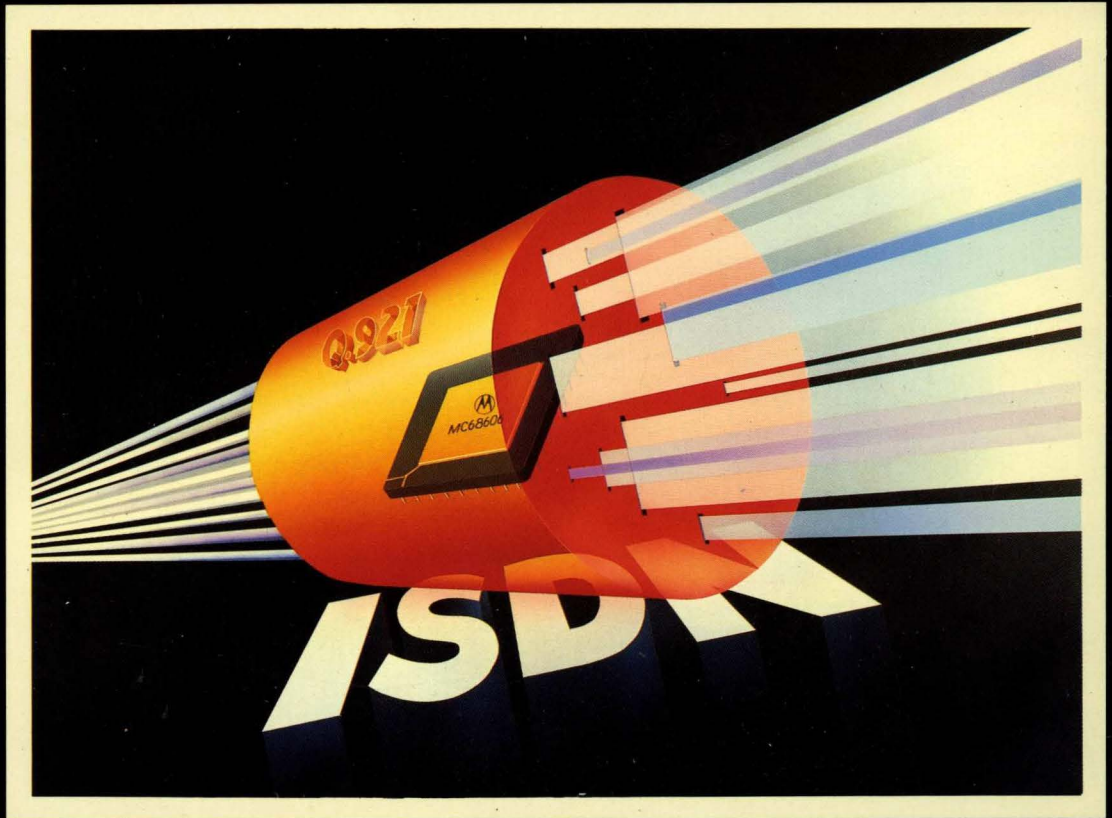



# MC68606 Multi-Link LAPD Protocol Controller User's Manual



This document contains information on a new product. Specifications and information herein are subject to change without notice. Motorola reserves the right to make changes to any products herein to improve functioning or design. Although the information in this document has been carefully reviewed and is believed to be reliable, Motorola 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.

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola 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. Motorola products are not authorized for use as components in life support devices or systems intended for surgical implant into the body or intended to support or sustain life. Buyer agrees to notify Motorola of any such intended end use whereupon Motorola shall determine availability and suitability of its product or products for the use intended. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

INTRODUCTION	1
MEMORY STRUCTURES	2
INTERNAL REGISTERS	3
COMMAND SET	4
TRANSMIT PROCESS	5
RECEIVE PROCESS	6
EXCEPTION PROCESSING	7
MLAPD IMPLEMENTATION OF LAPD	8
MLAPD IMPLEMENTATION OF SPECIAL MODES	9
SIGNAL DESCRIPTION	10
BUS OPERATION	11
ELECTRICAL SPECIFICATIONS	12
MECHANICAL DATA	13
SOFTWARE INTERFACE FLOWS FOR THE MLAPD	A
ABBREVIATIONS	B

- 1 INTRODUCTION**
- 2 MEMORY STRUCTURES**
- 3 INTERNAL REGISTERS**
- 4 COMMAND SET**
- 5 TRANSMIT PROCESS**
- 6 RECEIVE PROCESS**
- 7 EXCEPTION PROCESSING**
- 8 MLAPD IMPLEMENTATION OF LAPD**
- 9 MLAPD IMPLEMENTATION OF SPECIAL MODES**
- 10 SIGNAL DESCRIPTION**
- 11 BUS OPERATION**
- 12 ELECTRICAL SPECIFICATIONS**
- 13 MECHANICAL DATA**
- A SOFTWARE INTERFACE FLOWS FOR THE MLAPD**
- B ABBREVIATIONS**



## TABLE OF CONTENTS

Paragraph Number	Title	Page Number
<b>Section 1</b>		
<b>Introduction</b>		
1.1	Feature Overview.....	1-1
1.2	ISDN Communication Model.....	1-3
1.3	LAPD Frame Format.....	1-3
1.4	System Environment.....	1-6
1.4.1	Physical Link Interface.....	1-7
1.4.2	Microprocessor System Bus Interface.....	1-8
1.4.3	Shared-Memory Control Components.....	1-8
1.5	Overview of MLAPD Functional Operation.....	1-9
1.5.1	MLAPD Internal States and Command Structure Overview.....	1-9
1.5.2	Interrupt Structure Overview.....	1-10
1.5.3	Initialization Overview.....	1-11
1.5.4	Frame Reception Overview.....	1-11
1.5.5	Frame Transmission Overview.....	1-12
<b>Section 2</b>		
<b>Memory Structures</b>		
2.1	Global Configuration Block.....	2-2
2.1.1	Constants Area.....	2-3
2.1.1.1	Option Bits 1.....	2-5
2.1.1.2	Option Bits 2.....	2-8
2.1.1.3	Match Table Pointer.....	2-9
2.1.1.4	L2 Queue Pointer.....	2-9
2.1.1.5	L2 Queue Length.....	2-9
2.1.1.6	Timer Table Pointer.....	2-9
2.1.1.7	LLID-LLT Table Pointer.....	2-9
2.1.1.8	Interrupt Queue Pointer.....	2-9
2.1.1.9	Interrupt Queue Length.....	2-10
2.1.1.10	Pool Table Pointer.....	2-10
2.1.2	Reloadable Variables Area.....	2-10
2.1.2.1	Pad Time Select.....	2-10
2.1.2.2	Nonstandard Control.....	2-10
2.1.2.3	T203 Value.....	2-11
2.1.2.4	I Frame Retransmit Threshold.....	2-12
2.1.2.5	N200 Value.....	2-12

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.1.2.6	REJ Transmit Threshold .....	2-12
2.1.2.7	RNR Transmit Threshold .....	2-12
2.1.2.8	REJ Receive Threshold.....	2-13
2.1.2.9	RNR Receive Threshold.....	2-13
2.1.2.10	CTS Timeout Threshold.....	2-13
2.1.2.11	Scan Length I Queue 0.....	2-13
2.1.2.12	Scan Length I Queue 1.....	2-13
2.1.2.13	Scan Length I Queue 2.....	2-13
2.1.2.14	Scan Length I Queue 3.....	2-14
2.1.2.15	Scan Length XID/UI Queue 0 .....	2-14
2.1.2.16	Scan Length XID/UI Queue 1 .....	2-14
2.1.2.17	Interrupt Mask.....	2-14
2.1.2.18	Protocol Error Mask.....	2-14
2.1.2.19	Nonprotocol Error Mask.....	2-16
2.1.2.20	Filter Mask and Filter Match .....	2-18
2.1.3	Statistics Threshold Area .....	2-18
2.1.3.1	RxFIFO Overrun Threshold.....	2-18
2.1.3.2	TxFIFO Underrun Threshold .....	2-18
2.1.3.3	Inactive DLCI Threshold.....	2-19
2.1.3.4	Invalid Address Threshold .....	2-19
2.1.3.5	Discarded Frame Threshold.....	2-19
2.1.3.6	Short Frame Threshold .....	2-19
2.1.3.7	CRC Error Threshold.....	2-19
2.1.3.8	Abort or Nonoctet Threshold.....	2-19
2.1.4	Command Arguments Area .....	2-20
2.1.4.1	Command Argument 1.....	2-20
2.1.4.2	Command Argument 2.....	2-20
2.1.4.3	Command Argument 3.....	2-20
2.1.5	User Area .....	2-20
2.1.5.1	User Interrupt Queue Read Pointer .....	2-20
2.1.5.2	User Global XID/UI Tx Next Confirm Pointer .....	2-20
2.1.5.3	User Global XID/UI Tx Last Queued Pointer .....	2-20
2.1.5.4	User XID/UI Tx Next Confirm Pointer .....	2-20
2.1.5.5	User XID/UI Tx Last Queued Pointer.....	2-20
2.1.5.6	User XID/UI Tx Next Confirm Pointer .....	2-21
2.1.5.7	User XID/UI Tx Last Queued Pointer.....	2-21
2.2	Match Table.....	2-21
2.3	LLID-LLT Table .....	2-22
2.4	Logical-Link Table .....	2-23
2.4.1	Transmit Status .....	2-24
2.4.2	DLCI and Configuration Bits.....	2-26

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.4.3	Maximum Number of Outstanding I Frames (K) .....	2-26
2.4.4	V(A) .....	2-26
2.4.5	V(S)/V(R) .....	2-26
2.4.6	Link Status .....	2-27
2.4.7	Tx I Queue Number .....	2-29
2.4.8	T200 Value .....	2-29
2.4.9	N201 Value .....	2-29
2.4.10	Receive Pool Number and Error Mask Valid (EVM) .....	2-31
2.4.11	Tx Next Pointer .....	2-31
2.4.12	Tx Next Acknowledge Pointer .....	2-31
2.4.13	Tx Next LLT Pointer .....	2-31
2.4.14	REJ Tx Counter .....	2-32
2.4.15	RNR Tx Counter .....	2-32
2.4.16	REJ Rx Counter .....	2-32
2.4.17	RNR Rx Counter .....	2-33
2.4.18	User Tx Next Confirm Pointer .....	2-33
2.4.19	User Tx Last Queued Pointer .....	2-33
2.5	Receive Pool Pointers Table .....	2-34
2.6	Transmit Frame Descriptor .....	2-35
2.6.1	Status Bits .....	2-36
2.6.2	Next Tx Frame Descriptor Pointer .....	2-37
2.6.3	Data Buffer Pointer .....	2-37
2.6.4	Control Bits and Data Length .....	2-37
2.6.5	Frame Type and LLID .....	2-38
2.6.6	Header Pointer .....	2-39
2.6.7	Header Length .....	2-39
2.6.8	Retransmit Count .....	2-39
2.7	Receive Frame Descriptor .....	2-39
2.7.1	Control Bits .....	2-40
2.7.2	Next Rx Frame Descriptor Pointer .....	2-41
2.7.3	Data Buffer Pointer .....	2-41
2.7.4	Data Length .....	2-41
2.7.5	Frame Type and LLID .....	2-42
2.7.6	Time Stamp .....	2-44
2.7.7	Error Code .....	2-44
2.8	Interrupt Queue .....	2-44
2.8.1	Interrupt Events .....	2-47
2.8.2	Interrupt Arguments .....	2-50
2.8.2.1	MDL Error Indication Arguments .....	2-50
2.8.2.2	Link Counter Threshold Reached Arguments .....	2-52

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
2.9	Timer Table .....	2-52
2.9.1	Timer Operations .....	2-53
2.10	Level 2 Queue .....	2-55

### Section 3

#### Internal Registers

3.1	Directly Accessible Registers .....	3-1
3.1.1	Command Register .....	3-1
3.1.2	Semaphore Register .....	3-2
3.1.3	Interrupt Vector Register .....	3-2
3.1.4	Data Register .....	3-3
3.2	Indirectly Accessible Registers .....	3-4
3.2.1	CAM .....	3-4
3.2.2	N200 Value .....	3-5
3.2.3	T203 Value .....	3-5
3.2.4	Highest Active LLID .....	3-5
3.2.5	T200 Counter .....	3-6
3.2.6	T203 Counter .....	3-6
3.2.7	Pool Number .....	3-6
3.2.8	L2 Tail Displacement .....	3-6
3.2.9	External L2 Queue Displacement Registers .....	3-6
3.2.10	Tx XID/UI LLID .....	3-7
3.2.11	Tx XID/UI DLCI .....	3-7
3.2.12	Internal L2 Queue Displacement Registers .....	3-7
3.2.13	Tx FD Status .....	3-7
3.2.14	Tx V(S)/V(R) .....	3-7
3.2.15	CTS Timeout Counter .....	3-8
3.2.16	Tx Maximum Number of Outstanding I Frames (K) .....	3-8
3.2.17	Tx V(A) .....	3-8
3.2.18	Tx Link Status .....	3-8
3.2.19	Current Queue In-Service .....	3-8
3.2.20	Scan Length Counter .....	3-9
3.2.21	Tx I LLID .....	3-9
3.2.22	Tx I DLCI .....	3-9
3.2.23	Retransmit Threshold .....	3-9
3.2.24	Rx Address .....	3-10
3.2.25	Rx N201 .....	3-10
3.2.26	Rx Link Status .....	3-10
3.2.27	Rx LLID .....	3-10
3.2.28	Rx DLCI .....	3-10

