal chip selects the EPROM and enables it to be read by the motherboard at the time of boot-up. It is generated by comparing the address inputs from the AT and the jumpers on the board.

-RESET: This signal acts as an active low reset signal to the SONIC board.

LA<18,19>: These are two of the output address lines when the SONIC is bus master. This PAL acts as an address buffer to enable these lines.

```
module epromdec;

flag '-r0';

title
'PAL20L8 EPROM DECODE on SONIC AT board 15/March/90'

EPROMDEC device 'p2018';

"declarations
         gnd,vcc
         pin 12,24;

         x,z = .X.,.Z.;

"outputs
         _epromrd,LA18,LA19,_RESET
         pin 22,16,17,15;

"inputs
         AEN,_SMEMR,SA19,SA18,SA17,SA16,SA15,SA14,JP2,JP1,JP0
         pin 1,2,3,4,5,6,7,8,9,10,11;
         HLDA,RESDRV,A18,A19
         pin 13,14,18,19;

"equates
         addr = [SA19,SA18,SA17,SA16,SA15,SA14];
         jpsel = [JP2,JP1,JP0];
         ctrl = [AEN,_SMEMR,HLDA];

equations

!_epromrd = (ctrl == ^h0) & (addr == ^h32) & (jpsel == ^h0) #
                      (ctrl == ^h0) & (addr == ^h33) & (jpsel == ^h1) #
                      (ctrl == ^h0) & (addr == ^h34) & (jpsel == ^h2) #
                      (ctrl == ^h0) & (addr == ^h35) & (jpsel == ^h3) #
                      (ctrl == ^h0) & (addr == ^h36) & (jpsel == ^h4) #
                      (ctrl == ^h0) & (addr == ^h37) & (jpsel == ^h5) #
```

TL/F/11180–26

```
                    (ctrl == ^h0) & (addr == ^h38) & (jpsel == ^h6);

!_RESET = RESDRV;

LA18 = A18;

LA19 = A19;

enable LA18 = HLDA;

enable LA19 = HLDA;

test_vectors
([AEN,_SMEMR,HLDA,SA19,SA18,SA17,SA16,SA15,SA14,JP2,JP1,JP0] -> [_epromrd])

[1,x,x,x,x,x,x,x,x,x,x,x] -> [1];
[x,1,x,x,x,x,x,x,x,x,x,x] -> [1];
[x,x,1,x,x,x,x,x,x,x,x,x] -> [1];
[0,0,0,0,0,0,0,0,0,0,0,0] -> [1];
[0,0,0,1,1,1,1,1,1,1,1,1] -> [1];
[0,0,0,1,1,0,0,1,0,0,0,0] -> [0];
[0,0,0,1,1,0,0,1,1,0,0,1] -> [0];
[0,0,0,1,1,0,1,0,0,0,1,0] -> [0];
[0,0,0,1,1,0,1,0,1,0,1,1] -> [0];
[0,0,0,1,1,0,1,1,0,1,0,0] -> [0];
[0,0,0,1,1,0,1,1,1,1,0,1] -> [0];
[0,0,0,1,1,1,0,0,0,1,1,0] -> [0];
[0,0,0,1,1,0,1,1,1,1,1,0] -> [1];

test_vectors
([RESDRV] -> [_RESET])

[1] -> [0];
[0] -> [1];
end epromdec;
```

TL/F/11180–27

PAL U10, COMMAND:

This PAL converts the control signals between the AT and SONIC during slave cycle. The signal SWR is a straight-forward conversion from AT signals -IOR and -IOW, while the signal -SAS is generated by a two-state asynchronous state machine because the correspondent AT signal BALE does not meet the SONIC timing specifications.

This PAL also enables the Port Page and the PROM with appropriate -IOR and -IOW signals after they are chip-selected.

One last task of this PAL is generating the direction and enable signals for the data transceivers.

Descriptions of the output signals:

en_iop: This output signal enables the Port Page register to be written when the Port Page has been addressed.

en_prom: This signal is similar to that of en_iop, except that it enables the PROM to be read.

-sas: This signal is generated by the asynchronous state machine. It signals the SONIC that a valid register address is on the bus.

swr: This signal is directly generated from the AT signals -IOR and -IOW. This output is low when a read and high when a write.

ddir: This is the direction signal for the data transceivers, with 1 being the direction from SONIC to AT.

d_en: This is the enable signal for the lower data byte transceivers. It is only enabled when SONIC is either in master or slave operation and the direction has already been set.

d_enm: This is the enable signal for the higher data byte transceivers. It is disabled for 8 bit transfers.

_cs: This output signal enables the SONIC's registers to be accessed.

Descriptions of the inputs:

_enw: This signal determines whether if the SONIC is a master and it's doing a memory write operation.

```
module cmd;
flag '-r1';

title
'PAL20l0
SONIC slave control signals conversion, 6/June/90
filename: cmd.abl';

CMD device 'P2018';

"declarations
        x,z = .X.,.Z.;

        gnd,vcc
        pin 12,24;

"outputs
        en_iop,en_prom,swr,ddir,d_enm,_sas,_cs,d_en
        pin 15,16,17,18,19,20,21,22;

"inputs
        _ior,_iow,_csp,_csiop,_csprom,_cseprom,hlda,mwr
        pin 1,2,4,5,6,7,8,9;
        bale,_memr,_enw,_reset
```

TL/F/11180–28

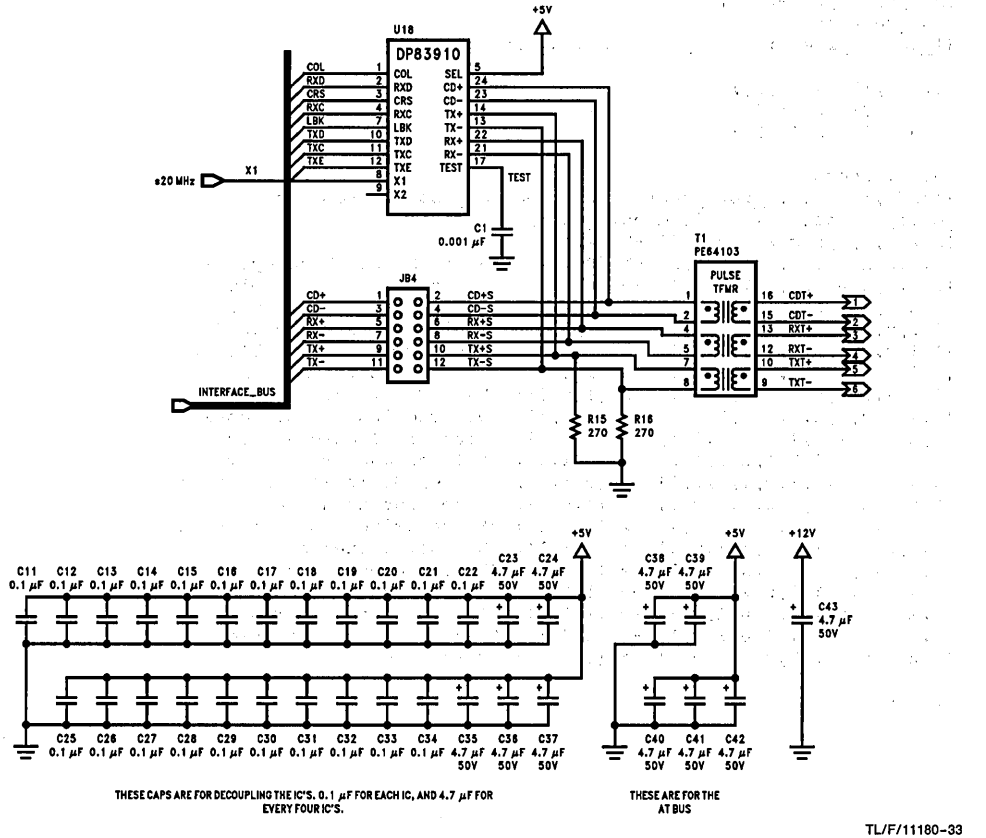
```
        pin 10,11,13,23;

equations

        swr = (_ior # !_iow) & !_csp;

        !_cs = !_csp & !_ior #
                  !_csp & !_iow;

        _sas = !(_sas & bale & !_csp #
                    !_sas & _ior & _iow #
                    swr & !_sas & !_ior #
                    !swr & !_sas & !_iow);

        !d_enm = !_enw & _reset #
                        hlda & !_memr #
                        !_iow & !_csp & _reset #
                        !_cs & _reset #
                        !_iow & !_csiop & _reset;

        en_iop = _csiop # _iow;

        en_prom = _csprom # _ior;

        !ddir = _csp & _csprom & _cseprom & _enw & !mwr #
                   _csp & _csprom & _cseprom & !hlda #
                   _csprom & _cseprom & _ior & _enw & !mwr #
                   _csprom & _cseprom & _ior & !hlda;

        !d_en = !_enw & _reset #
                   hlda & !_memr #
                   !_iow & !_csp & _reset #
                   !_cs & _reset #
                   !_iow & !_csiop & _reset #
                   !_ior & !_csprom & _reset #
                   !_memr & !_cseprom & _reset;

test_vectors
([_csp,_ior,_iow,_reset] -> [_cs,swr])

"Register read
[1,x,x,x] -> [1,x];
[0,1,1,1] -> [1,x];
[0,1,1,1] -> [1,x];
[0,0,1,1] -> [0,0];
[0,0,1,1] -> [0,0];
[0,0,1,1] -> [0,0];

"Register write
[0,1,1,1] -> [1,x];
[0,1,0,1] -> [0,1];
[0,1,0,1] -> [0,1];
[0,1,0,1] -> [0,1];

test_vectors
([_csp,bale,_ior,_iow,swr,_sas] -> [_sas])

[1,x,x,x,x,1] -> [1]; "Not chip selected
[0,0,1,1,x,1] -> [1]; "Chip selected, but bale not valid
[0,1,1,1,x,1] -> [0]; "Bale ready, but no IO command
[0,x,0,1,0,0] -> [1]; "IO command, and swr ready
[0,x,1,0,1,0] -> [1]; "Same as above
```

TL/F/11180–29

```
[x,x,x,x,x,0] -> [1];
test_vectors
([_csiop,_iow,_ior,_csprom] -> [en_iop,en_prom])

[1,x,x,1] -> [1,1];
[1,1,0,0] -> [1,0];
[0,0,1,1] -> [0,1];

test_vectors
([hlda,mwr,_ior,_iow,_memr,_enw,_cseprom,_csp,_csprom,_cs,
_csiop,_reset] -> [ddir,d_en,d_enm])
[x,x,x,x,x,x,x,x,x,x,0] -> [x,1,1];
[1,0,x,x,1,1,1,1,1,1,x] -> [0,1,1]; "Sonic Master Read cycles
[1,0,x,x,0,1,1,1,1,1,1] -> [0,0,0]; "Sonic read data enable
[1,x,x,x,1,1,1,1,1,1,1] -> [x,1,1]; "Sonic master(no rd/wt)
[1,1,x,x,1,1,1,1,1,1,1] -> [1,1,1]; "Sonic starts write in T1
[1,1,x,x,1,0,1,1,1,1,1] -> [1,0,0]; "Sonic wrt, data enable, TW
[1,1,x,x,1,0,1,1,1,1,1] -> [1,0,0]; "Same as above, but TW1
[1,x,x,x,1,0,1,1,1,1,1] -> [1,0,0]; "Next T1 or idle
[1,x,x,x,1,1,1,1,1,1,1] -> [0,1,1]; "dir remains, enable goes
[1,x,x,x,1,1,1,1,1,1,1] -> [0,1,1]; "both dir & enable goes
[0,x,x,x,x,1,1,1,1,1,1] -> [0,1,1]; "Neither master or slave
[0,x,1,1,x,1,1,0,1,1,x,1] -> [0,1,1]; "Slave, Sonic Read selected
[0,x,0,1,x,1,x,0,x,0,x,1] -> [1,0,0]; "same, but data enabled
[0,x,1,0,x,1,1,0,1,0,x,1] -> [0,0,0]; "Slave write, both dir & en
[0,x,1,1,x,1,1,x,1,1,x,1] -> [0,1,1]; "Neither master nor slave
[0,x,1,1,1,1,1,0,x,x,1,x,1] -> [1,1,1]; "Eprom selected, dir set
[0,x,1,1,0,1,0,x,x,1,x,1] -> [1,0,1]; "Same, with data enabled
[0,x,1,1,x,1,1,1,0,1,x,1] -> [1,1,1]; "Prom selected, dir set
[0,x,0,1,x,1,1,1,0,1,x,1] -> [1,0,1]; "Same, with data enabled
[0,x,1,0,x,1,1,x,1,1,0,1] -> [0,0,0]; "Port Page

end cmd;
```

TL/F/11180–30

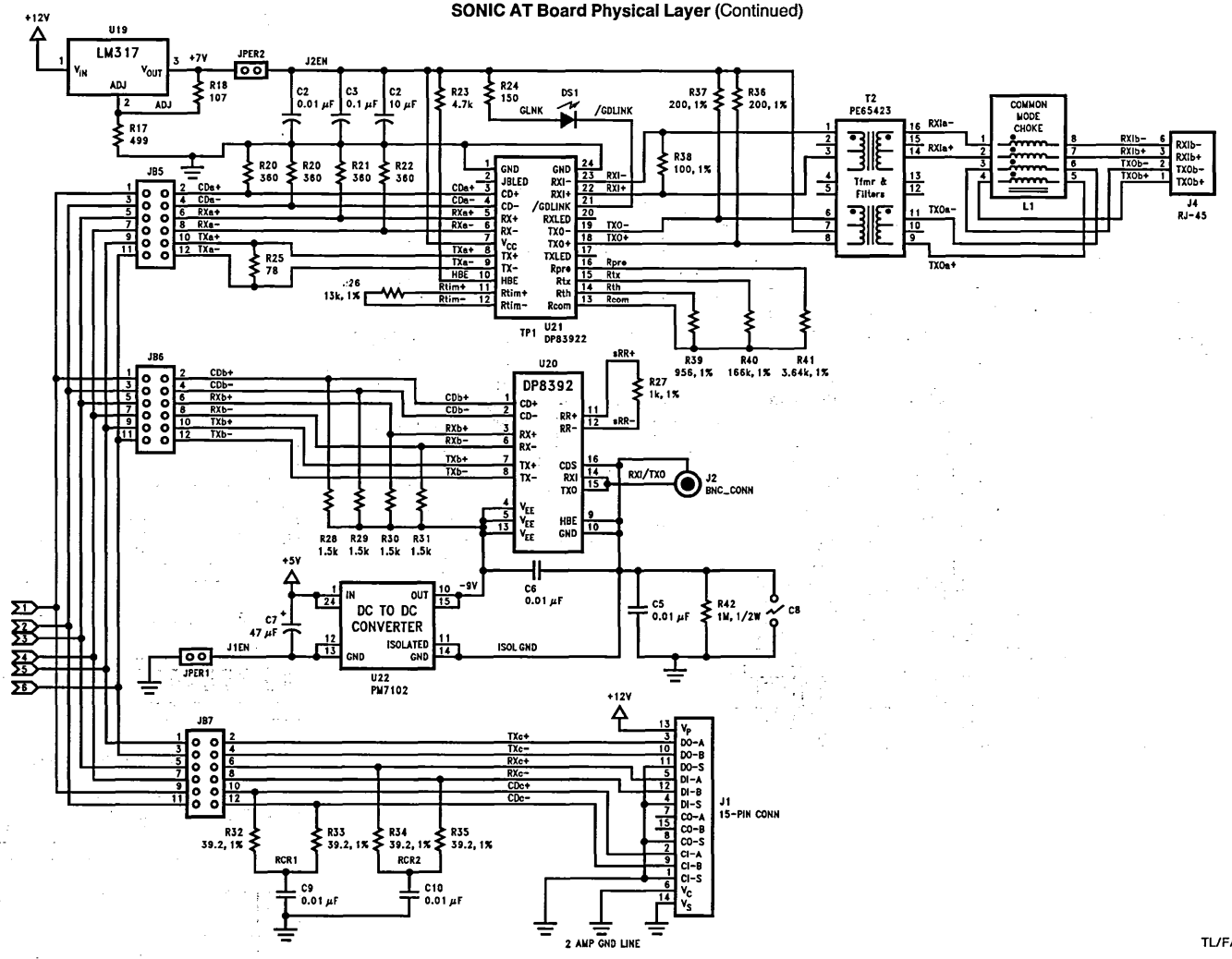
SONIC AT Board Physical Layer



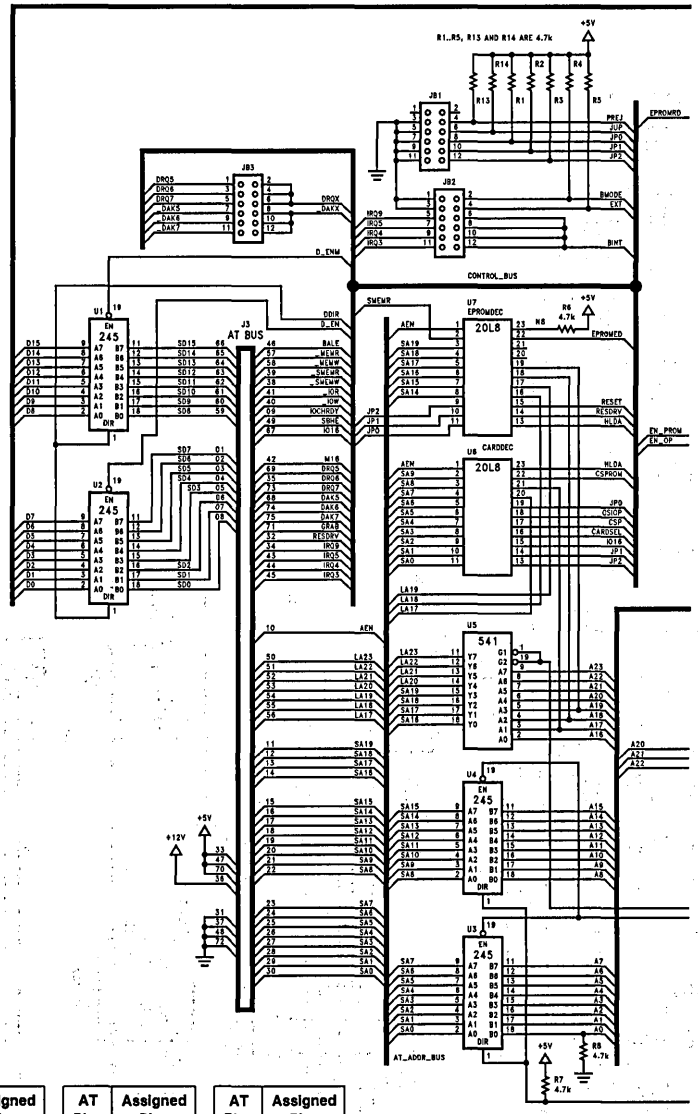
TL/F/11180–33

Note: All resistors to be 5%, ¼W unless otherwise indicated.

SONIC AT Board Physical Layer (Continued)



SONIC AT Board AT Interface



TL/F/11180–32

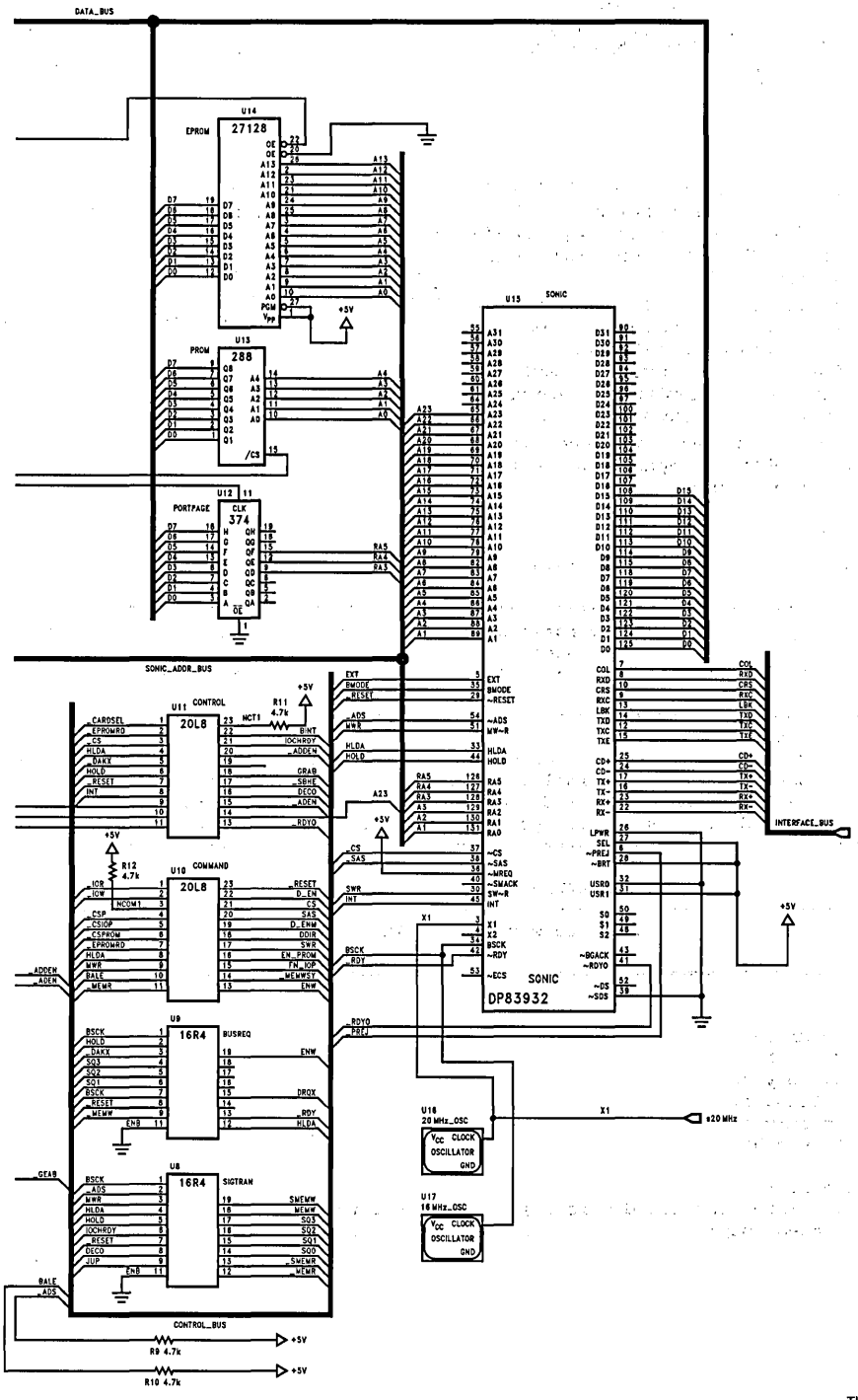
AT Pins	Assigned Pins
A2	1
A3	2
A4	3
A5	4
A6	5
A7	6
A8	7
A9	8
A10	9
A11	10
A12	11
A13	12
A14	13
A15	14
A16	15
A17	16
A18	17
A19	18
A20	19

AT Pins	Assigned Pins
A21	20
A22	21
A23	22
A24	23
A25	24
A26	25
A27	26
A28	27
A29	28
A30	29
A31	30
B1	31
B2	32
B3	33
B4	34
D13	35
B9	36
B10	37
B11	38

AT Pins	Assigned Pins
B12	39
B13	40
B14	41
D1	42
B23	43
B24	44
B25	45
B28	46
B29	47
B31	48
C1	49
C2	50
C3	51
C4	52
C5	53
C6	54
C7	55
C8	56
C9	57

AT Pins	Assigned Pins
C10	58
C11	59
C12	60
C13	61
C14	62
C15	63
C16	64
C17	65
C18	66
D2	67
D10	68
D11	69
D16	70
D17	71
D18	72
D15	73
D12	74
D14	75

SONIC AT Board AT Interface (Continued)



TL/F/11180–34

DP839EB-ATS SONIC Packet Driver for PC/TCP by FTP Software

National Semiconductor
Application Note 748

INTRODUCTION

This is a program listing for a driver for the DP839EB-ATS SONIC Ethernet Adapter. This driver enables the DP839EB-ATS to operate with a TCP/IP software package from FTP Software Inc. called PC/TCP. This driver is written to version 2.0x of this software package.

This software program listing is provided primarily as a programming example for writing software for the DP83932 Systems Oriented Network Interface Controller. This driver is written in Microsoft C 5.1 and Microsoft Assembler 5.1. Since the bulk of the software is written in C, the concepts provided are easily portable to other environments.

This example driver was not written to achieve optimum performance with PC/TCP, but primarily to show how the SONIC Controller can be programmed.

This software does not make use of higher performance upper level features, and performance is limited by this.

The driver is listed by modules in the order listed below.

1. pktdrv.c
2. far.c
3. isr.c
4. sonic.c
5. pktdrv.h
6. sonic.h
7. isrlib.asm
8. pktint.asm
9. pktdrv.mak (make file)

FILENAME: pktdrv.c

```
static char Pktdrv_Sid[]    = "%W%      %G%";
/*
*****************************************************************************
*         Copyright (c) 1990 by National Semiconductor Corporation          *
*                           All Rights Reserved                             *
*****************************************************************************

============================================================================
FILE:       pktdrv.c
NOTES:      This program is a packet driver that provides a common interface
            between PC/TCP's kernel and NSC's SONIC hardware. This program
            was based on a set of drivers provided by Clarkson from FTP.
            This driver is NOT for performance testing due to PC/TCP limitations.
============================================================================
UPDATE LOG:
When/Who                Why/What/Where
----------------------------------------------------------------------------
10/23/90 Mike Lui       Convert to work for SONIC

============================================================================
*/

#include <stdio.h>
#include <dos.h>
#include <memory.h>
#include <string.h>
#include "pktdrv.h"
#include "sonic.h"

  /* externals */
extern void (interrupt far drv_isr)();    /* the interrupt we use */
extern unsigned _psp;                     /* segment address of PSP */
```

TL/F/11142–1

```
  /* Driver information */
static unsigned int     drv_version = 1;  /* driver version */
static unsigned char    drv_class = 1;          /* driver class */
static unsigned int     drv_type = 14;          /* driver type */
static unsigned char    drv_number = 0;         /* driver number */
static unsigned int     drv_funct = 1;          /* basic driver function */
static char drv_name[] =            /* driver name */
    "National Semiconductor SONIC/TCP Packet Driver";
static char cpy_msg[] =
    "Copyright (c) 1990 by National Semiconductor Corporation";
static char drv_rev[] = "1.0";              /* current driver rev */

static HANDLE handle_tbl[MAX_HANDLES];    /* create handle structs */
void (interrupt far *sys_isr)();    /* remember system isr */
char far *pkt_signature = "PKT DRVR";
unsigned int packet_int_no = 0x60;  /* interrupt for communications */
static unsigned far *psp_ptr;       /* pointer to PSP */
unsigned mem_sz;              /* program memory size in paragraphs */

union REGS r_regs;
struct SREGS s_regs;
int send_pending;             /* required for Synernetics */
static int syn_installed;           /* required for Synernetics */

extern int  opterr;
extern int  optind;
extern char *optarg;

/*
 * main()
 *
 *  Main procedure.
 *  Once initialization is complete terminate and stay resident.
 */
main(argc, argv)
int argc;
char *argv[];
{
    psp_ptr = (unsigned far *)((unsigned long)_psp << 16);
    mem_sz = (psp_ptr[1] - _psp);

    init_drv(argc, argv);           /* initialize driver and hardware */

    outp(pagebase, 0);
    outpw(regbase+cr, 8);         /* enable receiver */

    /* terminate and stay resident */
    _dos_keep(0, mem_sz);

}


/*
 * int_handler()
 *
 *  This routine is called from an assembly isr routine "drv_isr"
 *  to handle the application interrupt. The isr routine passes a
 *  set of pointers of the registers to this routine. Register AH
 *  contains which function is to be performed. These registers will
 *  be restored in "drv_isr" before returning from the interrupt.
 *
```

TL/F/11142-2

```
 *    Return values:    If an error occurred the value will be in
 *                 the DH register and the carry bit of cflag
 *                 will be set.
 */
int_handler(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    int ret_val;

    switch(regs->h.ah) {
    case 1:
      ret_val = driver_info(regs, sregs);
      break;
    case 2:
      ret_val = access_type(regs, sregs);
      break;
    case 3:
      ret_val = release_type(regs, sregs);
      break;
    case 4:
      ret_val = send_packet(regs, sregs);
      break;
    case 5:
      ret_val = terminate(regs, sregs);
      break;
    case 6:
      ret_val = get_address(regs, sregs);
      break;
    case 7:
      ret_val = reset_interface(regs, sregs);
      break;
    case 24:
      ret_val = get_stats(regs, sregs);
      break;
    default:
      ret_val = BAD_COMMAND;
    }

    if(ret_val) {
      regs->h.dh = ret_val;              /* put error code into dh */
      regs->x.cflag |= 0x1;              /* and set carry bit */
    }
}

/*
 * driver_info()
 *
 *  Return information on the driver interface. Handle is optional
 *  and is not used in new driver??
 *
 *    Return values:  0 - Success
 */
driver_info(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    regs->x.bx = drv_version;            /* driver version */
    regs->h.ch = drv_class;              /* driver class */
    regs->x.dx = drv_type;               /* driver type */
    regs->h.cl = drv_number;             /* driver number */
    regs->x.si = (unsigned)drv_name;          /* driver name */
```

TL/F/11142–3

```
    sregs->ds  = (unsigned long)((char far *)drv_name) >> 16;
    regs->h.al = drv_funct;                 /* driver function */
    return 0;
}


/*
 * access_type()
 *
 *  Initiate access to packets for the specific type. Since the packet
 *  type field needs to have the bytes of 16 bit values swaped, the
 *  handle will store the type field byte swapped.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
access_type(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    int  i, n,
       open_handle = OPEN,                  /* available handle */
       type_cnt;
    unsigned char type_buf[MAX_TYPE_LEN];

    /* first check a few things to make sure packet access is ok */

    /* check class */
    if(regs->h.al != drv_class) {
      return NO_CLAS;
    }

    /* check type (ours or generic) */
    if(!((regs->x.bx == drv_type) || (regs->x.bx == -1))) {
      return NO_TYPE;
    }

    /* check number (ours or generic) */
    if(!((regs->h.dl == 0) || (regs->h.dl == 1))) {
      return NO_NUMBER;
    }

    /* check packet type length, if too long its not ours */
    if(regs->x.cx > MAX_TYPE_LEN) {
      return TYPE_INUSE;
    }

    /*
     * now check for an available handle and if the handle already
     * exists with same packet type.
     */
    type_ptr = (char far *)(((unsigned long)sregs->ds << 16) | regs->x.si);

    for(i = 0; i < regs->x.cx; i++)
      type_buf[i] = type_ptr[i];

    for(n = 0; n < MAX_HANDLES; n++) {
      if(handle_tbl[n].in_use) {            /* check packet type */
          type_cnt = MIN(regs->x.cx, handle_tbl[n].len);
          if(!far_memcmp((char far *)type_buf,
                         (char far *)handle_tbl[n].type, type_cnt))
            return TYPE_INUSE;              /* duplicate types */
```

TL/F/11142–4

```
        }
        else if(open_handle == OPEN)
            open_handle = n;                    /* grab first open handle */
    }

    if(open_handle == OPEN)
      return BAD_HANDLE;                        /* no available handles */

    /* copy the handle */
    handle_tbl[open_handle].in_use++;

    for(i = 0; i < regs->x.cx; i++) {
      handle_tbl[open_handle].type[i] = type_buf[i];
    }
    handle_tbl[open_handle].len    = regs->x.cx;
    handle_tbl[open_handle].rec_es = sregs->es;
    handle_tbl[open_handle].rec_di = regs->x.di;

    regs->x.ax = open_handle;                   /* return handle */
    return 0;                                   /* return success */

}
/*
 * release_type()
 *
 *  Release access to packets with a particular handle.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
release_type(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    if(chk_handle(regs->x.bx))
      return BAD_HANDLE;

    /* release handle */
    handle_tbl[regs->x.bx].in_use = 0;
    return 0;
}

/*
 * send_packet()
 *
 *  Send packet buffer.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
send_packet(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    char far *frame_ptr;                        /* pointer to frame */
    unsigned long pkt_addr;                         /* physical address of packet */
    unsigned int buf_len;                       /* frame length */
    int i;

    short previous_tda;

    /* check if frame is too big */
```

TL/F/11142–5

```
if(regs->x.cx > BUF_SZ) {
  return NO_SPACE;
}

/* update driver stats */
drv_stats.packets_out++;
drv_stats.bytes_out += regs->x.cx;

/* point to the app's send frame */
frame_ptr = (char far *)(((unsigned long)sregs->ds << 16) |
                         regs->x.si);
pkt_addr = (unsigned long) sregs->ds * 16 + regs->x.si;

buf_len = regs->x.cx;                  /* frame+FC+SNAP length */


/* save current tda */
previous_tda=curtda;

if (transmitactive) {
    /* network is currently busy transmitting, just queue up the tda */
    if (curtda==TDANUM-1) {
    /* load tda with the transmit fragment */
        tda[0].pkt_size=buf_len;
        tda[0].frag_count=1;
        tda[0].frag_ptr0=(unsigned short) pkt_addr;
        tda[0].frag_ptr1=pkt_addr >> 16;
        tda[0].frag_size=buf_len;
        tda[0].link |= 1;
        curtda=0;
    }
    else {
    /* load tda with the transmit fragment */
        tda[curtda+1].pkt_size=buf_len;
        tda[curtda+1].frag_count=1;
        tda[curtda+1].frag_ptr0=(unsigned short) pkt_addr;
        tda[curtda+1].frag_ptr1=pkt_addr >> 16;
        tda[curtda+1].frag_size=buf_len;
        tda[curtda+1].link |= 1;
        curtda++;
    }
    tda[previous_tda].link &= 0x0fffe;
}
else {
    /* network is free */
    retry=0;
    /* load tda with the transmit fragment */
    tda[0].pkt_size=buf_len;
    tda[0].frag_count=1;
    tda[0].frag_ptr0=(unsigned short) pkt_addr;
    tda[0].frag_ptr1=pkt_addr >> 16;
    tda[0].frag_size=buf_len;
    tda[0].link |= 1;
    curtda=0;
    outp(pagebase, 0);
    outpw(regbase+ctda, tda_addr);  /* load ctda */
    transmitactive=1;               /* set network flag to busy */
}
outp(pagebase, 0);          /* get the first page */
outpw(regbase+cr, 2);           /* issue the transmit command */

return 0;
```

TL/F/11142–6

```
}

/*
 * terminate()
 *
 *  Terminate the driver.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
terminate(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    int sonic_irq;

    sonic_irq=3;

    _dos_setvect(packet_int_no, sys_isr); /* put back system isr */

    sonic_isr_disable(sonic_irq);                    /* remove sonic interrupt
*/
    /* free environment memory */
    _dos_freemem(psp_ptr[0x16]);

    /* free memory and return to app */
    if(_dos_freemem(_psp))
      return CANT_TERMINATE;

    return 0;
}


/*
 * get_address()
 *
 *  Get the local net address.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
get_address(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    char buf[6];
    int i, old_mode;
    char far *addr_ptr;                  /* pointer to address */

    if(chk_handle(regs->x.bx))
      return BAD_HANDLE;

    /* get buffer */
    addr_ptr = (char far *)(((unsigned long)sregs->es << 16) | regs->x.di);

    /*
     * copy ethernet address from hardware.
     * regs->x.cx is the length of buffer, fail if address
     * is too big to fit in buffer - NO_SPACE
     */
    if(regs->x.cx < 6)
      return NO_SPACE;
```

TL/F/11142–7

```
    regs->x.cx = 6;

    for(i = 0; i < regs->x.cx; i++) {
      addr_ptr[i] = inp(iobase+i);
    }

    return 0;
}


/*
 * reset_interface()
 *
 *  Reset the interface for the particular handle. If more than one
 *  handle is open return CANT_RESET so other applications (handles)
 *  will not get confused.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
reset_interface(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    char far *addr_ptr;                    /* pointer to address */
    int          i, handle_cnt = 0;

    if(chk_handle(regs->x.bx))
      return BAD_HANDLE;

    /* check if there is more than one handle is open */
    for(i = MIN_HANDLE; i <  MAX_HANDLES; i++)
        if(handle_tbl[i].in_use != 0) handle_cnt++;
    if(handle_cnt > 1)
      return CANT_RESET;

    /* Reset the hardware to a known state */
    /* Will need something maybe ??? */

    return 0;
}


/*
 * get_stats()
 *
 *  Return driver statistics.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
get_stats(regs, sregs)
union REGS far *regs;
struct SREGS far *sregs;
{
    if(chk_handle(regs->x.bx))
      return BAD_HANDLE;

    regs->x.si = (unsigned)&drv_stats;          /* driver stats */
    sregs->ds  = (unsigned long)((char far *)&drv_stats) >> 16;
```

TL/F/11142–8

```
    return 0;
}


/*
 * drv_rcvr()
 *
 *  Receiver procedure. Once a frame is recieved, we need to make two upcall
 *  with the receiving routine provided by the application. The first
 *  call (AX == 0) is to request a buffer to copy the frame to. The second
 *  call (AX == 1) indicates that the frame has been copied.
 *
 *    Return values:     0 - Success
 *                      >0 - Failure
 */
/* void far drv_rcvr()  */
drv_rcvr()
{
    int i;
    int handle_found = OPEN;        /* set if valid frame recieved */
    unsigned char far *frame;
    char far *cp_ptr;


    /* get the frame */
    while (rda[currda].status != 0) {
       frame=(unsigned char far *) (((unsigned long) rda[currda].pkt_ptr1 << 28)
| rda[currda].pkt_ptr0);
      /* validate the received frame */
      for(i = MIN_HANDLE; i < MAX_HANDLES; i++) {
          if((handle_tbl[i].in_use == 0) ||
                        ((((unsigned long)handle_tbl[i].rec_es << 16) |
                                handle_tbl[i].rec_di) == NULL))
            continue;          /* go to next handle */
          if(!far_memcmp((char far *)handle_tbl[i].type,
                  &frame[ETYPE_OFS], handle_tbl[i].len)) {
            handle_found = i;
            break;
          }
      }
      if(handle_found == OPEN) {
          drv_stats.packets_dropped++;
            free_rda();
          continue;
      }

        /* update driver stats */
      drv_stats.packets_in++;
      drv_stats.bytes_in += rda[currda].byte_count;

      /* first upcall, tell them frame size */
      app_recv(0, handle_found, MAX(rda[currda].byte_count-4, 64),
                  (char far *)&cp_ptr, handle_tbl[handle_found].rec_di,
                  handle_tbl[handle_found].rec_es);

      /* check if copy is permitted */
      if(cp_ptr == NULL) {
          drv_stats.packets_dropped++;
            free_rda();
          continue;
      }
```

TL/F/11142-9

```
        /* copy the frame */
        far_memcpy(&cp_ptr[0], &frame[0], rda[currda].byte_count-4);

        /* second upcall, tell them frame has been copied */
        app_recv(1, handle_found, rda[currda].byte_count-4, (char far *)&cp_ptr,
                handle_tbl[handle_found].rec_di,
                handle_tbl[handle_found].rec_es);

        /* free rda */
          free_rda();
    }
    return;
}



/*
 * free_rda()
 *
 * This routine is to free up the currently examined rda for later use
 *
 */

free_rda()
{
    static int first;
    unsigned short tmp_value;

    /* check fifo overrun */
    outp(pagebase, 0);
    if (inpw(regbase+isr) & ISR_RFO)
        outpw(regbase+isr, ISR_RFO);

    /* reinitialize the rda */
    rda[currda].status=0;
    rda[currda].byte_count=0;
    rda[currda].pkt_ptr0=0;
    rda[currda].pkt_ptr1=0;
    rda[currda].in_use=0x0ffff;
    rda[currda].pkt_link |= 1;

    /* link the previous rda to the current rda */
    if (currda==0)
        rda[RDANUM-1].pkt_link&=0x0fffe;
    else
        rda[currda-1].pkt_link&=0x0fffe;

    /* get the first buffer number */
    if (!first) {
        previous_seqno=rda[currda].seq_no >> 8;
        first=1;
    }

    /* check whether rba can be reused */
    if (rda[currda].seq_no >> 8 != previous_seqno) {
        previous_seqno=rda[currda].seq_no >> 8;
        tmp_value=rwp_table[cur_rwp];
        if (cur_rwp==2)
            cur_rwp=0;
        else
            cur_rwp++;
        outp(pagebase, 0x18);
```

TL/F/11142–10

```
        outpw(regbase + rwp, tmp_value);
        outp(pagebase, 0);
        tmp_value=inpw(regbase + isr);
        if (tmp_value & ISR_RBE)
            outpw(regbase + isr, ISR_RBE);
    }

    /* check rde */
    outp(pagebase, 0);
    if (inpw(regbase+isr) & ISR_RDE) {
        outpw(regbase+isr, ISR_RDE);
        outp(pagebase, 0x0d);
        tmp_value=inpw(regbase+crda) & 0x0fffe;
        outpw(regbase+crda, tmp_value);
    }

    if (currda == RDANUM-1)
        currda=0;
    else
        currda++;

}


/*
 * init_drv()
 *
 *  Initialize the driver and hardware.
 */
init_drv(argc, argv)
int argc;
char *argv[];
{
    char far *ptr;
    int kill_drv;

    fprintf(stderr,
      "%s -- Version %s\n%s\n", drv_name, drv_rev, cpy_msg);

    kill_drv = do_args(argc, argv);       /* process command line */

    sys_isr = _dos_getvect(packet_int_no);      /* get system isr */
    ptr = (char far *)sys_isr + 3;

    if(kill_drv)                      /* terminate active driver */
      kill_driver(ptr);

    if((ptr != NULL) && (far_strcmp(ptr, pkt_signature) == 0)) {
      fprintf(stderr,
            "Error: a packet driver already exist at interrupt 0x%x\n",
            packet_int_no);
      exit(1);
    }


    _dos_setvect(packet_int_no, drv_isr); /* install driver isr */

    init();         /* init SONIC */

    fprintf(stderr,
      "Packet Driver is using INT 0x%x and %ld bytes of memory\n",
       packet_int_no, (unsigned long)mem_sz * 16);
```

TL/F/11142–11

```
}

/*
 * chk_handle()
 *
 *  Check if handle is valid.
 *
 *    Return values:     0 - Success
 *                    >0 - Failure
 */
chk_handle(handle)
unsigned int handle;
{
    /* check if handle is in range */
    if((handle < MIN_HANDLE) || (handle >= MAX_HANDLES))
      return BAD_HANDLE;

    /* check if handle is in use */
    if(handle_tbl[handle].in_use == 0)
      return BAD_HANDLE;

    return 0;
}

/*
 * kill_driver()
 *
 *  Terminate driver from memory
 *
 *    Return values:     none - exits from program
 */
kill_driver(ptr)
char far *ptr;
{
    if((ptr == NULL) || (far_strcmp(ptr, pkt_signature) != 0)) {
      fprintf(stderr,
            "Error: no packet driver at interrupt 0x%x\n",
            packet_int_no);
      exit(1);
    }
    r_regs.h.ah = 5;
    r_regs.x.bx = 0;
    int86(packet_int_no, &r_regs, &r_regs);
    if(r_regs.x.cflag) {
      fprintf(stderr, "Error: packet driver can not terminate\n");
      exit(1);
    }
    printf("Terminated packet driver at interrupt 0x%x\n", packet_int_no);
    exit(0);
}



/*
 * do_args()
 *
 *  Process program arguments using getopt().
 *
 *    Return values:     0 - Success
 *                    1 - Terminate driver
 */
do_args(argc, argv)
```

TL/F/11142–12

```
int  argc;
char *argv[];
{
    int in, done = 0;
    char *sptr;

    if(argc == 1)              /* use default packet_int_no */
      return 0;

#ifdef MSDOS
    if((sptr = strrchr(*argv, '\\')) != NULL)
      strcpy(*argv, sptr + 1);
    if((sptr = strrchr(*argv, '.')) != NULL)
      *sptr = '\0';
#endif

    while (!done && ((in = getopt(argc, argv, "?hi:t:")) != -1)) {
      switch(in) {
      case 'i':
      case 't':
          if(sscanf(optarg, "0x%x", &packet_int_no) != 1)
            if(sscanf(optarg, "%d", &packet_int_no) != 1) {
                done = 1;
                break;
            }
          /*
          if(!strncmp(optarg, "0x", 2))
            sscanf(&optarg[2], "%x", &packet_int_no);
          else
            sscanf(optarg, "%d", &packet_int_no);
          */
          if((packet_int_no < 0x60) || (packet_int_no > 0x80)) {
            fprintf(stderr,
            "Error: packet_int_no should be in the range 0x60 to 0x80\n");
            exit(1);
          }
          done = 1;
          if(optind == argc) {
            if(in == 't')
                return 1;
            else
                return 0;
          }
          break;
      }
    }
    fprintf(stderr,
      "Usage: %s [-h] [-i packet_int_no] [-t packet_int_no]\n", *argv);
    fprintf(stderr,
      "  -h = this help message\n");
    fprintf(stderr,
      "  -i = set packet interrupt number, default is 0x60\n");
    fprintf(stderr,
      "  -t = terminate packet driver\n");
    exit(1);
}

int  opterr = 1;
int  optind = 1;
char *optarg;
/*
 * getopt() -- Gets options from command line and breaks them up for analysis.
```

TL/F/11142–13

```
 *            It is functionally compatible with the UNIX version.
 * By Ted Thi
 */
getopt(argc, argv, ctrlStr)
int  argc;
char **argv,
     *ctrlStr;
{
  extern char   *strchr();
  register char *s_ptr;
  static int    i;
  if (optind < argc && argv[optind][++i] == '\0') {
    if (i == 1 || ++optind >= argc)
      return(-1);
    i = 1;
  }
  if (i <= 1) {
    if (optind >= argc || (*argv[optind] != '-' && *argv[optind] != '/') ||
      argv[optind][1] == '\0')
      return(-1);
    if (strcmp(argv[optind] + 1, "-") == 0) {
      optind++;
      return(-1);
    }
  }
  if (argv[optind][i] == ':' || (s_ptr = strchr(ctrlStr, argv[optind][i]))
      == NULL) {
    if (opterr)
      fprintf(stderr, "%s: illegal option -- %c\n", *argv, argv[optind][i]);
    return('?');
  }
  if (s_ptr[1] == ':') {
    if (argv[optind][++i] == '\0') {
      i = 0;
      if (++optind >= argc) {
        if (opterr)
          fprintf(stderr, "%s: option requires an argument -- %c\n", *argv,
                  *s_ptr);
        return('?');
      }
    }
    optarg = argv[optind++] + i;
    i = 0;
  } else
    optarg = NULL;
  return(*s_ptr);
}                                       /* of getopt() */
```

TL/F/11142–14

FILENAME: far.c

```
/*
 ****************************************************************************
 *          Copyright (c) 1990 National Semiconductor Corporation           *
 *                            All Rights Reserved                           *
 ****************************************************************************
 */
#include <dos.h>


void far_memcpy(dest, src, cnt)
register char far *dest;
register char far *src;
register unsigned cnt;
{
  while (cnt--) *dest++ = *src++;
}


char far *far_strcpy(s1, s2)
register char far *s1, far *s2;
{
    char far *s3 = s1;
    while (*s2) *s1++ = *s2++;
    return (s3);
}


far_strcmp(s1, s2)
register char far *s1, far *s2;
{
    while(*s1) {
      if(*s1 != *s2) return(*s1 - *s2);
      s1++; s2++;
    }
    return(*s1 - *s2);
}


far_memcmp(s1, s2, cnt)
register char far *s1, far *s2;
register int cnt;
{
    while(--cnt > 0) {
      if(*s1 != *s2)
          return(*s1 - *s2);
      s1++; s2++;
    }
    return(*s1 - *s2);
}
```

TL/F/11142–15

FILENAME: isr.c

```
/*
 ****************************************************************************
 *           Copyright (c) 1990 National Semiconductor Corporation          *
 *                           All Rights Reserved                            *
 ****************************************************************************
 */

#include <dos.h>
#include "sonic.h"

#define ISR_STACK_SZ    2048

static char irq_map[] = {
    0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
    0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77
};

static int pic_ctl;
static int pic_mask;
static int old_mask_val;

void (interrupt far *sys_irq_int)();

void interrupt far sonic_isr();

void sonic_isr_enable(irq)
int irq;
{
    pic_ctl = irq < 8 ? 0x20 : 0xa0;
    pic_mask = pic_ctl + 1;

    old_mask_val = inp(pic_mask);
    sys_irq_int = _dos_getvect(irq_map[irq]);

    _disable();
    _dos_setvect(irq_map[irq], sonic_isr);
    outp(pic_mask, old_mask_val & ~(1 << irq));
    _enable();
}


void sonic_isr_disable(irq)
int irq;
{
    _disable();
    _dos_setvect(irq_map[irq], sys_irq_int);
    outp(pic_mask, old_mask_val);
    _enable();
}


static char far *old_sp;
static char isr_stack[ISR_STACK_SZ];


void interrupt far sonic_isr()
{
    char far *(far get_sp)();
    void (far set_sp)();
    unsigned short isr_reg;
```

TL/F/11142–16

```
    unsigned short activetda;

    outp(pagebase, 0);
    outpw(regbase+imr, 0);          /* unmask the imr */

    old_sp = get_sp();
    set_sp((char far *)isr_stack + ISR_STACK_SZ);

    _enable();

    outp(pagebase, 0);           /* get the right page */
    isr_reg=inpw(regbase+isr);

    while (isr_reg) {
        if (isr_reg & ISR_PKTRX)  {          /* is there a receive */
            outp(pagebase, 0);
            outpw(regbase+isr, ISR_PKTRX);  /* clear receive bit */
            drv_rcvr();                      /* process rda */
        }
        if (isr_reg & ISR_TXDN) {            /* is there is transmit done */
            outp(pagebase, 0);           /* clear transmit done bit */
            outpw(regbase+isr, ISR_TXDN);

            transmitactive=0;
        }
        if (isr_reg & ISR_TXER) {            /* is there a transmit error */
            outp(pagebase, 0);           /* clear transmit error bit */
            outpw(regbase+isr, ISR_TXER);
            if (retry > 10) {                /* if retry 10 and still not succeed
to transmit this tda */
                outp(pagebase, 0);       /* throw away this tda */
                activetda=inpw(regbase+ctda);
                activetda &= 0x0fffe;
                outpw(regbase+ctda, activetda+16);
            }
            else {                           /* try again */
                outp(pagebase, 0);
                outp(regbase+cr, 2);         /* transmit */
            }
        }
        outp(pagebase, 0);
        isr_reg=inpw(regbase+isr);
    }

    _disable();

    set_sp(old_sp);

    outp(pic_ctl, 0x20);

    outp(pagebase, 0);
    outpw(regbase+imr, 0x0700);

}
```

TL/F/11142–17

FILENAME: sonic.c

```
#include "sonic.h"
#include "dos.h"

/*
 * init()
 *
 * This routine is from init_drv() to initialize sonic buffer and sonic
 * registers.
 *
 * Return values: 0 if success
 *                1 if fail
 *
 */

init()
{
    short i;
    unsigned short  cur_loc;
    int sonic_irq;

    /* set up DMA controller */
    outp(0xd0, 0x10);
    outp(0xd6, 0xd2);
    outp(0xd4, 0x02);
    outp(0xde, 0x00);

    /* initialize valuables */
    transmitactive=0;
    curtda=0;
    currda=0;
    sonic_irq=3;
    /* install sonic interrupt */
    sonic_isr_enable(sonic_irq);

    /* initialize sonic register */
    outp(pagebase, 0);        /* set the right page */
    outpw(regbase+cr, 0x94);      /* reset sonic */
    outpw(regbase+dcr, 0x12de);   /* set configuration: 3 wait state
                                                        16-bit data path
                                                        block mode
                                                        8 words receive fifo
                                                        12 words transmit fifo */
    outpw(regbase+cr, 0);         /* out of reset mode */
    outpw(regbase+rcr, 0x2000);
    outpw(regbase+isr, 0x0ffff);  /* reset isr */
    outpw(regbase+imr, 0x700);    /* set mask to xmit done, xmit error and
                                     receive packet */

    init_tda();                 /* init tda */
    init_rda();                 /* init rda */
    init_rra();                 /* init rra */
    init_cam();                 /* init cam */

    /* initialize rwp location table */
    outp(pagebase, 0x15);
    cur_loc=inpw(regbase+rsa);
    for (i=0; i<3; i++) {
        rwp_table[i]=cur_loc;
        cur_loc+=8;
```

TL/F/11142–18

```
    }
    cur_rwp=0;

    /* normal operation */
    outp(pagebase, 0);
    outpw(regbase+cr, 0x100);       /* read rra */

    return(0);
}

/*
 * init_tda()
 *
 * This routine is to link the tda so as to make transmission more
 * efficient. It also initialize the utda and ctda registers.
 *
 */

init_tda()
{
    unsigned short i, u16, l16;
    unsigned long addr32;
    char far *ptr;
    struct SREGS segregs;

    segread(&segregs);      /* Read the segment register value */
    /* link the first nine tda */
    for (i=0; i<TDANUM-1; i++) {
      addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &tda[i+1]));
        u16=addr32>>16;
        l16=(unsigned short)addr32;
        addr32=(unsigned long)u16 * 16 + l16;
        tda[i].config=0x1000;
        tda[i].link=(unsigned short) addr32;
    }

    /* set the last tda link field to the first tda */
    addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &tda[0]));
    u16=addr32>>16;
    l16=(unsigned short)addr32;
    addr32=(unsigned long)u16 * 16 + l16;
    tda[TDANUM-1].link=(unsigned short) addr32;

    /* set the utda and ctda register */
    outp(pagebase,0);                /* get the correct page */
    outpw(regbase+utda, addr32>>16);      /* set utda */
    outpw(regbase+ctda, (unsigned short)addr32);          /* set ctda */
    tda_addr=(unsigned short)addr32;
}




/*
 * init_rda()
 *
 * This routine is to link the rda together. It also initialize the urda and
 * crda registers.
 *
 */
```

TL/F/11142–19

```
init_rda()
{
    unsigned short i, u16, l16;
    unsigned long addr32;
    struct SREGS segregs;

    segread(&segregs);      /* Read the segment register value */

    /* link the rda */
    for (i=0; i<RDANUM-1; i++) {
        addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short)
&rda[i+1]));
        u16=addr32>>16;
        l16=(unsigned short)addr32;
        addr32=(unsigned long)u16 * 16 + l16;
        rda[i].pkt_link=(unsigned short) addr32;
        rda[i].in_use=0x0ffff;
    }

    /* set the last rda link field to the first rda */
    addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &rda[0]));
    u16=addr32>>16;
    l16=(unsigned short)addr32;
    addr32=(unsigned long)u16 * 16 + l16;
    rda[RDANUM-1].in_use=0x0ffff;
    rda[RDANUM-1].pkt_link=(unsigned short) addr32;
    rda[RDANUM-1].pkt_link|=1;                  /* set EOL */

    /* set the urda and crda register */
    outp(pagebase,0x0d);                /* get the correct page */
    outpw(regbase+urda, addr32>>16);      /* set urda */
    outpw(regbase+crda, (unsigned short)addr32);          /* set crda */

}

/*
 * init_rra()
 *
 * This routine is initialize the rra and set rsa, rea, rrp, rwp registers
 *
 */

init_rra()
{
    unsigned short i, u16, l16;
    unsigned long addr32;
    struct SREGS segregs;

    segread(&segregs);      /* Read the segment register value */

    /* initialize the rra slot */
    for (i=0; i<RRANUM; i++) {
        addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &rba[i]));
        u16=addr32>>16;
        l16=(unsigned short)addr32;
        addr32=(unsigned long)u16 * 16 + l16;
        rra[i].buff_ptr0=(unsigned short)addr32;
        rra[i].buff_ptr1=addr32>>16;
        rra[i].buff_wc0=0x800;
        rra[i].buff_wc1=0;
    }
```

TL/F/11142–20

```
    addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &rra[0]));
    u16=addr32>>16;
    l16=(unsigned short)addr32;
    addr32=(unsigned long)u16 * 16 + l16;

    /* set urra, rsa, and rrp */
    outp(pagebase, 0x14);        /* set the right page */
    outpw(regbase+urra, addr32 >> 16);    /* set urra */
    outpw(regbase+rsa, (unsigned short)addr32);          /* set rsa */
    outpw(regbase+rrp, (unsigned short)addr32);          /* set rrp */

    /* set rea and rwp */
    addr32+=24;
    outpw(regbase+rea, (unsigned short) addr32);         /* set rea */
    outp(pagebase, 0x18);
    outpw(regbase+rwp, (unsigned short) addr32);         /* set rwp */
}

/*
 * init_cam()
 *
 * This routine is initialize the cam and set cdp, cdc registers. Also,
 * load the cam.
 *
 */

init_cam()
{
    unsigned short i, u16, l16;
    unsigned long addr32;
    struct SREGS segregs;

    segread(&segregs);      /* Read the segment register value */
    addr32=(((unsigned long) segregs.ds << 16) | ((unsigned short) &cam));
    u16=addr32>>16;
    l16=(unsigned short)addr32;
    addr32=(unsigned long)u16 * 16 + l16;

    outp(pagebase, 0x26);
    outpw(regbase+cdp, (unsigned short) addr32);    /* load cdp */
    outpw(regbase+cdc, 16);                          /* load cdc */

    /* load the cda with node physical address */
    cam.cam_port_info[0].port0=inpw(iobase);
    cam.cam_port_info[0].port1=inpw(iobase+2);
    cam.cam_port_info[0].port2=inpw(iobase+4);
    for(i=0; i<16; i++)
        cam.cam_port_info[i].entry_ptr=i;
    cam.cam_enable=1;                                /* load cam enable */
    /* load cam */
    outp(pagebase, 0);
    outpw(regbase+cr, CMD_LCAM);
    /* to ensure load cam is properly executed and clear LCD bit in isr */
    for (;;) {
        if (inpw(regbase+isr) & ISR_LCD) {
            outpw(regbase+isr, ISR_LCD);
            break;
        }
    }
}
```

TL/F/11142–21

FILENAME: pktdrv.h

```
/*
*****************************************************************************
*         Copyright (c) 1990 by National Semiconductor Corporation          *
*                           All Rights Reserved                             *
*****************************************************************************
*/

/* Packet Driver Error numbers */
#define BAD_HANDLE      1     /* invalid handle number */
#define NO_CLAS         2     /* no interfaces of specified class found */
#define NO_TYPE         3     /* no interfaces of specified type found */
#define NO_NUMBER 4     /* no interfaces of specified number found */
#define BAD_TYPE  5     /* bad packet type specified */
#define NO_MULTICAST    6     /* this interface does not support multicast*/
#define CANT_TERMINATE  7     /* this packet driver cannot terminate */
#define BAD_MODE  8     /* an invalid receiver mode was specified */
#define NO_SPACE  9     /* failed because of insufficient space */
#define TYPE_INUSE      10    /* the type has already been accessed */
                        /* and not released. */
#define BAD_COMMAND     11    /* command out of range, or not implemented */
#define CANT_SEND 12    /* packet couldn't be sent (usually hardware) */
#define CANT_SET  13    /* hardware address couldn't be changed */
                        /* (more than 1 handle open) */
#define BAD_ADDRESS     14    /* hardware address has bad length or format */
#define CANT_RESET      15    /* couldn't reset interface */
                        /* (more than 1 handle open) */


#define RUNT            60    /* smallest legal size packet, no fcs */
#define GIANT           1514  /* largest legal size packet, no fcs */
#define EADDR_LEN 6     /* Ethernet address length. */


#define MAX_HANDLES 10        /* max number of handles at one time */
#define MIN_HANDLE 0          /* handles are 0 thru 9 */
#define MAX_TYPE_LEN   2      /* max packet type length */
#define OPEN  -1        /* available handle */

#define MIN(a,b)  (((a) < (b)) ? (a) : (b))
#define MAX(a,b)  (((a) > (b)) ? (a) : (b))

/* handle structure */
typedef struct _handle {
    int  in_use;        /* non-zero if handle exist */
    char type[MAX_TYPE_LEN];  /* packet type */
    int  len;                 /* packet length */
    unsigned int rec_es;      /* receiver address segment */
    unsigned int rec_di;      /* receiver address offset */
} HANDLE;

static unsigned char bit_swap[256] = {
    0x00, 0x80, 0x40, 0xc0, 0x20, 0xa0, 0x60, 0xe0,
    0x10, 0x90, 0x50, 0xd0, 0x30, 0xb0, 0x70, 0xf0,
    0x08, 0x88, 0x48, 0xc8, 0x28, 0xa8, 0x68, 0xe8,
    0x18, 0x98, 0x58, 0xd8, 0x38, 0xb8, 0x78, 0xf8,
    0x04, 0x84, 0x44, 0xc4, 0x24, 0xa4, 0x64, 0xe4,
    0x14, 0x94, 0x54, 0xd4, 0x34, 0xb4, 0x74, 0xf4,
    0x0c, 0x8c, 0x4c, 0xcc, 0x2c, 0xac, 0x6c, 0xec,
    0x1c, 0x9c, 0x5c, 0xdc, 0x3c, 0xbc, 0x7c, 0xfc,
```

TL/F/11142–22

```
    0x02, 0x82, 0x42, 0xc2, 0x22, 0xa2, 0x62, 0xe2,
    0x12, 0x92, 0x52, 0xd2, 0x32, 0xb2, 0x72, 0xf2,
    0x0a, 0x8a, 0x4a, 0xca, 0x2a, 0xaa, 0x6a, 0xea,
    0x1a, 0x9a, 0x5a, 0xda, 0x3a, 0xba, 0x7a, 0xfa,
    0x06, 0x86, 0x46, 0xc6, 0x26, 0xa6, 0x66, 0xe6,
    0x16, 0x96, 0x56, 0xd6, 0x36, 0xb6, 0x76, 0xf6,
    0x0e, 0x8e, 0x4e, 0xce, 0x2e, 0xae, 0x6e, 0xee,
    0x1e, 0x9e, 0x5e, 0xde, 0x3e, 0xbe, 0x7e, 0xfe,
    0x01, 0x81, 0x41, 0xc1, 0x21, 0xa1, 0x61, 0xe1,
    0x11, 0x91, 0x51, 0xd1, 0x31, 0xb1, 0x71, 0xf1,
    0x09, 0x89, 0x49, 0xc9, 0x29, 0xa9, 0x69, 0xe9,
    0x19, 0x99, 0x59, 0xd9, 0x39, 0xb9, 0x79, 0xf9,
    0x05, 0x85, 0x45, 0xc5, 0x25, 0xa5, 0x65, 0xe5,
    0x15, 0x95, 0x55, 0xd5, 0x35, 0xb5, 0x75, 0xf5,
    0x0d, 0x8d, 0x4d, 0xcd, 0x2d, 0xad, 0x6d, 0xed,
    0x1d, 0x9d, 0x5d, 0xdd, 0x3d, 0xbd, 0x7d, 0xfd,
    0x03, 0x83, 0x43, 0xc3, 0x23, 0xa3, 0x63, 0xe3,
    0x13, 0x93, 0x53, 0xd3, 0x33, 0xb3, 0x73, 0xf3,
    0x0b, 0x8b, 0x4b, 0xcb, 0x2b, 0xab, 0x6b, 0xeb,
    0x1b, 0x9b, 0x5b, 0xdb, 0x3b, 0xbb, 0x7b, 0xfb,
    0x07, 0x87, 0x47, 0xc7, 0x27, 0xa7, 0x67, 0xe7,
    0x17, 0x97, 0x57, 0xd7, 0x37, 0xb7, 0x77, 0xf7,
    0x0f, 0x8f, 0x4f, 0xcf, 0x2f, 0xaf, 0x6f, 0xef,
    0x1f, 0x9f, 0x5f, 0xdf, 0x3f, 0xbf, 0x7f, 0xff,
};
#define         BIT_SWAP(a)          bit_swap[(unsigned char )(a)]

#define         BYTE_SWAP(a, b)    { *(a) = *(b+1); *(a+1) = *(b); }

#define BUF_SZ 1514
static unsigned char s_buf[BUF_SZ];

static unsigned char snap[] =
    /* SNAP */
    { 170, 170, 3, 0, 0, 0 };

#define ETYPE_OFS 12
#define DATA_OFS 14
#define MAC_LEN 14

static struct {
    unsigned long  packets_in;
    unsigned long  packets_out;
    unsigned long  bytes_in;
    unsigned long  bytes_out;
    unsigned long  errors_in;
    unsigned long  errors_out;
    unsigned long  packets_dropped;
} drv_stats;
```

TL/F/11142–23

FILENAME: sonic.h

```
/* SONIC definition and data structures */

#define     iobase          0x300
#define     pagebase        0x30f
#define     regbase         0x310
#define     TDANUM          5
#define     RDANUM          40
#define     RRANUM          3
#define     RBA_BUF_SIZE    4096

/* isr bit pattern */
#define     CMD_LCAM        0x0200
#define     ISR_RFO         0x0001
#define     ISR_RBE         0x0020
#define     ISR_RDE         0x0040
#define     ISR_PKTRX       0x0400
#define     ISR_TXDN        0x0200
#define     ISR_TXER        0x0100
#define     ISR_LCD         0x1000

/*************************************************
 *                                               *
 * Offset of the register from the i/o base address *
 *                                               *
 *************************************************/

#define cr      0       /* Command */
#define dcr     2       /* Data Configuration */
#define rcr     4       /* Receive Control */
#define tcr     6       /* Transmit Control */
#define imr     8       /* Interrupt Mask */
#define isr     10      /* Interrupt Status */
#define utda    12      /* Upper Transmit Descriptor Addr */
#define ctda    14      /* Current Transmit Descriptor Addr */
#define tps     0       /* Transmit Packet Size */
#define tfc     2       /* Transmit Fragment Count */
#define tsa0    4       /* Transmit Start Address 0 */
#define tsa1    6       /* Transmit Start Address 1 */
#define tfs     8       /* Transmit Fragment Size */
#define urda            10      /* Upper Receive Descriptor Addr */
#define crda            12      /* Current Receive Descriptor Addr */
#define crba0           14      /* Current Receive Buffer Addr 0 */
#define crba1           0       /* Current Receive Buffer Addr 1 */
#define rbwc0           2       /* Remaining Buffer Word Count 0 */
#define rbwc1           4       /* Remaining Buffer Word Count 1 */
#define eobc            6       /* End of Buffer Word Count */
#define urra            8       /* Upper Receive Resource Addr */
#define rsa             10      /* Resource Start Addr */
#define rea             12      /* Resource End Addr */
#define rrp             14      /* Resource Read Addr */
#define rwp             0       /* Resource Write Addr */
#define trba0           2       /* Temp Recv. Buffer Addr 0 */
#define trba1           4       /* Temp Recv. Buffer Addr 1 */
#define tbwc0           6       /* Temp Buffer Word Count 0 */
#define tbwc1           8       /* Temp Buffer Word Count 1 */
#define addr0           10      /* Address Generator 0 */
#define addr1           12      /* Address Generator 1 */
#define llfa            14      /* Last link Field Addr */
#define ttda            0       /* Temp Transmit Descriptor Addr */
```

TL/F/11142–24

```
#define cep             2       /* CAM entry Point */
#define cap2            4       /* CAM Address Port 2 */
#define cap1            6       /* CAM Address Port 1 */
#define cap0            8       /* CAM Address Port 0 */
#define ce              10      /* CAM Enable */
#define cdp             12      /* CAM Descriptor Pointer */
#define cdc             14      /* CAM Descriptor Count */
#define sr              0       /* Silicon Revision */
#define wt0             2       /* Watchdog Timer 0 */
#define wt1             4       /* Watchdog Timer 1 */
#define rsc             6       /* Receive Sequence Counter */
#define crct            8       /* CRC Error Tally */
#define faet            10      /* FAE Error Tally */
#define mpt     12      /* Missed Packet Tally */
#define mdt     14      /* Maximum Deferral Timer */
#define rtc     0       /* Receive Test Control */
#define ttc     2       /* Transmit Test Control */
#define dtc     4       /* DMA Test Control */
#define cc0     6       /* CAM Comparison 0 */
#define cc1     8       /* CAM Comparison 1 */
#define cc2     10      /* CAM Comparison 2 */
#define cm      12      /* CAM Match */
#define reserve1    14  /* Reserved */
#define reserve2    0   /* Reserved */
#define rbc     2       /* Receiver Byte Count */
#define reserve3    4   /* Reserved */
#define tbc     6       /* Transmitter Backoff Counter */
#define trc     8       /* Transmitter Random Counter */
#define tbm     10      /* Transmitter Backoff Mask */
#define reserve4    12  /* Reserved */
#define reserve5    14  /* Reserved */


/* tda structure */
typedef struct tda_construct {
      unsigned short    status;
      unsigned short    config;
      unsigned short    pkt_size;
      unsigned short    frag_count;
      unsigned short    frag_ptr0;
      unsigned short    frag_ptr1;
      unsigned short    frag_size;
      unsigned short    link;
}   tda_struct;

/* rda structure */
typedef struct rda_construct {
      unsigned short    status;
      unsigned short    byte_count;
      unsigned short    pkt_ptr0;
      unsigned short    pkt_ptr1;
      unsigned short    seq_no;
      unsigned short  pkt_link;
      unsigned short    in_use;
}   rda_struct;


/* rra structure */
typedef struct rra_construct {
        unsigned short  buff_ptr0;
        unsigned short  buff_ptr1;
        unsigned short  buff_wc0;
```

TL/F/11142–25

```
        unsigned short  buff_wcl;
}   rra_struct;

/* rba structure */
typedef struct rba_construct {
        unsigned char   buff[RBA_BUF_SIZE];
}   rba_struct;

typedef struct cam_port {
        unsigned short  entry_ptr;
        unsigned short  port0;
        unsigned short  port1;
        unsigned short  port2;
} cam_port_struct;

typedef struct cam_construct {
        cam_port_struct cam_port_info[16];
        unsigned short  cam_enable;
} cam_struct;

rba_struct   rba[RRANUM];
tda_struct   tda[TDANUM];
rda_struct   rda[RDANUM];
rra_struct   rra[RRANUM];
cam_struct   cam;

short transmitactive;              /* transmission currently active flag */
short curtda;                      /* current tda */
short currda;                      /* current rda */
short previous_seqno;              /* previous sequence number */
short retry;                       /* transmit retry counter */
unsigned short rwp_table[6];       /* RRA location table structure */
short cur_rwp;                     /* pointer to rwp_table */
unsigned short tda_addr;            /* tda starting address */
unsigned char far *type_ptr;       /* pointer for packet type */
```

TL/F/11142-26

FILENAME: lsrlib.asm

```
;*****************************************************************************
;*          Copyright (c) 1990 National Semiconductor Corporation            *
;*                            All Rights Reserved                            *
;*****************************************************************************

_TEXT SEGMENT  WORD PUBLIC 'CODE'
_TEXT ENDS
_DATA SEGMENT  WORD PUBLIC 'DATA'
_DATA ENDS
CONST SEGMENT  WORD PUBLIC 'CONST'
CONST ENDS
_BSS  SEGMENT  WORD PUBLIC 'BSS'
_BSS  ENDS
DGROUP      GROUP CONST, _BSS, _DATA
      ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP

_TEXT segment     word public 'CODE'
      assume      cs:_TEXT

      public      _get_sp
_get_sp           proc far
      mov   ax,sp
      add   ax,4
      mov   dx,ss
      ret
_get_sp           ENDP

      public      _set_sp
_set_sp           proc far
      mov   bx,ss
      mov   es,bx
      mov   bx,sp

      pushf
      cli
      pop   dx

      mov   sp,word ptr ss:[bx+4]
      mov   ss,word ptr ss:[bx+6]

      and   dx,512
      jz    skip
      sti

skip: sub   sp,4
      mov   ax, word ptr es:[bx+2]
      push  ax
      mov   ax, word ptr es:[bx]
      push  ax
      ret
_set_sp           ENDP

      public      _get_if
_get_if           proc far
      pushf
      pop   dx
      mov   ax,0
      and   dx,512
      jz    ifret
```

TL/F/11142-27

```
        mov     ax,1
ifret:  ret
_get_if         ENDP

ARG_OFS     equ   6                 ;near = 4, far = 6 (from bp)
        public      _int_fddi
_int_fddi   proc far
        push  bp
        mov   bp, sp
        sub   sp, 8             ;work area for INT code

        ;put INT code on stack
        mov   byte ptr[bp - 2], 0cbh
        mov   ax, word ptr[bp + ARG_OFS]
        mov   [bp - 3], al
        mov   byte ptr[bp - 4], 0cdh
        mov   word ptr[bp - 6], ss
        lea   ax, word ptr[bp - 4]
        mov   word ptr[bp - 8], ax

        ;get regs values off sp, pointers are far
        push  bp
        mov   es, [bp + ARG_OFS + 4]
        mov   bp, [bp + ARG_OFS + 2]
        mov   ax, es:[bp]
        mov   bx, es:[bp + 2]
        mov   cx, es:[bp + 4]
        mov   dx, es:[bp + 6]
        mov   si, es:[bp + 8]
        mov   di, es:[bp + 10]
        pop   bp

        call  dword ptr[bp - 8] ;do INT

        ;get carry bit
        push  ax
        pushf
        pop   ax
        and   ax, 1             ;mask carry bit

        ;put regs values on sp
        mov   es, [bp + ARG_OFS + 8]
        mov   bp, [bp + ARG_OFS + 6]
        mov   es:[bp + 12], ax  ;cflag
        pop   ax
        mov   es:[bp], ax
        mov   es:[bp + 2], bx
        mov   es:[bp + 4], cx
        mov   es:[bp + 6], dx
        mov   es:[bp + 8], si
        mov   es:[bp + 10], di

        add   sp, 8
        pop   bp
        ret
_int_fddi   ENDP

_TEXT   ends
        end
```

TL/F/11142-28

FILENAME: pktint.asm

```
; ****************************************************************************
; *         Copyright (c) 1990 by National Semiconductor Corporation         *
; *                           All Rights Reserved                            *
; ****************************************************************************

      title TEXT - Interrupt service routine

      extrn _int_handler:near

_TEXT SEGMENT  WORD PUBLIC 'CODE'
_TEXT ENDS
_DATA SEGMENT  WORD PUBLIC 'DATA'
_DATA ENDS
CONST SEGMENT  WORD PUBLIC 'CONST'
CONST ENDS
_BSS  SEGMENT  WORD PUBLIC 'BSS'
_BSS  ENDS
DGROUP      GROUP CONST, _BSS, _DATA
      ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP

_DATA SEGMENT     WORD PUBLIC 'DATA'
      assume      ds:DGROUP
rcvr_ptr    dd    ?
segmoffs    struc
offs        dw    ?
segm        dw    ?
segmoffs    ends

_DATA ENDS


_TEXT segment     word public 'CODE'
      assume      cs:_TEXT

CFLAG_OFFSET      equ   2
FLAG_OFFSET equ   6
REGS_OFFSET equ   14
SREGS_OFFSET      equ   22


      public _drv_isr
_drv_isr    proc  far

      jmp   start
      db    'PKT DRVR',0            ;driver signature

;setup registers on stack for MSC's union REGS and struct SREGS
start:


      assume      ds:nothing

      push  bp
      mov   bp, sp
      and   word ptr[bp+FLAG_OFFSET], not 1     ;clear carry bit
      push  word ptr[bp+FLAG_OFFSET] ;put in cflag field of structure
      push  di                ;save regular registers
      push  si
      push  dx
```

TL/F/11142-29

```
        push  cx
        push  bx
        push  ax
        push  ds                ;save segment registers
        push  ss
        push  cs
        push  es

        push  ss
        lea   ax, word ptr [bp-SREGS_OFFSET]      ;pass sregs pointer
        push  ax
        push  ss
        lea   ax, word ptr [bp-REGS_OFFSET] ;pass regs pointer -> ax
        push  ax

        mov   ax, DGROUP              ;get global data segment
        mov   ds, ax                        ;make segment addressable
        assume ds: DGROUP
        cld
        call  _int_handler                  ;call C interrupt handler
        add   sp, 8

        mov   ax, word ptr[bp-CFLAG_OFFSET] ;mov cflag to flag reg
        mov   word ptr[bp+FLAG_OFFSET], ax

        pop   es                      ;restore registers
        pop   ax                      ;dummy pop for cs
        pop   ss
        pop   ds
        pop   ax
        pop   bx
        pop   cx
        pop   dx
        pop   si
        pop   di
        pop   bp                      ;pop cflag of structure
        pop   bp

        iret                          ;return from interrupt
_drv_isr    endp


        public _app_recv
_app_recv  proc near
ax_ofs      equ   4

        assume  ds:DGROUP
        push  bp
        mov   bp, sp
        push  ds
        push  es
        push  bx

        mov   bx, [bp+ax_ofs+10]      ;set-up app reciever
        mov   rcvr_ptr.offs, bx
        mov   bx, [bp+ax_ofs+12]
        mov   rcvr_ptr.segm, bx

        les   bx, dword ptr[bp+ax_ofs+6] ;buffer
        mov   si, word ptr es:[bx]
        push  ds
        mov   ds, word ptr es:[bx+2]
```

TL/F/11142–30

```
        mov   ax, [bp+ax_ofs]
        mov   bx, [bp+ax_ofs+2]
        mov   cx, [bp+ax_ofs+4]
        pop   es
        assume  es:DGROUP

        call  es:rcvr_ptr

        mov   ax, es
        les   bx, dword ptr[bp+ax_ofs+6]    ;update pointer ES:DI
        mov   word ptr es:[bx], di
        mov   word ptr es:[bx+2], ax
        pop   bx
        pop   es
        pop   ds
        pop   bp
        ret                                 ;return

_app_recv   endp

_TEXT   ends
        end
```

TL/F/11142-31

FILENAME: pktdrv.mak

```
ZI     =
INC    = ..\include
CFLAGS       = $(ZI) -Gs -c -I$(INC)
MFLAGS       = -Ml

OBJ    = pktdrv.obj sonic.obj pktint.obj far.obj isr.obj isrlib.obj
#LIB   = ..\lib\frame.lib
LIB      =

sonic.obj:  sonic.c $(INC)\sonic.h
    cl $(CFLAGS) $*.c

pktdrv.obj: pktdrv.c $(INC)\pktdrv.h $(INC)\sonic.h
    cl $(CFLAGS) $*.c

far.obj: far.c $(INC)\sonic.h
    cl $(CFLAGS) $*.c

isr.obj: isr.c $(INC)\sonic.h
    cl $(CFLAGS) $*.c

isrlib.obj: isrlib.asm
    masm $(MFLAGS) $*.asm;

pktint.obj: pktint.asm
    masm $(MFLAGS) $*.asm;

pktdrv.exe: $(OBJ)
    cl $(ZI) $(OBJ)  -o $*

#pktdrv.exe: $(OBJ)
#     link /CO /LI /MAP $(OBJ), $*, ,$(LIB);
#     msym pktdrv
```

TL/F/11142–32

Section 2

Ethernet Physical Layer Transceivers

Section 2 Contents

National Semiconductor

DP8392C/DP8392C-1
Coaxial Transceiver Interface

General Description

The DP8392C Coaxial Transceiver Interface (CTI) is a coaxial cable line driver/receiver for Ethernet/Thin Ethernet (Cheapernet) type local area networks. The CTI is connected between the coaxial cable and the Data Terminal Equipment (DTE). In Ethernet applications the transceiver is usually mounted within a dedicated enclosure and is connected to the DTE via a transceiver cable. In Cheapernet applications, the CTI is typically located within the DTE and connects to the DTE through isolation transformers only. The CTI consists of a Receiver, Transmitter, Collision Detector, and a Jabber Timer. The Transmitter connects directly to a 50 ohm coaxial cable where it is used to drive the coax when transmitting. During transmission, a jabber timer is initiated to disable the CTI transmitter in the event of a longer than legal length data packet. Collision Detection circuitry monitors the signals on the coax to determine the presence of colliding packets and signals the DTE in the event of a collision.

The CTI is part of a three chip set that implements the complete IEEE 802.3 compatible network node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC).

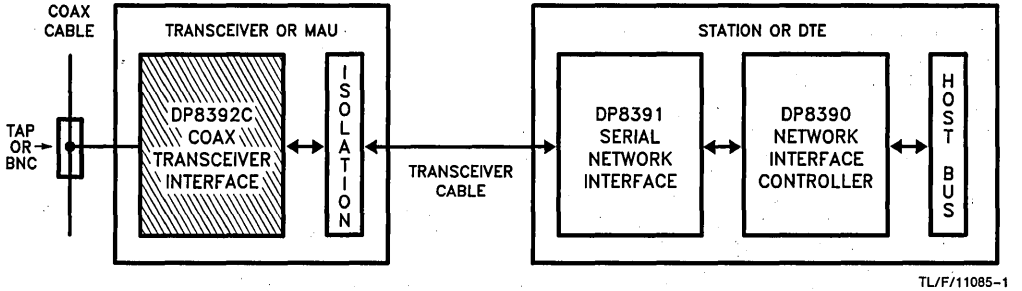
The SNI provides the Manchester encoding and decoding functions; whereas the NIC handles the Media Access Protocol and the buffer management tasks. Isolation between the CTI and the SNI is an IEEE 802.3 requirement that can be easily satisfied on signal lines using a set of pulse transformers that come in a standard DIP. However, the power isolation for the CTI is done by DC-to-DC conversion through a power transformer.

Features

- Compatible with Ethernet II, IEEE 802.3 10Base5 and 10Base2 (Cheapernet)
- Integrates all transceiver electronics except signal & power isolation
- Innovative design minimizes external component count
- Jabber timer function integrated on chip
- Externally selectable CD Heartbeat allows operation with IEEE 802.3 compatible repeaters
- Precision circuitry implements receive mode collision detection
- Squelch circuitry at all inputs rejects noise
- Designed for rigorous reliability requirements of IEEE 802.3
- Standard Outline 16-pin DIP uses a special leadframe that significantly reduces the operating die temperature

Table of Contents

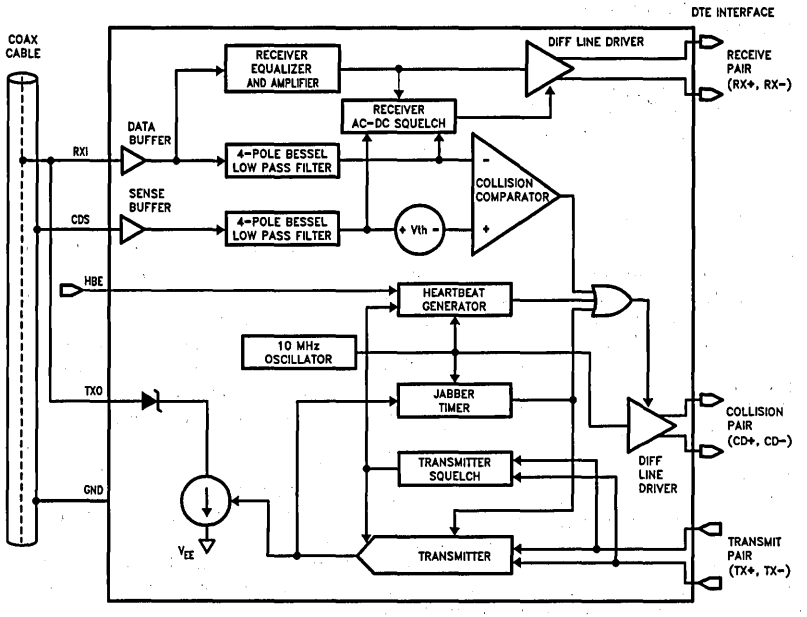
1.0 System Diagram



TL/F/11085–1

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set

2.0 Block Diagram



TL/F/11085-2

FIGURE 1. DP8392C Block Diagram

3.0 Functional Description

The CTI consists of four main logical blocks:

a) the Receiver - receives data from the coax and sends it to the DTE

b) the Transmitter - accepts data from the DTE and transmits it onto the coax

c) the Collision Detect circuitry - indicates to the DTE any collision on the coax

d) the Jabber Timer - disables the Transmitter in case of longer than legal length packets

3.1 RECEIVER FUNCTIONS

The Receiver includes an input buffer, a cable equalizer, a 4-pole Bessel low pass filter, a squelch circuit, and a differential line driver.

The buffer provides high input impedance and low input capacitance to minimize loading and reflections on the coax.

The equalizer is a high pass filter which compensates for the low pass effect of the cable. The composite result of the maximum length cable and the equalizer is a flatband response at the signal frequencies to minimize jitter.

The 4-pole Bessel low pass filter extracts the average DC level on the coax, which is used by both the Receiver squelch and the collision detection circuits.

The Receiver squelch circuit prevents noise on the coax from falsely triggering the Receiver in the absence of the signal. At the beginning of the packet, the Receiver turns on when the DC level from the low pass filter is lower than the DC squelch threshold. However, at the end of the packet, a quick Receiver turn off is needed to reject dribble bits. This is accomplished by an AC timing circuit that reacts to high level signals of greater than typically 200 ns in duration. The Receiver then stays off only if within about 1 μs, the DC level from the low pass filter rises above the DC squelch threshold. *Figure 2* illustrates the Receiver timing.

The differential line driver provides ECL compatible signals to the DTE with typically 3 ns rise and fall times. In its idle state, its outputs go to differential zero to prevent DC standing current in the isolation transformer.

3.2 TRANSMITTER FUNCTIONS

The Transmitter has a differential input and an open collector output current driver. The differential input common mode voltage is established by the CTI and should not be altered by external circuitry. The transformer coupling of TX$\pm$ will satisfy this condition. The driver meets all IEEE 802.3/Ethernet Specifications for signal levels. Controlled rise and fall times (25 ns V $\pm$5 ns) minimize the higher harmonic components. The rise and fall times are matched to minimize jitter. The drive current levels of the DP8392C meet the tighter recommended limits of IEEE 802.3 and are set by a built-in bandgap reference and an external 1% resistor. An on chip isolation diode is provided to reduce the Transmitter's coax load capacitance. For Ethernet compatible applications, an external isolation diode (see *Figure 4*) may be added to further reduce coax load capacitance. In Cheapernet compatible applications the external diode is not required as the coax capacitive loading specifications are relaxed.