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
3.2.29	Rx Frame ID .....	3-10
3.2.30	Tx Address .....	3-11
3.2.31	Time Stamp .....	3-11
3.2.32	RxFIFO Overrun Counter .....	3-11
3.2.33	TxFIFO Underrun Counter .....	3-11
3.2.34	Invalid Address Counter .....	3-11
3.2.35	Inactive DLCI Counter .....	3-11
3.2.36	Discarded Frame Counter .....	3-12
3.2.37	Short Frame Counter .....	3-12
3.2.38	CRC Error Counter .....	3-12
3.2.39	Abort/Nonoctet Counter .....	3-12
3.2.40	Tx REJ/RNR Counter .....	3-12
3.2.41	Rx REJ/RNR Counter .....	3-13
3.2.42	CTS Timeout Threshold .....	3-13
3.2.43	Scan Length I Queues 0-3 .....	3-13
3.2.44	Scan Length XID/UI Queues 0, 1 .....	3-14
3.2.45	Interrupt Mask .....	3-14
3.2.46	Protocol Error Mask .....	3-14
3.2.47	Nonprotocol Error Mask .....	3-14
3.2.48	Rx Maximum Number of Outstanding Frames or Filter Mask (Word 1) .....	3-14
3.2.49	Rx V(A) or Filter Mask (Word 2) .....	3-15
3.2.50	Rx V(S)/V(R) or Filter Match (Word 1) .....	3-15
3.2.51	Rx Control or Filter Match (Word 2) .....	3-15
3.2.52	Tx LLT Transmit Status .....	3-15
3.2.53	Nonstandard Control Field .....	3-16
3.2.54	Revision Number .....	3-16
3.2.55	Tx Next Pointer .....	3-16
3.2.56	Tx LLT Pointer .....	3-16
3.2.57	Global XID/UI Head Pointer .....	3-16
3.2.58	I Queue 0-3 Head Pointer Registers .....	3-16
3.2.59	I Queue 0-3 Tail Pointer Registers .....	3-17
3.2.60	L2 Queue Pointer .....	3-17
3.2.61	Match Table Pointer .....	3-17
3.2.62	LLID-LLT Table Pointer .....	3-17
3.2.63	Timer Table Pointer .....	3-17
3.2.64	Interrupt Queue Tail Pointer .....	3-17
3.2.65	Interrupt Queue Pointer .....	3-18
3.2.66	Interrupt Queue Write Pointer .....	3-18
3.2.67	First Rx FD Pointer .....	3-18
3.2.68	GCB Pointer .....	3-18

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
3.2.69	Temporary Rx FD Pointer .....	3-18
3.2.70	Rx LLT Pointer .....	3-19
3.2.71	Rx Pool Pointer .....	3-19
3.2.72	Rx Next Acknowledge Pointer .....	3-19
3.2.73	Rx Next Tx Pointer .....	3-19
3.2.74	Rx Current FD Pointer .....	3-19
3.2.75	Rx Next FD Pointer .....	3-19
3.2.76	Pool Table Pointer .....	3-20
3.2.77	XID/UI Queue 0 Head Pointer .....	3-20
3.2.78	XID/UI Queue 1 Head Pointer .....	3-20
3.2.79	Tx XID/UI LLT Pointer .....	3-20
3.2.80	Tx XID/UI FD Pointer .....	3-20
3.2.81	Bus/Address Error Pointers .....	3-20
3.2.82	DMA Rx Data Pointer .....	3-21
3.2.83	DMA Tx Data Pointer .....	3-21
3.2.84	DMA Rx Data Counter .....	3-21
3.2.85	DMA Tx Data Counter .....	3-21
3.2.86	DMA General-Purpose Pointers .....	3-21

### Section 4 Command Set

4.1	Initialization .....	4-1
4.1.1	RESET Command .....	4-1
4.1.2	SET_BUS_WIDTH_8 Command .....	4-2
4.1.3	SET_BUS_WIDTH_16 Command .....	4-2
4.1.4	INIT Command .....	4-2
4.2	Test/Diagnostic .....	4-3
4.2.1	DMA_TEST Command .....	4-3
4.2.2	DUMP Command .....	4-4
4.3	Host — MLAPD Interface .....	4-9
4.3.1	OFF-LINE Command .....	4-9
4.3.2	ON-LINE Command .....	4-9
4.3.3	RELOAD Command .....	4-9
4.3.4	DUMP_STATISTICS Command .....	4-11
4.3.5	PRESET_STATISTICS Command .....	4-11
4.3.6	ENABLE_IRQ Command .....	4-12
4.3.7	ASSIGN_POOL_POINTER Command .....	4-12
4.4	Protocol .....	4-12
4.4.1	MDL_ASSIGN_REQUEST Command .....	4-13
4.4.2	DL_ESTABLISH_REQUEST Command .....	4-13

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
4.4.3	DL_DATA_REQUEST Command .....	4-14
4.4.4	RELINK_REQUEST Command .....	4-15
4.4.5	GLOBAL_XID/UI_REQUEST Command .....	4-15
4.4.6	XID/UI_QUEUE_0_REQUEST Command .....	4-16
4.4.7	XID/UI_QUEUE_1_REQUEST Command .....	4-16
4.4.8	MDL_ERROR_RESPONSE Command .....	4-17
4.4.9	DL_RELEASE_REQUEST Command .....	4-17
4.4.10	MDL_REMOVE_REQUEST Command .....	4-18
4.5	Protocol Extension .....	4-18
4.5.1	ACTIVATE_LL Command .....	4-18
4.5.2	DEACTIVATE_LL Command .....	4-19
4.5.3	REMOTE_STATUS_REQUEST Command .....	4-20
4.5.4	SET_LOCAL_BUSY Command .....	4-20
4.5.5	CLEAR_LOCAL_BUSY Command .....	4-21
4.5.6	STOP_TX_I Command .....	4-21
4.5.7	STOP_GLOBAL_XID/UI Command .....	4-21
4.5.8	STOP_XID/UI_QUEUE_0 Command .....	4-22
4.5.9	STOP_XID/UI_QUEUE_1 Command .....	4-22
4.6	Command Summary .....	4-22

### Section 5

#### Transmit Process

5.1	Transmit Servicing Scheme .....	5-1
5.2	Transmit I Frame Process .....	5-2
5.2.1	I Frame Queue Structure .....	5-3
5.2.2	Logical-Link Transmit Queue Structure .....	5-4
5.2.3	I Frame Transmission Queueing .....	5-5
5.2.4	MLAPD I Frame Queue Processing .....	5-6
5.2.5	Stopping I Frame Transmission .....	5-7
5.2.6	Collecting Acknowledged Frames .....	5-8
5.3	Transmit XID/UI Frame Processing .....	5-8
5.3.1	XID/UI Queue Structure .....	5-8
5.3.2	XID/UI Transmit Queue Usage .....	5-9
5.3.3	XID/UI Frame Transmission Queueing .....	5-9
5.3.4	MLAPD XID/UI Queue Processing .....	5-10
5.3.5	Stopping XID/UI Frame Transmission .....	5-11
5.3.6	Collecting Transmitted XID/UI Frames .....	5-11
5.4	Errors During Frame Transmission .....	5-12

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
<b>Section 6</b>		
<b>Receive Process</b>		
6.1	Receive Data Structures .....	6-1
6.1.1	Receive Pool Format .....	6-1
6.1.2	Receive Queue Format .....	6-3
6.1.3	User Receive Pointers.....	6-3
6.2	MLAPD Receive Processing.....	6-4
6.3	Collecting Received Frames.....	6-4
<b>Section 7</b>		
<b>Exception Processing</b>		
7.1	Interrupt Mechanism .....	7-1
7.1.1	Interrupt Queue Handling.....	7-1
7.1.2	Selecting Polled or Interrupt-Driven Operation.....	7-1
7.1.3	Collecting Interrupt Events .....	7-2
7.1.4	Interrupt Queue Overflow.....	7-2
7.2	Bus Error Operation .....	7-3
7.3	Address Error Operation.....	7-3
<b>Section 8</b>		
<b>MLAPD Implementation of LAPD</b>		
8.1	MLAPD Initialization Procedure .....	8-1
8.2	Initialization of Logical Links.....	8-2
8.2.1	Broadcast Link Initialization .....	8-2
8.2.1.1	Broadcast Link Initialization Procedure.....	8-3
8.2.2	Signaling Link Initialization .....	8-3
8.3	Assignment Procedure.....	8-4
8.4	Link Setup Procedure.....	8-4
8.5	Link Release Procedure .....	8-5
8.6	Collision of Unnumbered Command Frames.....	8-7
8.7	Transmit Procedure.....	8-7
8.7.1	Information Frame Preparation.....	8-7
8.7.2	Information Frame Transmission .....	8-7
8.7.3	XID/UI/Nonstandard-control Frame Preparation .....	8-8
8.7.4	XID/UI/Nonstandard-control Frame Transmission.....	8-9
8.8	Reception Procedure.....	8-9
8.8.1	Information Frame Reception.....	8-9
8.8.2	Invalid Frames Reception .....	8-10



## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
8.8.3	Bad Frames Reception.....	8-11
8.8.4	Frame Reject Mode.....	8-11
8.8.5	XID/UI/Nonstandard-control Frame Reception .....	8-11
8.8.6	Receiving an Out-Of-Sequence I Frame .....	8-11
8.8.7	Receiving a FRMR Frame .....	8-12
8.9	Frame Types Allowed in Each Link State.....	8-12
8.9.1	TEI_UNASSIGN and EST_WAIT_TEI States .....	8-12
8.9.2	TEI_ASSIGNED, AWAIT_EST, and AWAIT_REL States .....	8-12
8.9.3	Connected States .....	8-13
8.10	Flow Control and Error Control Procedures.....	8-13
8.10.1	Local Busy Condition .....	8-14
8.10.2	Awaiting Acknowledgement .....	8-14
8.10.3	Receiving Acknowledgement .....	8-15
8.10.4	Receiving a REJ Frame.....	8-15
8.10.5	Receiving a RNR Frame .....	8-15
8.11	Error Handling Options .....	8-15
8.11.1	Frame Reject Mode.....	8-16
8.11.2	CCITT/DMI Mode.....	8-16

### Section 9

#### MLAPD Implementation of Special Modes

9.1	Nonprotocol Links.....	9-1
9.1.1	Setup Procedure .....	9-2
9.1.2	Release Procedure.....	9-2
9.1.3	Queuing Frames for Transmission .....	8-2
9.1.4	MLAPD Transmit Queue Processing .....	9-3
9.1.5	MLAPD Frame Transmission.....	9-3
9.1.6	Stopping Frame Transmission.....	9-4
9.1.7	Collecting Transmitted Frames.....	9-4
9.1.8	Receive Structures.....	9-5
9.1.9	MLAPD Frame Reception .....	9-5
9.1.10	Restrictions .....	9-6
9.2	Promiscuous Receive Mode .....	9-6
9.2.1	Multibuffer Mode .....	9-7
9.2.2	Filter Mode .....	9-7
9.2.3	Restrictions.....	9-8
9.3	Parallel Assist Mode.....	9-8
9.4	Line Monitor.....	9-9
9.5	System Loopback Testing.....	9-10
9.6	Memory-to-Memory Operation .....	9-10
9.6.1	Handling Received Frames .....	9-11
9.6.2	Handling Transmit Frames .....	9-12

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
<b>Section 10</b>		
<b>Signal Description</b>		
10.1	Serial Interface .....	10-2
10.1.1	Modem Control Signals .....	10-2
10.1.1.1	Request-to-Send ( $\overline{RTS}$ ) or Transmit Start ( $\overline{TSRART}$ ) .....	10-2
10.1.1.2	Clear-to-Send ( $\overline{CTS}$ ) or Receive Start ( $\overline{RSTART}$ ) .....	10-2
10.1.2	Transmit Signals .....	10-3
10.1.2.1	Transmit Clock (TxCLK) .....	10-3
10.1.2.2	Transmit Data (TxD) .....	10-3
10.1.3	Receive Signals .....	10-3
10.1.3.1	Receive Clock (RxCLK) .....	10-3
10.1.3.2	Receive Data (RxD) .....	10-4
10.2	Motorola Bus Interface .....	10-4
10.2.1	Motorola/Intel Mode (MOT/INT) .....	10-4
10.2.2	Address Bus (A1–A23) .....	10-4
10.2.3	Data Bus (D0–D15) .....	10-5
10.2.4	Bus Control Signals .....	10-5
10.2.4.1	Chip Select ( $\overline{CS}$ ) .....	10-5
10.2.4.2	Address Strobe ( $\overline{AS}$ ) .....	10-5
10.2.4.3	Read/Write ( $R/\overline{W}$ ) .....	10-5
10.2.4.4	Upper Data Strobe ( $\overline{UDS}/A0$ ) and Lower Data Strobe ( $\overline{LDS}/\overline{DS}$ ) .....	10-5
10.2.4.5	Data Transfer Acknowledge ( $\overline{DTACK}$ ) .....	10-6
10.2.5	Bus Arbitration Signals .....	10-6
10.2.5.1	Bus Request ( $\overline{BR}$ ) .....	10-6
10.2.5.2	Bus Grant ( $\overline{BG}$ ) .....	10-6
10.2.5.3	Bus Grant Acknowledge ( $\overline{BGACK}$ ) .....	10-6
10.2.6	Interrupt Control Signals .....	10-7
10.2.6.1	Interrupt Request ( $\overline{IRQ}$ ) .....	10-7
10.2.6.2	Interrupt Acknowledge ( $\overline{IACK}$ ) .....	10-7
10.2.7	Bus Exception Signals .....	10-7
10.2.7.1	Reset ( $\overline{RESET}$ ) .....	10-7
10.2.7.2	Halt ( $\overline{HALT}$ ) .....	10-7
10.2.7.3	Bus Error ( $\overline{BERR}$ ) .....	10-7
10.2.7.4	Retry ( $\overline{RETRY}$ ) .....	10-7
10.2.8	System Clock (CLK) Signal .....	10-8
10.2.9	Motorola Signal Summary .....	10-8
10.3	Intel-Compatible Bus Interface .....	10-9
10.3.1	Motorola/Intel Mode (MOT/INT) .....	10-9
10.3.2	Address Bus (A0–A23) .....	10-9
10.3.3	Data Bus (D0–D15) .....	10-9

## TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
10.3.4	Bus Control Signals .....	10-9
10.3.4.1	Chip Select ( $\overline{CS}$ ).....	10-9
10.3.4.2	Address Strobe ( $\overline{AS}$ ).....	10-9
10.3.4.3	Read ( $\overline{RD}$ ).....	10-10
10.3.4.4	Write ( $\overline{WR}$ ) .....	10-10
10.3.4.5	Bus High Enable ( $\overline{BHE}$ ).....	10-10
10.3.4.6	Ready (READY).....	10-10
10.3.5	Bus Arbitration Signals.....	10-11
10.3.5.1	Hold Request (HRQ) .....	10-11
10.3.5.2	Hold Acknowledge (HOLDA).....	10-11
10.3.6	Interrupt Control Signals.....	10-11
10.3.6.1	Interrupt Request (INTR).....	10-11
10.3.6.2	Interrupt Acknowledge (INTA).....	10-11
10.3.7	Bus Exception Signals .....	10-11
10.3.7.1	Reset ( $\overline{RESET}$ ) .....	10-11
10.3.7.2	Halt ( $\overline{HALT}$ ).....	10-12
10.3.7.3	Bus Error ( $\overline{BERR}$ ) .....	10-12
10.3.7.4	Retry ( $\overline{RETRY}$ ) .....	10-12
10.3.8	System Clock (CLK) Signal .....	10-12
10.3.9	Intel Signal Summary.....	10-12

### Section 11 Bus Operation

11.1	Motorola Bus Operation .....	11-1
11.1.1	Slave Operation Mode.....	11-1
11.1.1.1	Host Processor Read Cycle.....	11-1
11.1.1.2	Host Processor Write Cycle .....	11-2
11.1.2	Interrupt Acknowledge Cycle.....	11-2
11.1.3	Master Operation .....	11-3
11.1.3.1	DMA Priority Scheme .....	11-3
11.1.3.2	MLAPD Read Cycles .....	11-4
11.1.3.3	MLAPD Write Cycles.....	11-5
11.1.4	Bus Arbitration .....	11-6
11.1.5	Bus Exception Control Functions .....	11-6
11.1.5.1	Halt.....	11-7
11.1.5.2	Bus Error .....	11-7
11.1.5.3	Retry.....	11-7
11.1.5.4	Reset.....	11-8

## TABLE OF CONTENTS (Concluded)

Paragraph Number	Title	Page Number
11.2	Intel-Compatible Bus Operation .....	11-8
11.2.1	Slave Operation Mode.....	11-8
11.2.1.1	Host Processor Read Cycle.....	11-10
11.2.1.2	Host Processor Write Cycle .....	11-10
11.2.2	Interrupt Acknowledge Cycle.....	11-11
11.2.3	Master Operation .....	11-12
11.2.3.1	DMA Priority Scheme .....	11-12
11.2.3.2	MLAPD Read Cycle.....	11-13
11.2.3.3	MLAPD Write Cycle .....	11-13
11.2.4	Bus Arbitration .....	11-13
11.2.5	Bus Exception Control Functions .....	11-14
11.2.5.1	Halt.....	11-15
11.2.5.2	Bus Error .....	11-15
11.2.5.3	Retry.....	11-16
11.2.5.4	Reset.....	11-16

### Section 12

#### Electrical Specifications

12.1	Maximum Ratings.....	12-1
12.2	Thermal Characteristics.....	12-1
12.3	Power Considerations.....	12-1
12.4	DC Electrical Characteristics .....	12-2
12.5	AC Electrical Characteristics .....	12-2
12.5.1	Motorola AC Electrical Characteristics .....	12-2
12.5.2	Intel-Compatible AC Electrical Characteristics .....	12-17

### Section 13

#### Mechanical Data

13.1	Package Types .....	13-1
13.2	Pin Assignments.....	13-1
13.3	Package Dimensions.....	13-3

### Appendix A

#### Software Interface Flows for the MALPD

### Appendix B

#### Abbreviations

## LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
1-1	Data Switching of the Future .....	1-2
1-2	LAPD In Relation to ISDN and ISO Models .....	1-4
1-3	LAPD Frame Formats .....	1-5
1-4	Address Field Format .....	1-5
1-5	Control Field Formats .....	1-5
1-6	Command and Response Frames .....	1-6
1-7	MLAPD Protocol Controller Application Environment .....	1-7
1-8	Receive Process for Expanded Operation Mode .....	1-12
1-9	Receive Process for On-Chip Operation Mode .....	1-13
2-1	Shared Memory Tables .....	2-1
2-2	GCB Format — Constants .....	2-2
2-3	GCB Format — Reloadable Variables .....	2-3
2-4	GCB Format — Statistics Threshold .....	2-4
2-5	GCB Format — Command Arguments .....	2-4
2-6	GCB — User Area .....	2-5
2-7	Interrupt Mask Bits .....	2-15
2-8	Match Table Format .....	2-22
2-9	LLID-LLT Table .....	2-22
2-10	Logical-Link Table Format .....	2-23
2-11	Transmit Status Bits Transition Table .....	2-25
2-12	Receive Pool Pointers Table .....	2-34
2-13	Transmit Frame Configuration .....	2-35
2-14	Transmit Frame Descriptor .....	2-25
2-15	Receive Frame Descriptor .....	2-40
2-16	Interrupt Queue Structure .....	2-45
2-17	Interrupt Queue Entry Format .....	2-45
2-18	Timer Table Format .....	2-53
2-19	MLAPD Generated Level 2 Frames .....	2-55
2-20	Level 2 Queue Structure .....	2-56
4-1	DMA Transfer Configuration .....	4-4
4-2	Dump Registers Memory Map .....	4-4
5-1	Transmit Servicing Scheme .....	5-2
5-2	Transmit Servicing Flowchart .....	5-3
5-3	Structure of Each I Frame Queue .....	5-4
5-4	Logical-Link Transmit Queue Structure .....	5-5
5-5	XID/UI Transmit Queue Structure .....	5-9

## LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
6-1	Receive Data Structure.....	6-2
6-2	User Receive Pointers Table.....	6-3
8-1	MLAPD Protocol States.....	8-2
9-1	Memory-to-Memory Operation.....	9-11
10-1	MC68606 Signals.....	10-1
11-1	Motorola Typical Host Byte Read Cycle Timing Diagram.....	11-2
11-2	Motorola Typical Host Word Write Cycle Timing Diagram.....	11-3
11-3	Motorola Typical Interrupt Acknowledge Cycle Timing Diagram.....	11-3
11-4	Motorola Typical DMA Read Cycle Timing Diagram.....	11-5
11-5	Motorola Typical DMA Write Cycle Timing Diagram.....	11-5
11-6	Motorola Typical Bus Arbitration Timing Diagram.....	11-6
11-7	Motorola Typical Write Cycle with $\overline{\text{HALT}}$ Timing Diagram.....	11-7
11-8	Motorola Typical Read Cycle with $\overline{\text{BERR}}$ Timing Diagram.....	11-8
11-9	Motorola Typical Read Cycle with $\overline{\text{RETRY}}$ Timing Diagram.....	11-9
11-10	Intel-Compatible Typical Host Byte Read Cycle Timing Diagram.....	11-10
11-11	Intel-Compatible Typical Host Word Write Cycle Timing Diagram.....	11-11
11-12	Intel-Compatible Typical Interrupt Acknowledge Cycle Timing Diagram.....	11-11
11-13	Intel-Compatible Typical DMA Read Cycle Timing Diagram.....	11-13
11-14	Intel-Compatible Typical DMA Write Cycle Timing Diagram.....	11-14
11-15	Intel-Compatible Typical Bus Arbitration Timing Diagram.....	11-14
11-16	Intel-Compatible Typical Write Cycle with $\overline{\text{HALT}}$ Timing Diagram.....	11-15
11-17	Intel-Compatible Typical Read Cycle with $\overline{\text{BERR}}$ Timing Diagram.....	11-16
11-18	Intel-Compatible Typical Read Cycle with $\overline{\text{RETRY}}$ Timing Diagram.....	11-17
12-1	Motorola Host Processor Read Cycle Timing Diagram.....	12-6
12-2	Motorola Host Processor Write Cycle Timing Diagram.....	12-7
12-3	Motorola Interrupt Acknowledge Cycle Timing Diagram.....	12-8
12-4	Motorola Bus Arbitration Timing Diagram.....	12-9
12-5	Motorola DMA Read Cycle and Slow Read Cycle Timing Diagram....	12-10
12-6	Motorola DMA Write Cycle and Slow Write Cycle Timing Diagram...	12-11
12-7	Motorola Late Synchronous Exception $\overline{\text{DTACK}}$ Active Timing Diagram.....	12-12
12-8	Motorola Early Synchronous Exception $\overline{\text{DTACK}}$ Not Active Timing Diagram.....	12-13
12-9	Motorola Hardware RESET Timing Diagram.....	12-14
12-10	Motorola Read Cycle with $\overline{\text{RETRY}}$ Timing Diagram.....	12-15
12-11	Serial Data and Serial Clocks Timing Diagram.....	12-16
12-12	Intel-Compatible Host Processor Read Cycle Timing Diagram.....	12-20

## LIST OF ILLUSTRATIONS (Concluded)

Figure Number	Title	Page Number
12-13	Intel-Compatible Host Processor Write Cycle Timing Diagram.....	12-21
12-14	Intel-Compatible Interrupt Acknowledge Cycle Timing Diagram .....	12-22
12-15	Intel-Compatible Bus Arbitration Timing Diagram .....	12-23
12-16	Intel-Compatible DMA Read Cycle and Slow Read Cycle Timing Diagram.....	12-24
12-17	Intel-Compatible DMA Write Cycle and Slow Write Cycle Timing Diagram.....	12-25
12-18	Intel-Compatible Late Synchronous Exception READY Active Timing Diagram.....	12-26
12-19	Intel-Compatible Early Synchronous Exception READY Not Active Timing Diagram.....	12-27
12-20	Intel-Compatible Hardware RESET Timing Diagram.....	12-28
12-21	Intel-Compatible Read Cycle with $\overline{\text{RETRY}}$ Timing Diagram .....	12-29
12-22	Serial Data and Serial Clocks Timing Diagram.....	12-30





## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
1-1	Guide to Memory Structures .....	1-9
2-1	Examples of Error Mask Handling.....	2-15
2-2	Receive Frame Parameter Summary .....	2-30
3-1	Motorola Bus Selection of Directly Accessible Registers .....	3-2
3-2	Intel-Compatible Bus Selection of Directly Accessible Registers .....	3-3
4-1	MLAPD Command Definitions .....	4-23
4-2	MLAPD Command Summary.....	4-24
8-1	List of Link States .....	8-1
10-1	TxD Three-State Logic Control.....	10-4
10-2	Motorola Data Strobe Control of Data Bus.....	10-6
10-3	Motorola Signal Summary.....	10-8
10-4	Intel-Compatible Byte Addressing.....	10-10
10-5	Intel-Compatible Signal Summary .....	10-13



## SECTION 1 INTRODUCTION

The Motorola MC68606 Multi-Link LAPD (MLAPD) protocol controller is an integrated circuit implementing the link-access procedure (LAPD) protocol. LAPD is the proposed protocol for use at the link layer (ISO-Layer 2) for both signaling and data transfer in Integrated Services Digital Network (ISDN) configurations. The LAPD protocol is specified in International Telegraph and Telephone Consultative Committee (CCITT) Recommendation Q.920/Q.921.

The MLAPD device simplifies interfacing a microcomputer system to a packet network by providing the link-layer services of sequencing, error control, flow control, and multiplexing of logical links. Sequencing refers to numbering the frames to ensure arrival in the order of transmission. Error control ensures that frames arrive without errors; frames are retransmitted when errors are detected. Flow control ensures that the sender does not overwhelm the receiver with data. Multiplexing allows multiple logical-link connections to share a single physical circuit.