The Transmitter squelch circuit rejects signals with pulse widths less than typically 20 ns (negative going), or with levels less than -175 mV. The Transmitter turns off at the end of the packet if the signal stays higher than -175 mV for more than approximately 300 ns. *Figure 3* illustrates the Transmitter timing.

3.0 Functional Description (Continued)

3.3 COLLISION FUNCTIONS

The collision circuitry consists of two buffers, two 4-pole Bessel low pass filters (section 3.1), a comparator, a heartbeat generator, a 10 MHz oscillator, and a differential line driver.

Two identical buffers and 4-pole Bessel low pass filters extract the DC level on the center conductor (data) and the shield (sense) of the coax. These levels are monitored by the comparator. If the data level is more negative than the sense level by at least the collision threshold (Vth), the collision output is enabled.

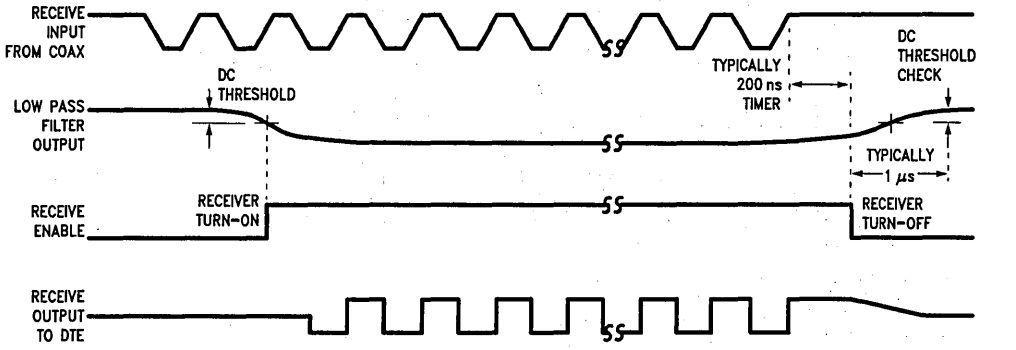
At the end of every transmission, the heartbeat generator creates a pseudo collision for a short time to ensure that the collision circuitry is properly functioning. This burst on collision output occurs typically 1.1 μs after the transmission, and has a duration of about 1 μs. This function can be disabled externally with the HBE (Heartbeat Enable) pin to allow operation with repeaters.

The 10 MHz oscillator generates the signal for the collision and heartbeat functions. It is also used as the timebase for all the jabber functions. It does not require any external components.

The collision differential line driver transfers the 10 MHz signal to the CD± pair in the event of collision, jabber, or heartbeat conditions. This line driver also features zero differential idle state.

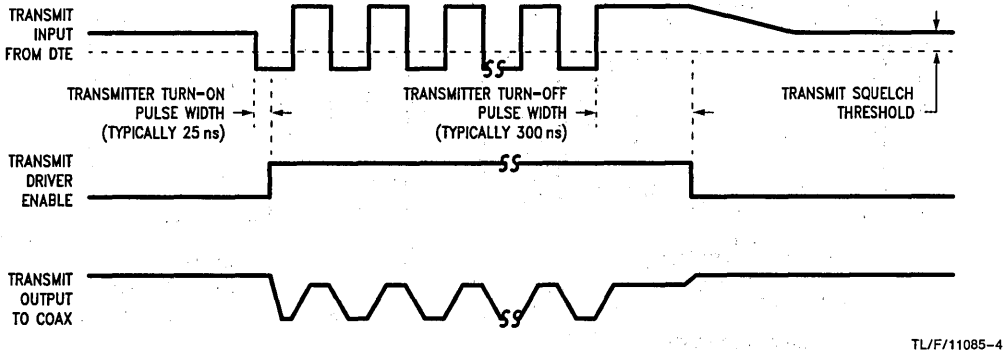
3.4 JABBER FUNCTIONS

The Jabber Timer monitors the Transmitter and inhibits transmission if the Transmitter is active for longer than 20 ms (fault). It also enables the collision output for the fault duration. After the fault is removed, The Jabber Timer waits for about 500 ms (unjab time) before re-enabling the Transmitter. The transmit input must stay inactive during the unjab time.



TL/F/11085–3

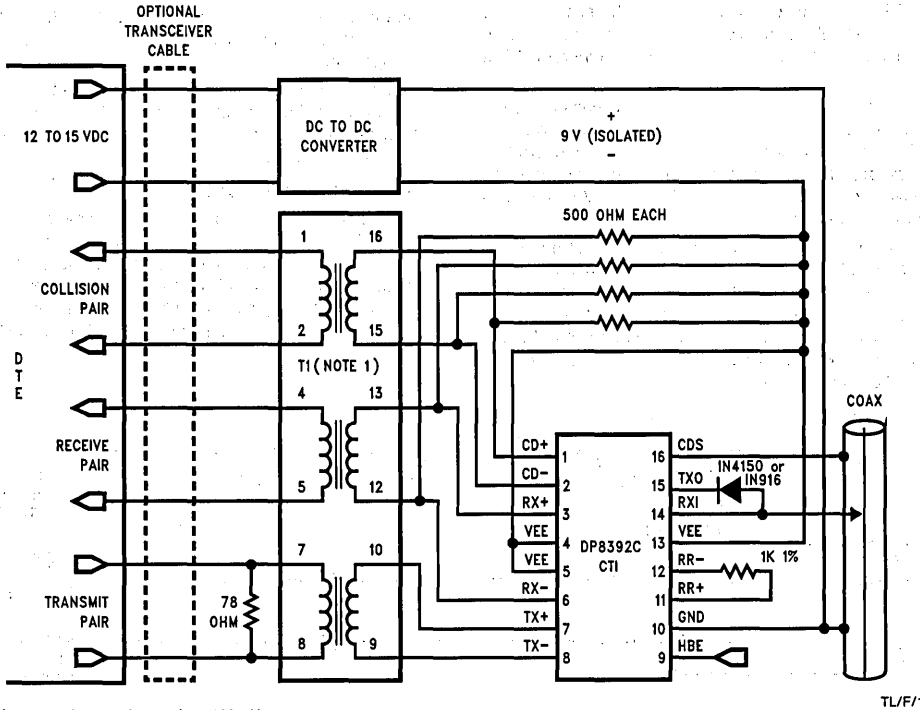
FIGURE 2. Receiver Timing



TL/F/11085–4

FIGURE 3. Transmitter Timing

4.0 Typical Application

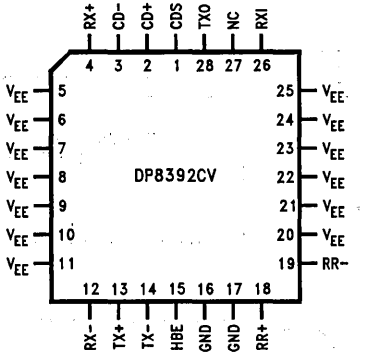


TL/F/11085–5

Note 1: T1 is a 1:1 pulse transformer, L = 100 μH
Pulse Engineering (San Diego) Part No. 64103
Valor Electronics (San Diego) Part No.
LT6003 or equivalent

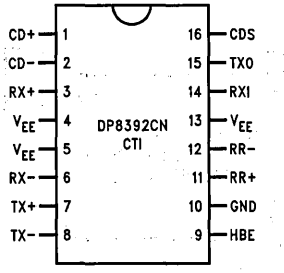
FIGURE 4

5.0 Connection Diagrams



TL/F/11085–6

Order Number DP8392CV
See NS Package Number V28A



TL/F/11085–16

Top View
Order Number DP8392CN
See NS Package Number N16A

FIGURE 5

6.0 Pin Descriptions

28-Pin PLCC	16-Pin DIP	Name	I/O	Description
2 3	1 2	CD+* CD−	O	**Collision Output.** Balanced differential line driver outputs from the collision detect circuitry. The 10 MHz signal from the internal oscillator is transferred to these outputs in the event of collision, excessive transmission (jabber), or during CD Heartbeat condition. These outputs are open emitters; pulldown resistors to VEE are required. When operating into a 78Ω transmission line, these resistors should be 500Ω. In Cheapernet applications, where the 78Ω drop cable is not used, higher resistor values (up to 1.5k) may be used to save power.
4 12	3 6	RX+* RX−	O	**Receive Output.** Balanced differential line driver outputs from the Receiver. These outputs also require 500Ω pulldown resistors.
13 14	7 8	TX+* TX−	I	**Transmit Input.** Balanced differential line receiver inputs to the Transmitter. The common mode voltage for these inputs is determined internally and must not be externally established. Signals meeting Transmitter squelch requirements are waveshaped and output at TXO.
15	9	HBE	I	**Heartbeat Enable.** This input enables CD Heartbeat when grounded, disables it when connected to VEE.
18 19	11 12	RR+ RR−	I	**External Resistor.** A fixed 1k 1% resistor connected between these pins establishes internal operating currents.
26	14	RXI	I	**Receive Input.** Connects directly to the coaxial cable. Signals meeting Receiver squelch requirements are equalized for inter-symbol distortion, amplified, and outputted at RX±.
28	15	TXO	O	**Transmit Output.** Connects either directly (Cheapernet) or via an isolation diode (Ethernet) to the coaxial cable.
1	16	CDS	I	**Collision Detect Sense.** Ground sense connection for the collision detect circuit. This pin should be connected separately to the shield to avoid ground drops from altering the receive mode collision threshold.
16, 17	10	GND		**Positive Supply Pin.** A 0.1 μF ceramic decoupling capacitor must be connected across GND and VEE as close to the device as possible.
5–11 20–25	4 5 13	VEE		**Negative Supply Pins.** In order to make full use of the 3.5W power dissipation capability of this package, these pins should be connected to a large metal frame area on the PC board. Doing this will reduce the operating die temperature of the device thereby increasing the long term reliability.

*IEEE names for CD± = CI±, RX± = DI±, TX± = DO±

6.1 P.C. BOARD LAYOUT

The DP8392C package is uniquely designed to ensure that the device meets the 1 million hour Mean Time Between Failure (MTBF) requirement of the IEEE 802.3 standard. In order to fully utilize this heat dissipation design, the three V_{EE} pins are to be connected to a copper plane which should be included in the printed circuit board layout.

There are two basic considerations in designing a PCB for the DP8392A, B, and C CTI. The first is ensuring that the layout does not degrade the electrical characteristics of the DP8392, and enables the end product to meet the IEEE 802.3 specifications. The second consideration is meeting the thermal requirements to the DP8392.

Since the DP8392 is highly integrated the layout is actually quite simple, and there are just a few guidelines:

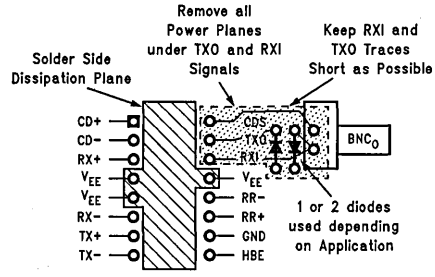
1. Ensure that the parasitic capacitance added to the RXI and TXO pins is minimized. To do this keep these signal traces short, and remove any power planes under these signals, and under any components that connect to these signals. *Figure 6* shows the component placement for the DIP package. The PLCC component placement would be similar, as shown in *Figure 7*.
2. The power supply layout to the CTI should be relatively clean. Usually the CTI's power is supplied directly by a DC-DC converter. The power should be routed either through separate isolated planes, or via thick PCB traces.

For the second consideration, the packaged DP8392 must have a thermal resistance of 40°C–45°C/W to meet the full 0°C–70°C temperature range. The CTI dissipates more power when transmitting than while it is idle. In order to do this the thermal resistance of the device must be 40°C–45°C/W. To meet this requirement during transmission, it is recommended that a small printed circuit board plane be connected to all V_{EE} pins on the solder side of the PCB.

The size of the trace plane depends on the package used and the duty cycle of transmissions. For the DIP package the plane should be connected to pins 4–5, 13, and the size should be approximately 0.2 square inches for applications where the duty cycle of the transmitter is very low (<10%). This would be typical of adapter or motherboard applications. In applications where the transmitter duty cycle may be large (repeaters and external transceivers) the total area should be increased to 0.4 in^2. *Figure 6* illustrates a recommended component side layout for these planes.

6.0 Pin Descriptions (Continued)

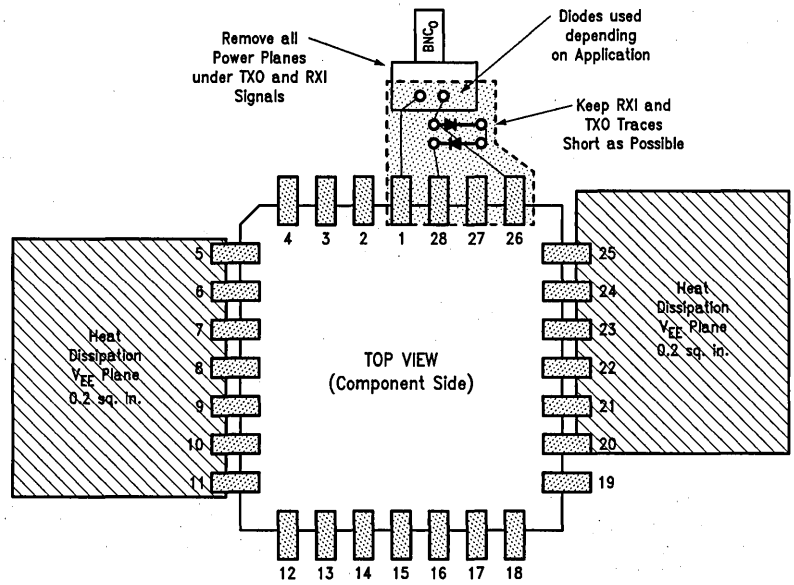
For the PLCC packaged DP8392, it is recommended that a small printed circuit board V_{EE} plane be connected to pins 5–11, and a second one be connected to pins 20–25. To reduce the thermal resistance to the required value, the area of the plane on EACH set of pins should be ≥ 0.20 in^2 for applications with low transmitter duty cycle, and ≥ 0.4 in^2 for high transmit duty cycle applications. *Figure 7* illustrates a recommended component side layout for these planes.



TL/F/11085–14

Layout as viewed from component side

FIGURE 6. Typical Layout Considerations for DP8392CN (Not to Scale)



TL/F/11085–15

FIGURE 7. Recommended Layout and Dissipation Planes for DP8392CV (Not to Scale)

7.0 Absolute Maximum Ratings (Note 1)

Supply Voltage (V_{EE})	−12V
Package Power Rating at 25°C (PC Board Mounted) Derate linearly at the rate of 28.6 mW/°C	3.5 Watts* See Section 5
Input Voltage	0 to −12V
Storage Temperature	−65° to 150°C
Lead Temp. (Soldering, 10 seconds)	260°C

*For actual power dissipation of the device please refer to section 7.0.

Recommended Operating Conditions

Supply Voltage (V_{EE})	−9v ±5%
Ambient Temperature	0° to 70°C

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

8.0 DP8392C Electrical Characteristics V_{EE} = −9V ±5%, T_A = 0° to 70°C (Notes 2 & 3)

All parameters with respect to CD± and RX± are measured after the pulse transformer except V_{OC}.

Symbol	Parameter	Min	Typ	Max	Units
I_{EE1}	Supply current out of V_{EE} pin—non transmitting		−85	−130	mA
I_{EE2}	Supply current out of V_{EE} pin—transmitting		−125	−180	mA
I_{RXI}	Receive input bias current (RXI)	−2		+25	μA
I_{TDC}	Transmit output dc current level (TXO)	37	41	45	mA
I_{TAC}	Transmit output ac current level (TXO)	±28		I_{TDC}	mA
V_{CD}	Collision threshold (Receive mode)	−1.45	−1.53	−1.58	V
V_{OD}	Differential output voltage (RX±, CD±)	±550		±1200	mV
V_{OC}	Common mode output voltage (RX±, CD±)	−1.5	−2.0	−2.5	V
V_{OB}	Diff. output voltage imbalance (RX±, CD±)			±40	mV
V_{TS}	Transmitter squelch threshold (TX±)	−175	−225	−300	mV
C_X	Input capacitance (RXI)		1.2		pF
R_{RXI}	Shunt resistance—non transmitting (RXI)	100			KΩ
R_{TXO}	Shunt resistance—transmitting (TXO)		10		KΩ

9.0 DP8392C-1 Electrical Characteristics V_{EE} = −9V ±5%, T_A = 0° to 70°C (Notes 2 & 3)

All parameters with respect to CD± and RX± are measured after the pulse transformer except V_{OC}.

Symbol	Parameter	Min	Typ	Max	Units
I_{EE1}	Supply current out of V_{EE} pin—non transmitting		−85	−130	mA
I_{EE2}	Supply current out of V_{EE} pin—transmitting		−125	−180	mA
I_{RXI}	Receive input bias current (RXI)	−2		+25	μA
I_{TDC}	Transmit output dc current level (TXO)	37	41	45	mA
I_{TAC}	Transmit output ac current level (TXO)	±28		I_{TDC}	mA
V_{CD}	Collision threshold (Receive mode)	−1.45	−1.53	−1.58	V
V_{OD}	Differential output voltage (RX±, CD±)	±550		±1200	mV
V_{OC}	Common mode output voltage (RX±, CD±)	−1.5	−2.0	−2.5	V
V_{OB}	Diff. output voltage imbalance (RX±, CD±)			±40	mV
V_{TS}	Transmitter squelch threshold (TX±)	−175	−225	−275	mV
C_X	Input capacitance (RXI)		1.2		pF
R_{RXI}	Shunt resistance—non transmitting (RXI)	100			KΩ
R_{TXO}	Shunt resistance—transmitting (TXO)	7.5K	10		KΩ

Note 1: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.

Note 2: All currents into device pins are positive, all currents out of device pins are negative. All voltages referenced to ground unless otherwise specified.

Note 3: All typicals are given for V_{EE} = −9V and T_A = 25°C.

10.0 DP8392C Switching Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Note 3)

Symbol	Parameter	Fig	Min	Typ	Max	Units
t_{RON}	Receiver startup delay (RXI to RX±)	8 & 14		4		bits
t_{Rd}	Receiver propagation delay (RXI to RX±)	8 & 14		15	50	ns
t_{Rr}	Differential outputs rise time (RX±, CD±)	8 & 14		4		ns
t_{Rf}	Differential outputs fall time (RX±, CD±)	8 & 14		4		ns
t_{RJ}	Receiver & cable total jitter	13		±2		ns
t_{TST}	Transmitter startup delay (TX± to TXO)	9 & 14		1		bits
t_{Td}	Transmitter propagation delay (TX± to TXO)	9 & 14		25	50	ns
t_{Tr}	Transmitter rise time —10% to 90% (TXO)	9 & 14		25		ns
t_{Tf}	Transmitter fall time —90% to 10% (TXO)	9 & 14		25		ns
t_{TM}	t_{Tr} and t_{Tf} mismatch			0.5		ns
t_{TS}	Transmitter skew (TXO)			±0.5		ns
t_{TON}	Transmit turn-on pulse width at V_{TS} (TX±)	9 & 14		20		ns
t_{TOFF}	Transmit turn-off pulse width at V_{TS} (TX±)	9 & 14		250		ns
t_{CON}	Collision turn-on delay	10 & 14		7		bits
t_{COFF}	Collision turn-off delay	10 & 14			20	bits
f_{CD}	Collision frequency (CD±)	10 & 14	8.0		12.5	MHz
t_{CP}	Collision pulse width (CD±)	10 & 14	35		70	ns
t_{HON}	CD Heartbeat delay (TX± to CD±)	11 & 14	0.6		1.6	μs
t_{HW}	CD Heartbeat duration (CD±)	11 & 14	0.5	1.0	1.5	μs
t_{JA}	Jabber activation delay (TX± to TXO and CD±)	12 & 14	20	29	60	ms
t_{JR}	Jabber reset unjab time (TX± to TXO and CD±)	12 & 14	250	500	750	ms

DP8392C-1 Switching Characteristics $V_{EE} = -9V \pm 5\%$, $T_A = 0°$ to 70°C (Note 3)

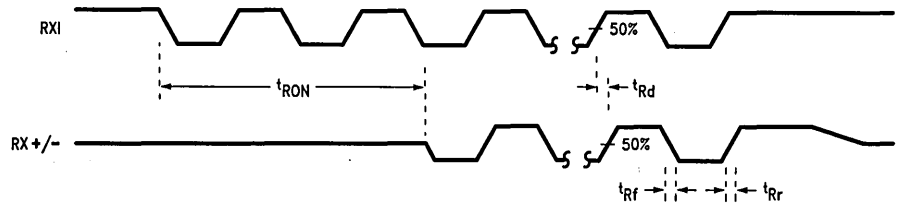
Symbol	Parameter	Fig	Min	Typ	Max	Units
t_{RON}	Receiver startup delay (RXI to RX±)	8 & 14		4	5	bits
t_{Rd}	Receiver propagation delay (RXI to RX±)	8 & 14		15	50	ns
t_{Rr}	Differential outputs rise time (RX±, CD±)	8 & 14		4	7	ns
t_{Rf}	Differential outputs fall time (RX±, CD±)	8 & 14		4	7	ns
t_{RJ}	Receiver & cable total jitter	13		±2		ns
t_{TST}	Transmitter startup delay (TX± to TXO)	9 & 14		1	2	bits
t_{Td}	Transmitter propagation delay (TX± to TXO)	9 & 14	5	25	50	ns
t_{Tr}	Transmitter rise time —10% to 90% (TXO)	9 & 14	20	25	30	ns
t_{Tf}	Transmitter fall time —90% to 10% (TXO)	9 & 14	20	25	30	ns
t_{TM}	t_{Tr} and t_{Tf} mismatch			0.5		ns
t_{TS}	Transmitter skew (TXO)			±0.5		ns
t_{TON}	Transmit turn-on pulse width at V_{TS} (TX±)	9 & 14	5	20	40	ns
t_{TOFF}	Transmit turn-off pulse width at V_{TS} (TX±)	9 & 14	110		270	ns
t_{CON}	Collision turn-on delay	10 & 14		7	13	bits
t_{COFF}	Collision turn-off delay	10 & 14			20	bits
f_{CD}	Collision frequency (CD±)	10 & 14	8.5		12.5	MHz
t_{CP}	Collision pulse width (CD±)	10 & 14	35		70	ns
t_{HON}	CD Heartbeat delay (TX± to CD±)	11 & 14	0.6		1.6	μs
t_{HW}	CD Heartbeat duration (CD±)	11 & 14	0.5	1.0	1.5	μs
t_{JA}	Jabber activation delay (TX± to TXO and CD±)	12 & 14	20	29	60	ms
t_{JR}	Jabber reset unjab time (TX± to TXO and CD±)	12 & 14	250	500	750	ms

Note 1: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.

Note 2: All currents into device pins are positive, all currents out of device pins are negative. All voltages referenced to ground unless otherwise specified.

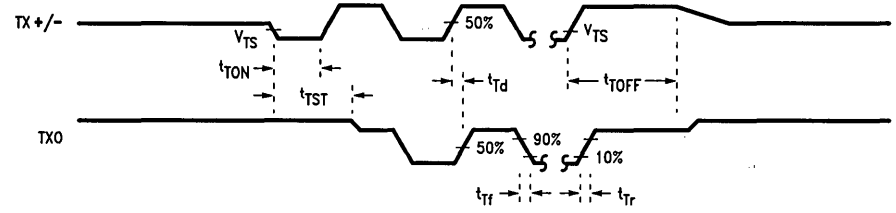
Note 3: All typicals are given for $V_{EE} = -9V$ and $T_A = 25°C$.

11.0 Timing and Load Diagrams



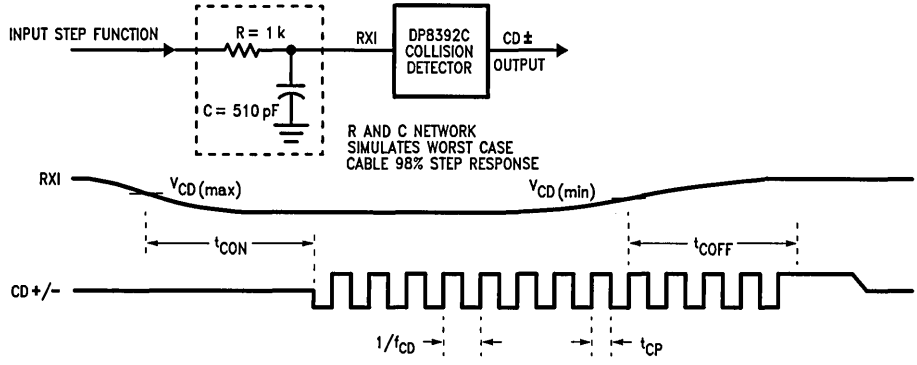
TL/F/11085–7

FIGURE 8. Receiver Timing



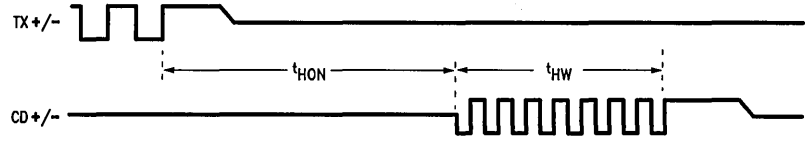
TL/F/11085–8

FIGURE 9. Transmitter Timing



TL/F/11085–9

FIGURE 10. Collision Timing

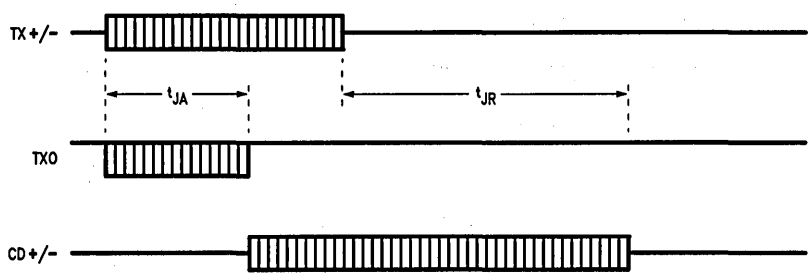


TL/F/11085–10

FIGURE 11. Heartbeat Timing

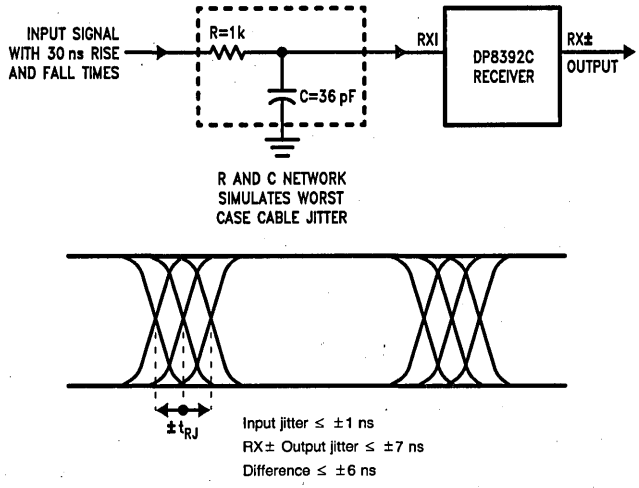
2

11.0 Timing and Load Diagrams (Continued)



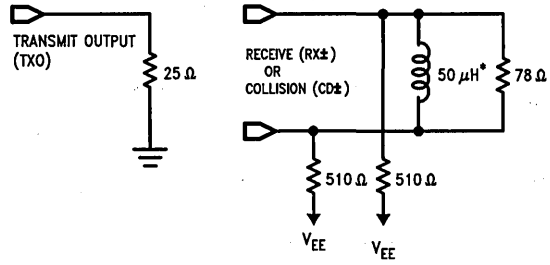
TL/F/11085-11

FIGURE 12. Jabber Timing



TL/F/11085-12

FIGURE 13. Receive Jitter Timing



TL/F/11085-13

*The 50 μH inductance is for testing purposes. Pulse transformers with higher inductances are recommended (see *Figure 4*)

FIGURE 14. Test Loads

Reliability Data Summary for DP8392

REF: TEST LAB FILES

RDT25406 RDT26627
RDT25500 RDT26638
RDT26562

ABSTRACT

DP8392 Coaxial Transceiver Interface parts from 8 lots were subjected to Operating Life Test, Temperature and Humidity Bias Test, Temperature Cycle Test, and Electrostatic Discharge Test.

PURPOSE OF TEST

Evaluation of new device and qualification of U.K. fab.

TESTS PERFORMED

Operating Life Test (OPL) (100°C; biased)

Operating Life Test (OPL) (125°C; biased)

Temperature and Humidity Bias Test (THBT) (85°C; 85% R.H.; biased)

Temperature Cycle Test (TMCL) (−40°C, +125°C; unbiased)

Electrostatic Discharge Test (ESD) (Human body model: R = 1500Ω; C = 120 pF)

CONCLUSIONS

1. The DP8392AN exceeds the IEEE 802.3 specification of 1 million hours Mean Time Between Failure (MTBF).
2. U.K. fab results are comparable to those of Santa Clara. On ESD testing all pins passed at 1000V except for pin 7 (TX^+).

TEST SAMPLE DESCRIPTION/HISTORY

Lot	Device	Package	Date Code	Fab Location	Assembly Location
1	DP8392	N, 16 Leads	8509	NSSC	NSEB
2	DP8392	N, 16 Leads	8513	NSSC	NSEB
3	DP8392	N, 16 Leads	8526	NSSC	NSEB
4	DP8392	N, 16 Leads	8552	NSSC	NSEB
5	DP8392A(-4)	N, 16 Leads	8620	NSUK	NSEB
6	DP8392A(-5)	N, 16 Leads	8637	NSUK	NSEB
7	DP8392A(-5)	N, 16 Leads	8637	NSUK	NSEB
8	DP8392A(-5)	N, 16 Leads	8637	NSUK	NSEB
9	DP8392C	N, 16 Leads	9106	NSUK	NSEB
10	DP8392C	N, 16 Leads	9106	NSUK	NSEB
11	DP8392C	N, 16 Leads	9106	NSUK	NSEB
12	DP8392C	N, 16 Leads	9106	NSUK	NSEB
13	DP8392C	N, 16 Leads	9106	NSUK	NSEB
14	DP8392C	N, 16 Leads	9106	NSUK	NSEB

RESULTS

Test	Temperature	Lot	Fab	Time Point—Number of Failures				
				Hours				
				168	336	500	1000	2000
OPL	100°C	1	NSSC	0/50		0/50	0/50	
	100°C	2	NSSC	0/50		0/50	0/50	
	125°C	3	NSSC	0/74				
	125°C	4	NSSC	0/100		0/100	0/100	0/100
	100°C	5	NSUK	0/60				
	100°C	6	NSUK		0/33	0/33	0/33	0/33
	100°C	7	NSUK		0/31	0/31	0/31	0/31
	100°C	8	NSUK		0/33	0/31	0/31	0/31
	85°C	9	NSUK			0/77	0/77	
	85°C	10	NSUK			0/77	0/77	
	85°C	11	NSUK			0/77	0/77	
	100°C	12	NSUK	0/64		0/64	0/64	0/64
	100°C	13	NSUK	0/25		0/25	0/25	0/25
	100°C	14	NSUK	0/10		0/10	0/10	0/10
THBT	85°C; 85% R.H.	1	NSSC	0/50		0/50	0/50	
		2	NSSC	0/50		0/50	0/50	
		3	NSSC	0/75		0/75	0/75	
		9	NSUK	0/30		0/30	0/30	
		10	NSUK	0/30		0/30	0/30	
		11	NSUK	0/30		0/30	0/30	

RESULTS (Continued)

Test	Temperature	Lot	Fab	Time Point—Number of Failures				
				Hours				
				168	336	500	1000	2000
ACLV	121°C; 100% R.H.	9	NSUK	0/77		0/77		
		10	NSUK	0/66		0/66		
		11	NSUK	0/77		0/77		
				Cycles				
				500	1000	2000	3000	
TMCL	−40°C, +125°C	4	NSSC	0/70	0/70	0/70	0/70	
	−65°C, +150°C	9	NSUK	0/77	0/77	0/77		
	−65°C, +150°C	10	NSUK	0/66	0/66	0/66		
	−65°C, +150°C	11	NSUK	0/77	0/77	0/77		

ELECTROSTATIC DISCHARGE TEST (ESD) RESULTS

26 parts from 4 wafer lots were tested by the Human Body Model test condition; R = 1500Ω; C = 120 pF. First ground was held common, then V_{EE}. 5 positive and 5 negative pulses were applied for each pin/voltage combination.

Pin	Function	Voltage—Number of Failures	
		500V	1000V
1	CD+	0/26	0/20
2	CD−	0/26	0/20
3	RX+	0/26	0/20
4	V_{EE}	0/26	0/20
5	V_{EE}	0/26	0/20
6	RX−	0/26	0/20
7	TX+	6/26	13/20
8	TX−	0/26	0/20
9	HBE	0/26	0/20
10	GND	0/26	0/20
11	RR+	0/26	0/20
12	V_{EE}	0/26	0/20
13	V_{EE}	0/26	0/20
14	RXI	0/26	0/20
15	TXO	0/26	0/20
16	CDS	0/26	0/20

Further characterization has been done to determine individual pin ESD damage thresholds. In particular, for pin 7 (TX+), 80 parts from 4 wafer lots were tested. Pin 7 ESD damage thresholds varied from 200V–300V to 2000V–3000V, with a mean of 1800V.

MTBF (MEAN TIME BEFORE FAILURE) CONSIDERATIONS

Results total: 212, 000 device hours at 125°C, 0 failures
301,000 device hours at 100°C, 0 failures

Assume: E_a = 0.7 eV
P_d = 800 mW
θ_{ja} = 45°C/W
Chi-square statistics, 60% confidence

Then: $MTBF_{min}$ at 25°C ambient = 93,000,000 device hours.
$MTBF_{min}$ at 70°C ambient = 5,100,000 device hours.

Ethernet/Cheapernet Physical Layer Made Easy with DP8391/92

National Semiconductor
Application Note 442
Alex Djenguerian

With the integration of the node electronics of IEEE 802.3 compatible local area networks now on silicon, system design is simplified. This application note describes the differences between the Ethernet and Cheapernet versions of the standard, and provides design guidelines for implementing the node electronics with National Semiconductor's DP8390 LAN chip set.

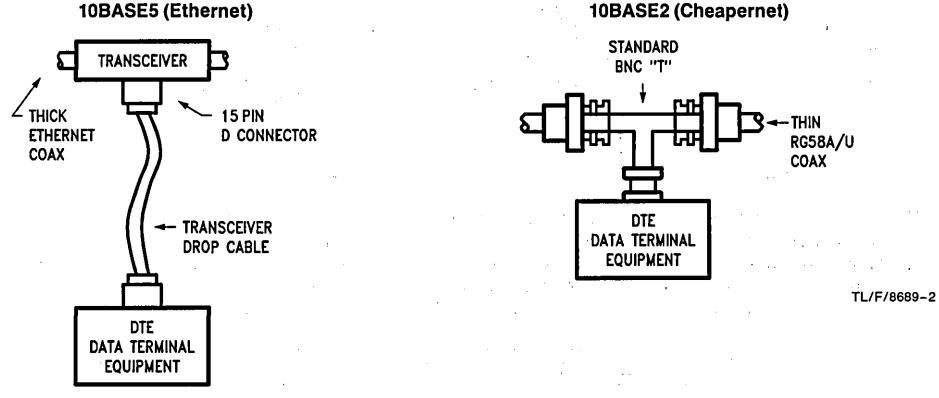
INTRODUCTION

The DP8390 chip set is designed to provide the physical and media access control layer functions of local area networks as specified in IEEE 802.3 standard. This standard is based on the access method known as carrier-sense multiple access with collision detection (CSMA/CD). In this scheme, if a network station wants to transmit, it first "listens" to the medium; if someone else is transmitting, the station defers until the medium is clear before it begins to transmit. However, two or more stations could still begin transmitting at the same time and give rise to a collision. When this happens, the two nodes detect this condition, back off for a random amount of time before making another attempt.

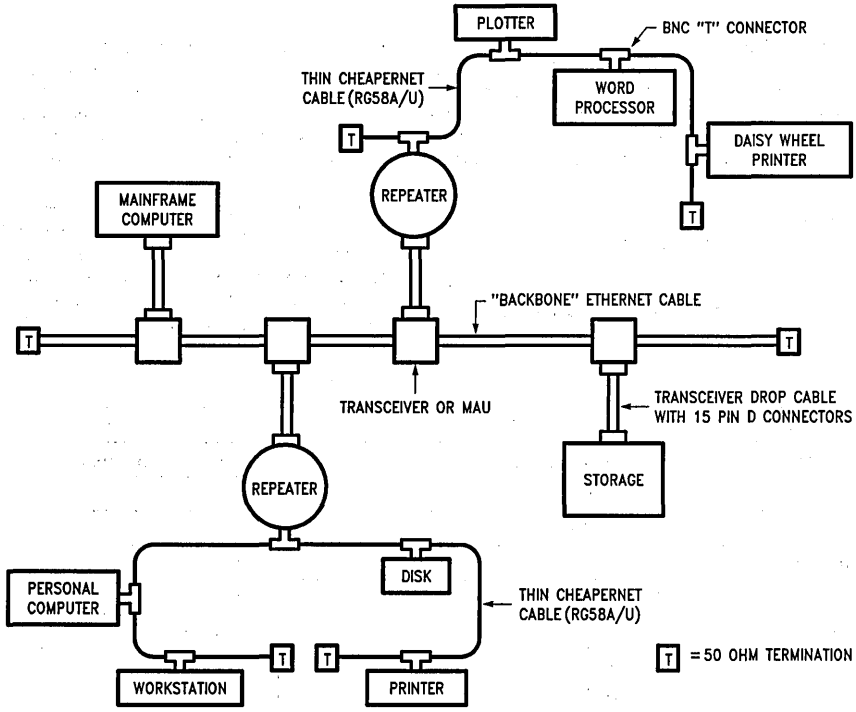
The IEEE 802.3 standard supports two different versions for the media, 10BASE5 (commonly known as Ethernet) and 10BASE2 (Cheapernet). These can be used separately, or together in a hybrid form. Both versions have similar electrical specifications and can be implemented using the same transceiver chip (DP8392). Cheapernet is the low cost version and is user installable. The following table compares the two:

Parameter	10BASE5 (Ethernet)	10BASE2 (Cheapernet)
Data Rate	10 Mbit/s baseband	10 Mbits/s baseband
Segment Length	500 m	185 m
Network Span	2500 m	925 m
Nodes per Segment	100	30
Node Spacing	2.5 m (cable marked)	0.5 m min
Capacitance per Node	4 pF max	8 pF max
Cable	0.4 in diameter 50Ω Double Shielded Rugged N-Series Connectors	0.2 in diameter 50Ω (RG58A/U) Single Shielded Flexible BNC Connectors
Tranceiver Drop Cable	0.39 in diameter multiway cable with 15 pin D connectors 50 m max length	Not needed due to the high flexibility of the RG58A/U cable

Typical Connection Diagram for a Station



Although Cheapernet is intended for local use, several 185 meter segments can be joined together with simple repeaters to provide for a larger network span. Similarly, several Cheapernet segments can be tied into a longer Ethernet "backbone". In this hybrid configuration, the network combines all the benefits of Cheapernet, flexibility and low cost, with the ruggedness and the much larger geographic range of standard Ethernet. *Figure 1* illustrates a typical hybrid LAN configuration.

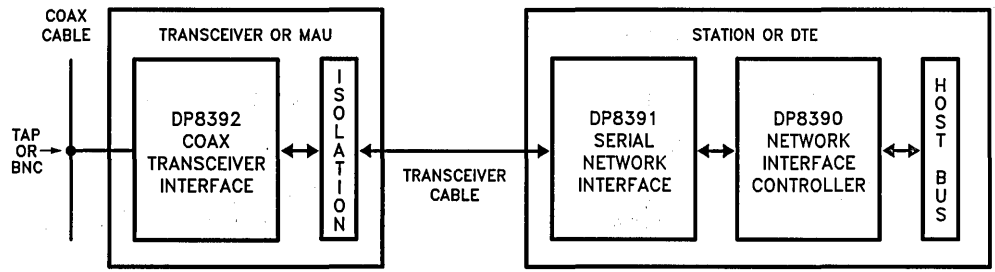


TL/F/8689-3

FIGURE 1. A Hybrid Ethernet/Cheapernet System

TRANSMITTING AND RECEIVING PACKETS WITH THE DP8390 CHIPSET

Node Block Diagram



TL/F/8689-4

The node electronics is integrated into three chips, the DP8390 Network Interface Controller (NIC), the DP8391 Serial Network Interface (SNI), and the DP8392 Coaxial Transceiver Interface (CTI). To transmit a packet, the host processor issues a transmit command to the NIC, which normally transfers the data to a local buffer memory. The NIC then automatically handles the transmission of the packet (from the local buffer through an on-board FIFO to the SNI) according to the CSMA/CD protocol. The packet has to be in the following format:

PREAMBLE	SFD	DESTINATION	SOURCE	LENGTH	DATA	CRC
62-bits	2-bits	6-bytes	6-bytes	2-bytes	46–1500 bytes	4-bytes

PREAMBLE: This section consists of alternating 1 and 0 bits. As the packet travels through the network, some of these bits would be lost as most of the network components are allowed to provide an output some number of bits after being presented with a valid input.

START OF A FRAME DELIMITER (SFD): This field consists of two consecutive 1's to signal that the frame reception should begin.

DESTINATION AND SOURCE ADDRESSES: Each one of these frames is 6 bytes long and specifies the address of the corresponding node.

LENGTH: This 2 byte field indicates the number of bytes in the data field.

DATA: This field can be from 46 to 1500 bytes long. Messages shorter than 46 bytes require padding to bring the data field to the minimum length. If the data field is padded, the host can determine the number of valid data bytes by looking at the length field. Messages longer than 1500 bytes must be broken into multiple packets.

CRC: This field contains a Cyclic Redundancy Code calculation performed on the Destination address through the Data field for error control.

The shortest packet length thus adds up to be 512 bits long (excluding the preamble and the SFD). At 10 Mbit/sec this amounts to 51.2 μs, which is twice as much as the 25 μs maximum end-to-delay time that is allowed by the IEEE 802.3 protocol. This ensures that if a collision arises in the network, it would be recognized at all node locations.

The SNI combines the NRZ data packet received from the controller with a clock signal and encodes them into a serial bit stream using standard Manchester encoding. In this coding scheme, the first half of the bit cell contains the complementary data and the second half contains the true data. Thus a transition is always guaranteed in the middle of a bit cell.

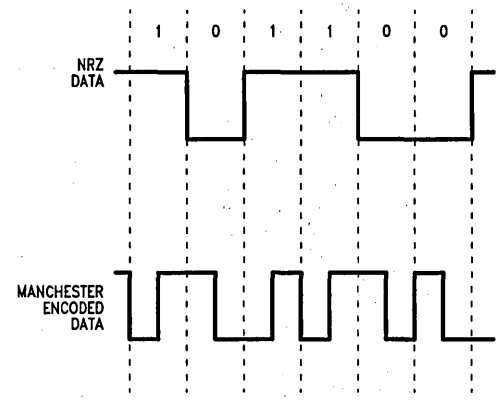
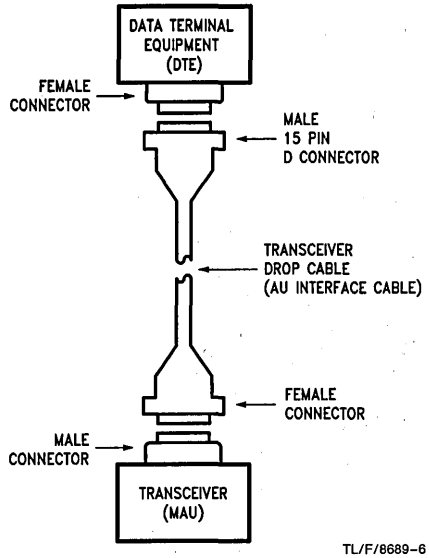


FIGURE 2. Manchester Coding

The encoded signal appears in differential form at the SNI's output. In 10BASE5 (Ethernet) applications, this signal is sent to the transceiver or the Medium Attachment Unit (MAU) through the twisted pair Tranceiver Drop cable (also known as the Attachment Unit Interface cable). This cable typically consists of four individually shielded twisted wire pairs with an overall shield covering these individually shielded pairs. The signal pairs, which have a differential characteristic impedance of 78Ω $\pm$5Ω, should be terminated at the receiving ends. The cable can be up to 50 meters in length and have a maximum delay of 257 ns. The shields of the individual pairs should be connected to the logic ground in the Data Terminal Equipment (DTE) and the outer shield to the chassis ground. *Figure 3* shows a picture of the cable and the corresponding pin assignments.



Pin	IEEE 802.3 Name	Pairs	DP8391/2 Name	Signal from	
				DTE	MAU
3	DO + (Data Out +)	Transmit	TX+	X	
10	DO − (Data Out −)	Pair	TX−	X	
11	DO S (DO Shield)			X	
5	DI + (Data In +)	Receive	RX+		X
12	DI − (Data In −)	Pair	RX−		X
4	DI S (DI Shield)			X	
7	CO + (Control Out +)	Optional		X	
15	CO − (Control Out −)	Pair		X	
8	CO S (CO Shield)			X	
2	CI + (Control In +)	Collision	CD+		X
9	CI − (Control In −)	Pair	CD−		X
1	CI S (CI Shield)			X	
6	VC (Voltage Common)	Power		X	
13	VP (Voltage Plus)	Pair		X	
14	VS (Voltage Shield)			X	
Shell	PG (Protective GND)			X	

FIGURE 3. Transceiver Cable Pin Assignments

The transmitted packet from the SNI as well as all other signals (receive, collision, and DC power) must be electrically isolated from the coax in the MAU. The isolation means provided must withstand 500 V_{AC} rms for one minute for 10BASE2 and 2000 V_{AC} rms for 10BASE5. In order to detect collisions reliably, the electrical isolation is not done at the coax; instead it is done on the side of the Attachment Unit Interface. The isolation for the three signal lines can be easily provided by using three pulse transformers that come in a standard 16 pin plastic DIP from several manufacturers (Pulse Engineering, Valor Electronics). The inductance value for these transformers vary from 50 μH to 150 μH with the larger inductance values slowing the rise and fall times, and the smaller ones causing more voltage droop.

The Manchester encoded data from the SNI now reaches the CTI's transmit input after passing through the isolation transformer. A noise filter at this input provides a static noise margin of −175 mV to −300 mV. These thresholds assure that differential Transmit (TX±) data signals less than −175 mV or narrower than 10 ns are always rejected, while signals greater than −300 mV and wider than 30 ns are always accepted. The −300 mV threshold provides sufficient margin since the differential drivers for the transceiver drop cable provide a minimum signal level of ±450 mV after inductive droop, and the maximum attenuation allowed for the drop cable is 3 dB at signal frequencies. Signals meeting the squelch requirements are waveshaped and outputted to the coax medium. This is done as follows:

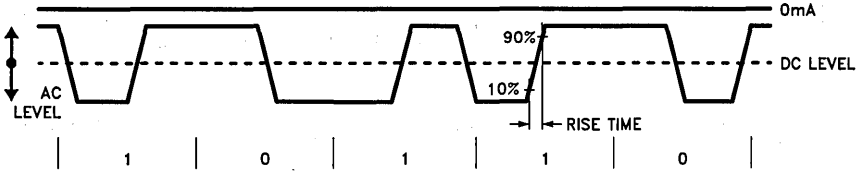
The transmitter's output driver is a switching current source that drives a purely resistive load of 25Ω presented by the coax to produce a voltage swing of approximately 2V. This signal has to meet several critical electrical requirements:

RISE/FALL TIMES: The 10%–90% rise and fall times have to be 25 ns ±5 ns at 10 Mbit/sec. This spec helps to minimize electro-magnetic radiation by reducing the higher harmonic content of the signal and contributes to the smaller reflection levels on the coax. In addition, the rise and fall times are required to be matched to within 1 ns to minimize the overall jitter in the system.

DC LEVEL: The DC component of the signal has to be between −37 mA and −45 mA. The tolerance here is tight since collisions are detected by monitoring the average DC level on the coax.

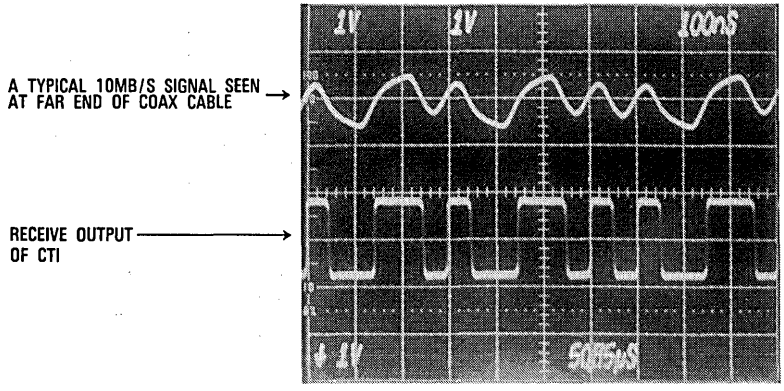
AC LEVEL: The AC component of the signal has to be between ±28 mA and the DC level. This specification guarantees a minimum signal at the far end of the coax cable in the worst case condition.

The signal shown in Fig. 4 would be attenuated as it travels along the coax. The maximum cable attenuation per segment is 8.5 dB at 10 MHz and 6 dB at 5 MHz. This applies for both the 500 meters of Ethernet cable and the 185 meters of Cheapernet cable. With 10 Mbit/sec Manchester data, this cable attenuation results in approximately 7 ns of edge jitter in either direction. The CTI's receiver has to compensate for at least a portion of this jitter to meet the ±6 ns combined jitter budget. The receiver also should not overcompensate the signal in the case of a short cable. An equalizer filter in the CTI accomplishes this task. *Figure 5* shows a typical waveform seen at the far end of the cable and the corresponding differential output from the CTI's receiver.



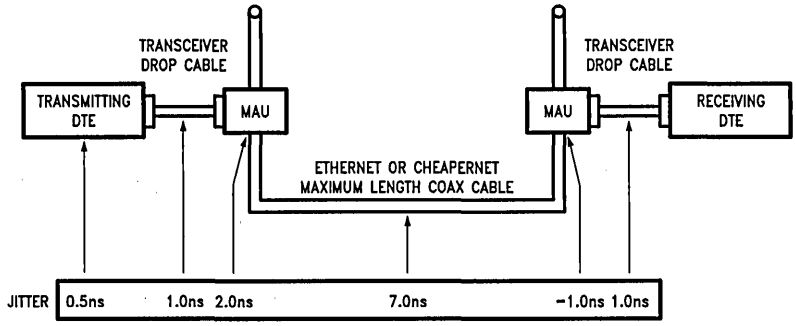
TL/F/8689–7

FIGURE 4. Coax Transmit Waveform



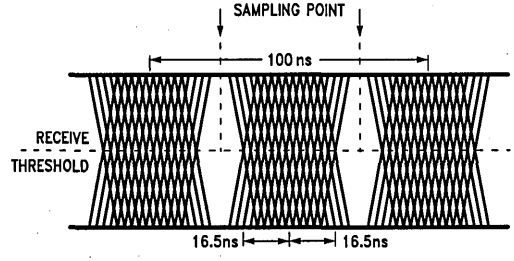
TL/F/8689–8

FIGURE 5. Oscilloscope Waveforms



TL/F/8689–9

Total Jitter without Noise = 0.5 + 1.0 + 2.0 + 7.0 − 1.0 + 1.0 = 10.5 ns
Additional Jitter from Noise on Coax Cable = 5.0 ns
Additional Jitter from Noise on Drop Cables = 1.0 ns
Total System Jitter = 16.5 ns



TL/F/8689–10

FIGURE 6. Typical Signal Waveform at SNI's Input

In addition to the equalizer, an AC/DC squelch circuit at the coax input prevents noise on the cable from falsely triggering the receiver in the absence of a valid signal. The Receive differential line from the CTI should be isolated before it reaches the SNI for Manchester decoding. This signal now could have accumulated as much as ±16.5 ns of jitter. *Figure 6* illustrates the jitter allocations for different network components and a typical signal waveform at the SNI's input. The digital phase-locked loop of the SNI can decode Manchester data with up to ±20 ns of random jitter which provides enough margin for implementation.

The SNI converts the Manchester received packet to NRZ data and clock pulses and sends them to the controller. Upon reception, the NIC checks the destination address, and if it is valid, verifies the CRC with the one generated on board and stores the packet in the local buffer memory. The packet is then moved to the host by the NIC, and when this is completed the buffer area is reclaimed for storing new packets. If a collision occurs during this transfer process, the CTI will detect it by sensing the average DC level on the coax and will send a 10 MHz collision signal to the SNI. The SNI will translate this information to the controller in TTL form, and the transmitting controllers will backoff for different times and retransmit later. Also in case of illegally long packets (longer than 20 ms), a jabber timer in the CTI will disable the coax driver so that the "jabbering" station will not bring down the entire network. The collision pair is activated in this case to inform the controller of the faulty condition. After the fault is removed, the jabber timer holds for 500 ms before re-enabling the coax driver.

COLLISION DETECTION SCHEMES

There are two different collision detection schemes that can be implemented with the CTI; receive, transmit modes. The IEEE 802.3 standard allows the use of receive, transmit, and transhybrid modes for non-repeater nodes for both Ethernet and Cheapernet applications. Repeaters are required to have the receive mode implementation.

RECEIVE MODE: Detects a collision between any two stations on the network with certainty at all times.

TRANSMIT MODE: Detects collisions with certainty only when the station is transmitting.

RECEIVE MODE: The receive mode scheme has a very simple truth table; however, the tight threshold limits make the design of it difficult. The threshold in this case has to be between the maximum DC level of one station (−1300 mV) and the minimum DC level of two far end stations (−1581 mV). Several factors such as the termination resistor variation, coax center conductor resistance, driver current level variation, signal skew, and input bias current of non-transmitting nodes contribute to this tight margin. On

Truth Table for Various Collision Detection Schemes

Mode	Receive				Transmit			
No. of Stations	0	1	2	>2	0	1	2	>2
Transmitting	N	N	Y	Y	N	N	Y	Y
Non-Transmitting	N	N	Y	Y	N	N	M	Y

Y = It will detect a collision, N = It will not detect a collision,
M = It may detect a collision

top of the −1300 mV minimum level, the impulse response of the internal low pass filter has to be added. The CTI incorporates a 4 pole Bessel filter in combination with a trimmed on board bandgap reference to provide this mode of collision detection. However it would be difficult in receive mode to extend the cable length beyond the limits of the standard. It is also argued that it is not necessary for non-repeater nodes to detect collisions between other stations.

TRANSMIT MODE: In this case collisions have to be detected with certainty only when the station is transmitting. Thus, collisions caused by two other nodes may or may not be detected. This feature relaxes the upper limit of the threshold from −1581 mV to −1782 mV. As a result of this, longer cable segments can be used. With the CTI, a resistor divider can be used at the Collision Detect Sense pin (CDS) to lower the threshold from receive to transmit mode. Typical resistor values can be 120Ω from CDS to GND and 10k from CDS to V_{EE} (This moves the threshold by about −100 mV).

IMPLEMENTING A 10 BASE5 (ETHERNET) MAU WITH THE DP8392

The CTI provides all the MAU (transceiver) functions except for signal and power isolation. Signal isolation can easily be provided by a set of three pulse transformers that come in a single Dual-In-Line package. These are available from transformer vendors such as Pulse Engineering (PE64103) and Valor (LT1101). However, for the power isolation a DC to DC converter is required. The CTI requires a single −9 (±5%) volt supply. This power has to be derived from the power pair of the drop cable which is capable of providing 500 mA in the 12 (−6%) to 15 (+5%) volt range. The low supply current of the CTI makes the design of the DC to DC converter quite easy. Such converters are being developed in hybrid packages by transformer manufacturers (Pulse Engineering PE64430 and Reliability Inc. 2E12R9). They provide the necessary voltage isolation and the output regulation. One can also build a simple DC to DC converter with a two transistor self oscillating primary circuit and some regulation on the secondary as shown in *Figure 7*.

Several areas of the PC board layout require special care. The most critical of these is for the coax connection. Ethernet requires that the CTI capacitance be less than 2 pF on the coax with another 2 pF allocated for the tap mechanism. The Receive Input (RXI) and the Transmit Output (TXO) lines should be kept to an absolute minimum by mounting the CTI very close to the center pin of the tap. Also, for the external diode at TXO (see *Figure 8*), the designer must minimize any stray capacitance, particularly on the anode side of the diode. To do this, all metal lines, especially the ground and V_{EE} planes, should be kept as far as possible from the RXI and TXO lines.

In order to meet the stringent capacitive loading requirements on the coax, it is imperative that the CTI be directly soldered to the PC board without a socket. A special lead frame in the CTI package allows direct conduction of heat from the die through these leads to the PC board, thus reducing the operating die temperature significantly. For good heat conduction the V_{EE} pins (4, 5 and 13) should be connected to large metal traces or planes.

A separate voltage sense pin (CDS) is provided for accurate detection of collision levels on the coax. In receive mode, where the threshold margin is tight, this pin should be independently attached to the coax shield to minimize errors due to ground drops. A resistor divider network at this pin can be used for transmit mode operation as described earlier.

The differential transmit pair from the DTE should be terminated with a 78Ω differential resistive load. By splitting the termination resistor into two equal values and capacitively AC grounding the center node, the common mode impedance is reduced to about 20Ω, which helps to attenuate common mode transients.

To drive the 78Ω differential line with sufficient voltage swings, the CTI's collision and receive drivers need external 500Ω resistors to V_{EE}. By using external resistors, the power dissipation of the chip is reduced, enhancing long term reliability. The only precision component required for the CTI is one 1k 1% resistor. This resistor sets many important parameters of the chip such as the coax driving levels, output rise and fall times, 10 MHz collision oscillator frequency, jabber timing, and receiver AC squelch timing. It should be connected between pins 11 (RR+) and 12 (RR−).

The DP8392 features a heartbeat function which can be externally disabled using pin 9. This function activates the collision output for a short time (10 ±5 bit cells) at the end of every transmission. It is used to ensure the controller that the collision circuitry is intact and properly functioning. Pin 9 enables CD Heartbeat when grounded, and disables it when connected to V_{EE}.

The IEEE 802.3 standard requires a static discharge path to be provided between the shield of the coax cable and the DTE ground via a 1 MΩ, 0.25W resistor. The standard also requires the MAU to have low susceptibility levels to electromagnetic interference. A 0.01 μF capacitor will provide a sufficient AC discharge path from the coaxial cable shield to the DTE ground. The individual shields should also be capacitively coupled to the Voltage Common in MAU. A typical Ethernet MAU connection diagram using the CTI in receive mode with the CD Heartbeat enabled is shown in Figure 8.

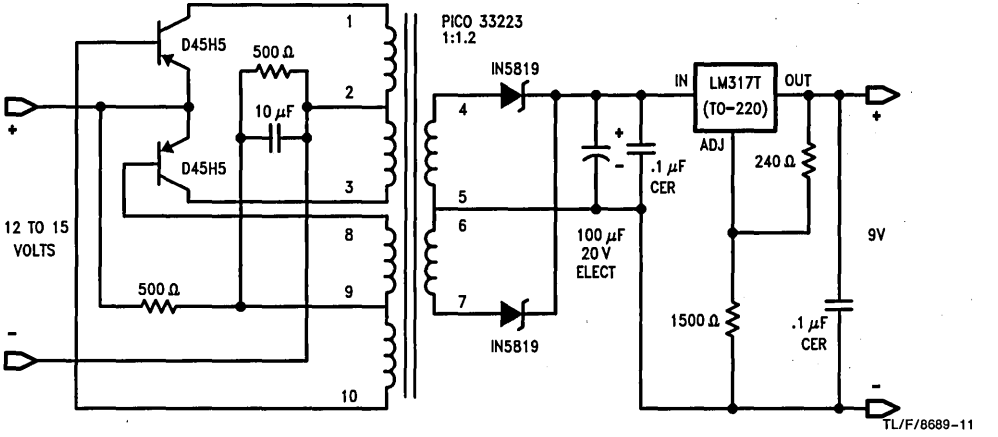


FIGURE 7. A Simple Low Cost DC to DC Converter

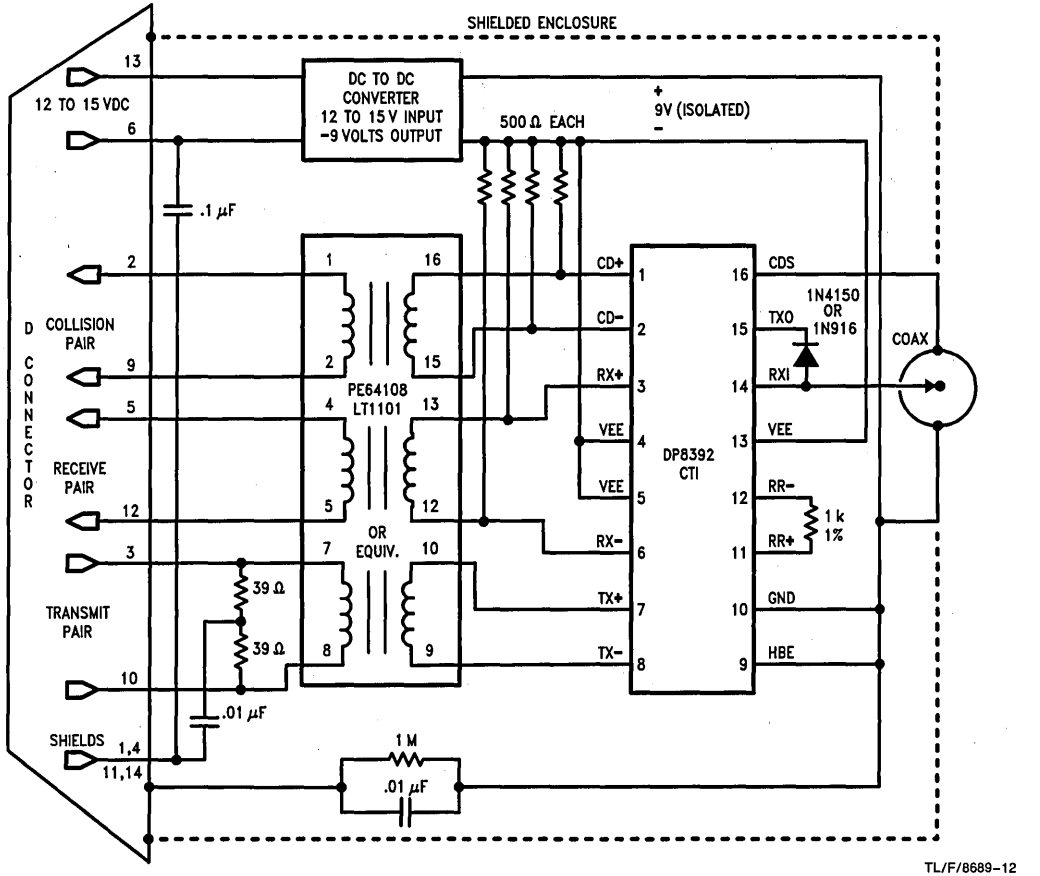


FIGURE 8. An Ethernet MAU Implementation with the CTI

CHEAPERNET APPLICATION WITH THE DP8391 AND DP8392

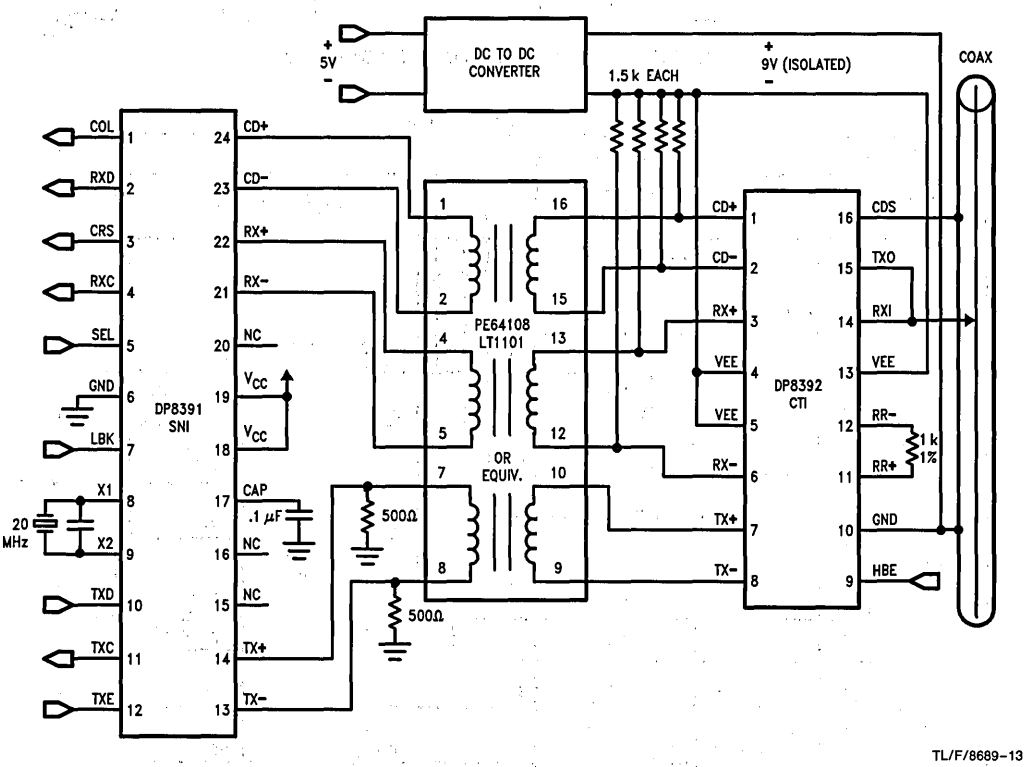
The pin assignment of both the CTI and the SNI are designed to minimize the crossover of any printed circuit traces. Some of the components needed for an Ethernet like interface are not needed for Cheapernet. For instance, Cheapernet's relaxed load capacitance (8 pF, compared with 4 pF for Ethernet) obviates the need for an external capacitance isolation diode at TXO. Also, since the transceiver drop cable is not used in Cheapernet, there's no need for the 78Ω termination resistors. Moreover, without the 78Ω loading on the differential outputs, the pulldown resistors for both the CTI's collision and receive drivers and the SNI's transmit driver can be larger to save power. These resistors can be 1.5k instead of 500Ω for the CTI and 500Ω instead of 270Ω for the SNI.

The 20 MHz crystal connection to the SNI requires special care. The IEEE 802.3 standard requires a 0.01% absolute accuracy on the transmitted signal frequency. An external capacitor between the X1 and X2 pins is normally needed to get the required frequency range. Section 3.1 of the data sheet describes how to choose the value of this capacitor.

The SNI also provides loopback capability for fault diagnosis. In this mode, the Manchester encoded data is internally diverted to the decoder input and sent back to the controller. Thus both the encoding and the decoding circuits are tested. The transmit differential output driver and the differential input receiver circuits are disabled during loopback. This mode can be enabled by a TTL active high input at pin 7.

Two different modes, half step and full step, can be selected at the SNI's transmit output. The standards require half step mode of operation, where the output goes to differential zero during idle to eliminate large idle currents through the pulse transformers. On the other hand, the differential output remains in a fixed state during idle in full step mode. The SNI thus can be used with transceivers which work in either mode. The two different modes can be selected with a TTL input at pin 5.

Figure 9 shows a typical Cheapernet connection diagram using the DP8391 and the DP8392.



TL/F/8689–13

FIGURE 9. Cheapernet Connection Diagram

The power isolation is similar here as in the Ethernet application, except the DC input is now usually 5V instead of 12V. Hybrid DC to DC converters are also being developed for this application (Ex: Pulse Engineering PE64381). *Figure 10* shows a discrete implementation with 5V input and −9V output.

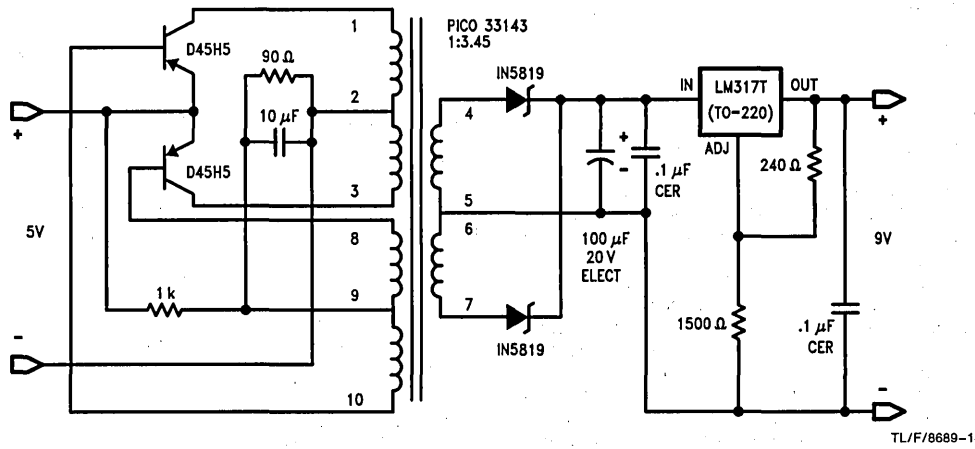


FIGURE 10. DC to DC Converter (5V to −9V)

Measuring Ethernet Tap Capacitance

National Semiconductor
Application Note 757
Larry Wakeman

INTRODUCTION

When a node is added to an Ethernet network, its nodal capacitance changes the impedance of the cable at the point of connection to the cable. The impedance change causes a reflection of the Ethernet waveform, which distorts the waveform. The more the capacitance the greater the distortion, and eventually with large enough node capacitances the Ethernet signal could become so distorted that the packet data would become corrupted when decoded by a network node. For this reason the IEEE802.3 standard specifies a maximum value of capacitance that a node may add to the network, as well as a minimum node to node distance spacing. Since the capacitance of a node includes stray inductances, the effective capacitance of a node connection cannot be measured simply by using a capacitance meter. This note presents the method for measuring capacitance of an Ethernet tap for 10BASE5 or a BNC "T" for 10BASE2.

THE STANDARD'S REQUIREMENTS

To properly make the measurement, it is important to understand how the standard specifies the capacitance of a node. To quote the IEEE802.3 standard:

8.3.1.1 Input Impedance: The shunt capacitance presented to the coaxial cable by the MAU circuitry (not including the means of attachment to the coaxial cable) is recommended to be no greater than 2 pF. The resistance to the coaxial cable shall be greater than 100 kΩ.

The total capacitive load due to MAU circuitry and the mechanical connector as specified in 8.5.3.2 shall be no greater than 4 pF.

These conditions shall be met in the power-off and power-on, not transmitting states (over the frequencies BR/2 to BR).

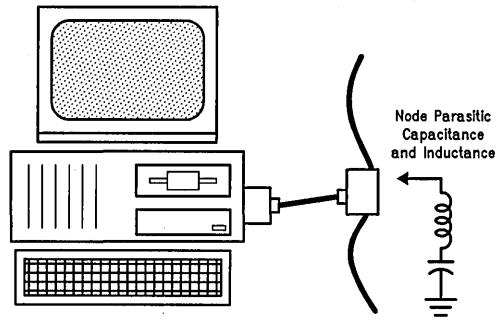
The magnitude of the reflection from a MAU shall not be more than that produced by a 4 pF capacitance when measured by both a 25 ns rise time and 25 ns fall time waveform. This shall be met in both the power-on and power-off, not transmitting states.

To summarize the maximum allowable capacitance specifications for both Thinwire and Thickwire Ethernet the following table is provided.

TABLE I. Maximum Capacitance Allowed in IEEE802.3

Standard	Electrical Circuitry	Mechanical Connector
10BASE5	2 pF	2 pF
10BASE2	4 pF	4 pF

Note: Thickwire or Thick Ethernet refers to 10BASE5 and Thinwire or Thin Ethernet refers to 10BASE2.



TL/F/11163–1

FIGURE 1. Simple Model of the Parasitics Presented to the Ethernet Cable

THE TEST METHOD

Due to the nature of the capacitance of a DTE (Data Terminal Equipment), rather than perform a simple capacitive measurement using a meter, the capacitance of the network node is more accurately measured by testing it in an environment where the actual signal reflection caused by the capacitance of a node attachment is measured when applying a typical Ethernet signal. The magnitude of the reflection is then correlated to an equivalent capacitance. This is the most appropriate method, since it is the signal degradation due to the capacitive load that is the important consideration in defining the above specifications.

With the above in mind, the test is performed by first measuring the reflection caused by the attachment of a node. Then the DTE is replaced with a reference variable capacitor, and the capacitor's value is adjusted until the capacitance that causes the same size reflection is determined. The capacitance of the node is therefore the same as the reference capacitance value that causes the same amplitude reflection.

TEST SETUP AND CABLE

An example test configuration which measures the capacitance of the Thickwire Ethernet is shown in *Figure 2.* The waveform applied to the test node is an important consideration in setting up the test, as it will affect the resultant value of capacitance. In particular the rise and fall times must be carefully chosen to reflect the capacitance seen in an Ethernet network, as described in the next section.

The cable lengths and spacing between the scope input and the transceiver's connection are chosen to ensure that the reflection due to the transceiver appears on the flat portion of the test waveform. This allows accurate measurement. The total cable length is equivalent to the full 10BASE5 length of 500m.

An oscilloscope is used to measure the voltage of the reflection. The scope, with a 1 MΩ input impedance, as shown in *Figure 2,* is connected directly to the cable without a probe. This eliminates any errors due to the probe. The distance between transceiver connection point "A" and the scope is set so that the reflections will arrive at the scope right after the signal rise and fall times. Moving point "A" any further makes the reflections smaller in amplitude (cable attenuation) and therefore harder to measure.

On the scope's display measurements are made at the point immediately after the rise time. Reflections are then compared to the ones for known discrete capacitors.

THE TEST WAVEFORM

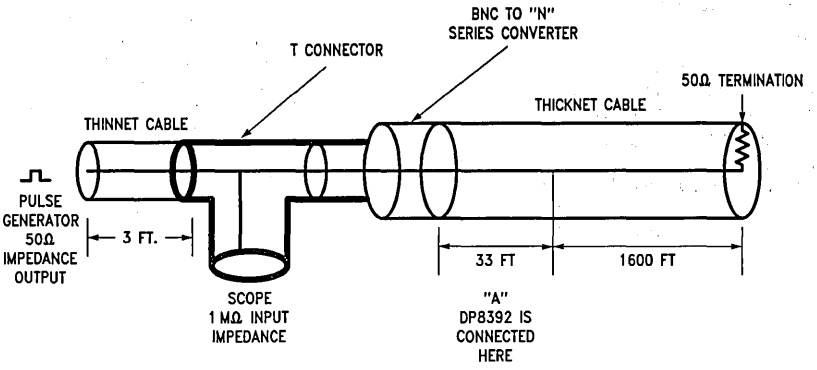
In normal network operation the signal on the coax cable has rise and fall times of 25 ns ±5 ns (defined by the IEEE802.3 standard). With a purely capacitive load applying signals with faster (or slower) edges cause larger (or smaller) reflections than would be seen on a typical network. If the node were purely capacitive this would not affect the measurement. The larger (or smaller) node reflection for a given parasitic capacitance would track with the reference capacitance's reflection yielding accurate measurements.

However, the node is actually not a pure capacitance, but has some series inductance associated with the network connection as shown in *Figure 1.* The application of signals with faster than 20 ns rise and fall times actually result in an unrealistically low capacitance measurement. This is because the nodes capacitance is buffered by the stray series inductances which reduce the reflection magnitude when compared to the pure capacitance. This correlates to a lower than actual capacitance.

On the other hand applying very slow rise and fall times (slower than 30 ns) result in the measurement of a larger capacitance than actual. This is because the series inductance effects are less than would be seen with a nominal waveform.

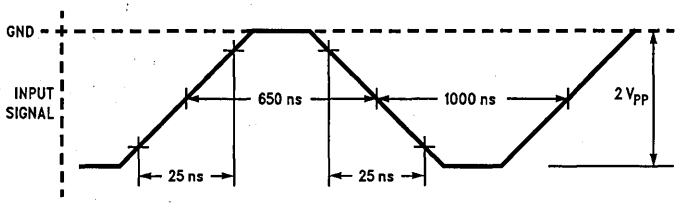
Since it is desirable to measure the capacitance in such a way as to correlate to the effective capacitance seen when IEEE802.3 signaling is used, the best compromise choice is to select a 25 ns rise and fall times for this test. (This is the reason for this choice in the actual standard.)

Again, the reason behind this decision is that although the ≥ 30 ns edges indicate larger capacitances a signal with 25 ns edge produces results that more correctly represent the actual effect of the attached node's capacitance.



TL/F/11163–2

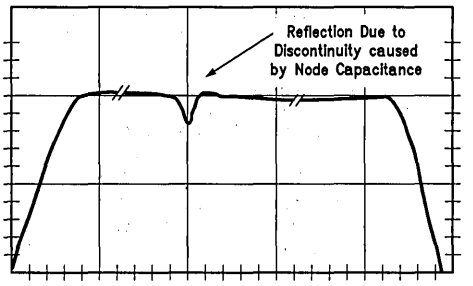
FIGURE 2. Test Setup



TL/F/11163–3

FIGURE 3. Input Test Waveform

As shown in *Figure 3,* a low frequency trapezoidal signal is used. This will keep the reflections from each edge of the signal well away from the next edge enabling easier measurement. The 2 V_{PP} test input signal is the typical voltage swing on the coax cable in normal operation. In the case of a discrete capacitor the voltage level of the signal may not be important. However, due to the non-linearity of the node and DP8392 capacitance a typical voltage signal should be used following the same rational as was used for the signal rise and fall times.



TL/F/11163–4

Note: This figure is conceptual. It does not show the waveform details.

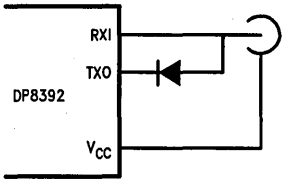
FIGURE 4. Example of Reflection

TEST RESULTS

A special jig was built to connect the ICs to point "A" in *Figure 2.* This greatly improves measurement repeatability. Data repeatability of 0.01 pF is achieved.

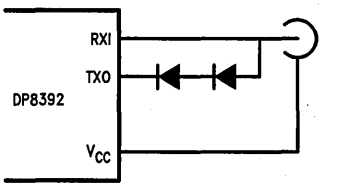
Typical data for RXI and TXO capacitances are 1.0 pF and 2.0 pF respectively. Total node capacitance can be reduced to around 1.6 pF with the addition of a small capacitance diode in series with the TXO output, as shown in *Figure 5.* For Ethernet applications two diodes in series can be used instead.

Thinwire



TL/F/11163–5

Thickwire



TL/F/11163–6

FIGURE 5. DP8392 Connection Diagram

INACCURACIES OF THE CAPACITANCE METER

As stated, in a real network, it is not the node capacitance that creates a problem, but too large a reflection caused by this capacitance. This reflection distorts the cable signal. Therefore the best method of test is to measure the reflection under true network waveforms. By the same analogy capacitance meters which have a test signal frequency that does not correspond to 25 ns rise and fall time do not reveal a true measurement of capacitance, and so capacitive measurements done only with a capacitance meter are usually (almost always) inaccurate to the true effective capacitance as seen by the network cable.

Interfacing the DP8392 to 93Ω and 75Ω Cable

National Semiconductor
Application Note 620
Mohammed Rajabzadeh

The DP8392 Ethernet Coaxial Transceiver Interface (CTI) is designed primarily for 10BASE2 and 10BASE5 applications which use 50Ω coaxial cable. However, with minor modifications it is possible to use this transceiver with larger impedance cables. This article shows how to use the DP8392 with 75Ω or 93Ω cable. The trade off is that segment span is reduced to accommodate for higher series DC resistance of these cables. The CTI is a current driver. The two important factors that must be handled properly in using the chip with 75Ω and 93Ω cables are the dynamic range of the transmitter and collision detection levels.

DYNAMIC RANGE

The dynamic range of the transmitter is important in the following case:

Suppose two stations collide with one-another. To detect collisions properly, each station must sink at least as much DC current as it would in a non-collision case. This would mean that with the 93Ω cable when a collision occurs the chips should be able to sustain approximately −4V DC level. If the signals from the colliding stations are in phase the AC signal could be 8V peak to peak.

The DP8392's transmitter clamps before it pulls to −8V. However, when it clamps it also changes the duty cycle enough to sustain the −4V DC collision level.

An internal diode is included in series with the transmitter's output to isolate its capacitance and thereby minimizing the tap capacitance. For more dynamic range margin, it is recommended that external isolating diodes at the transmitter output not be used. It is also advisable to design the power supply to operate at the higher end of the 8.55V to 9.45V range.

COLLISION LEVELS—RECEIVE MODE

In order to understand the concerns with collision levels, it is necessary to calculate the levels for Cheapernet (10BASE2) 50Ω cable (RG58AU) as an example.

50Ω Cable Example (RG58A/U)

Table I shows the parameter values that are used in calculating the collision levels. Please note that all the levels in this article are for receive mode collision detection.

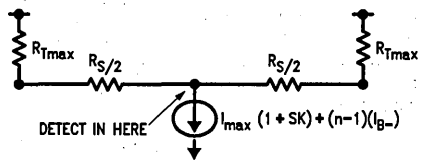
TABLE I. Assumptions and Definitions

R_T	= Termination Resistor at 20°C	= 50 ±1%	802.3
t_T	= Temp. Coef. of the Terminator	= 0.0001/°C	ASSUMPTION
L	= Maximum Segment Length	= 185m	802.3
R_{DC}	= Maximum Cable DC Res. at 20°C	= 0.0489Ω/m	BELDEN
tc	= Temp. Coef. of Copper	= 0.004/°C	PHYSICS
Tm	= Maximum Cable Temp.	= 50°C	ASSUMPTION
SR	= Step Response at Max Cable Length	= 0.98	NATIONAL
R_C	= Max Connector Res./Station	= 0.0034Ω	MIL SPEC
I_{B+}	= Max Positive Bias Current	= 2 μA	802.3
I_{B-}	= Max Negative Bias Current	= 25 μA	802.3
I_{max}	= Max DC Drive Current	= 45 mA	802.3
I_{min}	= Min. DC Drive Current	= 37 mA	802.3
R_O	= Non Transmitting Output Impedance	= 100 kΩ	802.3
N	= Max Nodes per Segment	= 30	802.3
SK	= Skew Factor, Effect of Encoder Skew on DC Level = (SKEW × 4)/100	= 0.02 for 0.5 ns Skew	802.3
R_S	= Max DC Loop Res. of a Segment		DEFINITION
R_L	= Load Resistance Seen by a Driver		DEFINITION
SEO	= Sending End Overshoot	= 0.08	ASSUMPTION

The collision levels that need to be calculated are V_{max} and V_{min}. The V_{max} or "no detect" level is the maximum DC voltage generated by one node. The worst case here occurs when the transmitting node is at the center of a maximum length cable, and the collision is being detected either by itself or by a station right next to it. On the other hand, the V_{min} or "must detect" level is the minimum DC voltage generated by two minimum stations transmitting at one end of a maximum length cable, and the collision is being detected by a node on the other side of the cable.

The filter impulse response is not included in these calculations since it is mutually exclusive with the Sending End Overshoot. If the impulse response is larger than the Sending End Overshoot, the exceeding portion should be added on to the limits.

Maximum Non Collision Level V_{max} (No-Detect)—Receive Mode—50Ω Cable



TL/F/10444–1

$R_{Tmax} = R_T \times 1.01 \times [(Tm - 20) \times t_T + 1]$

$R_S = R_{DC} \times L \times [(Tm - 20) \times tc + 1] + N \times R_C$

$R_L = (R_{Tmax} + R_S/2)/2$

$V_{max} = [I_{max} \times (1 + SK) + (N - 1)(I_{B-})] \times R_L \times (1 + SEO)$

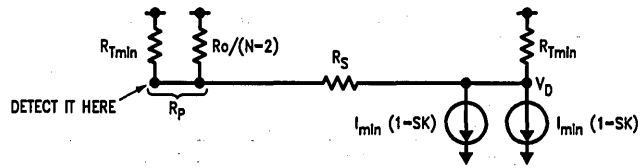
$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1] = 50.652\Omega$

$R_S = 0.0489 \times 185[(50 - 20) \times 0.004 + 1] + 30 \times 0.0034 = 10.234\Omega$

$R_L = (50.652 + 10.234/2)/2 = 27.885\Omega$

$V_{max} = [45 \times 1.02 + 29 \times 0.025] \times 27.885 \times 1.08 = \mathbf{1404\ mV}$

Minimum Collision Level V_{Min} (Must-Detect)—Receive Mode—50Ω Cable



TL/F/10444–2

R_P = NEAR END SHUNT RESISTANCE
$= [R_O/(N - 2)]//R_{Tmin}$

$R_{Tmin} = R_T \times 0.99$

V_D = TRANSMITTER'S END DC VOLTAGE
$= 2 \times I_{min} \times (1 - SK) \times [R_{Tmin}//(R_S + R_P)]$

$V_{min} = V_D \times [R_P/(R_S + R_P)] \times SR$

$R_P = [100k/28]//(50 \times 0.99) = 3571//49.5 = 48.823\Omega$

$V_D = 2 \times 37 \times 0.98 \times [49.5//(10.234 + 48.823)] = 1952\ mV$

$V_{min} = 1952 \times [48.823/(10.234 + 48.823)] \times 0.98 = \mathbf{1581\ mV}$

The calculations show that the V_{max} and V_{min} are properly placed outside the collision threshold range of the DP8392 (1450 mV to 1580 mV).