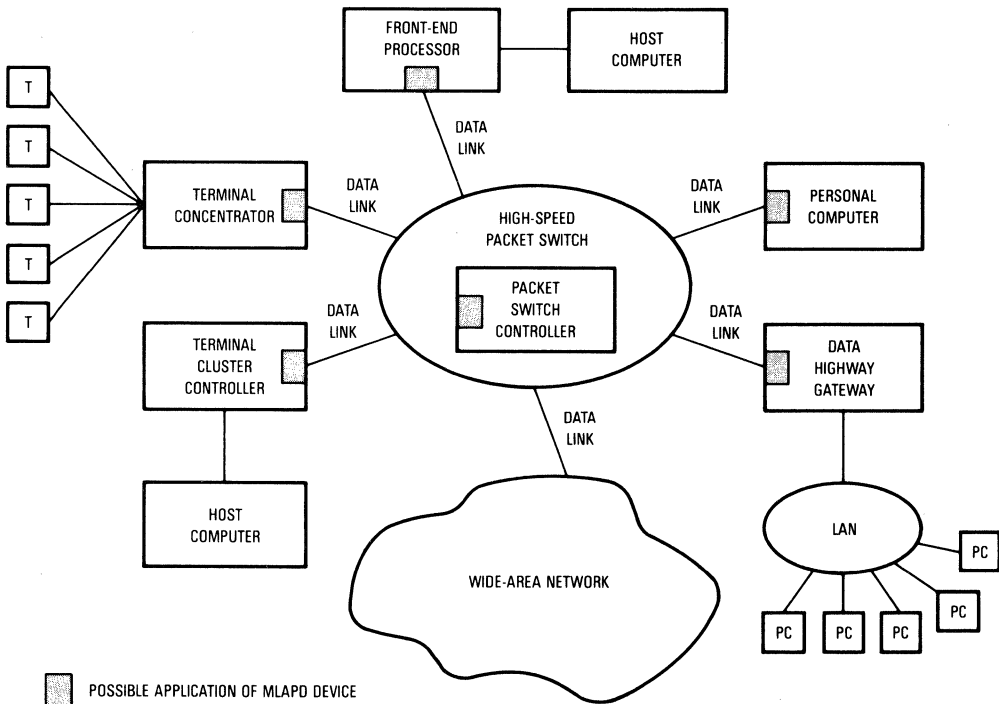
Current product implementations of this link-level protocol are accomplished with firmware. Firmware implementation significantly loads the local processor and presents limitations to both the maximum potential throughput of data and to the number of logical links which may be supported by such packet data interfaces. Figure 1-1 is a generic view of where the MLAPD device can be used to interconnect a variety of data endpoints in a high-speed packet switch network. The data links illustrated could differ functionally and provide data rates up to 2.048 Mbps.

The MLAPD functions as an intelligent peripheral device to a central processing unit (CPU) in a microcomputer system. An on-chip direct memory access (DMA) controller transfers data packets to and from a buffer memory with minimal CPU assistance. A microcoded buffer-management scheme queues packets during transmission and reception. All link management duties are handled by the MLAPD device to maximize the bandwidth available for CPU operation and to increase the throughput for packet data transfer. This VLSI implementation provides a cost-effective solution, while encouraging a universal implementation of the LAPD protocol.

### 1.1 FEATURE OVERVIEW

Key features of the MLAPD include:

- Full implementation of CCITT Recommendation Q.920/Q.921 LAPD with independent generation of commands and responses for each logical link.



**Figure 1-1. Data Switching of the Future**

- Control of up to 8192 logical links using a memory-based architecture, wherein the protocol controller and the supervising microprocessor communicate through shared memory.
- Reliable, interleaved data transfers for multiple logical links with the following protocol actions:
  - HDLC framing with zero-bit insertion/deletion for a serial bit stream; or optional parallel assist mode where zero insertion/deletion is disabled and frame delineation is provided by external pins to simplify the external logic required to support a parallel interface to the physical level.
  - Error control using a 16-bit cyclical redundancy check (CRC).
  - Flow control to prevent data from accumulating at the receiving end faster than data can be processed.
- Termination of a nonchannelized serial bit stream with an aggregate rate in excess of 2.048 Mbps or optional memory-to-memory operation allowing the MLAPD to act as a LAPD controller independent of the system's physical level characteristics.
- User-defined priority for information (I) frame and exchange identification/unnumbered information (XID/UI) frame transmission.
- Optional reception and transmission of a user-defined nonstandard LAPD unnumbered (U) frame.

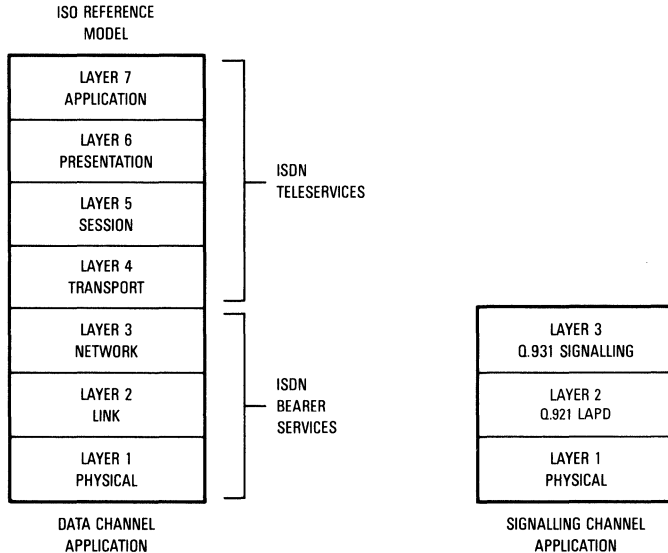
- On-chip, content addressable memory (CAM) provides address translation for up to 16 logical links. When supporting more than 16 logical links, translation is provided via a match table in shared memory.
- Error/statistical counters and maskable interrupts to the Level 3 process.
- Optional nonprotocol mode on a per-logical link basis, which allows the host to receive and transmit frames without application of the LAPD procedures by the MLAPD.
- Promiscuous receive mode in which the MLAPD receives all frames from the line and transfers the entire frame to memory:
  - A received frame may be split between multiple memory buffers.
  - A filtering mechanism allows the user to selectively receive frames based on the first 32-bits of each frame.
- System interface tailored for different microprocessor system implementations:
  - Motorola M68000 and Intel iAPX86 Family bus interface options.
  - 8- and 16-bit data bus support.
  - Direct addressing of 16 Mbytes of system memory.
- Available in 12.5 MHz and 16.67 MHz system clock versions.
- 84-lead pin grid array (PGA) package and plastic, leadless chip carrier (PLCC) J-lead surface mount package.
- 1.5 micron HCMOS technology.

## 1.2 ISDN COMMUNICATION MODEL

A review of the protocol conventions of the Open Systems Interconnect (OSI) model and of the proposed Integrated Services Digital Network (ISDN) model is pertinent to show where the LAPD protocol fits into both the OSI and ISDN models (see Figure 1-2). The LAPD protocol is the recommended data-link-layer protocol for a logical link used for signaling and also for a logical link used for data, or *bearer*, service. The physical level interface may also be the same for both signaling and data applications. The Level 3 interface depends upon the type of services provided. For termination of the link in a signaling case, the CCITT specifies the protocol in Recommendation Q.931. For data transport services, CCITT Recommendation X.25 Data Phase may be used at Level 3. In both applications, the LAPD protocol can be used as a universal link-layer protocol in an ISDN product family.

## 1.3 LAPD FRAME FORMAT

LAPD is specified by CCITT Recommendations Q.920/Q.921 (I.440/I.441). LAPD is a multiplexed derivative of the CCITT Recommendation X.25 LAPB, which is based on the HDLC frame format. The HDLC frame format consists of an address field (two bytes for LAPD), a control field (one or two bytes for LAPD), an optional information field of length *n* bytes (default of 128 or 260 bytes for LAPD), and a 16-bit CRC sequence. Frames are delineated by flag characters (01111110), and zero insertion/deletion is used to ensure data transparency. A minimum of two flags (an opening and closing flag) are transmitted between

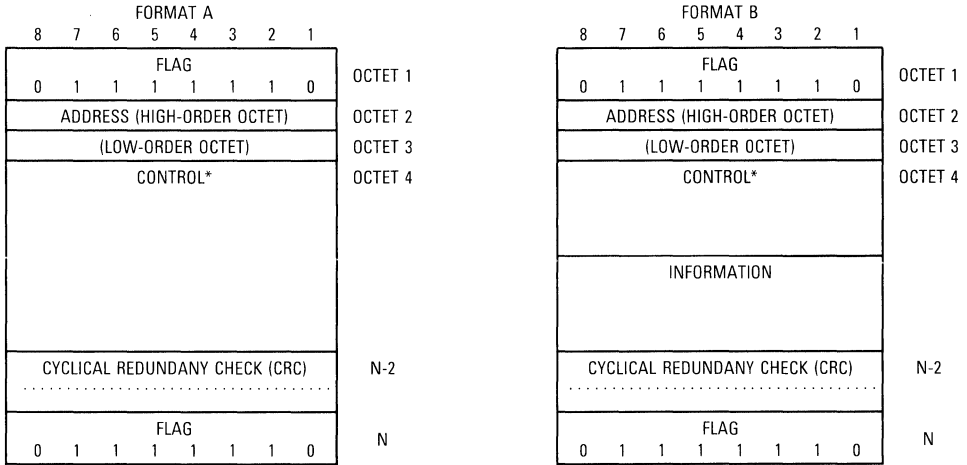


**Figure 1-2. LAPD In Relation to ISDN and ISO Models**

frames. All frames are octet aligned. Octets within a frame are transmitted in ascending numerical order, with the least significant bit transmitted first. An exception to this rule is the transmission of the CRC sequence, where the coefficient of the highest order exponential term is transmitted first. The LAPD formats for frames containing and not containing an information field are shown in Figure 1-3.

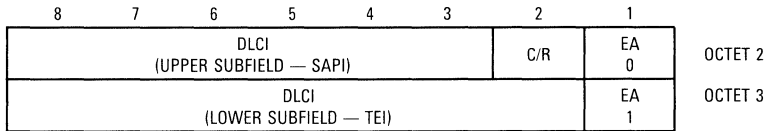
The address field format shown in Figure 1-4 specifies two Data-Link-Connection Identifier (DLCI) subfields, address field extension bits, and a command/response indication bit. The two DLCI subfields, service-access-point identifier (SAPI) and terminal-endpoint identifier (TEI), define the DLCI which is one of 8192 potential addresses, or logical links, that may be active on a single physical channel. The address field extension bit allows the address to span multiple octets by indicating which octet is the final address octet. The address field extension bit in the high order address byte is always set to zero and in the low order address byte is always set to one. The command/response bit identifies a frame as either a command or response. The endpoint designated as the user side sends commands with the C/R bit set to zero and responds with the C/R bit set to one. The endpoint designated as the network side sends commands with the C/R bit set to one and responds with the C/R bit set to zero.

The control field defines the type of command or response frame. The three control field formats, supervisory (S), unnumbered (U) and information (I), are shown in Figure 1-5. The command and response frames defined for these three frame types are shown in Figure 1-6. S frames are used to perform data link supervisory control functions, such as I frame acknowledgement, I frame retransmission requests, and flow control. U frames provide



\* Unacknowledged Operation: One octet  
 Multiple Frame Operation: Two octets for frames with sequence numbers  
 (Modulo 128) One octet for frames without sequence numbers

Figure 1-3. LAPD Frame Formats



C/R = Command/response field bit  
 EA = Address field extension bit  
 DLCI = Data-Link-Connection Identifier  
 SAPI = Service-Access Point Identifier  
 TEI = Terminal-Endpoint Identifier

Figure 1-4. Address Field Format

CONTROL FIELD BITS (MODULO 128)	8	7	6	5	4	3	2	1		
I FORMAT	N(S)								0	OCTET 4
	N(R)								P	OCTET 5
S FORMAT	X	X	X	X	S	S	0	1	OCTET 4	
	N(R)								P/F	OCTET 5
U FORMAT	M	M	M	P/F	M	M	1	1	OCTET 4	

N(S) Send sequence number  
 N(R) Receive sequence number  
 S Supervisory function bit  
 M Modifier function bit  
 P Poll bit  
 P/F Poll bit when issued as a command  
 Final bit when issued as a response  
 X Reserved and set to zero

Figure 1-5. Control Field Formats

Format	Commands	Responses	Encoding								
			8	7	6	5	4	3	2	1	
Information Transfer	I		N(S)							0	
			N(R)							P	
Supervisory	RR	RR	0	0	0	0	0	0	0	1	
			N(R)							P/F	
	RNR	RNR	0	0	0	0	0	1	0	1	
			N(R)							P/F	
	REJ	REJ	0	0	0	0	1	0	0	1	
			N(R)							P/F	
Unnumbered	SABME		0	1	1	P	1	1	1	1	
		DM	0	0	0	F	1	1	1	1	
	UI		0	0	0	P	0	0	1	1	
	DISC		0	1	0	P	0	0	1	1	
		UA		0	1	1	F	0	0	1	1
		FRMR		1	0	0	F	0	1	1	1
	XID	XID		1	0	1	P/F	1	1	1	1

DISC	Disconnect
DM	Disconnected mode
F	Final bit
FRMR	Frame reject
I	Information
N(R)	Receive sequence number
N(S)	Send sequence number
P	Poll bit
P/F	Poll bit when issued as a command/ Final bit when issued as a response
REJ	Reject
RNR	Receive-not-ready
RR	Receive ready
SABME	Set asynchronous balanced mode extended
UA	Unnumbered acknowledgement
UI	Unnumbered information
XID	Exchange identification

**Figure 1-6. Command and Response Frames**

additional data link control functions and unacknowledged information transfer. I frames transfer sequentially numbered frames containing information fields provided by Level 3.

The LAPD procedures define how the S, U, and I frames are used for peer-to-peer communication between data link entities. The details of the MLAPD implementation of these procedures is given in **SECTION 8 MLAPD IMPLEMENTATION OF LAPD**.

## 1.4 SYSTEM ENVIRONMENT

The MLAPD protocol controller provides simultaneous control of a maximum of 8192 logical links, while under the overall supervision of a microprocessor. The communication between



the microprocessor, hereafter referred to as the host processor or CPU, and the protocol controller is established through command and data block structures stored in shared memory. The Global Configuration Block (GCB), Logical-Link Tables (LLTs), Transmit and Receive Queues, Receive Pools, and Interrupt Queue are command and data blocks highlighted in Figure 1-7. This figure provides a simplistic overview of the MLAPD's shared memory structure.

### 1.4.1 Physical Link Interface

The MLAPD serial interface provides HDLC-type decoding/encoding for the data stream entering/exiting the full-duplex serial interface. The three input signals from the physical interface are the transmit clock, receive clock, and receive data. The transmit data signal is an output to the physical interface. Nonreturn-to-zero (NRZ) data encoding/decoding is implemented. Two modem control signals, request-to-send (RTS) and clear-to-send (CTS), are also available. Electrical signal levels are transistor-transistor logic (TTL) compatible.

In addition, the MLAPD may be optionally configured to interface to a physical level which does not implement a serial HDLC-framed bit stream. In this mode, the MLAPD transmitter does not provide zero insertion, and the MLAPD receiver does not provide zero deletion

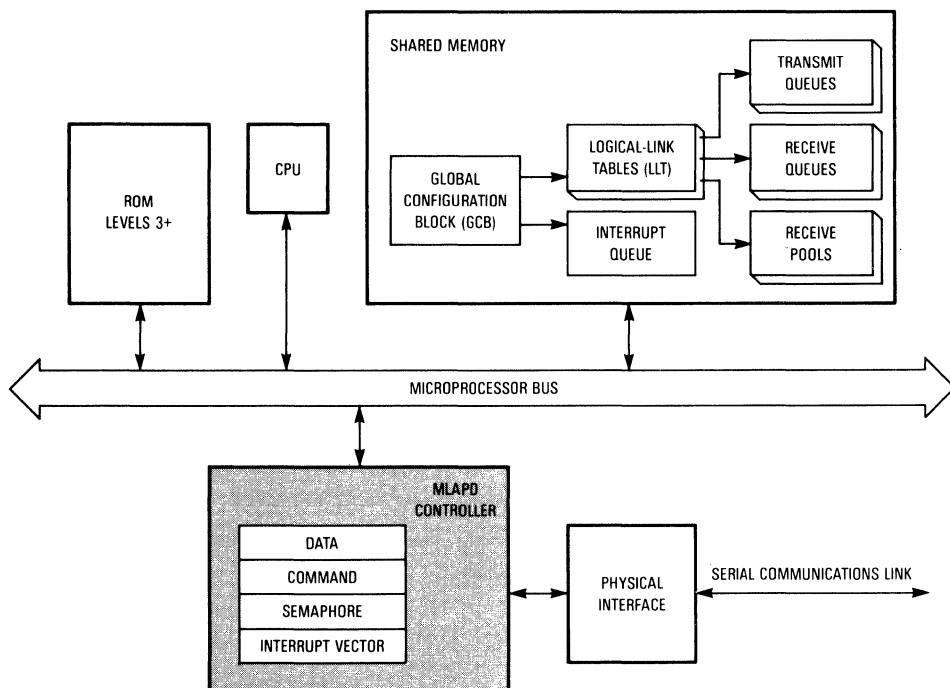


Figure 1-7. MLAPD Protocol Controller Application Environment

for data transparency. Additionally, the MLAPD does not delineate frames with flag sequences. Instead, the RTS and CTS signals function as transmit start (TSTART) and receive start (RSTART), respectively, to delineate a frame for the transmit and receive data lines. In particular, this mode can simplify the external logic required to support a physical level that implements a parallel interface, such as a backplane in a network switch or host computer.

### 1.4.2 Microprocessor System Bus Interface

The host CPU can be any 8- or 16-bit microprocessor that supports multibus master capability, compatible with either the Motorola M68000 Family or the Intel iAPX86 Family bus interface definition. The important differences between the two families are the means of providing read, write, and data-byte selection signals. The desired bus operation mode of the MLAPD is selected at powerup.

The MLAPD has a 24-bit address bus to directly support a 16-Mbyte memory address space. A separate 16-bit data bus is provided, which may be configured for 8-bit operation. The MLAPD handles byte ordering in an appropriate fashion for the selected microprocessor family option.

The format for memory usage has 16-bit and 32-bit values stored at even boundaries enabling 16-bit microprocessors to operate efficiently in shared memory. To ensure compatibility with both 8-bit and 16-bit microprocessors, the configuration, command, and status information for logical-link control are built as linked table structures on a byte-wide basis.

### 1.4.3 Shared-Memory Control Components

Referring to Figure 1-7, which illustrates the MLAPD protocol controller's application environment, the shared memory consists of several key control blocks. The control blocks are functionally partitioned into the following structures:

- Global Configuration Block (GCB)
- Logical-Link Tables (LLTs)
- Receive and Transmit Queues
- Receive Pool Pointers Table
- Interrupt Queue
- Match Table
- Logical-Link Identification (LLID) number to LLT Table (LLID-LLT)
- Timer Table
- Level 2 Queue

Table 1-1 provides a summary of the functions performed by these blocks.

During system initialization, the host processor loads protocol and operational parameters into the shared-memory control structures and then stores pointers to the various memory

**Table 1-1. Guide to Memory Structures**

<b>Block Type</b>	<b>Purpose</b>
Global Configuration Block (GCB)	Contains pointers to other memory tables. Contains global information for all logical links. Contains user system and link options.
Logical-Link Tables (LLTs)	Contains specific link control parameters for the given link. Also contains pointers to the receive/transmit queues. Contains working protocol status for the link.
Receive and Transmit Queues	Contains linked list of receive and transmit frame descriptors for information (i), unnumbered information (Ui), and exchange identification (XiD) frames.
Receive Pools	Linked list of available receive frame descriptors with associated data buffers.
Interrupt Queue	Contains time-sequential list of interrupt status for each interrupt event.
Match Table	Contains translation from Data-Link-Connection Identifier (DLCI) to Logical-Link Identification (LLID) number for incoming frames (expanded operation mode).
LLID-LLT Tables	Translates LLID into the address of the corresponding LLT.
Level 2 Transmit Queue	Contains Level 2 supervisory (S) and unnumbered (U) frames generated by MLAPD.
Timer Table	Implements T200/T203 for all logical links that are in multiple-frame established mode of operation.

tables into the GCB. Next, the host programs the MLAPD protocol controller with the address of the GCB and commands the MLAPD to initialize itself by loading its internal registers from the shared memory tables. From this point on, the host and the MLAPD communicate primarily via software flags and an Interrupt Queue in shared memory.

## 1.5 OVERVIEW OF MLAPD FUNCTIONAL OPERATION

Before describing the shared memory structures, internal registers, command set, and bus operation, a high-level summary of the command structure, the Interrupt Queue structure, and the MLAPD operational sequences may prove helpful.

### 1.5.1 MLAPD Internal States and Command Structure Overview

The MLAPD functions in one of three internal states: off-line, on-line, or bus/address error. In the off-line state, the MLAPD will only execute host commands. In the on-line state, the MLAPD will execute host commands, receive frames, and transmit frames. In the bus/address error state, the MLAPD will only accept an OFF-LINE or RESET command from the host.

Transitions between these three states occur as the result of host commands and system error conditions. After reset, the MLAPD is in the off-line state. INIT and ON-LINE commands transfer the MLAPD to the on-line state. The OFF-LINE command transfers the MLAPD to the off-line state from either the on-line or the bus/address error state. The bus/address

error state is entered when the  $\overline{\text{BERR}}$  signal is asserted during an MLAPD bus master cycle or when  $\overline{\text{CS}}$  or  $\overline{\text{IACK}}$  ( $\overline{\text{INTA}}$ ) signals are asserted during an MLAPD bus master cycle.

The host issues a command to the MLAPD by first writing the command arguments (if any) into the command argument fields in the GCB in shared memory. Second, the host performs a write operation to enter the command into the on-chip command register. These commands belong to one of the following five categories:

- 1) Initialization
- 2) Test/Diagnostics
- 3) Host/MLAPD Interface
- 4) Protocol
- 5) Protocol Extension

Upon receipt of a command, the MLAPD sets its on-chip semaphore register (SR) to the value 'FE' hex. After either command completion or command acceptance, depending on the specific command, the MLAPD sets the value of the SR to 'FF' hex. The host must always read the SR before writing a new command. See **3.1.2 Semaphore Register**.

### 1.5.2 Interrupt Structure Overview

An Interrupt Queue is located in shared memory to support the speed requirement of reporting many interrupting events occurring in rapid succession across the various active logical links. Each entry in this Interrupt Queue consists of a Logical-Link Identification (LLID) number, if applicable, and a cause indication. Each of the potential interrupt sources may be individually masked by the host to prevent the reporting of a specific event. The Interrupt Queue is described in detail; refer to **2.8 INTERRUPT QUEUE**.

The MLAPD supports both systems using the interrupt-driven method and systems using the polled method for interrupt handling. After the MLAPD has written an entry into the Interrupt Queue, a hardware interrupt request may be asserted. This interrupt feature is enabled or disabled by the host. If interrupt-driven operation (interrupts are allowed in the system) is desired, the interrupt\_vector register (IVR) must be programmed by the host during the initialization procedure, or it will assume the default value '0F' hex. If polled operation is desired, the host programs the MLAPD for this method at initialization.

The IVR is on-chip and allows the MLAPD to return an indication of the interrupt cause as part of an interrupt acknowledge cycle on the system bus. The two possible vectors reported across the bus are: normal interrupt and severe interrupt.

Interrupt information is written into the Interrupt Queue to fully identify the specific cause of the interrupt in either a polled or interrupt-driven system. Interrupts are the principle mechanism used to report protocol events and errors (including exceeded error thresholds). Since the Interrupt Queue is circular and of limited size, the host is responsible for reading the interrupt information and maintaining free entries to record new interrupt events.

### 1.5.3 Initialization Overview

As a part of initialization, the host must:

- Issue a RESET command to the MLAPD.
- Issue a SET\_BUS\_WIDTH\_16,(8) command to the MLAPD, which specifies a 16- or 8-bit data bus configuration.
- Load the IVR in the MLAPD device.
- Load the address of the GCB into the data register in the MLAPD device.
- Prepare the GCB. The host specifies the addresses of the various memory tables in the GCB and programs several global parameters required to process the LAPD protocol for all links.
- Clear the following tables: Match Table, Interrupt Queue, and Timer Table.
- Prepare the LLT(s) with the local parameters for the link(s) in service.
- Construct receive pool(s) of frame descriptors.
- Construct the necessary entry(s) in the LLID-LLT Table.
- Issue the INIT command to the MLAPD.

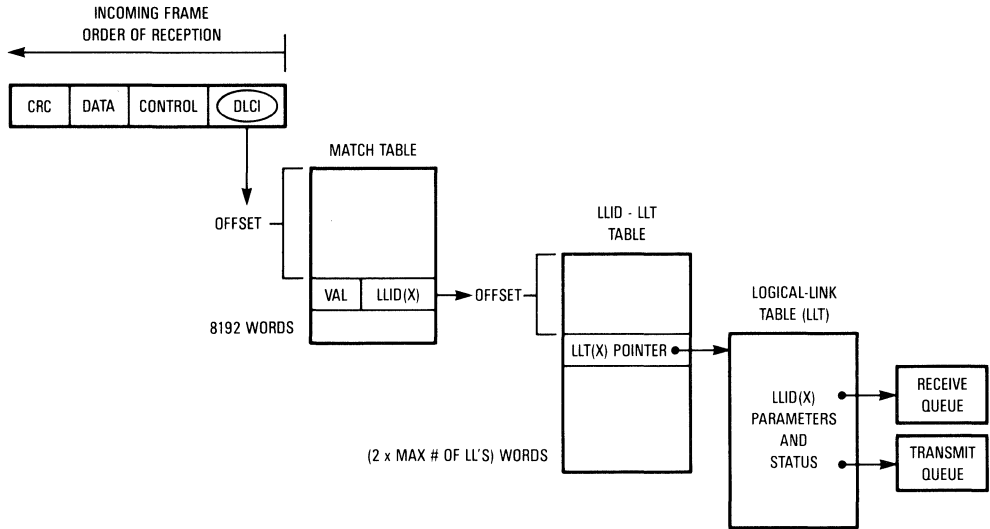
The MLAPD now enters the on-line state and begins monitoring the receive data line.