93Ω Cable Collision Level Calculation

A few parameters need to be changed when using a different impedance cable. Here are those parameters for 93Ω cable (RG62A/U TYPE, BELDEN 9269);

TABLE II

R_T	= termination resistor at 20°C	= 93 ±1%	
L	= maximum segment length	= 130m	
R_{DC}	= maximum cable DC res. at 20°C	= 0.1437 Ω/m	BELDEN

Considering the new values the V_{max} and V_{min} levels are;

Maximum Non Collision Level V_{max} (No Detect)—Receive Mode—93Ω Cable

R_{Tmax}	= 93 × 1.01 × [(50 − 20) × 0.0001 + 1]	= 94.212Ω
R_S	= 0.1437 × 130[(50 − 20) × 0.004 + 1] + 30 × 0.0034	= 21.025Ω
R_L	= (94.212 + 21.025/2)/2	= 52.362Ω
V_{max}	= [45 × 1.02 + 29 × 0.025] × 52.362 × 1.08	= **2636.692 mV**

Minimum Collision Level V_{min} (Must Detect)—Receive Mode—93Ω Cable

R_P	= [100k/28]//(93 × 0.99) = 3571//92.07	= 89.756Ω
V_D	= 2 × 37 × 0.98 × [92.070//(21.025 + 89.756)]	= 3646.396 mV
V_{min}	= 3646.396 × [89.756/(21.025 + 89.756)] × 0.98	= **2895.272 mV**

93Ω IMPLEMENTATION WITH DP8392

Figure 1 shows the connection diagram with 93Ω cable (100 meters and 30 stations). The design parameters defined below are summarized in Table III. The resistor divider ratio needs to be calculated to attenuate the receiver input signal. The two resistors R_1 and R_2 should center the calculated thresholds (2636 mV to 2895 mV) to the internal level of DP8392 (1450 mV to 1580 mV).

The resistor divider and the capacitor Cp, *Figure 1*, (Cp includes the RXI input capacitance, typically 1 pF, and the pc trace capacitance associated with it) form a low pass filter effect. It may be necessary to add the capacitor Cc (capacitor Cc creates a high pass effect) to compensate the low pass effect. The equation to calculate the capacitor Cc is;

$$Cc \times R_2 = Cp \times R_1$$

It is also necessary to add the resistor R_3 ($R_3 = R_1//R_2$) in series with the CDS pin. This will assure that the voltage drop due to the biasing currents into CDS and RXI pins are duplicated.

To check the design;

[54.8k/(54.8k + 45.2k)] × 2636 mV = 1444 mV
[54.8k/(54.8k + 45.2k)] × 2895 mV = 1586 mV

The DP8932's internal collision range is within this window.

75Ω CABLE IMPLEMENTATION

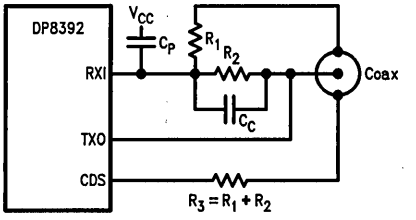
This method can also be successfully implemented for 80 meters of 75Ω cable (RG59/U BELDEN 8241). The collision thresholds are 2127.8 mV and 2339.6 mV. The corresponding R_1 and R_2 values are 67.8 kΩ and 32.2 kΩ respectively. Table IV summarizes the design parameters.

TABLE III

CABLE	BELDEN RG62A/U Type 93Ω Cable
L	130 meters
R_{DC}	0.1437 Ω/m
N	30
R_1	54.8k
R_2	45.2k

TABLE IV

CABLE	BELDEN RG59/U 75Ω Cable
L	80 meters
R_{DC}	0.1894 Ω/m
N	30
R_1	67.8k
R_2	32.2k



TL/F/10444–3

FIGURE 1

Designing the DP8392 for Longer Cable Applications

National Semiconductor
Application Note 621
Mohammed Rajabzadeh

The IEEE 802.3 standard is designed for 500 meters of Ethernet cable and 185 meters of Cheapernet (RG58A/U) cable. To extend such segments to 1000 meters of Ethernet cable and 300 meters of Cheapernet cable requires utilization of Transmit mode collision detection. This method is described below.

COLLISION DETECTION SCHEMES

The collision circuitry monitors the coaxial DC level. If the level is more negative than the collision threshold, the collision output is enabled.

There are two different collision detection schemes that can be implemented with the CTI; Receive mode, and Transmit mode. The IEEE 802.3 standard allows the use of receive and transmit modes for non-repeater node applications. Repeaters are required to have to receive mode implementation. These different modes are defined as follows:

Receive Mode: Detects a collision between any two stations on the network with certainty at all times.

Transmit Mode: Detects a collision with certainty only when the station is transmitting.

Table I summarizes the receive and transmit mode definitions:

TABLE I

Mode	Receive				Transmit			
No. of Stations	0	1	2	>2	0	1	2	>2
Transmitting	N	N	Y	Y	N	N	Y	Y
Non-Transmitting	N	N	Y	Y	N	N	M	Y

Y = Detects Collision
N = Does Not Detect Collision
M = Might Detect Collision

Receive Mode: The Receive mode scheme has a very simple truth table. However, the tight threshold limits make the design of it difficult. The threshold in this case has to be between the maximum DC level of one station (−1300 mV) and the minimum DC level of two far stations (−1581 mV). Several factors such as the termination resistor variation, signal skew, and input bias current of non-transmitting nodes contribute to this tight margin. On top of the −1300 mV minimum level, the impulse response of the internal low pass filter has to be added. The CTI incorporates a 4-pole Bessel filter in combination with a trimmed on board bandgap reference to provide this mode of collision detection.

Transmit Mode: In this case, collision has to be detected only when the station is transmitting. Thus, collision caused by two other nodes may or may not be detected. This feature relaxes the upper limit of the threshold. As a result of this, longer cable segments can be used. With the CTI, a resistor divider can be used at the Collision Detection Sense (CDS) pin to lower the threshold from receive to transmit mode.

COLLISION LEVELS—TRANSMIT MODE

Table II shows the parameter values that are used in calculating the collision levels in transmit mode.

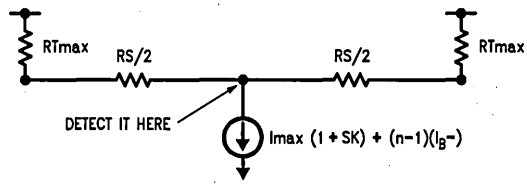
TABLE II. Assumptions and Definitions

R_T	= Termination Resistor at 20°C	= 50 ±1°%Ω		802.3
t_T	= Temp. Coef. of the Terminator	= 0.0001/Deg.		ASSUMPTION
L	= Maximum Segment Length	= 300m	Cheapernet	802.3
		= 1000m	Ethernet	802.3
R_{DC}	= Maximum Cable DC Res. at 20°C	= 0.0489Ω/m	Cheapernet	BELDEN
		= 0.0100Ω/m	Ethernet	BELDEN
tc	= Temp. Coef. of Copper	= 0.004/°C		PHYSICS
Tm	= Maximum Cable Temp.	= 50°C		ASSUMPTION
SR	= Step Response at Max Cable Length	= 0.97	Cheapernet	NATIONAL
		= 0.94	Ethernet	NATIONAL
R_C	= Max. Connector Res./Station	= 0.0034Ω	Cheapernet	MIL SPEC
		= 0.0001Ω	Ethernet	ASSUMPTION
I_{B+}	= Max. Positive Bias Current	= 2 μA		802.3
I_{B-}	= Max. Negative Bias Current	= 25 μA		802.3
I_{max}	= Max. DC Drive Current	= 45 mA		802.3
I_{min}	= Min. DC Drive Current	= 37 mA		802.3
Ro	= Non Transmitting Output Impedance	= 100 kΩ		802.3
N	= Max Nodes per Segment	= 100	Cheapernet	802.3
		= 100	Ethernet	802.3
SK	= Skew Factor, Effect of Encoder Skew on DC Level			802.3
	= (SKEW×4)/100	= 0.02 for 0.5 ns Skew		
R_S	= Max. DC Loop Res. of a Segment			DEFINITION
R_L	= Load Resistance Seen by a Driver			DEFINITION
SEO	= Sending End Overshoot	= 0.10	Cheapernet	ASSUMPTION
		= 0.14	Ethernet	ASSUMPTION

The calculations below explain how the values for the resistor divider in *Figure 1* are obtained. First, collision levels V_{max} and V_{min} must be calculated. The V_{max} or "no detect" level is the maximum DC voltage generated by one node. The worst case here occurs when the transmitting node is at the center of a maximum length cable, and the collision is being detected either by itself or by a station right next to it. On the other hand, the V_{min} or "must detect" level is the minimum DC voltage generated by one minimum transmitting station and another minimum transmitting station at the other end of a maximum length cable.

The filter impulse response is not included in these calculation since it is mutually exclusive with the Sending End Overshoot. If the impulse response is larger than the Sending End Overshoot, the exceeding portion should be added on to the limits.

Maximum Non Collision Level V_{Max} (NO DETECT)—Transmit Mode



TL/F/10445–1

$R_{Tmax} = R_T \times 1.01 \times [(Tm - 20) \times t_T + 1]$

$R_S = R_{DC} \times L \times [(Tm - 20) \times t_c + 1] + N \times RC$

$R_L = (R_{Tmax} + R_S/2)/2$

$V_{Max} = [I_{Max} \times (1 + SK) + (N - 1)(I_{B-})] \times R_L \times (1 + SEO)$

CHEAPERNET Cable, 300 Meters, 100 Stations:

$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1] = 50.652\Omega$

$R_S = 0.0489 \times 300\ [(50 - 20) \times 0.004 + 1] + 100 \times 0.0034 = 16.770\Omega$

$R_L = (50.652 + 16.770/2)/2 = 29.519\Omega$

$V_{Max} = [45 \times 1.02 + 99 \times 0.025] \times 29.519 \times 1.10 = \mathbf{1571\ mV}$

ETHERNET Cable, 1000 Meters, 100 Stations:

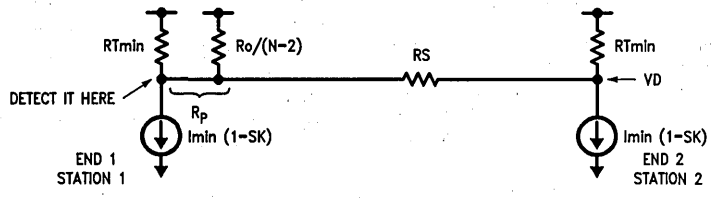
$R_{Tmax} = 50 \times 1.01 \times [(50 - 20) \times 0.0001 + 1) = 50.652\Omega$

$R_S = 0.01 \times 1000\ [(50 - 20) \times 0.004 + 1] + 100 \times 0.0001 = 11.21\Omega$

$R_L = (50.652 + 11.21/2)/2 = 28.129\Omega$

$V_{Max} = [45 \times 1.02 + 99 \times 0.025] \times 28.129 \times 1.14 = \mathbf{1551\ mV}$

Minimum Collision Level V_{Min} (MUST DETECT)—Transmit Mode



TL/F/10445–2

R_P = Near End Shunt Resistance
= $[Ro/(N - 2)]//R_{Tmin}$

$R_{T_{Min}}$ = $R_T \times 0.99$

VS1(1) = Station 1's DC Voltage at End 1
= $I_{Min} \times (1 - SK) \times [R_P//(R_S + R_{Tmin})]$

VS2(2) = Station 2's DC Voltage at End 2
= $I_{Min} \times (1 - SK) \times [R_{Tmin}//(R_S + R_P)]$

VS2(1) = Station 2's DC Voltage at End 1
= VS2(2) × $[R_P/(R_S + R_P)]$ × SR

V_{Min} = VS1(1) + VS2(1)

CHEAPERNET Cable, 300 Meters, 100 Stations:

R_P = [100k/98]//(50 × 0.99)
= 1020//49.5 = 47.209Ω

VS1(1) = 37 × 0.98 × [47.209//(16.770 + 49.5)]
= 1000 mV

VS2(2) = 37 × 0.98 × [49.5//(16.770 + 47.209)]
= 1012 mV

VS2(1) = 1012 × [47.209/(47.209 + 16.770)] × 0.97
= 724 mV

V_{Min} = 1000 + 724 = **1724 mV**

ETHERNET Cable, 1000 Meters, 100 Stations:

R_P = [100k/98]//(50 × 0.99) = 1020//49.5
= 47.209Ω

VS1(1) = 37 × 0.98 × [47.209//(11.21 + 49.5)]
= 963 mV

VS2(2) = 37 × 0.98 × [49.5//(11.21 + 47.209)]
= 972 mV

VS2(1) = 972 × [47.209/(47.209 + 11.21)] × 0.94
= 738 mV

V_{Min} = 963 + 738 = **1701 mV**

CIRCUIT IMPLEMENTATION

Table III summarizes the design parameters.

TABLE III

Parameter	ETHERNET	CHEAPERNET
L	1000 Meter	300 Meter
N	100	100
V_{Min}	1701 mV	1724 mV
V_{Max}	1551 mV	1571 mV
R_1	125Ω ±1%	150Ω ±1%
R_2	10 kΩ ±1%	10 kΩ ±1%

Circuit implementation is shown in *Figure 1*

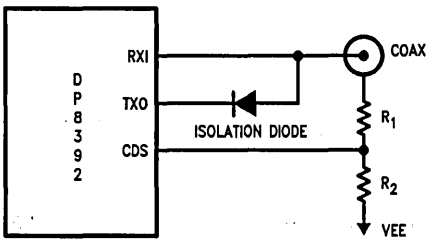


FIGURE 1

To check the design, subtract the additional offset generated by the resistor divider from these levels (V_{Max} and V_{Min}) and make sure that the internal 8392 collision levels (1450 mV to 1580 mV) are within this window. The supply voltage is assumed to be 9V±5%.

Ethernet

1551 mV − 8.55V (125Ω/(10 kΩ + 125Ω)) = 1445 mV

1701 mV − 9.45V (125Ω/10 kΩ + 125Ω)) = 1584 mV

Cheapernet

1571 mV − 8.55V (150Ω/(10 kΩ + 150Ω) = 1445 mV

1724 mV − 9.45V (150Ω/(10 kΩ + 150Ω) = 1584 mV

These calculations show that the resistor values are properly selected.

National Semiconductor

PRELIMINARY

DP83922A
Twisted Pair Transceiver Interface (TPI)

General Description

The DP83922A Twisted Pair Transceiver is used to connect IEEE 802.3 stations and repeaters to a twisted-pair cable medium. It integrates all the transceiver medium attachment unit (MAU) functions as specified in the IEEE 802.3 10BASE-T standard.

The DP83922A contains a full AUI interface, which makes it ideal for use both in stand-alone and integrated MAU applications. The DP83922A's primary functional blocks include transmitter, receiver, collision detection, jabber timer, link test, and SQE test (CD Heartbeat) with a disable pin for repeater applications.

The DP83922A also provides display outputs that directly drive LEDs to indicate status for all MAU functions. The TPI indicates receive and transmit activity, jabber and collision indication, and link (cable connectivity) status.

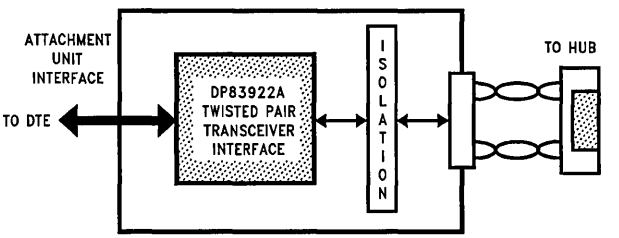
The DP83922A is part of a chip set that implements the complete IEEE802.3 10BASE-T compatible network electronics. In node applications, it can be used with the DP83910 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC) or the DP83932 SONIC™. The TPI may also be used in repeater applications.

Features

- Compatible with IEEE802.3 10BASE-T standard
- Integrates transceiver electronics, including:
  - — Transmitter
  - — Receiver
  - — Collision Detection
  - — CD Heartbeat
  - — Jabber Timer
  - — Link Integrity Test
- Link disable enables operation with pre-standard 10BASE-T twisted pair implementations
- Fully AUI compatible interface for both stand-alone and embedded MAU applications
- Programmable transmit and equalization levels
- Complete differential transmit and receive path for optimum jitter performance
- Transmit output waveform shaping reduces filter requirements
- Status LED Outputs
  - — Link, Transmit, Receive, Jabber, and Collision
- 24-pin narrow DIP package

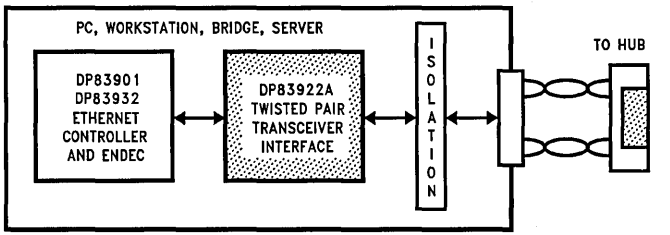
System Diagrams

Stand-Alone Twisted Pair MAU



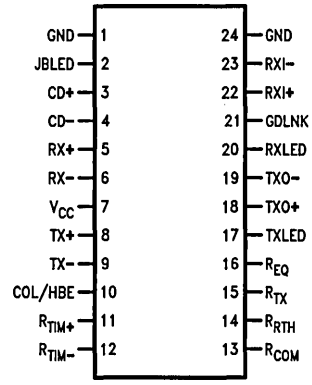
TL/F/10490–1

Node with Integrated Twisted Pair MAU



TL/F/10490–2

Connection Diagram



TL/F/10490–3

Top View

Order Number DP83922AN
See NS Package Number N24C

Pin Description

Pin No.	Pin Name	I/O	Description
1, 24	GND	—	**Ground Supply Pin**
2	JBLED	O	**Jabber LED:** This output indicates that the DTE or repeater transmitting into the TX± is jabbering (transmitting an excessively long packet). This is indicated when this output is low. This information may be displayed using a Light Emitting Diode (LED).
3 4	CD+, (CI+) CD−, (CI−)	O (AUI*)	**Collision Output:** Balanced differential line driver outputs. The 10 MHz signal from the internal oscillator is transferred to these outputs in the event of collision, excessive transmission (jabber), or during SQE Test. These outputs are emitter followers and require external pulldown resistors to GND.
5 6	RX+, (DI+) RX−, (DI−)	O (AUI)	**Receive Output:** Balanced differential line driver outputs from the TPI's receiver. These outputs are also emitter followers and require pulldown resistors to GND.
7	V_{CC}	—	**Positive 7V Supply:** A 0.1 μF ceramic decoupling capacitor must be connected across GND and V_{CC} as close to the device as possible.
8 9	TX+, (DO+) TX−, (DO−)	I (AUI)	**Transmit Input:** Balanced differential inputs to the twisted pair transmitter. The common mode voltage for these inputs is set internally and must not be externally biased.
10	COL/HBE	I/O	**Collision LED/Heartbeat Enable:** This dual-function pin disables the Heartbeat (SQE test) function when it is tied to GND. When connected to V_{CC} through an LED and series resistor, this output indicates that a collision is occurring.
11 12	R_{TIM+} R_{TIM-}	I	**Timing Resistor Pins:** A resistor is connected across these pins. This resistor is used to set the timing reference for the device.
13	R_{COM}	I	**Common Resistor Input:** Resistors are attached to this pin. This input is the common reference point for the R_{RTH}, R_{TX}, and R_{EQ}.
14	R_{RTH}	I	**Receiver Threshold Resistor:** A resistor attached to this pin sets the threshold of the twisted pair receiver squelch circuit.

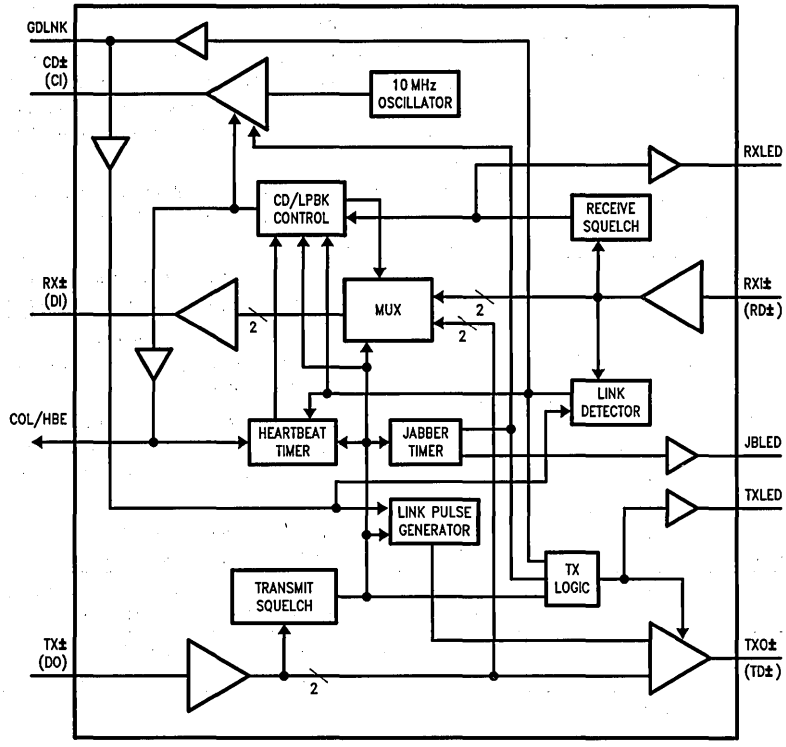
*AUI—Pins connect to Attachment Unit Interface.

Pin Description (Continued)

Pin No.	Pin Name	I/O	Description	
15 16	R_{TX} R_{EQ}	I	**Transmit Level Resistor** **Equalization Resistor**	Resistors connected to these pins are used to determine the transmit levels for the equalized output.
17	TXLED	O	**Transmit LED:** This output goes low when valid transmissions are output from the TPI. When the transmitter is disabled due to a Jabber or link fault this output is off (TRI-STATE®). This can be used to drive a status LED.	
18 19	TXO+ (TD+) TXO− (TD−)	O (TP*)	**Twisted Pair Transmit Outputs:** Balanced differential current drivers that allow external resistors and filter network to determine the output impedance. (the details of the operation of these outputs can be found in the functional description)	
20	RXLED	O	**Receive LED:** This output goes low when data is received by the TPI. This can be used to drive a status LED.	
21	GDLNK	I/O	**Good Link:** This dual-function pin indicates successful reception of link pulses by going low. This information may be displayed using a Light Emitting Diode (LED). When this pin is tied to ground the link generator, and link receive state logic is disabled.	
22 23	RXI+ (RD+) RXI− (RD−)	I (TP)	**Twisted Pair Receive Inputs:** These high impedance inputs feed a differential amplifier which then transmits the signal differentially along the receive path. The common mode voltage for these inputs is set internally and should not be altered.	

*TP—Pins connect to Twisted Pair Interface.

Block Diagram



TL/F/10490–4

Functional Description

The TPI consists of 6 basic functional blocks:

1. Receiver: This block receives data from the twisted pair wire and sends it to the DTE (Data Terminal Equipment) via the Attachment Unit Interface (AUI). In the block diagram this section is composed of the RXI± input receiver, receive squelch, the MUX and the RXI output driver.
2. Transmitter: This accepts data from the DTE connected via the AUI and transmits it onto the twisted pair cabling. In the block diagram this section is composed of the TX± input receiver, transmit squelch, and the TXO± output driver.
3. Collision Detection: This indicates to the DTE that a collision is occurring by transmitting a 10 MHz signal to the DTE. In the block diagram this section consists of the CD/Loopback Control, the 10 MHz oscillator and the CD± output driver.
4. Jabber Timer: This disables the transmitter if it was transmitting a longer than legal packet.
5. SQE (Signal Quality Error) Test: This is accomplished by the Heartbeat Timer block which generates a short burst of collision signal after every transmission.
6. Link Test: This block consists of the Link Pulse Generator and Link Detector. This block checks the integrity of the cable connecting two twisted pair MAUs.

RECEIVE FUNCTION

Receive Path

The receive path logic consists of the twisted pair receiver, loopback multiplexer, and AUI driver. From input to output the receive path is fully differential and both RX+ and RX− are matched. This provides low signal skews and jitter.

After the received signal passes through the differential input buffer, it enters the loopback multiplexer. The multiplexer routes either the received data or transmit data to the RX± pins. When the TPI is receiving valid data from the twisted pair cable, this data is sent to RX± outputs.

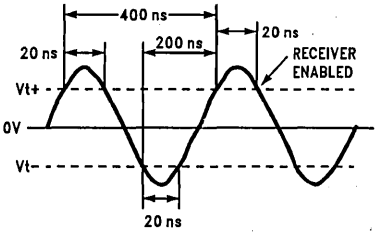
The multiplexer output is enabled by the CD/loopback control block. This block will enable data to the receiver's driver when receive squelch is turned off. (i.e., when valid data is present on RXI±). In addition, the RX± outputs have internal 5 kΩ pull-down resistors. These resistors eliminate the need for external pull downs when the TPI is located near the ENDEC.

The differential RX± line driver provides ECL compatible signals to the DTE with typically 3 ns rise and fall times. In its idle state, these outputs go to differential zero to prevent DC standing current in the isolation transformer.

Receive Squelch

The DP83922A implements an intelligent receive squelch intended to ensure external noise that appears on the receiver's inputs is not mistaken for a valid signal. This squelch uses a combination of amplitude and timing measurements to determine when to enable the receive circuitry. The operation of the squelch is as follows: An input signal must first exceed the input voltage threshold for typically 20 ns, and then typically within 150 ns later it must exceed the opposite input threshold for the same 20 ns duration. Finally the signal must exceed the original threshold within typically 150 ns after the last threshold detection time in order to turn off the receiver squelch and enable the receive circuitry. This is shown in the following illustration.

Squelch Timing for Receive Input



TL/F/10490–5

The voltage threshold level of the receive squelch is set by an external resistor, R_{RTH}. There are two levels, one level to enable the receiver (V_{Ron}), and a second smaller level to disable the receiver (V_{Roff}), providing hystersis. V_{Ron} is one third the voltage across the R_{RTH} resistor (V_{RTH}). V_{RTH} is given by:

$$V_{RTH} = \left(\frac{I_{TX(out)}}{48}\right) R_{RTH}$$

where $I_{TX(out)}$ is the total transmit current set by R_{TX} and R_{EQ} (see the description of the transmit function that follows). This equation can be rewritten in terms of R_{TX} and R_{EQ} as follows:

$$V_{RTH} = 1.245\left(\frac{1}{R_{TX}} + \frac{1}{2R_{EQ}}\right) R_{RTH}.$$

V_{Roff} is equal to half of V_{Ron}. This adds hysterisis to the squelch circuit making it more reliable, especially in noisy environments.

Note that the receive threshold voltage is dependent on transmit output levels. It is therefore recommended to design the transmit section first, then choose the R_{RTH} resistor value.

TRANSMIT FUNCTION

Transmit Path

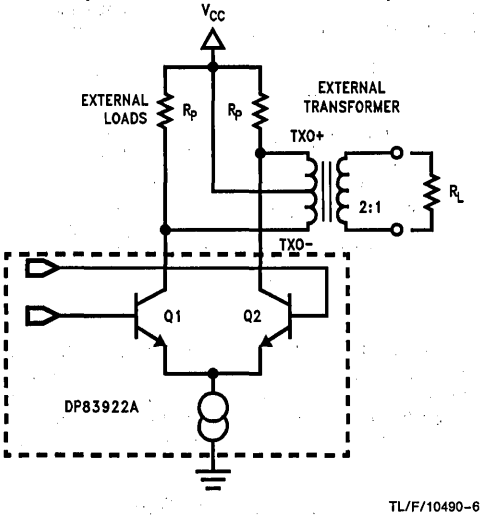
The transmitter has a differential input (TX±) and a differential open collector current driver (TXO±). From input to output the transmit path is fully differential and both TX+ and TX− are matched. This provides low signal skews and jitter. The differential input common mode voltage is established by the TPI and should not be altered by external circuitry. Either transformer or capacitive coupling of the TX± inputs will accomplish this. The general transmit waveform is shown below.

On the TX± inputs the transmitter squelch circuit rejects signals with pulse widths less than 20 ns typically, or with levels that are less than −175 mV. The transmitter turns off at the end of the packet if the signal stays higher than −175 mV for more than approximately 150 ns.

The transmitter differential outputs when coupled through a 2:1 pulse transformer can meet all the 10BASE-T output level specifications.

Functional Description (Continued)

Simplified Transmitter Differential Output



TL/F/10490–6

The transmitter differential outputs, when coupled through a 2:1 pulse transformer, can meet all the 10Base-T output levels specifications defined by IEEE.

As is shown in simplified form, the TXO± is a differential current mode output. This output integrates the equalization and wave shaping circuitry to pre-filter the output waveform. The driver level and equalization are externally programmable. Driver output current levels of the DP83922A are set by a built-in bandgap reference and two external resistors, R_{TX}, R_{EQ}. These two resistors set the output current for both the maximum and equalized portions of the output waveform. Controlled rise and fall times of the driver output circuits minimize the higher harmonic components, and therefore ease external filtering requirements. The rise and fall times are also matched to minimize jitter.

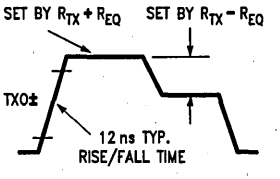
The relationship of the R_{TX} and R_{EQ} resistor values to the output current are:

$$I_{TX(out)} = \frac{59.76}{R_{TX}} + \frac{29.88}{R_{EQ}} \text{ mA}$$

$$I_{EQ(out)} = \frac{59.76}{R_{TX}} - \frac{29.88}{R_{EQ}} \text{ mA}$$

Note: Above resistor values are in kΩ.

TXO± Transmit Output Current Waveform



TL/F/10490–7

As can be seen above the transmit output will first go to an output current level set by the sum of the currents set by the R_{TX} and R_{EQ} resistor, and if the waveform is wider than 50 ns, the output current will drop to a value set by the difference between the currents set by R_{TX} and R_{EQ} resistors. Due to the tight IEEE output level specifications it is recommended to use 1% resistors for R_{TX} and R_{EQ}.

For each TXO output, the output current from the TXO± outputs will generate a voltage across R_P, the pulse transformer, and the twisted pair line impedance. The approximate equation for the single ended maximum output voltage output on the primary side (chip side) of the transformer is shown below:

(**Note:** The equations calculate the TPI's output voltage from the output pin to ground.)

$$V_{TXOUT} = \frac{I_{TX}}{2}\left(R_P \,||\, \frac{R'_L}{2}\right)$$

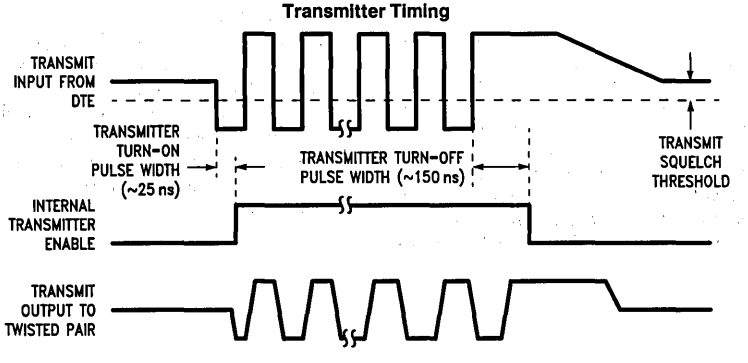
For the equalized single ended output voltage (the || operator is the parallel resistor combination):

$$V_{EQOUT} = \frac{I_{EQ}}{2}\left(R_P \,||\, \frac{R'_L}{2}\right)$$

Where R_P is the pullup resistor (typically 200Ω), and R'_L is the impedance of the twisted pair line as seen through the 2:1 transformer, $R'_L = 4\ R_L$ (typically 400Ω). For example if I_{TX} = 50 mA and I_{EQ} = 30 mA, then V_{TXOUT} peak is $\frac{50}{2}\left(100\Omega\right)$ = 2.5V and the V_{EQOUT} would be 1.5V.

An example of TXO outputs is shown on the following page. This diagram shows the typical TXO output waveform. This waveform is a 5V peak-to-peak signal centered at V_{CC}. The TXO+ and TXO− feed into the 2:1 transformer and filter, and result in the bottom waveform shown in the figure. The resultant output waveform is a ±2.5V differential typical signal per the IEEE standard.

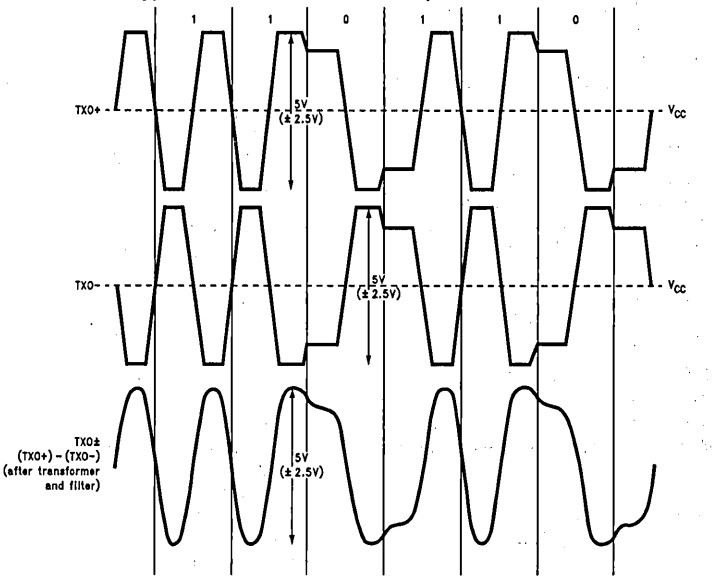
Transmitter Timing



TL/F/10490–8

Functional Description (Continued)

Typical Twisted Pair Transmit Output Waveforms



TL/F/10490–18

TOP: TXO+ Output voltage waveform measured relative to ground
MIDDLE: TXO− Output voltage waveform measured relative to ground
BOTTOM: Differential output voltage waveform on TXO+ to TXO−

COLLISION FUNCTION

The collision detection circuitry detects when the TX± inputs and the RXI± inputs are active simultaneously. This is done by logically ANDing the output of the transmit squelch and the receive squelch blocks. If both squelch indicate valid activity then the collision oscillator is enabled.

The collision circuits output a 10 MHz signal during a collision. The 10 MHz is generated by an internal oscillator. This oscillator is also used as a timebase for other internal functions.

HEARTBEAT FUNCTION

The collision circuits also generate the Heartbeat (SQE Test) signal at the end of every transmission to ensure that the collision circuitry is functioning. The burst of 10 MHz on the collision output occurs typically 1.1 μs after the transmission, and has a duration of about 1 μs. For repeaters this function can be disabled via the COL/HBE input.

JABBER TIMER FUNCTION

The Jabber Timer monitors the transmitter and inhibits the transmission if the Transmitter is active for longer than about 50 ms typical. It also enables the collision output for the fault duration. After the fault is removed, the Jabber Timer waits for about 500 ms typical (unjab time) before re-enabling the Transmitter. The transmit input must stay inactive during the unjab time.

LINK FUNCTION

The link integrity function consists of two parts, one part generates the link pulses, and the other detects the reception of link pulses. Both the transmission of the link pulse and then monitoring receptions of link pulses can be disabled by grounding the GDLNK pin.

Link Detector

The link detector circuitry is connected to the twisted pair receive circuits, and implements the IEEE 10BASE-T link state diagram. This block checks for the arrival of link pulses from the remote MAU, and if they arrive properly this logic keeps the transmit and receive functions enabled. If the link pulses fail to arrive then the link detector will disable the receive, transmit, collision, and heartbeat functions. When transmitting a packet the DTE is informed of the link failure by not receiving the loopback or the subsequent heartbeat signals from the TPI.

Link Generator

The link generator is a timer circuit that generates a link pulse by enabling the transmit output drivers. This 100 ns wide pulse will be generated if the transmitter is idle for 16 ms ±8 ms. The output of this circuit feeds into the TXO± driver circuit.

STATUS LED LOGIC

To ease the design of external LEDs that indicate to the end user various MAU status functions, the TPI includes 5 LED output drivers on-chip. These outputs are directly derived by the internal circuitry, and they indicate the following status information:

Reception: The RXLED output will go low whenever the receive (RXI±) squelch is disabled (receiver turned on).

Transmission: The TXLED output will go low (lighting the LED) whenever the AUI side transmit (TX±) squelch is disabled (transmitter turned on). The TXLED output will not go low during link pulse transmission or during a Jabber condition.

Functional Description (Continued)

Good Link: The GDLNK output will remain low while the Link state machine receives valid link pulses from the remote MAU. If the Link state logic does not properly receive link pulses, then the output will turn off the LED.

Collision: The HBE/COL output will turn on an LED if a collision is detected by the TPI (output goes low). This driver is directly driven by the output of the logical ANDing of the TX± and RXI± squelch.

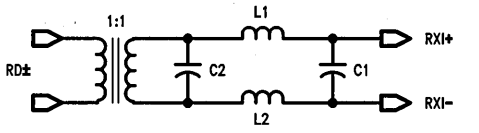
Jabber: The JBLED output is derived directly from the Jabber timer circuits. The output will go low (light the LED) whenever the TPI has determined that a Jabbering transmission is in progress, and will stay on until the transmit output is re-enabled.

All these LED outputs are driven directly by the internal circuits, and are not delayed or otherwise altered. This feature allows these outputs to be used logically if desired.

EXTERNAL RECEIVE AND TRANSMIT FILTER DESIGN CONSIDERATIONS

A requirement to interface to the twisted pair cabling is to provide adequate filtering on both the receive and transmit signal lines.

Three Pole Receive Butterworth Filter (100Ω)



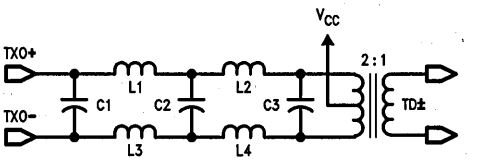
TL/F/10490–9

On the receive path, the 10BASE-T standard recommends a filter. A 3 pole Butterworth low pass filter yields acceptable high frequency rejection, for example a filter with the following characteristics may be used:

1. 3 dB Cut Off Frequency:	15 MHz
2. Insertion Loss (5 MHz–10 MHz)	≤ 1.0 dB
3. 30 MHz Attenuation:	17.5 dB min
4. I/O Impedance:	100Ω
5. Return Loss: (5 MHz–10 MHz)	≥20 dB

Additionally it may be desirable to add on the input to the pulse transformer a common mode choke to further filter out common mode noise.

Five Pole Transmit Butterworth Filter (400Ω)



TL/F/10490–10

On the transmit path there are numerous criteria for selection of the filter. To meet the general requirements a 5 pole Butterworth filter that can meet the following characteristics can be used:

1. 3 dB Cut Off Frequency	15 MHz
2. Insertion Loss: (5 MHz–10 MHz)	≤1.0 dB
3. 30 MHz Attenuation	27 dB Min
4. I/O Impedance	400Ω
5. Return Loss: (5 MHz–10 MHz) (Including Load Impedances)	≥20 dB

The Butterworth filter can be replaced by other filter types should the requirements demand it, but given the equalization-filtering the harmonic content of the TPI's transmit waveform, the filter requirements are reduced compared to pure digital implementations thus simplifying the design.

The transmit output is designed to drive an equivalent 400Ω impedance. This reduces the power/driver requirements of the transmitter, but also requires a 2:1 windings ratio on the transmit pulse transformer.

Integrated Filter Module Vendors

Several vendors are supplying filter, and transformer components that may be used with the DP83922A. The following companies provide products for use with the DP83922A.

Pulse Engineering
P.O. Box 12235
San Diego, California USA
92112
Phone: (619) 268-2400
Product: PE65423

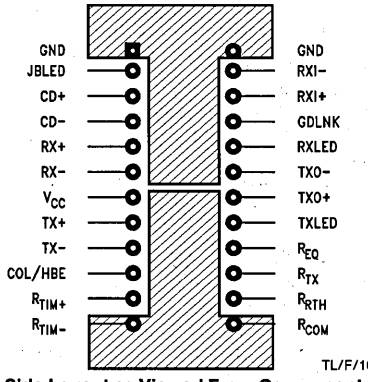
Belfuse
198 Van Vorst St.
Jersey City, NJ USA
07302
Phone: (201) 432-0463
Product: 0556-3899-00

Valor Electronics
6275 Nancy Ridge Dr.
San Diego, California USA
92121
Phone: (619) 458-1471
Product: PT3884

PCB LAYOUT CONSIDERATIONS

For heat dissipation purposes, it is recommended that a heat dissipation plane be added to the PCB layout of the TPI. The minimum required plane area should be 0.5 sq. in. This plane should be located on the solder side of the PCB, and can be most conveniently placed directly under the TPI package, between the pins as shown in the following figure. Alternately, the designer may add a heatsink to the top of the chip itself.

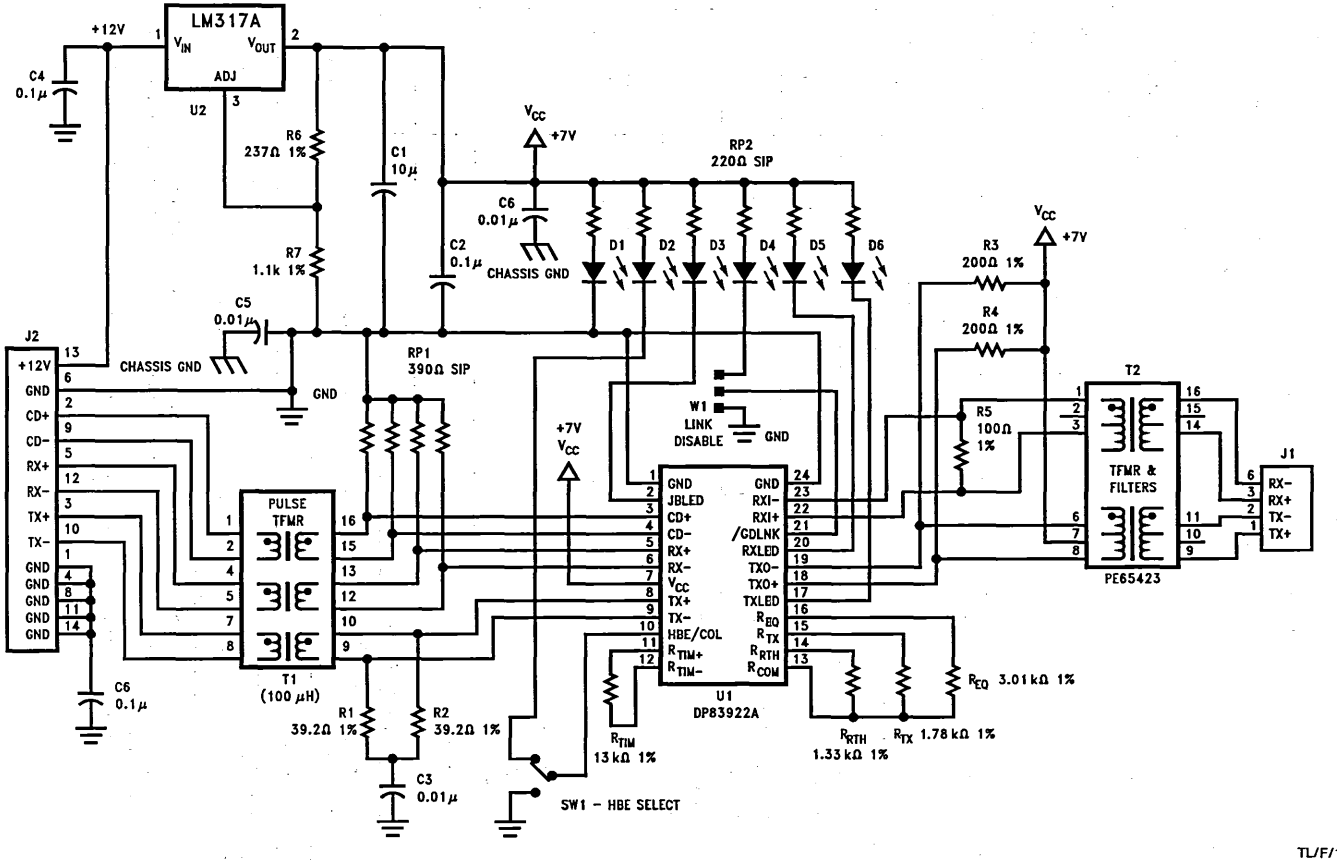
Ground Dissipation Trace



TL/F/10490–21

Solder Side Layout as Viewed From Component Side

Typical Medium Attachment Unit Application Using the DP83922A



TL/F/10490–11

Note: For T2, some integrated filter modules incldue R3, R4 and R5 inside the module. In such a case, those resistors should not be added externally. Please contact manufacturer for specifications details.

Absolute Maximum Ratings

(Notes 1 and 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +9.0V
DC Input Voltage (V_{IN})	−0.5V to +9V
Storage Temperature (T_{STG})	−65°C to +165°C
Package Power Dissipation (P_D)	3.5W
Lead Temperature (T_L) (Soldering, 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	6.65	7.35	V
Operating Temperature (T_A)	0	+70	°C
ESD Tolerance:	TBD		V
C_{ZAP} = 100 pF, R_{ZAP} = 1.5 kΩ (Note 3)			

PARAMETRICS DISCLAIMER

The current AC and DC specifications contained in this document are considered target design specifications and may not represent actual guaranteed tested timing parameters. This information represents simulated, as well as, limited sampled empirical "bench test" data. Guaranteed specifications will be provided after full device characterization.

Do not use these specifications for final production designs without directly contacting National Semiconductor.

Electrical Characteristics

V_{CC} = 7V ±5%, R_{TX} = 1.78 kΩ 0.1%, R_{EQ} = 3.01 kΩ 0.1%, R_{RTH} = 1.33k 0.1%, R_{TIM} = 13 kΩ 0.1%, T_A = 0°C − 70°C, unless otherwise specified (Note 4)

Symbol	Description	Conditions	Min	Max	Units
I_{CC}	Power Supply Current	RX± and CD± Open		200	mA
AUI					
V_{ROD1}	Differential Output Voltage (RX±, CD±)	Test *Figure 1*	±550	±1200	mV
V_{ROD2}	Differential Output Voltage (RX±, CD±)	Test *Figure 2*	±350		mV
V_{TS}	Transmitter Squelch Threshold (TX±)			−175	mV
TWISTED PAIR INTERFACE					
V_{Ron}	Receive Turn On Threshold (Note 5)		300	585	mV
r_{Roff}	Ratio of Receive Squelch Turn Off Threshold to Turn On Threshold $\left(\frac{V_{Roff}}{V_{Ron}}\right)$ (Note 5)		1.8	2.2	
V_{TOD}	Differential Output Voltage (TXO±) (Note 5)		±4.4	±5.6	V
RX_{IN}	Receiver Input Impedance		10		kΩ
LED OUTPUTS, HBE AND GDLNK INPUT					
V_{LOL}	LED Output Voltage	I_{OL} = 10 mA		2.4	V
V_{IL}	HBE, GDLNK Input Low Voltage			0.2	V
V_{IH}	HBE, GDLNK Input High Voltage		1.8		V

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified, all voltages are referenced to ground.

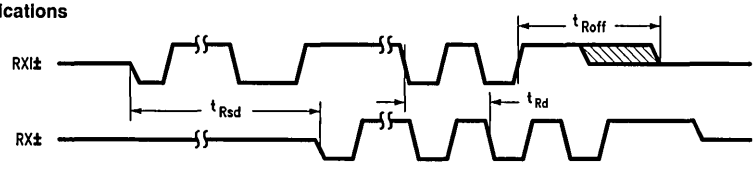
Note 3: Value based on test complying with NSC SOP 5-028 human body model ESD testing using the ETS-910 tester.

Note 4: These DC Electrical Characteristics are measured statically, and not under dynamic conditions. 0.1% resistors are used for test purposes. 1% resistors should be used in the end application.

Note 5: This specification is dependent on selection of R_{TX}, R_{EQ}, and R_{RTH} which have not been determined at this time. Typical values for these specifications can be derived from the design equations presented previously in this datasheet.

Switching Characteristics V_{CC} = 7V ±5%, T_A = 0°C to +70°C unless otherwise specified

Receiver Specifications

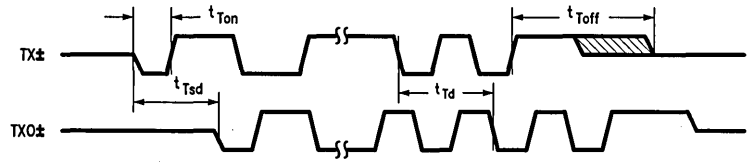


TL/F/10490–12

Symbol	Description	IEEE	Min	Max	Units
t_{Rsd}	Receiver Startup Delay (RXI± to RX±) (Note 6)	M1		800	ns
t_{Roff}	Receiver Turn Off Delay (RXI± to RX±)		130	250	ns
t_{Rd}	Receiver Propagation Delay (RXI± to RX±)	M1		200	ns
f_{RON}	Receiver Turn On Frequency (Squelch Disable)				MHz

Note 6: t_{Rsd} = (Propagation Delay) + (Invalid Bits) + (Bit Loss)

Transmitter Specifications

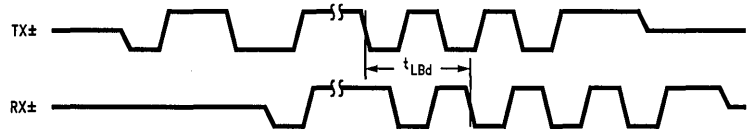


TL/F/10490–13

Symbol	Description	IEEE	Min	Max	Units
t_{Tsd}	Transmit Startup Delay (TX± to TXO±) (Note 7)	M2		500	ns
t_{Ton}	Transmit Turn On Pulse Width (TX± to TXO±)		5	40	ns
t_{Toff}	Transmit Turn Off Pulse Width (TX± to TXO±)		110	200	ns
t_{Td}	Transmitter Propagation Delay (TX± to TXO±)			200	ns

Note 7: t_{Tsd} = (Propagation Delay) + (Invalid Bits) + (Bit Loss)

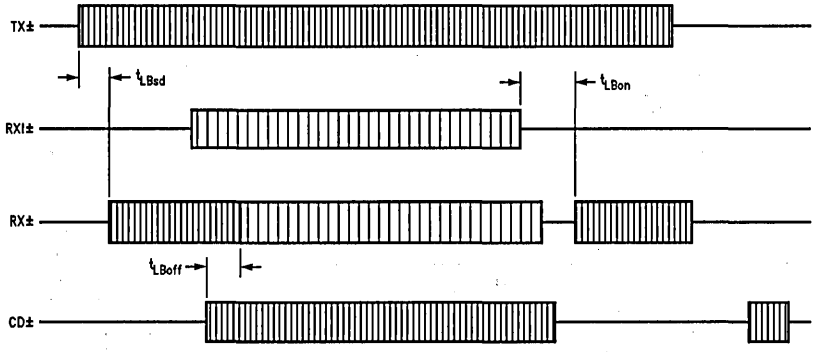
Loopback Specifications



TL/F/10490–14

Symbol	Description	IEEE	Min	Max	Units
t_{LBd}	Loopback Propagation Delay (TX± to RXI±)			100	ns

Switching Characteristics V_{CC} = 7V ±5%, T_A = 0°C to +70°C unless otherwise specified (Continued)

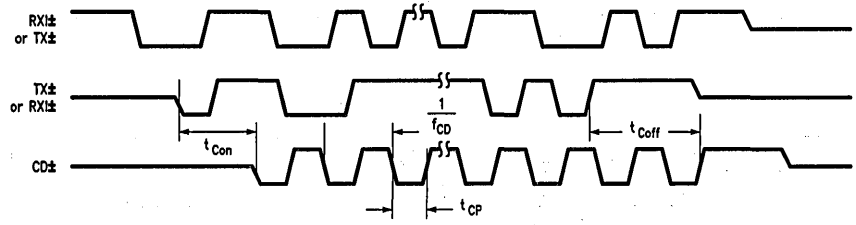


TL/F/10490–22

Symbol	Description	IEEE	Min	Max	Units
t_{LBsd}	Loopback Startup Delay (Start of TX± to start of RX±) (Note 8)	M9		700	ns
t_{LBoff}	Loopback Deassert (CD± to RX±)	M5		900	ns
t_{LBon}	Loopback Assert (TX± to RX±)	M6		900	ns

Note 8: t_{LBsd} = (Propagation Delay) + (Invalid Bits) + (Bit Loss)

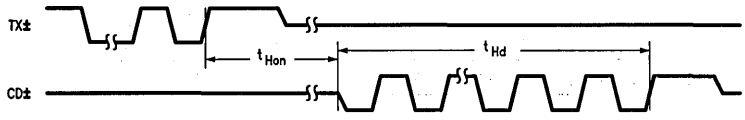
Collision Specifications



TL/F/10490–15

Symbol	Description	IEEE	Min	Max	Units
t_{Con}	Collision Turn On Delay	M3		900	ns
t_{Coff}	Collision Turn Off Delay	M4		900	ns
f_{CD}	Collision Frequency		8.5	12.5	MHz
t_{CP}	Collision Pulse Width		35	70	ns

Heartbeat Specifications

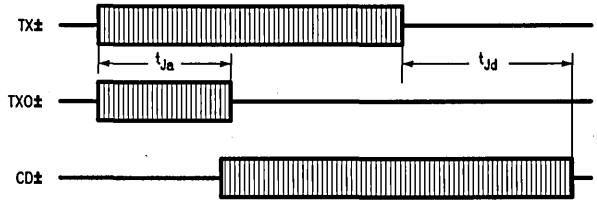


TL/F/10490–16

Symbol	Description	IEEE	Min	Max	Units
t_{Hon}	CD Heartbeat Delay (TX± to CD±)	M7	600	1600	ns
t_{Hd}	CD Heartbeat Duration (CD± to CD±)	M8	500	1500	ns

Switching Characteristics V_{CC} = 7V ±5%, T_A = 0°C to +70°C unless otherwise specified (Continued)

Jabber Specifications



TL/F/10490–17

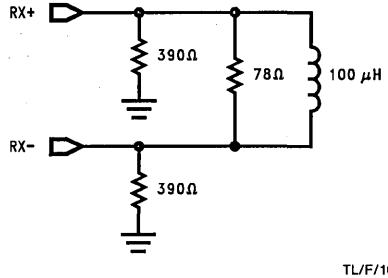
Symbol	Description	IEEE	Min	Max	Units
t_{Ja}	Jabber Activation Time (TX± to TXO± and CD±)		20	150	ms
t_{Jd}	Jabber Deactivation Time (TX± to TXO± and CD±)		0.25	0.75	S

Link Specifications

Symbol	Description	IEEE	Min	Max	Units
t_{LP}	Transmit Link Pulse Spacing		8	24	ms
t_{Lloss}	Received Link Pulse Loss Time		50	150	ms
t_{Lmax}	Received Link Maximum Time		25	150	ms
t_{Ltmin}	Received Link Minimum Time		2	7	ms
LC	Link Count		2	10	Pulses

Switching Specifications Test Loads

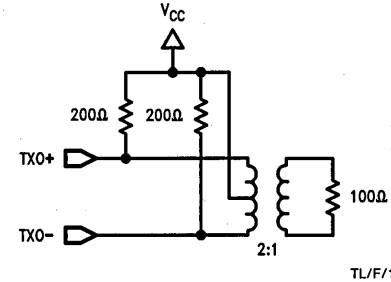
RX± and CD± Outputs Test Load



TL/F/10490–19

Test Figure 1

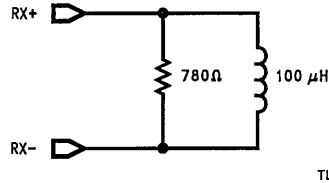
TXO± Outputs Test Load



TL/F/10490–20

Test Figure 3

RX± and CD± Output Test Load



TL/F/10490–23

Test Figure 2

10BASE-T Transceiver Design Using the DP83922

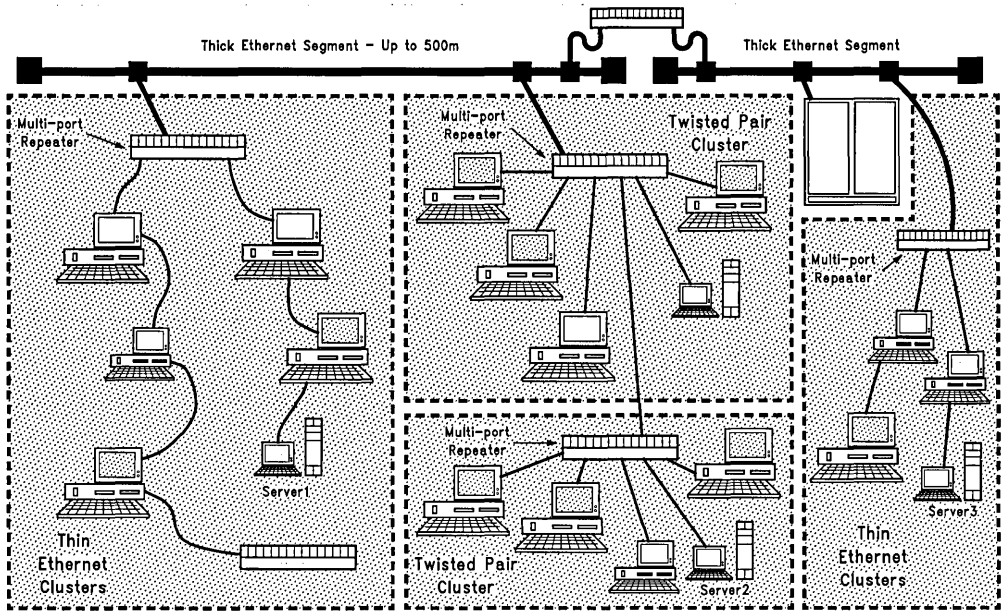
National Semiconductor
Application Note 743
Prasun K. Paul

INTRODUCTION

With the advent of the IEEE standard 802.3 type 10BASE-T, Ethernet users now may connect their terminals to Local Area Networks by using simple, inexpensive twisted-pair telephone cable. The IEEE 802.3 standard type 10BASE-T provides the electrical specifications and protocol to use twisted-pair wires as a media. The standard was designed so that twisted-pair Ethernet can co-exist with normal analog telephone traffic, RS232, RS422, ISDN and other office equipment. In addition this twisted-pair network can be interfaced with the already existing thick wire (10BASE5) and thin wire (10BASE2) Ethernet.

The 10BASE-T standard, as with the other 802.3 standards, is based on Carrier Sense Multiple Access with Collision Detection (CSMA/CD) media access method. 10BASE-T has a data rate of 10 Mbits/sec, baseband transmission and the physical medium of transmission is unshielded twisted-pair with a maximum 100m length per segment.

Table I compares the three Ethernet physical media (cable) standards, and *Figure 1* illustrates the basic connectivity of all three media. Coaxial medium in Ethernet is a bus topology in which all nodes tap onto the same cable and have equal access to the cable. In *Figure 1* Thick Ethernet is shown at the top of the figure, and Thin Ethernet is shown on either side. Thin and Thick Ethernet may be interconnected via a repeater as shown.



TL/F/11132–1

FIGURE 1. A Network with the Three Standard Types of Media

TABLE I. Comparison among Different Types of Ethernet Media

Parameter	10BASE5 (Ethernet, Thick Wire Ethernet)	10BASE2 (Thin Wire Ethernet)	10BASE-T (Twisted-Pair Ethernet)
Data Rate	10 Mbits/s, Baseband	10 Mbits/s, Baseband	10 Mbits/s, Baseband
Segment Length	500m	185m	100m
Type of Configuration	Bus (Multport)	Bus (Multiport)	Star (Point to Point)
Cable (Medium)	0.4 in. Diameter 50Ω Double Shielded, Same Wire to Transmit & Receive	0.2 in. Diameter 50Ω (RG58A/U) Single Shielded, Same Wire to Transmit & Receive	24 Gauge (0.5mm) Wire, Twisted-Pair, 100Ω Unshielded, Different Pairs to Transmit & Receive
Connector	Precision "TAP"	BNC	8-Pin Telephone Connector (RJ-45)
Transceiver Drop Cable/AUI Interface	0.39 in. Diameter Multi-Way Cable with 15 Pin D Connectors 50m Max Length	Same AUI Available for Backward and Forward Compatibility	Same AUI Available for Backward and Forward Compatibility
Nodes per Segment	100	30	2
Isolation on Medium Side	Not Needed	Not Needed	Needed

Twisted-pair Ethernet is organized in a Star topology. As seen in the central part of *Figure 1*, the twisted-pair network is a point-to-point link where each node connects up to a central hub. This hub in turn may connect up to another hub or to a network backbone such as Thick Ethernet. The twisted-pair hub is nothing more than a multiport repeater with a twisted-pair cable interface.

The major advantage of 10BASE-T is that the cabling used is very inexpensive twisted-pair wiring typically already installed. A second advantage is that 10BASE-T is topologically layed out the same as a phone system. This makes the installation and maintenance of the network easy. However, since unshielded twisted-pair is used, several technical issues must be addressed by the hardware manufacturer to ensure a reliable network. Fortunately, all these issues are addressed in the DP83922, Twisted-Pair Transceiver Interface (TPI), making the TPI a good choice for implementing a twisted-pair network physical interface.

Data Terminal Equipment (DTE) can be connected to the Ethernet network with twisted-pair medium using the standard Ethernet chip set with TPI instead of Coaxial Transceiver Interface (CTI). The TPI, DP83922 is compliant to the IEEE 802.3 standard type 10BASE-T. The DP83922 is ideal for stand-alone Medium Attachment Unit (MAU), since it is optimized for driving both the twisted-pair cable and the Attachment Unit Interface (AUI) cable. This MAU is a desirable product as it would be used for upgrading existing network installations to twisted-pair. The DP83922 may also be integrated into the DTE.

Table I highlights the differences between twisted-pair media, and other media in Ethernet standards.

OVERVIEW OF THE 10BASE-T STANDARD

The IEEE standard 802.3 type 10BASE-T defines functional, electrical and mechanical characteristics of a 10BASE-T MAU. For transmitting and receiving data, the TPI is to overcome the technical difficulties associated with unshielded twisted-pair media and its environment. Additionally, the standard addresses AUI interface on one side and a twisted-pair interface on the other side. The 10BASE-T MAU provides backward compatibility to the other standards by ensuring that the DTE sees no difference between the 10BASE-T MAU and any other types of Ethernet MAUs.

The standard also addresses the various technical concerns for driving 10 Mbit/sec data down the twisted-pair cables. The following section discusses the technical and functional issues of the 10BASE-T standard in order to give a background of the basic concerns indicated by the standard. This leads up to the next section, which discusses specific design issues for implementing a 10BASE-T MAU.

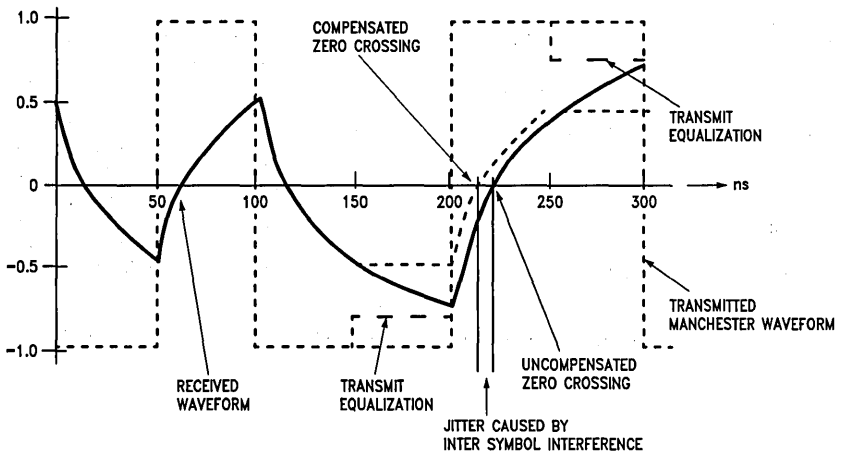
10BASE-T Electrical Characteristics

When transmitting and receiving 5 MHz and 10 MHz signals over unshielded twisted-pair cable, there are numerous effects that must be accounted for to ensure a reliable data link. The major issues addressed in the standard are described below in brief.

1. **Signal Jitter:** Manchester encoded signal gets distorted as it traverses the twisted-pair cable and results in signal jitter. When the 10 MHz and 5 MHz signal is passed through the twisted-pair, we should get signal pulses of 50 ns and 100 ns respectively. For the reasons described below the pulse widths vary due to signal distortion which results in jitter. It is important that the total jitter introduced by the cable stay within certain limits. These limits have been previously defined in other 802.3 standards and the 10BASE-T standard maintains these limits. Main causes of jitter are:
   a. INTER-SYMBOL INTERFERENCE (ISI): Depending on the data pattern, the Manchester encoded signal may appear as either 5 MHz or 10 MHz signals. "010101" data patterns cause a 5 MHz signal. Consecutive 1s or 0s cause a 10 MHz signal. A twisted-pair cable shows the characteristics of a low pass filter and group delay for 5 MHz and 10 MHz signals through the twisted-pair media is different. So in a random signal of frequencies of 5 MHz and 10 MHz, the 100 ns and 50 ns pulse widths change causing signal jitter. As shown in *Figure 2*, the effect of the cable on a signal can be described by the behavior of a resistor-capacitor circuit. A transmitted square wave through this is observed as an exponential signal at the end of the cable. In a 10 MHz pattern the charge/discharge curves cross time axis with a certain delay. But a 5 MHz signal has more time to charge/discharge and as a result crosses the time axis later than the 10 MHz signal. This difference in delay causes ISI which causes jitter.

      The difference in charged levels for 5 MHz and 10 MHz is decreased by discarding some energy from transmitted 5 MHz signals. As in *Figure 2* the amplitude of the 2nd 50 ns of 100 ns pulses is reduced by a certain amount. This helps to maintain equal delay in time axis crossing of the 5 MHz and 10 MHz signals. This compensation of transmitted signal is called equalization or pre-emphasis. The 10BASE-T standard defines that equalization be performed on the transmitted waveform. (It could also have been defined to be done to the received waveform as post equalization, but was not.)
   b. SOURCE AND MEDIUM IMPEDANCE VARIATION: The twisted-pair medium can vary from 85Ω to 111Ω for a frequency range of 5 MHz to 10 MHz. For any mismatch due to the impedance variation of the twisted-pair and termination causes reflection of the signals which introduces jitter. This variation is taken care of by return loss specification.
   c. IMPULSE NOISE: In the twisted-pair environment the sources of impulse noise are telephone ringing, dialing signals and other office equipment. This noise causes jitter.
   d. NEAR END CROSS TALK: The cross talk coupling between any two pairs of cable is affected by the cable size and the geometry of the cables in a cable binder group. Maximum allowed Near End Cross Talk loss is defined in the specification.

      A jitter budget of ±8.0 ns is allocated for noise.



TL/F/11132-2

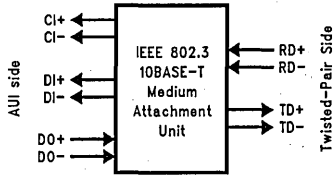
FIGURE 2. Equalization for Inter Symbol Interference

2. **Faulty Connection:** The integrity of the cable link is a significant feature of the 10BASE-T standard. If the twisted-pair link attached to the receive pair is disconnected or the MAU on the opposite end is not transmitting link pulses or data, the MAU should be aware of this fault (and should inform the user). This function is performed by Link Integrity Test Function which monitors the cable for activity and notifies the DTE if the cable shows no activity for a predetermined period of time.
3. **Bit Error Rate:** Since the twisted-pair cable is unshielded, the signal to noise ratio on the cable is a concern. Noise on the segment may be caused by noise sources listed above in number 1. This is addressed by specifying transmit amplitude and specifying a filter requirement on the transceiver's receive side.
4. **False Receiver Turn On:** Because of the coexistence of unshielded twisted-pair with other equipment, the receiver might get turned on for any impulse from the noisy environment. Receiver squelch level setting and other receiver rejection requirements take care of false turning on of the receiver.
5. **RFI Emission:** Agencies of various governments, such as FCC (Federal Communications Commission) of the US limits transmission of high frequency components above specified levels which may cause electromagnetic interference to other equipment. This is partially taken care by recommending a transmit filter, level of harmonic components on the test load and common mode voltage. The standard does not guarantee that the recommended filter will meet FCC requirements and mentions that it is the responsibility of the implementers. Other than transmission circuitry, noise generated by any circuitry in the MAU may also affect RFI.

The above considerations lead to the definition of a 10BASE-T Medium Attachment Unit that is described in the following section.

10BASE-T MAU Functions

The IEEE standard defines the basic functions that the 10BASE-T MAU must perform. The standard defines the MAU as an independent function. So it assumes an AUI port to interface the DTE and the twisted-pair interface as a medium. The signal designators for the MAU's inputs are shown in *Figure 3* and these signal names are used to describe the functional capabilities as listed below. RD±, TD± circuitry are on the twisted-pair side and CI±, DI±, DO± are on the AUI side of the MAU. MAU receives data on RD± circuitry and transmits data by TD± circuitry. CI± circuitry is to give collision and SQE signals to the DTE, DO± is for transmitting data from the AUI. DI± circuit is to send the data received on RD± and to loop back the transmitting data to AUI.



TL/F/11132–3

FIGURE 3. Signal Designations for 802.3 Type 10BASE-T MAU

1. **Transmit Function:** The MAU transfers Manchester encoded data from the DTE or repeater via the AUI interface to the twisted-pair medium via the Medium Dependent Interface (MDI) connector which is basically a telephone jack on the MAU and a plug connector on the twisted-pair link segment. The transmitter must have the following characteristics:
   a. At the start of a packet transmission, no more than 2 bits received on the DO± may not be transmitted to the TD± circuit. The first transmitted bit on the TD± circuit may be an invalid bit i.e., the first bit may violate amplitude and jitter specification. Maximum steady state propagation delay must not exceed 2 bit times.
   b. The jitter added to the signal received on the DO± circuit by the combination of the MAU transmitter and twisted-pair model terminated with a 100Ω shall be no more than ±3.5 ns (noise jitter is not included here). Also if the TD± circuit is directly driving a 100Ω resistive load, the MAU shall add no more than ±8 ns of jitter to the signal received on the DO± circuit.
   c. The transmitter terminated directly with a 100Ω resistive load on the TD± circuit shall produce a 2.2V to 2.8V peak differential voltage for all data sequences. When the DO± circuit is driven by an all ones Manchester signal, all harmonics measured on TD± circuit must be less than 27 dB below the 10 MHz fundamental.
   d. The transmit signal is equalized as shown in *Figure 2* to reduce the ISI jitter induced by the twisted-pair medium. Narrow data pulses (50 ns) are driven unaltered, but wide pulses (100 ns) are driven at full amplitude for first 50 ns and then the amplitude is reduced by about 40% during the 2nd 50 ns.
   e. At the end of a transmitted packet TP_IDL signal is pasted. TP_IDL signal is followed by a silence period of 16 ms ±8 ms and a Link Test Pulse repetitively. During the silence period the differential voltage on TD± circuit shall not exceed ±50 mV peak.
2. **Receive Function:** The MAU transfers Manchester encoded data from RD± circuit of the MDI to DI± circuit of the AUI side.
   a. According to the standard, at the start of a packet reception at the RD± circuit no more than 5 bits received on the RD± circuit may not be transmitted onto the DI± circuit. The first bit of the received data on DI± circuit may be invalid. In addition the propagation delay of the received data must be less than 2 bit times.
   b. The receiver must detect properly all the signals of amplitude 585 mV to 3.1V peak differential and having a jitter up to ±13.5 ns for 5 MHz and 10 MHz signals. This ±13.5 ns includes jitter caused by an encoder, AUI cable and MAU transmitter, the twisted-pair and noise.
   c. The standard states that the MAU receiver shall add no more than ±1.5 ns of jitter.
   d. The receiver, while in idle state, shall have the following rejection capabilities: 1) All signals which is measured at the output of the following test filter would produce a peak magnitude less than 300 mV.

The test filter is a 3-pole low-pass Butterworth with a 3 dB cut off at 15 MHz and has an insertion loss of less than or equal to 1 dB, 2) All continuous sinusoidal signals of amplitude less than 6.2 V_{p-p} and frequency less than 2 MHz, 3) All sine waves of single cycle duration, starting with phase 0 or 180 degrees and of amplitude less than 6.2 V_{p-p} where the frequency is between 2 MHz and 15 MHz. For a period of 4 BT before and after this single cycle, the signal shall be less than 300 mV when measured through the test filter specified above in (1). A smart squelch on the receive path can distinguish between noise and valid received data. A simple squelch circuit is only voltage level dependent. A smart squelch not only looks at the amplitude of the incoming signal, but also checks the times staying above the threshold level and the time for crossing opposite thresholds. (See DP83922 Datasheet for more details on the operation of the squelch.)

3. **Loopback Function:** In the case of 802.3 coax type MAUs, the transmit and receive lines are connected together at the cable, and this causes the transceiver to inherently receive any data that is transmitted by the same transceiver. The 10BASE5 and 10BASE2 MAUs loop back the transmitted signal to the AUI side. In 10BASE-T since the data is transmitted on a different pair of wires from the data received, the transceiver itself must loop back the data to maintain compatibility to the other 802.3 MAUs. The loopback function has been defined as follows:

   When the MAU is transmitting on the TD± circuit and is not receiving signals on the RD± circuit, the MAU shall transmit on the DI± circuit the signals received on the DO± circuit in order to provide loopback of the transmitted signal. At the start of packet transmission on the TD± circuit, no more than 5 bits of information may be received from the DO± circuit and not transmitted to the DI± circuit. It is permissible for the first bit sent on the DI± circuit to be an invalid data. All successive bits can have a jitter no more than ±13.5 ns plus ±1.5 ns and the amplitude should be valid. Steady state propagation delay should not exceed 1 bit time.

4. **Collision Function:** According to the IEEE standard 802.3 type 10BASE-T, the MAU does not support receive mode collision detection, but it does support transmit mode collision detection. In receive mode, collision is detected with certainty when any other two or more stations transmit at the same time. But in transmit mode, collision is detected with certainty only if the station itself is transmitting and one or more other stations are transmitting at the same time. That is, only simultaneous presence of data on the TD± and RD± circuit will cause collision. A 10 MHz signal is put on CI± circuits to indicate a collision to the DTE or repeater.

5. **Signal Quality Error Message (SQE) Test Function (Heartbeat):** When the 10BASE-T transceiver is connected to a DTE, the collision circuit must generate the Heartbeat (SQE Test) signal at the end of every transmission to ensure that the collision circuitry is functioning properly. This is identical to the 10BASE5 or 10BASE2 MAUs. A burst of 10 MHz signal on the CI± circuits should occur at 0.6 μs to 1.6 μs after the transmission and has a duration of 10 ±5 bit times. This SQE signal is not to be sent while the MAU is in any of the Link Test Fail states. When the transceiver is connected to a repeater this function should be disabled.

6. **Jabber Timer Function:** The jabber timer monitors the transmitter and inhibits the transmission if the transmitter is active for longer than 20 ms to 150 ms. It also enables the collision output for the duration of the fault and unjab time. After the fault is removed, the jabber timer waits for about 250 ms to 750 ms (unjab time) before re-enabling the transmitter. The transmit input on the DO± circuit must stay idle during the unjab time.

7. **Link Integrity Test:** This test checks to see if the remote MAU is properly connected to the twisted-pair media and if the remote MAU is transmitting Link Pulses or data. If it is not connected, the MAU disables the transmitter, receiver, loopback, collision and SQE test signal. At this state the MAU transmits link pulses and watches for link pulses or data. The 10BASE-T standard specifies a particular state diagram (*Figure 14-6* of the IEEE standard 802.3) for this function. The general description of the link operation is as follows:

   At power on the MAU may stay in Link Test Pass state or Link Test Fail Reset state. The MAU enters the Link Test Fail state if it does not receive data or link signals for 50 ms to 150 ms (link_loss_time). The MAU will exit from Link Test Fail state if it counts 2 to 10 link pulses. Link pulses with a gap of less than 'link_test_min' time (2 ms to 7 ms) are treated as noise and the link count is reset. Also if the consecutive link pulses are more than 25 ms to 150 ms apart, the link counter will be reset. If the MAU receives data, the MAU will go to Link Fail Extend state. In this state if the MAU does not get any data on RD± and DO± circuits the MAU will go to the Link Test Pass state.

8. **Link Generator:** At the end of data transmission, a signal called TP_IDL is generated which is a positive pulse about 250 ns to 450 ns wide. After TP_DL, as long as the MAU is not transmitting anything, Link Test Pulses of about 100 ns wide will be generated and transmitted at an interval of 16 ms ±8 ms.

Mechanical Characteristics

MAU TO AUI: If the MAU is a separate stand alone transceiver, it should be provided with a 15-pin male D-connector according to AUI standard. The MAU may be plugged directly to a DTE or may be connected by one or more cable segments whose total length is less than or equal to 50m. All the pins connected to shields i.e., pins 1, 4, 8, 11, and 14 are capacitively coupled to the power ground of the MAU. The shell (Protective ground) shall be plated with conductive material to ensure the integrity of the cable shield to chassis current path.

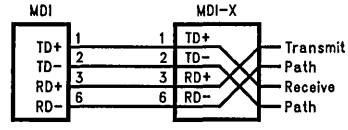
MAU to Twisted-Pair: This connection is defined as the Medium Depended Interface (MDI). A regular 8-pin telephone connector (ISO 8877) is used for this purpose. The standard specifies the receive pair (RD±) to be pins 3 and 6 and the transmit pair (TD±) to be pins 1 and 2. Table II shows the assignment of signals to connector contacts.

TABLE II. Telephone Jack (ISO 8877) Pin Assignments

Pin	MDI Signal
1	Transmit Data+ (TD+)
2	Transmit Data− (TD−)
3	Receive Data+ (RD+)
4	Not Used
5	Not Used
6	Receive Data− (RD−)
7	Not Used
8	Not Used

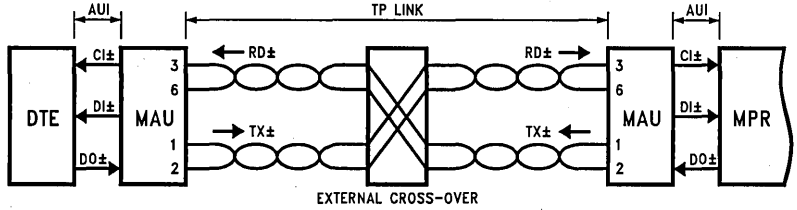
The transmit pins of a MAU have to be connected to the receive pins of the MAU on the other side of the segment and vise versa. This is done by crossover function. The crossover function can be internal or external to the MAU. A MAU which does not implement internal crossover function, the signal names refer to their internal circuits. As shown in *Figure 4*, MDIs of the MAUs do not have crossover. So the twisted-pair wires must implement the crossover function for network connectivity.

For a MAU which does implement the internal crossover function, the signal names refer to the remote MAU of the twisted-pair link. In this case pins 1 and 2 (TD±) are the transmit data of the remote MAU and internally these are connected to the receive data line. A MAU which implements the crossover internally must have a special designation. Generally repeaters are recommended to implement internal crossover function and the MDI should be marked "X" as in *Figure 5*. If both the MAUs have internal crossover, another external crossover is required in twisted-pair for proper connection.



TL/F/11132–5

FIGURE 5. Twisted-Pair Connection with Internal Crossover (Marked MDI-X)



TL/F/11132–4

FIGURE 4. Twisted-Pair Link with External Cable Crossover

IMPLEMENTING A 10BASE-T MEDIUM ATTACHMENT UNIT WITH THE DP83922

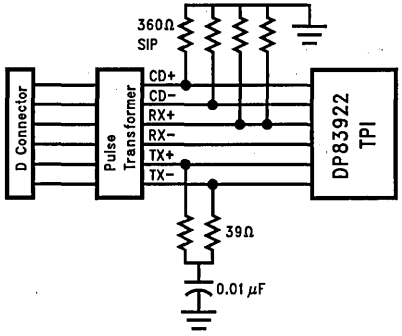
The Twisted-Pair Transceiver Interface (DP83922) provides all the MAU (transceiver) functions except signal isolation (both on AUI and twisted-pair side), power isolation and filtering of the receive and transmit signal on the twisted-pair side. The AUI side of the TPI is almost the same as the CTI (Coaxial Transceiver Interface), but the medium side interface is unique. A typical MAU block diagram is shown in *Figure 6.*

Connection to AUI Cable

The differential input common mode voltage is established by the MAU and should not be altered by external circuitry. Either transformer or capacitive coupling will accomplish this. Signal isolation can be provided by a set of three pulse transformers that come in single Dual-in-line packages. These are available from transformer vendors such as Pulse Engineering (PE64103) and Valor (LT1101).

The differential transmit pair from the DTE should be terminated with a 78Ω differential resistive load. By splitting the termination resistor into two equal values and capacitively AC coupling the center node to ground (0.01 μF capacitor), the common mode impedance is reduced to about 20Ω which helps to attenuate common mode noise transients.

To drive the 78Ω differential line with sufficient voltage swing, the TPI's receive and collision drivers need external 360Ω resistors to ground. This differs from the CTI's (DP8392) requirement of 510Ω resistors.



TL/F/11132–7

FIGURE 7. AUI Interface Design

Collision Function and Timing Resistor

Collision is determined internally in the TPI by logically ANDing the output of the transmit squelch and receive squelch blocks. If both the squelches indicate valid activity, then the collision oscillator is enabled.

Collision circuit output is a 10 MHz signal. This 10 MHz signal is generated by internal oscillator whose frequency is set by an external resistor (R_{TIM}, 13K 1%) between pin 11 and 12 of DP83922. This oscillator is used as a time base for other internal functions such as the receive squelch timer, link function and jabber.

Status LEDs

DP83922 provides 5 LED driver pins to indicate the status of the MAU. They are:

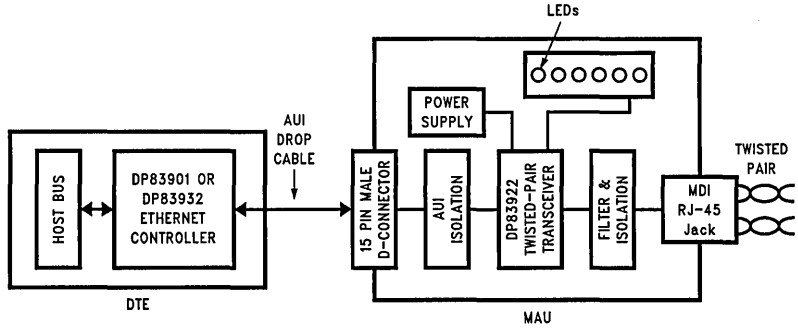
Collision: Simultaneous presence of data both on RD± and DO± circuit causes a collision. At the time of collision the HBE/COL pin goes low causing the connected LED to glow.

Jabber: This pin is active low. The LED connected to this pin continues glowing as long as the transmitter is disabled for jabber activity.

Gook Link: As long as the MAU is in Link Pass state i.e., accepting data or receiving link pulses at the recommended intervals this pin stays low and the connected LED stays on. Again if the MAU does not receive data or link pulses properly this pin will go high and the LED will be turned off indicating a bad link. Again if the MAU is in Link Fail state but receiving link pulses or data, the LED will be flashing.

Reception: The RXLED connected to pin 20 will be turned on whenever the receive squelch is off. The LED stays on while the MAU is receiving data. But it does not glow while it is receiving link pulses.

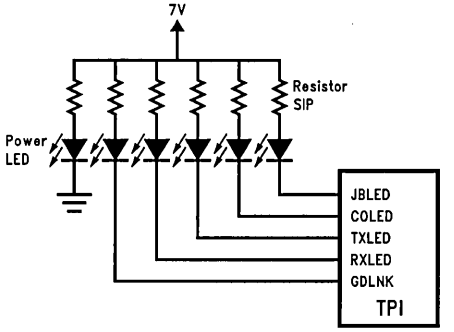
Transmission: At the time of transmission, the transmit squelch stays off and the LED connected to pin 17 (TXLED) glows. At the time of jabber condition and while the TPI is transmitting link pulses this LED does not glow.



TL/F/11132–6

FIGURE 6. DP83922 Based MAU Block Diagram

During normal operation the LINK LED will glow and will go off in Link Fail state. RX, TX, JAB, COL LEDs are on when any of these events occur. The duration that these LEDs stay on is according to the real time of a reception, transmission, jabber and collision event. To extend the duration of the LED on time to enhance visible display of the event, one-shots can be used to stretch the output low time a little longer.



TL/F/11132–8

FIGURE 8. LED Options of TPI

When the LED pins are active low, the voltages of JBLED, /GDLNK, COL are 2.4V (max), the TXLED is 1.7 (max), and RXLED is 0.77V (max).

Calculations for current limiting resistors: Minimum current required for each LED is 10 mA. Considering the voltage across LEDs to be 2.3V, the resistor value for /GDLNK etc. pins is:

$$R_{LED} = \frac{7V - 2.3V - 2.4V}{0.01A} = 230\Omega.$$

In *Figure 22* at the end of this note a SIP of 220Ω is shown. For TXLED, RXLED and power LED different resistors can be used.

In addition, an LED can be used to indicate the presence of 7V power supply.

If the LED options are not desired, the concerned pins should stay floating, i.e., connected to nothing, unless some jumper options are to be invoked as described below.

Possible Jumper or Switch Options

Heartbeat: The collision circuit also generates the heartbeat (SQE test) signal at the end of every transmission to ensure the controller that the collision circuitry is functioning properly. A burst of 10 MHz signal occurs at 0.6 μs to 1.6 μs after the end of every packet with a duration of about 10 ±5 bit times. This SQE signal will not be sent while the MAU is in any of the Link Fail states. For repeaters, this function can be disabled by grounding pin 10 (HBE/COL). If the HBE/COL is grounded, the collision LED should not be used.

Link Function: As long as the MAU is in Link Pass state, i.e., accepting data or receiving Link Test pulses at recommended intervals, this GDLNK pin 21 stays low causing the connected LED to glow. Again if the MAU does not receive data or link pulses properly, this pin will go high and the LED will be turned off indicating a bad link. Both the transmission of Link Pulses and then monitoring reception of link pulses can be disabled by grounding the GDLNK pin. In link disable mode DP83922 will not generate link pulses and the MAU is always in Link Pass state i.e., the TPI will not disable Transmit, Loopback or Receive functions irrespective of reception of link pulses or data on the RD± circuits. At link disable mode if an LED is connected to this pin it will always be on.