### 1.5.4 Frame Reception Overview

Frame reception for a specific logical link is enabled by performing the initialization procedure followed by a DLCI assignment procedure. (The DLCI, which is contained in the address field of LAPD frames, is formed by the concatenation of the 6-bit SAPI and the 7-bit TEI.) This logical link then enters a protocol-defined state in which it can receive UI frames, using data buffers from its assigned receive pool. After the exchange of set asynchronous balanced mode extended (SABME) and unnumbered acknowledgement (UA) frames, the link setup procedure is complete, and the data connection is established. The MLAPD device is now ready to receive numbered I frames containing the assigned DLCI, provided receive buffers are available.

When the expanded-system operation mode is selected by the host, the DLCI field in an incoming frame is used by the MLAPD as an index into an external Match Table in shared memory to determine whether the DLCI has been assigned to an active logical link by Level 3. The Match Table also contains the LLID number associated with each active DLCI. (Level 3 defines a 13-bit LLID number corresponding to each active DLCI. Translation to an LLID, which is only of local significance to the link-level process, serves to reduce the external memory requirements.) If the DLCI is marked as invalid, then the incoming DLCI has not been assigned to a logical link, and the frame is ignored. The receive process for expanded-system operation mode is shown in Figure 1-8.

If the DLCI is marked valid, the MLAPD uses the LLID associated with the DLCI as an offset into a lookup table, which is the LLID-LLT Table. This link's LLT address is stored in the



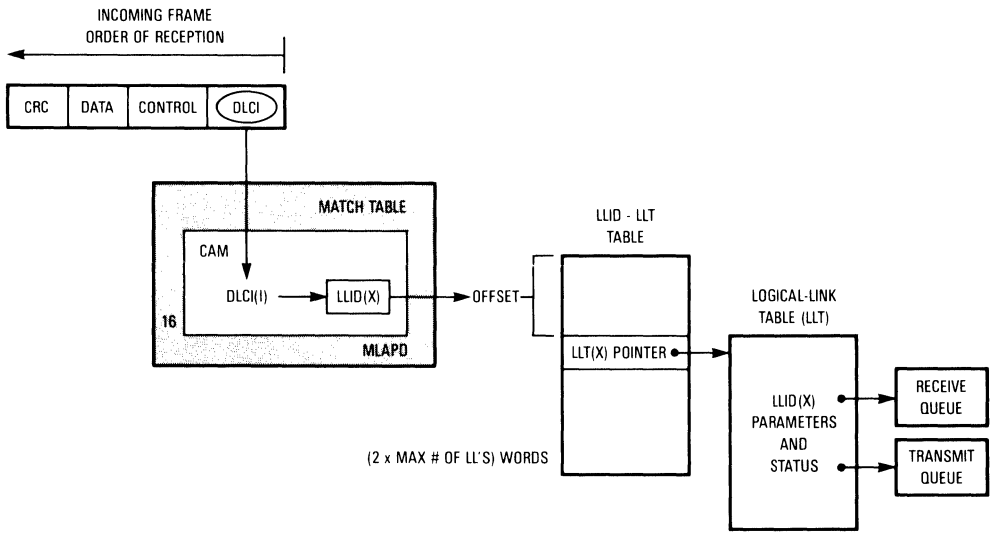
**Figure 1-8. Receive Process for Expanded Operation Mode**

location indexed by this offset. The MLAPD accesses the link's LLT to obtain the specific protocol parameters for this link, which are required for subsequent protocol processing of the received frame. In the case of an information bearing frame, the MLAPD uses the receive\_pool\_number entry in the LLT as an offset into the Receive Pool Pointers Table to locate the first available receive frame descriptor for this logical link. (Sixteen pool pointers are stored on-chip to reduce the number of memory accesses for the receive operation.) The MLAPD stores the data field in the incoming frame in this buffer for processing by higher level software.

When the application is limited to supporting 16 logical links or less, the host selects on-chip system operation mode. In the on-chip operation mode, the DLCI match operation is performed by an internal CAM circuit, and subsequent indexing performed in external memory is the same as for the expanded-system operation mode. The internal CAM reduces the external memory requirements for endpoints such as personal computers, since the 8K-word Match Table is not required. Figure 1-9 maps the receive process in the on-chip operation mode.

### 1.5.5 Frame Transmission Overview

Frame transmission for a specific logical link is enabled by performing the initialization procedure followed by a DLCI assignment procedure. The logical link is then in a protocol-defined state in which it can transmit UI frames. The host must initiate a link setup procedure before numbered information frames may be sent.



**Figure 1-9. Receive Process for On-Chip Operation Mode**

When data is ready for transmission on a given logical link, the host fragments (divides) the data into one or more data buffers, where each buffer will be transmitted as an I frame. The host then arranges the frame descriptors (which point to the actual data buffers) into a linked list and issues a data request to the MLAPD. In response, the MLAPD device links these frame descriptors into one of four transmit I frame queues. The host assigns each logical link to an I frame queue by programming a parameter in the link's LLT. By providing four I frame queues, the MLAPD allows the host to prioritize transmit frame servicing.

In comparison, the transmission process is more complex than the reception process, in that the MLAPD device must wait for acknowledgment from the Level 2 peer before declaring the data request operation to be complete. After transmission, the MLAPD maintains control of a *transmitted-awaiting-acknowledgement* frame in case retransmission is required. Once the frame has been acknowledged, the MLAPD sets the confirmation bit in the status word of its I frame descriptor.

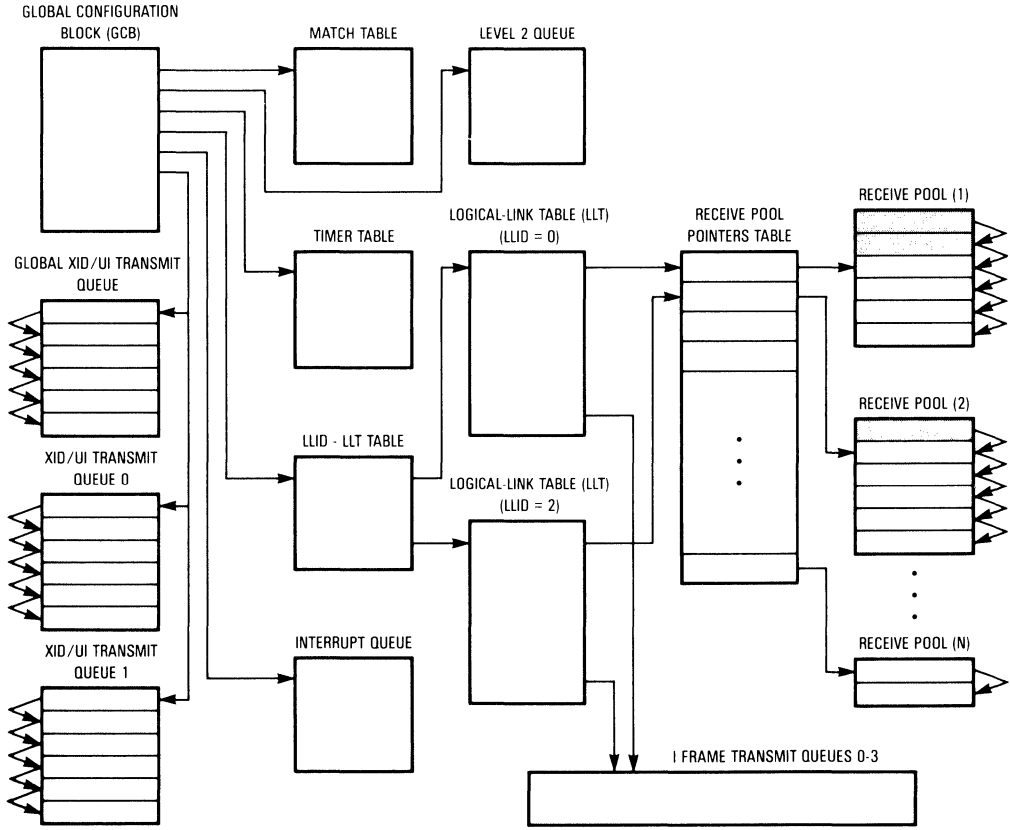
After all frames have been transmitted and acknowledged, the MLAPD writes a DL\_data\_confirmation for the specific LLID into the Interrupt Queue in shared memory. Additionally, if the interrupt option is enabled, the MLAPD activates the hardware interrupt signal (if it is not already asserted). The host is thus notified that confirmed frames on the transmit queue can be removed and reused. The host can also remove confirmed frames before receiving the confirmation interrupt by checking the confirmation bit in each frame descriptor.





## SECTION 2 MEMORY STRUCTURES

The MLAPD uses several memory structures to communicate with the host processor. These memory structures include the Global Configuration Block (GCB), Logical-Link Tables (LLTs), Receive and Transmit Queues, and the Interrupt Queue. In addition to these structures, four memory tables are required for the implementation of the LAPD procedure. These tables include the Match Table, Logical-Link Identification to Logical-Link Table (LLID-LLT) Table, Receive Pool Pointers Table, and Timer Table. The Level 2 Queue is a private area in shared memory accessed only by the MLAPD. Figure 2-1 shows the interrelationship of these tables in memory.



**Figure 2-1. Shared Memory Tables**

All MLAPD pointers to memory tables are 32 bits in length, even though the MLAPD only provides 24 bits of address during bus master cycles. In a 16-bit system, all memory tables and data structures must begin on an even-byte boundary, with the exception of transmit data buffers. The MLAPD does not check for an *odd pointer* and does not generate an address error for an odd-word boundary. For an 8-bit data bus, an odd pointer is proper. For a 16-bit bus, the MLAPD zeros the least significant address bit and therefore always presents an even-word address to the bus. The system designer must ensure that the MLAPD is not supplied with an odd pointer in a 16-bit configuration.

## 2.1 GLOBAL CONFIGURATION BLOCK

The GCB format is shown on the following pages (refer to Figures 2-2 through 2-5). The GCB is written by the host, is read by the MLAPD, and is divided into four areas: constants, reloadable variables, statistics threshold variables, and command arguments. The host has

HEX DISPLACEMENT	15	8	7	0
0	OPTION BITS 1			
2	OPTION BITS 2			
4	-----	MATCH TABLE POINTER		-----
6				
8	-----	L2 QUEUE POINTER		-----
A				
C	L2 QUEUE LENGTH			
E	-----	TIMERS TABLE POINTER		-----
10				
12	-----	LLID-LLT TABLE POINTER		-----
14				
16	-----	INTERRUPT QUEUE POINTER		-----
18				
1A	INTERRUPT QUEUE LENGTH			
1C	-----	POOL TABLE POINTER		-----
1E				
20	RESERVED			
22	RESERVED			
24	RESERVED			
26	RESERVED			
28	RESERVED			
2A	RESERVED			
2C	RESERVED			
2E	RESERVED			

Figure 2-2. GCB Format — Constants

HEX DISPLACEMENT	15	8	7	0
30			PAD TIME SELECT	
32	NONSTANDARD CONTROL			
34			T203 VALUE	
36	I FRAME RETRANSMIT THRESHOLD		N200 VALUE	
38	REJ TRANSMIT THRESHOLD		RNR TRANSMIT THRESHOLD	
3A	REJ RECEIVE THRESHOLD		RNR RECEIVE THRESHOLD	
3C	CTS TIMEOUT THRESHOLD			
3E	SCAN LENGTH I QUEUE 0			
40	SCAN LENGTH I QUEUE 1			
42	SCAN LENGTH I QUEUE 2			
44	SCAN LENGTH I QUEUE 3			
46	SCAN LENGTH XID/UI QUEUE 0			
48	SCAN LENGTH XID/UI QUEUE 1			
4A	-----	INTERRUPT MASK		-----
4C				
4E	PROTOCOL ERROR MASK			
50	NONPROTOCOL ERROR MASK			
52				
54	-----	FILTER MASK		-----
56				
58	-----	FILTER MATCH		-----
5A	RESERVED			
5C	RESERVED			
5E	RESERVED			

**Figure 2-3. GCB Format — Reloadable Variables**

the option to reserve additional words beyond the end of the GCB for use by higher level software. The additional entries required by the higher level software to interface with the MLAPD are specified in the user area and may be considered as part of the GCB. However, these entries may actually reside anywhere in memory as the MLAPD does not access them. (Refer to Figure 2-6.)

### 2.1.1 Constants Area

The length of this area is 48 bytes from '0' hex to '2F' hex. The MLAPD reads all constant entries at initialization (INIT command) and stores them internally. The MLAPD has an internal register corresponding to each entry with the same name. If the user wishes to change any of these constants, an INIT command must be issued directing the MLAPD to reload the entire GCB.