Transmit Design of Twisted-Pair MAU

To get a properly designed transmit output, the transmit circuit should meet a number of parameters. These are:

1. Output amplitude across a 100Ω load at the MAU transmitter (TD±) has to be between 2.2V to 2.8V peak.
2. Output impedance of the transmitter at 5 MHz to 10 MHz expressed in terms of return loss has to be less than 15 dB.
3. The transmitted signal at the end of 100m twisted-pair cable terminated with a 100Ω load has to be within the Voltage Template given in the standard.
4. All the harmonics of the transmitted waveform must be −27 dB below the fundamental 10 MHz.

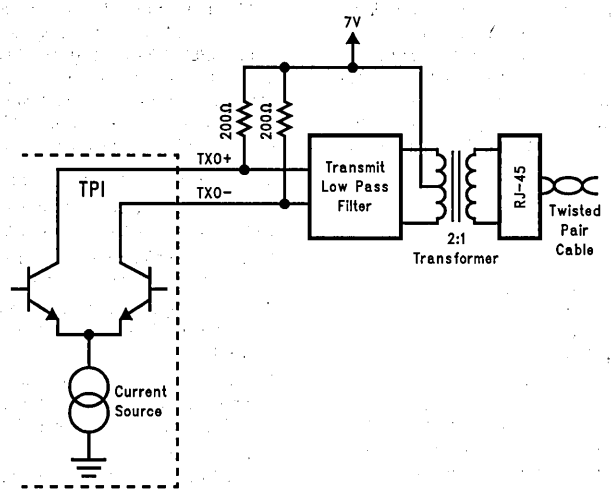
All these items will be addressed in the following sections.

The differential transmit output drivers are current sources. These open collector outputs are connected to the twisted-pair cable through a number of elements. First each of the TXO± output circuits are connected to pullup resistors. Then they are connected to a transmit filter which in turn is connected to an isolation transformer and finally to the cable connector (RJ-45) to twisted-pair. This is illustrated in *Figure 9*. The standard requires isolation on the twisted-pair media side.

A 2:1 isolation transformer is used in the transmit path of the DP83922. The output of the TPI must match the 100Ω impedance of the twisted-pair cable, and since a 2:1 transformer converts impedance by a factor of 2^2, a 400Ω impedance across the TXO+ and TXO− lines is needed. The two 200Ω resistors as shown in *Figure 9* provide the proper impedance.

It should also be noted that the transmit filter in *Figure 9* is a 400Ω filter. This filter may optionally be located on the cable side of the transformer and in this case a 100Ω filter can be used.

Additionally, a common mode choke can be added between the transformer and RJ-45 connector. The common mode choke helps in common mode noise rejection and reducing RFI emissions. In the common mode choke, the direction of winding and connection of the signal lines is very important. The way it works: the coils in the common mode choke are wound in a way that the electromagnetic force for the current in one coil opposes the electromagnetic force of the current in the same direction through the other coil. So, if a common noise is induced on the twisted-pair, it will be cancelled by opposite electromagnetic forces and the noise will be filtered out by the choke. Part numbers of some recommended common mode chokes are: PT3868 (Valor Electronics) and PE64681 (Pulse Engineering). These are 8-pin packages and they have separate channels for receive and transmit path. The common mode choke is not shown in the MAU diagram.



TL/F/11132–9

FIGURE 9. Connection Diagram of TPI Transmit Path

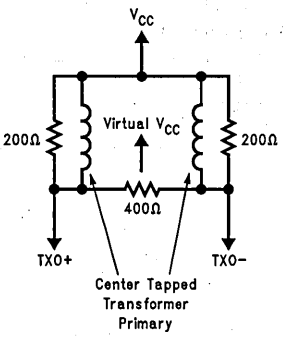
Transmit Output Theory of Operation

It is useful to understand how the connection of the transmit output actually works. Doing so will provide a better understanding of how the external resistors set the transmit output waveform's amplitude and shape.

Transmit output circuits of DP83922 are current sources. Output current levels of the DP83922 are set by a built-in band gap reference and two external resistors R_{TX}, R_{EQ}. These two resistors set the output current for both the maximum and equalized portions of the output waveform. Controlled rise and fall times (about 12 ns) of the driver output circuits minimize the higher harmonic components.

The transmit signal is equalized to reduce the jitter induced by the twisted-pair medium. In this mechanism narrow pulses (50 ns) are driven normally. But wide pulses (100 ns) are driven at full amplitude, V_O for first 50 ns and then the amplitude is reduced by approximately 40% during the 2nd 50 ns to about 0.60 V_O.

To understand the operation, the external components can be shown in an equivalent circuit. *Figure 10* shows this circuit. (Note: we are using the 10BASE-T case of driving a 100Ω cable, but the analysis applies to other impedances except the values of the 200Ω resistors are different.) The two 200Ω resistors in *Figure 10* are the same as shown in *Figure 9*. The two inductors are the two halves of the center tapped transformer with it's center tied to V_{CC}. The center tap connection to V_{CC} of the transformer helps to set the common mode voltage V_{CM} to V_{CC} i.e., $V_{CM} = V_{CC}$. The 400Ω resistor is the 100Ω load that the TPI sees. Since the impedance is seen through a 2:1 transformer, the transformed impedance is 400Ω.

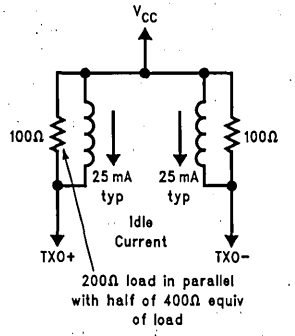


TL/F/11132–10

FIGURE 10. Transmit Circuit with 400Ω Reflected Impedance

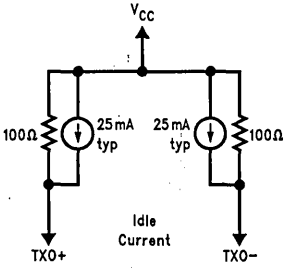
The reflected 400Ω can be modeled as two 200Ω resistors with the connection between them being a virtual V_{CC} (because the common mode voltage is set at V_{CC}). These two resistors are in parallel to the 200Ω pullup resistors. Thus the overall impedance on each of the TXO± lines becomes 100Ω as shown in *Figure 11*.

The inductance represents the center tapped transformer. At idle time when the transmitter is not transmitting, an idle current will be drawn by each of the TXO± circuits. This current will pass through the inductance of the transformer only. The time constant due to the inductance of the transformer and pullup resistor is much higher compared to the pulses for 5 MHz and 10 MHz signals. Therefore, this current through the inductances can be represented by equivalent current sources as in *Figure 11(b)*.



TL/F/11132–11

(a) Equivalent Circuit Showing Loads



TL/F/11132–12

(b) Equivalent Circuit Converting Transformer to a Current Source

FIGURE 11. Transmitter at Idle

For the explanation of the operation of the transmit output, we can assume that the R_{TX} and R_{EQ} resistors have been set to give a 50 mA peak current through TXO+ or TXO−, and a 10 mA equalization current. These current values are arbitrarily chosen to demonstrate transmitter operation.

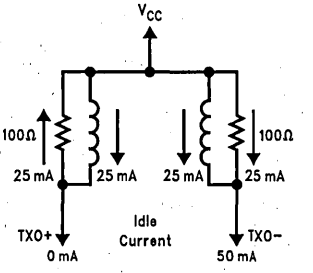
At idle each of the TXO± lines draw equal current, in this case 25 mA will flow through each of the TXO± lines. At this time all the currents will be drawn by the current sources. So, the voltage on each of the TXO± circuits is set to V_{CC} and the differential voltage across TXO± circuits becomes zero.

Figures 12 and *13* represent the positive voltage swings of the transmitted signals. When a 10 MHz signal or the first 50 ns of 5 MHz signal is being transmitted, TXO+ draws no current, and 50 mA current is drawn by TXO− circuit. As shown in *Figure 12*, 25 mA current will discharge through the 100Ω resistor on TXO+ branch. Out of the 50 mA current through TXO− circuit, the current through the inductor is 25 mA, while the remaining 25 mA passes through the 100Ω resistor. Overall 25 mA current will flow in the same direction from 100Ω on TXO+ to 100Ω on TXO− producing a differential voltage of 5V peak across TXO± circuits. The calculations for the differential voltage is shown below:

$$V_{TXO-} = 7V - 100 \times 25 \times 10^{-3} = 4.5V$$

$$V_{TXO+} = 7V + 100 \times 25 \times 10^{-3} = 9.5V$$

$$V_{DIFF} = 9.5V - 4.5V = 5V$$



TL/F/11132–13

FIGURE 12. Current for 50 ns Cycles

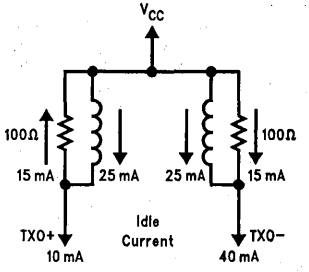
To compensate for jitter due to ISI, a pre-emphasized 5 MHz signal is transmitted. The amplitude of signal at second 50 ns of 100 ns pulses of 5 MHz signals is reduced. At this time, approximately 10 mA current is drawn by TXO+ and rest 15 mA will pass through the 100Ω resistor. Current through TXO+ (10 mA in this case) will be subtracted from TXO−, resulting in a 40 mA current being drawn by TXO− circuit as in *Figure 13.* Over all, 15 mA current will flow from the 100Ω on TXO+ to 100Ω on TXO− producing 3V peak differentially across the TXO± circuits. The calculations for equalized portion of the 5 MHz signal is shown below:

$$V_{TXO-} = 7V - 100 \times 15 \times 10^{-3} = 5.5V$$

$$V_{TXO+} = 7V + 100 \times 15 \times 10^{-3} = 8.5V$$

$$V_{DIFF} = 8.5V - 5.5V = 3V$$

Notice that the 25 mA current through the transformer is considered unchanged.



TL/F/11132–14

FIGURE 13. Current for Last 50 ns of 100 ns Pulses

For negative pulses, the same amount of current will be drawn by opposite branches of TXO± circuits. That will produce differentially −5V and −3V peak on the TXO± circuits. Therefore, at the time of transmitting data 5V peak (10V peak to peak) voltage will be observed on the TXO± circuits and the equalized portion will be ±3V peak (6V peak to peak). When these waveforms pass through the 2:1 step down transformer they appear as ±2.5V and ±1.5V on the TD± circuits across a 100Ω load. Different wave shapes are shown in *Figure 14*. In this figure, the top two waveforms are TXO+ and TXO− signals with respect to ground. The bottom wave is the voltage across a 100Ω load on TD± circuits and this wave looks smooth because this signal is after passing through the low pass filter.

The current that flows into the TXO± pins is set by the R_{TX} and R_{EQ} resistors. The relationship of the R_{TX} and R_{EQ} resistor values to output current is:

$$I_{TX(out)} = \frac{59.76}{R_{TX}} + \frac{29.88}{R_{EQ}} A$$

$$I_{EQ(out)} = \frac{59.76}{R_{TX}} - \frac{29.88}{R_{EQ}} A$$

The differential current I_{TX} sets the output voltage of the transmitted signal and this current is 50 mA as in *Figure 15(a)*. The equalized portion of the voltage is set by differential current I_{EQ} as in *Figure 15(b)* and this is 40 mA − 10 mA = 30 mA. The above current equations can be used to obtain the desired values of R_{TX} and R_{EQ}.

The above current values can be used to calculate the differential voltage across an equivalent output circuit on TXO± in an alternate way. With the simplified output model in *Figure 14* and the calculations below the figure shows the same differential output voltage as before.

Calculations for *Figure 14a* are:

$V_{TXO-} = 7V - 100 \times 50 \times 10^{-3}V = 2V$

$V_{TXO+} = 7V$

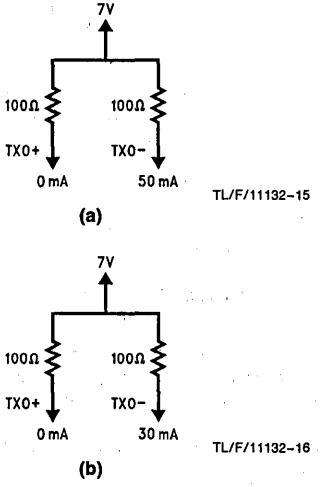
$V_{DIFF} = V_{TXO+} - V_{TXO-} = 7V - 2V = 5V$

Calculations for *Figure 14b* are:

$V_{TXO-} = 7V - 100 \times 30 \times 10^{-3}V = 4V$

$V_{TXO+} = 7V$

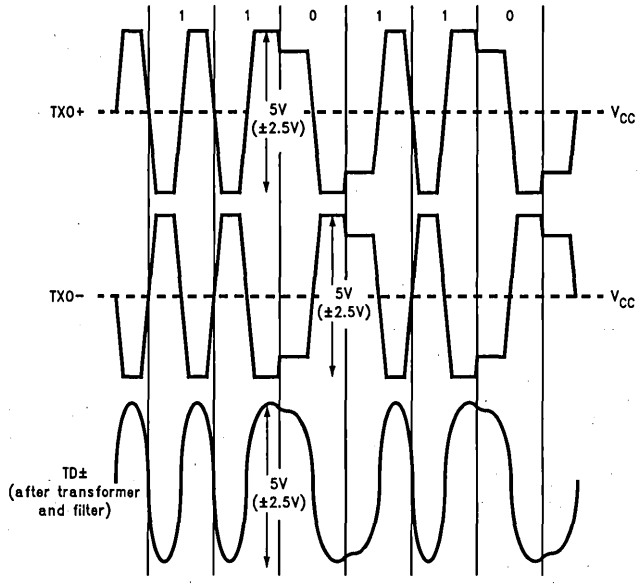
$V_{DIFF} = V_{TXO+} - V_{TXO-} = 7V - 3V = 3V$



TL/F/11132–15

TL/F/11132–16

FIGURE 15. Simplified Output Circuitry of the TPI



TL/F/11132–17

FIGURE 14. Transmitter Output Waveforms

Determining the Transmit Voltage Setting Resistors

We can now determine approximate values of R_{TX} and R_{EQ} by solving the equations above. We will assume that the desired values for $I_{TX(out)}$ = 50 mA and $I_{EQ(out)}$ = 30 mA. By adding the I_{TX} and I_{EQ} equations from the previous page we get:

$$I_{TX} + I_{EQ} = 2\left(\frac{59.76}{R_{TX}}\right)$$

and

$$I_{TX} - I_{EQ} = 2\left(\frac{29.88}{R_{EQ}}\right)$$

We can now take the above equations and solve for R_{TX} and R_{EQ} which yield the following:

$$R_{TX} = 2\left(\frac{59.76}{I_{TX} + I_{EQ}}\right)$$

and

$$R_{EQ} = 2\left(\frac{29.88}{I_{TX} - I_{EQ}}\right)$$

Now plugging in the values I_{TX} = 50 mA and I_{EQ} = 30 mA, we obtain the resistor values as:

$$R_{TX} = 1494\Omega$$

$$R_{EQ} = 298\Omega.$$

We can round up these values to standard resistors available in the market as:

$$R_{TX} = 1.5k\Omega \text{ and } R_{EQ} = 3k\Omega$$

Later it is shown how to optimize these values experimentally.

Isolation Transformer on the Medium Side

According to the standard, the MAU should have 1.5k V_{RMS} isolation. From the test load it is obvious that the transformer on the receive channel shall have inductance of 200 μH. On the receive side DP83922 requires a 1:1 transformer and the transmit side of DP83922 needs a 2:1 transformer to step down the output voltage. For the isolation transformer on the transmit side, it is recommended to have an inductance of 800 μH on the primary side and 200 μH on the secondary side. This inductance is used to get higher time constant so that the idle current through the inductance remains constant when the MAU is transmitting. Also this higher inductance helps impedance matching and consequently provides better return loss specifications.

If any system designer is interested in using discrete filters, any of the following isolation transformers can be used: 500-1741 (B & H), 68124820 (SCHOTT). These transformers provide the above mentioned characteristics.

Transmit Filter Design

High frequency components of the transmit waveform cause electromagnetic interference. To reduce the RF interference we need to add a low pass filter external to the TXO± output. According to the standard when the TX± circuit is driven by all ones Manchester signal, all harmonics measured on the TD± circuits across a 100Ω load must be 27 dB lower than the fundamental 10 MHz. It is required that the filter have minimum attenuation up to 10 MHz and high attenuation at higher frequencies. Also the pass band ripple of the filter must be small.

The twisted-pair media impedance is allowed to vary from 85Ω to 111Ω. A second requirement is that the differential output impedance as measured on the transmitter output shall be such that any reflection caused by a signal incident upon the transmitter from twisted-pair shall be at least 15 dB below the incident over the frequency range of 5.0 MHz to 10 MHz. This is defined as the return loss of the transmitter (and it applies in the same way to the receiver). The equation for calculating return loss from impedance is:

$$R_L = 20 \log_{10} \left(\frac{|Z_{transmitter} + Z_{cable}|}{|Z_{transmitter} - Z_{cable}|} \right)$$

and also

$$R_L = 20 \log_{10} \left(\frac{|V_i|}{|V_r|} \right)$$

where

$Z_{transmitter}$ is the impedance of the transmitter

Z_{cable} is the impedance of the twisted-pair cable

V_i is the differential voltage incident upon the transmitter

V_r is the differential voltage reflected from the transmitter.

The basic termination is set by two 200Ω pullup resistors. The filter will affect the impedance, since the filter will start attenuating the signal slightly above 10 MHz. So the impedance of the filter should be properly matched up to 10 MHz to maintain the return loss requirement. The impedance of the filter should match both at the source and loads. The recommended value of the return loss for the filter by itself is greater than 20 dB from 5 MHz to 10 MHz.

The following calculations show that a 4-pole low pass Butterworth filter meets the requirements:

The order of a Butterworth Filter is given by

$$n \geq \frac{\log \left(\frac{(10^{\alpha min/10} - 1)}{(10^{\alpha max/10} - 1)} \right)}{2 \left(\log \left(\frac{\omega_s}{\omega_p} \right) \right)}$$

Where:

α_{min} = minimum attenuation required at stop band

α_{max} = maximum attenuation in pass band

ω_s = frequency at stop band

ω_p = frequency at pass band

$$n \geq \frac{\log \left(\frac{(10^{27/10} - 1)}{(10^{1/10} - 1)} \right)}{2 \log \left(\frac{30}{10} \right)} = 3.44$$

The order, n = 4. A 4-pole filter can meet the requirements. For RFI considerations we suggest to use a filter of higher order of 7-pole Butterworth.

The normalized values of filter elements for R_{source}/R_{load} = 1, 7-pole Butterworth filter (π-shaped filter, Ref. Table from "Handbook of Filter Synthesis" by Anatol I. Zverev) are:

C_1 = 0.445F

L_2 = 1.247H

C_3 = 1.8019F

L_4 = 2.0H

C_5 = 1.8019F

L_6 = 1.247H

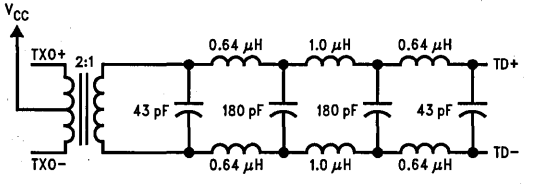
C_7 = 0.445F

We can get the required values of inductors and capacitors by properly scaling the capacitors and inductors as below:

$$C = \frac{C_{normalized}}{(FSF)\,Z} \text{ and } L = L_{normalized} \frac{Z}{FSF}$$

where FSF = Frequency Scaling Factor = $2\pi (16 \times 10^6)$ and Z = Impedance Scaling Factor = 100Ω

Then the inductor values were divided by two to make the filter balanced. The rounded values after calculations are shown in *Figure 16.*



TL/F/11132–18

FIGURE 16. Seven Pole Butterworth Low Pass Filter and 2:1 Transformer

The filter in *Figure 16* provides the following specifications:

1) Seven pole low pass Butterworth filter, balanced
2) 3 dB cut-off frequency: 16 MHz
3) Insertion loss (5 MHz to 10 MHz): ≈0.5 dB
4) Attenuation at 30 MHz: >35 dB
5) I/O impedance: 100Ω
6) Return loss of transmitter: >20 dB
   (5 MHz to 10 MHz)

Note that the impedance of the filter is 100Ω. Here the filter is connected to the twisted-pair side and the isolation transformer is connected directly to the TXO± circuits of DP83922. For proper transmitter output voltage swings, two 200Ω resistors are connected from the TXO± circuits to V_{CC}. These resistors give 400Ω resistance differentially. This 400Ω reflects as 100Ω on the secondary side of the 2:1 transformer. A 400Ω filter can be used if we swap the locations of the filter and transformer.

The Twisted-Pair Receive Interface

The receive inputs of the TPI (RXI±) are connected to a 100Ω resistor to properly terminate the receive line. The resistor then connects to a low pass Butterworth filter which filters out the noise picked up by the twisted-pair medium. The receiver receives all the valid signals specified by 10BASE-T specifications. The intelligent receive squelch of DP83922 rejects the noise and maintains the characteristics required by the 10BASE-T standard. The receiver squelch threshold level is externally set by the R_{RTH} resistor. The equation for receiver threshold voltage is as follows:

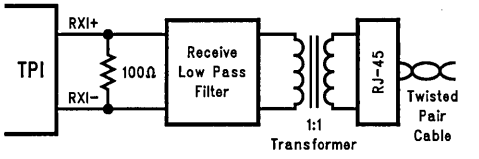
$$V_{RON} = \frac{1}{3}\left(\frac{I_{TX(out)}}{48}\right) R_{RTH}$$

or

$$V_{RON} = \frac{1.245}{3}\left(\frac{1}{R_{TX}} + \frac{1}{2\,R_{EQ}}\right) R_{RTH}$$

Note that the receive threshold voltage is dependent on the resistors that set transmit output levels. It is therefore recommended to design the transmit section first i.e., determine the values of R_{TX} and R_{EQ}, then calculate the R_{RTH} resistor value. (Note: In the previous section the filter effect was not considered while we were calculating the values for R_{TX} and R_{EQ}. The equalization procedure to get the exact values of R_{TX}, R_{EQ}, R_{RTH} is described in the MAU optimization section.)

A common mode choke can be used on the receive path as it was indicated for the transmit circuitry. Common mode choke modules mentioned in the transmit section integrate common mode chokes for both receive and transmit side. The common mode choke in this case provides attenuation to common mode noise on the twisted-pair lines that may not be affected by the receive filter.



TL/F/11132–20

FIGURE 17. Receiver Connection Diagram of TPI

The unshielded twisted-pair medium picks up a lot of noise. The receiver rejection requirement was described before in the Receive Function (d). All the noise that produces those specified levels at the output of the following filter should be rejected by the receiver. The test filter specification is as follows:

1) 3 dB cut off frequency: 15 MHz
2) Insertion loss (5 MHz to 10 MHz): ≤1.0 dB
3) 30 MHz attenuation: 17.5 dB min
4) Input impedance (5 MHz to 10 MHz): 100Ω
5) Return loss with 100Ω load: ≥20 dB
   (5 MHz to 10 MHz)

Again the differential input impedance shall be such that any reflection due to differential signals incident upon the receive circuitry from the twisted-pair side having source impedance from 85Ω to 111Ω, shall be at least 15 dB below the incident over the frequency range of 5 MHz to 10 MHz. The following 5 pole low pass Butterworth filter provides the required needs.

Design of a Butterworth Filter using previous equations:

The normalized values of filter elements for R_{source}/R_{load} = 1, 5-pole Butterworth filter (π-shaped filter) are:

C_1 = 0.618F

L_2 = 1.618H

C_3 = 2.0F

L_4 = 1.618H

C_5 = 0.618F

We can get the exact values of inductors and capacitors by scaling as before in transmit filter.

The inductor value was divided by two to make the filter balanced. *Figure 18* shows the final values of the filter elements.

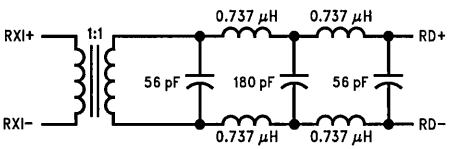
The filter in *Figure 18* provides the following specifications:

1) Five pole low pass Butterworth filter, balanced
2) 3 dB cut-off frequency: 16 MHz
3) Insertion loss (5 MHz to 10 MHz): ≈0.5 dB
4) Attenuation at 30 MHz: >21 dB
5) I/O impedance: 100Ω
6) Return loss of the receiver: >20 dB
   (5 MHz to 10 MHz)

The IEEE standard recommends a filter for both transmit and receive paths, but disclaims the ability to meet RFI requirements. Finally the system designer should take care of the FCC and similar requirements. This may require the use of a higher order filter or the use of a common mode choke.

Integrated Standard Components

Different vendors are integrating receive filter, transmit filter, receive and transmit isolation transformers in simple packages. The part numbers are: PE65423 (Pulse Engineering), PT3884 (Valor Electronics), A556-3899-00 (BEL Fuse). PT3884 integrates the 200Ω pullup resistors on the transmit side and 100Ω termination on the receive path along with the filter and transformer.



TL/F/11132–19

FIGURE 18. Five Pole Butterworth Low Pass Filter and 1:1 Isolation Transformer

Non-10BASE-T Applications

In DP83922 transmit voltage level, transmit signal equalization and receive threshold voltage level are programmable externally. Additionally, the load impedance can be adjusted. The transmit circuitry of the DP83922 is differential throughout the chip. So, the transmitter has a very low skew, and as a result the DP83922 gives very low jitter typically about ±0.5 ns (where the transmitter jitter budget in the standard is ±2 ns). So, this part can be easily used for longer cable applications, and can be used for different types of cable like 150Ω shielded twisted-pair, or in special applications which require modified receive threshold, different transmit level, or non equalized transmit waveform. (The details of implementation is beyond the scope of this document.)

OPTIMIZATION OF MAU TRANSMITTER AND RECEIVER

The values calculated previously for R_{TX}, R_{EQ} and R_{RTH} are approximate. The filter characteristics have an effect on the optimization. Therefore, it is required to optimize the system using the filter and isolation transformer. (Please note that the recommended values specified in the data sheet have already been optimized.)

The IEEE specification requires a MAU to meet the following objectives:

1. Jitter: Jitter added by the transmitter at the end of a twisted-pair cable model for 100m terminated with a 100Ω resistive load has to be within ±3.5 ns and with this equalization the jitter added by the transmitter while the TD± circuit is directly terminated with a 100Ω resistive load (without cable model), has to be within ±8 ns. Moreover, the output voltage waveform has to meet the transmit Voltage Template (*Figure 14-9* of the standard 802.3) at the end of cable model terminated with a 100Ω load.
2. Transmit signal amplitude: Without the twisted-pair cable model while the transmit circuitry is directly terminated with a 100Ω resistive load, the peak differential voltage shall be between 2.2V and 2.8V (4.4V and 5.6V peak-to-peak) for all data sequences.
3. The receiver shall reject all the signals below 300 mV peak.

The objectives of the optimization procedure are:

1. to equalize the transmit signal for minimum jitter at the end of a cable model,
2. to set the transmit signal to a desired amplitude (2.5V peak),
3. to set the receiver threshold to a desired level (400 mV).

To optimize the MAU, the following equipment is required: 1) 10BASE-T MAU with DP83922 and a 10k variable resistor, 2) Twisted-pair cable model (for 100m), 3) A packet generator, 4) 3 power supplies, 5) Oscilloscope. The equipment setup is shown in *Figure 19*.

Step 1: Generate the random signal from a packet generator and check the signal on the oscilloscope. The packet generator should generate pseudo random Manchester encoded data sequence with a minimum repetitive period of 511 bits. Check the jitter given by the packet generator itself and record it.

Step 2: Power up the MAU. Put the MAU in test mode by applying −3V to the TXLED pin (pin 17) with respect to the ground. This will enable the TPI for free running (i.e., disables the Jabber function). Disable the Link function by grounding the GDLNK pin (pin 21). Choose a value of R_{TX} to be 1.5 kΩ and use a 10 kΩ variable resistor for R_{EQ}.

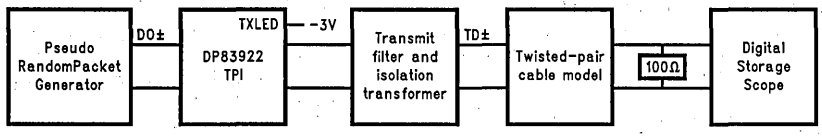
Step 3: Transmit the random signal through the MAU transmit circuitry, DO±. Across the 100Ω resistive load at the end of the cable model, watch the signal jitter on the oscilloscope. Now minimize the jitter by varying R_{EQ}. Get the values of R_{TX} and R_{EQ}. Now take the cable model off and terminate the TD± circuitry directly with a 100Ω resistive load. Measure the output voltage across this 100Ω load.

Step 4: Calculate the ratio of the output amplitude measured to the desired amplitude (2.5V). Multiply R_{TX} and R_{EQ} with this ratio. Replace R_{TX} and R_{EQ} by these calculated values. These values of R_{TX} and R_{EQ} will give the minimum jitter as optimized and the required signal amplitude level of 2.5V.

Step 5: To set the receive threshold the value of R_{RTH} has to be adjusted. For a receive threshold of 400 mV; putting the values of R_{TX}, R_{EQ} in the equation for V_{RON}, R_{RTH} can be calculated.

The transmit level and equalization are effected by the filter being used. So, the values of R_{TX}, R_{EQ} might be different from the values calculated in the previous section "Transmit Design of Twisted-Pair MAU". The values of these resistors by optimizing the MAU with Twisted-Pair cable model and PE65423 are:

$$R_{TX} = 1.65k \text{ and } R_{EQ} = 3.65k.$$



TL/F/11132–21

FIGURE 19. Set Up For Jitter Test

MAU POWER SUPPLY

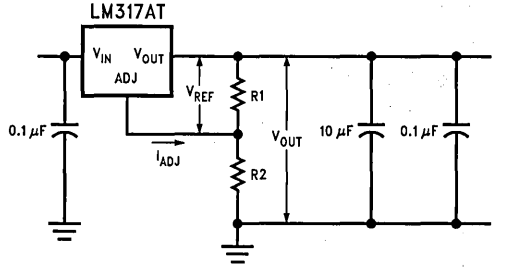
TPI (DP83922) requires a power supply of 7V ±5% and up to 300 mA. This current is the sum of current required by DP83922 itself, other pullup/pulldown resistors and LEDs in the MAU.

$$I_{TOTAL} = I_{CC} + I_{AUI} + I_{TX} + I_{LED}$$

Current through the V_{CC} pin of TPI, I_{CC} = 155 mA typically, total current through the AUI pulldown 360Ω resistors, I_{AUI} = 44 mA, current required by transmit pullup 200Ω resistors = 50 mA, current required by each LED = 10 mA. Typically the Link LED stays on. In the MAU board DP839EB-TP, the power indication LED takes about 20 mA.

This 7V power has to be derived from the power pair of the AUI drop cable which is capable of providing 500 mA in the 12V (−6%) to 15V (+5%) range. An LM317AT-three terminal adjustable regulator can provide adequate regulation. LM317AT is an adjustable 3-terminal positive voltage regulator capable of supplying in excess of 1.5A over a 1.2V to 37V output range. These are easy to use and require only two external resistors to set the output voltage. The LM317 regulator output voltage can be calculated using:

$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right) + I_{adj}R_2.$$



TL/F/11132–22

FIGURE 20. Simple Regulator Power Supply

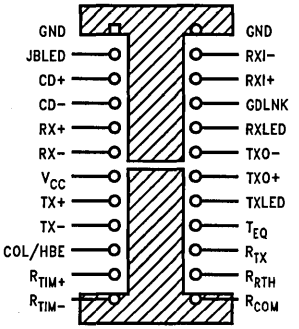
The 100 μA I_{adj} current from the adjustment terminal represents an error term, the LM317AT is designed to minimize I_{adj} and make it very constant with line and load changes. Therefore, to maintain an output voltage of 7V, resistor values of R_1 = 237Ω and R_2 = 1.1k were chosen. An input bypass capacitor of 0.1 μF is recommended to filter out input power noise. A 10 μF tantalum is used to improve ripple rejection. 10 μF capacitor does not reject good frequencies higher than 120 Hz. So a 0.1 μF capacitor is connected to reduce noise.

GENERAL LAYOUT CONSIDERATIONS

For proper heat dissipation there should be enough PCB ground plane on the board. It is highly recommended to have two separate ground planes. One is connected to pins 1 and 24. Other ground plane is connected to pins 12 and 13. This is shown in *Figure 21*. To maintain the reliability of the part at high ambient temperature, it is suggested to use a heat sink. The part number of a clip on heat sink is 580400 manufactured by AAVID Engineering Inc.

Note: The power ground plane connected to pins 1 and 24 should never be connected to ground plane of pins 12 and 13. If only one ground PCB is being used pins 1 and 24 should be used, and not pins 12 and 13.

Solder Side Layout as Viewed from Component Side



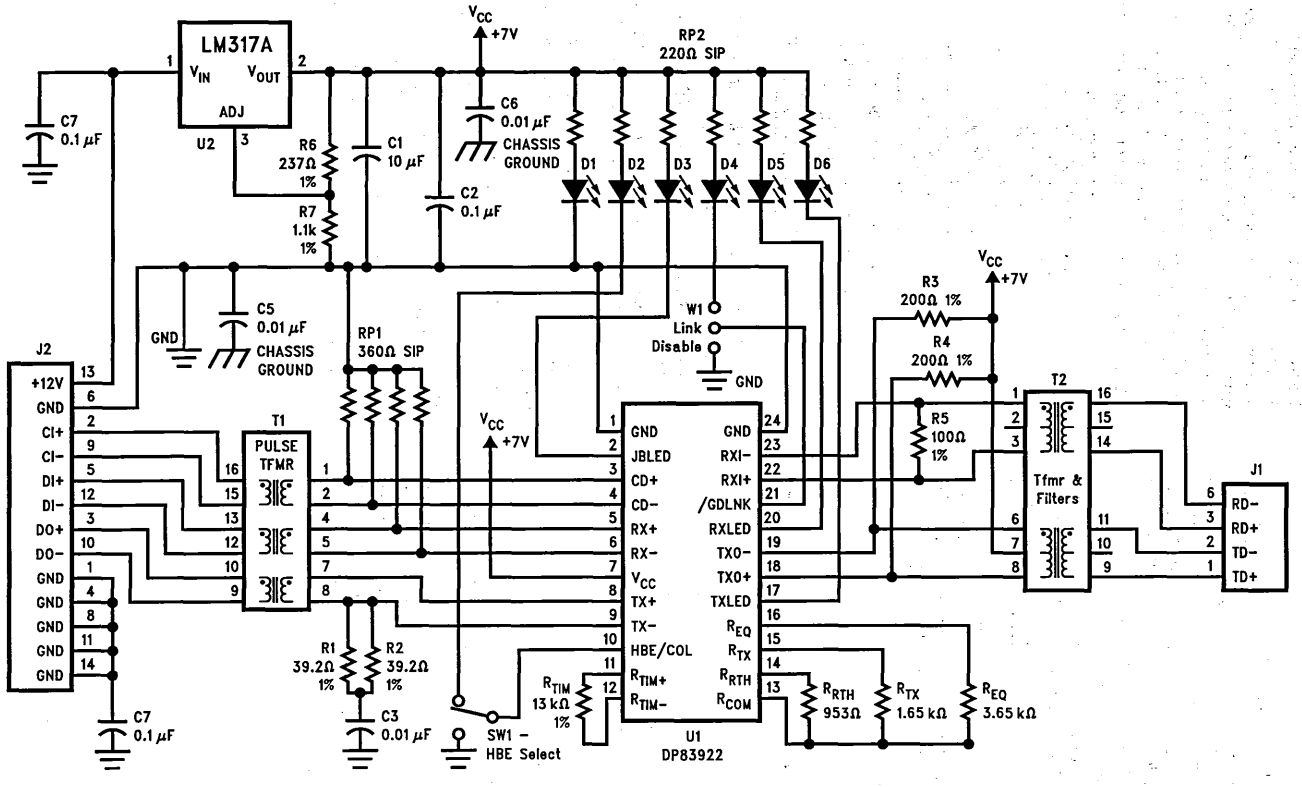
TL/F/11132–23

FIGURE 21. Heat Dissipating Ground Trace

The power supply (LM317A) is good enough to give the required 7 V_{DC} supply. 0.01 μF capacitors (C_3, C_6) as shown in *Figure 22* helps supply noise rejection. These capacitors should be placed close to the power supply. Another 0.01 μF capacitor (C_5) provides a sufficient AC discharge path from chassis ground to DTE ground. The layout path for all the differential signals should be kept equal and symmetric. So that parasitic effects remain equal in both signal paths.

PUTTING THE MAU TOGETHER

Taking all of the pieces that we have discussed the medium attachment unit transceiver can be assembled. The complete transceiver is shown in *Figure 22*. This design combines all the elements discussed previously. This can be implemented in a very small package due to the highly integrated DP83922. In this figure, an integrated filter/transformer module has been used to further simplify the design.



TL/F/11132–24

FIGURE 22. Typical Medium Attachment Unit Application Using the DP83922

Section 3

Ethernet Repeater Interface Controller Products

Section 3 Contents

National Semiconductor

PRELIMINARY

DP83950A
Repeater Interface Controller (RIC)

General Description

The DP83950A Repeater Interface Controller "RIC" may be used to implement an IEEE 802.3 multiport repeater unit. It fully satisfies the IEEE 802.3 repeater specification including the functions defined by the repeater, segment partition and jabber lockup protection state machines.

The RIC has an on-chip phase-locked-loop (PLL) for Manchester data decoding, a Manchester encoder and an Elasticity Buffer for preamble regeneration.

Each RIC can connect to 13 cable segments via its network interface ports. One port is fully AUI compatible and is able to connect to an external MAU using the maximum length of AUI cable. The other 12 ports have integrated 10BASE-T transceivers. These transceiver functions may be bypassed so that the RIC may be used with external transceivers, for example DP8392 coaxial transceivers. In addition, large repeater units, containing several hundred ports may be constructed by cascading RICs together over an Inter-RIC bus.

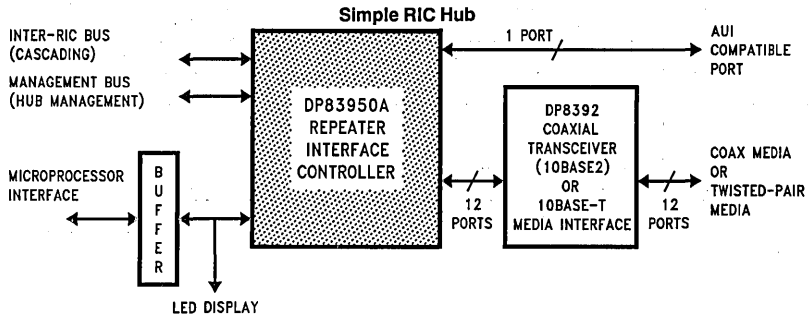
The RIC is configurable for specific applications. It provides port status information for LED array displays and a simple interface for system processors. The RIC posseses multifunction counter and status flag arrays to facilitate network statistics gathering. A serial interface, known as the Management Interface is available for the collection of data in Managed Hub applications.

Features

- Compliant with the IEEE 802.3 Repeater Specification
- 13 network connections (ports) per chip
- Selectable on-chip twisted-pair transceivers
- Cascadable for large hub applications
- Compatible with AUI compliant transceivers
- On-chip Elasticity Buffer, Manchester encoder and decoder
- Separate partition state machines for each port
- Provides port status information for LED displays including: receive, collision, partition and link status
- Power-up configuration options:
  Repeater and Partition Specifications, Transceiver Interface, Status Display, Processor Operations
- Simple processor interface for repeater management and port disable
- On-chip Event Counters and Event Flag Arrays
- Serial Management Interface to combine packet and repeater status information together
- CMOS process for low power dissipation
- Single 5V supply

Table of Contents

1.0 System Diagram



TL/F/11096–1

2.0 Connection Diagram

Pin Table (12 T.P. Ports + 1 AUI Bottom View)

Pin Name	Pin No.
TXO12P−	A15
TXO12+	A14
TXO12−	B14
TXO12P+	C13
RXI12−	B13
RXI12+	A13
V_{CC}	C12
GND	C11
RXI11−	B12
RXI11+	B11
TXO11P+	A12
TXO11−	A11
TXO11+	C10
TXO11P−	A10
V_{CC}	B10
GND	B9
TXO10P−	C9
TXO10+	C8
TXO10−	A9
TXO10P+	A8
RXI10−	B8
RXI10+	B7
V_{CC}	C7
GND	A7
RXI9−	A6
RXI9+	B6
TXO9P+	C6
TXO9−	C5
TXO9+	B5
TXO9P−	A5
V_{CC}	A4
GND	B4
TXO8P−	C4
TXO8+	A3
TXO8−	C3
TXO8P+	D4
RXI8−	B3
RXI8+	B2
V_{CC}	A2
GND	D3

Pin Name	Pin No.
RXI7−	C2
RXI7+	A1
TXO7P+	B1
TXO7−	D2
TXO7+	E3
TXO7P−	F3
V_{CC}	C1
GND	D1
TXO6P−	E2
TXO6+	G3
TXO6−	F2
TXO6P+	E1
RXI6−	G2
RXI6+	H3
NC	F1
NC	G1
V_{CC}	H2
GND	J3
RXI5−	J2
RXI5+	H1
TXO5P+	J1
TXO5−	K1
TXO5+	K3
TXO5P−	K2
V_{CC}	L1
GND	L2
TXO4P−	M1
TXO4+	L3
TXO4−	M2
TXO4P+	N1
RXI4−	N2
RXI4+	M3
V_{CC}	P1
GND	R1
RXI3−	P2
RXI3+	N3
TXO3P+	P3
TXO3−	R2
TXO3+	N4
TXO3P−	R3

Pin Name	Pin No.
V_{CC}	S1
GND	P4
TXO2P−	S2
TXO2+	S3
TXO2−	R4
TXO2P+	P5
RXI2−	R5
RXI2+	S4
V_{CC}	S5
GND	S6
RX1−	P6
RX1+	R6
CD1−	S7
CD1+	R7
TX1−	P7
TX1+	P8
V_{CC}	R8
GND	S8
V_{CC}	S9
GND	R9
CLKIN	P9
RA4	S10
RA3	R10
RA2	S11
RA1	P10
RA0	R11
V_{CC}	S12
GND	R12
$\overline{MLOAD}$	P11
$\overline{CDEC}$	S13
$\overline{WR}$	R13
$\overline{RD}$	S14
D7	P12
D6	R14
D5	S15
D4	P13
D3	P14
D2	R15
D1	S16
D0	R16

Pin Name	Pin No.
V_{CC}	N13
GND	P15
IRC	N14
IRE	P16
IRD	N15
COLN	N16
V_{CC}	M15
GND	M14
PKEN	L14
RXM	L15
$\overline{BUFEN}$	M16
$\overline{RDY}$	L16
$\overline{ELI}$	K16
$\overline{RTI}$	K14
$\overline{STR1}$	K15
V_{CC}	J16
GND	J15
$\overline{STR0}$	J14
ACTND	H16
ANYXND	H15
$\overline{ACKO}$	H14
MRXC	G14
MEN	G15
MRXD	G16
MCRS	F16
V_{CC}	F14
GND	F15
$\overline{ACKI}$	E15
ACTNS	E14
ANYXNS	E16
$\overline{PCOMP}$	D16
RXI13−	D15
RXI13+	D14
TXO13P+	C16
TXO13−	C15
TXO13+	B16
TXO13P−	B15
V_{CC}	D13
GND	C14

Note: NC = No Connect

2.0 Connection Diagram (Continued)

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
S	V_{CC} 98	TXO2P− 96	TXO2+ 95	RXI2+ 91	V_{CC} 90	GND 89	CD1− 86	GND 81	V_{CC} 80	RA4 77	RA2 75	V_{CC} 72	$\overline{CDEC}$ 69	$\overline{RD}$ 67	D5 64	D1 60
R	GND 105	TXO3− 101	TXO3P− 99	TXO2− 94	RXI2− 92	RX1+ 87	CD1+ 85	V_{CC} 82	GND 79	RA3 76	RA0 73	GND 71	$\overline{WR}$ 68	D6 65	D2 61	D0 59
P	V_{CC} 106	RXI3− 104	TXO3P+ 102	GND 97	TXO2P+ 93	RX1− 88	TX1− 84	TX1+ 83	CLKIN 78	RA1 74	$\overline{MLOAD}$ 70	D7 66	D4 63	D3 62	GND 57	IRE 55
N	TXO4P+ 109	RXI4− 108	RXI3+ 103	TXO3+ 100									V_{CC} 58	IRC 56	IRD 54	COLN 53
M	TXO4P− 112	TXO4− 110	RXI4+ 107											GND 51	V_{CC} 52	$\overline{BUFEN}$ 48
L	V_{CC} 114	GND 113	TXO4+ 111											PKEN 50	RXM 49	$\overline{RDY}$ 47
K	TXO5− 117	TXO5P− 115	TXO5+ 116											$\overline{RTI}$ 45	$\overline{STR1}$ 44	$\overline{ELI}$ 46
J	TXO5P+ 118	RXI5− 120	GND 121											$\overline{STR0}$ 41	GND 42	V_{CC} 43
H	RXI5+ 119	V_{CC} 122	RXI6+ 125											$\overline{ACK0}$ 38	ANYXND 39	ACTND 40
G	NC 123	RXI6− 126	TXO6+ 129											MRXC 37	MEN 36	MRXD 35
F	NC 124	TXO6− 128	TXO7P− 133											V_{CC} 33	GND 32	MCRS 34
E	TXO6P+ 127	TXO6P− 130	TXO7+ 134											ACTNS 30	$\overline{ACKI}$ 31	ANYXNS 29
D	GND 131	TXO7− 135	GND 139	TXO8P+ 143									V_{CC} 21	RXI13+ 26	RXI13− 27	$\overline{PCOMP}$ 28
C	V_{CC} 132	RXI7− 138	TXO8− 144	TXO8P− 146	TXO9− 151	TXO9P+ 152	V_{CC} 156	TXO10+ 2	TXO10P− 3	TXO11+ 7	GND 12	V_{CC} 13	TXO12P+ 16	GND 20	TXO13− 24	TXO13P+ 25
B	TXO7P+ 136	RXI8+ 141	RXI8− 142	GND 147	TXO9+ 150	RXI9+ 153	RXI10+ 157	RXI10− 158	GND 4	V_{CC} 5	RXI11+ 10	RXI11− 11	RXI12− 15	TXO12− 17	TXO13P− 22	TXO13+ 23
A	RXI7+ 137	V_{CC} 140	TXO8+ 145	V_{CC} 148	TXO9P− 149	RXI9− 154	GND 155	TXO10P+ 159	TXO10− 1	TXO11P− 6	TXO11− 8	TXO11P+ 9	RXI12+ 14	TXO12+ 18	TXO12P− 19	

TL/F/11096-2

Bottom View
1 AUI + 2–13 T.P. Ports

2.0 Connection Diagram (Continued)

Pin Table (1-5 AUI + 6-13 T.P. Ports)

Pin Name	Pin No.
TXO12P−	A15
TXO12+	A14
TXO12−	B14
TXO12P+	C13
RXI12−	B13
RXI12+	A13
V_{CC}	C12
GND	C11
RXI11−	B12
RXI11+	B11
TXO11P+	A12
TXO11−	A11
TXO11+	C10
TXO11P−	A10
V_{CC}	B10
GND	B9
TXO10P−	C9
TXO10+	C8
TXO10−	A9
TXO10P+	A8
RXI10−	B8
RXI10+	B7
V_{CC}	C7
GND	A7
RXI9−	A6
RXI9+	B6
TXO9P+	C6
TXO9−	C5
TXO9+	B5
TXO9P−	A5
V_{CC}	A4
GND	B4
TXO8P−	C4
TXO8+	A3
TXO8−	C3
TXO8P+	D4
RXI8−	B3
RXI8+	B2
V_{CC}	A2
GND	D3

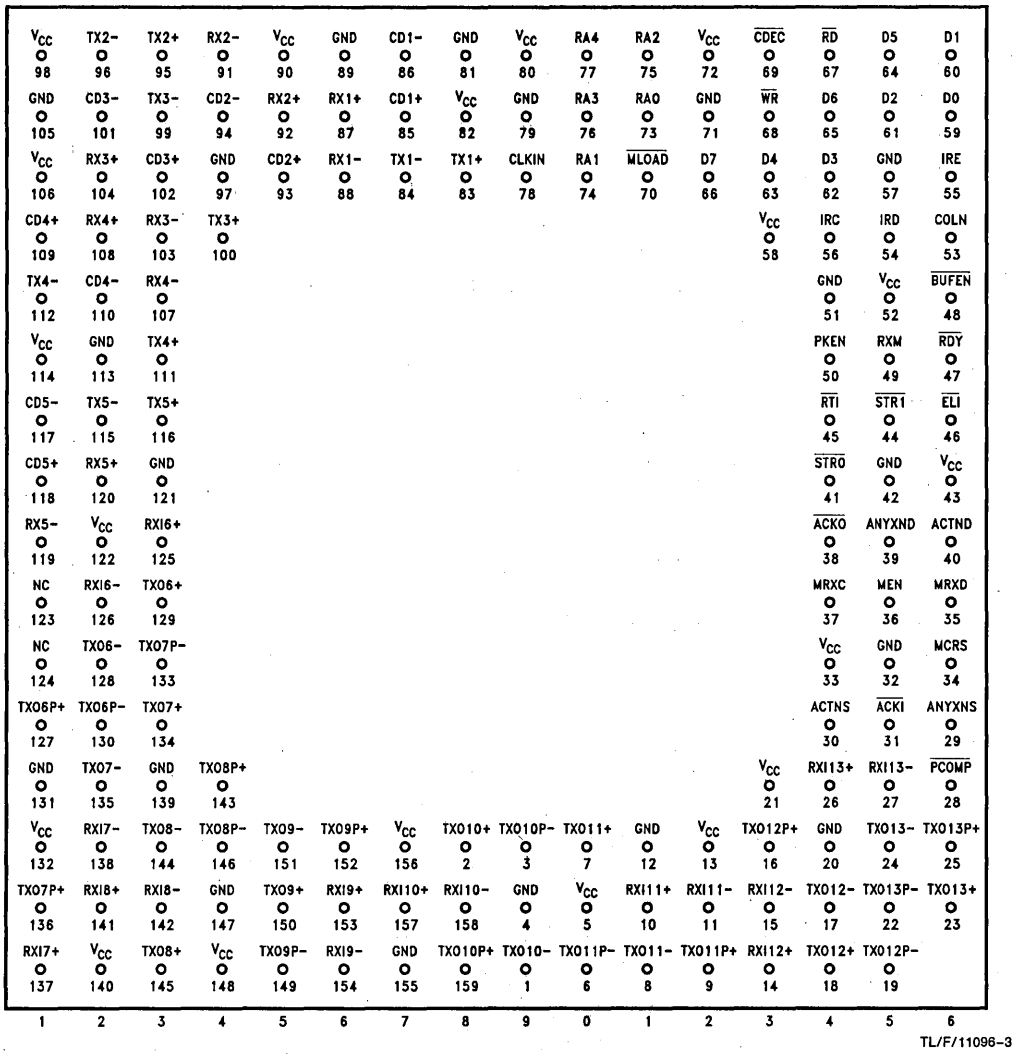
Pin Name	Pin No.
RXI7−	C2
RXI7+	A1
TXO7P+	B1
TXO7−	D2
TXO7+	E3
TXO7P−	F3
V_{CC}	C1
GND	D1
TXO6P−	E2
TXO6+	G3
TXO6−	F2
TXO6P+	E1
RXI6−	G2
RXI6+	H3
NC	F1
NC	G1
V_{CC}	H2
GND	J3
RX5+	J2
RX5−	H1
CD5+	J1
CD5−	K1
TX5+	K3
TX5−	K2
V_{CC}	L1
GND	L2
TX4−	M1
TX4+	L3
CD4−	M2
CD4+	N1
RX4+	N2
RX4−	M3
V_{CC}	P1
GND	R1
RX3+	P2
RX3−	N3
CD3+	P3
CD3−	R2
TX3+	N4
TX3−	R3

Pin Name	Pin No.
V_{CC}	S1
GND	P4
TX2−	S2
TX2+	S3
CD2−	R4
CD2+	P5
RX2+	R5
RX2−	S4
V_{CC}	S5
GND	S6
RX1−	P6
RX1+	R6
CD1−	S7
CD1+	R7
TX1−	P7
TX1+	P8
V_{CC}	R8
GND	S8
V_{CC}	S9
GND	R9
CLKIN	P9
RA4	S10
RA3	R10
RA2	S11
RA1	P10
RA0	R11
V_{CC}	S12
GND	R12
$\overline{MLOAD}$	P11
$\overline{CDEC}$	S13
$\overline{WR}$	R13
$\overline{RD}$	S14
D7	P12
D6	R14
D5	S15
D4	P13
D3	P14
D2	R15
D1	R16
D0	R16

Pin Name	Pin No.
V_{CC}	N13
GND	P15
IRC	N14
IRE	P16
IRD	N15
COLN	N16
V_{CC}	M15
GND	M14
PKEN	L14
RXM	L15
$\overline{BUFEN}$	M16
$\overline{RDY}$	L16
$\overline{ELI}$	K16
$\overline{RTI}$	K14
$\overline{STR1}$	K15
V_{CC}	J16
GND	J15
$\overline{STR0}$	J14
ACTND	H16
ANYXND	H15
$\overline{ACKO}$	H14
MRXC	G14
MEN	G15
MRXD	G16
MCRS	F16
V_{CC}	F14
GND	F15
$\overline{ACKI}$	E15
ACTNS	E14
ANYXNS	E16
$\overline{PCOMP}$	D16
RXI13−	D15
RXI13+	D14
TXO13P+	C16
TXO13−	C15
TXO13+	B16
TXO13P−	B15
V_{CC}	D13
GND	C14

Note: NC = No Connect

2.0 Connection Diagram (Continued)



	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
S	V_{CC} 98	TX2− 96	TX2+ 95	RX2− 91	V_{CC} 90	GND 89	CD1− 86	GND 81	V_{CC} 80	RA4 77	RA2 75	V_{CC} 72	$\overline{CDEC}$ 69	$\overline{RD}$ 67	D5 64	D1 60
R	GND 105	CD3− 101	TX3− 99	CD2− 94	RX2+ 92	RX1+ 87	CD1+ 85	V_{CC} 82	GND 79	RA3 76	RA0 73	GND 71	$\overline{WR}$ 68	D6 65	D2 61	D0 59
P	V_{CC} 106	RX3+ 104	CD3+ 102	GND 97	CD2+ 93	RX1− 88	TX1− 84	TX1+ 83	CLKIN 78	RA1 74	$\overline{MLOAD}$ 70	D7 66	D4 63	D3 62	GND 57	IRE 55
N	CD4+ 109	RX4+ 108	RX3− 103	TX3+ 100									V_{CC} 58	IRC 56	IRD 54	COLN 53
M	TX4− 112	CD4− 110	RX4− 107											GND 51	V_{CC} 52	$\overline{BUFEN}$ 48
L	V_{CC} 114	GND 113	TX4+ 111											PKEN 50	RXM 49	$\overline{RDY}$ 47
K	CD5− 117	TX5− 115	TX5+ 116											$\overline{RTI}$ 45	$\overline{STR1}$ 44	$\overline{ELI}$ 46
J	CD5+ 118	RX5+ 120	GND 121											$\overline{STRO}$ 41	GND 42	V_{CC} 43
H	RX5− 119	V_{CC} 122	RXI6+ 125											$\overline{ACKO}$ 38	ANYXND 39	ACTND 40
G	NC 123	RXI6− 126	TXO6+ 129											MRXC 37	MEN 36	MRXD 35
F	NC 124	TXO6− 128	TXO7P− 133											V_{CC} 33	GND 32	MCRS 34
E	TXO6P+ 127	TXO6P− 130	TXO7+ 134											ACTNS 30	$\overline{ACKI}$ 31	ANYXNS 29
D	GND 131	TXO7− 135	GND 139	TXO8P+ 143									V_{CC} 21	RXI13+ 26	RXI13− 27	$\overline{PCOMP}$ 28
C	V_{CC} 132	RXI7− 138	TXO8− 144	TXO8P− 146	TXO9− 151	TXO9P+ 152	V_{CC} 156	TXO10+ 2	TXO10P− 3	TXO11+ 7	GND 12	V_{CC} 13	TXO12P+ 16	GND 20	TXO13− 24	TXO13P+ 25
B	TXO7P+ 136	RXI8+ 141	RXI8− 142	GND 147	TXO9+ 150	RXI9+ 153	RXI10+ 157	RXI10− 158	GND 4	V_{CC} 5	RXI11+ 10	RXI11− 11	RXI12− 15	TXO12− 17	TXO13P− 22	TXO13+ 23
A	RXI7+ 137	V_{CC} 140	TXO8+ 145	V_{CC} 148	TXO9P− 149	RXI9− 154	GND 155	TXO10P+ 159	TXO10− 1	TXO11P− 6	TXO11− 8	TXO11P+ 9	RXI12+ 14	TXO12+ 18	TXO12P− 19	

TL/F/11096–3

Bottom View
1–5 AUI + 6–13 T.P. Ports

2.0 Connection Diagram (Continued)

Pin Table (1–7 AUI + 8–13 T.P. Ports)

Pin Name	Pin No.
TXO12P−	A15
TXO12+	A14
TXO12−	B14
TXO12P+	C13
RXI12−	B13
RXI12+	A13
V_{CC}	C12
GND	C11
RXI11−	B12
RXI11+	B11
TXO11P+	A12
TXO11−	A11
TXO11+	C10
TXO11P−	A10
V_{CC}	B10
GND	B9
TXO10P−	C9
TXO10+	C8
TXO10−	A9
TXO10P+	A8
RXI10−	B8
RXI10+	B7
V_{CC}	C7
GND	A7
RXI9−	A6
RXI9+	B6
TXO9P+	C6
TXO9−	C5
TXO9+	B5
TXO9P−	A5
V_{CC}	A4
GND	B4
TXO8P−	C4
TXO8+	A3
TXO8−	C3
TXO8P+	D4
RXI8−	B3
RXI8+	B2
V_{CC}	A2
GND	D3

Pin Name	Pin No.
RX7+	C2
RX7−	A1
CD7+	B1
CD7−	D2
TX7+	E3
TX7−	F3
V_{CC}	C1
GND	D1
TX6−	E2
TX6+	G3
CD6−	F2
CD6+	E1
RX6+	G2
RX6−	H3
NC	F1
NC	G1
V_{CC}	H2
GND	J3
RX5+	J2
RX5−	H1
CD5+	J1
CD5−	K1
TX5+	K3
TX5−	K2
V_{CC}	L1
GND	L2
TX4−	M1
TX4+	L3
CD4−	M2
CD4+	N1
RX4+	N2
RX4−	M3
V_{CC}	P1
GND	R1
RX3+	P2
RX3−	N3
CD3+	P3
CD3−	R2
TX3+	N4
TX3−	R3

Pin Name	Pin No.
V_{CC}	S1
GND	P4
TX2−	S2
TX2+	S3
CD2−	R4
CD2+	P5
RX2+	R5
RX2−	S4
V_{CC}	S5
GND	S6
RX1−	P6
RX1+	R6
CD1−	S7
CD1+	R7
TX1−	P7
TX1+	P8
V_{CC}	R8
GND	S8
V_{CC}	S9
GND	R9
CLKIN	P9
RA4	S10
RA3	R10
RA2	S11
RA1	P10
RA0	R11
V_{CC}	S12
GND	R12
$\overline{MLOAD}$	P11
$\overline{CDEC}$	S13
$\overline{WR}$	R13
$\overline{RD}$	S14
D7	P12
D6	R14
D5	S15
D4	P13
D3	P14
D2	R15
D1	S16
D0	R16

Pin Name	Pin No.
V_{CC}	N13
GND	P15
IRC	N14
IRE	P16
IRD	N15
COLN	N16
V_{CC}	M15
GND	M14
PKEN	L14
RXM	L15
$\overline{BUFEN}$	M16
$\overline{RDY}$	L16
$\overline{ELI}$	K16
$\overline{RTI}$	K14
$\overline{STR1}$	K15
V_{CC}	J16
GND	J15
$\overline{STR0}$	J14
ACTND	H16
ANYXND	H15
$\overline{ACKO}$	H14
MRXC	G14
MEN	G15
MRXD	G16
MCRS	F16
V_{CC}	F14
GND	F15
$\overline{ACKI}$	E15
ACTNS	E14
ANYXNS	E16
$\overline{PCOMP}$	D16
RXI13−	D15
RXI13+	D14
TXO13P+	C16
TXO13−	C15
TXO13+	B16
TXO13P−	B15
V_{CC}	D13
GND	C14

Note: NC = No Connect

2.0 Connection Diagram (Continued)

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
S	V_{CC} 98	TX2− 96	TX2+ 95	RX2− 91	V_{CC} 90	GND 89	CD1− 86	GND 81	V_{CC} 80	RA4 77	RA2 75	V_{CC} 72	$\overline{CDEC}$ 69	$\overline{RD}$ 67	D5 64	D1 60
R	GND 105	CD3− 101	TX3− 99	CD2− 94	RX2+ 92	RX1+ 87	CD1+ 85	V_{CC} 82	GND 79	RA3 76	RA0 73	GND 71	$\overline{WR}$ 68	D6 65	D2 61	D0 59
P	V_{CC} 106	RX3+ 104	CD3+ 102	GND 97	CD2+ 93	RX1− 88	TX1− 84	TX1+ 83	CLKIN 78	RA1 74	$\overline{MLOAD}$ 70	D7 66	D4 63	D3 62	GND 57	IRE 55
N	CD4+ 109	RX4+ 108	RX3− 103	TX3+ 100									V_{CC} 58	IRC 56	IRD 54	COLN 53
M	TX4− 112	CD4− 110	RX4− 107											GND 51	V_{CC} 52	$\overline{BUFEN}$ 48
L	V_{CC} 114	GND 113	TX4+ 111											PKEN 50	RXM 49	$\overline{RDY}$ 47
K	CD5− 117	TX5− 115	TX5+ 116											$\overline{RTI}$ 45	$\overline{STR1}$ 44	$\overline{ELI}$ 46
J	CD5+ 118	RX5+ 120	GND 121											$\overline{STRO}$ 41	GND 42	V_{CC} 43
H	RX5− 119	V_{CC} 122	RX6− 125											$\overline{ACKO}$ 38	ANYXND 39	ACTND 40
G	NC 123	RX6+ 126	TX6+ 129											MRXC 37	MEN 36	MRXD 35
F	NC 124	CD6− 128	TX7− 133											V_{CC} 33	GND 32	MCRS 34
E	CD6+ 127	TX6− 130	TX7+ 134											ACTNS 30	$\overline{ACKI}$ 31	ANYXNS 29
D	GND 131	CD7− 135	GND 139	TX08P+ 143									V_{CC} 21	RXI13+ 26	RXI13− 27	$\overline{PCOMP}$ 28
C	V_{CC} 132	RX7+ 138	TX08− 144	TX08P− 146	TX09− 151	TX09P+ 152	V_{CC} 156	TX010+ 2	TX010P− 3	TX011+ 7	GND 12	V_{CC} 13	TX012P+ 16	GND 20	TX013− 24	TX013P+ 25
B	CD7+ 136	RXI8+ 141	RXI8− 142	GND 147	TX09+ 150	RXI9+ 153	RXI10+ 157	RXI10− 158	GND 4	V_{CC} 5	RXI11+ 10	RXI11− 11	RXI12− 15	TX012− 17	TX013P− 22	TX013+ 23
A	RX7− 137	V_{CC} 140	TX08+ 145	V_{CC} 148	TX09P− 149	RXI9− 154	GND 155	TX010P+ 159	TX010− 1	TX011P− 6	TX011− 8	TX011P+ 9	RXI12+ 14	TX012+ 18	TX012P− 19	

TL/F/11096–4

Bottom View
1–7 AUI + 8–13 T.P. Ports

3

2.0 Connection Diagram (Continued)

Pin Table (All AUI Ports)

Pin Name	Pin No.
TX12−	A15
TX12+	A14
CD12−	B14
CD12+	C13
RX12+	B13
RX12−	A13
V_{CC}	C12
GND	C11
RX11+	B12
RX11−	B11
CD11+	A12
CD11−	A11
TX11+	C10
TX11−	A10
V_{CC}	B10
GND	B9
TX10−	C9
TX10+	C8
CD10−	A9
CD10+	A8
RX10+	B8
RX10−	B7
V_{CC}	C7
GND	A7
RX9+	A6
RX9−	B6
CD9+	C6
CD9−	C5
TX9+	B5
TX9−	A5
V_{CC}	A4
GND	B4
TX8−	C4
TX8+	A3
CD8−	C3
CD8+	D4
RX8+	B3
RX8−	B2
V_{CC}	A2
GND	D3

Pin Name	Pin No.
RX7+	C2
RX7−	A1
CD7+	B1
CD7−	D2
TX7+	E3
TX7−	F3
V_{CC}	C1
GND	D1
TX6−	E2
TX6+	G3
CD6−	F2
CD6+	E1
RX6+	G2
RX6−	H3
NC	F1
NC	G1
V_{CC}	H2
GND	J3
RX5+	J2
RX5−	H1
CD5+	J1
CD5−	K1
TX5+	K3
TX5−	K2
V_{CC}	L1
GND	L2
TX4−	M1
TX4+	L3
CD4−	M2
CD4+	N1
RX4+	N2
RX4−	M3
V_{CC}	P1
GND	R1
RX3+	P2
RX3−	N3
CD3+	P3
CD3−	R2
TX3+	N4
TX3−	R3

Pin Name	Pin No.
V_{CC}	S1
GND	P4
TX2−	S2
TX2+	S3
CD2−	R4
CD2+	P5
RX2+	R5
RX2−	S4
V_{CC}	S5
GND	S6
RX1−	P6
RX1+	R6
CD1−	S7
CD1+	R7
TX1−	P7
TX1+	P8
V_{CC}	R8
GND	S8
V_{CC}	S9
GND	R9
CLKIN	P9
RA4	S10
RA3	R10
RA2	S11
RA1	P10
RA0	R11
V_{CC}	S12
GND	R12
$\overline{MLOAD}$	P11
$\overline{CDEC}$	S13
$\overline{WR}$	R13
$\overline{RD}$	S14
D7	P12
D6	R14
D5	S15
D4	P13
D3	P14
D2	R15
D1	S16
D0	R16

Pin Name	Pin No.
V_{CC}	N13
GND	P15
IRC	N14
IRE	P16
IRD	N15
COLN	N16
V_{CC}	M15
GND	M14
PKEN	L14
RXM	L15
$\overline{BUFEN}$	M16
$\overline{RDY}$	L16
$\overline{ELI}$	K16
$\overline{RTI}$	K14
$\overline{STR1}$	K15
V_{CC}	J16
GND	J15
$\overline{STR0}$	J14
ACTND	H16
ANYXND	H15
$\overline{ACKO}$	H14
MRXC	G14
MEN	G15
MRXD	G16
MCRS	F16
V_{CC}	F14
GND	F15
$\overline{ACKI}$	E15
ACTNS	E14
ANYXNS	E16
$\overline{PCOMP}$	D16
RX13+	D15
RX13−	D14
CD13+	C16
CD13−	C15
TX13+	B16
TX13−	B15
V_{CC}	D13
GND	C14

Note: NC = No Connect

2.0 Connection Diagram (Continued)

	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
S	V_{CC} 98	TX2− 96	TX2+ 95	RX2− 91	V_{CC} 90	GND 89	CD1− 86	GND 81	V_{CC} 80	RA4 77	RA2 75	V_{CC} 72	$\overline{CDEC}$ 69	$\overline{RD}$ 67	D5 64	D1 60
R	GND 105	CD3− 101	TX3− 99	CD2− 94	RX2+ 92	RX1+ 87	CD1+ 85	V_{CC} 82	GND 79	RA3 76	RA0 73	GND 71	$\overline{WR}$ 68	D6 65	D2 61	D0 59
P	V_{CC} 106	RX3+ 104	CD3+ 102	GND 97	CD2+ 93	RX1− 88	TX1− 84	TX1+ 83	CLKIN 78	RA1 74	$\overline{MLOAD}$ 70	D7 66	D4 63	D3 62	GND 57	IRE 55
N	CD4+ 109	RX4+ 108	RX3− 103	TX3+ 100									V_{CC} 58	IRC 56	IRD 54	COLN 53
M	TX4− 112	CD4− 110	RX4− 107											GND 51	V_{CC} 52	$\overline{BUFEN}$ 48
L	V_{CC} 114	GND 113	TX4+ 111											PKEN 50	RXM 49	$\overline{RDY}$ 47
K	CD5− 117	TX5− 115	TX5+ 116											$\overline{RTI}$ 45	$\overline{STR1}$ 44	$\overline{ELI}$ 46
J	CD5+ 118	RX5+ 120	GND 121											$\overline{STR0}$ 41	GND 42	V_{CC} 43
H	RX5− 119	V_{CC} 122	RX6− 125											$\overline{ACK0}$ 38	ANYXND 39	ACTND 40
G	NC 123	RX6+ 126	TX6+ 129											MRXC 37	MEN 36	MRXD 35
F	NC 124	CD6− 128	TX7− 133											V_{CC} 33	GND 32	MCRS 34
E	CD6+ 127	TX6− 130	TX7+ 134											ACTNS 30	$\overline{ACK1}$ 31	ANYXNS 29
D	GND 131	CD7− 135	GND 139	CD8+ 143									V_{CC} 21	RXI13+ 26	RXI13− 27	$\overline{PCOMP}$ 28
C	V_{CC} 132	RX7+ 138	CD8− 144	TX8− 146	CD9− 151	CD9+ 152	V_{CC} 156	TX10+ 2	TX10− 3	TX11+ 7	GND 12	V_{CC} 13	CD12+ 16	GND 20	CD13− 24	CD13+ 25
B	CD7+ 136	RX8− 141	RX8+ 142	GND 147	TX9+ 150	RX9− 153	RX10− 157	RX10+ 158	GND 4	V_{CC} 5	RX11− 10	RX11+ 11	RX12+ 15	CD12− 17	TX13− 22	TX13+ 23
A	RX7− 137	V_{CC} 140	TX8+ 145	V_{CC} 148	TX9− 149	RX9+ 154	GND 155	CD10+ 159	CD10− 1	TX11− 6	CD11− 8	CD11+ 9	RX12− 14	TX12+ 18	TX12− 19	

TL/F/11096–5

Bottom View
All AUI Ports

3.0 Pin Description

Pin No.	Pin Name	Driver Type	I/O	Description
NETWORK INTERFACE PINS (On-Chip Transceiver Mode)				
	RXI2− to RXI13−	TP	I	Twisted Pair Receive Input Negative
	RXI2+ to RXI13+	TP	I	Twisted Pair Receive Input Positive
	TXOP2− to TXOP13−	TT	O	Twisted Pair Pre-emphasis Transmit Output Negative
	TXO2− to TXO13−	TT	O	Twisted Pair Transmit Output Negative
	TXO2+ to TXO13+	TT	O	Twisted Pair Transmit Output Positive
	TXOP2+ to TXOP13+	TT	O	Twisted Pair Pre-emphasis Transmit Output Positive
	CD1+	AL	I	AUI Collision Detect Input Positive
	CD1−	AL	I	AUI Collision Detect Input Negative
	RX1+	AL	I	AUI Receive Input Positive
	RX1−	AL	I	AUI Receive Input Negative
	TX1+	AD	O	AUI Transmit Output Positive
	TX1−	AD	O	AUI Transmit Output Negative
NETWORK INTERFACE PINS (External Transceiver Mode **AUI Signal Level Compatibility Selected)**				
	TX2+ to TX13+	AL	O	Transmit Output Positive
	TX2− to TX13−	AL	O	Transmit Output Negative
	CD2+ to CD13+	AL	I	Collision Input Positive
	CD2− to CD13−	AL	I	Collision Input Negative
	RX2+ to RX13+	AL	I	Receive Input Positive
	RX2− to RX13−	AL	I	Receive Input Negative
	CD1+	AL	I	AUI Collision Detect Input Positive
	CD1−	AL	I	AUI Collision Detect Input Negative
	RX1+	AL	I	AUI Receive Input Positive
	RX1−	AL	I	AUI Receive Input Negative
	TX1+	AD	O	AUI Transmit Output Positive
	TX1−	AD	O	AUI Transmit Output Negative

Note: AD = AUI level and Drive compatible, **TP** = Twisted Pair interface compatible, **AL** = AUI Level compatible, **TT** = TTL compatible, **I** = Input, **O** = Output.

3.0 Pin Description (Continued)

Pin No.	Pin Name	Driver Type	I/O	Description
PROCESSOR BUS PINS				
	RA0–RA4	TT	I	**R**EGISTER **A**DDRESS INPUTS: These five pins are used to select a register to be read or written. The state of these inputs are ignored when the read, write and mode load input strobes are high. (Even under these conditions these inputs must not be allowed to float at an undefined logic state).
	$\overline{\text{STR0}}$	C	O	DISPLAY UPDATE **STR**OBE 0 **Maximum Display Mode:** This signal controls the latching of display data for network ports 1 to 7 into the off chip display latches. **Minimum Display Mode:** This signal controls the latching of display data for the RIC into the off chip display latch. During processor access cycles (read or write is asserted) this signal is inactive (high).
	$\overline{\text{STR1}}$	C	O	DISPLAY UPDATE **STR**OBE 1 **Maximum Display Mode:** This signal controls the latching of display data for network ports 8 to 13 into the off chip display latches. **Minimum Display Mode:** No operation During processor access cycles (read or write is asserted) this signal is inactive (high).
	D0–D7	TT	B, Z	**D**ATA BUS **Display Update Cycles:** These pins become outputs providing display data and port address information. Address information only available in Maximum Display mode. **Processor Access Cycles:** Data input or output is performed via these pins. The read, write and mode load inputs control the direction of the signals. **Note:** The data pins remain in their display update function, i.e., asserted as outputs unless either the read or write strobe is asserted.
	$\overline{\text{BUFEN}}$	C	O	**BUF**FER **EN**ABLE: This output controls the TRI-STATE® operation of the bus transceiver which provides the interface between the RIC's data pins and the processor's data bus. **Note:** The buffer enable output indicates the function of the data pins. When it is high they are performing display update cycles, when it is low a processor access or mode load cycle is occurring.
	$\overline{\text{RDY}}$	C	O	DATA **READY** STROBE: The falling edge of this signal during a read cycle indicates that data is stable and valid for sampling. In write cycles the falling edge of $\overline{\text{RDY}}$ denotes that the write data has been latched by the RIC. Therefore data must have been available and stable for this operation to be successful.
	$\overline{\text{ELI}}$	C	O	**E**VENT **L**OGGING **I**NTERRUPT: A low level on the $\overline{\text{ELI}}$ output indicates the RIC's hub management logic requires CPU attention. The interrupt is cleared by accessing the Port Event Recording register or Event Counter that produced it. All interrupt sources may be masked.
	$\overline{\text{RTI}}$	C	O	**R**EAL **T**IME **I**NTERRUPT: A low level on the $\overline{\text{RTI}}$ output indicates the RIC's real time (packet specific) interrupt logic requires CPU attention. The interrupt is cleared by reading the Real Time Interrupt Status register. All interrupt sources may be masked.
	$\overline{\text{CDEC}}$	TT	I	**C**OUNTER **DEC**REMENT: A low level on the $\overline{\text{CDEC}}$ input strobe decrements all of the RIC's Port Event Counters by one. This input is internally synchronized and if necessary the operation of the signal is delayed if there is a simultaneous internally generated counting operation.
	$\overline{\text{WR}}$	TT	I	**WR**ITE STROBE: Strobe from the CPU used to write an internal register defined by the RA0–RA4 inputs.
	$\overline{\text{RD}}$	TT	I	**R**EAD STROBE: Strobe from the CPU used to read an internal register defined by the RA0–RA4 inputs.
	$\overline{\text{MLOAD}}$	TT	I	DEVICE RESET AND **M**ODE **LOAD**: When this input is low all of the RIC's state machines, counters and network ports are reset and held inactive. On the rising edge of $\overline{\text{MLOAD}}$ the logic levels present on the D0–7 pins and RA0–RA4 inputs are latched into the RIC's configuration registers. The rising edge of $\overline{\text{MLOAD}}$ also signals the beginning of the display test operation.

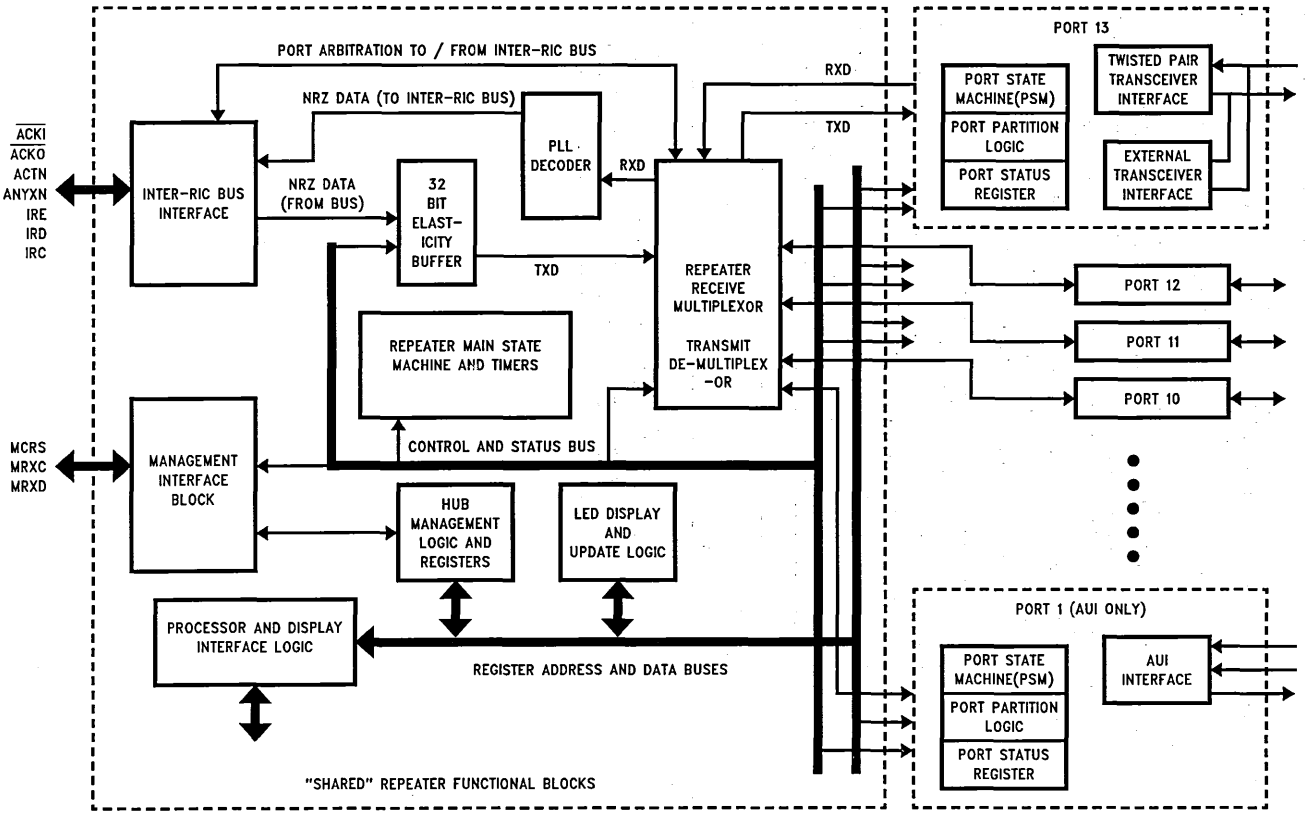
3.0 Pin Description (Continued)

Pin No.	Pin Name	Driver Type	I/O	Description
INTER-RIC BUS PINS				
	$\overline{\text{ACKI}}$	TT	I	**ACK**NOWLEDGE INPUT: Input to the network ports' arbitration chain.
	$\overline{\text{ACKO}}$	TT	O	**ACK**NOWLEDGE **O**UTPUT: Output from the network ports' arbitration chain.
	IRD	TT	B, Z	INTER-**R**IC **D**ATA: When asserted as an output this signal provides a serial data stream in NRZ format. The signal is asserted by a RIC when it is receiving data from one of its network segments. The default condition of this signal is to be an input. In this state it may be driven by other devices on the Inter-RIC bus.
	IRE	TT	B, Z	INTER-**R**IC **E**NABLE: When asserted as an output this signal provides an activity framing enable for the serial data stream. The signal is asserted by a RIC when it is receiving data from one of its network segments. The default condition of this signal is to be an input. In this state it may be driven by other devices on the Inter-RIC bus.
	IRC	TT	B, Z	INTER-**R**IC **C**LOCK: When asserted as an output this signal provides a clock signal for the serial data stream. Data (IRD) is changed on the falling edge of the clock. The signal is asserted by a RIC when it is receiving data from one of its network segments. The default condition of this signal is to be an input. When an input IRD is sampled on the rising edge of the clock. In this state it may be driven by other devices on the Inter-RIC bus.
	COLN	TT	B, Z	**COL**LISION ON PORT **N**: This denotes that a collision is occurring on the port receiving the data packet. The default condition of this signal is to be an input. In this state it may be driven by other devices on the Inter-RIC bus.
	PKEN	C	O	**P**A**C**KET **EN**ABLE: This output acts as an active high enable for an external bus transceiver (if required) for the IRE, IRC IRD and COLN signals. When high the bus transceiver should be transmitting on to the bus, i.e., this RIC is driving the IRD, IRE, IRC and COLN bus lines. When low the bus transceiver should receive from the bus.
	CLKIN	TT	I	40 MHz **CLOCK IN**PUT: This input is used to generate the RIC's timing reference for the state machines, and phase lock loop decoder.
	ACTND	OD	O	**ACT**IVITY ON PORT **N D**RIVE: This output is active when the RIC is receiving data or collision information from one of its network segments.
	ACTNS	TT	I	**ACT**IVITY ON PORT **N S**ENSE: This input senses when this or another RIC in a multi-RIC system is receiving data or collision information.
	ANYXND	OD	O	**A**CTIVITY ON **ANY** PORT **EX**CLUDING PORT **N D**RIVE: This output is active when a RIC is experiencing a transmit collision or multiple ports have active collisions on their network segments.
	ANYXNS	TT	I	**A**CTIVITY ON **ANY** PORT **EX**CLUDING PORT **N S**ENSE: This input senses when this RIC or other RICs in a multi-RIC system are experiencing transmit collisions or multiple ports have active collisions on their network segments.

3.0 Pin Description (Continued)

Pin No.	Pin Name	Driver Type	I/O	Description
MANAGEMENT BUS PINS				
	MRXC	TT	O, Z	**M**ANAGEMENT **R**ECEIVE **C**LOCK: When asserted this signal provides a clock signal for the MRXD serial data stream. The MRXD signal is changed on the falling edge of this clock. The signal is asserted when a RIC is receiving data from one of its network segements. Otherwise the signal is inactive.
	MCRS	TT	B, Z	**M**ANAGEMENT **C**ARRIE**R** **S**ENSE: When asserted this signal provides an activity framing enable for the serial output data stream (MRXD). The signal is asserted when a RIC is receiving data from one of its network segments. Otherwise the signal is an input.
	MRXD	TT	O, Z	**M**ANAGEMENT **R**ECEIVE **D**ATA: When asserted this signal provides a serial data stream in NRZ format. The data stream is made up of the data packet and RIC status information. The signal is asserted when a RIC is receiving data from one of its network segments. Otherwise the signal is inactive.
	MEN	C	O	**M**ANAGEMENT BUS OUTPUT **EN**ABLE: This output acts as an active high enable for an external bus transceiver (if required) for the MRXC, MCRS and MRXD signals. When high the bus transceiver should be transmitting on to the bus.
	$\overline{\text{PCOMP}}$	TT	I	PACKET **COMP**RESS: This input is used to activate the RIC's packet compress logic. A low level on this signal when MCRS is active will cause that packet to be compressed. If $\overline{\text{PCOMP}}$ is tied low all packets are compressed, if $\overline{\text{PCOMP}}$ is tied high packet compression is inhibited.
POWER AND GROUND PINS				
	V_{CC}			Positive Supply
	GND			Negative Supply
EXTERNAL DECODER PINS				
	RXM	TT	O	**R**ECEIVE DATA MANCHESTER FORMAT: This output makes the data, in Manchester format, received by port N available for test purposes. If not used for testing this pin should be left open.

Note: TT = **T**TL compatible, **B** = **B**i-directional, **C** = **C**MOS compatible, **OD** = **O**pen **D**rain, **I** = **I**nput, **O** = **O**utput, **Z** = TRI-STATE



TL/F/11096–6

FIGURE 5.1

5.0 Functional Description

The I.E.E.E. repeater specification details a number of functions a repeater system must perform. These requirements allied with a need for the implementation to be multiport strongly favors the choice of a modular design style. In such a design, functionality is split between those tasks common to all data channels and those exclusive to each individual channel. The RIC follows this approach, certain functional blocks are replicated for each network attachment, (also known as a repeater port), and others are shared. The following section briefly describes the functional blocks in the RIC.

5.1 OVERVIEW OF RIC FUNCTIONS

Segment Specific Block: Network Port

As shown in the Block Diagram, the segment specific blocks consist of:

1. One or more physical layer interfaces.
2. A logic block required for performing repeater operations upon that particular segment. This is known as the "port" logic since it is the access "port" the segment has to the rest of the network.

This function is repeated 13 times in the RIC (one for each port) and is shown on the right side of the Block Diagram, *Figure 5.1*.

The physical layer interfaces provided depends upon the port under examination. Port 1 has an AUI compliant interface for use with AUI compatible transceiver boxes and cable. Ports 2 to 13 may be configured for use with one of two interfaces: twisted pair or an external transceiver. The former utilizes the RIC's on-chip 10BASE-T transceivers, the latter allows connection to external transceivers. When using the external transceiver mode the interface is AUI compatible. Although AUI compatible transceivers are supported the interface is not designed for use with an interface cable, thus the transceivers are necessarily internal to the repeater equipment.