HEX DISPLACEMENT	15	8 7	0
60	Rx FIFO OVERRUN THRESHOLD		
62	Tx FIFO UNDERRUN THRESHOLD		
64	INACTIVE DLCI THRESHOLD		
66	INVALID ADDRESS THRESHOLD		
68	DISCARDED FRAME THRESHOLD		
6A	SHORT FRAME THRESHOLD		
6C	CRC ERROR THRESHOLD		
6E	ABORT/NON-OCTET THRESHOLD		
70	RESERVED		
72	RESERVED		
74	RESERVED		
76	RESERVED		
78	RESERVED		
7A	RESERVED		
7C	RESERVED		
7E	RESERVED		

**Figure 2-4. GCB Format — Statistics Threshold**

HEX DISPLACEMENT	15	8 7	0
80	COMMAND ARGUMENT 1		
82	-----	COMMAND ARGUMENT 2	-----
84			
86	-----	COMMAND ARGUMENT 3	-----
88			
8A	RESERVED		
8C	RESERVED		
8E	RESERVED		

**Figure 2-5. GCB Format — Command Arguments**

HEX DISPLACEMENT	15	8	7	0
90	USER INTERRUPT QUEUE READ POINTER			
92				
94	USER GLOBAL XID/UI Tx NEXT CONFIRM POINTER			
96				
98	USER GLOBAL XID/UI Tx LAST QUEUED POINTER			
9A				
9C	USER XID/UI_0 Tx NEXT CONFIRM POINTER			
9E				
A0	USER XID/UI_0 Tx LAST QUEUED POINTER			
A2				
A4	USER XID/UI_1 Tx NEXT CONFIRMED POINTER			
A6				
A8	USER XID/UI_1 Tx LAST QUEUED POINTER			
AA				

Figure 2-6. GCB — User Area

2.1.1.1 OPTION BITS 1. The option\_bits\_1 word defines user programmable options as follows:

15	14	13	12	11	10	9	8
0	0	SERIAL ECHO SELECT	RTS SELECT	ZERO DELETE SELECT	INTERNAL SELECT	BURST CONTROL	ZERO INSERT SELECT
7	6	5	4	3	2	1	0
0	0	MEMORY TO MEMORY	PROMISCUOUS RECEIVE SELECT	FILTER SELECT	POLLING SELECT	Tx FRMR SELECT	SYSTEM MODE

Bits 15 and 14 are not used.  
 These bits must be set to zero by the host.

Serial\_echo\_select (bit 13)

- 1 RxD and TxD pins are connected internally, causing any data received from the network on RxD to be retransmitted to the network on TxD. (The MLAPD receiver and transmitter are not connected to the RxD and TxD.)
- 0 RxD and TxD are not connected internally to echo data back to the network.

**RTS-select (bit 12)**

This bit is meaningful only when the zero-insertion-select bit (bit 8) in the option\_bits-1 entry is set to zero. Otherwise, RTS-select is not used and must be set to zero.

- 1  $\overline{\text{RTS}}$  is asserted the first time a DL\_DATA\_REQUEST, GLOBAL\_XID/UI\_REQUEST, XID/UI\_QUEUE\_0\_REQUEST, or XID/UI\_QUEUE\_1\_REQUEST command is issued following INIT.  $\overline{\text{RTS}}$  remains asserted until a RESET or INIT command is issued.
- 0  $\overline{\text{RTS}}$  is asserted only when there is a pending transmit frame. When no transmit frames are queued,  $\overline{\text{RTS}}$  is negated.

**Zero-deletion-select (bit 11)**

- 1 No zero deletion is performed on the receive bit stream. The  $\overline{\text{CTS}}$  signal now operates as the  $\overline{\text{RSTART}}$  input for the MLAPD receiver. See **10.1.1.2 CLEAR-TO-SEND ( $\overline{\text{CTS}}$ ) OR RECEIVE START ( $\overline{\text{RSTART}}$ )**.
- 0 Zero deletion is performed on the receive bit stream.

**Internal-loopback-select (bit 10)**

- 1 Internal loopback is enabled. The MLAPD transmitter is connected to the MLAPD receiver. The transmitter is not connected to TxD, and the receiver is not connected to RxD.  $\overline{\text{RTS}}$  is not asserted. TxCLK is used to synchronize the data.
- 0 Internal loopback is disabled. The MLAPD transmitter is not connected to the MLAPD receiver.

**Burst-control (bit 9)**

- 1 Each MLAPD DMA burst is limited to eight successive memory cycles.
- 0 Each MLAPD DMA burst is unlimited.

**Zero-insertion-select (bit 8)**

- 1 No zero insertion is performed on the transmit bit stream. The  $\overline{\text{RTS}}$  output now operates as the  $\overline{\text{TSTART}}$  output signal for the MLAPD transmitter. See **10.1.1.1 REQUEST-TO-SEND ( $\overline{\text{RTS}}$ ) OR TRANSMIT START ( $\overline{\text{TSTART}}$ )**.
- 0 Zero insertion is performed on the transmit bit stream.

Bits 7 and 6 are not used.

These bits must be set to zero.

**Memory-to-memory select (bit 5)**

- 1 Memory-to-memory mode is enabled. In this mode the MLAPD performs the LAPD procedures for memory-resident receive and transmit frames. See **9.6 MEMORY-TO-MEMORY OPERATION**.
- 0 Memory-to-memory mode is disabled.

#### Promiscuous\_receive\_select (bit 4)

- 1 Promiscuous receive is enabled. In this mode the MLAPD receives all incoming frames. No address recognition is performed. The MLAPD places all receive frames in the receive pool assigned to DLCl=0. When promiscuous receive is enabled, the user must assign the link with DLCl=0 to operate in nonprotocol mode. See **9.2 PROMISCUOUS RECEIVE MODE**. The MLAPD transmitter operation is not affected by this bit. All transmitter functions are still available.
- 0 Promiscuous receive is disabled. In this mode, address matching is performed on the incoming DLCIs, and the MLAPD may perform LAPD processing for the received frames.

#### Filter\_select (bit 3)

This bit is meaningful only when the promiscuous\_receive\_select bit (bit 4) in the option\_bits\_1 entry is set to one. Otherwise, this bit is not used and must be set to zero.

- 1 Filtering is enabled on the incoming frames. See **9.2.2 Filter Mode**.
- 0 Filtering is disabled.

#### Polling\_select (bit 2)

- 1 Polling enabled. Interrupting events are reported via the Interrupt Queue, but  $\overline{IRQ}$  (INTR) is not asserted. In the case of a bus/address error condition, however, the MLAPD will assert  $\overline{IRQ}$  (INTR). This action addresses the possibility that a bus/address error may occur during an access to the Interrupt Queue. When polling mode is selected, the host must periodically scan the Interrupt Queue.
- 0 External interrupt indication enabled. Interrupting events are reported via the Interrupt Queue and  $\overline{IRQ}$  (INTR) is asserted.

#### Tx\_FRMR\_select (bit 1)

- 1 Transmission of a FRMR frame is enabled when a FRMR condition is determined. See **8.11.1 FRAME REJECT MODE**.
- 0 Transmission of a FRMR frame is disabled when a FRMR condition is determined.

#### System\_mode (bit 0)

- 1 On-chip system operation mode enabled. Up to 16 logical links are supported via on-chip CAM. An external Match Table is not supported. This system mode should be selected for a system designed to support 16 or fewer logical links. The on-chip operation mode uses less external memory resources, i.e. the 8K-word Match Table is not required.
- 0 Expanded-system operation mode enabled. Up to 8192 logical links are supported. An external 8K-word Match Table is required. To reduce external memory requirements, the host must always assign the lowest available LLID when activating a logical link.

**2.1.1.2 OPTION BITS 2.** The option\_bits\_2 word defines user programmable options as follows:

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	CCITT/ DMI SELECT	FLIP SELECT	LINK STATISTICS SELECT	RETRANSMIT STATISTICS SELECT	MULTI- BUFFER SELECT

Bits 15 thru 5 are not used.  
These bits must be set to zero.

#### CCITT/DML-select (bit 4)

- 1 The MLAPD implements the LAPD protocol as defined by CCITT Recommendation Q.920/Q.921. See **8.11.2 CCITT/DML MODE**.
- 0 The MLAPD implements the LAPD protocol as defined by DMI. See **8.11.2 CCITT/DML MODE**.

#### Flip-select (bit 3)

- 1 Flipping is enabled. The MLAPD inverts the least significant bit of the DLCI for each receive frame. By altering the frame address, the user can perform a system loop-back (either internal or external) where the user software and the MLAPD operate as though a connection has been established with a remote station. Since the transmit and receive addresses are different, the MLAPD can implement the LAPD procedures for these two logical links.
- 0 Flipping is disabled.

#### Link\_statistics\_select (bit 2)

- 1 Link statistics are enabled. For every link assigned as a LAPD protocol link, the MLAPD maintains four down counters which record the number of transmitted RNR frames, transmitted REJ frames, received RNR frames, and received REJ frames. The user specifies threshold values for each counter via the corresponding GCB entries.
- 0 Link statistics are disabled.

#### Retransmit\_statistics\_select (bit 1)

- 1 Retransmit statistics are enabled. For each I frame, the MLAPD maintains an up counter that counts how many times the I frame is retransmitted. This count is recorded in its transmit frame descriptor.
- 0 Retransmit statistics are disabled.



#### Multibuffer\_select (bit 0)

This bit is meaningful only for links assigned as nonprotocol. This bit does not affect the receive process for protocol links, regardless of its value.

- 1 The MLAPD will use more than one receive memory buffer as necessary to store an incoming frame.
- 0 The MLAPD will only use a single receive memory buffer to store an incoming frame. When the buffer is not large enough to store the entire frame, the frame may or may not be received based upon the nonprotocol\_error\_mask entry.

**2.1.1.3 MATCH TABLE POINTER.** This 32-bit GCB entry is the address of the first word of the Match Table. The match\_table\_pointer is loaded into the match\_table\_pointer register by the MLAPD during execution of the INIT command. The host must zero the Match Table area before presenting the area to the MLAPD.

**2.1.1.4 L2 QUEUE POINTER.** This 32-bit GCB entry is the address of the first word of the Level 2 (L2) Queue in memory. This pointer is loaded into the L2\_queue\_pointer register by the MLAPD during the execution of the INIT command.

**2.1.1.5 L2 QUEUE LENGTH.** The number of words allocated by the host for the Level 2 Queue in external memory is contained in this 16-bit GCB entry. The value of  $\{(L2\_queue\_length - 5 \text{ words}) \times 2\}$  is loaded into the L2\_tail\_displacement register during execution of the INIT command. The maximum value for L2\_queue\_length is  $2^{15} - 1$  words, so the host must set bit 15 to zero in this entry. It is recommended that the host provide a minimum of 128 words for the external Level 2 Queue. Internally, the MLAPD provides 32 words in on-chip system mode or 48 words in expanded-system mode for the Level 2 Queue. The external Level 2 Queue is only used when the internal queue is filled with pending frames to minimize shared memory accesses.

**2.1.1.6 TIMER TABLE POINTER.** This 32-bit GCB entry is the address of the first word of the Timer Table in memory. This pointer is loaded into the timer\_table\_pointer register by the MLAPD during the execution of the INIT command. The host must zero the Timer Table area before presenting the area to the MLAPD. The Timer Table length (in words) must not be less than the maximum LLID number, which may be assigned to any logical link by higher layer software.

**2.1.1.7 LLID-LLT TABLE POINTER.** This 32-bit GCB entry is the address of the first word of the LLID-LLT Table. This pointer is loaded into the LLID-LLT\_table\_pointer register by the MLAPD during execution of the INIT command.

**2.1.1.8 INTERRUPT QUEUE POINTER.** The 32-bit address of the first word of the Interrupt Queue is contained in this GCB entry. The MLAPD loads the interrupt\_queue\_pointer into

the `interrupt_queue_pointer` register during the execution of the INIT command. The host must zero the Interrupt Queue before presenting the area to the MLAPD.

**2.1.1.9 INTERRUPT QUEUE LENGTH.** The number of entries in the Interrupt Queue is contained in this 14-bit GCB entry. Bits 15-14 are not used and must be set to zero. Each entry consists of two words. The size of the queue is defined by the user, based on the interrupt response time of the host and the expected occurrence of interrupting events. The Interrupt Queue may support up to 16K interrupt entries. The value of  $\{\text{interrupt\_queue\_pointer} + (4 \times \text{interrupt\_queue\_length}) - 2 \text{ words}\}$  is loaded into the `interrupt_queue_tail_pointer` register during the execution of the INIT command.

**2.1.1.10 POOL TABLE POINTER.** This 32-bit GCB entry is the address of the Receive Pool Pointers Table in memory. This pointer is loaded into the `pool_table_pointer` register by the MLAPD during execution of the INIT command. Since the MLAPD stores the pointers for receive pools 0 to 15 on-chip, the first `receive_pool_pointer` in the memory table corresponds to receive pool 16. The host must allocate enough memory for the Receive Pool Pointers Table to accommodate pointers for all receive pools that will be assigned to the various logical links by higher level software.

## 2.1.2 Reloadable Variables Area

The length of this area is 48 bytes from '30' hex to '5F' hex. The MLAPD reads these reloadable entries at initialization (INIT command) and stores them internally. The MLAPD has an internal register corresponding to each entry with the same name. The user can change any of these constants while the MLAPD is in the on-line state and then issue a RELOAD command to cause the MLAPD to reload this portion of the GCB into its internal registers.

**2.1.2.1 PAD TIME SELECT.** The 8-bit `pad_time_select` specifies the minimum number of flags to be transmitted before a frame. When a frame is awaiting transmission and a pad value of zero is programmed, the MLAPD transmits only a terminating flag to indicate the end of the current frame and a beginning flag to indicate the start of the next frame. Additional flags may be transmitted if frame transmission cannot begin for a protocol link because less than 6 bytes of data have been transferred to the internal transmit first-in first-out (FIFO). Additional flags may be transmitted if frame transmission cannot begin for a nonprotocol link because 1) less than 10 bytes of data have been transferred to the internal transmit FIFO when the `transmit_address_enable` bit in the frame descriptor is set to zero or 2) less than eight bytes of data have been transferred to the internal transmit FIFO when `transmit_address_enable` is set to one.