Inside the port logic there are 3 distinct functions:

1. The port state machine "PSM" is required to perform data and collision repetition as described by the repeater specification, for example, it determines whether this port should be receiving from or transmitting to its network segment.
2. The port partition logic implements the segment partitioning algorithm. This algorithm is defined by the IEEE specification and is used to protect the network from malfunctioning segements.
3. The port status register reflects the current status of the port. It may be accessed by a system processor to obtain this status or to perform certain port configuration operations, such as port disable.

Shared Functional Blocks: Repeater Core Logic

The shared functional blocks consist of the Repeater Main State Machine (MSM) and Timers, a 32 bit Elasticity Buffer, PLL Decoder, and Receive and Transmit Multiplexors. These blocks perform the majority of the operations needed to fulfill the requirements of the IEEE repeater specification.

When a packet is received by a port it is sent via the Receive Multiplexor to the PLL Decoder. Notification of the data and collision status is sent to the main state machine via the receive multiplexor and collision activity status signals. This enables the main state machine to determine the source of the data to be repeated and the type of data to be transmitted. The transmit data may be either the received packet's data field or a preamble/jam pattern consisting of a 1010 . . . bit pattern.

Associated with the main state machine are a series of timers. These ensure various IEEE specification times (referred to as the TW1 to TW6 times) are fulfilled.

A repeater unit is required to meet the same signal jitter performance as any receiving node attached to a network segment. Consequently, a phase locked loop Manchester decoder is required so that the packet may be decoded, and the jitter accumulated over the receiving segment recovered. The decode logic outputs data in NRZ format with an associated clock and enable. In this form the packet is in a convenient format for transfer to other devices, such as network controllers and other RICs, via the Inter-RIC bus (described later). The data may then be re-encoded into Manchester data and transmitted.

Reception and transmission via physical layer transceiver units causes a loss of bits in the preamble field of a data packet. The repeater specification requires this loss to be compensated for. To accomplish this an elasticity buffer is employed to temporarily store bits in the data field of the packet.

The sequence of operation is as follows:

Soon after the network segment receiving the data packet has been identified, the RIC begins to transmit the packet preamble pattern (1010 . . .) onto the other network segments. While the preamble is being transmitted the Elasticity Buffer monitors the decoded received clock and data signals (this is done via the Inter-RIC bus as described later). When the start of frame delimiter "SFD" is detected the received data stream is written into the elasticity buffer. Removal of data from the buffer for retransmission is not allowed until a valid length preamble pattern has been transmitted.

Inter-RIC Bus Interface

Using the RIC in a repeater system allows the design to be constructed with many more network attachments than can be supported by a single chip. The split of functions already described allows data packets and collision status to be transferred between multiple RICs, and at the same time the multiple RICs still behave as a single logical repeater. Since all RICs in a repeater system are identical and capable of performing any of the repetition operations, the failure of one RIC will not cause the failure of the entire system. This is an important issue in large multiport repeaters.

RICs communicate via a specialized interface known as the Inter-RIC bus. This allows the data packet to be transferred from the receiving RIC to the other RICs in the system. These RICs then transmit the data stream to their segments. Just as important as data transfer is the notification of collisions occurring across the network. The Inter-RIC bus has a set of status lines capable of conveying collision information between RICs to ensure their main state machines operate in the appropriate manner.

5.0 Functional Description (Continued)

LED Interface and Hub Management Function

Repeater systems usually possess optical displays indicating network activity and the status of specific repeater operations. The RIC's display update block provides the system designer with a wide variety of indicators. The display updates are completely autonomous and merely require SSI logic devices to drive the display devices, usually made up of light emitting diodes, LEDs. The status display is very flexible allowing the user to choose those indicators appropriate for the specification of the equipment.

The RIC has been designed with special awareness for system designers implementing large repeaters possessing hub management capabilities. Hub management uses the unique position of repeaters in a network to gather statistics about the network segments they are attached to. The RIC provides hub management statistical data in 3 steps. Important events are gathered by the management block from logic blocks throughout the chip. These events may then be stored in on-chip latches or counted in on-chip counters according to user supplied latching and counting masks.

The fundamental task of a hub management system implementation is to associate the current packet and any management status information with the network segment, i.e., repeater port where the packet was received. The ideal system would place this combined data packet and status field in system memory for examination by hub management software. The ultimate function of the RIC's hub management support logic is to provide this function.

To accomplish this the RIC utilizes a dedicated hub management interface. This is similar to the Inter-RIC bus since it allows the data packet to be recovered from the receiving RIC. Unlike the Inter-RIC bus the intended recipient is not another RIC but National Semiconductor's DP83932 "SONIC™" Network controller. The use of a dedicated bus allows a management status field to be appended at the end of the data packet. This can be done without affecting the operation of the repeater system.

Processor Interface

The RIC's processor interface allows connection to a system processor. Data transfer occurs via an octal bi-directional data bus. The RIC has a number of on-chip registers indicating the status of the hub management functions, chip configuration and port status. These may be accessed by providing the chosen address at the Register Address (RA4–RA0) input pins.

Display update cycles and processor accesses occur utilizing the same data bus. An on-chip arbiter in the processor/display block schedules and controls the accesses and ensures the correct information is written into the display latches. During the display update cycles the RIC behaves as a master of its data bus. This is the default state of the data bus. Consequently, a TRI-STATE buffer must be placed between the RIC and the system processor's data bus. This ensures bus contention is avoided during simultaneous display update cycles and processor accesses of other devices on the system bus. When the processor accesses a RIC register, the RIC enables the data buffer and selects the operation, either input or output, of the data pins.

5.2 DESCRIPTION OF REPEATER OPERATIONS

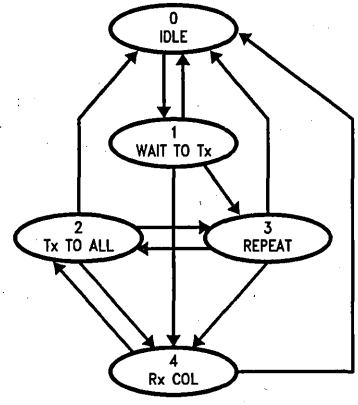
In order to implement a multi-chip repeater system which behaves as though it were a single logical repeater, special consideration must be paid to the data path used in packet repetition. For example, where in the path are specific operations such as Manchester decoding and elasticity buffering performed. Also the system's state machines which utilize available network activity signals, must be able to accommodate the various packet repetition and collision scenarios detailed in the repeater specification.

The RIC contains two types of inter-acting state machines. These are:

1. Port State Machines (PSMs). Every network attachment has its own PSM.
2. Main State Machine (MSM). This state machine controls the shared functional blocks as shown in the block diagram *Figure 5.1.*

Repeater Port and Main State Machines

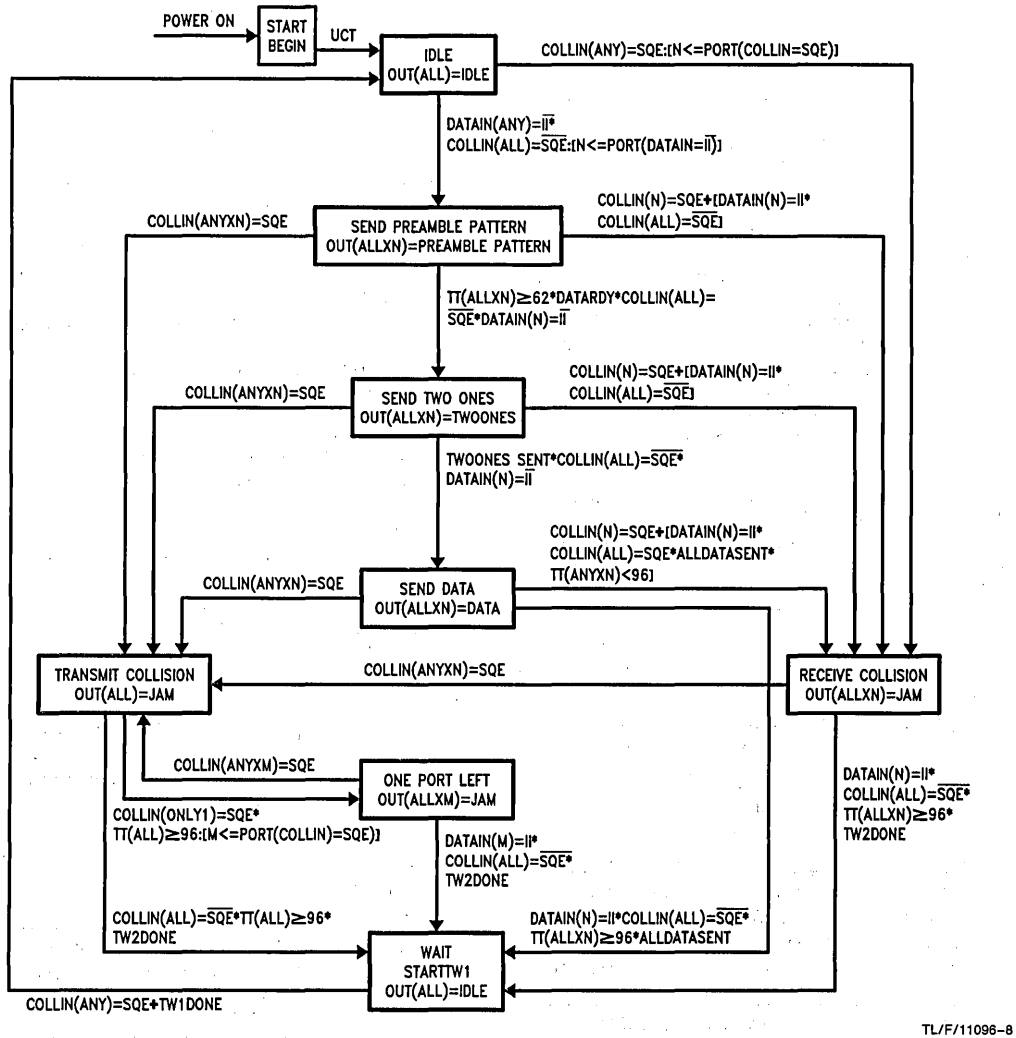
These two state machines are described in the following sections. Reference is made to expressions used in the IEEE Repeater specification. For the precise definition of these terms please refer to the specification. To avoid confusion with the RIC's implementation, where references are made to repeater states or terms as described in the IEEE specification, these items are written in *italics.* The IEEE state diagram is shown in *Figure 5-3*, the Inter-RIC bus state diagram is shown in *Figure 5-2.*



TL/F/11096–7

FIGURE 5.2. Inter-RIC Bus State Diagram

5.0 Functional Description (Continued)



TL/F/11096–8

FIGURE 5.3. IEEE Repeater Main State Diagram

5.0 Functional Description (Continued)

Port State Machine (PSM)

There are two primary functions for the PSM as follows:

1. Control the transmission of repeated data and jam signals over the attached segment.
2. Decide whether a port will be the source of data or collision information which will be repeated over the network. This repeater port is known as *PORT N*. An arbitration process is required to enable the repeater to transition from the *IDLE* state to the *SEND PREAMBLE PATTERN* or *RECEIVE COLLISION* states, see *Figure 5.3*. This process is used to locate the port which will be *PORT N* for that particular packet. The data received from this port is directed to the PLL decoder and transmitted over the Inter-RIC bus. If the repeater enters the *TRANSMIT COLLISION* state a further arbitration operation is performed to determine which port is *PORT M*. *PORT M* is differentiated from the repeater's other ports if the repeater enters the *ONE PORT LEFT* state. In this state *PORT M* does not transmit to its segment; where as all other ports are still required to transmit to their segments.

Main State Machine (MSM)

The MSM controls the operation of the shared functional blocks in each RIC as shown in the block diagram, *Figure 5.1*, and it performs the majority of the data and collision propagation operations as defined by the IEEE specification, these include:

Function	Action
Preamble Regeneration	Restore the length of the preamble pattern to the defined size.
Fragment Extension	Extend received data or collision fragments to meet the minimum fragment length of 96 bits.
Elasticity Buffer Control	A portion of the received packet may require storage in an Elasticity Buffer to accommodate preamble regeneration.
Jam/ Preamble Pattern Generation	In cases of receive or transmit collisions a RIC is required to transmit a jam pattern (1010 . . .). **Note:** This pattern is the same as that used for preamble regeneration.
Transmit Collision Enforcement	Once the *TRANSMIT COLLISION* state is entered a repeater is required to stay in this state for at least 96 network bit times.
Data Encoding Control	NRZ format data from the elasticity buffer must be encoded into Manchester format data prior to retransmission.
Tw1 Enforcement	Enforce the Transmit Recovery Time specification.
Tw2 Enforcement	Enforce Carrier Recovery Time specification on all ports with active collisions.

The interaction of the main and port state machines is visible, in part, by observing the Inter-RIC bus.

Inter-RIC Bus Operation

Overview

The Inter-RIC Bus consists of eight signals. These signals implement a protocol which may be used to connect multiple RICs together. In this configuration, the logical function of a single repeater is maintained. The resulting multi-RIC system is compliant to the IEEE 802.3 repeater specification and may connect several hundred network segments. An example of a multi-RIC system is shown in *Figure 5.4*.

The Inter-RIC Bus connects multiple RICs to realize the following operations:

Port N Identification (which port the repeater receives data from)
Port M Identification (which port is the last one experiencing a collision)
Data Transfer
RECEIVE COLLISION identification
TRANSMIT COLLISION identification
DISABLE OUTPUT (jabber protection)

The following tables briefly describes the operation of each bus signal, the conditions required for a RIC to assert a signal and which RICs (in a multi-RIC system) would monitor a signal:

$\overline{\text{ACKI}}$	
Function	Input signal to the PSM arbitration chain. This chain is employed to identify *PORT N* and *PORT M*. **Note:** A RIC which contains *PORT N* or *PORT M* may be identified by its $\overline{\text{ACKO}}$ signal being low when its $\overline{\text{ACKI}}$ input is high.
Conditions required for a RIC to drive this signal	Not applicable
RIC Receiving the signal	This is dependent upon the method used to cascade RICs, described in a following section.

$\overline{\text{ACKO}}$	
Function	Output signal from the PSM arbitration chain.
Conditions required for a RIC to drive this signal	This is dependent upon the method used to cascade RICs, described in a following section.
RIC Receiving the Signal	Not applicable

5.0 Functional Description (Continued)

ACTN	
Function	This signal denotes there is activity on *PORT N* or *PORT M*.
Conditions required for a RIC to drive this signal	A RIC must contain *PORT N* or *PORT M*. **Note:** Although this signal normally has only one source asserting the signal active it is used in a wired-or configuration.
RIC Receiving the Signal	The signal is monitored by all RICs in the repeater system.

ANYXN	
Function	This signal denotes that a repeater port that is not *PORT N* or *PORT M* is experiencing a collision.
Conditions required for a RIC to drive this signal	Any RIC which satisfies the above condition. **Note:** This bus line is used in a wired-or configuration.
RIC Receiving the Signal	The signal is monitored by all RICs in the repeater system.

COLN	
Function	Denotes *PORT N* or *PORT M* is experiencing a collision.
Conditions required for a RIC to drive this signal	A RIC must contain *PORT N* or *PORT M*.
RIC Receiving the Signal	The Signal is monitored by all other RICs in the repeater system.

IRE	
Function	This signal acts as an activity framing signal for the IRC and IRD signals.
Conditions required for a RIC to drive this signal	A RIC must contain *PORT N*.
RIC Receiving the Signal	The Signal is monitored by all other RICs in the repeater system.

IRD	
Function	Decoded serial data, in NRZ format, received from the network segment attached to *PORT N*.
Conditions required for a RIC to drive this signal	A RIC must contain *PORT N*.
RIC Receiving the Signal	The signal is monitored by all other RICs in the repeater system.

IRC	
Function	Clock signal associated with IRD and IRE.
Conditions required for a RIC to drive this signal	A RIC must contain *PORT N*.
RIC Receiving the Signal	The signal is monitored by all other RICs in the repeater system.

Methods of RIC Cascading

In order to build multi-RIC repeaters *PORT N* and *PORT M* identification must be performed across all the RICs in the system. Inside each RIC the PSMs are arranged in a logical arbitration chain where port 1 is the highest and port 13 the lowest. The top of the chain, the input to port 1 is accessible to the user via the RIC's $\overline{\text{ACKI}}$ input pin. The output from the bottom of the chain becomes the $\overline{\text{ACKO}}$ output pin. In a single RIC system *PORT N* is defined as the highest port in the arbitration chain with receive or collision activity. *Port N* identification is performed when the repeater is in the *IDLE* state. *PORT M* is defined as the highest port in the chain with a collision when the repeater leaves the *TRANSMIT COLLISION* state. In order for the arbitration chain to function, all that needs to be done is to tie the $\overline{\text{ACKI}}$ signal to a logic high state. In multi-RIC systems there are two methods to propagate the arbitration chain between RICs:

The first and most straight forward is to extend the arbitration chain by daisy chaining the $\overline{\text{ACKI}}$ $\overline{\text{ACKO}}$ signals between RICs. In this approach one RIC is placed at the top of the chain (its $\overline{\text{ACKI}}$ input is tied high), then the $\overline{\text{ACKO}}$ signal from this RIC is sent to the $\overline{\text{ACKI}}$ input of the next RIC and so on. This arrangement is simple to implement but it places some topological restrictions upon the repeater system. In particular, if the repeater is constructed using a backplane with removable printed circuit boards. (These boards contain the RICs and their associated components). If one of the boards is removed then the $\overline{\text{ACKI}}$ $\overline{\text{ACKO}}$ chain will be broken and the repeater will not operate correctly.

5.0 Functional Description (Continued)

The second method of *PORT N* or *M* identification avoids this problem. This second technique relies on an external parallel arbiter which monitors all of the RIC's $\overline{ACKO}$ signals and responds to the RIC with the highest priority. In this scheme each RIC is assigned with a priority level. One method of doing this is to assign a priority number which reflects the position of a RIC board on the repeater backplane, i.e., its slot number. When a RIC experiences receive activity and the repeater system is in the *IDLE* state, the RIC board will assert $\overline{ACKO}$. External arbitration logic drives the identification number onto an arbitration bus and the RIC containing *PORT N* will be identified. An identical procedure is used in the *TRANSMIT COLLISION* state to identify *PORT M*. This parallel means of arbitration is not subject to the problems caused by missing boards, i.e., empty slots in the backplane. The logic associated with asserting this arbitration vector in the various packet repetition scenarios could be implemented in programmable logic type devices.

To perform *PORT N* or *M* arbitration both of the above methods employ the same signals: $\overline{ACKI}$, $\overline{ACKO}$ and ACTN.

The Inter-RIC bus allows multi-RIC operations to be performed in exactly the same manner as if there is only a single RIC in the system. The simplest way to describe the operation of Inter-RIC bus is to see how it is used in a number of common packet repetition scenarios. Throughout this description the RICs are presumed to be operating in external transceiver mode. This is advantageous for the explanation since the receive, transmit and collision signals from each network segment are observable. In internal transceiver mode this is not the case, since the collision signal for the non-AUI ports is derived by the transceivers inside the RIC.

5.3 EXAMPLES OF PACKET REPETITION SCENARIOS

Data Repetition

The simplest packet operation performed over the Inter-RIC Bus is data repetition. In this operation a data packet is received at one port and transmitted to all other segments.

The first task to be performed is *PORT N* identification. This is an arbitration process performed by the Port State Machines in the system. In situations where two or more ports simultaneously receive packets the Inter-RIC bus operates by choosing one of the active ports and forcing the others to transmit data. This is done to faithfully follow the IEEE specification's allowed exit paths from the *IDLE* state, i.e., to the *SEND PREAMBLE PATTERN* or *RECEIVE COLLISION* states.

The packet begins with a preamble pattern derived from the RIC's on chip jam/preamble generator. The data received at *PORT N* is directed through the receive multiplexor to the PLL decoder. Once phase lock has been achieved, the decoded data, in NRZ format, with its associated clock and enable signals are asserted onto the IRD IRE and IRC Inter-RIC bus lines. This serial data stream is received from the bus by all RICs in the repeater and directed to their Elasticity Buffers. Logic circuits monitor the data stream and look for the Start of Frame Delimiter (SFD). When this has been detected data is loaded into the elasticity buffer for later transmission. This will occur when sufficient preamble has been transmitted and certain internal state machine operations have been fulfilled.

Figure 5.4 shows two RICs A and B, daisy chained together with RIC A positioned at the top of the chain. A packet is received at port B1 of RIC B and is then repeated by the other ports in the system. *Figure 5.5* shows the functional timing diagram for this packet repetition represented by the signals shown in *Figure 5.4*. In this example only two ports in the system are shown, obviously the other ports also repeat the packet. It also indicates the operation of the RICs' state machines in so far as can be seen by observing the Inter-RIC bus. For reference, the repeater's state transitions are shown in terms of the states defined by the IEEE specification. The location, i.e., which port it is, of *PORT N* is also shown. The following section describes the repeater and Inter-RIC bus transitions shown in *Figure 5.5*.

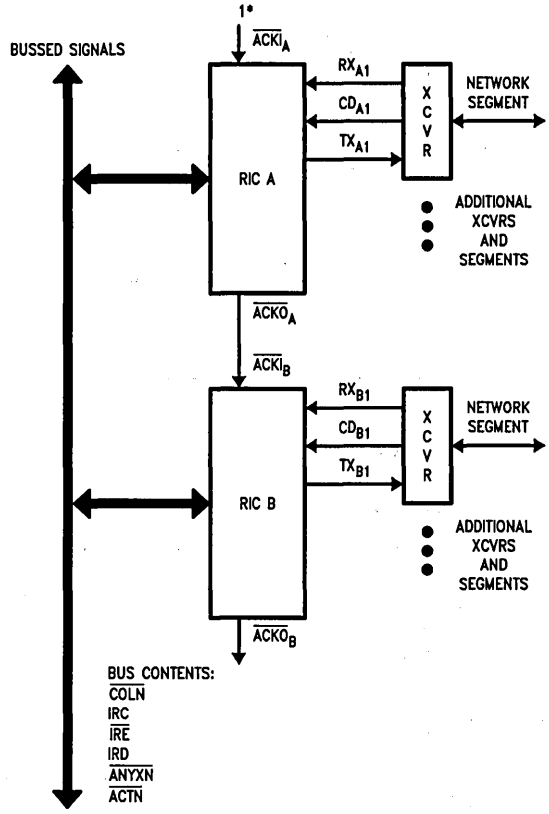
The repeater is stimulated into activity by the data signal received by port B1. The RICs in the system are alerted to forthcoming repeater operation by the falling edges on the $\overline{ACKI}$ $\overline{ACKO}$ daisy chain and the ACTN bus signal. Following a defined start up delay the repeater moves to the *SEND PREAMBLE* state. The RIC system utilizes the start up delay to perform port arbitration. When packet transmission begins the RIC system enter the REPEAT state.

The expected, for normal packet repetition, sequence of repeater states, *SEND PREAMBLE*, *SEND SFD* and *SEND DATA* is followed but is not visible upon the Inter-RIC bus. They are merged together into a single REPEAT state. This is also true for the *WAIT* and *IDLE* states, they appear as a combined Inter-RIC bus IDLE state.

Once a repeat operation has begun, i.e., the repeater leaves the *IDLE* state. It is required to transmit at least 96 bits of data or jam/preamble onto its network segments. If the duration of the received signal from *PORT N* is smaller than 96 bits, the repeater transitions to the *RECEIVE COLLISION* state (described later). This behavior is known as fragment extension.

After the packet data has been repeated, including the emptying of the RICs' elasticity buffers, the RIC performs the *Tw1* transmit recovery operation. This is performed during the *WAIT* state shown in the repeater state diagram.

5.0 Functional Description (Continued)

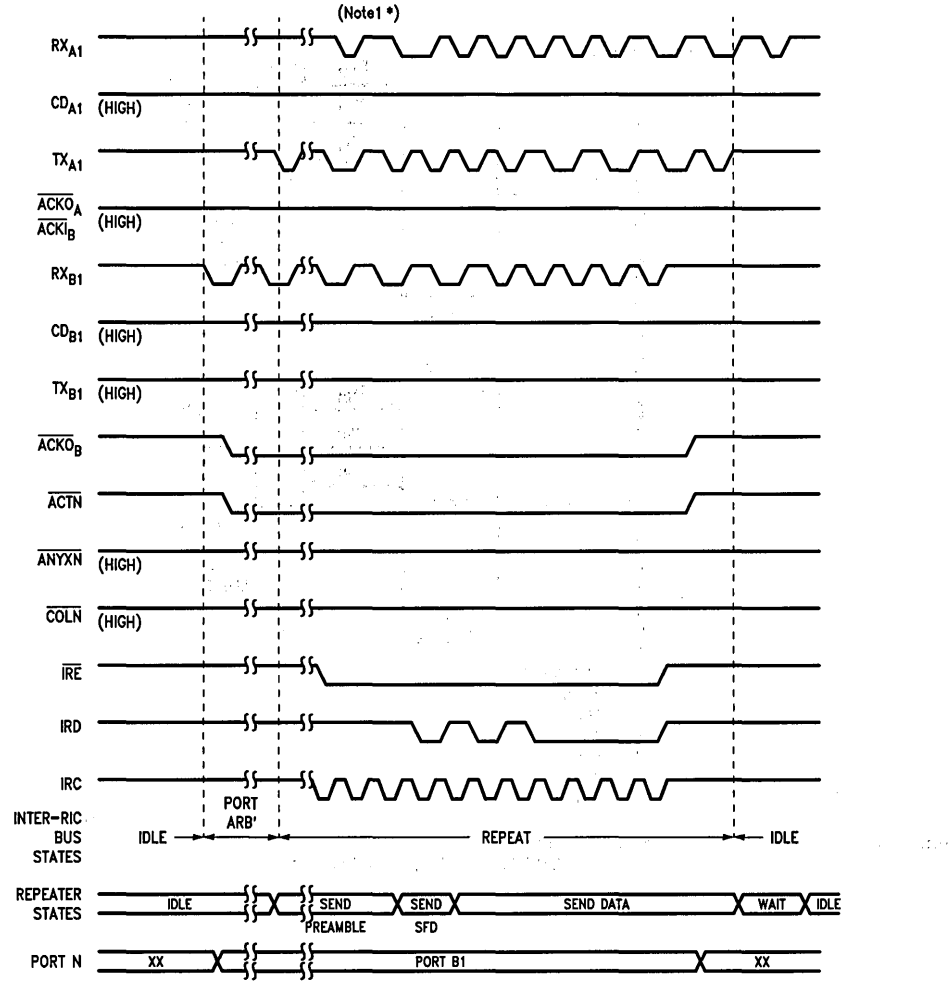


TL/F/11096–9

Note: In this example the Inter-RIC bus is configured to use active low signals.

FIGURE 5.4. RIC System Topology

5.0 Functional Description (Continued)



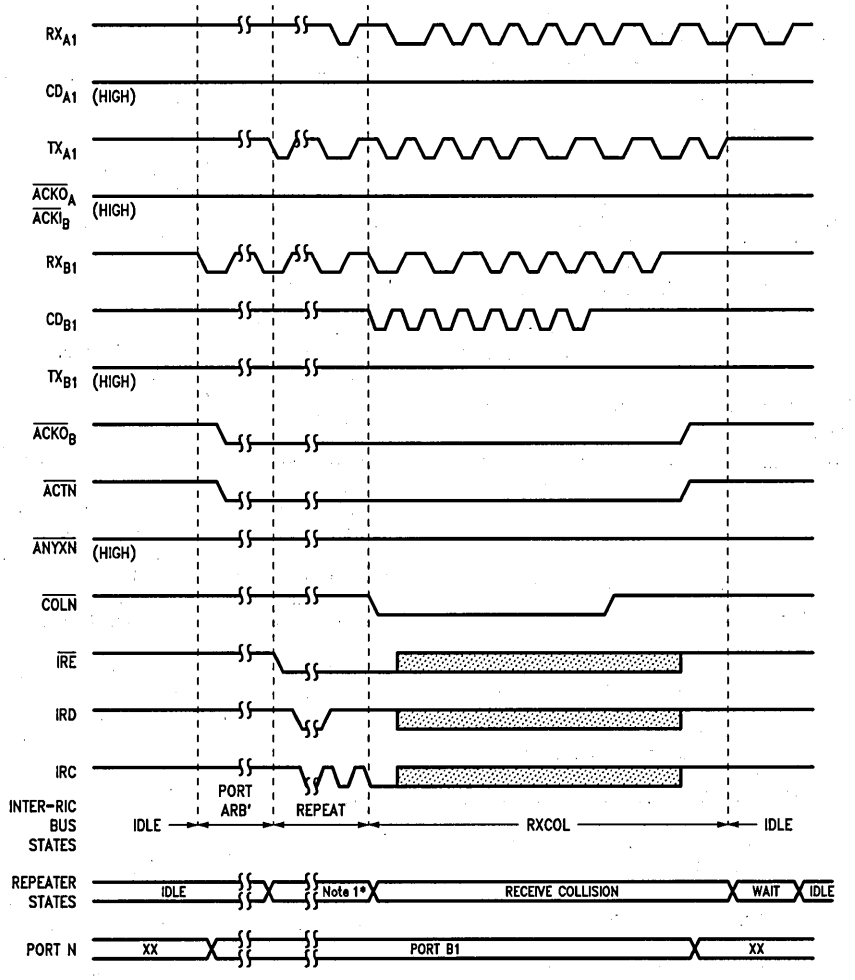
TL/F/11096–10

*Note 1: The activity shown in RX_{A1} represents the transmitted signal on TX_{A1} after being looped back by the attached transceiver.

Note: In this example the Inter-RIC bus is configured to use active low signals.

FIGURE 5.5. Data Repetition

5.0 Functional Description (Continued)



TL/F/11096–11

*Note 1: *SEND PREAMBLE, SEND SFD, SEND DATA.*

Note: In this example the Inter-RIC bus is configured to use active low signals.

FIGURE 5.6. Receive Collision

5.0 Functional Description (Continued)

Receive Collisions

A receive collision is a collision which occurs on the network segment attached to *PORT N*, i.e., the collision is "received" in a similar manner as a data packet is received and then repeated to the other network segments. Not surprisingly receive collision propagation follows a similar sequence of operations as is found with data repetition:

An arbitration process is performed to find *PORT N* and a preamble/jam pattern is transmitted by the repeater's other ports. When *PORT N* detects a collision on its segment the COLN Inter-RIC bus signal is asserted. This forces all the RICs in the system to transmit a preamble/jam pattern to their segments. This is important since they may be already transmitting data from their elasticity buffers. The repeater moves to the *RECEIVE COLLISION* state when the RICs begin to transmit the jam pattern. The repeater remains in this state until both the following conditions have been fulfilled:

1. At least 96 bits have been transmitted onto the network,
2. The activity has ended.

Under close examination the repeater specification reveals that the actual end of activity has its own permutations of conditions:

1. Collision and receive data signals may end simultaneously,
2. Receive data may appear to end before collision signals,
3. Receive data may continue for some time after the end of the collision signal.

Network segments using coaxial media may experience spurious gaps in segment activity when the collision signal goes inactive. This arises from the inter-action between the receive and collision signal squelch circuits, implemented in coaxial transceivers, and the properties of the coaxial cable itself. The repeater specification avoids propagation of these activity gaps by extending collision activity by the *Tw2* wait time. Jam pattern transmission must be sustained throughout this period. After this, the repeater will move to the *WAIT* state unless there is a data signal being received by *PORT N*.

The functional timing diagram, *Figure 5.6*, shows the operation of a repeater system during a receive collision. The system configuration is the same as earlier described and is shown in *Figure 5.4*.

The RICs perform the same *PORT N* arbitration and data repetition operations as previously described. The system is notified of the receive collision on port B1 by the COLN bus signal going active. This is the signal which informs the main state machines to output the jam pattern rather than the data held in the elasticity buffers. Once a collision has occurred the IRC, IRD AND $\overline{\text{IRE}}$ bus signals may become undefined. When the collision has ended and the *Tw2* operation performed, the repeater moves to the *WAIT state.*

Transmit Collisions

A transmit collision is a collision that is detected upon a segment to which the repeater system is transmitting. The port state machine monitoring the colliding segment asserts the ANYXN bus signal. The assertion of ANYXN causes *PORT M* arbitration to begin. The repeater moves to the *TRANSMIT COLLISION* state when the port which has been *PORT N* starts to transmit a Manchester encoded 1 on to its network segment. Whilst in the *TRANSMIT COLLISION* state all ports of the repeater must transmit the 1010 . . . jam pattern and *PORT M* arbitration is performed. Each RIC is obliged, by the IEEE specification, to ensure all of its ports transmit for at least 96 bits once the *TRANSMIT COLLISION* state has been entered. This transmit activity is enforced by the ANYXN bus signal. Whilst ANYXN is active all RIC ports will transmit jam. To ensure this situation lasts for at least 96 bits, the MSMs inside the RICs assert the ANYXN signal throughout this period. After this period has elapsed, ANYXN will only be asserted if there are multiple ports with active collisions on their network segments.

There are two possible ways for a repeater to leave the *TRANSMIT COLLISION* state. The most straight forward is when network activity, i.e., collisions and their *Tw2* extensions, end before the 96 bit enforced period expires. Under these conditions the repeater system may move directly to the *WAIT* state when 96 bits have been transmitted to all ports. If the MSM enforced period ends and there is still one port experiencing a collision the *ONE PORT LEFT* state is entered. This may be seen on the Inter-RIC bus when ANYXN is deasserted and *PORT M* stops transmitting to its network segment. In this circumstance the Inter-RIC bus transitions to the RECEIVE COLLISION state. The repeater will remain in this state whilst *PORT M's* collision, *Tw2* collision extension and any receive signals are present. When these conditions are not true, packet repetition finishes and the repeater enters the *WAIT* state.

Figure 5.7 shows a multi-RIC system operating under transmit collision conditions. There are many different scenarios which may occur during a transmit collision, this figure illustrates one of these. The diagram begins with packet reception by port A1. Port B1 experiences a collision, since it is not *PORT N* it asserts ANYXN. This alerts the main state machines in the system to switch from data to jam pattern transmission.

Port A1 is also monitoring the ANYXN bus line. Its assertion forces A1 to relinquish its *PORT N* status, start transmitting, stop asserting ACTN and release its hold on the PSM arbitration signals ($\overline{\text{ACKO}}$ A and $\overline{\text{ACKI}}$ B). The first bit it transmit will be a Manchester encoded "1" in the jam pattern. Since port B1 is the only port with a collision it attains *PORT M* status and stops asserting ANYXN. It does however assert ACTN, and exert its presence upon the PSM arbitration chain (forces$\overline{\text{ACKO}}$ B low). The MSMs ensure that ANYXN stays active and thus force all of the ports, including *PORT M*, to transmit to their segments.

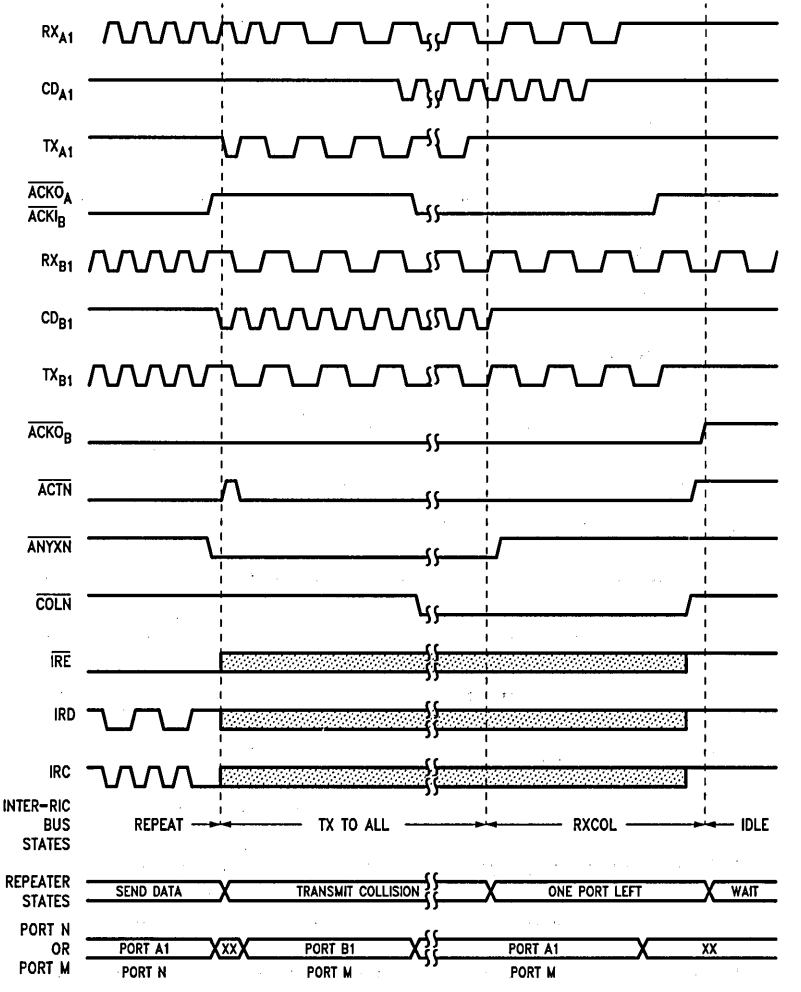
After some time port A1 experiences a collision. This arises from the presence of the packet being received from port A1's segment and the jam signal the repeater is now transmitting onto this segment. Two packets on one segment results in a collision. *PORT M* now moves from B1 to A1. Port A1 fulfills the same criteria as B1, i.e., it has an active collision on its segment, but in addition it is higher in the arbitration chain. This priority yields no benefits for port A1 since the ANYXN signal is still active. There are now two sources driving ANYXN, the MSMs and the collision on port B1.

Eventually the collision on port B1 ends and the ANYXN extension by the MSMs expires. There is only one collision

5.0 Functional Description (Continued)

on the network (this may be deduced since ANYXN is inactive) so the repeater will move to the *ONE PORT LEFT* state. The RIC system treats this state in a similar manner to a receive collision with *PORT M* fulfilling the role of the receiving port. The difference from a true receive collision is that the switch from packet data to the jam pattern has already been made (controlled by ANYXN). Thus the state of COLN has no effect upon repeater operations. In common with the operation of the *RECEIVE COLLISION* state, the repeater remains in this condition until the collision and receive activity on *PORT M* sibsides. The packet repetition operation completes when the *Tw1* recovery time in the *WAIT* state has been performed.

Note: In transmit collision conditions COLN will only go active if the RIC which contained *PORT N* at the start of packet repetition contains *PORT M* during the *TRANSMIT COLLISION* and *ONE PORT LEFT* states.



TL/F/11096–12

Note: In this example the Inter-RIC bus is configured to use active low signals.

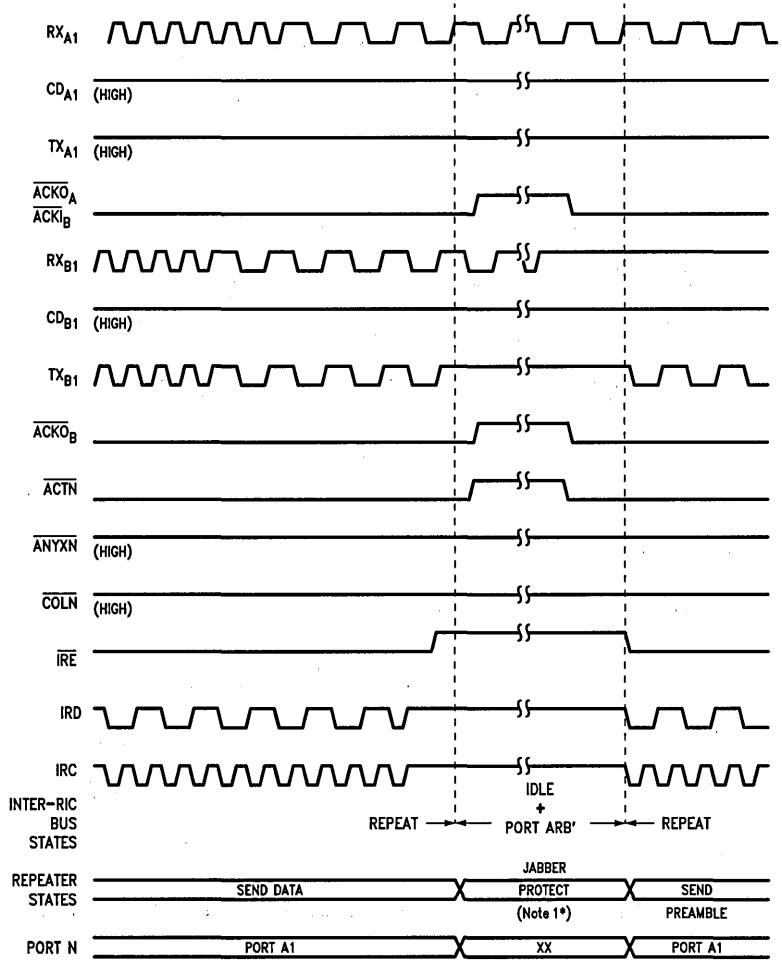
FIGURE 5.7. Transmit Collision

5.0 Functional Description (Continued)

Jabber Protection

A repeater is required to disable transmit activity if the length of its current transmission reaches the jabber protect limit. This is defined by the specification's *Tw3* time. The repeater disables output for a time period defined by the *Tw4* specification, after this period normal operation may resume.

Figure 5.8 shows the effect of a jabber length packet upon a RIC based repeater system. The **JABBER PROTECT** state is entered from the *SEND DATA* state. While the *Tw4* period is observed the Inter-RIC bus displays the IDLE state. This is misleading since new packet activity or continuous activity (as shown in the diagram) does not result in packet repetition. This may only occur when the *Tw4* requirement has been satisfied.



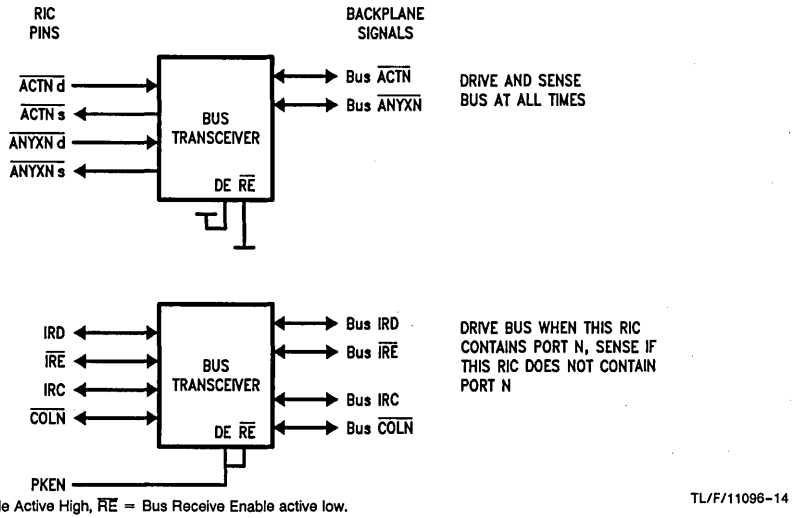
TL/F/11096–13

***Note 1:** The IEEE Specification does not have a jabber protect state defined in its main state diagram, this behaviour is defined in an additional MAU Jabber Lockup Protection state diagram.

Note: In this example the Inter-RIC bus is configured to use active low signals.

FIGURE 5.8. Jabber Protect

5.0 Functional Description (Continued)

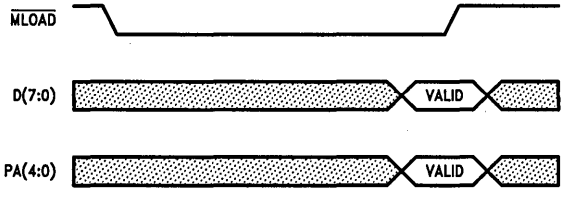


TL/F/11096–14

Note: DE = Bus Drive Enable Active High, $\overline{RE}$ = Bus Receive Enable active low.

Note: In this example the Inter-RIC bus is shown as using active low signals.

FIGURE 5.9. External Bus Transceiver Connection Diagram



TL/F/11096–15

FIGURE 5.10. Mode Load Operation

5.0 Functional Description (Continued)

5.4 DESCRIPTION OF HARDWARE CONNECTION FOR INTER-RIC BUS

When considering the hardware interface the Inter-RIC bus may be viewed as consisting of three groups of signals:

1. Port Arbitration chain, namely: $\overline{ACKI}$ and $\overline{ACKO}$.
2. Simultaneous drive and sense signals, i.e., ACTN and ANYXN. (Potentially these signals may be driven by multiple devices).
3. Drive or sense signals, i.e., IRE, IRD, IRC and COLN. (Only one device asserts these signals at any instance in time.)

The first set of signals are either used as point to point links or with external arbitration logic. In both cases the load on these signals will not be large so that the on-chip drivers are adequate. This may not be true for signal classes (2) and (3).

The Inter-RIC bus has been designed to connect RICs together directly or via external bus transceivers. The latter is advantageous in large repeaters. In the second application the backplane is often heavily loaded and is beyond the drive capability of the on-chip bus drivers. The need for simultaneous sense and drive capabilities on the ACTN and ANYXN signals and the desire to allow operation with external bus transceivers makes it necessary for these bus signals to each have a pair of pins on the RIC. One driving the bus the other sensing the bus signal. When external bus transceivers are used they must be open collector/open drain to allow wire-ORing of the signals. Additionally, the drive and sense enables of the bus transceiver should be tied in the active state.

When the RIC is used in a stand alone configuration, it is required to tie $ACTN_D$ to $ACTN_S$ and $ANYXN_D$ to $ANYXN_S$.

The uni-directional nature of information transfer on the IRE, IRD, IRC and COLN signals, means a RIC is either driving these signals or receivng them from the bus but not both at the same time. Thus a single bi-directional input/output pin is adequate for each of these signals. In an external bus transceiver is used with these signals the Packet Enable "PKEN" RIC output pin performs the function of a drive enable and sense disable.

Figure 5.9 shows the RIC connected to the Inter-RIC bus via external bus transceivers, such as National's DS3893A bus transceivers.

Some bus transceivers are of the inverting type. To allow the Inter-RIC bus to utilize these transceivers the RIC may be configured to invert the active states of the ACTN, ANYXN, COLN and IRE signals. Instead of being active low they are active high.

Thus they become active low once more when passed through an inverting bus driver. This is particularly important for the ACTN and ANYXN bus lines, since these signals must be used in a wired-or configuration. Incorrect signal polarity would make the bus unusable.

5.5 PROCESSOR AND DISPLAY INTERFACE

The processor interface pins, which include the data bus, address bus and control signals, actually perform three operations which are multiplexed on these pins. These operations are:

1. The Mode Load Operation, which performs a power up initialization cycle upon the RIC.
2. Display Update Cycles, which are refresh operations for updating the display LEDs.
3. Processor Access Cycles, which allows μP's to communicate with the RIC's registers.

These three operations are described below.

Mode Load Operation

The Mode Load Operation is a hardware initialization procedure performed at power on. It loads vital device configuration information into on-chip configuration registers. In addition to its configuration function the $\overline{MLOAD}$ pin is the RIC's reset input. When $\overline{MLOAD}$ is low all of the RIC's repeater timers, state machines, segment partition logic and hub management logic are reset.

The Mode Load Operation may be accomplished by attaching the appropriate set of pull up and pull down resistors to the data and register address pins to assert logic high or low signals onto these pins, and the providing a rising edge on the $\overline{MLOAD}$ pin as is shown in *Figure 5.10*. The mapping of chip functions to the configuration inputs is shown in Table 5.1. Such an arrangement may be performed using a simple resistor, capacitor, diode network. Performing the Mode Load Operation in this way enables the configuration of a RIC that is in a simple repeater system (one without a processor).

Alternatively in a complex repeater system, the Mode Load Operation may be performed using a processor write cycle. This would require the $\overline{MLOAD}$ pin be connected to the CPU's write strobe via some decoding logic, and included in the processor's memory map.

5.0 Functional Description (Continued)

TABLE 5.1. Pin Definitions for Options in the Mode Load Operation

Pin Name	Programming Function	Effect When Bit is 0	Effect When Bit is 1	Function
D0	resv	Not Permitted	Required	To ensure correct device operation, this bit must be written with a logic one during the mode load operation.
D1	tw2	5 bits	3 bits	This allows the user to select one of two values for the repeater specification tw2 time. The lower limit (3 bits) meets the IEEE specification. The upper limit (5 bits) is not specification compliant but may provide users with higher network throughput by avoiding spurious network activity gaps when using coaxial (10BASE2, 10BASE5) network segments.
D2	$\overline{\text{CCLIM}}$	63	31	The partition specification requires a port to be partitioned after a certain number of consecutive collisions. The RIC has two values available to allow users to customize the partitioning algorithm to their environment. Please refer to the Partition State Machine, in data sheet Section 7.3.
D3	$\overline{\text{LPPART}}$	Selected	Not Selected	The RIC may be configured to partition a port if the segment transceiver does not loopback data to the port when the port is transmitting to it, as described in the Partition State Machine.
D4	$\overline{\text{OWCE}}$	Selected	Not Selected	This configuration bit allows the on-chip partition algorithm to include out of window collisions into the collisions it monitors, as described in the Partition State Machine.
D5	$\overline{\text{TXONLY}}$	Selected	Not Selected	This configuration bit allows the on-chip partition algorithm to restrict segment reconnection, as described in the Partition State Machine.
D6	$\overline{\text{DPART}}$	Selected	Not Selected	The Partition state machines for all ports may be disabled by writing a logic zero to this bit during the mode load operation.
D7	MIN/MAX	Minimum Mode	Maximum Mode	The operation of the display update block is controlled by the value of this configuration bit, as described in the Display Update Cycles section.

5.0 Functional Description (Continued)

TABLE 5.1 Pin Definitions for Options in the Mode Load Operation (Continued)

<table>
<tr><th>Pin Name</th><th>Programming Function</th><th>Effect When Bit is 0</th><th>Effect When Bit is 1</th><th>Function</th></tr>
<tr><td>RA0

RA1</td><td>BYPAS1

BYPAS2</td><td></td><td></td><td>These configuration bits select which of the repeater ports (numbers 2 to 13) are configured to use the on-chip internal 10BASE-T transceivers or the external transceiver interface. The external transceiver interface operates using AUI compatible signal levels.
<table>
<tr><th>BYPAS2</th><th>BYPAS1</th><th>Information</th></tr>
<tr><td>0</td><td>0</td><td>All ports (2 to 13) use the external Transceiver Interface.</td></tr>
<tr><td>0</td><td>1</td><td>Ports 2 to 7 use the external interface, 8 to 13 use the internal 10BASE-T transceivers.</td></tr>
<tr><td>1</td><td>0</td><td>Ports 2 to 5 use the external interface, 6 to 13 use the internal 10BASE-T transceivers.</td></tr>
<tr><td>1</td><td>1</td><td>All ports (2 to 13) use the internal 10BASE-T transceivers.</td></tr>
</table></td></tr>
<tr><td>RA2</td><td>BINV</td><td>Active High Signals</td><td>Active Low Signals</td><td>This selection determines whether the Inter-RIC signals: IRE, ACTN, ANYXN, COLN and Management bus signal MCRS are active high or low.</td></tr>
<tr><td>RA3</td><td>EXPLL</td><td>External PLL</td><td>Internal PLL</td><td>If desired, the RIC may be used with an external decoder, this configuration bit performs the selection.</td></tr>
<tr><td>RA4</td><td>resv</td><td>Not Permitted</td><td>Required</td><td>To ensure correct device operation, this bit must be written with a logic one during the mode load operation.</td></tr>
</table>

5.0 Functional Description (Continued)

5.6 DESCRIPTION OF HARDWARE CONNECTION FOR PROCESSOR AND DISPLAY INTERFACE

Display Update Cycles

The RIC possesses control logic and interface pins which may be used to provide status information concerning activity on the attached network segments and the current status of repeater functions. These status cycles are completely autonomous and require only simple support circuitry to produce the data in a form suitable for a light emitting diode "LED" display. The display may be used in one of two modes:

1. Minimum Mode: General Repeater Status LEDs

2. Maximum Mode: Individual Port Status LEDs

Minimum mode, intended for simple LED displays, makes available four status indicators. The first LED denotes whether the RIC has been forced to activate its jabber protect functions. The remaining 3 LEDs indicate if any of the RIC's network segments are: (1) experiencing a collision, (2) receiving data, (3) currently partitioned. When minimum display mode is selected the only external components required are a 74LS374 type latch, the LEDs and their current limiting resistors.

Maximum mode differs from minimum mode by providing display information specific to individual network segments. This information denotes the collision activity, packet reception and partition status of each segment. In the case of 10BASE-T segments the link integrity status and polarity of the received data are also made available. The wide variety of information available in maximum mode may be used in its entirety or in part. Thus allowing the system designer to choose the appropriate complexity of status display commensurate with the specification of the end equipment.

The signals provided and their timing relationships have been designed to interface directly with 74LS259 type addressable latches. The number of latches used being dependant upon the complexity of the display. Since the latches are octal, a pair of latches is needed to display each type of segment specific data (13 ports means 13 latch bits). The accompanying tables (5.1 and 5.2) show the function of the interface pins in minimum and maximum modes. *Figure 5.12* shows the location of each port's status information when maximum mode is selected. This may be compared with the connection diagram *Figure 5.11*.

Immediately following the Mode Load Operation (when the $\overline{\text{MLOAD}}$ pin transitions to a high logic state), the display logic performs an LED test operation. This operation lasts one second and while it is in effect all of the utilized LEDs will blink on. Thus an installation engineer is able to test the operation of the display by forcing the RIC into a reset cycle ($\overline{\text{MLOAD}}$ forced low). The rising edge on the $\overline{\text{MLOAD}}$ pin starts the LED test cycle. **During the LED test cycle the RIC does not perform packet repetition operations.**

The status display possesses a capability to lengthen the time an LED is active. At the end of the repetition of a packet, the display is frozen showing the current activity. This freezing lasts for 30 milliseconds or until a subsequent packet is repeated. Thus at low levels of packet activity the display stretches activity information to make it discernable to the human eye. At high traffic rates the relative brightness of the LEDs indicates those segments with high or low activity.

TABLE 5.2. Status Display Pin Functions in Minimum Mode

Signal Pin Name	Function in MINIMUM MODE
D0	No operation
D1	Provides status information indicating if there is a collision occurring on one of the segments attached to this RIC.
D2	Provides status information indicating if one of this RIC's ports is receiving a data or collision packet from a segment attached to this RIC.
D3	Provides status information indicating that the RIC has experienced a jabber protect condition.
D4	Provides Status information indicating if one of the RIC's segments is partitioned.
D(7:5)	No operation
$\overline{\text{STR0}}$	This signal is the latch enable for the 374 type latch.
$\overline{\text{STR1}}$	This signal is held at a logic one.

5.0 Functional Description (Continued)

Table 5.3 Status Display Pin Functions in MAXIMUM MODE

Signal Pin Name	Function in Maximum Mode
D0	Provides status information concerning the Link Integrity status of 10BASE-T segments. This signal should be connected to the data inputs of the chosen pair of 74LS259 latches.
D1	Provides status information indicating if there is a collision occurring on one of the segments attached to this RIC. This signal should be connected to the data inputs of the chosen pair of 74LS259 latches.
D2	Provides status information indicating if one of this RIC's ports is receiving a data or a collision packet from its segment. This signal should be connected to the data inputs of the chosen pair of 74LS259 latches.
D3	Provides Status information indicating that the RIC has experienced a jabber protect condition. Additionally it denotes which of its ports are partitioned. This signal should be connected to the data inputs of the chosen pair of 74LS259 latches.
D4	Provides status information indicating if one of this RIC's ports is receiving data of inverse polarity. This status output is only valid if the port is configured to use its internal 10BASE-T transceiver. The signal should be connected to the data inputs of the chosen pair of 74LS259 latches.
D(7:5)	These signals provide the repeater port address corresponding to the data available on D(4:0).
$\overline{STR0}$	This signal is the latch enable for the lower byte latches, that is the 74LS259s which display information concerning ports 1 to 7.
$\overline{STR1}$	This signal is the latch enable for the upper byte latches, that is the 74LS259s which display information concerning ports 8 to 13.

Maximum Mode LED Definitions

74LS259 Latch Inputs = $\overline{STR0}$

259 Output	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
259 Addr S2–0	000	001	010	011	100	101	110	111
RIC Port Number		1 (AUI)	2	3	4	5	6	7
RIC D0 259 #1			LINK	LINK	LINK	LINK	LINK	LINK
RIC D1 259 #2	ACOL	COL	COL	COL	COL	COL	COL	COL
RIC D2 259 #3	AREC	REC	REC	REC	REC	REC	REC	REC
RIC D3 259 #4	JAB	PART	PART	PART	PART	PART	PART	PART
RIC D4 259 #5			BDPOL	BDPOL	BDPOL	BDPOL	BDPOL	BDPOL

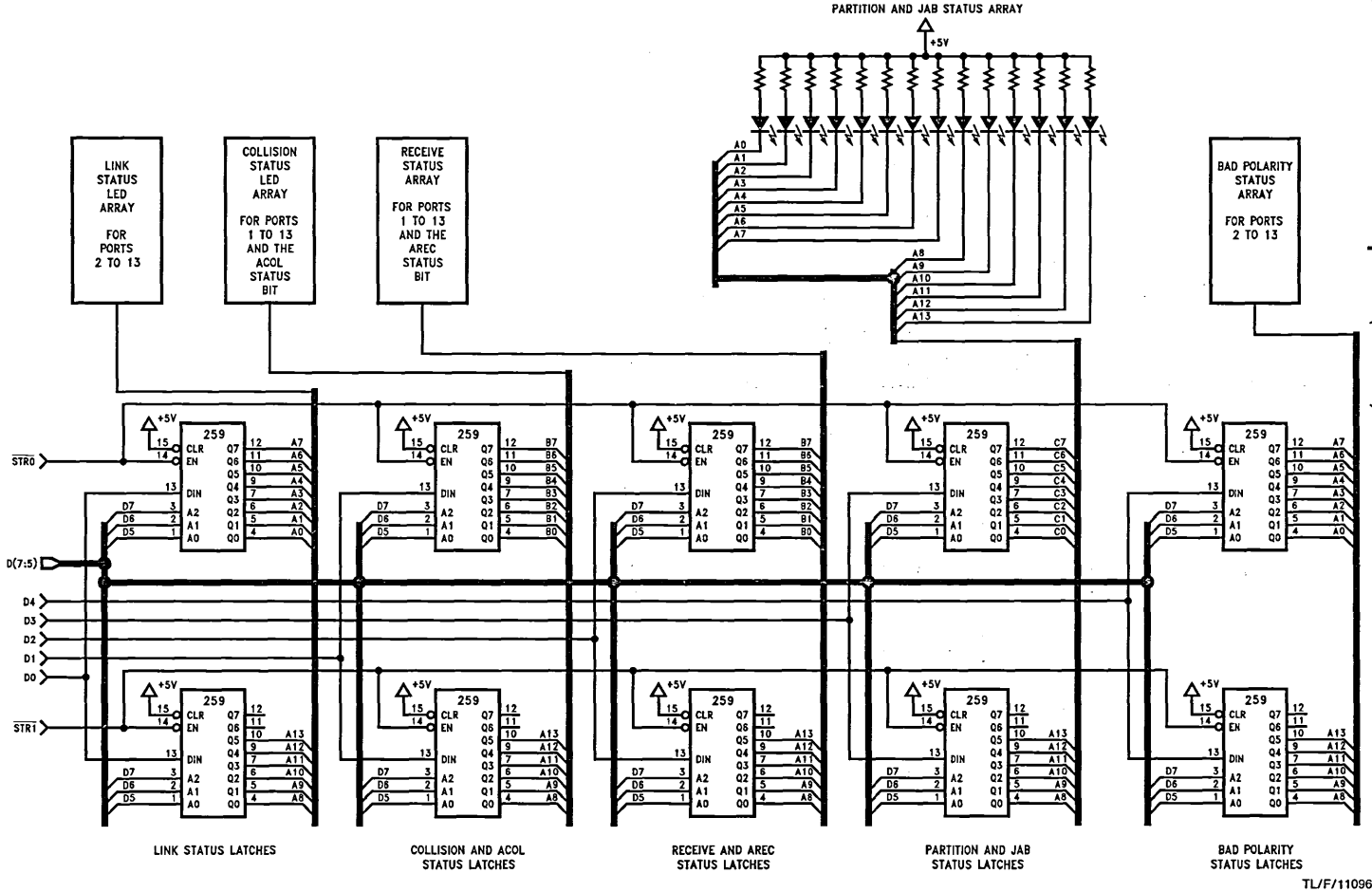
74LS259 (or Equiv.) Latch Inputs = $\overline{STR1}$

259 Output	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
259 Addr S2–0	000	001	010	011	100	101	110	111
RIC Port Number	8	9	10	11	12	13		
RIC D0 259 #6	LINK	LINK	LINK	LINK	LINK	LINK		
RIC D1 259 #7	COL	COL	COL	COL	COL	COL		
RIC D2 259 #8	REC	REC	REC	REC	REC	REC		
RIC D3 259 #9	PART	PART	PART	PART	PART	PART		
RIC D4 259 #10	BDPOL	BDPOL	BDPOL	BDPOL	BDPOL	BDPOL		

This shows the LED Output Functions for the LED Drivers when 74LS259s are used. The top table refers to the bank of 4 74LS259s latched with $\overline{STR0}$, and the lower table refers to the bank of 4 74LS259s latched with $\overline{STR1}$. For example the RIC's D0 data signal goes to 259 #1 and #5. These two 74LS259s then drive the LINK LEDs).

Note: ACOL = Any Port Collision, AREC = Any Port Reception, JAB = Any Port Jabbering, LINK = Port Link, COL = Port Collision, REC = Port Reception, PART = Port Partitioned, BDPOL = Bad (inverse) Polarity or received data.

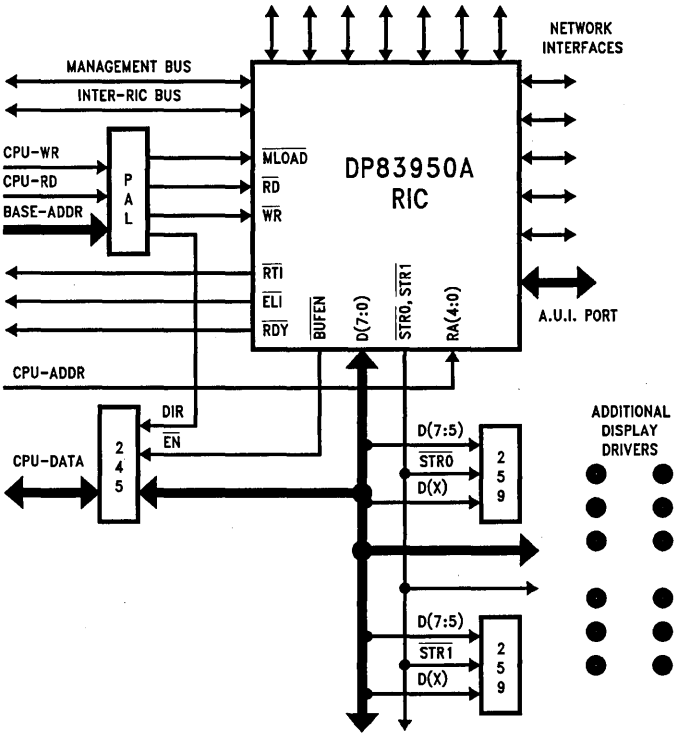
FIGURE 5.12



TL/F/11096–16

FIGURE 5.11. Maximum Mode LED Display (All Available Status Bits Used)

5.0 Functional Description (Continued)



TL/F/11096–17

FIGURE 5.13. Processor Connection Diagram

5.0 Functional Description (Continued)

Processor Access Cycles

Access to the RIC's on-chip registers is made via its processor interface. This utilizes conventional non-multiplexed address (five bit) and data (eight bit) busses. The data bus is also used to provide data and address information to off chip display latches during display update cycles. While performing these cycles the RIC behaves as a master of its data bus. Consequently a TRI-STATE bi-directional bus transceiver, e.g., 74LS245 must be placed between the RIC and any processor bus.

The processor requests a register access by asserting the read "$\overline{RD}$" or write "$\overline{WR}$" input strobes. The RIC responds by finishing any current display update cycle and asserts the tri-state buffer enable signal "$\overline{BUFFEN}$". If the processor cycle is a write cycle then the RIC's data buffers are disabled to prevent contention. In order to interface to the RIC in a processor controlled system it is likely a PAL device will be used to perform the following operations:

1. Locate the RIC in the processor's memory map (address decode),
2. Generate the RIC's read and write strobes,
3. Control the direction signal for the 74LS245.

An example of the processor and display interfaces is shown in *Figure 5.13*.

6.0 Hub Management Support

The RIC provides information regarding the status of its ports and the packets it is repeating. This data is available in three forms:

1. Counted Events—Network events accumulated into the RIC's 16-bit Event Counter Registers.
2. Recorded Events—Network events that set bits in the Event Record Registers.
3. Hub Management Status Packets—This is information sent over the Management Bus in a serial function to be decoded by an Ethernet Controller board.

The counted and recorded event information is available through the processor interface. This data is port specific and may be used to generate interrupts via the Event Logging Interrupt "$\overline{ELI}$" pin. Since the information is specific to each port, each repeater port has its own event record register and event counter. The counters and event record registers have user definable masks which enable them to be configured to count and record a variety of events. The counters and record registers are designed to be used together so that detailed information, i.e., a count value can be held on-chip for a specific network condition, and more general information, i.e., certain types of events have occurred, may be retained in on-chip latches. Thus the user may configure the counters to increment upon a rapidly occurring event (most likely to be used to count collisions), and the record registers may log the occurrence of less frequent error conditions such as jabber protect packets.

6.1 EVENT COUNTING FUNCTION

The counters may increment upon the occurrence of one of the categories of event as described below.

Potential sources for Counter increment:

Jabber Protection (JAB): The port counter increments if the length of a received packet from its associated port, causes the repeater state machine to enter the jabber protect state.

Elasticity Buffer Error (ELBER): The port counter increments if a Elasticity Buffer underflow or overflow occurs during packet reception. The flag is held inactive if a collision occurs during packet reception or if a phase lock error, described below, has already occurred during the repetition of the packet.

Phase Lock Error (PLER): A phase lock error is caused if the phase lock loop decoder looses lock during packet reception. Phase lock onto the received data stream may or may not be recovered later in the packet and data errors may have occurred. This flag is held inactive if a collision occurs.

Non SFD Packet (NSFD): If a packet is received and the start of frame delimiter is not found, the port counter will increment. Counting is inhibited if the packet suffers a collision.

Out of Window Collision (OWC): The out of window collision flag for a port goes active when a collision is experienced outside of the network slot time.

Transmit Collision (TXCOL): The transmit collision flag for a port is enabled when a transmit collision is experienced by the repeater. Each port experiencing a collision under these conditions is said to have suffered a transmit collision.

Receive Collision (RXCOL): The receive collision flag for a port goes active when the port is the receive source of network activity and suffers a collision, provided no other network segments experience collision then the receive collision flag for the receiving port will be set.

Partition (PART): The port counter increments when a port becomes partitioned.

Bad Link (BDLNK): The port counter increments when a port is configured for 10BASE-T operation has entered the link lost state.

Short Event reception (SE): The port counter increments if the received packet is less than 74 bits long and no collision occurs during reception.

Packet Reception (REC): When a packet is received the port counter increments.

In order to utilize the counters the user must choose, from the above list, the desired statistic for counting. This counter mask information must be written to the appropriate, Event Count Mask Register. There are two of these registers, the Upper and Lower, Event Count Mask registers. For the exact bit patterns of these registers please see Section 8 of the data sheet.

For example if the counters are configured to count network collisions and the appropriate masks have been set, then whenever a collision occurs on a segment, this information is latched by the hub management support logic. At the end of repetition of the packet the collision status, respective to each port, is loaded into that port's counter. This operation is completely autonomous and requires no processor intervention.

6.0 Hub Management Support (Continued)

Each counter is 16 bits long and may be directly read by the processor. Additionally each counter has a number of decodes to indicate the current value of the count. There are three decodes:

Low Count (a value of 00FF Hex and under),
High Count (a value of C000 Hex and above),
Full Count (a value of FFFF Hex).

The decodes from each counter are logically "ORed" together and may be used as interrupt sources for the $\overline{ELI}$ interrupt pin. Additionally the status of these bits may be observed by reading the Page Select Register (PSR), (see Section 8 for register details). In order to enable any of these threshold interrupts, the appropriate interrupt mask bit must be written to the Management and Interrupt Configuration Register; see Section 8 for register details.

In addition to their event masking functions the Upper Event Counting Mask Register (UECMR) possesses two bits which control the operation of the counters. When written to a logic one, the reset on read bit "ROR" resets the counter after a processor read cycle is performed. If this operation is not selected then in order to zero the counters they must either be written with zeros by the processor or allowed to roll over to all zeros. The freeze when full bit "FWF" prevents counter roll over by inhibiting count up cycles (these happen when chosen events occur), thus freezing the particular counter at FFFF Hex.

The port event counters may also be controlled by the Counter Decrement ($\overline{CDEC}$) pin. As its name suggests a logic low state on this pin will decrement all the counters by a single value. The pulses on $\overline{CDEC}$ are internally synchronized and scheduled so as not to conflict with any "up counting" activity. If an up count and a down count occur simultaneously then the down count is delayed until the up count has completed. This combination of up and down counting capability enables the RIC's on-chip counters to provide a simple rolling average or be used as extensions of larger off chip counters.

Note: If the FWF option is enabled then the count down operation is disabled from those registers which have reached FFFF Hex and consequently have been frozen. Thus, if FWF is set and $\overline{CDEC}$ has been employed to provide a rate indication. A frozen counter indicates that a rate has been detected which has gone out of bounds, i.e., too fast increment or too slow increment. If the low count and high count decodes are employed as either interrupt sources or in a polling cycle, the direction of the rate excursion may be determined.

Reading the Event Counters

The RIC's external data bus is eight bits wide, since the event counters are 16 bits long two processor read cycles are required to yield the counter value. In order to ensure that the read value is correct and to allow simultaneous event counts with processor accesses, a temporary holding register is employed. A read cycle to either the lower or upper byte of a counter, causes both bytes to be latched into the holding register. Thus when the other byte of the counter is obtained the holding register is accessed and not the actual counter register. This ensures that the upper and lower bytes contain the value sampled at the same instance in time, i.e., when the first read cycle to that counter occurred.

There is no restriction concerning whether the upper or lower byte is read first. However to ensure the "same instance value" is obtained, the reads of the upper then lower byte (or vice versa) should be performed as consecutive reads of the counter array. Other NON COUNTER registers may be read in between these read cycles and also write cycles may be performed. If another counter is read or the same byte of the original counter is read, then the holding register is updated from the counter array and the unread byte is lost.

If the reset on read option is employed then the counter is reset after the transfer to the holding register is performed. Processor read and write cycles are scheduled in such a manner that they do not conflict with count up or count down operations. That is to say, in the case of a processor read the count value is stable when it is loaded into the holding register. In the case of a processor write, the newly written value is stable so it maybe incremented or decrement by any subsequent count operation. During the period the $\overline{MLOAD}$ pin is low, (power on reset) all counters are reset to zero and all count masks are forced into the disabled state. Section 8 of the data sheet details the address location of the port event counters.

6.2 EVENT RECORD FUNCTION

As previously stated each repeater port has its own Event Recording Register. This is an 8-bit status register each bit is dedicated to logging the occurrence of a particular event (see Section 8 for detailed description). The logging of these events is controlled by the Event Recording Mask Register, for an event to be recorded the particular mask bit must be set, (see Section 8 description of this register). Similar to the scheme employed for the event counters, the recorded events are latched during the repetition of a packet and then automatically loaded into the recording registers at the end of transmission of a packet. When one of the unmasked events occurs, the particular port register bit is set. This status is visible to the user. All of the register bits for all of the ports are logically "ORed" together to produce a Flag Found "FF" signal. This indicator may be found by reading the Page Select Register. Additionally an interrupt may be generated if the appropriate mask bit is enabled in the Management and Interrupt Configuration Register.

A processor read cycle to a Event Record Register resets any of the bits set in that register. Read operations are scheduled to guarantee non changing data during a read cycle. Any internal bit setting event which immediately follows a processor read will be successful. The events which may be recorded are described below:

Jabber Protection (JAB): This flag goes active if the length of a received packet from the relevant port, causes the repeater state machine to enter the Jabber Protect state.

Elasticity Buffer Error (ELBER): This condition occurs if an Elasticity Buffer underflow or overflow occurs during packet reception. The flag is held inactive if a collision occurs during packet reception or if a phase lock error has already occurred during the repetition of the packet.

Phase Lock Error (PLER): A phase lock error is caused if the phase lock loop decoder loses lock during packet reception. Phase lock onto the received data stream may or may not be recovered later in the packet and data errors may have occurred. This flag is held inactive if a collision occurs.

Non SFD Packet (NSFD): If a packet is received and the start of frame delimiter is not found, the flag will go active. The flag is held inactive if a collision occurs in during packet repetition.

6.0 Hub Management Support (Continued)

Out of Window Collision (OWC): The out of window collision flag for a port goes active when a collision is experienced outside of the network slot time.

Partition (PART): This flag goes active when a port becomes partitioned.

Bad Link (BDLNK): The flag goes active when a port is configured for 10BASE-T operation has entered the link lost state.

Short Event reception (SE): This flag goes active if the received packet is less than 74 bits long and no collision occurs during reception.

6.3 MANAGEMENT INTERFACE OPERATION

The HUB Management interface provides a mechanism to combine repeater status information with packet information to form a hub management status packet. The interface, a serial bus consisting of carrier sense, received clock and received data, is designed to connect one or multiple RIC's over a backplane bus to a DP83932 "SONIC" network controller. The SONIC and the RICs form a powerful entity for network statistics gathering.

The interface consists of four pins:

MRXC	Management Receive Clock—10 MHz NRZ Clock output.
MCRS	Management Carrier Sense—Input/Output indicating of valid data stream.
MRXD	Management Receive Data—NRZ Data output synchronous to MRXC.
$\overline{\text{PCOMP}}$	Packet Compress—Input to truncate the packet's data field.

The first three signals mimic the interface between an Ethernet controller and a phase locked loop decoder (specifically the DP83932 SONIC and DP83910 SNI), these signals are driven by the RIC receiving the packet. MRXC and MRXD compose an NRZ serial data stream compatible with the DP83932. The $\overline{\text{PCOMP}}$ signal is driven by logic on the processor board. The actual data stream transferred over MRXD is derived from data transferred over the IRD Inter-RIC bus line. These two data streams differ in two important characteristics:

1. At the end of packet repetition a hub management status field is appended to the data stream. This status field, consisting of 7 bytes is shown in *Figure 6.1* and *6.2*. The information field is obtained from a number of packet status registers described below. In common with the 802.3 protocol the least significant bit of a byte is transmitted first.
2. While the data field of the repeated packet is being transferred over the management bus, received clock signals on the MRXC pin may be inhibited. This operation is under the control of the Packet Compress pin $\overline{\text{PCOMP}}$. If $\overline{\text{PCOMP}}$ is asserted during repetition of the packet then MRXC signals are inhibited when the number of bytes (after SFD) transferred over the management bus equals the number indicated in the Packet Compress Decode Register. This register provides a means to delay the effect of the $\overline{\text{PCOMP}}$ signal, which may be generated early in the packet's repetition, until the desired moment. Packet compression may be used to reduce the amount of memory required to buffer packets when they are received and are waiting to be processed by hub management software. In this kind of application an address decoder, which forms part of the packet compress logic, would monitor the address fields as they are received over the management bus. If the destination address is not the address of the management node inside the hub, then packet compression could be employed. In this manner only the portion of the packet meaningful for hub management interrogation, i.e., the address fields, is transferred to the SONIC and is buffered in memory.

   If the repeated packet ends before $\overline{\text{PCOMP}}$ is asserted or before the required number of bytes have been transferred, then the hub management status field is directly appended to the received data at a byte boundary. If the repeated packet is significantly longer than the value in the Decode Register requires and $\overline{\text{PCOMP}}$ is asserted the status fields will be delayed until the end of packet repetition. During this delay period MRXC clocks are inhibited but the MCRS signal remains asserted.

Note: If $\overline{\text{PCOMP}}$ is asserted late in the packet, i.e., after the number of bytes defined by the packet compression register, then packet compression will not occur.

The Management Interface may be fine tuned to meet the timing consideration of the SONIC and the access time of its associated packet memory. This refinement may be performed in two ways:

1. The default mode of operation of the Management interface is to only transfer packets over the bus which have a start of frame delimiter. Thus "packets" that are only preamble/jam and do not convey any source or destination address information are inhibited. This filtering may be disabled by writing a logic zero to the Management Interface Configuration or "MIFCON" bit in the Management and Interrupt Configuration Register. See Section 8 for details.
2. The Management bus has been designed to accommodate situations of maximum network utilization, for example when collision generated fragments occur; (these collision fragments may violate the IEEE802.3 IFG specification). The IFG required by the SONIC is a function of the time taken to release space in the receive FIFO and to perform end of packet processing (write status information into memory). These functions are primarily memory operations and consequently depend upon the bus latency and the memory access time of the system. In order to allow the system designer some discretion in choosing the speed of this memory, the RIC may be configured to protect the SONIC from a potential FIFO overflow. This is performed by utilizing the Inter Frame Gap Threshold Select Register.

   The value held in this register, plus one, defines, in network bit times, the minimum allowed gap between frames on the management bus. If the gap is smaller than this number then MCRS is asserted but MRXC clocks are inhibited. Consequently no data transfer is performed.

   Thus the system designer may make the decision whether to gather statistics on all packets even if they occur with very small IFGs or to monitor a subset.

The status field, shown in *Figure 6.1*, contains information which may be conveniently analyzed by considering it as

6.0 Hub Management Support (Continued)

providing information of six different types. They are held in seven Packet Status Registers "PSRs":

1. The RIC and port address fields [PSR(0) and (1)] can uniquely identify the repeater port receiving the packet out of a potential maximum of 832 ports sharing the same management bus (64 RICs each with 13 ports). Thus all of the other status fields can be correctly attributed to the relevant port.
2. The status flags the RIC produces for the event counters or recording latches are supplied with each packet [PSR(2)]. Additionally the clean receive CLN status is supplied to allow the user to determine the reliability of the address fields in the packet. The CLN status bit [PSR(1)] is set if no collisions are experienced during the repetition of the address fields.
3. The RIC has an on-chip timer to indicate when, relative to the start of packet repetition, a collision, if any, occurred [PSR(3)]. There is also a timer which indicates how many bit times of IFG was seen on the network between repetition of this packet and the preceding one. This is provided by [PSR(6)].
4. If packet compression is employed, the receive byte count contained in the SONIC's packet descriptor will indicate the number of bytes transferred over the management bus rather than the number of bytes in the packet. For this reason the RIC which receives the packet, counts the number of received bytes and transfers this over the management bus [PSR(4), (5)].
5. Appending a status field to a data packet will obviously result in a CRC error being flagged by the SONIC. For this reason the RIC monitors the repeated data stream to check for CRC and FAE errors. In the case of FAE errors the RIC provides additional dummy data bits, so that the status fields are always byte aligned.
6. As a final check upon the effectiveness of the management interface, the RIC transfers a bus specific status bit to the SONIC. This flag Packet Compress Done PCOMPD [PSR(0)], may be monitored by hub management software to check if the packet compression operation is enabled.

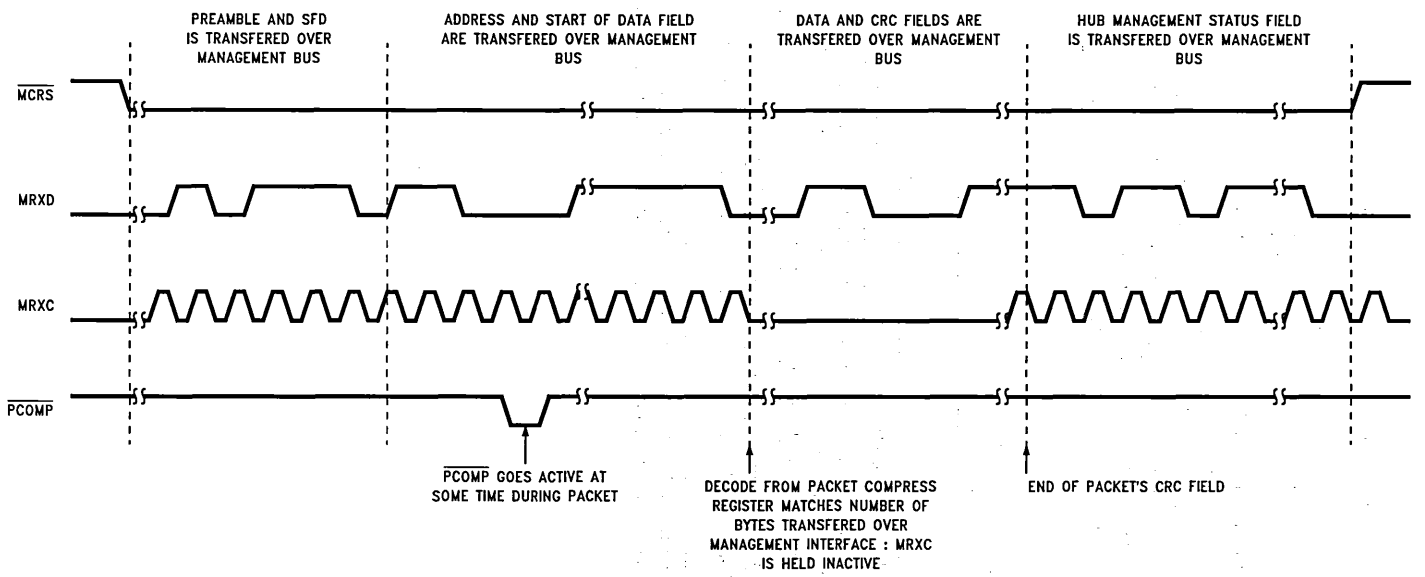
Figure 6.2 shows an example of a packet being transmitted over the management bus. The first section of the diagram (moving from left to right) shows a short preamble and SFD pattern. The second region contains the packet's address and the start of the data fields. During this time logic on the processor/SONIC card would determine if packet compression should be used on this packet. The $\overline{\text{PCOMP}}$ signal is asserted and packet transfer stops when the number of bytes transmitted equals the value defined in the decode register. Hence the MRXC signal is idle for the remainder of the packet's data and CRC fields. The final region shows the transfer of the RIC's seven bytes of packet status.

The following pages describe these Hub Management registers which constitute the management status field.

Packet Status Register PSR	D7	D6	D5	D4	D3	D2	D1	D0
PSR(0)	A5	A4	A3	A2	A1	A0	PCOMPD	resv
PSR(1)	CRCER	FAE	COL	CLN	PA3	PA2	PA1	PA0
PSR(2)	SE	OWC	NSFD	PLER	ELBER	JAB	CBT9	CBT8
PSR(3) Collision Bit Timer	CBT7	CBT6	CBT5	CBT4	CBT3	CBT2	CBT1	CBT0
PSR(4) Lower Repeat Byte Count	RBY7	RBY6	RBY5	RBY4	RBY3	RBY2	RBY1	RBY0
PSR(5) Upper Repeat Byte Count	RBY15	RBY14	RBY13	RBY12	RBY11	RBY10	RBY9	RBY8
PSR(6) Inter Frame Gap Bit Timer	IBT7	IBT6	IBT5	IBT4	IBT3	IBT2	IBT1	IBT0

Note: These registers may only be reliably accessed via the management interface. Due to the nature of these registers they may not be accessed (read or write cycles) via the processor interface.

FIGURE 6.1. Hub Management Status Field



Note: In this example the Management Bus is configured to use active low signals.

FIGURE 6.2. Operation of the Management Bus

6.0 Hub Management Support (Continued)

Packet Status Register 0

D7	D6	D5	D4	D3	D2	D1	D0
A5	A4	A3	A2	A1	A0	PCOMPD	resv

Bit	Symbol	Description
D0	resv	**RESERVED FOR FUTURE USE:** This bit is currently undefined, management software should not examine the state of this bit.
D1	PCOMPD	**PACKET COMPRESSION DONE:** If packet compression is utilized, this bit informs the user that compression was performed, i.e., the packet was long enough to require compression.
D(7:2)	A(5:0)	**RIC ADDRESS (5:0):** This address is defined by the user and is supplied when writing to the RIC Address Register. It is used by hub management software to distinguish between RICs in a multi-RIC system.

Packet Status Register 1

D7	D6	D5	D4	D3	D2	D1	D0
CRCER	FAE	COL	CLN	PA3	PA2	PA1	PA0

Bit	Symbol	Description
D(3:0)	PA(3:0)	**PORT ADDRESS:** This field defines the port which is receiving the packet.
D4	CLN	**CLEAN RECEIVE:** This bit is asserted from the start of reception, and is deasserted if a collision occurs within a window from the start of reception to the end of the 13th byte after SFD detection. If no SFD is detected the window is extended to the end of reception.
D5	COL	**COLLISION:** If a receive or transmit collision occurs during packet repetition the collision bit is asserted.
D6	FAE	**FRAME ALIGNMENT ERROR:** This bit is asserted if a Frame Alignment Error occurred in the repeated packet.
D7	CRCER	**CRC ERROR:** This bit is asserted if a CRC Error occurred in the repeated packet. This status flag should not be tested if the COL bit is asserted since the error may be simply due to the collision.

6.0 Hub Management Support (Continued)

Packet Status Register 2

D7	D6	D5	D4	D3	D2	D1	D0
SE	OWC	NSFD	PLER	ELBER	JAB	CBT9	CBT8

Bit	Symbol	Description
D(1:0)	CT(9:8)	**COLLISION TIMER BITS 9 AND 8:** These two bits are the upper bits of the collision bit timer.
D2	JAB	**JABBER EVENT:** This bit indicates that the receive packet was so long the repeater was forced to go into a jabber protect condition.
D3	ELBER	**ELASTICITY BUFFER ERROR:** During the packet an Elasticity Buffer under/overflow occurred.
D4	PLER	**PHASE LOCK LOOP ERROR:** The packet suffered sufficient jitter/noise corruption to cause the phase lock loop decoder to lose lock.
D5	NSFD	**NON SFD:** The repeated packet did not contain a Start of Frame Delimiter. When this bit is set the Repeat Byte Counter counts the length of the entire packet. When this bit is not set the byte counter only counts post SFD bytes. **Note:** The operation of this bit is not inhibited by the occurrence of a collision during packet repetition (see description of the Repeat Byte Counter below).
D6	OWC	**OUT OF WINDOW COLLISION:** The packet suffered an out of window collision.
D7	SE	**SHORT EVENT:** The receive activity was so small it met the criteria to be classed as a short event.

The other registers comprise the remainder of the collision timer register [PSR(3)], the Repeat Byte Count registers [PSR(4), (5)], and the Inter Frame Gap Counter "IFG" register [PSR(6)].

Collision Bit Timer

The Collision Timer counts in bit times the time between the start of repetition of the packet and the detection of the packet's first collision. The Collision counter increments as the packet is repeated and freezes when a collision occurs. The value in the counter is only valid when the collision bit "COL" in [PSR(1)] is set.

Repeat Byte Counter

The Repeat Byte Counter is a 16 bit counter which can perform two functions. In cases where the transmitted packet possesses an SFD, the byte counter counts the number of received bytes after the SFD field. Alternatively if no SFD is repeated the counter reflects the length of the packet, counted in bytes, starting at the beginning of the preamble field. When performing the latter function the counter is shortened to 8 bits. Thus the maximum count value is 255 bytes. The mode of counting is indicated by the "NSFD" bit in [PSR(2)]. In order to check if the received packet was genuinely a Non-SFD packet, the status of the COL bit should be checked. During collisions SFD fields may be lost or created, Management software should be robust to this kind of behaviour.

Inter Frame Gap (IFG) Bit Timer

The IFG counter counts in bit times the period in between repeater transmissions. The IFG counter increments whenever the RIC is not transmitting a packet. If the IFG is long, i.e., greater than 255 bits the counter sticks at this value. Thus an apparent count value of 255 should be interpreted as 255 or more bit times.

6.4 DESCRIPTION OF HARDWARE CONNECTION FOR MANAGEMENT INTERFACE

The RIC has been designed so it may be connected to the Management bus directly or via external bus transceivers. The latter is advantageous in large repeaters. In this application the system backplane is often heavily loaded beyond the drive capabilities of the on-chip bus drivers.

The uni-directional nature of information transfer on the MCRS, MRXD and MRXC signals, means a single open drain output pin is adequate for each of these signals. The Management Enable (MEN) RIC output pin performs the function of a drive enable for an external bus transceiver if one is required.

In common with the Inter-RIC bus signals ACTN, ANYXN, COLN and IRE the MCRS active level asserted by the MCRS output is determined by the state of the BINV Mode Load configuration bit.

7.0 Port Block Functions

The RIC has 13 port logic blocks (one for each network connection). In addition to the packet repetition operations already described, the port block performs two other functions:

1. The physical connection to the network segment (transceiver function).
2. It provides a means to protect the network from malfunctioning segments (segment partition).

Each port has its own status register. This register allows the user to determine the current status of the port and configure a number of port specific functions.

7.0 Port Block Functions (Continued)

7.1 TRANSCEIVER FUNCTIONS

The RIC may connect to network segments in three ways:

1. Over AUI cable to transceiver boxes,
2. Directly to board mounted transceivers,
3. To twisted pair cable via a simple interface.

The first method is only supported by RIC port 1 (the AUI port). Options (2) and (3) are available on ports 2 to 13. The selection of the desired option is made at device initialization during the Mode Load operation. The Transceiver Bypass XBYPAS configuration bits are used to determine whether the ports will utilize the on-chip 10BASE-T transceiver or bypass these in favour of external transceivers. Four possible combinations of port utilization are supported:

All ports (2 to 13) use the external Transceiver Interface.

Ports 2 to 5 use the external interface, 6 to 13 use the internal 10BASE-T transceivers.

Ports 2 to 7 use the external interface, 8 to 13 use the internal 10BASE-T transceivers.

All ports (2 to 13) use the internal 10BASE-T transceivers.

10BASE-T Transceiver Operation

The RIC contains virtually all the digital and analog circuits required for connection to 10BASE-T network segments. The only additional active component is an external driver packet. The connection for a RIC port to a 10BASE-T segment is shown in *Figure 7.1*. The diagram shows the components required to connect one of the RIC's ports to a 10BASE-T segment. The major components are the driver package, a member of the 74ACT family, and an integrated filter/choke network.

The operation of the 10BASE-T transceiver's logical functions may be modified by software control. The default mode of operation is for the transceivers to transmit and expect reception of link pulses. This may be modified if a logic one is written to the $\overline{\text{GDLNK}}$ bit of a port's status register. The port's transceiver will operate normally but will not transmit link pulses nor monitor their reception. Thus the entry to a link fail state and the associated modification of transceiver operation will not occur.

The on-chip 10BASE-T transceivers automatically detect and correct the polarity of the received data stream. This polarity detection scheme relies upon the polarity of the received link pulses and the end of the packet waveform. Polarity detection and correction may be disabled under software control as follows:

1) Write the value 07H to the Page Select Register (address 10H).
2) Write the value 02H to the address 11H. (Note that address 11H will read back 00H after writing 02H to it).

This is the only exception for accessing any of the reserved pages 4 to 7.

External Transceiver Operation

RIC ports 2 to 13 may be connected to media other than twisted-pair by opting to bypass the on-chip transceivers. When using external transceivers the user must have the external transceivers perform collision detection and the other functions associated with an IEEE 802.2 Media Access Unit. *Figure 7.2* shows the connection between a repeater port and a coaxial transceiver using the AUI type interface.

7.2 SEGMENT PARTITION

Each of the RIC's ports has a dedicated state machine to perform the functions defined by the IEEE partition algorithm as shown in *Figure 7.3*. To allow users to customize this algorithm for different applications a number of user selected options are available during device configuration at power up (the Mode Load Cycle).

Five different options are provided:

1. Operation of the 13 partition state machines may be disabled via the disable partition $\overline{\text{DPART}}$ configuration bit (Pin D6).
2. The value of consecutive counts required to partition a segment (the CCLimit specification) may be set at either 31 or 63 consecutive collisions.
3. The use of the TW5 specification in the partition algorithm differentiates between collisions which occur early in a packet (before TW5 has elapsed) and those which occur late in the packet (after TW5 has elapsed). These late or "out of window" collisions can be regarded in the same manner as early collisions if the Out of Window Collision Enable $\overline{\text{OWCE}}$ option is selected. This configuration bit is applied to the D4 pin during the Mode Load operation. The use of $\overline{\text{OWCE}}$ delays until the end of the packet the operation of the state diagram branch marked (1) and enables the branch marked (2) in *Figure 7.3*.
4. The operation of the ports' state machines when reconnecting a segment may also be modified by the user. The Transmit Only $\overline{\text{TXONLY}}$ configuration bit allows the user to prevent segment reconnection unless the reconnecting packet is being sourced by the repeater. In this case the repeater is transmitting on to the segment, rather than the segment transmitting when the repeater is idle. The normal mode of reconnection does not differentiate between such packets. The $\overline{\text{TXONLY}}$ configuration bit is input on Pin D5 during the Mode Load cycle. If this option is selected the operation of the state machine branch marked (3) in *Figure 7.3* is affected.
5. The RIC may be configured to use an additional criterion for segment partition. This is referred to as loop back partition. If this operation is selected the partition state machine monitors the receive and collision inputs from a network segment to discover if they are active when the port is transmitting. Thus determining if the network transceiver is looping back the data pattern from the cable. A port may be partitioned if no data or collision signals are seen by the partition logic in the following window: 61 to 96 network bit times after the start of transmission see data sheet Section 8 for details. A segment partitioned by this operation may be reconnected in the normal manner.

In addition to the autonomous operation of the partition state machines, the user may reset these state machines. This may be done individually to each port by writing a logic one to the $\overline{\text{PART}}$ bit in its status register. The port's partition state machine and associated counters are reset and the port is reconnected to the network. The reason why a port become partitioned may be discovered by the user by reading the port's status register.

7.0 Port Block Functions (Continued)

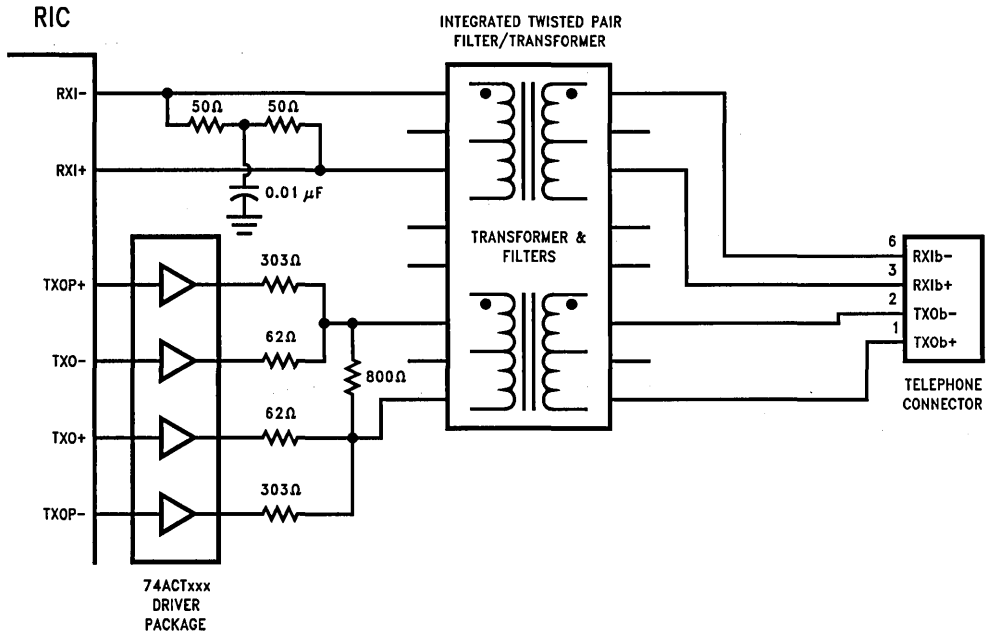
7.3 PORT STATUS REGISTER FUNCTIONS

Each RIC port has its own status register. In addition to providing status concerning the port and its network segment the register allows the following operations to be performed upon the port:

1. Port disable
2. Link Disable
3. Partition reconnection
4. Selection between normal and reduced squelch levels

Note that the link disable and port disable functions are mutually exclusive functions, i.e., disabling link does not affect receiving and transmitting from/to that port and disabling a port does not disable link.

When a port is disabled packet transmission and reception between the port's segment and the rest of the network is prevented.

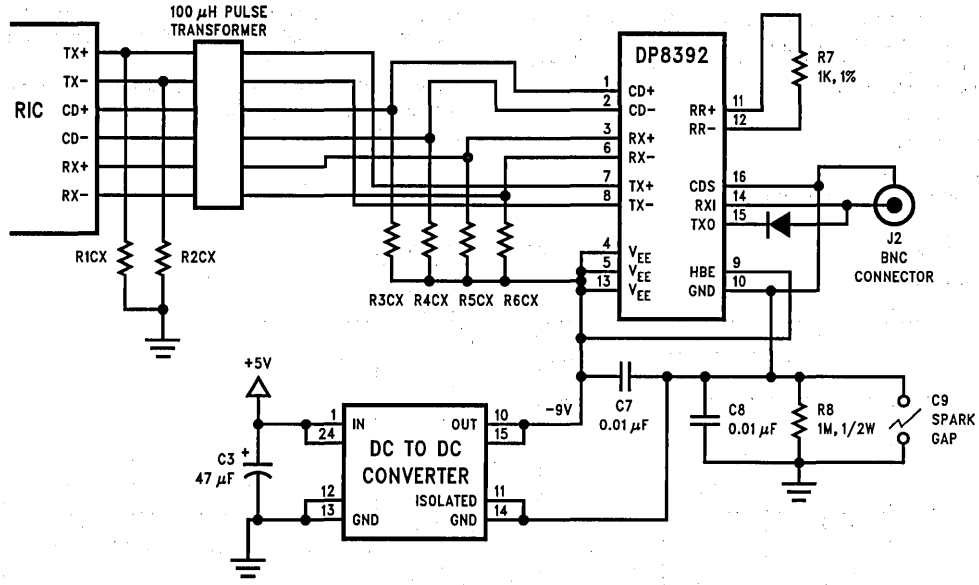


TL/F/11096–19

Recommended Modules:
Pulse Engineering PE65431, Filter-Transformer-Choke
Pulse Engineering PE65423, PE65424 (SIP) Filter-Transformer-Choke
Valor FL1012, Filter-Transformer-Choke
Belfuse 0556-3392-00 Filter-Transformer-Choke
Belfuse 0556-3899-02 Buffer-Filter-Transformer-Choke

FIGURE 7.1. Port Connection to a 10BASE-T Segment

7.0 Port Block Functions (Continued)

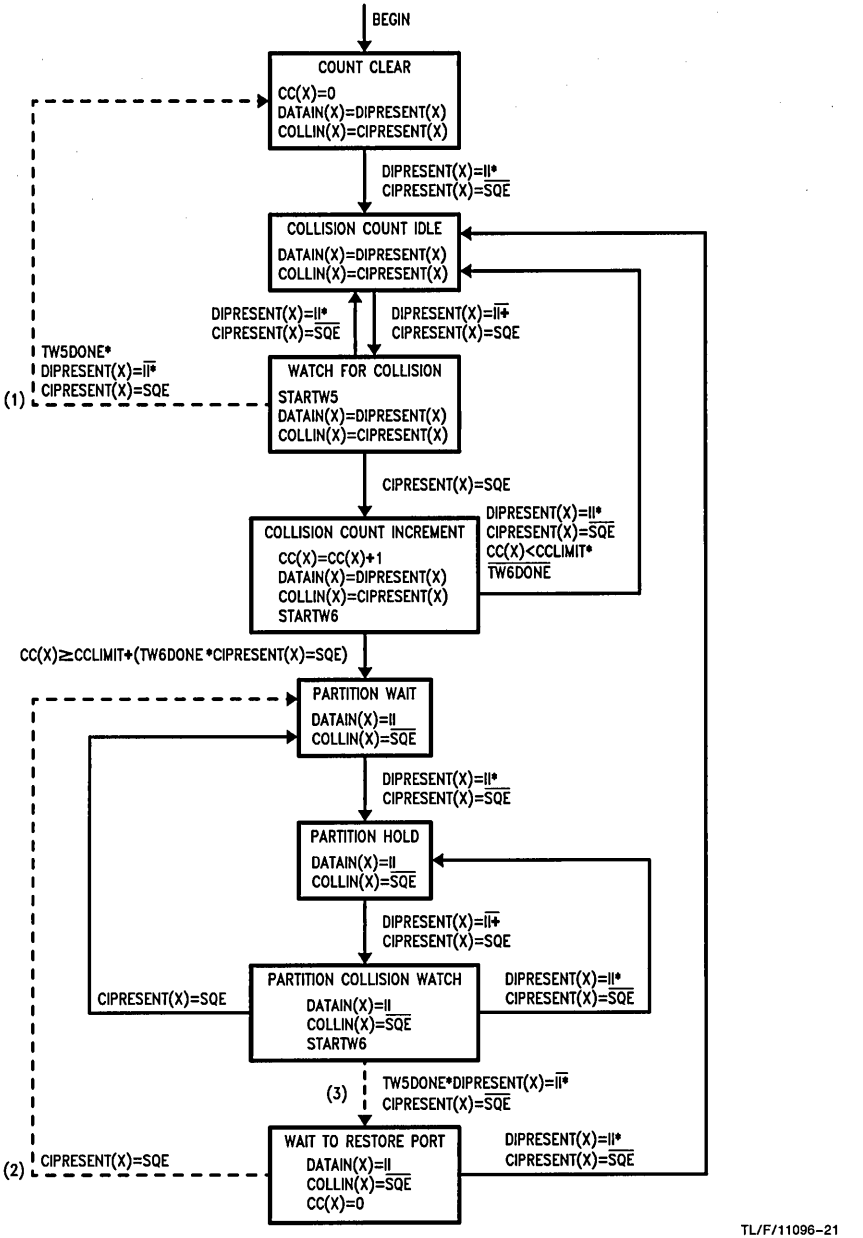


TL/F/11096–20

FIGURE 7.2. Port Connection to a 10BASE2 Segment (AUI Interface Selected)

The preceding diagrams show a RIC port (Numbers 2 to 13) connected to a 10BASE-T and a 10BASE2 segment. The values of any components not indicated above are to be determined.

7.0 Port Block Functions (Continued)



TL/F/11096–21

FIGURE 7.3. IEEE Segment Partition Algorithm

8.0 RIC Registers

RIC Register Address Map

The RIC's registers may be accessed by applying the required address to the five Register Address (RA(4:0)) input pins. Pin RA4 makes the selection between the upper and lower halves of the register array. The lower half of the register map consists of 16 registers:

1 RIC Real Time Status and Configuration register,
13 Port Real Time Status registers,
1 RIC Configuration Register
1 Real Time Interrupt Status Register.

These registers may be directly accessed at any time via the RA(4:0) pins, (RA4 = 0). The upper half of the register map, (RA4 = 1), is organized as 4 pages of registers:

Event Count Configuration page (0),
Event Record page (1),
Lower Event Count page (2)
Upper Event Count page (3)

Register access within these pages is also performed using the RA(4:0) pins, (RA4 = 1). Page switching is performed by writing to the Page Selection bits (PSEL2, 1, 0). These bits are found in the Page Select Register, located at address 10 hex on each page of the upper half of the register array. AT power on these bits default to 0 Hex, i.e., page zero.

8.0 RIC Registers (Continued)

Register Memory Map

Address	Name			
	Page (0)	Page (1)	Page (2)	Page (3)
00H	RIC Status and Configuration Register			
01H	Port 1 Status Register			
02H	Port 2 Status Register			
03H	Port 3 Status Register			
04H	Port 4 Status Register			
05H	Port 5 Status Register			
06H	Port 6 Status Register			
07H	Port 7 Status Register			
08H	Port 8 Status Register			
09H	Port 9 Status Register			
0AH	Port 10 Status Register			
0BH	Port 11 Status Register			
0CH	Port 12 Status Register			
0DH	Port 13 Status Register			
0EH	RIC Configuration Register			
0FH	Real Time Interrupt Register			
10H	Page Select Register			
11H	Device Type Register	Port 1 Event Record Register (ERR)		
12H	Lower Event Count Mask Register (ECMR)	Port 2 ERR	Port 1 Lower Event Count Register (ECR)	Port 8 Lower ECR
13H	Upper ECMR	Port 3 ERR	Port 1 Upper ECR	Port 8 Upper ECR
14H	Event Record Mask Register	Port 4 ERR	Port 2 Lower ECR	Port 9 Lower ECR
15H	resv	Port 5 ERR	Port 2 Upper ECR	Port 9 Upper ECR
16H	Management/Interrupt Configuration Register	Port 6 ERR	Port 3 Lower ECR	Port 10 Lower ECR
17H	RIC Address Register	Port 7 ERR	Port 3 Upper ECR	Port 10 Upper ECR
18H	Packet Compress Decode Register	Port 8 ERR	Port 4 Lower ECR	Port 11 Lower ECR
19H	resv	Port 9 ERR	Port 4 Upper ECR	Port 11 Upper ECR
1AH	resv	Port 10 ERR	Port 5 Lower ECR	Port 12 Lower ECR
1BH	resv	Port 11 ERR	Port 5 Upper ECR	Port 12 Upper ECR
1CH	resv	Port 12 ERR	Port 6 Lower ECR	Port 13 Lower ECR
1DH	resv	Port 13 ERR	Port 6 Upper ECR	Port 13 Upper ECR
1EH	resv		Port 7 Lower ECR	
1FH	IFG Threshold		Port 7 Upper ECR	

Note: All registers marked resv on pages 0 to 3 must not be accessed by the user. The other register pages, 4 to 7, are also reserved.

8.0 RIC Registers (Continued)

Register Array Bit Map Addresses 00H to 10H

Address (Hex)	D7	D6	D5	D4	D3	D2	D1	D0
00	BINV	BYPAS2	BYPAS1	$\overline{\text{APART}}$	$\overline{\text{JAB}}$	$\overline{\text{AREC}}$	$\overline{\text{ACOL}}$	resv
01 to 0D	DISPT	SQL	PTYPE1	PTYPE0	$\overline{\text{PART}}$	$\overline{\text{REC}}$	$\overline{\text{COL}}$	$\overline{\text{GDLNK}}$
0E	MINMAX	$\overline{\text{DPART}}$	$\overline{\text{TXONLY}}$	$\overline{\text{OWCE}}$	$\overline{\text{LPPART}}$	$\overline{\text{CCLIM}}$	Tw2	resv
0F	IVCTR3	IVCTR2	IVCTR1	IVCTR0	ISRC3	ISRC2	ISRC1	ISRC0

Register Array Bit Map Addresses 10H to 1FH Page (0)

Address (Hex)	D7	D6	D5	D4	D3	D2	D1	D0
10	FC	HC	LC	FF	resv	PSEL2	PSEL1	PSEL0
11	0	0	0	0	0	0	0	0
12	BDLNKC	PARTC	RECC	SEC	NSFDC	PLERC	ELBERC	JABC
13	resv	resv	OWCC	RXCOLC	TXCOLC	resv	FWF	ROR
14	BDLNKE	PARTE	OWCE	SEE	NSFDE	PLERE	ELBERE	JABE
16	$\overline{\text{IFC}}$	$\overline{\text{IHC}}$	$\overline{\text{ILC}}$	$\overline{\text{IFF}}$	$\overline{\text{IREC}}$	$\overline{\text{ICOL}}$	$\overline{\text{IPART}}$	MIFCON
17	A5	A4	A3	A2	A1	A0	resv	resv
18	PCD7	PCD6	PCD5	PCD4	PCD3	PCD2	PCD1	PCD0
1F	IFGT7	IFGT6	IFGT5	IFGT4	IFGT3	IFGT2	IFGT1	IFGT0

Register Array Bit Map Addresses 10H to 1FH Page (1)

Address (Hex)	D7	D6	D5	D4	D3	D2	D1	D0
10	FC	HC	LC	FF	resv	PSEL2	PSEL1	PSEL0
11 to 1D	BDLNK	PART	OWC	SE	NSFD	OWC	PLER	JAB

Register Array Bit Map Addresses 10H to 1FH Pages (2) and (3)

Address (Hex)	D7	D6	D5	D4	D3	D2	D1	D0
10	FC	HC	LC	FF	resv	PSEL2	PSEL1	PSEL0
11	—	—	—	—	—	—	—	—
Even Locations	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
Odd Locations	EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8

8.0 RIC Registers (Continued)

RIC Status and Configuration Register (Address 00H)

The lower portion of this register contains real time information concerning the operation of the RIC. The upper three bits represent the chosen configuration of the transceiver interface employed.

D7	D6	D5	D4	D3	D2	D1	D0
BINV	BYPAS2	BYPAS1	$\overline{\text{APART}}$	$\overline{\text{JAB}}$	$\overline{\text{AREC}}$	$\overline{\text{ACOL}}$	resv

Bit	R/W	Symbol Access	Description
D0	R	resv	**RESERVED FOR FUTURE USE:** Reads as a logic 0.
D1	R	$\overline{\text{ACOL}}$	**ANY COLLISIONS:** 0: A collision is occurring at one or more of the RIC's ports. 1: No collisions.
D2	R	$\overline{\text{AREC}}$	**ANY RECEIVE:** 0: One of the RIC's ports is the current packet or collision receiver. 1: No packet or collision reception within this RIC.
D3	R	$\overline{\text{JAB}}$	**JABBER PROTECT:** 0: The RIC has been forced into jabber protect state by one of its ports or by another port on the Inter-RIC bus, (Multi-RIC operations). 1: No jabber protect conditions exist.
D4	R	$\overline{\text{APART}}$	**ANY PARTITION:** 0: One or more ports are partitioned. 1: No ports are partitioned.
D5	R	BYPAS1	These bits define the configuration of ports 2 to 13 i.e., their use if the internal 10BASE-T transceivers or the external (AUI-like) transceiver interface.
D6	R	BYPAS2	
D7	R	BINV	**BUS INVERT:** This register bit informs whether the Inter-RIC signals: IRE, ACTN, ANYXN, COLN and Management bus signal MCRS are active high or low. 0: Active high 1: Active low

8.0 RIC Registers (Continued)

Port Real Time Status Registers (Address 01H to 0DH)

D7	D6	D5	D4	D3	D2	D1	D0
DISPT	EGP	PTYPE1	PTYPE0	$\overline{\text{PART}}$	$\overline{\text{REC}}$	$\overline{\text{COL}}$	$\overline{\text{GDLNK}}$

Bit	R/W	Symbol	Description
D0	R/W	$\overline{\text{GDLNK}}$	**GOOD LINK:** 0: Link pulses are being received by the port. 1: Link pulses are not being received by the port logic. **Note:** Writing a 1 to this bit will cause the 10BASE-T transceiver not the transmit or monitor the reception of link pulses. If the internal 10BASE-T transceivers are not selected or if port 1 (AUI port) is read, then this bit is undefined.
D1	R	$\overline{\text{COL}}$	**COLLISION:** 0: A collision is happening or has occurred during the current packet. 1: No collisions have occurred as yet during this packet.
D2	R	$\overline{\text{REC}}$	**RECEIVE:** 0: This port is now or has been the receive source of packet or collision information for the current packet. 1: This port has not been the receive source during the current packet.
D3	R/W	$\overline{\text{PART}}$	**PARTITION:** 0: This port is partitioned. 1: This port is not partitioned. Writing a logic one to this bit forces segment reconnection and partition state machine reset. Writing a zero to this bit has no effect.
D(5, 4)	R	PTYPE0 PTYPE1	**PARTITION TYPE 0** **PARTITION TYPE 1** The partition type bits provide information specifying why the port is partitioned.

PTYPE1	PTYPE0	Information
0	0	Consecutive Collision Limit Reached
0	1	Excessive Length of Collision Limit Reached
1	0	Failure to See Data Loopback from Transceiver in Monitored Window
1	1	Processor Forced Reconnection

D6	R/W	SQL	**SQUELCH LEVEL:** 0: Port operates with normal IEEE receive squelch level. 1: Port operates with reduced receive squelch level. Note: This bit has no effect when the external transceiver is selected.
D7	R/W	DISPT	**DISABLE PORT:** 0: Port operates as defined by repeater operations. 1: All port activity is prevented.

8.0 RIC Registers (Continued)

RIC Configuration Register (Address 0EH)

This register displays the state of a number of RIC configuration bits loaded during the Mode Load operation.

D7	D6	D5	D4	D3	D2	D1	D0
MINMAX	$\overline{\text{DPART}}$	$\overline{\text{TX ONLY}}$	$\overline{\text{OWCE}}$	$\overline{\text{LPPART}}$	$\overline{\text{CCLIM}}$	Tw2	resv

Bit	R/W	Symbol	Description
D0	R	resv	**RESERVED FOR FUTURE USE:** Value set at logic one.
D1	R	Tw2	**CARRIER RECOVERY TIME:** 0: Tw2 set at 5 bits. 1: Tw2 set at 3 bits.
D2	R	$\overline{\text{CCLIM}}$	**CONSECUTIVE COLLISION LIMIT:** 0: Consecutive collision limit set at 63 collisions. 1: Consecutive collision limit set at 31 collisions.
D3	R	$\overline{\text{LPPART}}$	**LOOPBACK PARTITION:** 0: Partitioning upon lack of loopback from transceivers is enabled. 1: Partitioning upon lack of loopback from transceivers is disabled.
D4	R	$\overline{\text{OWCE}}$	**OUT OF WINDOW COLLISION ENABLE:** 0: Out of window collisions are treated as in window collisions by the segment partition state machines. 1: Out of window collisions are treated as out of window collisions by the segment partition state machines.
D5	R	$\overline{\text{TXONLY}}$	**ONLY RECONNECT UPON SEGMENT TRANSMISSION:** 0: A segment will only be reconnected to the network if a packet transmitted by the RIC onto that segment fulfills the requirements of the segment reconnection algorithm. 1: A segment will be reconnected to the network by any packet on the network which fullfills the requirements of the segment reconnection algorithm.
D6	R	$\overline{\text{DPART}}$	**DISABLE PARTITION:** 0: Partitioning of ports by on-chip algorithms is prevented. 1: Partitioning of ports by on-chip algorithms is enabled.
D7	R	MINMAX	**MINIMUM/MAXIMUM DISPLAY MODE:** 0: LED display set in minimum display mode. 1: LED display set in maximum display mode.

8.0 RIC Registers (Continued)

Real Time Interrupt Register (Address 0FH)

The Real Time Interrupt register (RTI) contains information which may change on a packet by packet basis. Any remaining interrupts which have not been serviced before the following packet is transmitted are cleared. Since multiple interrupt sources may be displayed by the RTI a priority scheme is implemented. A read cycle to the RTI gives the interrupt source and an address vector indicating the RIC port which generated the interrupt. The order of priority for the display of interrupt information is as follows:

1. The receive source of network activity (Port N),
2. The first RIC port showing collision
3. A port partitioned or reconnected.

During the repetition of a single packet it is possible that multiple ports may be partitioned or alternatively reconnected. The ports have equal priority in displaying partition/reconnection information. This data is derived from the ports by the RTI register as it polls consecutively around the ports.

Reading the RTI clears the particular interrupt. If no interrupt sources are active the RTI returns a no valid interrupt status.

D7	D6	D5	D4	D3	D2	D1	D0
IVCTR3	IVCTR2	IVCTR1	IVCTR0	ISRC3	ISRC2	ISRC1	ISRC0

Bit	R/W	Symbol Access	Description
D(3:0)	R	ISCR(3:0)	**INTERRUPT SOURCE:** These four bits indicate the reason why the interrupt was generated.
D(7:4)	R	IVCTR(3:0)	**INTERRUPT VECTOR:** This field defines the port address responsible for generating the interrupt.

The following table shows the mapping of interrupt sources onto the D3 to D0 pins. Essentially each of the three interrupt sources has a dedicated bit in this field. If a read to the RTI produces a low logic level on one of these bits then the interrupt source may be directly decoded. Associated with the source of the interrupt is the port where the event is occurring. If no unmasked events (receive, collision, etc.), have occurred when the RTI is read then an all ones pattern is driven by the RIC onto the data pins.

D7	D6	D5	D4	D3	D2	D1	D0	Comments
PA3	PA2	PA1	PA0	1	1	0	1	First Collision PA(3:0) = Collision Port Address
PA3	PA2	PA1	PA0	1	0	1	1	Receive PA(3:0) = Receive Port Address
PA3	PA2	PA1	PA0	0	1	1	1	Partition Reconnection PA(3:0) = Partition Port Address
1	1	1	1	1	1	1	1	No Valid Interrupt

8.0 RIC Registers (Continued)

Page Select Register ((All Pages) Address 10H)

The Page Select register performs two functions:

1. It enables switches to be made between register pages,
2. It provides status information regarding the Event Logging Interrupts.

D7	D6	D5	D4	D3	D2	D1	D0
FC	HC	LC	FF	resv	PSEL2	PSEL1	PSEL0

Bit	R/W	Symbol	Description
D(2:0)	R/W	PSEL(2:0)	**PAGE SELECT BITS:** When read these bits indicate the currently selected Upper Register Array Page. Write cycles to these locations facilitates page swapping.
D3	R	resv	**RESERVED FOR FUTURE USE**
D4	R	FF	**FLAG FOUND:** This indicates one of the unmasked event recording latches has been set.
D5	R	LC	**LOW COUNT:** This indicates one of the port event counters has a value less than 00FF Hex.
D6	R	HC	**HIGH COUNT:** This indicates one of the port event counters has a value greater than C000 Hex.
D7	R	FC	**FULL COUNTER:** This indicates one of the port event counters has a value equal to FFFF Hex.

Device Type Register (Page 0H Address 11H)

This register may be used to distinguish different revisions of RIC. If this register is read it will return a different value each for DP83950 revisions. (Contact National Semiconductor for revision information.) Write operations to this register have no effect upon the contents.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	X	X

8.0 RIC Registers (Continued)

Lower Event Count Mask Register (Page 0H Address 12H)

D7	D6	D5	D4	D3	D2	D1	D0
BDLNKC	PARTC	RECC	PGPKC	NSFDC	PLERC	ELBER C	JABC

Bit	R/W	Symbol	Description
D0	R/W	JABC	**JABBER COUNT ENABLE:** Enables recording of Jabber Protect events.
D1	R/W	ELBERC	**ELASTICITY BUFFER ERROR COUNT ENABLE:** Enables recording of Elasticity Buffer Error events.
D2	R/W	PLERC	**PHASE LOCK ERROR COUNT ENABLE:** Enables recording of Carrier Error events.
D3	R/W	NSFDC	**NON SFD COUNT ENABLE:** Enables recording of Non SFD packet events.
D4	R/W	SEC	**SHORT EVENT COUNT ENABLE:** Enables recording of Short events.
D5	R/W	RECC	**RECEIVE COUNT ENABLE:** Enables recording of Packet Receive (port N status) events that do not suffer collisions.
D6	R/W	PARTC	**PARTITION COUNT ENABLE:** Enables recording of Partition events.
D7	R/W	BDLNKC	**BAD LINK COUNT ENABLE:** Enables recording of Bad Link events.

Upper Event Count Mask Register (Page 0H Address 13H)

D7	D6	D5	D4	D3	D2	D1	D0
resv	resv	OWCC	RXCOLC	TXCOLC	resv	FWF	ROR

Bit	R/W	Symbol	Description
D0	R/W	ROR	**RESET ON READ:** This bit selects the action a read operation has upon a port's event counter: 0: No effect upon register contents. 1: The counter register is reset.
D1	R/W	FWF	**FREEZE WHEN FULL:** This bit controls the freezing of the Event Count registers when the counter is full (FFFF Hex)
D2	R	resv	**RESERVED FOR FUTURE USE:** This bit should be written with a low logic level.
D3	R/W	TXCOLC	**TRANSMIT COLLISION COUNT ENABLE:** Enables recording of transmit collision events only.
D4	R/W	RXCOLC	**RECEIVE COLLISION COUNT ENABLE:** Enables recording of receive collision events only.
D5	R/W	OWCC	**OUT OF WINDOW COLLISION COUNT ENABLE:** Enables recording of out of window collision events only.
D(7: 6)	R	resv	**RESERVED FOR FUTURE USE:** These bits should be written with a low logic level.

Note 1: To count all collisions then both the TXCOLC and RXCOLC bits must be set. The OWCC bit should not be set otherwise the port counter will be incremented twice when an out of collision window collision occurs. The OWCC bit alone should be set if only out of window collision are to be counted.

Note 2: Writing a 1 enables the event to be counted.

8.0 RIC Registers (Continued)

Event Record Mask Register (Page 0H Address 14H)

D7	D6	D5	D4	D3	D2	D1	D0
BDLNKE	PARTE	OWCE	PGPKE	NSFDE	PLERE	ELBERE	JABE

Bit	R/W	Symbol	Description
D0	R/W	JABE	**JABBER ENABLE:** Enables recording of Jabber Protect events.
D1	R/W	ELBERE	**ELASTICITY BUFFER ERROR ENABLE:** Enables recording of Elasticity Buffer Error events.
D2	R/W	PLERE	**PHASE LOCK ERROR ENABLE:** Enables recording of Carrier Error events.
D3	R/W	NSFDE	**NON SFD ENABLE:** Enables recording of Non SFD packet events.
D4	R/W	SEE	**SHORT EVENT ENABLE:** Enables recording of Short Events.
D5	R/W	OWCE	**OUT OF WINDOW COLLISION COUNT ENABLE:** Enables recording of Out of Window Collision events only.
D6	R/W	PARTE	**PARTITION ENABLE:** Enables recording of Partition events.
D7	R/W	BDLNKE	**BAD LINK ENABLE:** Enables recording of Bad Link Events.

Note: Writing a 1 enables the event to be recorded.

8.0 RIC Registers (Continued)

Interrupt and Management Configuration Register (Page 0H Address 16H)

This register powers up with all bits set to one and must be initialized by a processor write cycle before any events will generate interrupts.

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{IFC}$	$\overline{IHC}$	$\overline{ILC}$	$\overline{IFF}$	$\overline{IREC}$	$\overline{ICOL}$	$\overline{IPART}$	MIFCON

Bit	R/W	Symbol	Description
D0	R/W	MIFCON	**MANAGEMENT INTERFACE CONFIGURATION:** 0: All Packets repeated are transmitted over the Management bus. 1: Packets repeated by the RIC which do not have a Start of Frame Delimiters are not transmitted over the Management bus.
D1	R/W	$\overline{IPART}$	**INTERRUPT ON PARTITION:** 0: Interrupts will be generated(1) if a port becomes Partitioned. 1: No interrupts are generated by this condition.
D2	R/W	$\overline{ICOL}$	**INTERRUPT ON COLLISION:** 0: Interrupts will be generated(1) if this RIC has a port which experiences a collision, Single RIC applications, or contains a port which experiences a receive collision or is the first port to suffer a transmit collision in a packet in Multi-RIC applications. 1: No interrupts are generated by this condition.
D3	R/W	$\overline{IREC}$	**INTERRUPT ON RECEIVE:** 0: Interrupts will be generated(1) if this RIC contains the receive port for packet or collision activity. 1: No interrupts are generated by this condition.
D4	R/W	$\overline{IFF}$	**INTERRUPT ON FLAG FOUND:** 0: Interrupts will be generated(2) if one or more than one of the flags in the flag array is true. 1: No interrupts are generated by this condition.
D5	R/W	$\overline{ILC}$	**INTERRUPT ON LOW COUNT:** 0: Interrupt generated(2) when one or more of the Event Counters holds a value less than 256 counts. 1: No effect
D6	R/W	$\overline{IHC}$	**INTERRUPT ON HIGH COUNT:** 0: Interrupt generated(2) when one or more of the Event Counters holds a value in excess of 49152 counts. 1: No effect
D7	R/W	$\overline{IFC}$	**INTERRUPT ON FULL COUNTER:** 0: Interrupt generated(2) when one or more of the Event Counters is full. 1: No effect

Note 1: ($\overline{RTI}$ pin goes active)

Note 2: ($\overline{ELI}$ pin goes active)

8.0 RIC Registers (Continued)

RIC Address Register (Page 0H Address 17H)

This register may be used to differentiate between RICs in a multi-RIC repeater system. The contents of this register form part of the information available through the management bus.

D7	D6	D5	D4	D3	D2	D1	D0
A5	A4	A3	A2	A1	A0	res	res

Packet Compress Decode Register (Page 0H Address 18H)

This register is used to determine the number of bytes in the data field of a packet which are transferred over the management bus when the packet compress option is employed. The register bits perform the function of a direct binary decode. Thus up to 255 bytes of data may be transferred over the management bus if packet compression is selected.

D7	D6	D5	D4	D3	D2	D1	D0
PCD7	PCD6	PCD5	PCD4	PCD3	PCD2	PCD1	PCD0

Inter Frame Gap Threshold Select Register (Page 0H Address 1FH)

This register is used to configure the hub management interface to provide a certain minimum inter frame gap between packets transmitted over the management bus. The value written to this register, plus one, is the magnitude in bit times of the minimum IFG allowed on the management bus.

D7	D6	D5	D4	D3	D2	D1	D0
IFGT7	IFGT6	IFGT5	IFGT4	IFGT3	IFGT2	IFGT1	IFGT0

Port Event Record Registers (Page 1H Address 11H to 1DH)

These registers hold the recorded events for the specified RIC port. The flags are cleared when the register is read.

D7	D6	D5	D4	D3	D2	D1	D0
BDLNK	PART	OWC	PGPK	NSFD	PLER	ELBER	JAB

Bit	R/W	Symbol	Description
D0	R	JAB	**JABBER:** A Jabber Protect event has occurred.
D1	R	ELBER	**ELASTICITY BUFFER ERROR:** A Elasticity Buffer Error has occurred.
D2	R	PLER	**PHASE LOCK ERROR:** A Phase Lock Error event has occurred.
D3	R	NSFD	**NON SFD:** A Non SFD packet event has occurred.
D4	R	SE	**SHORT EVENT:** A Short event has occurred.
D5	R	OWC	**OUT OF WINDOW COLLISION:** An out of window collision event has occurred.
D6	R	PART	**PARTITION:** A partition event has occurred.
D7	R	BDLNK	**BAD LINK:** A link failure event has occurred.

Port Event Count Register (Pages 2H and 3H)

The Event Count (EC) register shows the instantaneous value of the specified port's 16-bit counter. The counter increments when an enabled event occurs. The counter may be cleared when it is read and prevented from rolling over when the maximum count is reached by setting the appropriate control bits in the Upper Event Count mask register. Since the RIC's processor port is octal and the counters are 16 bits long a temporary holding register is employed for register reads. When one of the counters is read, either high or low byte first, all 16 bits of the counter are transferred to a holding register. Provided the next read cycle to the counter array accesses the same counter's, other byte, then the read cycle accesses the holding register. This avoids the problem of events occurring in between the two processor reads and indicating a false count value. In order to enter a new value to the holding register a different counter must be accessed or the same counter byte must be re-read.

Lower Byte

D7	D6	D5	D4	D3	D2	D1	D0
EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

Upper Byte

D7	D6	D5	D4	D3	D2	D1	D0
EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8

9.0 AC and DC Specifications

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	0.5V to 7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} +0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} +0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	TBD
Lead Temperature (T_L) (Soldering, 10 seconds)	260°C
ESD Rating (R_{zap} = 1.5k, C_{zap} = 120 pF)	TBD

PARAMETRICS DISCLAIMER

The Current AC and DC specifications contained in this document are considered target design specifications and may not represent actual guaranteed tested timing parameters. This information represents simulated, as well as, limited sampled empirical bench test data. Guaranteed specifications will be provided after full device characterization.

Do not use these specifications for final production designs without directly contacting National Semiconductor.

DC Specifications T_A = 0°C to +70°C, V_{CC} = 5V ±5% unless otherwise specified

PROCESSOR, LED, TWISTED PAIR PORTS, INTER-RIC AND MANAGEMENT INTERFACES

Symbol	Description	Conditions	Min	Max	Units
V_{OH}	Minimum High Level Output Voltage	I_{OH} = −20 μA I_{OH} = −8 mA	V_{CC} − 0.1 3.5		V V
V_{OL}	Minimum Low Level Output Voltage	I_{OL} = 20 μA I_{OL} = 8 mA		0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage		2.0		V
V_{IL}	Maximum Low Level Input Voltage			0.8	V
I_{IN}	Input Current	V_{IN} = V_{CC} or GND	−1.0	1.0	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND	−10	10	μA
I_{CC}	Average Supply Current	V_{IN} = V_{CC} or GND V_{CC} = 5.25V		380	mA
AUI (PORT 1)					
V_{OD}	Differential Output Voltage (TX±)	78Ω Termination and 270Ω Pulldowns	±550	±1200	mV
V_{OB}	Differential Output Voltage Imbalance (TX±)	78Ω Termination and 270Ω Pulldowns	Typical: 40 mV		
V_U	Undershoot Voltage (TX±)	78Ω Termination and 270Ω Pulldowns	Typical: 80 mV		
V_{DS}	Differential Squelch Threshold (RX±, CD±)		−175	−300	mV
V_{CM}	Differential Input Common Mode Voltage (RX±, CD±) (Note 1)		0	5.5	V

Note 1: This parameter is guaranteed by design and is not tested.

9.0 AC and DC Specifications (Continued)

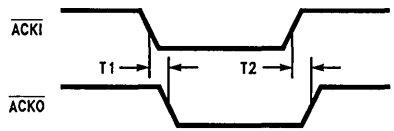
DC Specifications T_A = 0°C to +70°C, V_{CC} = 5V ±5% unless otherwise specified (Continued)

Symbol	Description	Conditions	Min	Max	Units
PSEUDO AUI (PORTS 2–13)					
V_{POD}	Differential Output Voltage (TX±)	270Ω Termination and 1 kΩ Pulldowns	±450	±1200	mV
V_{POB}	Differential Output Voltage Imbalance (TX±)	270Ω Termination and 1 kΩ Pulldowns	Typical: 40 mV		
V_{PU}	Undershoot Voltage (TX±)	270Ω Termination and 1 kΩ Pulldowns	Typical: 80 mV		
V_{PDS}	Differential Squelch Threshold (RX±, CD±)		−175	−300	mV
V_{PCM}	Differential Input Common Mode Voltage (Rx±, CD±) (Note 1)		0	5.5	V
TWISTED PAIR (PORTS 2–13)					
V_{RON}	Minimum Receive Squelch Threshold		±300	±585	mV

Note 1: This parameter is guaranteed by design and is not tested.

AC Specifications

PORT ARBITRATION TIMING



TL/F/11096–22

Number	Symbol	Parameter	Min	Max	Units
T1	ackilackol	ACKI Low to /ACKO Low		23	ns
T2	ackihackoh	ACKI High to /ACKO High		19	ns

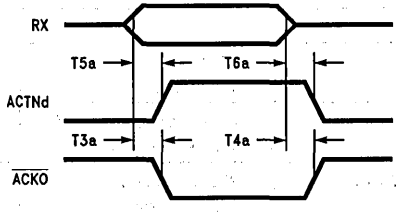
Note: Timing valid with no receive or collision activities.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

RECEIVING TIMINGS—AUI PORTS

Receive activity propagation start up and end delays for ports in **non** 10BASE-T mode



TL/F/11096–23

Number	Symbol	Parameter	Min	Max	Units
T3a	rxaackol	RX Active to /ACKO Low		63	ns
T4a	rxiackoh	RX Inactive to /ACKO High		310	ns
T5a	rxaactna	RX Active to ACTNd Active		105	ns
T6a	rxiactni	RX Inactive to ACTNd Inactive		310	ns

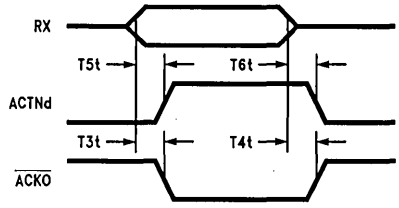
Note: /ACKI assumed high.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

RECEIVE TIMING—10BASE-T PORTS

Receive activity propagation start up and end delays for ports in 10BASE-T mode



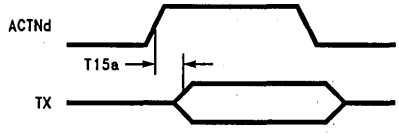
TL/F/11096–24

Number	Symbol	Parameter	Min	Max	Units
T3t	rxaackol	RX Active to /ACKO Low		240	ns
T4t	rxiackoh	RX Inactive to /ACKO High		250	ns
T5t	rxaactna	RX Active to ACTNd Active		240	ns
T6t	rxiactni	RX Inactive to ACTNd Inactive		255	ns

Note: /ACKI assumed high.

TRANSMIT TIMING—AUI PORTS

Transmit activity propagation start up and end delays for ports in **non** 10BASE-T mode



TL/F/11096–25

Number	Symbol	Parameter	Min	Max	Units
T15a	actnatxa	ACTNd Active to TX Active		560	ns

Note: /ACKI assumed high.

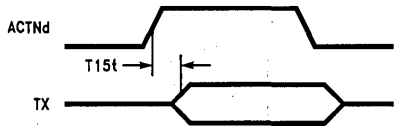
Note: $ACTN_d$ and $ACTN_s$ are tied together.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

TRANSMIT TIMING—10BASE-T PORTS

Receive activity propagation start up and end delays for ports in 10BASE-T mode



TL/F/11096–26

Number	Symbol	Parameter	Min	Max	Units
T15t	actnatxa	ACTNd Active to TX Active		680	ns

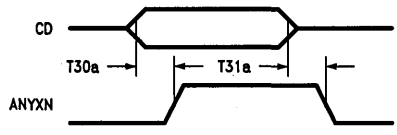
Note: /ACKI assumed high.

Note: $ACTN_d$ and $ACTN_s$ are tied together.

COLLISION TIMING—AUI PORTS

Collision activity propagation start up and end delays for ports in **non** 10BASE-T mode

TRANSMIT COLLISION TIMING



TL/F/11096–27

Number	Symbol	Parameter	Min	Max	Units
T30a	cdaanyxna	CD Active to ANYXN Active		60	ns
T31a	cdianyxni	CD Inactive to ANYXN Inactive (Notes 1, 2)		240	ns

Note 1: TX collision extension has already been performed and no other port is driving ANYXN.

Note 2: Includes TW2.

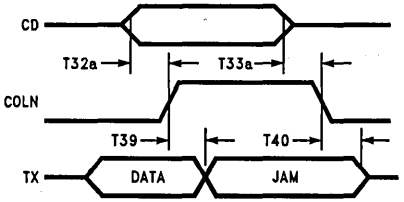
Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

COLLISION TIMING—AUI PORTS

Collision activity propagation start up and end delays for ports in **non** 10BASE-T mode.

RECEIVE COLLISION TIMING



TL/F/11096–28

Number	Symbol	Parameter	Min	Max	Units
T32a	cdacolna	CD Active to COLN Active (Note 1)		50	ns
T33a	cdicolni	CD Inactive to COLN Inactive		200	ns
T39	colnajs	COLN Active to Start of Jam		4.0	Bits
T40	colnije	COLN Inactive to End of Jam (Note 2)		8.0	Bits

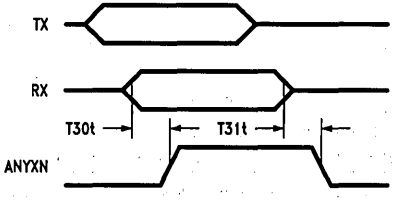
Note 1: PKEN assumed high.

Note 2: Assuming reception ended before COLN goes inactive. TW2 is included in this parameter. Assuming $ACTN_d$ to $ACTN_s$ delay is 0.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

COLLISION TIMING—10BASE-T PORTS

Collision activity propagation start up and end delays for ports in 10BASE-T mode



TL/F/11096–29

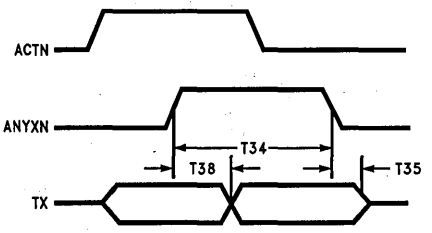
Number	Symbol	Parameter	Min	Max	Units
T30t	colaanya	Collision Active to ANYXN Active		8.0	Bits
T31t	colianyi	Collision Inactive to ANYXN Inactive (Note 1)		100	ns

Note 1: TX collision extension has alreay been performed and no other port is asserting ANYXN.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

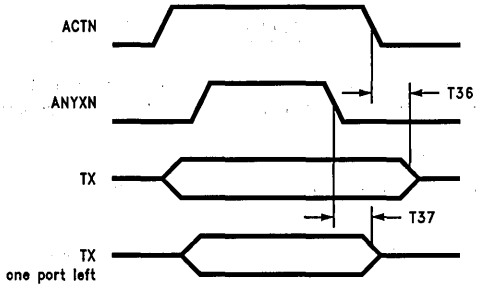
9.0 AC and DC Specifications (Continued)

COLLISION TIMING—ALL PORTS



TL/F/11096–38

Number	Symbol	Parameter	Min	Max	Units
T34	anyamin	ANYXN Active Time	96		Bits
T35	anyitxai	ANYXN Inactive to TX to all Inactive	125	160	ns
T38	anyasj	ANYXN Active to Start of Jam		4.0	Bits



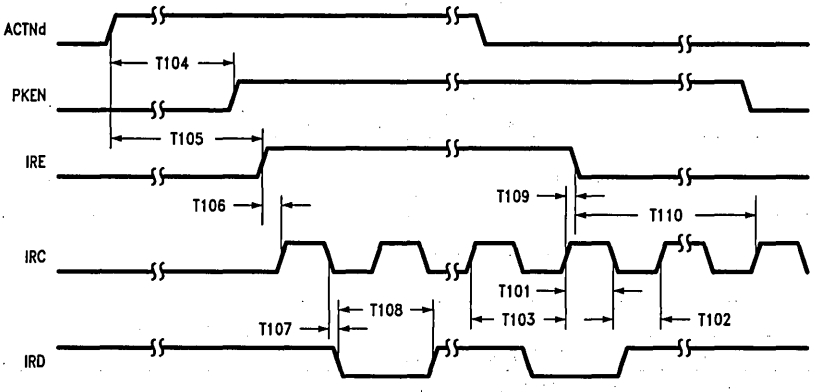
TL/F/11096–39

Number	Symbol	Parameter	Min	Max	Units
T36	actnitxi	ACTN Inactive to TX Inactive	220		ns
T37	anyitxoi	ANYXN Inactive to TX "One Port Left" Inactive	125	160	ns

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

INTER RIC BUS OUTPUT TIMING



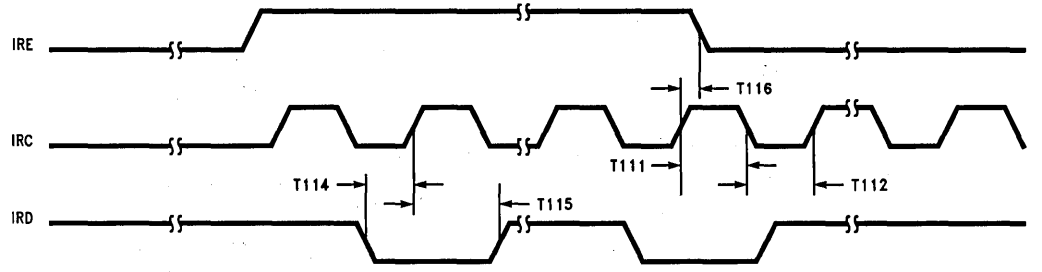
TL/F/11096–35

Number	Symbol	Parameter	Min	Max	Units
T101	ircoh	IRC Output High Time	40	60	ns
T102	ircol	IRC Output Low Time	40	60	ns
T103	ircoc	IRC Output Cycle Time	80	120	ns
T104	actndapkena	ACTNd Active to PKEN Active	580		ns
T105	actndairea	ACTNd Active to IRE Active	585		ns
T106	ireoairca	IRE Output Active to IRC Active		5	ns
T107	irdov	IRD Output Valid from IRC		10	ns
T108	irdos	IRD Output Stable Valid Time	90		ns
T109	ircohirei	IRC Output High to IRE Inactive	30	65	ns
T110	ircclks	Number of IRCs after IRE Inactive	5	5	clks

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

INTER RIC BUS INPUT TIMING



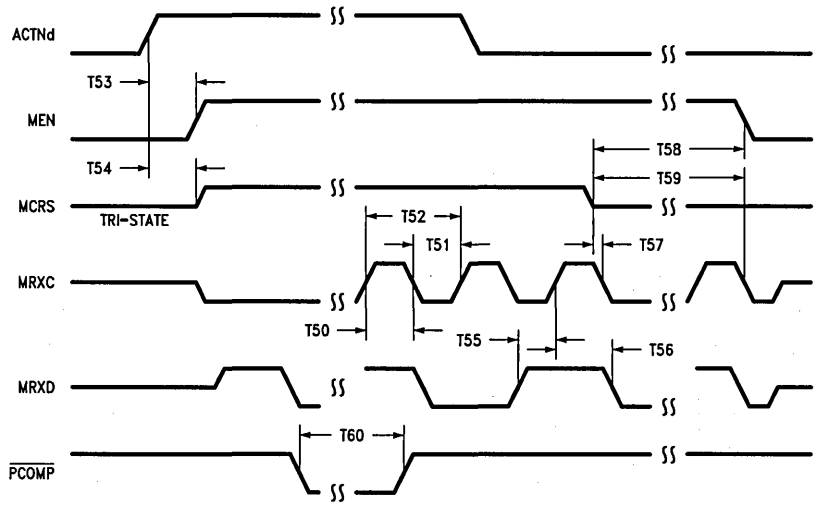
TL/F/11096–40

Number	Symbol	Parameter	Min	Max	Units
T111	ircih	IRC Input High Time	20		ns
T112	ircil	IRC Input Low Time	20		ns
T114	irdisirc	IRD Input Setup to IRC	5		ns
T115	irdihirc	IRD Input Hold from IRC	10		ns
T116	irchiire	IRC Input High to IRE Inactive	25	75	ns

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

MANAGEMENT BUS TIMING



TL/F/11096–30

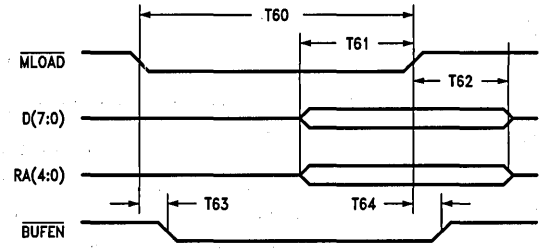
Number	Symbol	Parameter	Min	Max	Units
T50	mrxch	MRXC High Time	40	60	ns
T51	mrxcl	MRXC Low Time	40	60	ns
T52	mrxcd	MRXC Cycle Time	80	120	ns
T53	actndamena	ACTNd Active to MEN Active		600	ns
T54	actndamcrsa	ACTNd Active to MCRS Active		600	ns
T55	mrxds	MRXD Setup	40		ns
T56	mrxdh	MRXD Hold	45		ns
T57	mrxclmcrsi	MRXC Low to MCRS Inactive	−5	6	ns
T58	mcrsimenl	MCRS Inactive to MEN Low		510	ns
T59	mrxcclks	Min Number of MRXCs after MCRS Inactive	5	5	Clks
T60	pcompw	PCOMP Pulse Width	20		ns

Note: The preamble on this bus consists of the following string: 01011.

Note: In these diagrams the Inter-RIC and Management Busses are shown using active high signals, active low signals may also be used. See Section 5.5 Mode Load Operation.

9.0 AC and DC Specifications (Continued)

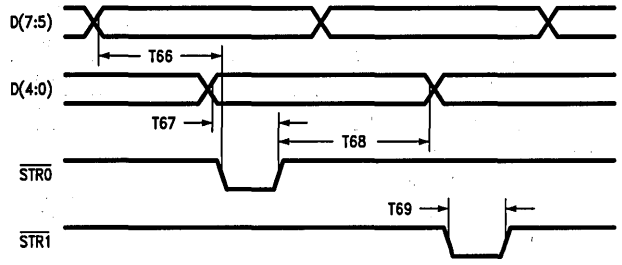
MLOAD TIMING



TL/F/11096–31

Number	Symbol	Parameter	Min	Max	Units
T61	mldats	Data Setup	10		ns
T62	mldath	Data Hold	10		ns
T63	mlabufa	MLOAD Active to BUFEN Active		35	ns
T64	mlibufi	MLOAD Inactive to BUFEN Inactive		30	ns
T65	mlw	MLOAD Width	80		ns

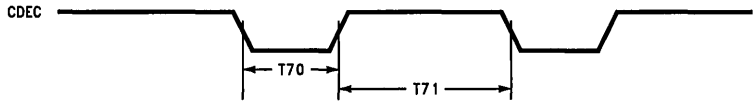
STROBE TIMING



TL/F/11096–32

Number	Symbol	Parameter	Min	Max	Units
T66	stradrs	Strobe Address Setup	135	170	ns
T67	strdats	Strobe Data Setup	40	65	ns
T68	strdath	Strobe Data Hold	135	160	ns
T69	strw	Strobe Width	30	65	ns

CDEC TIMING

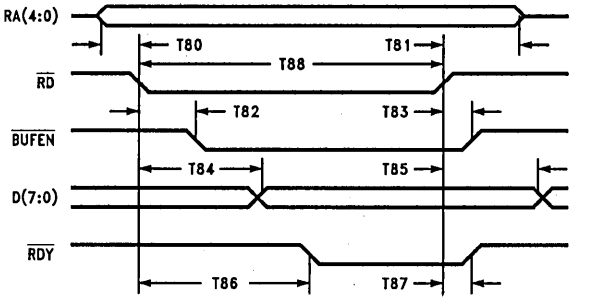


TL/F/11096–41

Number	Symbol	Parameter	Min	Max	Units
T70	cdecpw	CDEC Pulse Width	20	100	ns
T71	cdeccdec	CDEC to CDEC Width	200		ns

9.0 AC and DC Specifications (Continued)

REGISTER READ TIMING



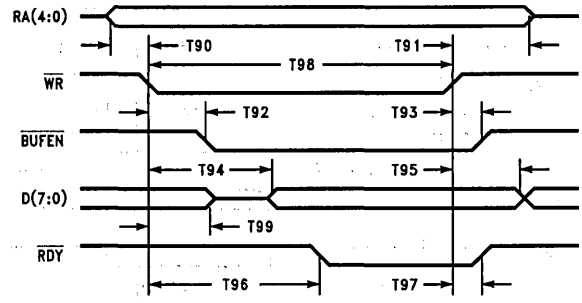
TL/F/11096–33

Number	Symbol	Parameter	Min	Max	Units
T80	rdadrs	Read Address Setup	0		ns
T81	rdadrh	Read Address Hold	0		ns
T82	rdabufa	Read Active to /BUFEN Active	105	340	ns
T83	rdibufi	Read Inactive to /BUFEN Inactive		15	ns
T84	rdadatv	Read Active to Data Invalid	245		ns
T85	rddath	Read Data Hold	75		ns
T86	rdardya	Read Active to /RDY Active	350	580	ns
T87	rdirdyi	Read Inactive to /RDY Inactive		10	ns
T88	rdw	Read Width	355		ns

Note: Minimum high time between read/write cycles is 25 ns.

9.0 AC and DC Specifications (Continued)

REGISTER WRITE TIMING



TL/F/11096–34

Number	Symbol	Parameter	Min	Max	Units
T90	wradrs	Write Address Setup	0		ns
T91	wradrh	Write Address Hold	0		ns
T92	wrabufa	Write Active to BUFEN Active	105	340	ns
T93	wribufi	Write Inactive to BUFEN Inactive		15	ns
T94	wradatv	Write Active to Data Valid		275	ns
T95	wrdath	Write Data Hold	−20		ns
T96	wrardya	Write Active to RDY Active	350	580	ns
T97	wrirdyi	Write Inactive to RDY Inactive		10	ns
T98	wrw	Write Width	355		ns
T99	wradt	Write Active to Data TRI-STATE		340	ns

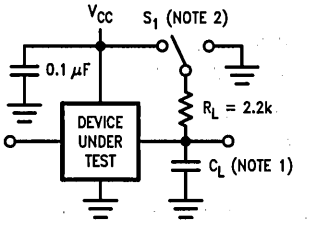
Note: Assuming zero propagation delay on external buffer.

Note: Minimum high time between read/write cycles is 25 ns.

10.0 AC Timing Test Conditions

All specifications are valid only if the mandatory isolation is employed and all differential signals are taken to be at the AUI side of the pulse transformer.

Input Pulse Levels (TTL/CMOS)	GND to 3.0V
Input Rise and Fall Times (TTL/CMOS)	5 ns
Input and Output Reference Levels (TTL/CMOS)	1.5V
Input Pulse Levels (Diff.)	−350 to −1315 mV
Input and Output Reference Levels (Diff.)	50% Point of the Differential
TRI-STATE Reference Levels	Float (ΔV) ±0.5V
Output Load (See *Figure* Below)	



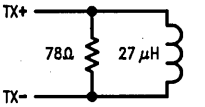
TL/F/11096–36

Note 1: 100 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.
S1 = V_{CC} for V_{OL} test.
S1 = GND for V_{OH} test.
S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.
S1 = GND for High Impedance to active high and active high to High Impedance measurements.

Capacitance T_A = 25°C, f = 1 MHz

Symbol	Parameter	Typ	Units
C_{IN}	Input Capacitance	7	pF
C_{OUT}	Output Capacitance	7	pF



TL/F/11096–37

Note: In the above diagram, the TX+ and TX− signals are taken from the AUI side of the isolation (pulse transformer). The pulse transformer used for all testing is the Pulse Engineering PE64103.

National Semiconductor

ADVANCE INFORMATION

LERIC™ Low End Repeater Interface Controller

General Description

The LERIC Low End Repeater Interface Controller can be used to ideally implement either a 10Base-T, coax or mixed 10Base-T/coax entry level mutliport repeater, in complete accordance with the IEEE 802.3 repeater specification. In addition to providing the functionality of an IEEE 802.3 repeater, the LERIC Controller also integrates 6 transceivers for 10Base-T, 1 AUI port, a cascading bus, and an LED interface.

When employed in a stand alone 10Base-T hub, as depicted below, the LERIC Controller provides an interface for a real AUI cable, as well as a cascade bus for the purpose of expanding the repeater by using multiple LERIC Controllers. Also shown below is the DP83955's ability to support a comprehensive LED display of network statistics, as well as provide for user configurability.

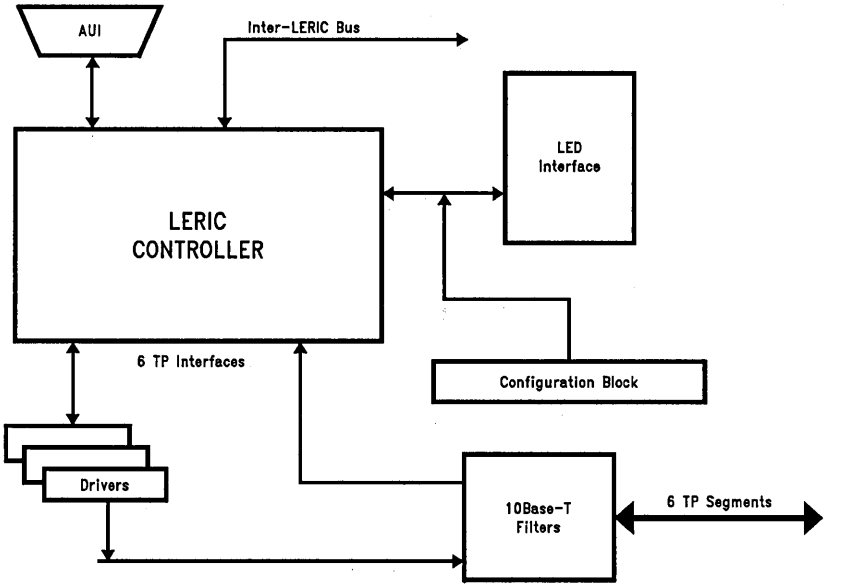
In the case of a coaxial multiport repeater, the LERIC Controller can be configured to bypass its internal 10Base-T transceivers. This allows for the DP83955 to be interfaced to 6 coaxial transceivers, as well as an AUI cable. It should be noted that the LED interface and the configuration capabilities are still available in the coax mode.

Finally, the LERIC Controller can also be employed in multiport repeaters that incorporate more than 6 10Base-T or coaxial ports. In order to address these applications, the LERIC Controller provides a cascading bus. This bus will enable multiple DP83955s to be connected and operate as a single logical multiport repeater.

Features

- Compliant with IEEE 802.3 repeater specification
- 7 network connections per chip
  - — 1 AUI port
  - — 6 selectable integrated 10Base-T transceivers
- Cascade bus to increase hub size
- On chip elasticity buffer, Manchester encoder and decoder
- Separate partition state machines for each port
- Provides port status information for LED displays
- Power-on configuration options
- CMOS process for low power dissipation
- 84-pin PLCC
- Single 5V power supply

System Diagram



TL/F/11240–1

DP83950EB-AT IEEE 802.3 Multi-Port Repeater Evaluation Kit

National Semiconductor
Application Note 781
Imad Ayoub

1.0 INTRODUCTION

The DP83950EB-AT is a three board kit (Main board, Display Assembly board and Backplane board) that forms an IEEE 802.3 Section 9 Repeater. The Main board has twelve 10Base-T ports and one AUI port and up to four Main boards can be cascaded using the Backplane board to form a larger hub.

The Main board contains the DP83950 Repeater Interface Controller (RIC™), which fits into an IBM PC-AT and compatible computers. The Main board repeats packets, provides management information, updates the Status LEDs, and can be cascaded to other Main boards. The Display Assembly board provides a full set of status LEDs for monitoring the repeater activity, and it provides a breakout to convert the 50-pin connector (with the cable coming from the Main board) to twelve ISO8877 (RJ45) phone connectors. The Backplane board is used for cascading two or more Main boards or to connect to a modified DP839EB-ATS System Oriented Network Interface Controller (SONIC™) Network evaluation board or the new SONIC Network evaluation board, the DP83932EB-AT.

Using the evaluation software the user can read or write to the RIC registers and counters, change the configuration options, enable and disable several features of the RIC, and display graphics of the RIC activities. There are several switches and jumpers on the Main board that configure the board to avoid conflicts with other adapters already installed on the AT bus.

2.0 MAIN BOARD OVERVIEW

The block diagram for the Main board is shown in *Figure 1.* The Main board allows the user to exercise all the functions of the RIC while using the twelve 10Base-T ports and the AUI port (the 10Base2 option of the RIC cannot be exercised).

The Main board is designed to perform Mload (refer to the RIC Data Sheet for more information on Mload) through either the Mload Logic (hardware Mload), or through the AT Bus Interface (software Mload). The switches SW1, SW2 are used during the hardware Mload.

The Inter-RIC Arbitration Logic performs the arbitration when there are two or more boards cascaded using the Backplane board. The arbitration is performed when two or more RICs have reception to determine which Main board (i.e., which RIC) is higher in the arbitration chain. The result of the arbitration will be used by the main state diagram of the RIC to determine which port within the RIC has PORT N (or PORT M), as described in the RIC Data Sheet.

The board was designed to allow for choosing between serial and parallel arbitration, and performs the arbitration function accordingly. In the Serial arbitration mode, the RIC logic performs all the arbitration (no additional logic is needed). In the parallel arbitration mode external PALs and logic are required (see Section 2.1.2).

The Inter-RIC BUS Transceivers are used to interface the RIC to the Backplane BUS. The Backplane BUS includes the Inter-RIC signals (IRC, IRD, IRE), the Management signals (MRXC, MRXD, MCRS, PCOMP), the Arbitration and Control signals (ACKI, ACKO, ACTN, ANYXN, COLN) as well as the parallel arbitration vector ARB(3:0). The transceivers are an example of how to perform the transmitting and receiving function over a backplane Bus and interfacing to drive and sense pins on the RIC. The BTL transceivers used allow for long bus applications due to their fast propagation delays and separate bus grounds.

The External Decoder is an example of how the RIC can be configured to run with an external decoder. The Received Manchester data is passed on to the external decoder through the RXM pin, and the decoded NRZ data is sent back from the decoder to the RIC through the Inter-RIC pins IRE, IRC and IRD.

The LED information is sent to the Display board through a driver and a 25-pin ribbon cable.

The 10Base-T Interface includes the buffers, resistors, filters and transformers necessary to interface the RIC ports to the external TP media.

The AUI Interface includes the necessary isolation and resistors to interface to the AUI cable.

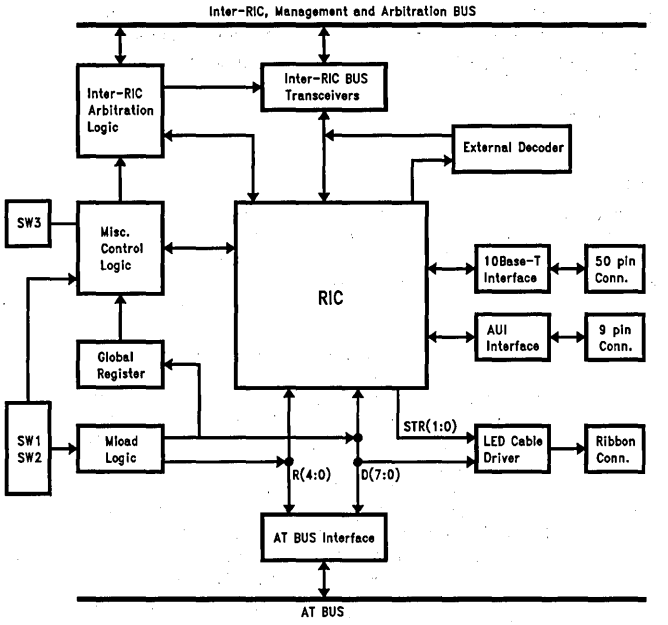
In order to enable using up to 16 boards in a PC without using an excessively large address space, all boards can be mapped to a single address block. A separate register is used to enable each individual board. This register is called the Global Register. The Global Register contains other control bits as shown below:

D6	D5	D4	D3	D2	D1	D0
EA2	EA1	EA0	RID3	RID2	RID1	RID0

Each board in a system is assigned a unique number by setting SW3. When the same number is loaded into the Global Register RID(3:0) as is set by SW3, the RIC Main board is enabled and the RIC's registers can be accessed.

The EA(2:0) bits are used to perform various functions on the selected board in the following manner:

EA2	EA1	EA0	Function
0	0	0	Normal Operation
0	0	1	Issue CDEC to All Boards
0	1	0	Issue CDEC to Selected Board
0	1	1	Issue MLOAD to All Boards
1	0	0	Issue MLOAD to Selected Board



TL/F/11230–1

FIGURE 1. Main Board Block Diagram

All 32 RIC registers and the Global register are IO mapped and can be relocated by setting BA(1:0) in SW1 as follows:

BA1	BA0	RIC Registers	Global Register
0	0	100 h–11F h	200 h
0	1	120 h–13F h	220 h
1	0	140 h–15F h	240 h
1	1	160 h–17F h	260 h

2.1 Detailed Description

2.1.1 Mload and AT Bus Interface

To perform the Mload pin configuration the board has the capability to load the D(7:0) and R(4:0) pins by either software or hardware.

Hardware Mload is done by the Mload Logic, which interfaces to pins D(7:0) and R(4:0) on the RIC to configure the RIC upon power up. Switches SW1 and SW2 allow for hardware setting of the Mload pin configuration. An RC network is used to provide a pulse (RSTB) at power up which will be used by a PAL (U41) to assert the Mload signal to the RIC. The PAL is needed to control the enables for the buffers for choosing between the hardware and software Mload.

Software Mload is implemented by passing the D(7:0) and R(4:0) signals from the PC-AT bus through the AT Bus Interface and onto the RIC pins.

When a Global Register write operation is performed by the PC-AT, it is written to a flip-flop (U44) which passes, first, the bits RID(3:0) to a comparator (U38), second, the bits EA(3:0) to the one of the control PALs. The comparator asserts a "RICHIT" if the Global Register RID matches the board *number.* This will enable the control PALs to perform the operation required by the Global register.

The ELI and RTI pins from the RIC can be passed onto the AT BUS to one of four interrupt request lines on the AT BUS (IRQ(15), IRQ(12), IRQ(11) or IRQ(10)) by selecting the appropriate jumper settings (JB1 and JB2, refer to schematic).

Three PALs (U36, U41, U43) are used to control the AT Bus Interface for software Mload and register Read/Write, and to enable the various buffers required for hardware Mload and the Global register decode. The PAL equations listings for all the PALs are included in Section 5.0 of this document.

U43 decodes the AT BUS address bits SA(9:0) and BA(1:0) to determine if the software operation is addressed to this board. A Global hit (Ghit) is asserted when a match occurs with the Main board's global *address.* A Base hit (Bhit) is asserted when a match occurs with a RIC *register.*

The AT BUS signals IOW, IOR, and AEN, and the Global register bits EA(2:0) are decoded by the PALs U36 and U41, resulting in the various control signals for the Mload buffers, the Read, Write and CDEC signals, the CHRDY signal to the PC-AT BUS, and the *receive* enable signals for the Inter-RIC and the management BTL transceivers (U4, U5).

2.1.2 Inter-RIC Arbitration Logic

Since there is no central arbiter, each Main board using the Inter-RIC BUS needs a way to tell if it owns the bus. This implementation uses one of two methods: serial or parallel arbitration (by setting JB3, refer to schematic).

In the serial arbitration method the RIC signals ACKI and ACKO are passed to and from the Backplane BUS directly (as SACKI and SACKO) without further arbitration. Therefore the serial arbitration is done by the RIC logic itself. A high level on ACKI tells the RIC that it can take the bus.

Therefore, the physical position of the board controls its priority in the chain. To participate properly in the chain the RIC will pass a high ACKI to ACKO if it does not want to transmit, but will force ACKO low if it does wish to transmit. In addition, ACKO needs to be held low if a low ACKI is seen, in order to tell the boards further down in the chain that they cannot take the bus.

The drawback for serial arbitration is that it requires all the Main boards to be inserted at all times, otherwise the SACKI/SACKO chain will be broken. The Backplane board is equipped with a jumper to connect these signals to continue the arbitration chain.

In the Parallel arbitration mode the SACKI/SACKO signals are controlled by two PALs according to the ACKI, ARBWIN and ENARB state machines shown in *Figures 2, 3* and *4*. A priority number is assigned via a set of dip switches SEL(3:0), where a higher number corresponds to a higher priority. Each board that wants to use the bus asserts this vector onto the bus. The bus is "wired-OR", so that only the board with the highest priority will see its own vector reflected back from the bus.

By monitoring the RIC signals and the arbitration vector, the parallel logic controls ACKI, so that normal functionality is maintained and the board's priority is independent of its physical location.

The arbitration takes place whenever a RIC is trying to take the bus, i.e., ACTNd is asserted, or when the RIC is experiencing a transmit collision, i.e., ANYXNd is asserted *(Figure 2)*. When either the ACTNd or ANYXNd are asserted the ENARB signal *(Figure 4)* is asserted enabling the arbitration BTL transceiver (U2).

A counter state machine is used in another PAL (U33) to give some delay for the arbitration to be completed, and asserts the arbitration done signal ARBDONE *(Figure 3)*. Based on the ACKI state machine the ACKIRIC signal is asserted to the RIC.

ACKI STATE MACHINE

Inputs: ACTND~, ANYXND~, ARBDONE, ARBWIN

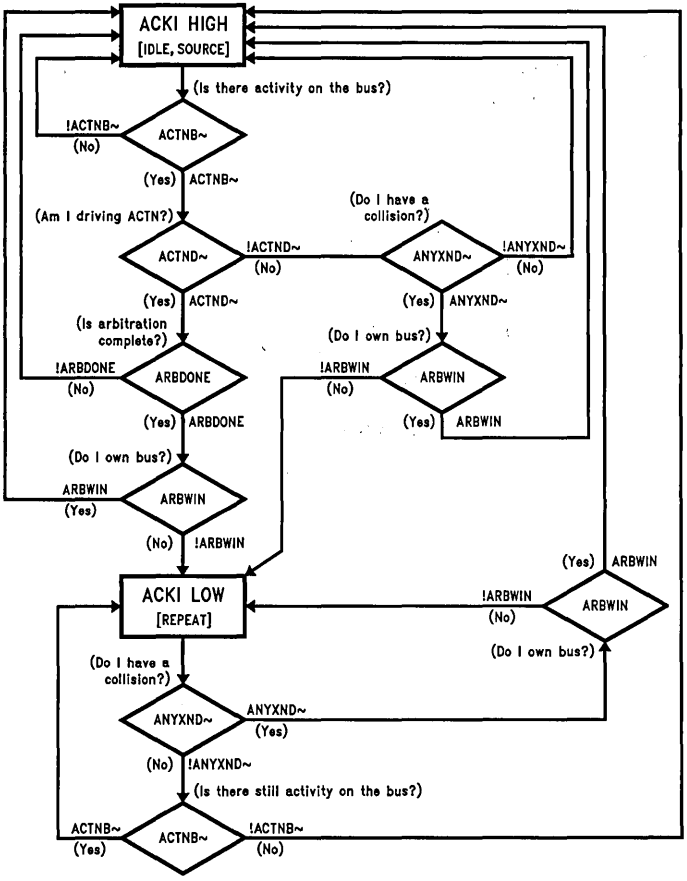
Output: ACKI

~ = Active Logic Low

! = Inactive

eg: !ACTNB~ = Inactive or Logic High

!ARBWIN = Inactive or Logic Low



TL/F/11230–2

FIGURE 2. ACKI State Machine

ARBWIN STATE MACHINE

Inputs: ANYXND~, ENARB, ARBDONE, BUSWIN~

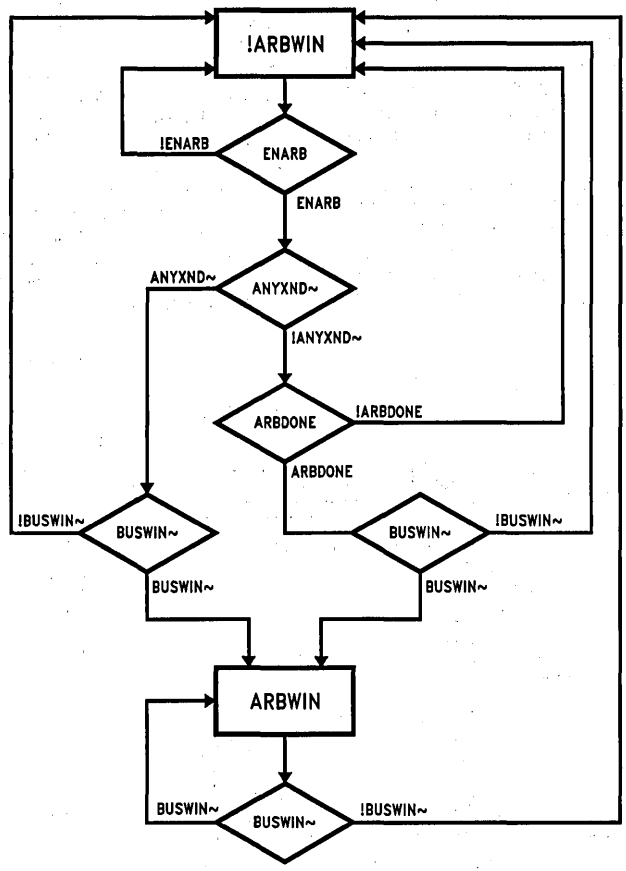
Output: ARBWIN

~ = Active Logic Low

! = Inactive

eg: !ANYXND~ = Inactive or Logic High

ANYXND = Active or Logic Low

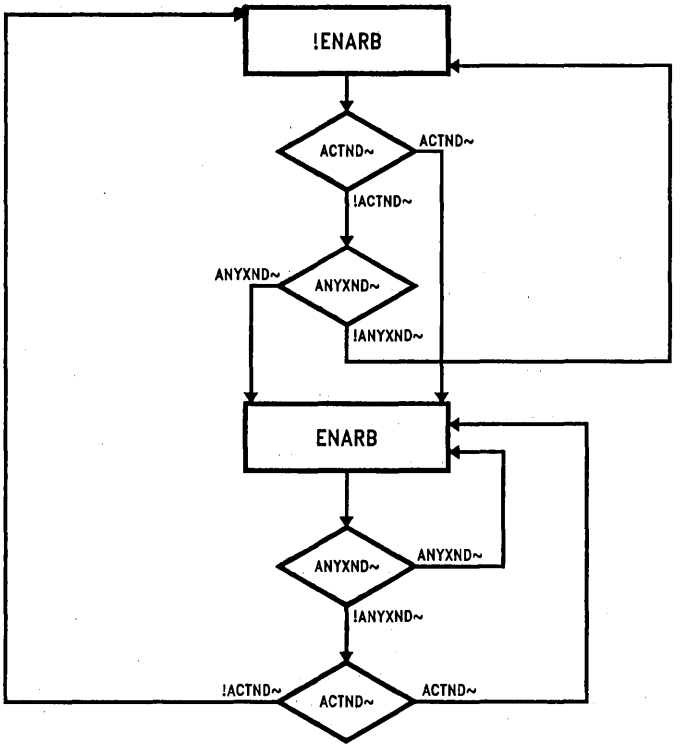


TL/F/11230–3

FIGURE 3. ARBWIN State Machine

ENARB STATE MACHINE
Inputs: ACTND~, ANYXND~
Output: ENARB
ENARB = ACTND~ # ANYXND~

~ = Active Logic Low
! = Inactive
eg: !ACTND~ = Inactive or Logic High
ACTND~ = Active or Logic Low



TL/F/11230–4

FIGURE 4. ENARD State Machine

2.1.3 Inter-RIC Bus Transceivers

To form a 10Base-T HUB with more than 12 Twisted Pair ports, up to four Main boards can be cascaded using the Backplane board (up to 16 Main boards can be cascaded with an extended Backplane board). The Backplane board can also be used to pass management information, as specified in the Hub management specification, from the management bus of the RIC to a modified DP839EB-ATS SONIC network evaluation board or the new SONIC Network evaluation board, the DP83932EB-AT.

To assure good speed and signal quality over the backplane bus four BTL (Turbo Transceivers) are used (U2, U3, U4, U5). The required BTL terminations are done on the Backplane board. Tying all the Ground pins of the BTLs together is not the optimum way to use these transceivers, however, it was necessary to assure proper operation when the Main board is in stand alone mode and the backplane board is not inserted. In a typical application were the Backplane BUS is always terminated these grounds would not be grounded together.

High level of assertion for the bidirectional "wired-OR" signals (ACTN, ANYXN, COLN, IRE, MCRS) of the RIC is required for proper operation with the inverting BTL transceivers. This makes these signals asserted low on the BUS side of the transceivers.

The parallel arbitration vectors ARBI(3:0) and ARBO(3:0) are transmitted and received over the BUS through one BTL Transceiver (U2). The receive enable for U2 is controlled by the stand alone (SLN) bit set during Mload. The drive enable for U2 is controlled by the enable arbitration (ENARB) signal from the arbitration PALs.

Another PAL (U3) transmits and receives the ACTN, ANYXN, and PCOMP signals onto the Backplane BUS. On the BUS side of U3, ACTN and ANYXN are bidirectional signals. They are asserted when any RIC asserts its ACTNd or ANYXNd signals. On the RIC side of U3, ACTN is split into ACTNd and ACTNs, and ANYXN is split into ANYXNd and ANYXNs. PCOMP is a unidirectional signal that is asserted onto the BUS by a separate controller board that can gather management statistics (or a modified DP839EB-ATS SONIC-AT board). The Drive enable for U3 is always enabled, allowing the ACTNd and ANYXNd signals to assert the BUS ACTN and ANYXN signals directly. The Receive enable is disabled only when the Main board is in the stand alone mode (i.e., there are no other Main boards in the HUB).

The RC network included on the ANYXNs signal is recommended (see Application Note #671 in the Interface Data Book for design details using BTL Transceivers).

A third PAL (U4) is used to transmit and receive the IRC, IRD, IRE and COLN signals. The COLN signal, which signals a receive collision, has no significance in the TP media. It is included here for completeness. These signals are bidirectional, however they are unidirectional at any one time. When the RIC is the receiving RIC, it asserts the packet enable signal PKEN signal which is used as the drive enable for U4. PKEN will be asserted as long as the RIC is the receiving RIC. The receive enable ENPKEN is asserted when the Main board is not in stand alone mode, and the PKEN signal is not asserted.

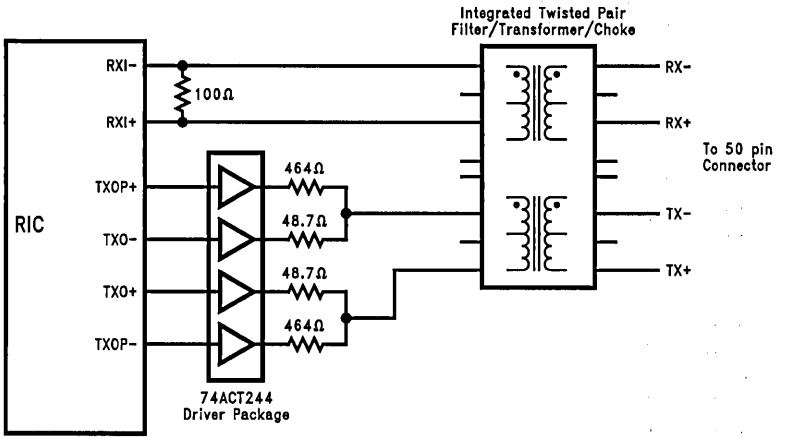
A fourth PAL (U5) is used to transmit the management bus signals MRXC, MCRS and MRXD. The MRXC and MRXC signals are unidirectional RIC output signals, while the MCRS is a bidirectional. The RIC senses the MCRS signal while the RIC is not the receiving RIC to assure the Interframe gap limit set in the RIC Inter-Frame Gap Threshold Select Register is not violated before sending another packet onto the Management BUS. The Management BUS information can be received by a SONIC, connected to the management bus. When the RIC is the receiving RIC, it asserts the management enable signal MEN, which is used as the drive enable for U5. MEN will be asserted as long as the RIC is the receiving RIC. The receive enable ENMEN is asserted when the Main board is not in stand alone mode, and the MEN signal is not asserted.

2.1.4 External Decoder

The RXM External pin decoder allows using the RIC with an external decoder. The RXM pin outputs the received Manchester Data from the RIC. This data is sent to the DP83910A decoder, which is decoded and passed onto the IRD, IRC and IRE BUS signals through a buffer back to the RIC. The buffer is enabled by PKEN, and a jumper is used to disable the buffer when using the internal decoder mode.

2.1.5 TP Interface

The interface is shown for one port in *Figure 5* below:



TL/F/11230–5

FIGURE 5. TP Interface

To drive the transmitted signal through 100 meters of Twisted Pair cable, the RIC requires external buffers. The resistor network on the transmit path shows the values used on the Main board. A more optimized network, which allows for better amplitude control is described in the TP Parametrics Evaluation document. The Filter/Transformer/Choke package used here is the PE65431 from Pulse Engineering. Other packages have been evaluated, and those results are described in the TP Parametrics Evaluation Document.

2.1.6 AUI Interface

The AUI includes the proper terminations and pulldowns, and the isolation transformer. A 9-pin connector is used instead of the standard 15-pin AUI connector (due to space limitations). A 9-pin to 15-pin special adapter cable is used to attach to the MAU.

3.0 DISPLAY BOARD DESCRIPTION

The Display board allows for the display of Maximum mode or Minimum mode LED configurations. The LED display address and data information RD(7:0), and the strobe signals STR(1:0) signals are received from the Main Board through the 25-pin ribbon cable and driver. The data is driven to two arrays of addressable latches and one flip-flop.

In the Maximum LED display mode all 66 LEDs are functional. Five sets of Latches are used and are arranged into five sets, with two latches per set. Each set controls one of the following groups of LEDs: Receive (REC), Collision (COL), Partition (PART), Good Link (GDLINK), and Bad Polarity (BDPOL).

The address bits for the latches are obtained from the RD(7:5) signals. On the top half of the array address 0 corresponds to the "any port", address 1 corresponds to the AUI port, and addresses 2 to 7 corresponds to ports 2 through 7. The top array is enabled by the STR0 signal. The bottom half is enabled by the STR1 signal, and addresses 0 through 5 correspond to ports 8 through 13.

The data is obtained from the RD(4:0) signals as follows: RD(0) for LINK, RD(1) for Collision, RD(2) for Receive, RD(3) for Partition, and RD(4) for Polarity.

In the Minimum LED display mode four LEDs are displayed: Any port collision (ACOL), Any port reception (AREC), Any port jabbering (JAB) and Any port partitioned (APRT), which indicate any activity on any of the RIC ports. In this mode the flip-flop ('ALS374) should be inserted into the socket (U6), which is left blank (default for the Maximum display Mode). The flip-flop passes the four LED signals to port 13, and STR0 is used as the clock.

4.0 BACKPLANE BOARD DESCRIPTION

This board forms the Backplane BUS for the HUB. There are four types of signals that are passed on this BUS.

1. The Inter-RIC BUS signals: IRE, IRC, IRE. These signals are passed from the Receiving RIC to all the other RICs in the HUB. They can also be used by a modified DP839EB-ATS SONIC board to allow the SONIC to transmit to the network through all the RICs on the HUB.
2. The Arbitration and Control signals: SACKI, SACKO, ACTN, ANYXN, COLN. These signals are passed between all the RICs on the HUB to assure the proper operation of the HUB per the IEEE802.3 state diagrams. When a board is inserted into a slot on the Backplane board, a jumper is removed to allow for the SACKI-SACKO signals to pass to and be asserted by the Main board. If there is no Main board inserted in a slot, that jumper should be inserted to short SACKI to SACKO, in order to complete the arbitration chain.
3. The Management BUS signals: MRXC, MRXD, MCRS, PCOMP. The MRXC, MRXD and MCRS signals are used to pass the management information from the receiving RIC to a SONIC board. PCOMP is a unidirectional signal that is asserted onto the BUS by a separate controller board that can gather management statistics to compress the data portion of the management information. The MCRS signal is also used as an input to the RIC as described in Section 2.1.3).
4. The Parallel arbitration vector, ARB(3:0), required for parallel arbitration (as described in Section 2.2.2).

Each of the BUS lines is terminated by approximately 20Ω (two 39Ω resistors in parallel).

5.0 PAL LISTINGS

```
U43  device 'p16L8';module RIC_DEC
title     'decode AT addresses


"inputs

sa0       pin  11;
sa1       pin  9;
sa2       pin  8;
sa3       pin  7;
sa4       pin  6;
sa5       pin  5;
sa6       pin  4;
sa7       pin  3;
sa8       pin  2;
sa9       pin  1;
ba0       pin  16;
ba1       pin  17;
aen       pin  18;
iow       pin  13;
ior       pin  14;

"outputs

ghit      pin 19;
bhit      pin 12;
io        pin 15;
base0 =  !aen & !sa9 & sa8 & !sa7 &  !ba1 & !ba0;
base1 =  !aen & !sa9 & sa8 & !sa7 &  !ba1 & ba0;
base2 =  !aen & !sa9 & sa8 & !sa7 &  ba1 & !ba0;
base3 =  !aen & !sa9 & sa8 & !sa7 &  ba1 & ba0;

low   =  !aen & sa9 & !sa8 & !sa7 & !sa4 & !sa3 & !sa2 & !sa1
      & !sa0;

equations

!ghit     =  low & !ba1 & !ba0 & !sa6 & !sa5
           # low & !ba1 & ba0 & !sa6 & sa5
           # low & ba1 & !ba0 & sa6 & !sa5
           # low & ba1 & ba0 & sa6 & sa5;

!bhit     =  !sa6 & !sa5 & base0
           # !sa6 & sa5 & base1
           # sa6 & !sa5 & base2
           # sa6 & sa5 & base3;

!io       = !iow # !ior;

END  RIC_DEC;
```

TL/F/11230–6

```
U36  device 'p20L8';ric_ctrl1
title 'ric control and some AT interface'

"inputs

     mloaddly  pin  23;
     ior       pin  14;
     iow       pin  13;
     rstdrv    pin  11;
     bufen     pin  10;
     rdy       pin   9;
     ea2       pin   8;
     ea1       pin   7;
     ea0       pin   6;
     richit    pin   5;
     ghit      pin   4;
     bhit      pin   3;
     io        pin   1;

"outputs

     chrdy     pin  21;
     ireg      pin  20;
     rd        pin  19;
     wr        pin  18;
     iorb      pin  17;
     dens      pin  16;
     cdec      pin  15;
     den       pin  22;

     norm      = !ea2 & !ea1 & !ea0;

equations

     !chrdy    = 1;

     enable chrdy  = !bhit & richit & rdy & !io;

     !dens     = !io & (!ghit # (!bhit & richit));

     !ireg     = !ghit & !iow;

     !rd       = richit & norm & !bhit & !ior;

     !wr       = richit & norm & !bhit & !iow;

     !iorb     = !ior;

     !cdec     = !ea2 & !ea1 & ea0 & !ghit & !ior
                 # !ea2 & ea1 & !ea0 & richit & !ghit & !ior;

     !den      = richit & !bhit & !bufen & norm & !io;


     end ric_ctrl1;
```

TL/F/11230–7

```
U41  device 'p20L8';ric_ctrl2;
title  'ric control 2a

"inputs

     mloaddly   pin 23;
     ior        pin 14;
     iow        pin 13;
     rstdrv     pin 11;
     bufen      pin 10;
     rdy        pin  9;
     ea2        pin  8;
     ea1        pin  7;
     ea0        pin  6;
     richit     pin  5;
     ghit       pin  4;
     bhit       pin  3;
     sln        pin  2;
     pken       pin  1;
     rstb       pin 16;
     men        pin 17;

"outputs

     swen       pin 21;
     raen       pin 20;
     rst        pin 19;
     mload      pin 18;
     enpken     pin 22;
     enmen      pin 15;

equations

     !swen      = !mload & !mloaddly;

     !raen      = mload & mloaddly;

     !rst       = rstdrv # !rstb;

     !mload     = rstdrv
                # ea2 & !ea1 & !ea0 & richit
                # !ea2 & ea1 & ea0
                # !rstb;

     !enpken    = !sln & !pken;

     !enmen     = !sln & !men;


     end ric_ctrl2;
```

TL/F/11230–8

```
U33  device 'p20v8r';module ric_ack
title 'ric arb state machine


"inputs

     enarb      pin  14;
     unused_1   pin  11;
     psel       pin  23;
     sln        pin  10;
     sacki      pin   9;
     actn_s     pin   8;
     actn_d     pin   7;
     anyxn_d    pin   6;
     unused_3   pin   5;
     unused_4   pin   4;
     unused_5   pin   3;
     match      pin   2;
     clk        pin   1;

"outputs

     ackiric    pin  15;
     arbwin     pin  18;
     arbdone    pin  20;
     q1         pin  21;
     q0         pin  22;
     Q20M       PIN  17;

"counter states
     s0 = ^b00;
     s1 = ^b01;
     s2 = ^b10;
     s3 = ^b11;

"counter modes
     mode = [enarb];
     count = [1];
     clear = [0];

state_diagram [q1,q0]

     state s0: case (mode == clear): s0;
                    (mode == count): s1;
          endcase;

     state s1: case (mode == clear): s0;
                    (mode == count): s2;
          endcase;

     state s2: case (mode == clear): s0;
                    (mode == count): s3;
          endcase;
```

TL/F/11230–9

```
     state s3: case (mode == clear): s0;
                    (mode == count): s3;
          endcase;

equations

arbdone = q1 & q0;

arbwin  = (!anyxn_d # arbdone) * match * enarb;

!ackiric = !sln & (psel & (ackiric & (actn_s & actn_d & arbdone &
          !arbwin # actn_s & !actn_d & anyxn_d & !arbwin)
          # !ackiric & (anyxn_d & !arbwin # !anyxn_d & actn_s))
          # !psel & !sacki);

!Q20M := Q20M;


end ric_ack;
```

TL/F/11230–10

```
U1  device 'p2018';module ric_arb
title 'arbitration pal


"inputs

     arbi0          pin  1;
     arbi1          pin  2;
     arbi2          pin  3;
     arbi3          pin  4;
     sel0           pin  5;
     sel1           pin  6;
     sel2           pin  7;
     sel3           pin  8;
     anyxn_d        pin  9;
     actn_d         pin 10;

"outputs

     match          pin 17;
     enarb          pin 22;
     arbo0          pin 18;
     arbo1          pin 19;
     arbo2          pin 20;
     arbo3          pin 21;

equations

     enarb     = actn_d # anyxn_d;

     match     = (!arbi3 # sel3) & (!arbi2 # sel2)
                 & (!arbi1 # sel1) & (!arbi0 # sel0);

     arbo3     = sel3;

     arbo2     = sel2 & (!arbi3 # sel3);

     arbo1     = sel1 & (!arbi3 # sel3) & (!arbi2 # sel2);

     arbo0     = sel0 &(!arbi3 # sel3) & (!arbi2 # sel2)
                 & (!arbi1 # sel1);


end ric_arb;
```

TL/F/11230–11

RIC™-SONIC™ Interface

National Semiconductor
Application Note 782
Imad Ayoub

INTRODUCTION

This document describes how the DP83950 Repeater Interface Controller (RIC) can be interfaced to a System Oriented Network Interface Controller (SONIC) controller. The emphasis in this note is on the hardware interface between the RIC and the SONIC. The software implementation of the Hub management protocols such as SNMP and CMIP are not discussed in this note, since each system's implementation would be different depending upon the processor used and the number of RICs and SONICs employed in the Hub. A description of the extra logic necessary to interface the RIC to a NIC (DP8390) is included for reference. And last, in order to provide a simple and fast solution for evaluating the RIC's management bus interface to the SONIC, a description of a simple way to hook the SONIC DP839EB-ATS evaluation board to the RIC's management Bus is included.

RIC-SONIC INTERFACE

The RIC transmits over the management bus every packet that is received from any of the ports (refer to the RIC datasheet for details). The management bus packet is different from the packets transmitted to/from the ports. First, the preamble on this bus is always five bits (01011). Second, at the end of the packet, after the CRC pattern, seven bytes of management status are appended to the packet by the RIC. These seven bytes are always aligned to start on a byte boundary. Third, the packet is in NRZ format.

A properly connected and configured SONIC receives every packet that is sent over the management bus, and therefore buffers the data as well as the seven bytes of status. The Packet Compression feature available on the RIC and the SONIC allows for specific handling of the data part of the packet, as described later.

Figure 1 shows the interface of one RIC to one SONIC. The SONIC is configured to run in the external decoder mode to receive the NRZ data from the management bus (refer to the SONIC datasheet for more details). The SONIC input pins CRS, RXC, and RXD tie directly to the RIC management bus output pins MCRS, MRXC and MRXD (with the RIC BINV selected for active high signals, refer to the RIC datasheet for details). The SONIC runs in Promiscuous Mode accepting all the packets from the management bus.

The packet compression output pin $\overline{\text{PCOMP}}$ of the SONIC ties directly to the $\overline{\text{PCOMP}}$ input pin of the RIC. The SONIC can be programmed to assert $\overline{\text{PCOMP}}$ upon a match or a mismatch of the packet destination address with a SONIC CAM address. For managed Hub applications, the SONIC asserts $\overline{\text{PCOMP}}$ if the destination address of the received packet does not match any address in the CAM of the SONIC. For managed bridge applications the SONIC asserts $\overline{\text{PCOMP}}$ if the destination address of the received packet matches any address in the CAM of the SONIC.

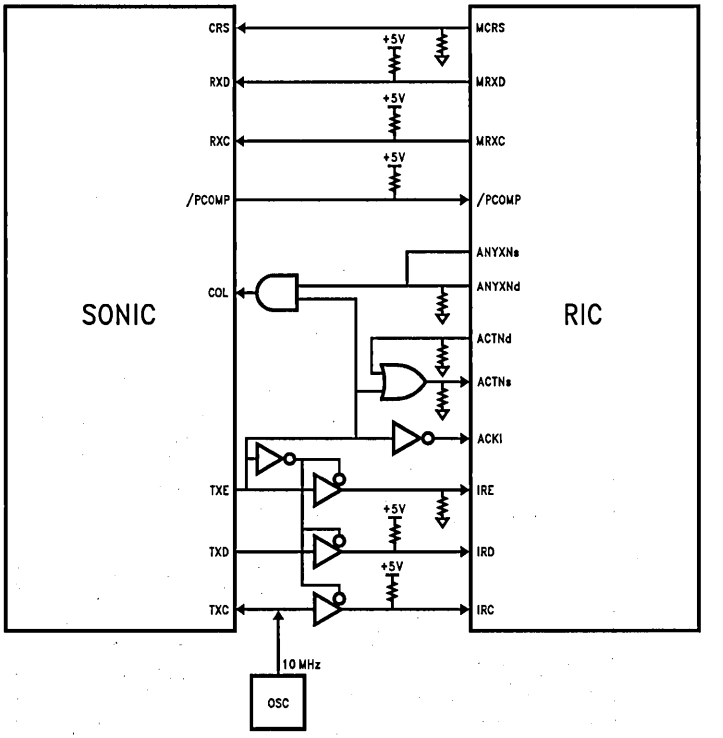
The managed Hub application is selected in this implementation. If the packet is addressed to the SONIC, the $\overline{\text{PCOMP}}$ pin will not be asserted by the SONIC and the RIC will not compress the data. The SONIC receives the whole packet with the seven bytes of status. If the packet is not addressed to the SONIC, the $\overline{\text{PCOMP}}$ pin will be asserted by the SONIC. The RIC will compress the data by inhibiting the clocks during the data part of the packet, and will re-enable the clock during the seven bytes of status.

The SONIC buffers the seven bytes of management status from the RIC to memory. These bytes can then be accessed by a processor. Utilizing the packet compression technique leads to an implementation that minimizes memory requirements, i.e., buffering only the data needed by the SONIC and the seven bytes of status. The RIC contains a Packet Compress Decode Register that can be used to determine the number of bytes, post SFD, which are transferred over the management bus when the packet compression option is employed.

To enable the SONIC to transmit to the network, the SONIC of *Figure 1* transmits a packet to the RIC through the Inter-RIC bus. The SONIC transmit signals TXE, TXD tie to the Inter-RIC signals IRE and IRD through a TRI-STATE® buffer (74F125), which is TRI-STATE when the SONIC is not transmitting. Since the SONIC is in external decoder mode, the TXC pin is an input. An external 10 MHz oscillator provides the input to the TXC pin of the SONIC and the IRC pin of the RIC. The SONIC will drive ACTN and ACKI of the RIC as soon as it wants to transmit. Driving ACTN informs the RIC that the SONIC wants to transmit. In this implementation the SONIC is placed on top of the arbitration chain with the RIC, therefore the SONIC drives the ACKI input of the RIC when it wants to transmit.

Since the seven bytes of status are appended after the CRC pattern, the SONIC will indicate that a CRC error occurs every time a packet is received. This should be ignored, and the SONIC should be set to save errored packets. The CRC bit in the seven bytes of status appended to the packet will indicate whether the packet has a CRC error or not.

The management bus does not experience any collisions, however any collisions on the network detected by the RIC are reported in the seven bytes of status. There will be no receive collisions on the SONIC, and the SONIC does not drive any collision signals to the RIC. The SONIC needs to be notified when a transmits collision occurs on the RIC. Therefore the COL input pin of the SONIC is driven by the ANYXNd output of the RIC whenever there is a transmit collision on the RIC and the SONIC is transmitting.



TL/F/11231–1

FIGURE 1

Figure 2 shows an implementation with several RICs sharing one SONIC for a managed Hub application. The SONIC is set in Promiscuous Mode to receive all packets, and it asserts $\overline{\text{PCOMP}}$ upon address mismatch. The Hub is addressable, and the SONIC can receive and transmit packets to the network as well as receive the seven bytes of RIC management status. All RICs share a single management bus to send the data to the SONIC. The SONIC transmits through the Inter-RIC bus to all RICs.

To interface the SONIC to the RIC in this implementation, a PAL is needed to generate the following signals:

$\overline{\text{ACKO}}$ = ACKI & TXE + $\overline{\text{ACKI}}$
ACTNd = ACKI & TXE
ANYXNd = TXE & $\overline{\text{ACKI}}$
COL = TXE & ANYXNs
TXEO = TXEI & ACKI

This implementation utilizes the serial arbitration method, and allows the SONIC to be placed anywhere in the arbitration chain. $\overline{\text{ACKO}}$ is asserted if the $\overline{\text{ACKI}}$ from the RIC above it is not asserted and the SONIC wants to transmit, i.e., TXE is asserted, or if $\overline{\text{ACKI}}$ from the RIC above it is asserted. ACTNd is asserted to tell all RICs that it wants to transmit when ACKI is not asserted and TXE is asserted. The SONIC could experience a transmit collision in this implementation since it could be in the middle of the arbitration chain. ANYXNd is asserted by the SONIC when TXE is asserted, and $\overline{\text{ACKI}}$ is asserted by any RIC higher in the arbitration chain. The SONIC is notified of a collision if it is transmitting and any RIC asserts ANYXN. Finally, TXEO is enabled when the SONIC wants to transmit and $\overline{\text{ACKI}}$ from the RIC above it is not asserted.

RIC-NIC INTERFACE

Any design that utilizes any controller other than the SONIC, such as the NIC (DP8390), to interface to the RIC should address the following points:

First, the packet compression feature of the RIC cannot be used by other controllers unless an external CAM and associated logic is used to generate the $\overline{\text{PCOMP}}$ signal to the RIC. If this logic is available the controller may not operate properly with the clock inhibited. The SONIC has an on board CAM, and asserts $\overline{\text{PCOMP}}$ to the RIC, and it works properly while the clocks are inhibited.

Second, due to the nature of the CSMA/CD protocol, there are situations when a collision will occur early in the packet (before SFD). This will lead to a packet transmitted onto the management bus containing only the seven bytes of status. This will be ignored by most controllers. Therefore extra logic will be required to stretch such packets to the controller's minimum acceptable packet length. The SONIC accepts such packets normally.

Third, knowing that the packet compression feature cannot be used, all packets that are transmitted over the network will need to be buffered by the controller. This requires a larger memory space, and may require a faster CPU.

Fourth, the SONIC will receive back to back packets from the management bus without missing packets due to insufficient gap (provided it is given access to memory). Other controllers may miss some packets if the gap is small.

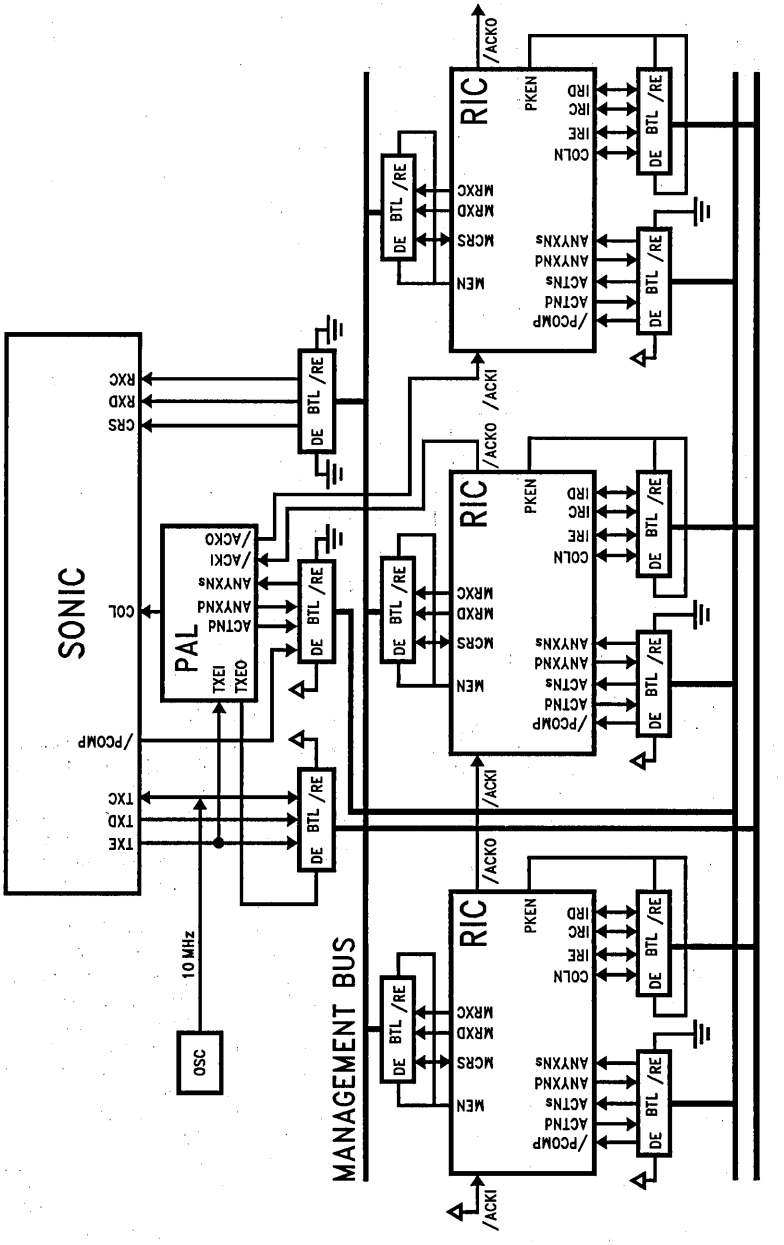

SONIC
RXC
RXD
CRS
COL
/PCOMP
TXC
TXD
TXE
PAL
TXEI
TXEO
ACTND
ANYXND
ANYXNs
/ACKI
/ACKO
10 MHz
OSC
MANAGEMENT BUS
RIC
DE
BTL
/RE
MEN
MCRS
MRXD
MRXC
COLN
IRE
IRC
IRD
PKEN
ACTNs
/ACKI
/ACKO
TL/F/11231–2

FIGURE 2

OTHER INTERFACE METHODS

The method described so far to interface the SONIC to the RIC's management and Inter-RIC busses is not the only way to interface a controller to the RIC. A controller could also transmit and receive packets through the Inter-RIC bus or through any of the ports. However these two methods do not allow the controller to obtain the management bus data from the RIC. There are several drawbacks for not receiving this data:

First, even though part of the information available in the seven bytes of status is available through the CPU bus of the RIC, the CRC error flag, the Collision Bit Timer, the Repeat Byte Count, and the Inter Frame Gap Bit Timer are not available from the RIC except through the management bus.

Second, every packet transmitted through the management bus contains the number of the port receiving the packet. If the management bus is not used, the only way to obtain the port number is by setting the RIC to generate a Real Time Interrupt to the processor on every packet received. The processor then reads the Real Time Interrupt register to find out which port received this packet.

Third, in a multi-RIC system, the RIC number is essential for associating the packets with the receiving RIC and receiving port. This is included in the seven bytes of management status, and cannot be obtained otherwise directly from the RIC.

Fourth, the packet sent over the management bus contains the Source and Destination addresses, and the Packet Compress Decode Register can be used to specify the number of bytes to send over the management bus before inhibiting the clocks when $\overline{\text{PCOMP}}$ is used. To perform this operation otherwise extra dedicated logic is needed to receive every packet on the network to read and save this data.

SONIC EVALUATION BOARD MODIFICATION (DP839EB-ATS ONLY)

This section describes a way to interface the SONIC to receive packets from the management bus of the RIC and to use the packet compression feature, using the Repeater Evaluation Kit (RICKIT) and a DP839EB-ATS board. A new SONIC evaluation board, the DP83932EB-AT, will not require modification. Contact your National Semiconductor representative regarding availability.

All that is needed for the SONIC to receive the management bus data is to tie CRS, RXO, RXD and $\overline{\text{PCOMP}}$ pins from the SONIC directly to the MCRS, MRXC, MRXD and $\overline{\text{PCOMP}}$ pins of the RIC (see *Figure 1*). This can be achieved as follows:

1. Place the DP839EB-ATS board in external decoder mode by removing the EXT jumper in the JB2 block, and removing all the jumpers in the JB4 block.
2. Take an SNI (DP8391, or DP83910) chip and clip off pins 2, 3, and 4, and place it in the appropriate socket (U18) on the DP839EB-ATS board.
3. Solder three wires to pins 2, 3, and 4 on the back of U18 on the DP839EB-ATS board and solder the other end to a female connector attached to the prototype area of the board.
4. To utilize the packet compression feature, use a SONIC (DP83932B) (pin 26 is the $\overline{\text{PCOMP}}$ pin). Bend pin 26 up in order for it to be accessible after inserting the SONIC back into the socket. Solder one end of a fourth wire to this pin and solder the other end to the fourth pin of the connector on the prototype area.
5. On the RICKIT (DP83950EB-AT) Main Board, solder four wires to the pin side of R31, R71, R40 and R36. Conect these wires to a female connecter. These four wires should be in the proper order to correspond to the proper pins from the DP839EB-ATS board.
6. Make a four wire ribbon cable that is approximately four inches long with a male pin connector at each end. This can now be used to connect between the two male connectors on the DP839EB-ATS board and the RICKIT Main Board.

The DP839EB-ATS board now has the proper modification to receive the management bus data from the RICKIT Main Board. These boards can now be installed into the same PC-AT, and operated using the Hub Manager (HM) software. See the HM software manual for details.

DP83950 Twisted Pair Parametric Evaluation

National Semiconductor
Application Note 783
Imad Ayoub
Prasun Paul

TWISTED PAIR PARAMETRIC EVALUATION

The following information lists the results of the Twisted Pair Parametric tests performed on the DP83950 Repeater Interface Controller (RIC™). The DP83950EB-AT Repeater Kit was used to perform the measurements. Four parts were evaluated at room temperature and 5V power supply, except where indicated.

The test results are divided into three areas; transmit, receive and miscellaneous. The tabular format used shows the parameter tested, the reference section and *Figures* of the "IEEE 802.3 10Base-T CSMA/CD Access Method and Physical Layer Specifications" document, and the values measured on the RIC. No details for the tests/setups are provided as they follow the IEEE document specifications for each test. Additional notes and tables are included for clarification where necessary.

National Semiconductor Corporation (NSC) does not guarantee any of the values indicated in this document. The parameters indicated in the AC/DC parameters section in the RIC data sheet are the ONLY parameters that are guaranteed by NSC.

Transmitter Specifications

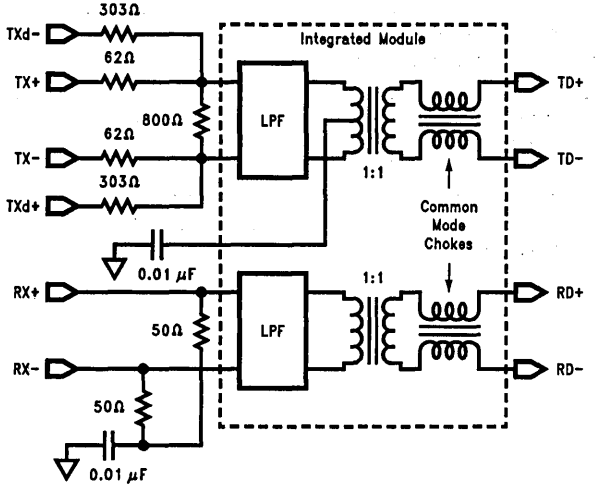
Test #	Parameter	IEEE Ref. Spec.	RIC Value/Comment
1	Peak differential output voltage: at TD± circuits terminated with a 100Ω load directly ±2.2V to ±2.8V	14.3.1.2.1	2.5V Peak (Note 1)
2	Harmonic contents with 10 MHz signal through the transmitter All harmonics should be ≥27 dB below the fundamental 10 MHz	14.3.1.2.1	Tested with a random signal, all harmonics were >30 dB below the fundamental signal
3	Output waveform with scaling Within *Figure 14-9* template	14.3.1.2.1	Waveforms are within template Measured values are shown in Tables Ia, Ib, Ic
4	Start of TP_IDL waveform with specified load in *Figure 14-11* and with or without cable model. The readings include idle high time and idle setting time Within *Figure 4-10* template	14.3.1.2.1	Waveforms are within template Measured values are shown in Table II
5	Link test pulse waveform, with specified load in *Figure 14-11* and with or without cable model. Readings include amplitude and pulse width Within *Figure 4-12* template	14.3.1.2.1	Waveforms are within template Measured values are shown in Table III
6	TD circuit differential output impedance or Return Loss spec. Reflection ≥15 dB below incident for all power on states and for impedances of 85Ω to 111Ω	14.3.1.2.2	Within spec. Measured values are shown in Table IV
7	TD output jitter: random signal through a 100m cable model terminated with a 100Ω load Equalized for max ±3.5 ns jitter at the end of cable model and with this equalization max ±8 ns while TD circuit is directly terminated with a 100Ω load	14.3.1.2.3	Within spec. Measured values are shown in Table V
8	Common mode to differential mode conversion. Test circuit as in *Figure 14-13* ≥29 − 17 log 10 (f/10) dB 1 < f < 20 MHz	14.3.1.2.4	Within spec. Measured values are shown in Table VI
9	TD circuit common mode output voltage. Test circuit is shown in *Figure 14-14* <50 mV peak	14.3.1.2.5	Within spec. (Note 2)

Transmitter Specifications (Continued)

Test #	Parameter	IEEE Ref. Spec.	RIC Value/Comment
10	TD short circuit current 300 mA max	14.3.1.2.7	Within spec. Approximately 0 mA
11	TD circuit common mode impulse withstand. Test circuit as in *Figure 14-15* Impulse E_{cm} applied 1000V min	14.3.1.2.7	Filter test—Guaranteed by filter manufacturer
12	TD silence voltage $\leq \pm 50$ mV	14.2.1.1	Within spec. 6 mV
13	Period of link pulses 16 ms ± 8 ms	14.2.1.1	16 ms
14	Transmit settling time	14.2.1.1	Within spec. Meets amplitude and jitter specifications (2nd bit on)
15	Power cycle behavior No extraneous signal on TD circuit	14.3.2.3	No extraneous signal on TD circuit where noticed

Note 1: The circuit used is shown in *Figure 1*. Three filters/transformer packages from three vendors were evaluated, and all of them met the amplitude required by this spec. The packages evaluated were: 1) Valor FL1012, 2) Pulse Engineering PE65431, 3) Bel Fuse 0556-3392-00

Note 2: The measurements were done on Valor FL1012, Valor PT3877, and Pulse Engineering PE65431. For all of these packages a 0.01 μF capacitor is required from the center tap to ground, as shown in *Figure 2*, to reduce common mode to within 50 mV.



TL/F/11232–1

FIGURE 2

Receiver Specifications

Test #	Parameter	IEEE Ref. Spec.	RIC Value/Comment
1	Signals accepted by RD circuits *Figures 14-16* and *14-17* templates	14.3.1.3.1	Test signals used did not include jitter Signals accepted met *14-17* and *14-16* templates
2	Jitter accepted by receiver ≥ ±13.5 ns	14.3.1.3.1	Guaranteed by 1 above
3	Jitter added by the receiver ≤ ±1.5 ns	14.3.1.3.1	Within spec. Approximatley 1.44 ns
4	RD circuit link test pulse acceptance *Figure 14-12* template	14.3.1.3.2	Within spec. Rejects <480 mV amplitude Accepts down to 35 ns width
5	Signals REJECTED by the receiver: a) Signals that will produce 300 mV peak signal at the output of a 3 pole test filter described in A.4.2 b) All sinusoidal signals of amplitude less than 6.2 V_{p-p} and frequency less than 2 MHz c) All sinusoidal single cycles of amplitude 6.2 V_{p-p} with 0° or 180° phase where the frequency is between 2 MHz to 15 MHz	14.3.1.3.2	Within the spec. Measured values are shown in Table VII
6	Idle detection by RD circuits Within 2.3 BT	14.3.1.3.3	Within spec. Within 2.05 BT
7	REC circuits differential input impedance or return loss Reflection ≥ 15 dB below incident for an impedance of 85Ω to 111Ω	14.3.1.3.4	Within spec. Measured values are shown in Table IV
8	RD short circuit fault tolerance Indefinite short shall be tolerable	14.3.1.3.6	RD short caused no faults
9	Receive delay	None*	40 ns
10	Bit loss and receive delay	None*	2.3 BT (270 ns − 40 ns)

*These are extra tests not specified in the standard.

Miscellaneous

Test #	Parameter	IEEE Ref. Spec.	RIC Value/Comment
1	Jabber timer	14.2.1.6	5 ms
2	Unjab time	14.2.1.6	Approx. 100 BT
3	Link loss timer 50 ms–150 ms (RIC set at 60 ms)	14.2.1.7	56 ms
4	Polarity correction: a) Inverted link pulses, b) Packets with inverted TP_IDL For both cases check if link pass state	None*	Functional
5	TX output at link fail No output data but link pulses	14.2.1.7	Functional
6	Receiver squelch level 300 mV–585 mV	Data Sheet	Within spec. (Note 1)
7	Receiver frequency acceptance: Input signal on RX± of 1.2V to 6.2V and sweep the frequency from 0 MHz to 30 MHz or higher	None*	Within spec. Accepts >3.61 MHz and up to 20 MHz (generator limit)
8	Power consumption	Data Sheet	I_{CC} max = 350 mA (Approx.)
9	Receive link_test_max timer 25 ms–150 ms (RIC: 32 ms)	14.2.1.7	Within spec. 32 ms
10	Recieve link_test_ min timer 2 ms–7 ms	14.2.1.7	Within spec. 5.75 ms
11	Link count: lc_max (RIC: 7 consecutive link counts	14.2.1.7	Functional

Note 1: With a SIN wave input:
Normal mode: Guaranteed on at 520 mV, guaranteed off at 460 mV.
Low squelch mode: Guaranteed on at 360 mV, guaranteed off at 260 mV. (For use with shielded TP and extended distances.)

* These are extra tests not specified in the standard.

TABLE Ia. Data at Different Points of the Transmit Signal at the End of the Cable Model

RIC#	Spec.		Port #2		Port #6		Port #13	
	Point	Value (V)	+ve TMPLT	−ve TMPLT	+ve TMPLT	−ve TMPLT	+ve TMPLT	−ve TMPLT
20	A	0	0	0	0	0	0	0
	B	1.0						
	C	0.4	0.75	0.6	0.58	0.66	0.75	0.78
	D	0.55	0.88	0.85	0.72	0.8	0.9	0.9
	E	0.45	0.74	0.83	0.58	0.86	0.83	0.78
	F	0	0.3	0.5	0.13	0.16	0.43	0.35
	G	−1.0	−0.45	−0.37	−0.7	−0.62	−0.46	−0.54
	H	0.7						
	I	0.6						
	J	0						
	K	−0.55	−0.96	−0.9	−1.024	−0.9	−0.97	−0.94
	L	−0.55	−0.96	−0.9	−1.024	−0.9	−0.97	−0.94
	M	0						
	N	1.0	0.8	0.9	1.0	0.9	0.78	0.78
	O	0.4						
	P	0.75						
	Q	0.15						
	R	0						
	S	−0.15						
	T	−1.0						
	U	−0.3	0	−0.13	0.26	−0.3	0.032	−0.06
	V	−0.7						
	W	−0.7	−0.6	−0.64	0.5	−0.43	−0.97	−0.58

TABLE Ic. Data at Different Points of the Transmit Signal at the End of the Cable Model

RIC#	Spec.		Port #2		Port #6	
	Point	Value (V)	+ve TMPLT	−ve TMPLT	+ve TMPLT	−ve TMPLT
22	A	0	0	0	0	0
	B	1.0				
	C	0.4	0.62	0.6	0.62	0.62
	D	0.55	0.74	0.85	0.62	0.62
	E	0.45	0.64	0.62	0.5	0.5
	F	0	0.26	0.18	0.04	0.04
	G	−1.0	−0.6	−0.7	−0.78	−0.78
	H	0.7				
	I	0.6				
	J	0				
	K	−0.55	−1.1	−1.024	−1.0	−1.0
	L	−0.55	−1.1	−1.024	−1.0	−1.0
	M	0				
	N	1.0	0.8	0.75	0.9	0.9
	O	0.4				
	P	0.75				
	Q	0.15				
	R	0				
	S	−0.15				
	T	−1.0				
	U	−0.3	0	0	0.14	0.14
	V	−0.7				
	W	−0.7	−0.62	−0.59	−0.38	−0.38

TABLE II. Start of TP_IDL Waveform

Test Load	Amplitude (V_P)	Width (ns)	Undershoot mV	@4.5 BT (mV)
155Ω // 180 μH with Cable Model	1.28	425	−220	−44
115Ω // 180 μH without Cable Model	1.5	431	−500	−36
76.8Ω // 229 μH with Cable Model	1.05	428	−120	−40
76.8Ω // 229 μH without Cable Model	1.27	438	−336	−32

TABLE III. Measurements of Different Corners of Link Pulses

Test Load	Amplitude (V)	Width at 0 to 0 Crossing (ns)	Width at 0/300 mV to 300 mV (ns)	Under-shoot (mV)	Amplitude at 4 BT (mV)	Amplitude at 42 BT (mV)
115Ω // 180 μH with Cable Model	1.6	333	176 ns at 300 mV to 300 mV	−80	−48	−12
115Ω // 180 μH without Cable Model	2.79	142.5	140 ns at 0 mV to 300 mV	−320	−100	−20
76.8Ω // 220 μH with Cable Model	1.32	340	164 ns at 300 mV to 300 mV	−56	−40	−14
7608Ω // 220 μH without Cable Model	2.26	158	152.5 ns at 0 mV to 300 mV	−240	−60	−16

TABLE IV. Return Loss on the Network

Port #	Receive		Transmit (Powered Up)	
	@ 5 MHz (dB)	@ 10 MHz (dB)	@ 5 MHz (dB)	@ 10 MHz (dB)
2	−31.9	−26.3	−34.3	−23.5
3	−39.5	−26.3	−32.5	−24.8
4	−31.3	−22.2	−33.3	−23.0
5	−35.4	−24.6	−38.9	−26.6
6	−35.2	−24.3	−34.0	−22.6
7	−30.1	−20.5	−36.0	−23.4
8	−29.7	−20.0	−26.8	−22.5
9	−30.7	−20.8	−31.5	−21.4
10	−30.6	−20.9	−31.7	−21.6
11	−32.0	−22.7	−36.1	−23.9
12	−34.5	−24.0	−31.7	−22.0
13	−30.5	−21.1	−34.0	−22.1

TABLE V. Transmit Signal Jitter at the End of a Cable Model

Filter	Jitter
Valor FL1012	±1.65 ns
Pulse Engineering PE65431	±1.60 ns
Bel Fuse 0556-3392-00	±2.05 ns

TABLE VI. Data for Transmitter Impedance Balance Test

Frequency MHz	29−17 $\log_{10}$ (f/10) dB	E_{cm} V_{p-p}	E_{dIff} V_{p-p}	20 $\log_{10}$ (E_{cm}/E_{dIff}) dB
1.0	46.0	10.2	28.8m	50.98
2.0	40.88	10.2	32.0m	50.0
3.0	37.88	10.2	35.0m	49.29
4.0	35.76	10.2	38.4m	48.48
5.0	34.18	10.0	41.6m	47.6
6.0	32.77	9.6	44.0m	46.7
7.0	31.63	9.4	46.4m	46.13
8.0	30.64	9.0	48.0m	45.46
9.0	29.72	8.4	48.0m	44.86
10.0	29.0	8.2	47.2m	44.79
11.0	28.29	8.8	84.8m	40.32
12.0	27.65	9.0	66.0m	42.69
13.0	27.06	8.6	38.0m	47.09
14.0	26.51	8.4	32.8	48.16
15.0	26.00	8.2	28.8m	49.0
16.0	25.52	8.0	26.8m	49.49
17.0	25.08	7.8	30.8m	48.07
18.0	24.66	7.6	29.6m	48.19
19.0	24.26	7.6	26.4m	49.18
20.0	23.88	7.4	21.6m	50.69

TABLE 7. Receiver Rejection Test Data

Test #	RIC #20		RIC #21		RIC #22	
	Port #5	Port #6	Port #5	Port #6	Port #5	Port #6
5 (a) @ 5 MHz	456 mVp	450 mVp	470 mVp	480 mVp	490 mVp	500 mVp
5 (a) @ 10 MHz	504 mVp	505 mVp	590 mVp	540 mVp	590 mVp	540 mVp
5 (b)	3.6 MHz	3.59 MHz	3.62 MHz	3.60 MHz	3.60 MHz	3.59 MHz

Section 4

Hardware and Software Support Products

Section 4 Contents

National Semiconductor

Ethernet Evaluation Hardware and Software Products, and 3rd Party Driver Developers

Overview

National Semiconductor provides a set of hardware and software platforms that allow quick and reliable evaluation of National's Ethernet products. These hardware and software platforms also provide a sound foundation for the development of end products.

National offers two types of example software packages. First, Evaluation Software, which comes with our evaluation boards, and second, Drivers for popular network operating systems. National's software platforms offer *key* example software and example drivers only to assist the designer in the development of drivers.

Please contact your local National Semiconductor Representative for availability of any Ethernet Evaluation Hardware and Software products.

1.0 Hardware

National's Ethernet evaluation boards can be divided into two categories. One set of boards is based on the DP8390 Network Interface Controller (NIC) architecture. The other set is based on the high performance DP83932 Systems Oriented Network Interface Controller (SONIC™).

1.1 DP83902EB-AT ST-NIC™ EVALUATION KIT

The DP83902EB-AT ST-NIC Evaluation Kit shows a complete, highly integrated Ethernet network interface design.

The DP83902EB-AT features Industry-Standard Silicon, the DP83902 Serial Network Interface Controller for Twisted Pair (ST-NIC) and the DP8392 Coaxial Transceiver Interface (CTI). The DP83902, a single-chip solution which meets all requirements of the IEEE 802.3 Specifications, combines the DP8390 Network Interface Controller, Manchester ENDEC, Attachment Unit Interface (AUI), and 10Base-T transceiver functions in low-power CMOS.

The ST-NIC Evaluation Kit demonstrates 10Base-T, 10Base2, and 10Base5 connections.

- 10Base-T: Twisted-Pair via RJ-45 connector
- 10Base2: Thin Coax via BNC connector
- 10Base5: Thick Coax via external AUI cable

Performance, flexibility, and full Ethernet functionality for 8- and 16-bit host bus network interface requirements are demonstrated for swift product development and reduced time to market.

- Novell Compatible
- Two-Third size, single-slot PC-AT adapter card and demonstration software
- Detailed documentation, including hardware and software manuals, installation guide, board schematics and bill of materials
- 5 LED network status indicators on board

The ST-NIC Evaluation Kit shows the benefits of increased reliability, reduced manufacturing and inventory requirements afforded by the single-chip ST-NIC design.

1.2 DP8390 and DP83901 BASED EVALUATION BOARDS

The original DP8390 NIC has several evaluation boards. All these boards have been discontinued except the original 8-bit evaluation card (DP839EB, see Table I) which is the basis for the design of many PC-based Network Interface cards (Like Novell's NE1000, but NOT identical).

The DP839EB-AT, DP839EB-MC, DP839EB-NB, and DP839EB-SE cards all utilize the DP8390. Though these have been discontinued, the information presented in these designs, and their drivers, are still useful. The MC and AT cards are implemented with a higher performance (and more expensive) shared RAM architecture. The DP839EB-NB and DP839EB-SE cards show how to implement Ethernet in a Macintosh environment, again using a shared RAM based design. (See Table VIII)

The DP839EB-ATN utilizes the DP83901 SNIC to provide a very low cost 16-bit I/O mapped design (very similar to Novell's NE2000).

The last two boards in Table I, are daughter boards. These are the DP839EB-DB and DP83902EB-DB. These boards enable customers using the DP8390, in the DIP package, to easily evaluate the DP83901 (SNIC) or the DP83902 (ST-NIC). These boards plug into DP8390–DP8391 sockets of an existing design, and allow the existing design to use the DP83901 or DP83902.

TABLE I. DP8390/DP83901/DP83902 BasedLAN Evaluation Boards

Part Number	Description
DP839EB	8-Bit IBM XT Compatible Ethernet Evaluation Board
DP839EB-ATN	16-Bit Low Cost PCAT Compatible Evaluation Board
DP83902EB-AT	ST-NIC PCAT Compatible Ethernet Evaluation Board
DP839EB-DB	SNIC Daughter Board
DP83902EB-DB	ST-NIC Daughter Board

1.3 SONIC™ BASED BOARDS

Table II shows the evaluation hardware for the 16-/32-bit DP83932 SONIC.

The DP839EB-ATS shows a simple high performance bus master interface for the AT. This card shows the basic features of the SONIC, but due to the slowness of the AT bus cannot show SONIC's ultimate performance.

The DP83932EB-AT replaces the DP839EB-ATS. This board uses an integrated bus interface from PLX Corp (AT 9010), and also adds an interface to connect to the DP83950EB-AT's management interface.

A better indicator of SONIC's performance (though the SONIC is still slowed) is the DP839EB-MCS which is a full 32-bit bus master card for IBM®'s PS/2® MICROCHANNEL® bus.

The DP839EB-EISA is a 32-bit bus master evaluation board for EISA bus based PCs. This adapter uses the PLX ES 9010 for the bus interface to EISA. This board provides full 33 MByte/sec bus performance in a cost effective design.

1.4 REPEATER EVALUATION BOARDS

DP83950EB-AT Evaluation Kit shows a complete IEEE 802.3 Multi-Port Repeater with Hub Management support. (See Table III)

The DP83950EB-AT Evaluation Kit features the DP83950 Repeater Interface Controller (RIC™), a single-chip solution which implements the requirements of the IEEE 802.3 Repeater, 10Base-T, and Hub Management Specifications in mixed-signal, low power CMOS.

- 12 Integrated 10Base-T transceivers and Repeater, Manchester ENDEC, and Attachment Unit Interface (AUI) functions
- Media independent—supports Twisted-Pair, Thin and Thick Coax connections
- Inter-RIC bus for cascading multiple RICs to create a single, logical repeater
- Hub Management data combined with network packets on a per port/per packet basis

Performance, flexibility, and Hub Management functionality for Multi-Port Repeater requirements demonstrated for swift product development and reduced time to market.

- Single-slot, standard-size PC-AT main board and backplane board for cascading multiple repeater cards
- 5 LED network status indicators per port on an external display board
- Demonstration software and detailed documentation, including, hardware and software manuals, installation guide, and board schematics and bill of materials

The RIC KIT shows the benefits of increased reliability and Multi-Port Repeater functionality and performance afforded by the single-chip RIC design.

2.0 Evaluation Software

Each board comes with Evaluation and Diagnostic software. This software provides example code for programming National's Ethernet Integrated circuit products (typically in Microsoft™ C). These programs can be used to evaluate the behavior and operation of National's products on a network. The evaluation software can be used and/or modified to facilitate the generation of hardware testing software. The evaluation software provided with the evaluation boards (as shown in Table III) can be requested separately.

TABLE II. DP83932 SONIC™ BasedLAN Evaluation Boards

Part Number	Description
DP839EB-ATS	16-Bit PCAT Bus Master Evaluation Board
DP839EB-MCS	32-Bit Micro Channel Bus Master Evaluation Board
DP83932EB-AT	16-Bit PCAT Bus Master Evaluation Board with Repeater Management Interface
DP83932EB-EISA	32-Bit EISA Bus Master Evaluation Board

TABLE III. LAN Software Availability

Product	Evaluation Software	NetWare 2.15	NetWare 3.11 (Note 1)	LAN Manager 2.0	FTP Packet Driver	Apple EtherTalk 2.0 (Note 2)	SCO UNIX386 V5 3.2
DP839EB	✓	✓		✓			
DP839EB-MC	✓	✓		✓			
DP839EB-AT	✓	✓		✓			
DP839EB-SE/NB						✓	
DP839EB-ATN DP83902EB-AT	✓	✓ (Note 3)	✓	✓	✓ (Note 4)		
DP839EB-MCS	✓	✓	✓	✓	✓		✓
DP839EB-ATS	✓	✓	✓	✓	✓		✓
DP83932EB-AT	✓	(Note 5)	(Note 5)	(Note 5)	(Note 5)		(Note 5)
DP83932EB-EISA	✓	(Note 5)	✓	(Note 5)	(Note 5)		(Note 5)
DP83950EB-AT	✓						

Note 1: Except OS/2 Drivers.

Note 2: Pre-System 7.0.

Note 3: Compatible with Novell NE2000 Drivers Supplied by Novell, no source code available.

Note 4: Public Domain Driver from Clarkston University.

Note 5: Contact your local National Semiconductor Representative for availability.

2.1 LAN DRIVER SOFTWARE

For each of National's Ethernet Evaluation boards, there are a set of drivers available (generally NOT supplied with the Evaluation Boards, but obtained separately). The available drivers are listed in Table III.

National develops a set of drivers for each of our evaluation boards which shows the operation and performance of the evaluation boards and provides example code for our Ethernet IC products which can be helpful in the development of similar products. These driver sets do not necessarily have the best possible performance, but do simplify a customer's development by providing reliable software examples. While the evaluation boards and their software examples may come very close to a production ready product, it is recommended that the hardware developer use them only as a sound foundation to extrapolate their end product.

2.2 GENERAL DRIVER INFORMATION

Three parameters are required to specify a useful driver or set of drivers. 1) A specific hardware implementation; 2) A networking operating system; 3) A computer operation system. If these are defined, then a specific software driver can be specified. For example, if the computer is a PC with the DP83932EB-AT, running NetWare 3.11 Net OS, and DOS OS, then a DOS ODI (Open Datalink Interface) driver is the specific driver specified. Another example, is a PS/2 with a SONIC Micro Channel card running OS/2, with Communication's Manager. This specifies and OS/2 NDIS (Network Driver Interface Specification) driver.

Depending on the Network Operating System, a different number of drivers are necessary to obtain a working network using a specific NOS. Table IV presents an example list of the drivers for each NOS.

Most NOS vendors have a developer program, that provides support and technical information to assist in the development of drivers. Address and phone information for some vendors is provided in Table VI. Also shown in Table V are the compiler/Assembler tools recommended to develop these software drivers.

TABLE IV. Driver Sets For Each National Supported Network OS

Operating System	NOS Drivers
NetWare 3.11/2.2	ODI Server ODI OS/2 Requester (Note 1) ODI DOS Workstation
NetWare 2.15	Server Workstation
LAN Manager	OS/2 NDIS (Note 2) DOS NDIS
SCO UNIX	Streams Driver (Note 3)
Appletalk	EtherTalk
PC/TCP (Note 3)	DOS Packet Driver

Note 1: At this time National does not provide this driver.

Note 2: NDIS Drivers are also supported by other networking packages like Banyan Vines, Sun PC/NFS.

Note 3: The SCO UNIX driver provides connectivity for a PC running SCO UNIX in a TCP/IP network. The packet driver enables DOS PC communication in a TCP/IP network.

TABLE V. Networking Operating System and Applications Vendors

Vendor	Software Name (Tools)
Novell Inc Independent Manufacturers Support Program, B-17-1 122 East 1700 South Provo, UT 84606 801-429-5713	NetWare (Phar Lap ASM for 3.1, Microsoft MASM for 2.15)
Microsoft Corporation System Software One Microsoft Way Redmond, WA 98052-6399 800-227-6444	LAN Manager (Microsoft MASM, Microsoft C)
Santa Cruz Operations SCO Developer Program 400 Encinal St. PO Box 1900 Santa Cruz, CA 95061 408-425-7222	SCO UNIX 386 V3.2 (UNIX Development System)
FTP Software PO Box 150 Kendall Square Branch Boston, MA 02142	DOS Based TCP/IP Software (Microsoft C)

2.2 NATIONAL SEMICONDUCTOR LAN DRIVER SOFTWARE DISTRIBUTION

Some Networking Operating System (NOS) vendors, require developers to sign up to a certification or developer program, which usually involves paying fees, and signing a licensing agreement. The license agreement that National has signed for the drivers listed in this document may have certain limitations on distribution of the source code. (Distribution of object or binary code is not restricted.) Table VI below highlights the requirements for National to distribute driver software.

In addition to the NOS vendor restrictions, National Semiconductor Corp. has an additional software license that must be agreed to before we distribute either source or executable code.

TABLE VI. Driver Source Code Distribution Requirements

Network OS	Source Code Distribution Requirements
Novell Network	Customer must supply signed Novell License agreement, or letter from Novell.
Microsoft LAN Manager	Customer must supply signed Microsoft License agreement, or letter from Microsoft.
SCO UNIX (LLI)	No restrictions on source code. Customer must sign non-disclosure with SCO to obtain driver specification only.
Apple Ethertalk	No Restrictions
FTP Software PC/TCP	No Restrictions

2.3 DRIVER CONTRACTORS AND DEVELOPERS

National Semiconductor develops drivers for LAN products through a combination of in-house programmers and external contractors. The list of contractors are shown below in Table VII with comments. These contractors are a good source of information for the products they developed and may be contacted directly for information or consulting services.

TABLE VII. LAN Driver Contractors

	Company and Address	Driver Products	Comments
1	**Ballard Synergy** 10601 S De Anza, Suite 212 Cupertino, CA 95014 408-257-8844	Novell Netware, UNIX, Other Drivers	Developed UNIX drivers for the DP839EB-ATS and DP839EB-MCS. Ballard-Synergy provides driver development support for National's Ethernet and FDDI products.
2	**The Impact Zone** 2823 Whipple Rd. Union City, CA 94587 415-489-6515	Novell Netware General Software	Software consulting company, that does general software development. Developed the Netware 286 and 386 drivers for the DP839EB-AT, -MC, -ATS and -MCS, and some Evaluation Software.
3	**GEE Technology** 6473 San Ignacio Ave San Jose, CA 95119 408-578-1123	Novell Netware, Ethertalk LAN Manager	Specializes in driver development primarily for NetWare, but also develops LAN Manager and Macintosh drivers. GEE developed the EtherTalk drivers for the DP839EB-NB and SE.

TABLE VIII. Discontinued Evaluation Boards

Part Number	Description
DP839EB-AT	16-Bit IBM PCAT Compatible Evaluation Board (Note 1)
DP839EB-MC	16-Bit IBM Micro Channel Compatible Evaluation Board
DP839EB-SE	16-Bit Apple Macintosh SE Compatible Evaluation Board
DP839EB-NB	32-Bit Apple Macintosh II Compatible Evaluation Board

Note 1: Replaced by DP839EB-ATN or DP83902EB-AT.

National Semiconductor

Ethernet Magnetics Vendors for 10BASE-T, 10BASE2, and 10BASE5

Enclosed is an overview of the magnetics components needed to interface National's Ethernet Products to each of the popular Ethernet cabling schemes.

10BASE-T TWISTED PAIR ETHERNET

This discusses the available components to interface National Semiconductor's 10BASE-T Ethernet LAN products to twisted-pair cable. The products offered by NSC include:

DP83922A Twisted Pair Interface (TPI)

DP83902 Serial Twisted Pair Network Interface Controller (ST-NIC™)

DP83950 Repeater Interface Controller (RIC™)

The types of solutions from these vendors vary and the designer is encouraged to contact these companies to obtain information on their various solutions. A brief overview of these products is presented here.

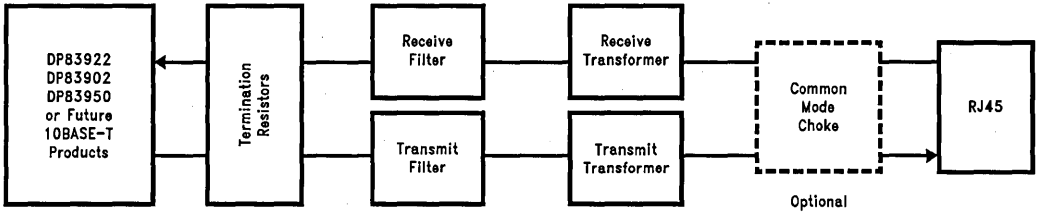
The interface from one of National Semiconductor's integrated circuits to the cable consists of the following blocks:

1. Termination resistors used to match the impedance of the twisted-pair interface to the cable.
2. Transmit and Receive Filters which are used to filter out receiver noise, and for the transmitter limit the harmonic content of the output waveform.
3. Transmit and Receive Pulse Transformers. These are required by the 10BASE-T standard to isolate the media from the data terminal equipment (DTE).
4. Optionally a common mode choke which is used to reduce common noise that could be emitted by the 10BASE-T interface. This may be necessary in some applications for meeting FCC or VDE EMI requirements as well as to meet 10BASE-T common mode output voltage noise specifications.

The components offered by the various manufacturers incorporate one or more or all of these. Table I shows some information on the available components. The components listed are primarily those that are more highly integrated. Also mostly DIP version part numbers of these devices are listed, however most vendors have surface mount versions of these products, as well as some products in SIP versions.

Table I is not a complete list of available components. The designer should use this list as a starting point for researching suitable products for his design. The addresses and phone numbers of the vendors listed are shown at the end of this paper in Table IV.

Twisted Pair Cable Interface Blocks



TL/F/11248–1

TABLE I. Partial List of 10BASE-T Transformer-Filter Products

Part Number	Filter	Transformer	Choke	Termination Resistor	NSC* Product	PIN Compatible	Tested** by NSC
PULSE ENGINEERING							
PE65423	✓	✓			922		✓
PE65421	✓	✓			902, 950	PT3877	
PE65431	✓	✓	✓		902, 950		✓
PE65424	✓	✓	✓		902, 950	PT3877	
PE65434	✓	✓	✓		902, 950		
VALOR							
PT3877	✓	✓			902, 950		✓
FL1010 (4 Channel)	✓	✓	✓	✓	950		
FL1012	✓	✓	✓†		902, 950	PT3877	✓
FL1020	✓	✓	✓	✓	902, 950		
PT3884	✓	✓		✓	922	PE65423	✓
BEL FUSE							
A556-2006-DE	✓	✓			902, 950	PT3877	✓
0556-3392-00	✓	✓	✓		902, 950		
0556-2006-00	✓	✓	✓†		902, 950	PT3877	✓
0556-2006-01	✓	✓	✓		902, 950		
0556-3899-02	✓	✓	✓	✓††	950		
A556-3899-00	✓	✓			922	PE65423	✓
FEE FIL-MAG							
78Z1120B-01	✓	✓			902, 950	PT3877	
78Z1120B-01	✓	✓	✓		902, 950	PT3877	
78Z1120B-04	✓	✓			922	PE65423	
78Z1120B-03	✓	✓			902, 950		
PCA ELECTRONICS							
EPA1533	✓	✓			902, 950	PT3877	
EPA1829	✓	✓	✓		902, 950	PT3877	

*902 = DP83902, 950 = DP83950, 922 = DP83922A.

**NSC has evaluated a sampling of 10BASE-T filter-transformer products for operation with the products listed in the NSC product column. Other products listed should provide suitable performance but have not been evaluated at this time. These products have been tested for compliance to the 10BASE-T standard when using National Semiconductor's integrated circuits. This testing includes waveshape, amplitude, jitter, harmonic content, insertion loss, and general interpretability. Testing for EMI has not been done as this varies dramatically between test setups and real applications. Contact National Semiconductor for further information.

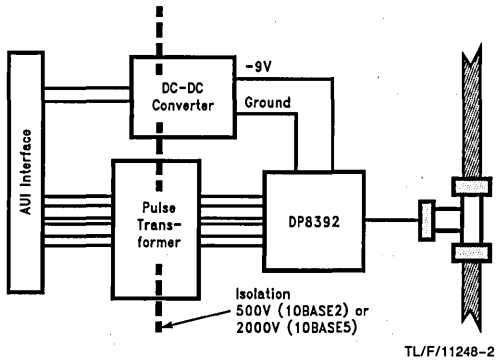
†There is a single common mode choke on the transmit channel only.

††In addition to resistors, this product internally includes a buffer that is required by the DP83950.

10BASE2 AND 10BASE5 THIN AND THICK ETHERNET

The interface for Thin (10BASE2) and Thick (10BASE5) Ethernet to the coaxial cable is nearly the same, and is illustrated by the block diagram in *Figure 2*. From the AUI (Attachment Unit Interface) to the coax cable there are three major blocks. A major requirement of the interface is the electrical isolation from the cable interface to the AUI. This isolation is requried to be 500V for 10BASE2, and 2000V for 10BASE5. Two of the three blocks provide this isolation.

Starting from the AUI, there are 4 pairs of wires. One pair provides power, and the other three are the data and collision signals. The signal pairs connect to a triple pulse transformer that provides voltage isolation. These signals then connect to the DP8392, Coax Transceiver Interface (CTI) which converts AUI signaling to coax transmission and reception signals.



TL/F/11248–2

FIGURE 2. Coax Ethernet Cable Interface Block Diagram

The DP8392 is powered from the AUI power pair which is fed from the AUI to a DC to DC Converter. The DC to DC Converter provides the voltage isolation of the power pair required by the IEEE standard, and in 10BASE5 converts the AUI 12V power to −9V required by the DP8392. (For 10BASE2 the DC to DC converter typically converts 5V to −9V).

Besides the DP8392 and a few discrete resistors and capacitors, the major additional magnetic components are the pulse transformer and the DC to DC converter. Both of these components are readily available from a number of sources. Table II shows a selection of manufacturers and their part numbers for Ethernet pulse transformers. All of these components are pin compatible and are available in 16-pin DIP package. Most of these manufacturers also have surface mount versions of these components.

Table III lists two vendors of DC to DC converters. As can be seen there are several different types of converters depending on input voltage and whether an enable function is desired.

Tables II and III are not a complete list of available components. The designer should use these lists as a starting point for researching suitable products for his design. The addresses and phone numbers of the vendors listed are shown at the end of this paper in Table IV.

TABLE II. Representative Ethernet Isolation Pulse Transformers for AUI (16-Pin DIP, Triple Transformer)

Inductance (Note 1)	Pulse Engineering	Valor Electronics	Bell Fuse (Note 2)	FEE Fil-Mag (Note 2)	PCA Electronics (Note 2)
50 μH	64101 (500V) 64106 (2 kV)	LT6001 (500V) LT6031 (2 kV)			EP9531-4
75 μH	64102 (500V) 64107 (2 kV)	LT6002 (500V) LT6032 (2 kV)	0553-0756-AB	23Z90	EP9531-5
100 μH	64103 (500V) 64108 (2 kV)	LT6003 (500V) LT6033 (2 kV)	0553-1006-AB	23Z91	EP9531-6
150 μH	64104 (500V) 64109 (2 kV)	LT6004 (500V) LT6034 (2 kV)		23Z92	EP9531-8
250 μH		LT6005 (500V) LT6035 (2 kV)			EP9531-11

Note 1: Generally inductances range from 35 μH to 300 μH, this table only shows the more commonly used values.

Note 2: Information provided by these manufacturers only listed 2 kV isolation components.

TABLE III. DC to DC Converters for the DP8392 (−9V Output, ≥200 mA)

	500V Isolation		2000V Isolation	
	+5V Input	+12V Input	+5V Input	+12V Input
Valor	PM7002 PM7102 (Note 1)	PM7104 PM7004 (Note 1)	PM7003 PM7103 (Note 1)	PM7105 PM7005 (Note 1)
(Switched)	PM9002	PM9003	PM9004	PM9005
PCA Electronics	EPC1000P (Note 1) EPC1015P (Note 1) EPC1007P	EPC1005P (Note 1) EPC1013P (Note 1) EPC1008P	EPC1000H (Note 1) EPC1015H (Note 1)	EPC1005H (Note 1) EPC1013H (Note 1)
(Switched)	EPC1002P (Note 1)		EPC1002H (Note 1)	
Bel Fuse	0740-0509-00 0740-0509-R0 (Note 1)	0740-1209-00 0740-1209-R0 (Note 1)	0740-0509-02 0740-0509-R2 (Note 1)	0740-1209-02 0740-1209-R2 (Note 1)

Note 1: These DC to DC Converters have a regulated output.

TABLE IV. Pulse Transformer Vendors

	Company and Address	Phone	FAX
1	**Pulse Engineering** P.O. Box 12235 San Diego, CA 92112	619-268-2400	619-268-2515
2	**Valor Electronics** 6275 Nancy Ridge Dr. San Diego, CA 92121	619-458-1471	619-458-0875
3	**Bel Fuse** 198 Van Vorst St. Jersey City, NJ 07302	201-432-0463	201-432-9542
4	**FEE Fil-Mag** 4787 Cardin Street San Diego, CA 92111-1416	619-569-6577	619-569-6073
5	**PCA Electronics** 16799 Schoenborn St. Sepulveda, CA 91343	818-892-0761	818-894-5791

Section 5

FDDI Products

Section 5 Contents

National Semiconductor

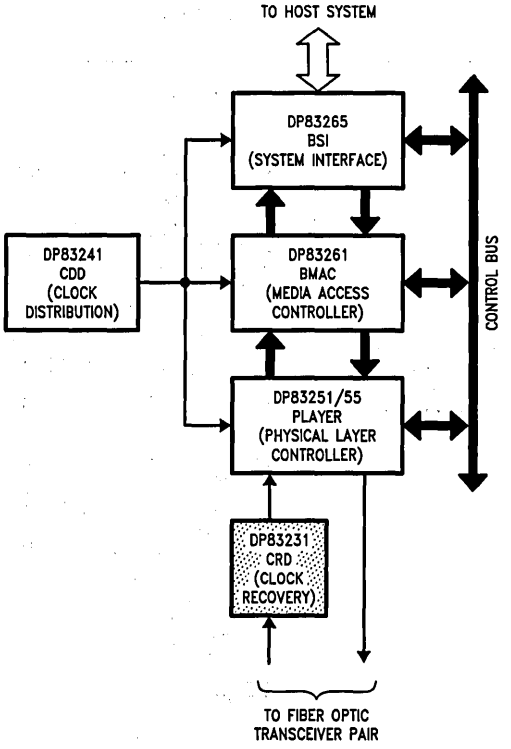
DP83231 CRD™ Device (FDDI Clock Recovery Device)

General Description

The DP83231 CRD device is a clock recovery device that has been designed for use in 100 Mbps FDDI (Fiber Distributed Data Interface) networks. The device receives serial data from a Fiber Optic Receiver in differential ECL NRZI 4B/5B group code format and outputs resynchronized NRZI received data and a 125 MHz received clock in differential ECL format for use by the DP83251/55 PLAYER™ device.

Features

- Clock recovery at 100 Mbps data rate
- Internal 250 MHz VCO
  - 0.1% VCO operating range
  - Crystal controlled
- Precision window centering delay line
- Single +5V supply
- 28-pin PLCC package
- BiCMOS processing



TL/F/10384–1

FIGURE 1-1. FDDI Chip Set Block Diagram

National Semiconductor

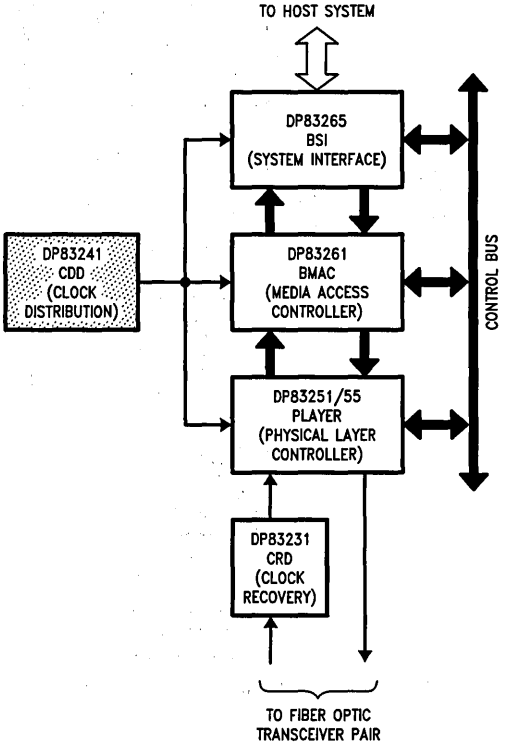
DP83241 CDD™ Device (FDDI Clock Distribution Device)

General Description

The CDD device is a clock generation and distribution device intended for use in FDDI (Fiber Distributed Data Interface) networks. The device provides the complete set of clocks required to convert byte wide data to serial format for fiber medium transmission and to move byte wide data between the PLAYER™ and BMAC™ devices in various station configurations. 12.5 MHz and 125 MHz differential ECL clocks are generated for the conversion of data to serial format and 12.5 MHz and 25 MHz TTL clocks are generated for the byte wide data transfers.

Features

- Provides 12.5 MHz and 25 MHz TTL clocks
- 12.5 MHz and 125 MHz ECL clocks
- 5 phase TTL local byte clocks eliminate clock skew problems in concentrators
- Internal VCO requires no varactors, coils or adjustments
- Option for use of High Q external VCO
- 125 MHz clock generated from a 12.5 MHz crystal
- External PLL synchronizing reference for concentrator configurations
- 28-pin PLCC package
- BiCMOS processing



TL/F/10385–1

FIGURE 1-1. FDDI Chip Set Block Diagram

National Semiconductor

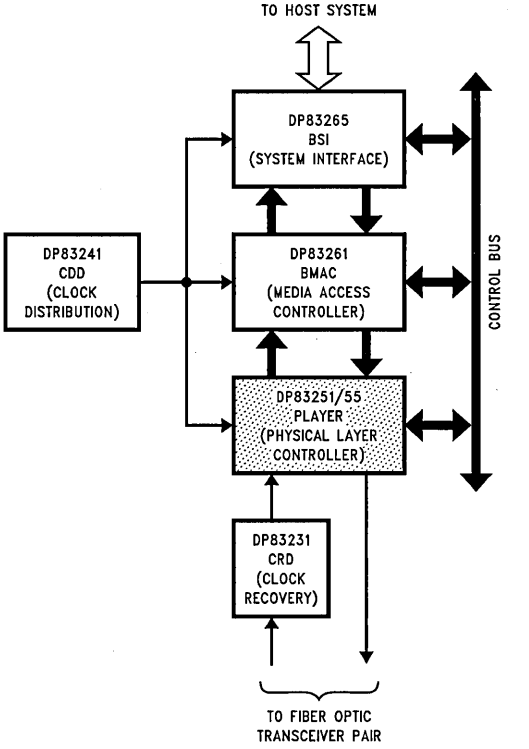
DP83251/55 PLAYER™ Device (FDDI Physical Layer Controller)

General Description

The DP83251/DP83255 PLAYER device implements one Physical Layer (PHY) entity as defined by the Fiber Distributed Data Interface (FDDI) ANSI X3T9.5 Standard. The PLAYER device contains NRZ/NRZI and 4B/5B encoders and decoders, serializer/deserializer, framing logic, elasticity buffer, line state detector/generator, link error detector, repeat filter, smoother, and configuration switch.

Features

- Low power CMOS-BIPOLAR process
- Single 5V supply
- Full duplex operation
- Separate management interface (Control Bus)
- Parity on PHY-MAC Interface and Control Bus Interface
- On-chip configuration switch
- Internal and external loopback
- DP83251 for single attach stations
- DP83255 for dual attach stations



TL/F/10386-1

FIGURE 1-1. FDDI Chip Set Block Diagram

5

National Semiconductor

DP83261 BMAC™ Device (FDDI Media Access Controller)

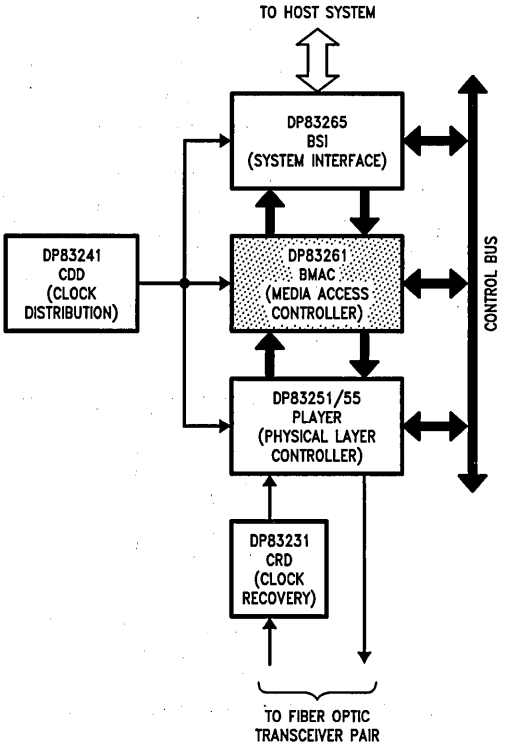
General Description

The DP83261 BMAC device implements the Media Access Control (MAC) protocol for operation in an FDDI token ring. The BMAC device provides a flexible interface to the BSI™ device. The BMAC device offers the capabilities described in the ANSI X3T9.5 MAC Standard and several functional enhancements allowed by the Standard.

The BMAC device transmits, receives, repeats, and strips tokens and frames. It uses a full duplex architecture that allows diagnostic transmission and self testing for error isolation. The duplex architecture also allows full duplex data service on point-to-point connections. Management software is also aided by an array of on chip statistical counters, and the ability to internally generate Claim and Beacon frames without program intervention. A multi-frame streaming interface is provided to the system interface device.

Features

- Full duplex operation with through parity
- Supports all FDDI ring scheduling classes (asynchronous, synchronous, restricted asynchronous, and immediate)
- Supports individual, group, short, long and external addressing
- Generates Beacon, Claim and Void frames without intervention
- Provides extensive ring and station statistics
- Provides extensions for MAC level bridging
- Provides separate management interface
- Uses low power microCMOS



TL/F/10387–1

FIGURE 1-1. FDDI Chip Set Block Diagram

National Semiconductor

PRELIMINARY

DP83265 BSI™ Device (FDDI System Interface)

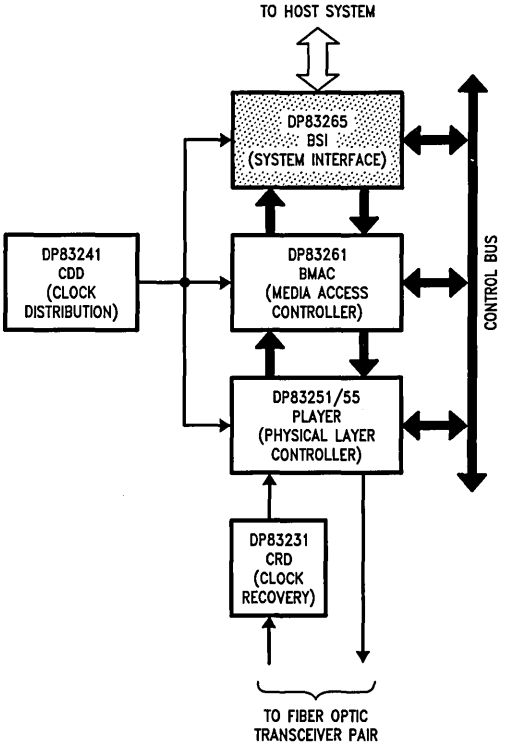
General Description

The DP83265 BSI device implements an interface between the National FDDI BMAC™ device and a host system. It provides a multi-frame, MAC-level interface to one or more MAC Users.

The BSI device accepts MAC User requests to receive and transmit multiple frames (Service Data Units). On reception (Indicate), it receives the byte stream from the BMAC device, packs it into 32-bit words and writes it to memory. On transmission (Request), it unpacks the 32-bit wide memory data and sends it a byte at a time to the BMAC device. The host software and the BSI device communicate via registers, descriptors, and an attention/notify scheme using clustered interrupts.

Features

- 32-bit wide Address/Data path with byte parity
- Programmable transfer burst sizes of 4 or 8 32-bit words
- Interfaces to low-cost DRAMs or directly to system bus
- 2 Output and 3 Input Channels
- Supports Header/Info splitting
- Bridging support
- Efficient data structures
- Programmable Big or Little Endian alignment
- Full Duplex data path allows transmission to self
- Confirmation status batching services
- Receive frame filtering services
- Operates from 12.5 MHz to 25 MHz synchronously with host system



TL/F/10791-1

FIGURE 1. FDDI Chip Set Block Diagram

Section 6

Glossary

Section 6 Contents

National Semiconductor

Ethernet and Networking Acronyms

Acronym	Description
ANSI	American National Standards Institute
CMIP	Common Management Information Protocol
CRC	Cyclic Redundancy Check
CRD	Clock Recovery Device—Part of National's Solution
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
CTI	Coaxial Transceiver Interface, NSC DP8392
DA	Destination Address
DAC	Dual Attach Concentrator
DAS	Dual Attach Station
DLL	Data Link Layer
ENDEC	ENcoder, DECoder combination, chip or function
FC	Frame Control
FCS	Frame Check Sequence
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Standards Organization
LAN	Local Area Network
LEM	Link Error Monitor
LERIC	Low End Repeater Interface Controller
LLC	Logical Link Control
MAC	Media Access Control Layer
MAU	Media Access Unit
MIB	Management Information Base

Acronym	Description
MPR	Multi-Port Repeater
NFS	Network File System
NIC	Network Interface Controller (add-in card); also NSC DP8390 Network Interface Controller
NOS	Network Operating System
NRZ	Non-Return to Zero
NRZI	Non-Return to Zero Invert on Ones
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PHY	Physical Layer
PMD	Physical Medium Dependent Layer
QLS	Quiet Line State
RIC	Repeater Interface Controller, NSC DP83950
SA	Source Address
SAS	Single Attach Station
SDU	Service Data Unit
SNA	Systems Network Architecture (IBM) Protocol
SNMP	Simple Network Management Protocol
SONIC	Systems Oriented Network Interface Controller, NSC DP83932
ST-NIC	Serial Network Interface Controller for Twisted Pair, NSC DP83902
TCP/IP	Transmission Control Protocol/Internet Protocol
WAN	Wide Area Network
XNS	Xerox Network System Protocol

National Semiconductor

Glossary of Local Area Networking and Data Communications Terms

If a term or acronym appears elsewhere in this databook and is not defined in the following glossary, please send your comments or inputs to:

Attn: LAN Applications
National Semiconductor MS-D3635
2900 Semiconductor Dr.
Santa Clara, CA 95952-8090
FAX: 408-739-6204

AARP (Apple Address Resolution Protocol): A protocol defined by Apple® for Appletalk® network similar in function to ARP.

ABI (Application Binary Interface): An application interface on AT&T's UNIX® System V Release 4. This interface enables binary compatibility for applications that run with UNIX on different platforms, and CPUs.

Access Method: In a data processing system, any of the techniques available to the user for moving data between main storage and an input/output device or channel. The techniques are usually part of the operating system.

ACK: Abbreviation for acknowledgement.

Acknowledgement: A response sent by a receiver to indicate successful reception of information. Acknowledgements may be implemented on any networking level.

Active Open: The operation that a client performs to establish a TCP connection with a server at a known address.

Address: A designator defining the ID of a DTE, peripheral device, or any other nodal component in a network. In Ethernet, each node is assigned a unique 6 byte address.

Address Resolution: Conversion of an Internet address into a corresponding physical address. This may require broadcasting on a local network. See ARP.

AFP (AppleTalk File Protocol): This is a network file system presentation layer protocol defined by Apple for use on Apple Macintosh® networks. It provides for access of remote file systems.

ALAP (AppleTalk Link Access Protocol): A Link level protocol that provides basic packet delivery transactions on Apple's Macintosh based networks.

ANSI: American National Standards Institute, an organization that sets information processing industry standards, represents the United States in the International Standards Organization (ISO).

ANS X3T9.5: The committee sponsored by ANSI which developed the standard for the Fiber Distributed Data Interface (FDDI).

APPLETALK: Originally defined by Apple Computer for Macintosh communication, AppleTalk protocols can now run over Ethernet networks, as well as the original, lower-speed AppleTalk network 230 Kbps (now called LocalTalk). AppleTalk is similar to NetBIOS™ in function and is supported by a number of vendors for communication between Macs and PCs, as well as simply between Macs.

AppleTalk was designed to be easy to use and requires little setup time compared to other networks.

APPC (Advanced Program to Program Communications): This is an interface that allows computers—and the programs running on them—to communicate over a network. APPC runs on IEEE 802.5, Ethernet, X.25 and SNA's synchronous data-link control. Although it was developed by IBM® and remains IBM-proprietary, work is under way within ISO to define an international protocol for "transaction programming" similar to APPC.

Application Layer: Level seven of the OSI model which provides the type of information transfer required, for example: file transfer or electronic messaging. See **OSI**.

API (Application Program Interface): A pre-defined software routine, that includes standardized and consistent interfaces to operating system functions, available for use by applications programmers designed to ensure, portability and accessibility to network resources.

ARCnet: A 2.5 Mbps baseband, token passing media-access protocol created by Datapoint Corp.

ARP (Address Resolution Protocol): Originally a Transmission Control Protocol/Internet Protocol (TCP/IP) process that maps IP addresses to Ethernet addresses; required by TCP/IP for use with Ethernet. Also referred to in other protocols as an address resolution protocol.

ARP HACK: See **Proxy ARP.**

ARQ (Automatic Request for Retransmission): A communications feature whereby the receiver asks the transmitter to resend a block or frame, generally due to errors detected by the receiver.

ASCII (American Standard Code for Information Interchange): A system used to represent alphanumeric data; a 7-bit-plus-parity character set established by ANSI and used for data communications and data processing; ASCII allows compatibility among data services; one of two such codes (see **EBCDIC**) used in data interchange, ASCII is normally used for asynchronous transmission.

Asynchronous Data Transmission: A mode of data transmission wherein the occurrence of each character is not related to a fixed time frame of reference. See Synchronous transmission.

ATP (Apple Transaction Protocol): A Transport layer protocol defined by Apple Computer. This protocol allows the reliable exchange of information between two processors on a Macintosh internet.

Attachment-Unit Interface (AUI): The interface between the Medium Attachment Unit (MAU) and the computing device or repeater.

Attenuation: The decrease in magnitude of the current, voltage or power of a signal transmitted over a wire, measured in decibels per kilometer. As attenuation increases, signal power decreases.

Backbone Network: High capacity network linking other networks of lower capacity. Example: FDDI as a backbone to multiple Ethernet and Token Ring LANs.

Backoff: In IEEE 802.3 networks when two or more nodes attempt a transmission and collide. The function of stopping transmission, and waiting a specified random time before retrying the transmission is considered backoff. In 802.3 networks a "truncated binary exponential backoff" algorithm is employed.

BALUN (Balanced/Unbalanced): In the IBM cabling system, refers to an impedance-matching device used to connect balanced twisted-pair cabling with unbalanced coaxial cables.

Bandwidth: 1. The difference, expressed in hertz, between the two limiting frequencies of a band. 2. The information capacity of a channel.

Baseband: A transmission scheme in which the entire bandwidth, or data-carrying capacity, of a medium (such as coaxial cable) is used to carry a single digital pulse (i.e., a signal) between multiple users. Because digital signals are not modulated, only one kind of data can be transmitted. Ethernet is among the most popular baseband LANs.

Baud: A unit of modulation rate or signaling speed used to designate the number of bits per second that can be transmitted in a given computer system.

Big-Endian: A binary data storage/transmission format in which the most significant byte (bit) comes first. DARPA Internet's standard is Big-Endian. See **Little-Endian.**

BISYNC (BSC): A family of IBM character-oriented binary synchronous communications protocols.

Bit: The smallest information unit in data processing. It has two possible states, "0" and "1". Bit is a contraction of BInary digiT.

Bit Duration: The time it takes one encoded bit to pass a point on the transmission medium; in serial communications, a relative unit of time measurement, used for comparison of delay times (e.g., propagation delay, access latency) where the data rate of a (typically high-speed) transmission channel can vary.

Bit-Oriented: Used to describe communications protocols in which control information may be coded in fields as small as a single bit.

BOOTP: A UNIX protocol that enables diskless workstations to boot their operating system from across a network.

BPS (Bits Per Second): The basic unit of measure for serial data transmission capacity; Kbps for kilo (thousands of) bits per second; Mbps for mega (millions of) bits per second.

Bridge: Network interconnection device operating at the Media Access Control (MAC) sublayer of the OSI model's Data Link layer which provides a communication path between logically or physically separate networks. A hardware/software device that permits high-speed communication between two local or remote networks with similar or dissimilar protocols. Provides packet filtering across networks based on the Ethernet source and destination address fields. Two major bridge classifications supported are spanning tree and source routing.

Broadband: A means of transmission in which users are allocated different frequency channels and can therefore send data across a common path simultaneously. A data transmission scheme in which multiple signals share the bandwidth, or data-carrying capacity, of a media. This allows transmitting voice, data and video signals, for example, over a single cable, such as coaxial cable. Cable television uses broadband techniques to deliver several dozen channels over a single cable. See **Baseband.**

Broadcast: A method of transmitting messages to two or more stations simultaneously, such as over a bus-type local area network or by satellite; protocol mechanism whereby group and universal addressing is supported.

Buffering: The process of temporarily storing data in a software program or in RAM, to allow transmission devices to accommodate differences in data transmission rates.

Bus: A length of wire or a set of parallel wires. Units which wish to intercommunicate are all connected to the bus. There must be a means of determining when each can transmit to the bus (access control), and a common method of sending and receiving the data (protocol), which includes a means of addressing to determine which unit a piece of information is for. A transmission path or channel; an electrical connection, with one or more conductors, by which all attached devices receive all transmissions simultaneously.

Bus Topology: The physical layout of a Local Area Network in which each node or workstation is connected directly to a length of cable or set of parallel wires.

Byte-Oriented: Similar to bit-oriented; control information may be coded in fields of one-byte (character) length.

Cable Access Method: The technique used to arbitrate the use of the communications medium by granting access selectively. (e.g., token passing and CSMA/CD).

Cable Plant: The physical cabling connectors, splices and patch panels in an installation.

Caching: A data-retrieval technique that places often-used data, such as file-allocation tables, in a computer's random-access memory where it can be accessed quickly.

Carrier Sense: The ability of each node on an Ethernet LAN to detect any traffic on the channel.

Carrier Sense Multiple Access with Collision Detect (CSMA/CD): The technique by which nodes on an Ethernet LAN share the transmission channel. See also Multiple Access; Carrier Sense; and Collision Detect.

CATV: Cable television technology commonly employed by broadband LANs for signal distribution.

CCITT (Consultative Committee International Telegraph and Telephony): An international association that sets worldwide communications standard (e.g., V.21, V.22, X.25, X.25, etc.).

Circuit Switching: A method of communication whereby an electrical connection between calling and called stations is established on demand for exclusive use of the circuit until the connection is released.

Channel: A path for the transmission of information.

Character: Standard 8-bit unit representing a symbol, letter, number, or punctuation mark; generally means the same as byte.

Character-Oriented: A communications protocol or a transmission procedure that carries control information encoded in fields of one or more bytes; (compare with bit-oriented and byte-oriented).

Character-Oriented Windows (COW) Interface: An SAA compatible user interface for OS/2® applications.

Characteristic Impedance: The impedance termination of an (approximately) electrically uniform transmission line that minimizes reflections from the end of the line.

CHEAPERNET: Colloquial term for thin wire Ethernet, defined by IEEE 802.3 as 10Base2.

Checksum: The total of a group of data items or a segment of data that is used for error-checking purposes. Both numeric and alphabetic fields can be used in calculating a checksum, since the binary content of the data can be added. Just as a check digit tests the accuracy of a single number, a checksum serves to test an entire set of data which has been transmitted or stored. Checksums can detect single-bit errors and some multiple-bit errors.

Cluster: Several pieces of data terminal equipment (DTE) in close proximity such that it is easy to run cabling between them.

CMIP/CMIS: Common Management Information Protocol (CMIP) and Common Management Information Services (CMIS) are two OSI protocols that provide a standard way of managing an OSI network.

CMOT (CMIP/CMIS over TCP): Use of the ISO CMIP/CMIS network management protocols to manage devices in an internet environment. See **CMIP.**

Coaxial Cable: A transmission medium with a central copper-wire conductor surrounded by concentric layers of plastic/polyvinyl chloride, aluminum or aluminized mylar and a copper tube that acts as an insulator (ground) and source of shielding from electromagnetic and radio frequency interference (CMI/RFI). Two types of coaxial cable—known as "thick" and "thin" for their respective diameters—are used in Ethernet data transmission.

Collision Detection: The ability of a transmitting node on an Ethernet LAN to sense a change in the energy level of the channel and to interpret the phenomenon as a collision.

Communications Server: A hardware/software combination that allows terminals and host computers to access a network without implementing the necessary network protocols. The communications server communicates with other devices using standard built in protocols.

Compression: Any of several techniques that reduce the number of bits required to represent information in data transmission or storage (thus conserving bandwidth and/or memory), in which the original form of the information can be reconstructed; also called "compaction".

Concentrator: Any communications device that allows a shared transmission medium to accommodate more data sources that there are channels currently available within the transmission medium.

Connectionless Service: The packet delivery service offered by most hardware and by the Internet Protocol. It treats each packet or datagram as a separate entity that contains both the source and destination address. Connectionless service can lose packets or deliver them out of order.

Conditioning: Extra-cost options that users may apply to leased, or dedicated, voice-grade telephone lines in which line impedances are carefully balanced; will generally allow for higher-quality and/or higher-speed data transmission; in increasing order of resultant line quality and cost, conditioning may be C1, C2, C4, or D1; allows improved line performance with regard to frequency response and delay distortion.

Contention: In communications, the situation when multiple users vie for access to a transmission channel, whether a PBX circuit, a computer port, or a time slot, within a multiplexed digital facility.

Core: The central region of an optical waveguide through which light is transmitted; typically 8 to 12 microns in diameter for single mode fiber, and 50 to 100 microns for multimode fiber.

Core Gateway: One of a set of Internet gateways which exchange routing updates periodically to ensure consistency in routing because all groups must advertise their network paths to core gateways using Exterior Gateway Protocols (EGP).

CRC (Cyclic Redundancy Check): A basic error-checking mechanism for link-level data transmission; a characteristic link-level feature of (typically) bit-oriented data communications protocols. The data integrity of a received frame or packet is checked via a polynominal algorithm based on the content of the frame and then matched with the result that is performed by the sender and included in a (most often, 16-bit) field appended to the frame.

CSMA/CD (Carrier Sense Multiple Access with Collision Detection): A LAN protocol access method for in which the nodes are attached to a cable. When a node transmits data onto the network and raises the carrier, the remaining nodes detect the carrier, Carrier Sense, and "listen" for the information to detect if it is intended for them. The nodes have network access, Multiple Access, and can send if no other transmission is taking place. If two attempt to send simultaneously a collision takes place, Collision Detection and both must retry at random intervals.

CSNET (Computer Science Network): A network providing Internet connections and mail delivery service using dial-up. CSNET also provides an Internet domain name server for members who cannot run their own. CSNET was originally funded by the National Science Foundation, but is now self sufficient.

CSU (Channel Service Unit): A component of customer premises equipment used to terminate a digital circuit (such as DDS or T1) at the customer site; performs certain line-conditioning functions, ensures network compliance with

FCC rules, and responds to loopback commands from the central office, and ensures proper "ones" density in transmitted bit stream and corrects biploar violations. See **DSU**.

CTI (Coax Transceiver Interface): Ethernet coaxial cable driver/receiver which interfaces the code electronics to the physical medium. National's DP8392.

D4 Framing: A T1 12-frame format in which the 193rd bit is used for framing and signaling information; ESF is an equivalent but newer 24-frame technology.

DARPA (Defense Advanced Research Projects Agency): Formerly ARPA. A government agency that funded ARPANET and later, the DARPA Internet.

DARPA Internet: The collection of gateways and networks, including ARPANET, MILNET, and NSFnet, that use the TCP/IP protocol suite and operate as a single, virtual network providing reliable full duplex stream delivery and unreliable connectionless packet delivery. It also features universal connectivity and applications level services such as electronic mail.

DAS (Dual Attach Station): A device attached to both rings of an FDDI network.

Data Communications: The transmission, reception, and validation of data; data transfer between data source (origin node) and data link (destination node) via one or more data links according to appropriate protocols.

Datagram: A packet that includes a complete destination address along with the data it carries. A finite-length packet with sufficient information to be independently routed from source to destination. Datagram transmission typically does not involve end-to-end session establishment and may or may not entail delivery confirmation acknowledgement.

Data Link: 1. The physical means of connecting one location to another for the purpose of trasmitting and receiving data. 2. Synonymous with communication link. Any serial data-communications transmission path, generally between two adjacent nodes or devices and without intermediate switching nodes.

Data Link Layer: Second layer in the OSI model; the network processing entity that establishes, maintains, and releases data-link connections between (adjacent) elements in a network to enable transmission over the physical link. See **OSI**.

Data Terminal Equipment (DTE): The equipment that serves as a message source or a message destination and provides for the communication control function; subscriber equipment.

Data Transfer Rate: The average number of bits, characters, or blocks per unit of time transferred from a data source to a data link.

DCE (Data Circuit Terminating Equipment): Devices that provide the functions required to establish, maintain, and terminate a data transmission connection; e.g., a modem.

DDCMP (Digital Data Communication Message Protocol): Digital Equipment Corporation's link level protocol. It uses serial lines, delimits frames with special characters and includes link level checksums. NSFnet incorporates DDCMP over its backbone lines.

DDN (Defense Department Network): MILNET and associated parts of the DARPA Internet which connect to military installations. DDN provides both local and long-haul data communications and interconnectivity for the Department of Defense systems and follows the DoD protocol suite. DDN is sometimes used to refer to MILNET, ARPANET and the TCP/IP protocols that they use.

DDS (Dataphone Digital Service): A private-line digital service offered interLATA by BOCs and interLATA by AT&T Communications, with data rates typically at 2.4, 4.8, 9.6, and 56 Kbps; part of the services listed by AT&T under the Accunet family.

DDS-SC: Dataphone® Digital Service with Secondary Channel; also referred to as DDS II. A tariffed private-line service offered by AT&T and certain BOCs that allows 64 Kbps clear-channel data with a secondary channel that provides end-to-end supervisory, diagnostic, and control functions.

DECnet®: Digital Equipment Corporation's proprietary network architecture developed for use in WAN and includes significant Ethernet LAN capabilities, endowed with a peer-to-peer methodology.

Dedicated Line: A dedicated circuit, a nonswitched channel; also called a private line. See **Leased line.**

Delay: In communications, the time between two events; See **Propagation delay, response time.**

Demultiplexor: A hardware device which separates a single signal from a transmission line into several signals based on time or carrier frequency. It is used on broadband systems in combination with a multiplexor to allow multiple, simultaneous signal transmissions over a single medium. It allows multiple hardware devices to use a single communication link at the same time.

DES (Data Encryption Standard): A scheme approved by the National Bureau of Standards that encrypts data for security purposes. DES is the data-communications encryption standard specified by Federal Information Processing Systems (FIPS) Publication 46.

Destination Field: A field in a message header that contains the address of the station to which a message is being directed.

Destination Node: A network node to which a message is addressed.

Disk/File Server: A mass storage device that can be accessed by several computers; enables the creation, storage, and sharing of files.

Disk Server: A network device which usually gives dedicated non-shared space on a disk drive to client hosts.

Distributed File Server: A system by which file systems on disks distributed throughout a network are made available to workstations distributed throughout the network.

Distribution Frame: A wall-mounted structure for terminating telephone wiring, usually the permanent wires from or at the telephone central office, where cross connections are readily made to extensions; also called distribution block.

DLC (Data Link Control): The set of rules (protocol) used by two nodes, or stations on a network to perform an orderly exchange of information. A data link includes the physical transmission medium, the protocol and associated devices and programs so it is both a physical and a logical link.

DMA (Direct Memory Access): A technique for high speed data transfer between a device and computer memory.

DNA (Digital Network Architecture): Digital Equipment Corporation's eight layer data communications protocol.

DNIC (Data Network Identification Code): A four digit number assigned to public data networks and to specific services within those networks.

Domain: A part of the Internet naming hierarchy consisting of a series of names separated by periods. For example: a host named bar.vax.edu, bar is in domain vax, vax is in domain edu.

Dotted Decimal Notation: The method of representing a 32-bit number with four 8-bit numbers written in base ten and separated by periods. For example 255.128.52.1.

Driver: See **Network Device Driver.**

Drop Cable: The cable that allows connection to and access from the distribution and trunk cables on an Ethernet network. Also called a transceiver cable because it runs from the network node to a transceiver (i.e., a transmitter/receiver) attached to the trunk cable.

DSU (Data Service Unit): A component of customer premises equipment used to interface to a digital circuit (say, DDS or T1) combined with a channel service unit (CSU) converts a customer's data stream to bipolar format for transmission.

DTE (Data Terminal Equipment): Equipment where a communications path terminates. The User's equipment can be called DTE, and may include PCs, etc.

EBCDIC (Extended Binary Coded Decimal Interchange Code): An 8-bit character code used primarily in IBM equipment; the code provides for 256 different bit patterns; compare with **ASCII.**

EGP (Exterior Gateway Protocol): The protocol between external gateways of autonomous systems to advertise the Internet addresses of their respective systems. Every autonomous system must use EGP to advertise network reachability to the core gateway system.

Encryption: The process of systematically altering or encoding data to prevent unauthorized access.

ENDEC: Short for Encoder/Decoder. A functional block within network adapters, that performs two basic functions. First, this function encodes the data from the controller to be transmitted over the network. Second, it decodes the data on the network to a form suitable for the network controller chip. In the case of Ethernet, this function converts NRZ controller data to Manchester data (and back again).

Ethernet: A branching broadcast communications system for carrying digital data packets among locally distributed computing stations. A 10 Mbit/s baseband, Local Area Network that has evolved into the IEEE 802.3 specification. A data-link protocol that specifies how data is placed on and retrieved from a common transmission medium. Ethernet is used as the underlying transport vehicle by several upper-level protocols, including TCP/IP and Xerox Network System (XNS). See **IEEE 802.3.**

FCC (Federal Communications Commission): Board of commissioners appointed by the President under the Communications Act of 1934, with the authority to regulate all interstate telecommunications originating in the United States.

FCS (Frame Check Sequence): Often referred to as CRC. This is a field in a network packet that is used to check for transmission errors in a packet sent across the network. See **CRC.**

FDDI (Fiber Distributed Data Interface): An ANSI Standard for high speed 100 Mbps optical fiber-based LAN with dual counter-rotating rings. Incorporates token passing and supports circuit-switched voice and packetized data. Attachment device may be through SAS or DAS.

FDM (Frequency Division Multiplexing): A method of transmitting multiple independent signals across a single medium by assigning each a unique carrier frequency. See **Multiplexor** and **Demultiplexor.**

FEP (Front End Processor): A dedicated computer linked to one or more host computers or multiuser minicomputers; performs data-communications functions and serves to offload the attached computers of network processing, in IBM SNA networks, an IBM 3704, 3705, 3725 or 3745 communications controller.

Fiber-Optic Cable: A transmission medium that uses glass or plastic fibers to transport data or voice signals. Information is imposed on the glass fiber via pulses (modulation) of light from a laser or light-emitting diode (LED). Its high bandwidth, 100 to 1,000 times the information-carrying capacity of copper wire, and lack of susceptibility to electromagnetic or radio frequency interference, make fiber-optic cable ideal for use in long-haul or noisy environments, and security applications.

File: A collection of logically related records, usually of the same type. A named increment of storage or an unstructured or user structured form of data storage.

File-Allocation Tables (FAT): An area of disk that acts as an index, or directory, that tells the operating system where data has been stored on the disk. A FAT substantially increases a disk system's ability to access stored information by "pointing" the operating system to the exact location of the data it is processing.

File Server: A specialized computer attached to a LAN that provides data-storage service to users on a Local Area Network.

File Transfer: The movement of files or data from one data terminal equipment to another.

Flag: In communications, a bit pattern of six consecutive "1" bits (character representation is 01111110) used in many bit-oriented protocols to mark the beginning of a frame.

Flow Control: The procedure or technique used to regulate the flow of data between devices; prevents the loss of data once a device's buffer has reached its capacity.

Fourth-Generation Language (4GL): A software productivity tool that aids programmers in the desgin and implementation of database management systems (DBMS).

Fragment: Part of a packet that is transmitted on the network. Fragments are usually generated during node collisions on 802.3 networks when one node transmits part of its packet before colliding. Also, one of the pieces that results from an Internet Gateway dividing an IP datagram into smaller pieces for transmission across a network which cannot handle the original datagram size.

Frame: A packet transmitted over a serial line; a physical level transmission. Derived from character-oriented protocols which added start-of-frame characters and end-of-frame characters when sending packets.

Framing: A control procedure used with multiplexed digital channels, such as T1 carriers, whereby bits are inserted so that the receiver can identify the time slots that are allocated to each subchannel. Framing bits may also carry alarm signals indicating specific conditions.

Front End Processor (FEP): A communications computer associated with a host computer. It may perform line control, message handling, code conversion, error control and applications functions such as control and operation of special purpose terminals.

FTP (File Transfer Protocol): A protocol and application primarily on UNIX machines that uploads and downloads files from remote systems across a network.

Full Duplex: The capability of transmitting in two directions simultaneously.

Gateway: A network interconnection device through which data flows from network to network. The gateway may reformat the data as necessary and also may participate in error and flow control protocols. Used to connect LANs employing different protocols or to public data networks.

GBPS (Giga Bits Per Second): A measure of the rate of data transmission referring to billions of bits per second.

GGP (Gateway to Gateway Protocol): The protocol used by core gateways to exchange routing information. GGP uses a shortest path routing computation.

Hardware Address: The low level addresses used by physical networks. Each type of hardware has its own addressing scheme.

Head-end: The point in a LAN where the inbound signals are transferred into outbound signals. The head-end may be passive or contain amplifier or frequency translation equipment. Used in broadband LANs and CATV.

Header: The control information added to the beginning of a message; contains the destination address, source address, and message number.

Heartbeat: In IEEE 802.3 networks this is a short burst of collision signal that is transmitted from the MAU to the DTE after every packet. Also called SQE (Signal Quality Error) test.

Hierarchical Routing: Routing based on hierarchical addressing by dividing the routing procedure into steps based on portions of the address. A gateway will use only the network portion of the address unless it can deliver the packet, then it also uses the host portion. Subnetting is a method of adding additional levels of hierarchical routing.

HDLC (High-level Data Link Control): The link level protocol defined by ISO for bit-oriented, frame-delimited data communications. An Internet standard link level communication protocol. Each frame ends with a frame check sequence for error detection. It is used in X.25 networks for link access protocol. It is increasingly used by PSN interfaces to transfer frames between a host and PSN.

Host: Any network node that a user can access for processing power, information files, and applications. Hosts are general purpose nodes that are not designed to perform network-specific functions.

ICMP (Internet Control Message Protocol): A protocol use primarily in UNIX networking. This protocol handles error and control messages, and low level functions.

Idling Signal: A signal used to communicate that no data is being transmitted but a connection is still established. Without idling signals, a pause in transmissions could be determined to be a lost connection and terminate.

IEEE 802.3: Standard set by the IEEE for CSMA/CD network protocol, that is a Physical Layer definition including specifications for cabling in addition to transmitting data and controlling cable access. See **Ethernet**.

IEEE 802.5: Called token ring and typically used by PCs and large computers to communicate with IBM computers. IEEE 802.5 can transmit up to 4 megabits per second, but with recent improvements can transmit up to 16 megabits per second. It runs on coaxial, twisted-pair and fiber-optic cable.

Unlike IEEE 802.3, the IEEE 802.5 network has a circular rather than a linear topology. And, rather than contending for resources, computers on the network take turns sending data by passing an electrical signal, a token, from one computer to the next. A computer can transmit data on the network only with possession of the token.

IGP (Interior Gateway Protocol): A term applied to any protocol used to communicate routing information and reachability within an autonomous system.

IMP (Interface Message Processor): Former name for Packet Switched Nodes, the Packet switches used in ARPANET. See **PSN**.

Impedance: The resistance a wire offers to a change in current, measured in ohms, as the current runs down the length of the wire. The greater the impedance, the shorter the current that can be sent down the wire. Ethernet, for example, calls for using coaxial cable with a 50Ω impedance factor, while cable television coax offers a 75Ω impedance factor.

Interface: A shared boundary; physical point of demarcation between two devices, where the electrical signals, connectors, timing, and handshaking are defined. The procedures, codes, and protocols that enable two entities to interact for exchange of information.

Internet: A collection of interconnected packet switched networks and gateways which function as one large network by adhering to common protocols.

Internet Address: The 32-bit address, consisting of an Internet address and local address, assigned to a host that wants to participate with the DARPA Internet using TCP/IP.

Internet Layer: A network protocol layer providing host-to-host delivery over an internet. This layer encapsulates messages in IP datagrams and determines delivery pathway information. It is also responsible for handling these datagrams when they are received.

Internet Protocol (IP): The TCP/IP Standard Protocol which defines Internet Datagram as the unit of information passed across the Internet and provides the basis for connectionless, best-effort delivery service.

Interoperability: The ability of many types of hardware and software to communicate and process information in a meaningful fashion.

IP (Internet Protocol): See **Internet Protocol**.

IP Datagram: The basic unit of information passed across the internet containing source and destination address along with data.

IPG (Interpacket Gap): In IEEE 802.3 the minimum time between the end of one packet and beginning of another. This time is 9.6 μs.

IPX/SPX (Internetwork Packet Exchange/Sequenced Packet Exchange): Network protocol. Similar in concept to TCP/IP. This is the proprietary protocol used by Novell in its NetWare products. Based on XNS protocol.

ISDN (Integrated Services Digital Network): An integrated digital network in which the same digital switches and digital paths are used to establish connections for voice and data traffic on the same digital links.

Inter-Networking: The connection of two or more networks so that work nodes on both can communicate with each other.

ISO: International Standards Organization

Jabber: A condition on an Ethernet LAN network when a node transmits for longer than the specified time.

Jam: In IEEE 802.3, networks when a collision occurs the colliding nodes ensure that the collision is seen by the entire network by continuing to transmit for a minimum time during a collision. Called jamming.

Jitter: The slight movement of a transmission signal in time or phase that can introduce errors and loss of synchronization in high-speed synchronous communications.

Jumper: A patch cable or wire used to establish a circuit for testing or diagnostics.

Kilobyte (kB): Term denoting a thousand bytes, a group of adjacent and related binary digits. A kilobyte is really 1024 bytes (this is arrived by multiplying two by itself ten times, i.e., 2^{10}).

LAN (Local Area Network): A communications system linking computers together to form a network whose dimensions typically are less than five kilometers. Transmissions within a Local Area Network generally are digital, carrying data among stations at rates usually above one megabit per second. An assembly of computing resources such as microcomputers (i.e., PCs), printers, minicomputers and mainframes linked by a common transmission medium, including coaxial cable or twisted-pair wiring.

LAN Manager: The multiuser network operating system co-developed by Microsoft and 3Com. LAN Manager offers a wide range of network management and control capabilities unavailable with existing PC-based network operating systems. It runs on Microsoft's OS/2 operating system.

LAP (Link Access Procedure): The data-link-level protocol specified in the CCITT X.25 interface standard; original LAP has been supplemented with LAPB (LAP-Balanced) and LAPD.

LAPB (Link Access Procedure Balanced): LAPB is the most common data link control protocol used to interface X.25 DTEs to X.25 DCEs. X.25 also specifies LAP (Link Access Protocol, not balanced). Both protocols are full duplex, bit synchronous protocols. The unit of transmission is a frame. Frames may contain one or more X.25 packets.

LAPD (Link Access Procedure-D): Link-level protocol devised for ISDN connections, differing from LAPB (LAP-Balanced) in its framing sequence. Likely to be used as basis for LAPM, the proposed CCITT modem error-control standard.

Leased Line: A dedicated circuit, typically supplied by the telephone company, that permanently interconnects two or more user locations; generally voice-grade in capacity and in range of frequencies supported; typically analog, though sometimes it refers to DDS sub-rate digital channels (2.4 Kbps to 9.6 Kbps); used for voice (2000 Series leased line) or data (3002 type); could be point-to-point or multipoint; may be enhanced with line conditioning; also, private line.

Latency: The time interval between when a network station seeks access to a transmission channel and when access is granted or received; equivalent to waiting time.

Link Integrity Test: A function specified by the 10BASE-T standard which provides indication of whether the cable linking the DTE to the HUB is properly connected.

Link Layer: Layer two of the OSI reference model; also known as the Data-Link Layer.

Little-Endian: A binary data storage/transmission format in which the least significant byte (bit) comes first. See **Big-Endian.**

LLC (Logical Link Control): A protocol developed by the IEEE 802 committee for data-link-level transmission control; the upper sublayer of the IEEE Layer 2 OSI protocol that complements the MAC protocol; IEEE standard 802.2; includes end-system addressing and error checking.

LU 6.2: In Systems Network Architecture, a set of protocols that provides peer-to-peer communications between applications.

M Bit: The More Data mark in an X.25 packet that allows the DTE or DCE to indicate a sequence of more than one packet.

Mail Bridge: A mail gateway which screens mail passing from one network to another for security and administrative purposes.

Mail Exploder: A program which accepts a piece of mail and a list of addresses, then sends a copy of the message to each listed address.

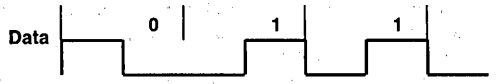
Mail Gateway: A machine which connects 2 or more mail systems and transfers mail among them, usually used between dissimilar systems on different networks; it will reformat messages according to destination's mailing system rules before forwarding the message.

Mail Server: The software and machine which provides message transfer services on a network.

MAN (Metropolitan Area Network): A high speed network provides facilities for data communication between sites within a neighborhood or city for distances up to 40 km and typically at data rates up to 2 Mb/s.

MAC (Media-Access Control): A sub-layer of the Data Link Layer, Level Two, of the ISO OSI model responsible for media control.

Manchester Encoding: Digital encoding technique (specified for the IEEE 802.3 Ethernet baseband network standard) in which each bit period is divided into two complementary halves; a negative-to-positive (voltage) transition in the middle of the bit period designates a binary "1", while a positive-to-negative transition represents a "0". The encoding technique also allows the receiving device to recover the transmitted clock from the incoming data stream (self-clocking).



MAP (Manufacturing Automation Protocol): A General Motors® originated suite of networking protocols, which tracks the seven layers of the OSI model.

Mapping: In networking operations, the logical association of one set of values, such as addresses on one network, with quantities or values of another set, such as devices on another network (e.g., name-address mapping, internetwork route mapping, protocol-to-protocol mapping).

MAU (Medium Attachment Unit): The physical and electrical component that provides the means of attaching computing devices to the local network medium. In 10BASE-T networks, it is a repeater or a network interface adapter board equipped with a medium-dependent interface.

MDI (Medium-Dependent Interface): The mechanical and electrical interface between the twisted-pair link and the MAU. In 10BASE-T networks, the MDI is an 8-pin RJ45 modular telephone connection.

Medium: Any material substance used for the propagation or transmission of signals, usually in the form of electrons or modulated radio, light, or acoustic waves; such as optical fiber, cable, wire, dielectric slab, water, or air.

MHS (Message Handling System): The standard defines by the CCITT as X.400 and by the ISO as Message Oriented Text Interchange Standard (MOTIS).

Mid-Level Net: One of many networks funded by the NSF which operated autonomously but was connected to the NSFnet Backbone.

MIF (Minimum Internetworking Functionality): A general principle within the ISO that calls for minimum Local Area Network station complexity when interconnecting with resources outside the Local Area Network.

MILNET (MILitary NETwork): A network which was separated from ARPANET to provide reliable service to the military while ARPANET was used for continued research.

Mini-MAP (Mini-Manufacturing Automation Protocol): A version of MAP consisting of only physical, link, and application layers intended for lower-cost process-control networks. With Mini-MAP, a device with a token can request a response from an addressed device; unlike a standard MAP protocol, the addressed Mini-MAP device need not wait for the token to respond.

MPR (Multi-Port Repeater): See Repeater. A repeater with numerous network connection ports at one point on the Ethernet. In coaxial networks these repeaters typically have 8 ports; in twisted pair Ethernet up to several hundred ports are possible.

MS OS/2® LAN Manager: The multiuser network operating system co-developed by Microsoft and 3Com. LAN Manager offers a wide range of network management and control capabilities unavailable with existing PC-based network operating systems.

MTBF (Mean Time Between Failures): A stated or published period of time for which a user may expect a device to operate before a failure occurs.

MTTR (Mean Time To Repair): The average time required to perform corrective maintenance on a failed device.

Multimode: An optical fiber designed to carry multiple signals, distinguished by frequency or phase, at the same time. Compare with **Single mode.**

Multiple Access: The ability of any node on an Ethernet LAN to send a message immediately upon sensing that the channel is free.

Multiple Routing: The process of sending a message to more than one recipient, usually when all destinations are specified in the header of the message.

Multiplexor: A hardware device which combines multiple signals from a transmission line based on time or carrier frequency. It is used on broadband systems in combination with a demultiplexor to allow multiple, simultaneous signal transmissions over a single medium. It allows multiple hardware devices to use a single communication link simultaneously.

Multipoint Line: A single communications line or circuit interconnecting several stations supporting terminals in several different locations. This type of line usually requires a polling mechanism, with each terminal having a unique address. Also called a multidrop line.

Multipoint Link: A single line that is shared by more than two nodes.

Multitasking: The concurrent execution of two or more tasks or applications by a computer; may also be the concurrent execution of a single program that is used by many tasks.

Multistation Access Unit (MAU): A multiport connector, or concentrator, for Token Ring networks which allows devices to be connected to the ring; also provides a built-in relay that prevents a break in the network when devices are attached or removed.

MUX: See **Multiplexor.**

NAK (Negative AcKnowledgement): A message sent from a receiver to a sender which indicates that the transmitted data contained errors from the transmission. Upon receiving a NAK, the sender will usually retransmit the data.

Name Resolution: The process of converting a host's name into a corresponding network address.

Named Pipe: The facility within the Microsoft OS/2 LAN Manager that allows processes on separate machines to communicate with each other across a network. Provide a simple way for application developers to write sophisticated distributed network applications.

NCC (Network Control Center): Any centralized network diagnostic and management station or site, such as that of a packet-switching network.

NDIS (Network Driver Interface Specification): Standard specification for network drivers for Microsoft's OS/2 LAN Manager.

NetBIOS (Network Basic Input/Output System): A programming interface to a data exchange protocol. Associated with several communications protocols and used to refer to the combination of the interface and the protocols.

It allows users and software developers to write PC programs that can communicate over a PC network in a peer-to-peer fashion.

Recent proposals define a way for NetBIOS—and, programs supporting it—to run over TCP/IP and OSI protocols.

NetView®: This is IBM's proprietary network management system that monitors, manages, and controls SNA networks. NetView allows other vendors' network management programs to communicate with it.

Network: An interconnected group of nodes; a series of points, nodes, or stations connected by communications channels; assembly of equipment with connections made between data stations.

Network Address: Refers to characters that identify the location of a node on a network. See **Address.**

Network Architecture: A set of design principles, including the organization of functions and the description of data formats and procedures, used as the basis for the design and implementation of a network (ISO).

Network Interface Controller: Electronic circuitry that connects a workstation to a network, usually a card that fits into one of the expansion slots inside a personal computer. It works with the network software and computer operating system to transmit and receive messages on the network; also, network interface card.

Network Layer: See **OSI.**

Network Management: Administrative services performed in managing a network; e.g., network topology and software configuration, software downloading, network statistics monitoring, maintenance of network operations, and troubleshooting and diagnosis.

Network Topology: The physical and logical relationship of nodes in building a network configuration; the schematic arrangement of the links and nodes of a network; networks are typically a star, ring, tree, or bus topology, or hybrid combination.

NFS (Network File System): An extension of TCP/IP, developed by SUN Microsystems, that allows files on remote nodes of a network to appear locally connected.

NIC (Network Interface Controller): National's Industry-Standard 8/16 bit Ethernet Controller. Part number DP8390.

NLM (Netware Loadable Module): Novell's NetWare 386 supports the ability to load and run programs on the server to enhance features of the server. Programs are called NLMs.

Node: An endpoint of any branch of a network, or a junction common to two or more branches of a network. In a data network, a point where one or more functional units interconnect data transmission lines. Distributed system nodes include information processors, network processors, terminal controllers and terminals.

ODI (Open Data Link Interface): Standard specification created by Novell to facilitate writing of drivers under NetWare 386. Similar intent as Microsoft's NDIS.

Off-Line: When a user, terminal, or other device is not connected to a computer or actively transmitting via a network.

On-Line: Condition in which a user, terminal, or other device is actively connected with the facilities of a communications network or computer. Opposite of off-line.

Optical Fiber: Any filament or fiber, made of dielectric materials, that is used to transmit laser- or LED-generated light signals. Optical fiber usually consists of a core which carries the signal, and cladding, a substance with a slightly higher refractive index than the core, which surrounds the core and serves to reflect the light signal.

OSI (Open Systems Interconnection): A logical structure model for network operations standardized within the ISO; a seven-layer network architecture used for the definition of network protocol standards to enable any OSI-compliant computer or device to communicate with any other OSI-compliant computer or device for a meaningful exchange of information. The layers are: Physical, Data Link, Network, Transport, Session, Presentation, Application:

1. **Physical Layer**—Network wire and cable systems, defines mechanical and electrical means by which devices are physically connected to a transmission medium.
2. **Data Link Layer**—Synchronizing the flow of data and handling error control across the physical data link.
3. **Network Layer**—Provides the means to establish, maintain, and terminate connections between systems; concerned with switching and routing of information.
4. **Transport Layer**—Checks the integrity of data transported over the network.
5. **Session Layer**—Standardizes the task of setting up a session and terminating it; coordination of the interaction between stations on the network.
6. **Presentation Layer**—Defines the character set and data code, and the way data is displayed on a screen or printer format, character set, and language.
7. **Application Layer**—Links the network operating system and the application programs to perform the type of information transfer required.

OSINET: A test network, sponsored by the National Bureau of Standards (NBS), designed to provide a forum for doing interoperability testing for vendors of products based on the OSI model.

Overhead: All information, such as control, routing, and error-checking characters, in addition to user-transmitted data, including information that carries network status or operational instructions, network routing information, and retransmissions of user data messages that are received in error.

Out of Window Collision: A collision on an IEEE 802.3 network that occurs outside of the specified time (for 10 Mbit/sec standards the legal collision occurs within the first 51.2 μs of the packet).

Packet: A series of bits forming a complete unit of information that is sent across a network. The packet has a defined format which includes who the packet is for and who sent it (destination address source address). See **Circuit Switching**. For IEEE 802.3 the physical layer packet consists of the following fields.

Preamble (62 bits)
Start of Frame Delimiter (SFD) (2 bits)
Destination Address (6 bytes)
Source Address (6 bytes)
Data Field (64–1500 bytes)
CRC (4 bytes)

Packet Buffer: A memory space set aside for storing a packet that is either waiting to be transmitted, or has been received. May be located in either the network interface controller or the computer attached to the controller.

Packet Switching: A mode of data transmission in which messages are broken into smaller increments called packets, each routed independently to the destination. 2. The process of routing and transferring data by means of addressed packets, whereby a channel is then available for the transfer of other packets.

PAD (Packet Assembler/Disassembler): For X.25, a PAD allows non-X.25 users to access an X.25 network. CCITT recommendations X.3, X.28 and X.29 define PAD parameters, terminal-to-PAD interface and PAD-to-X.25 host interface.

Pass-Through: Describing the ability to gain access to one network element through another.

PBX (Private Branch Exchange): A manual, user-owned telephone exchange.

Peer-to-Peer Communications: The ability of intelligent computing devices to communicate without relying on a host computer.

Physical Layer: See **OSI.**

PLS (Physical Layer Signaling): Is the portion of the interface that enables MAC function communications with the AUI interface in IEEE 802.3 specifications.

PLP (Packet Level Procedures): Defines protocols for the transfer of packets between X.25 DTE and X.25 DCE. X.25 PLP is a full duplex protocol that supports data sequencing, flow control, accountability, and error detection and recovery.

PMA (Physical Medium Attachment): In IEEE 802.3, the portion of the MAU that contains electronic circuitry.

Pipe: A communications process provided by the operating system that acts as an interface between a computer's devices—keyboard, disk drives, memory, etc.—and an application program. A pipe simplifies the development of applications programs by "buffering" the application program from the intricacies of the hardware and/or the software that controls the hardware: The application developer writes "code" to a single pipe, not several individual devices.

Point-to-Point Connection: 1. In data communications, a connection established between only two data stations for data transmission. The connection may include switching facilities. Describing a circuit that interconnects two points directly, where there are generally no intermediate processing nodes, computers, or branched circuits, although there could be switching facilities; a type of connection, such as a phone line circuit, that links two, and only two, logical entities, see **multipoint line, broadcast.**

Port: A point of access into a computer, a network , or other electronic device; the physical or electrical interface through which one gains access; the interface between a process and a communications or transmission facility.

Presentation Layer: See **OSI.**

Print Server: An intelligent device used to transfer information to a series of printers.

Printer Spooler: The software that allows a user to send a file to a shared printer over a network even when the printer is busy; the file is saved in temporary storage, then printed when the printer is free.

Promiscuous ARP: See **Proxy ARP.**

Protocol: Formal set of rules governing the format, timing sequencing, and error control of exchanged messages on a data network; may be oriented toward data transfer over an interface, between two logical units directly connected, or on an end-to-end basis between two users over a large and complex network.

Protocol Port: A method used by transport protocols to tell the difference between many possible destinations within a single host. Operating systems usually allow an application program to specify what port it wants to be.

Proxy ARP: When one machine, usually a gateway, answers ARP request intended for another by supplying its own physical address, thus accepting responsibility for routing packets to the correct machine. Proxy ARP is used to allow a site to use a single Internet address with more than one physical network. See **ARP Hack.**

PSDN (Packet-Switched Data Network): A vendor-managed network that uses the X.25 protocol to transport data between customers' computers connected to the PSDN. Tariffs for PSDNs are based on the volume of data sent rather than on the distance or connect time between communicating computers.

PSN (Packet Switched Node): The name for an ARPANET packet switch. Each PSN is connected to at least 2 others as well as up to 16 host computers.

Public Network: A network operated by common carriers or telecommunications administrations for the provision of circuit-switched, packet-switched, and leased-line circuits to the public.

Queue: Any group of items, such as computer jobs or messages, waiting for service. A line of prioritized tasks waiting to be executed. High priority tasks will be executed before low priority tasks.

Queuing: Sequencing of batch data sessions.

Radio Frequency (RF): A generic term referring to the technology used in cable television and broadband Local Area Networks. Uses electromagnetic waveforms, usually in the megahertz (MHz) range, for transmission.

RARP (Reverse Address Resolution Protocol): The inverse of ARP. A Transmission Control Protocol/Internet Protocol (TCP/IP) process that maps Ethernet addresses back to IP addresses; required by TCP/IP for use with Ethernet. Also referred to in other protocols as a reverse address resolution protocol.

Real Time: Operating mode that allows immediate interaction with data as it is created, as in a process-control system or computer-aided design system.

Redundancy: In data transmission, the portion of a message's gross information content that can be eliminated without losing essential information; also, duplicate facilities.

Repeater: A device used to extend the length and topology of a physical channel, particularly a LAN cable, up to the maximum allowable end-to-end channel propagation limit.

Response Time: For interactive sessions, the elapsed time between the end of an inquiry and the beginning of a response.

Retransmissive Star: In optical-fiber transmission, a passive component that permits the light signal on an input fiber to be retransmitted on multiple output fibers; formed by heating together a bundle of fibers to near the melting point; used mainly in fiber-based Local Area Network; also, star coupler.

RFC (Request For Comment): A series of notes which contain information about the development of the DARPA Internet, including proposed and accepted protocols for the Internet.

RF MODEM: A MOdulator-DEModulator that converts digital signals to analog signals (and vice versa), then modulates/demodulates them to/from their assigned frequencies. Used in broadband LANs.

RFS (Remote File Service): AT&T's network file protocol for UNIX networks. Similar to NFS, except that NFS was designed for connectivity of file structures across different platforms, and RFS provides complete support for the UNIX file system semantics, and therefore can only connect UNIX file systems across the network.

Ring: Two or more stations connected by a physical medium wherein information is passed sequentially between active stations, each station in turn examining or copying and repeating the information, finally returning it to the originating station.

Ring Network: A distributed system in which the information processors are connected via a circular arrangement of data communications facilities.

Ring Topology: A LAN configuration in which each computer, or other node, is connected to the next, with the "last" connected to the "first" to form a complete loop, or ring. This ring may be a true physical ring or an electrical (also called logical) ring. In a physical ring, nodes are connected serially, one after the other, in a closed loop. In a logical ring, nodes are connected to a central device that routes the data-carrying signal in a circular fashion.

RIP (Routing Information Protocol): An interior gateway protocol (IGP) used by Berkeley BSD 4.3 UNIX systems to exchange routing information between a small number of hosts.

RJE (Remote Job Entry): A service offered by many networks which allows a user to submit a batch job to a host from a remote site.

RLOGIN (Remote Login): A service offered by Berkeley 4.3 BSD UNIX systems allowing a user on one machine to connect with other internetworked UNIX machines as if their terminal were directly attached.

Routed (Route DAEMON): A 4.3 BSD UNIX program which updates routing information on local area networks using RIP protocols.

Route: The path taken by data traffic within a network or through an internet.

Router: A hardware-software device that connects geographically dispersed local area networks (often, of different types, such as Ethernet and token ring) together.

Routing: The process of selecting the correct circuit path for a message.

RPC (Remote Procedure Call): This is a UNIX session layer protocol for UNIX.

RTT (Round Trip Time): The time it takes for a single packet or datagram to leave one machine, reach its destination and then return to the source machine.

RUNT Packet: In 802.3 networks, a special case of a fragment packet when the length of the packet is less than 512 bit time.

SAA (System Application Architecture): An IBM developed set of standards that provides identical user interfaces for applications running on PCs, minicomputers, and mainframes.

SAS (Single Attach Station): A device attached to one ring of the FDDI network.

SDLC (Synchronous Data Link Control): A predecessor of HDLC defined by IBM Corporation and used in their SNA network products.

Segment: The unit of transfer between TCP's on different machines. Each segment contains a stream of bytes being sent between the machines as well as additional fields for identifications, error checking etc.

Serial Transmission: The sequential transmission of the bits constituting an entity of data over a data circuit.

Server: A specialized computer that provides a particular service, such as file or print service, to a network; increasingly, it comprises both the hardware and software which manage a network operation.

Session Layer: See **OSI**.

SFD (Start of Frame Delimiter): A bit pattern that enables the network board/controller to obtain byte synchronization to the incoming serial bit data from the network.

SFT (System Fault Tolerant): A version of Novell's NetWare that provide reliability features such as disk and file mirroring.

Shielding: Protective enclosure surrounding a transmission medium, such as coaxial cable, designed to minimize electromagnetic leakage and interference.

Signal Quality Error: See **Heartbeat**.

Single Mode: Describes an optical waveguide designed to propagate light of only a single wavelength and perhaps a single phase; essentially, an optical fiber that allows the transmission of only one light beam, or data-carrying lightwave channel, and is optimized for a particular lightwave frequency; compare with **multimode**.

SLIP (Serial Line Internet Protocol): Networking protocol to connect via a point-to-point serial link to network services.

SMTP (Simple Mail Transfer Protocol): The DARPA Internet standard protocol for transferring electronic mail messages from one machine to another. SMTP specifies how two mail systems interact and the format of control messages they exchange to transfer mail.

SNA (Systems Network Architecture): The IBM network architecture for communication among IBM devices, and between IBM and other IBM machines.

SNI (Serial Network Interface): National's Manchester encoder/decoder. Part number DP8391, or CMOS version DP83910.

SNIC (Serial Network Interface Controller): National's newer 8/16 Controller that also incorporates the SNI encoder/decoder function in addition to the controller. It is 100% software compatible with the NIC.

SNMP (Simple Network Management Protocol): Used to monitor IP devices and the networks they are attached to. The device must maintain a set of variables that specifies that all operations on the device are an effect of retrieving and storing to the data variables. It contains three parts: Structure of Management Information (SMI), Management Information Base (MIB), and the protocol. SMI and MIB define and store managed entities; SNMP conveys information to and from these entities. See **CMOT** or **CMIP**.

Sockets: A Berkeley BSD 4.3 network interface Specification to the transport layer. This interface provides 3 basic services. 1) Stream services which guarantees delivery of a sequence of data. 2) Datagram services which provides delivery of a data packet, but does not guarantee delivery. 3) Raw services which provide low level network functions.

Source Node: A network node that sends a message.

Source Route: A route is determined by the transmission source. The source establishes a sequence of machines that a datagram must visit to its destination.

SPOOL (Simultaneous Peripheral Operation On Line): A program or piece of hardware that controls data going to an output device.

STARLAN: A local area network design and specification, within the IEEE 802.3 standards, characterized by 1 Mbps baseband data transmission over two-pair twisted-pair wiring.

Star Network: 1. A computer network in which each peripheral network node is connected only to the computer or computers at a single central facility. 2. A configuration in which remote terminals and/or processors are connected radially to a central processing location.

STAR Topology: A LAN configuration in which nodes are connected individually to a common device, such as a concentrator, which acts as a focal point for network cabling.

Step-Index: A type of optical fiber that exhibits a uniform refractive index at its core and a sharp decrease in the refractive index at its core-cladding interface.

Store-and-Forward: A communications technique in which mesages are received at intermediate routing points and stored temporarily, then re-transmitted to an additional routing point or final destination.

Structured Query Language (SQL): A formal data sub-language for specifying common database operations, such as retrieving, adding, changing or deleting.

Subnet: A local area network which resides within another network.

Subnet Address: An extension of the DARPA Internet addressing scheme that allows a site to use a single internet address for many physical networks. The subnet address is not looked at by the Internet portion of the routing, it is only used by local gateways and hosts to deliver the datagram to the correct physical address.

SYN (Synchronizing Segment): The first segment sent by the TCP protocol, used to synchronize the two ends of a connection in preparation for opening another connection.

Synchronous: Communications link in which the data characters and bits are transmitted at a fixed rate with the transmitter and receiver are synchronized, eliminates the need for individual start bits and stop bits surrounding each byte, thus providing greater efficiency. Contrast with asynchronous transmission.

Synchronous Transmission: Data transmission in which the occurrence of each signal representing a bit is related to a fixed time frame. Compare with asynchronous data transmission.

TAP: A device that connects a device cable to a transceiver (on baseband networks), or transfers a signal from the trunk line to a drop line (on broadband networks).

T Carrier: A time-division-multiplexed, typically telephone-company-supplied, digital transmission facility, usually operating at an aggregate data rate of 1.544 Mbps and above.

TCP/IP: The Transmission Control Protocol/Internet Protocol was formerly used only in the military, technical and university communities, but it is enjoying a surge in popularity as the commercial sector discovers its value for communicating between computers of different vendors, especially Unix systems.

TCP/IP is governed by a body of vendors and users, keeping it stable over more than a 15-year life span. TCP/IP predates OSI, and it includes several functions that properly belong in the upper level of the OSI model, such as applications that provide electronic mail, terminal emulation and file transfer.

TELCO: Telephone central office, in most usages; but also, a generic abbreviation for "telephone company".

Telecommunications: A term encompassing both voice and data communcations in the form of coded signals over media.

TELNET: An application and protocol and program that primarily is to interface to UNIX computers. This allows terminal emulation across the network, allowing a user on one computer to log in to another computer as if the user's computer were a terminal.

Terminal: 1. A device, such as a teletypewriter or a keyboard/CRT device, which embodies a set of human/system interface functions. 2. A point in a system or communications network at which data can either enter or leave; a device, usually equipped with a keyboard, often with a display, capable of sending and receiving data over a communcations link; generically the same as data terminal equipment.

Terminal Emulation: A program which runs at a workstation or terminal that makes it appear to be a specific type of data terminal to both the user and the software.

Terminal Server: A special purpose device on an Ethernet LAN that enables up to 32 terminals to be connected to the Ethernet cable via a single physical line. A terminal server frees network nodes of the burden of establishing connections between local terminals and remote nodes. Terminals connected to the terminal server have access to all nodes on the network.

Terminated Line: A circuit with a resistance at the far end equal to the characteristic impedance of the line, so no reflections or standing waves are present when a signal is entered at the near end.

Text: In communications, transmitted characters forming the part of a message that carries information to be conveyed; in some protocols, the character sequence betwen start-of-text (STX) and end-of-text (ETX) control characters; information for human, as opposed to computer, comprehension, intended for presentation in a two-dimensional form.

TFTP (Trivial File Transfer Protocol): The DARPA Internet standard protocol for file transfer with minimal overhead and capability. It depends only on the unreliable, connectionless datagram delivery service (UDP), so it can be used on diskless workstations that keep software in ROM in order to bootstrap themselves.

T1: AT&T term for a digital carrier facility used to transmit a DS-1 formatted digital signal at 1.544 Mbps.

Timeout: Expiration of predefined time period, at which time some specified action occurs; in communications, timeouts are employed to avoid unnecessary delays and improve traffic flow; used, for example, to specify maximum response times to polling and addressing before a procedure is automatically reinitiated.

TPI (Twisted Pair Interface): National's twisted pair Ethernet Transceiver for 10BASE-T. Part number DP83922.

Token: The password or character sequence used by network nodes to gain access to a token ring network. This character sequence (i.e., the token) passes from one node to another around the network; hence, the term "token passing" is used to describe the process.

Token Ring: A data-signaling network architecture where a data packet and a token are passed from one station to another along an electrical ring. When a station transmits, it takes possession of the token, transmits its data, then frees the token after the data has made a complete circuit of the electrical ring.

TOP (Technical and Office Protocols): A Boeing version of the MAP protocol suite aimed at office and engineering applications.

Topology: Description of the physical connections of a specific network's nodes—such as bus, branching bus (tree or star), or ring.

Transaction: In communications, a message destined for an application program; a computer-processed task that accomplishes a particular action or result; in interactive communcations, an exchange between two devices, one of which is usually a computer; in batch or remote job entry, a job or job step.

Transceiver: A combined transmitter and receiver. An essential element of all LANs, its functions is required at each node of the network. For Ethernet it connects directly to the coaxial cable as a stand alone transceiver box. For Thin Ethernet the transceiver resides in the data terminal equipment.

Transmission: The dispatching of a signal, message, or other forms of intelligence by wire, radio, telegraphy, telephony, facsimile, or other means; a series of characters, messages, or blocks, including control information and user data; the signaling of data over communications channels.

Transport Layer: Layer four in the OSI reference model; provides a logical connection between processes on two machines. See **OSI**.

Tree: A LAN topology that recognizes only one route between two nodes on the network. The "map" resembles a tree or the letter T.

Trunk: A dedicated aggregated telephone circuit connecting two switching centers, central offices, or data-concentration devices.

Twisted-Pair Transmission System: In 10BASE-T terminology, refers to the twisted-pair wire link and its two attached MAUs.

Twisted-Pair Wire: A cable comprised of two 18 to 24 AWG (American Wire Gauge) solid copper strands twisted around each other. The twisting provides a measure of protection from electromagnetic and radio-frequency interference (EMI/RFI). Two types are available: shielded and unshielded. The former is wrapped inside a metallic sheath that provides protection from EMI/RFI. The latter, also known as telephone wire, is covered with plastic or PVC, which provides no protection from EMI/RFI.

Type 3 Cable: An unshielded twisted-pair wire that meets IBM specifications for use in token ring networks.

UDP (User Datagram Protocol): The TCP/IP transaction protocol used for applications such as remote network management and name service access; this lets users assign a name, such as "VAX™ 2", to a physical or numbered address.

UTP (Unshielded Twisted-Pair Cable): Also known as telephone wire. See **Twisted-Pair Wire**.

User Transparency: The quality in a network that enables users to access and transfer information without having to know how the network operates.

VAN (Value Added Network): A network whose services go beyond simple switching.

VC (Virtual Circuit): For X.25 a VC is a PLP logical connection between an X.25 DCE and an X.25 DTE. X.25 supports both switched VCs and permanent VCs. Switched VCs are analogous to dial up lines. They allow a particular X.25 DTE to establish connection with different X.25 DTEs on a per call basis. In contrast, permanent VCs are analogous to leased lines because they always connect two particular X.25 DTEs.

Virtual Disk: A portion of physical disk drive appearing to a dedicated host as a local disk resource.

Virtual Storage: Storage space that may be viewed as addressable main storage, but is actually auxiliary storage (usually peripheral mass storage) mapped into real addresses; amount of virtual storage is limited by the addressing scheme of the computer.

VMS™ (Virtual Memory System): An Operating System developed by Digital Equipment Corporation for the VAX computer series.

Well-Known Port: Any set of protocol port numbers preassigned for specific uses by the transport layer protocols (i.e., TCP and UDP). Clients can locate servers at well-known port assignments. File transfer servers, echo servers and time servers are some examples of servers using well-known port assignments.

Wide Area Network (WAN): A network covering a large geographic area (50 miles or more); may include packet-switched, public data, and Value-Added Networks.

Wide Band: A system in which multiple channels access a medium (usually coaxial cable) that has a large bandwidth (10 Mbps is typical) using radio frequency modems. Each channel is modulated to a different frequency slot on the cable and is demodulated to its original frequency at the receiving end.

Wiring Closet: Central location for termination and routing on-premises wiring systems.

Workstation: Input/Output equipment that an operator works.

XDR (External Data Representation): A presentation layer protocol used by SUN microsystems. It provides a common way of representing data on a network consisting of different machines.

XNS (Xerox Network System): Used as the basis for many network operating systems, including early version of 3Com's 3+. It performs functions similar to those of TCP/IP and runs on top of IEEE 802.3.

X.NN: The X.nn series of the CCITT standards relate to the connection of digital equipment to a public data network which employs digital signaling.

XON/XOFF (Transmitter On/Transmitter Off): A method of flow control used when a computer is attached to a slower device which cannot process information as fast as the computer sends it. A common device using XON/XOFF is a printer. XON is sent as a CONTROL-Q, XOFF is sent as a CONTROL-S.

X.25: Defined by CCITT, and used most commonly in Europe.

The X.25 protocol is best suited for small to medium amounts of data traffic among multiple locations. It operates over telephone lines at up to 1.5 megabits per second (the maximum speed of a T-1 connection). X.25 breaks a data message into smaller pieces known as "packets" and transmits the packets individually to their destination, where they are reassembled. Because each packet is routed individually, packets may travel different paths to their destination, thereby speeding transmission.

X.400: This standard, approved by ISO, defines a means for exchanging electronic mail between computers. Supporting this standard allows electronic mail packages from different vendors to exchange messages.

X.500: Still under development, X.500 is a standard for directory management. It will allow users to find files and data on networks of different types of computers.

10BASE5: IEEE 802.3 Physical Layer Standard for thick cable Ethernet, utilizing thick double shielded coaxial cable. 10BASE5 stands for; 10 = 10 Mbits/sec. data rate, BASE = Baseband, 5 = 500 meters segment length.

10BASE2: IEEE 802.3 Physical Layer Standard for thin wire Ethernet (sometimes called cheapernet). This standard uses RG58 standard coaxial cable. 10BASE2 stands for; 10 = 10 Mbit/sec. data rate, BASE = Baseband, 2 = 200 meter segment length (actually is 185m).

1BASE5: IEEE 802.3 Physical Layer Standard for StarLAN twisted pair network. 1BASE5 stands for; 1 = 1 Mbit/sec. data rate, BASE = Baseband, 5 = 500 meter segment length.

10BASE-T: IEEE 802.3 Physical Layer Standard for the new twisted pair Ethernet in a star topology. 10BASE-T stands for; 10 = 10 Mbit/sec. data rate, BASE = Baseband, T = twisted pair wire over 100 meters nominal segment length.

802.x: The Institute of Electrical and Electronic Engineers (IEEE) committees that developed a set of standards defining some networks. The IEEE committees generally work on standards below 50 Mb/s including:

IEEE 802.3 Ethernet
IEEE 802.4 Token Bus
IEEE 802.5 Token Ring
IEEE 802.6 Metropolitan Area Networks
IEEE 802.9 Integrated Data and Voice

3270 and 5250: These two IBM protocols have been around since the early 1970s. High Speed Serial Interface used with IBM mainframes and various peripherals. Data rates range from 1.2 Mbits/s to 52 Mbits/s.

Section 7

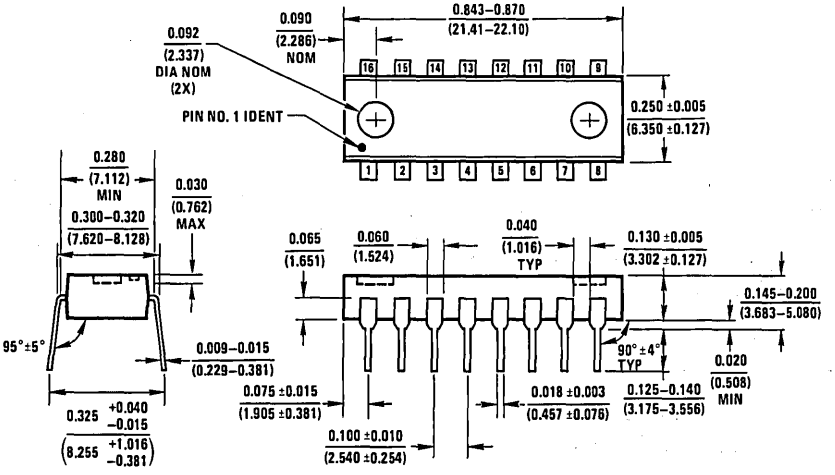
Appendix/ Physical Dimensions

Section 7 Contents

National Semiconductor

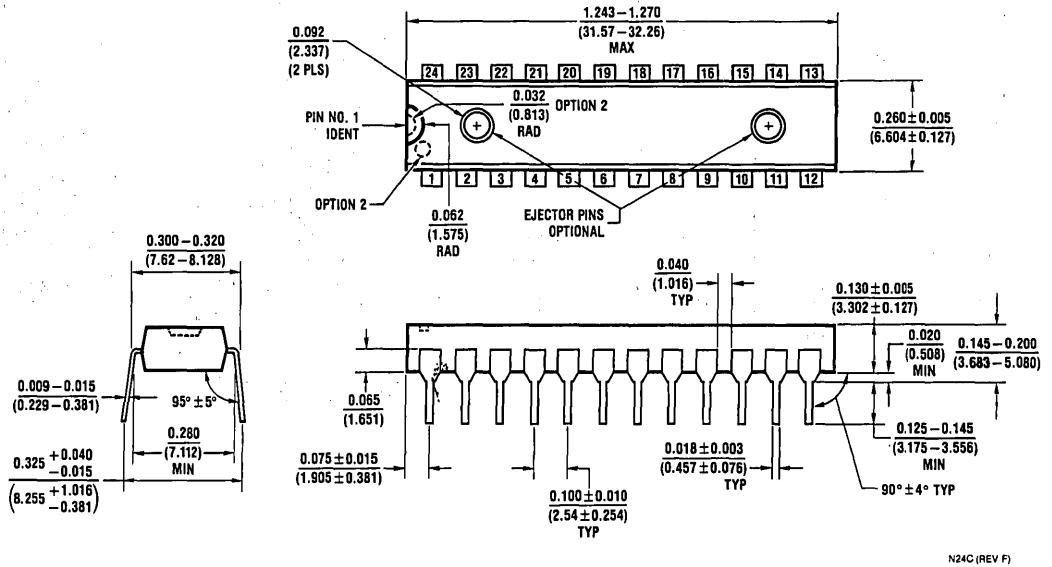
All dimensions are in inches (millimeters)

16 Lead Molded Dual-In-Line Package (N)
NS Package Number N16A



N16A (REV E)

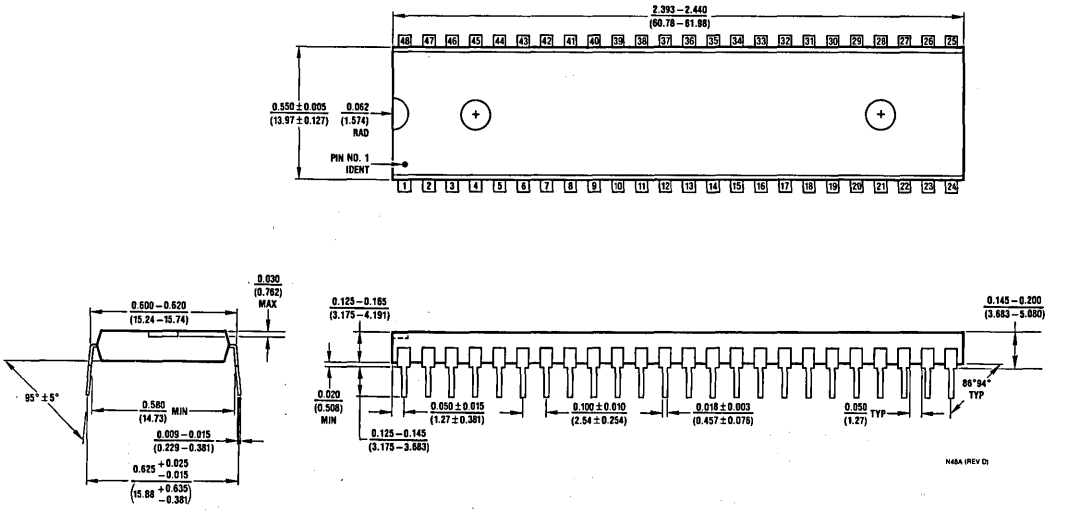
24 Lead Skinny Dual-In-Line Package (0.300" Centers Molded) (N)
NS Package Number N24C



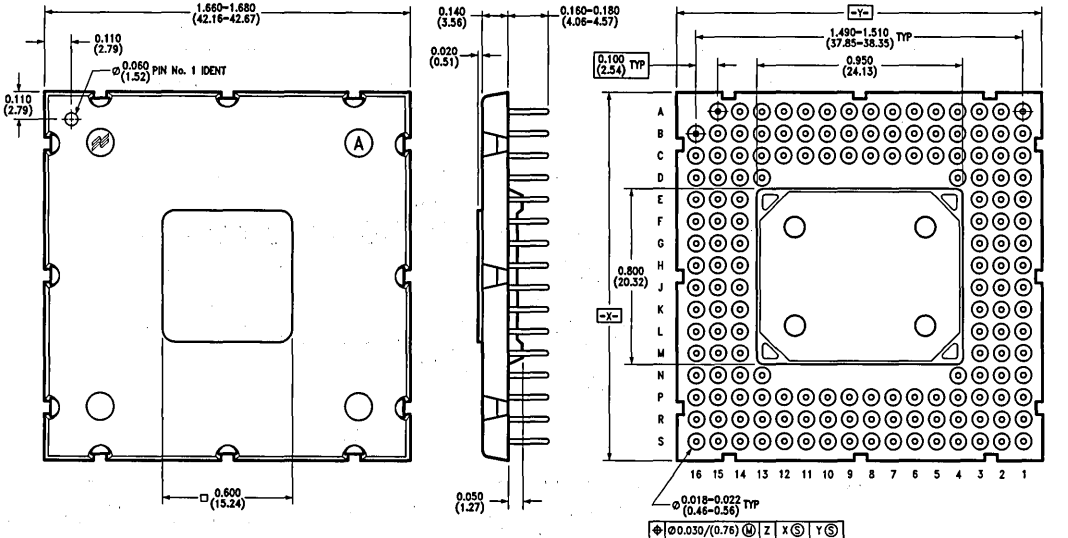
N24C (REV F)

7

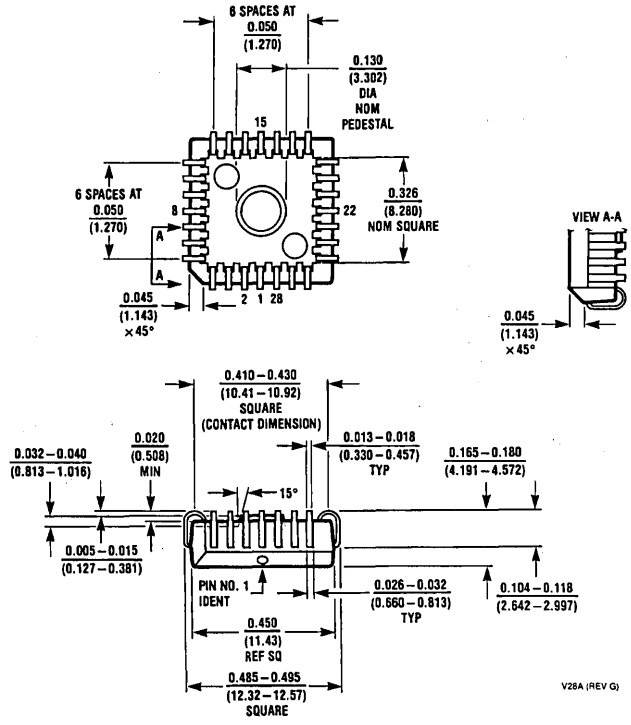
48 Lead Molded Dual-In-Line Package (N)
NS Package Number N48A



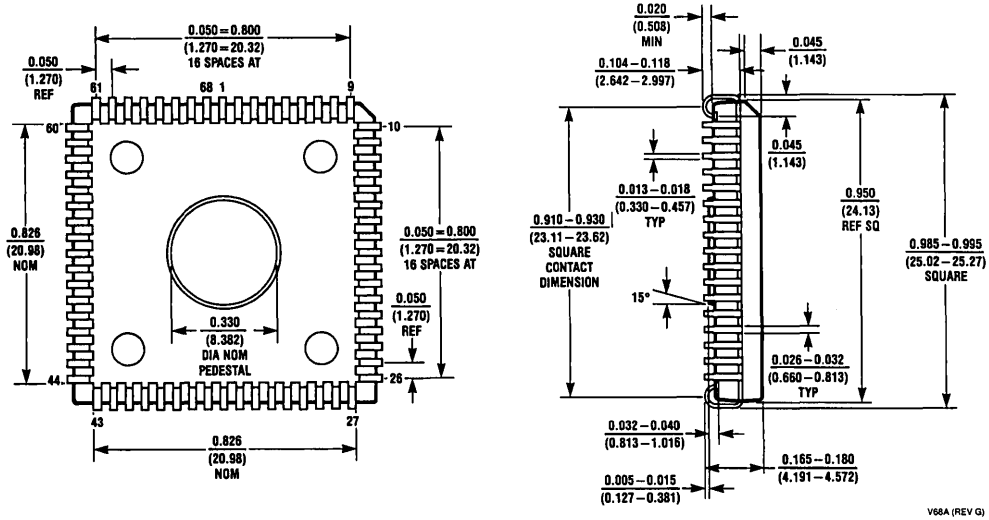
159 Pin Molded Pin Grid Array (UP)
NS Package Number UP159A



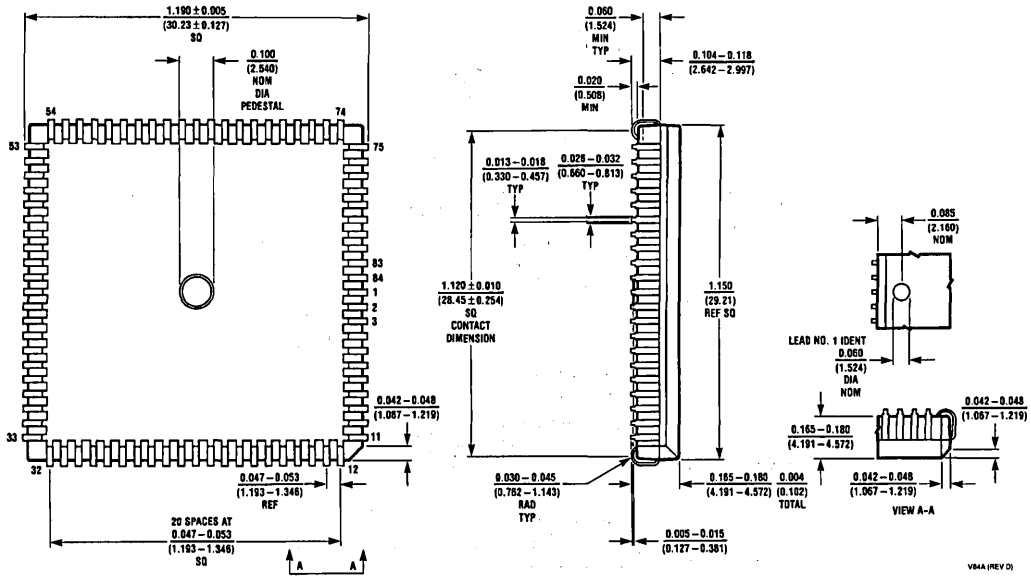
28 Lead Plastic Chip Carrier (V)
NS Package Number V28A



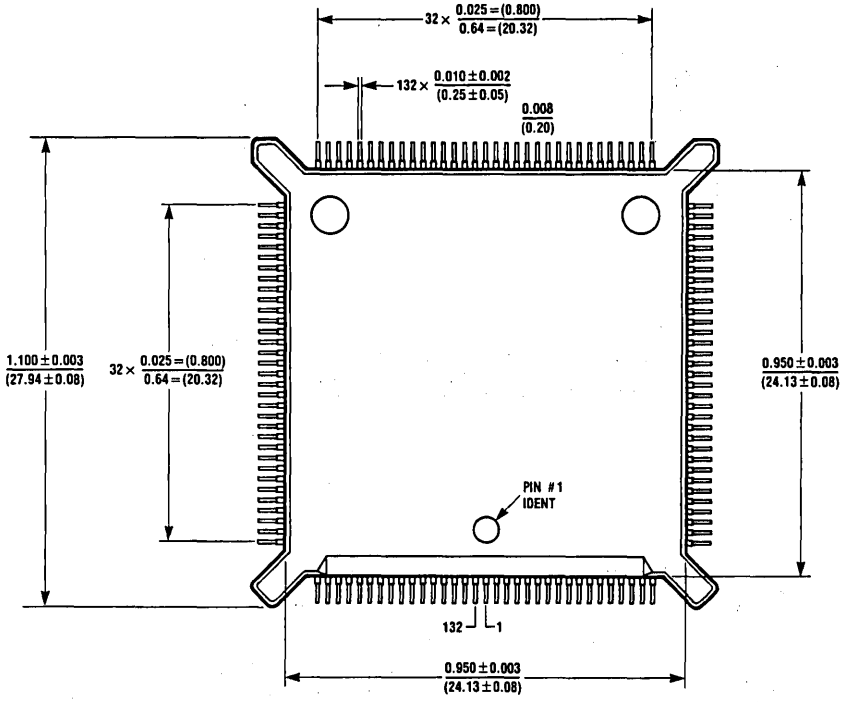
68 Lead Plastic Chip Carrier (V)
NS Package Number V68A

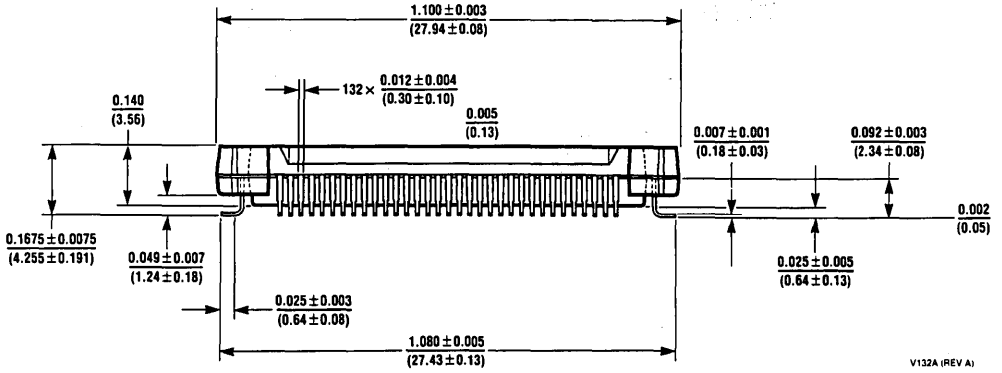


84 Lead Plastic Chip Carrier (V)
NS Package Number V84A

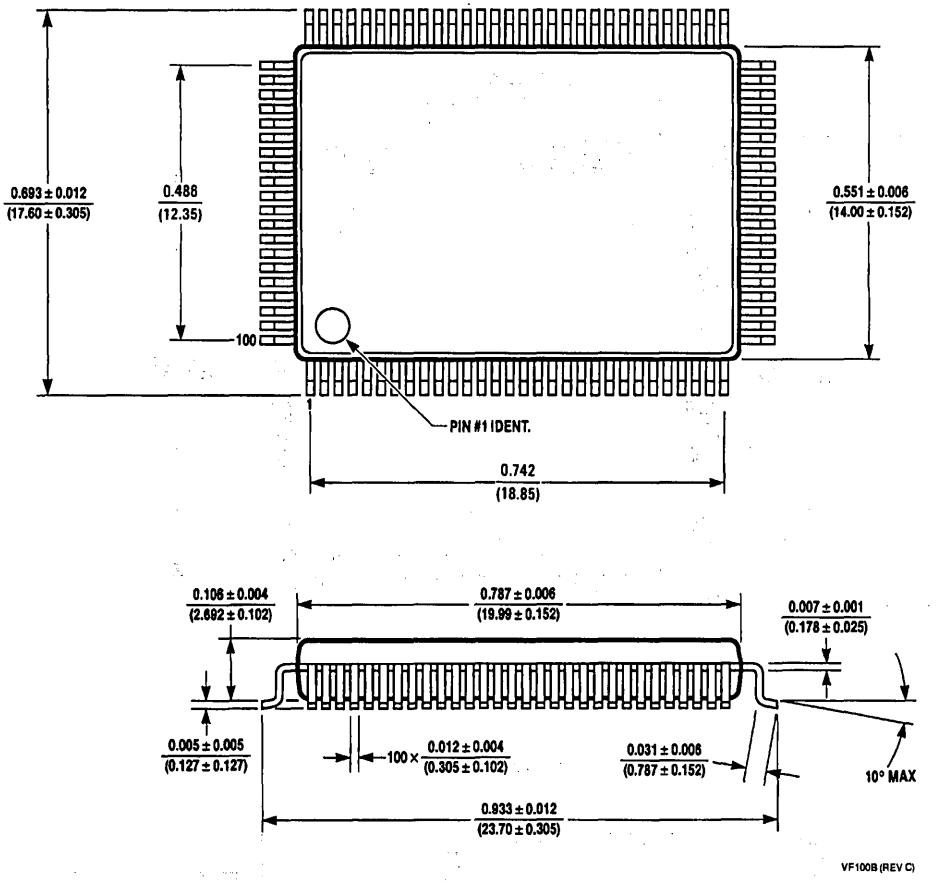


132 Lead Plastic Chip Carrier (V)
NS Package Number V132A





100 Lead Plastic Quad Flatpak (EIAJ) (VF)
NS Package Number VF100B



NOTES

NOTES

National Semiconductor

Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 16-300
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

ALS/AS LOGIC DATABOOK—1990

Introduction to Advanced Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA ACQUISITION LINEAR DEVICES—1989

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital Converters • Digital-to-Analog Converters Sample and Hold • Temperature Sensors • Voltage Regulators • Surface Mount

DISCRETE SEMICONDUCTOR PRODUCTS DATABOOK—1989

Selection Guide and Cross Reference Guides • Diodes • Bipolar NPN Transistors Bipolar PNP Transistors • JFET Transistors • Surface Mount Products • Pro-Electron Series Consumer Series • Power Components • Transistor Datasheets • Process Characteristics

DRAM MANAGEMENT HANDBOOK—1991

Dynamic Memory Control • Error Detection and Correction • Microprocessor Applications for the DP8408A/09A/17/18/19/28/29 • Microprocessor Applications for the DP8420A/21A/22A Microprocessor Applications for the NS32CG821

EMBEDDED SYSTEM PROCESSOR DATABOOK—1989

Embedded System Processor Overview • Central Processing Units • Slave Processors • Peripherals Development Systems and Software Tools

FDDI DATABOOK—1991

FDDI Overview • DP83200 FDDI Chip Set • Development Support • Application Notes and System Briefs

F100K ECL LOGIC DATABOOK & DESIGN GUIDE—1990

Family Overview • 300 Series (Low-Power) Datasheets • 100 Series Datasheets • 11C Datasheets ECL BiCMOS SRAM, ECL PAL, and ECL ASIC Datasheets • Design Guide • Circuit Basics • Logic Design Transmission Line Concepts • System Considerations • Power Distribution and Thermal Considerations Testing Techniques • Quality Assurance and Reliability • Application Notes

FACT™ ADVANCED CMOS LOGIC DATABOOK—1990

Description and Family Characteristics • Ratings, Specifications and Waveforms
Design Considerations • 54AC/74ACXXX • 54ACT/74ACTXXX • Quiet Series: 54ACQ/74ACQXXX
Quiet Series: 54ACTQ/74ACTQXXX • 54FCT/74FCTXXX • FCTA: 54FCTXXXA/74FCTXXXA

FAST® ADVANCED SCHOTTKY TTL LOGIC DATABOOK—1990

Circuit Characteristics • Ratings, Specifications and Waveforms • Design Considerations • 54F/74FXXX

FAST® APPLICATIONS HANDBOOK—1990

Reprint of 1987 Fairchild FAST Applications Handbook
Contains application information on the FAST family: Introduction • Multiplexers • Decoders • Encoders
Operators • FIFOs • Counters • TTL Small Scale Integration • Line Driving and System Design
FAST Characteristics and Testing • Packaging Characteristics

GENERAL PURPOSE LINEAR DEVICES DATABOOK—1989

Continuous Voltage Regulators • Switching Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators
Instrumentation Amplifiers • Surface Mount

GRAPHICS HANDBOOK—1989

Advanced Graphics Chipset • DP8500 Development Tools • Application Notes

INTERFACE DATABOOK—1990

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers
Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

LINEAR APPLICATIONS HANDBOOK—1991

The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LOCAL AREA NETWORK DATABOOK—1992

Integrated Ethernet Network Interface Controller Products • Ethernet Physical Layer Transceivers
Ethernet Repeater Interface Controller Products • Hardware and Software Support Products • FDDI Products • Glossary

LS/S/TTL DATABOOK—1989

Contains former Fairchild Products
Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • TTL—Low Power

MASS STORAGE HANDBOOK—1989

Rigid Disk Pulse Detectors • Rigid Disk Data Separators/Synchronizers and ENDECs
Rigid Disk Data Controller • SCSI Bus Interface Circuits • Floppy Disk Controllers • Disk Drive Interface Circuits
Rigid Disk Preamplifiers and Servo Control Circuits • Rigid Disk Microcontroller Circuits • Disk Interface Design Guide

MEMORY DATABOOK—1990

PROMs, EPROMs, EEPROMs • TTL I/O SRAMs • ECL I/O SRAMs

MICROCONTROLLER DATABOOK—1989

COP400 Family • COP800 Family • COPS Applications • HPC Family • HPC Applications
MICROWIRE and MICROWIRE/PLUS Peripherals • Microcontroller Development Tools

MICROPROCESSOR DATABOOK—1989

Series 32000 Overview • Central Processing Units • Slave Processors • Peripherals
Development Systems and Software Tools • Application Notes • NSC800 Family

PROGRAMMABLE LOGIC DATABOOK & DESIGN MANUAL—1990

Product Line Overview • Datasheets • Designing with PLDs • PLD Design Methodology • PLD Design Development Tools
Fabrication of Programmable Logic • Application Examples

REAL TIME CLOCK HANDBOOK—1991

Real Time Clocks and Timer Clock Peripherals • Application Notes

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETS™ Products • MILS/RETS™ Products • 883/RETS™ Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

SPECIAL PURPOSE LINEAR DEVICES DATABOOK—1989

Audio Circuits • Radio Circuits • Video Circuits • Motion Control Circuits • Special Function Circuits
Surface Mount

TELECOMMUNICATIONS—1990

Line Card Components • Integrated Services Digital Network Components • Analog Telephone Components
Application Notes

National Semiconductor

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1-800-272-9959
TWX: (910) 339-9240

SALES OFFICES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402
Fax: 52-5-524-9342

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 1409
6.0 Andar
Cep. 01451 J. Paulistano
Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Telex: 391 1131931
Fax: (55/11) 212-1181 NSBR BR

National Semiconductor GmbH
Eschborner Lanstr. 130-132
D-6000 Frankfurt 90
Germany
Tel: (069) 78 91 09-0
Fax: (069) 7 89 43 83

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
Germany
Tel: (0-81-41) 103-0
Telex: 527-649
Fax: (08141) 103554

National Semiconductor GmbH
Misburger Strasse 81D
D3000 Hannover 61
Germany
Tel: (0511) 560040
Fax: (0511) 561740

National Semiconductor GmbH
Untere Waldplatze 37
D-7000 Stuttgart 80
Germany
Tel: 711 686 511
Fax: 711 686 5260

National Semiconductor (UK) Ltd.
The Maple, Kembrey Park
Swindon, Wiltshire SN2 6UT
United Kingdom
Tel: (07-93) 61-41-41
Telex: 444-674
Fax: (07-93) 69-75-22

National Semiconductor Benelux
Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6-61-06-80
Telex: 61007
Fax: (02) 6-60-23-95

National Semiconductor (UK) Ltd.
Ringager 4A, 3
DK-2605 Brandy
Denmark
Tel: (02) 43-32-11
Telex: 15-179
Fax: (02) 43-31-11

National Semiconductor S.A.
Centre d'Affaires-La Boursidiere
Bâtiment Champagne, B.P. 90
Route Nationale 186
F-92357 Le Plessis Robinson
Paris, France
Tel: (1) 40-94-88-88
Telex: 631065
Fax: (1) 40-94-88-11

National Semiconductor (UK) Ltd.
Unit 2A
Clonskeagh Square
Clonskeagh Road
Dublin 14
Ireland
Tel: (01) 269-55-89
Telex: 91047
Fax: (01) 2830650

National Semiconductor S.p.A.
Strada 7, Palazzo R/3
I-20089 Rozzano
Milanofiori
Italy
Tel: (02) 57 50 03 00
Twx: 352647
Fax: (02) 57 50 04 00

National Semiconductor S.p.A.
Via del Cararaggio, 107
I-00147 Rome
Italy
Tel: (06) 5-13-48-80
Fax: (06) 5-13-79-47

National Semiconductor (UK) Ltd.
Isveien 45
Postboks 57
N-1393 Ostenstad
Norway
Tel: (2) 796500
Fax: (2) 796040

National Semiconductor AB
P.O. Box 1009
Grosshandlarvaegen 7
S-121 23 Johanneshov
Sweden
Tel: 46-8-7228050
Fax: 46-8-7229095
Telex: 10731 NSC S

National Semiconductor GmbH
Calle Agustin de Foxa, 27 (9°D)
E-28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133
Fax: (01) 733-8018

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
Ch-8304 Wallisellen-Zurich
Switzerland
Tel: (01) 830-2727
Telex: 828-444
Fax: (01) 830-1900

National Semiconductor
Kauppakartanonkatu 7 A22
SF-00930 Helsinki
Finland
Tel: (90) 33-80-33
Telex: 126116
Fax: (90) 33-81-30

National Semiconductor
Postbus 90
NL1380 AB Weesp
The Netherlands
Tel: (0-29-40) 3-04-48
Telex: 10-956
Fax: (0-29-40) 3-04-30

National Semiconductor Japan Ltd.
Sanseido Bldg. 5F
4-15-3 Nishi Shinjuku
Shinjuku-ku
Tokyo 160 Japan
Tel: (03) 3299-7001
Fax: (03) 3299-7000

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block
Ocean Centre
5 Canton Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: (852) 737-1600
Telex: 51292 NSHKL
Fax: (852) 736-9960

National Semiconductor (Australia) PTY, Ltd.
Bldg. 16, Business Park Dr.
Melbourne, 3168
Victoria, Australia
Tel: (03) 558-9999
Fax: 61-3-558-9998

National Semiconductor (PTE), Ltd.
200 Cantonment Road 13-02
Southpoint 200
Singapore 0208
Tel: 2252229
Telex: RS 50808
Fax: (65) 225-7080

National Semiconductor (Far East) Ltd.
Taiwan Branch
9th Floor, No. 18
Sec. 1, Chang An East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 521-3288
Telex: 22837 NSTW
Fax: 02 561-3054

National Semiconductor (Far East) Ltd.
Korea Branch
13th Floor, Dai Han Life Insurance
63 Building,
60, Yoido-dong, Youngdeungpo-ku,
Seoul, Korea 150-763
Tel: (02) 784-8051
Telex: 24942 NSPKLO
Fax: (02) 784-8054

 TL3443 RRD-RRD35M101/Printed in U.S.A.