**2.1.2.2 NONSTANDARD CONTROL.** This 16-bit GCB entry may specify a control field which is not defined by the LAPD frame structure. When an incoming unnumbered (U) frame

contains this nonstandard\_control field, the MLAPD will accept the frame. The MLAPD takes no protocol action on receipt of this frame. A frame reject (FRMR) condition is not established unless the information field exceeds the negotiated N201 for this link.

The MLAPD implements this feature when bits 0 and 1 of the incoming frame's control field identify the frame as an U frame and the remaining bits in the control field do not correspond to a LAPD-defined U frame. (The specified bit numbers consider that the control field bits are numbered from 0–7, not 1–8 as found in CCITT documents.) The MLAPD then compares the *M* bits in the frame's control field (see format following) against the corresponding bits in the user-specified nonstandard\_control entry and checks whether the frame is a command or response frame. When the frame's control field matches this entry and the frame is either a command or response frame as specified by the user, the MLAPD accepts the frame. The MLAPD identifies the frame as either a nonstandard\_control command frame or a nonstandard\_control response frame in the receive frame descriptor and places the data field (if any) in the addressed logical link's receive queue.

All the *M* bits are user-defined. The user sets bit 8 (*C* in the format) to one allowing the MLAPD to receive a command frame with this nonstandard\_control field and bit 9 (*R* in the format) to one allowing the MLAPD to receive a response frame with this nonstandard\_control field. The value of bit 4, the poll/final (*P/F*) bit, is don't care. The MLAPD will accept frames with the *P/F* bit set to zero or one, but the state of the *P/F* bit is not reported to Level 3. When the user does not wish to accept any nonstandard LAPD U frames, the nonstandard\_control entry must be set to zero.

When the user wishes to transmit a frame with a nonstandard control field, the low-order eight bits of this GCB entry specify the encoding of the control field. The frame\_type entry in the transmit frame descriptor selects either a nonstandard\_control command or response frame type, with the *P/F* bit set to zero or one. During transmission, the MLAPD provides the frame address and the control field.

15	14	13	12	11	10	9	8
X	X	X	X	X	X	R	C
7	6	5	4	3	2	1	0
M	M	M	X	M	M	1	1

nonstandard control field encoding

X = don't care  
 R = response frame  
 C = command frame  
 M = match bits

**2.1.2.3 T203 VALUE.** The nine least significant bits of this 16-bit GCB entry define the maximum time that an established link may be inactive (carries no frames). The CCITT default value for T203 is 60 seconds. If *F* is the MLAPD system clock frequency and *T* is

the desired T203 timer value in seconds, then the value to be loaded into this entry is determined by the following equation:

$$T203\_value = (T \times F) / (2^{20} \times 10)$$

Example 1: T = 10 sec, F = 10 MHz

$$10 \times 10 \times 10^6 / 2^{20} \times 10 = 9.53$$

T203\_value = 9 or 10 decimal

Example 2: T = 100 sec, F = 16.67 MHz

$$100 \times 16.67 \times 10^6 / 2^{20} \times 10 = 158.94$$

T203\_value = 159 decimal

**2.1.2.4 I FRAME RETRANSMIT THRESHOLD.** This 8-bit GCB entry defines the maximum number of times the MLAPD will retransmit an I frame before issuing an interrupt. This value ranges from 1 to 256. When this entry is set to all zeros, the MLAPD will retransmit an I frame 256 times before issuing an interrupt. Retransmission may continue after the interrupt is issued, according to the LAPD procedures.

**2.1.2.5 N200 VALUE.** The least significant five bits of this 8-bit GCB entry define the maximum number of times the MLAPD will retransmit a supervisory (S) command frame or a U command frame. After N200 retransmissions, the MLAPD issues an MDL\_error\_indication interrupt unless masked by the user. The CCITT default value for N200 is three. This value ranges from 0 to 31.

**2.1.2.6 REJ TRANSMIT THRESHOLD.** This 8-bit GCB entry defines the number of reject (REJ) frames transmitted, per link, that are required to generate an interrupt. When this threshold is reached on any link, an interrupt is issued. Once an entry is written into the Interrupt Queue, the REJ transmit frame counter for this logical link is reinitialized. REJ transmit frame counters on other links are not affected. The range of the threshold value is from 1 to 256. When this entry is set to all zeros, the MLAPD considers the threshold to be 256.

**2.1.2.7 RNR TRANSMIT THRESHOLD.** This 8-bit GCB entry defines the number of receive-not-ready (RNR) frames transmitted, per link, that are required to generate an interrupt. When this threshold is reached on any link, an interrupt is issued. Once an entry is written into the Interrupt Queue, the RNR transmit frame counter for this logical link is reinitialized. RNR transmit frame counters on other links are not affected. The range of the threshold value is from 1 to 256. When this entry is set to all zeros, the MLAPD considers the threshold to be 256.

**2.1.2.8 REJ RECEIVE THRESHOLD.** This 8-bit GCB entry defines the number of REJ frames received, per link, that are required to generate an interrupt. When this threshold is reached on any link, an interrupt is issued. Once an entry is written into the Interrupt Queue, the REJ receive frame counter for this logical link is reinitialized. REJ receive frame counters on other links are not affected. The range of the threshold value is from 1 to 256. When this entry is set to all zeros, the MLAPD considers the threshold to be 256.

**2.1.2.9 RNR RECEIVE THRESHOLD.** This 8-bit GCB entry defines the number of RNR frames received, per link, that are required to generate an interrupt. When this threshold is reached on any link, an interrupt is issued. Once an entry is written into the Interrupt Queue, the RNR receive frame counter for this logical link is reinitialized. RNR receive frame counters on other links are not affected. The range of the threshold value is from 1 to 256. When this entry is set to all zeros, the MLAPD considers the threshold to be 256.

**2.1.2.10 CTS TIMEOUT THRESHOLD.** This 16-bit GCB entry allows the user to program the time period that the MLAPD waits for clear-to-send ( $\overline{\text{CTS}}$ ) to be asserted after asserting request-to-send (RTS). This time period is determined by  $(\text{CTS\_timeout\_threshold} \times 2048)$  transmit clock (TxCLK) cycles. After this period expires, the MLAPD issues a CTS\_threshold\_reached interrupt. The CTS\_timeout\_counter is then reinitialized to the threshold value and counting begins again. Additional interrupts continue to be issued whenever the time period expires, until  $\overline{\text{CTS}}$  is asserted. When a value of zero is programmed by the user, the MLAPD will wait ('10000' hex  $\times 2048$ ) TxCLK cycles for  $\overline{\text{CTS}}$  to be asserted.

**2.1.2.11 SCAN LENGTH I QUEUE 0.** This 16-bit GCB entry contains the maximum number of frames from transmit information (I) frame Queue\_0 to be transmitted before switching service to I frame Queue\_1. The value ranges from 0 to 64K - 1. When the scan\_length for I Queue\_0 is defined as zero, the corresponding I frame queue is never serviced. In this way the user can implement fewer queues for I frame transmission.

**2.1.2.12 SCAN LENGTH I QUEUE 1.** This 16-bit GCB entry contains the maximum number of frames from I frame Queue\_1 to be transmitted before switching service to I frame Queue\_2. The value ranges from 0 to 64K - 1. When the scan\_length for I Queue\_1 is defined as zero, the corresponding I frame queue is never serviced. In this way the user can implement fewer queues for I frame transmission.

**2.1.2.13 SCAN LENGTH I QUEUE 2.** This 16-bit GCB entry contains the maximum number of frames from I frame Queue\_2 to be transmitted before switching service to I frame Queue\_3. The value ranges from 0 to 64K - 1. When the scan\_length for I Queue\_2 is defined as zero, the corresponding I frame queue is never serviced. In this way the user can implement fewer queues for I frame transmission.

**2.1.2.14 SCAN LENGTH I QUEUE 3.** This 16-bit GCB entry contains the maximum number of frames from I frame Queue\_3 to be transmitted before switching service to transmit exchange identification/unnumbered information (XID/UI) frames from XID/UI Queue\_0. The value ranges from 0 to 64K – 1. When the scan\_length for I Queue\_3 is defined as zero, the corresponding I frame queue is never serviced. In this way the user can implement fewer queues for I frame transmission.

**2.1.2.15 SCAN LENGTH XID/UI QUEUE 0.** This 16-bit GCB entry contains the maximum number of frames from XID/UI Queue\_0 to be transmitted before switching service to XID/UI Queue\_1. The value ranges from 0 to 64K – 1. When the scan\_length for XID/UI Queue\_0 is defined as zero, the corresponding XID/UI frame queue is never serviced. In this way the user can implement fewer queues for XID/UI frame transmission.

**2.1.2.16 SCAN LENGTH XID/UI QUEUE 1.** This 16-bit GCB entry contains the maximum number of frames from the XID/UI Queue\_1 to be transmitted before switching service to I frame Queue\_0. The value ranges from 0 to 64K – 1. When the scan\_length for XID/UI Queue\_1 is defined as zero, the corresponding XID/UI frame queue is never serviced. In this way the user can implement fewer queues for XID/UI frame transmission.

**2.1.2.17 INTERRUPT MASK.** This 32-bit GCB entry contains the MLAPD interrupt mask, where each bit of the mask corresponds to one interrupt condition. Figure 2-7 shows the interrupt mask bits. **2.8.1 Interrupt Events** defines the interrupt events. If a bit is set to one in this mask and the corresponding interrupt condition occurs, then an entry is written into the Interrupt Queue.  $\overline{\text{IRQ}}$  (INTR) is also asserted (if it is not already asserted), if polling\_select is set to zero. If the mask bit is zero and the corresponding interrupt event occurs, the Interrupt Queue is not updated and no interrupt request is generated. Bus/address\_error and interrupt\_queue\_overflow interrupt conditions cannot be masked by the host.

**2.1.2.18 PROTOCOL ERROR MASK.** As a frame is received, the MLAPD stores any associated data field into a receive buffer. If an error is detected during reception, the MLAPD usually discards any stored data by overwriting this data in the receive buffer with data from a subsequently received frame. Alternately, to allow the user to save these erroneous frames for inspection, the MLAPD supports an error mask.

When a receive frame contains multiple errors, the MLAPD checks the error mask for the errors detected according to the following order:

- 1) RxFIFO overrun
- 2) Abort/nonoctet
- 3) CRC error
- 4) Buffer length exceeded (N201 error)

If an error occurs and the mask bit is set, the MLAPD saves the frame and the corresponding error counter is not decremented. All errors are reported in the receive frame descriptor

31	30	29	28	27	26	25	24
0	UNDEFINED HOST COMMAND	MDL ERROR INDICATION	0	DL RELEASE INDICATION	DL RELEASE CONFIRMATION	DL ESTABLISH INDICATION	DL ESTABLISH CONFIRMATION
23	22	21	20	19	18	17	16
GLOBAL/ XID/UI CONFIRMATION	XID/UI QUEUE_1 CONFIRMATION	XID/UI QUEUE_0 CONFIRMATION	DL DATA CONFIRMATION	DATA INDICATION	LOCAL BUSY	REMOTE STATUS CONFIRMATION	MDL ASSIGN INDICATION
15	14	13	12	11	10	9	8
0	L2 QUEUE OVERFLOW	CAM OVERFLOW	POOL RED LINE	CTS LOST	CTS TIMEOUT THRESHOLD REACHED	I FRAME RETRANSMIT THRESHOLD REACHED	LINK ERROR THRESHOLD REACHED
7	6	5	4	3	2	1	0
ABORT/ NONOCTET THRESHOLD REACHED	CRC ERROR THRESHOLD REACHED	SHORT FRAME THRESHOLD REACHED	DISCARDED FRAME THRESHOLD REACHED	INVALID ADDRESS THRESHOLD REACHED	INACTIVE DLCI THRESHOLD REACHED	TxFIFO UNDERRUN THRESHOLD REACHED	RxFIFO OVERRRUN THRESHOLD REACHED

**Figure 2-7. Interrupt Mask Bits**

error\_code entry. Table 2-1 provides examples of the logic used by the MLAPD when saving a frame.

The least significant four bits of this 16-bit GCB entry contain a mask for four possible receive errors. This mask is used by the MLAPD during frame reception when two conditions are met:

- 1) The error\_mask\_valid bit in the addressed link's LLT receive\_pool\_number entry is set to one, and
- 2) The addressed link is in protocol mode.

**Table 2-1. Examples of Error Mask Handling**

Error Mask Bits				Errors Detected in the Receive Frame				Frame Saved in Memory?
OVERRUN	ABORT	CRC	N201	OVERRUN	ABORT	CRC	N201	
0	0	1	0	0	1	1	0	Yes
0	0	1	1	0	1	1	0	No
0	0	1	1	0	1	1	0	Yes
0	1	0	0	0	1	0	0	No
0	1	1	0	0	0	1	0	Yes
1	0	0	0	0	0	0	1	No
1	0	0	0	1	0	0	0	Yes
1	0	0	0	1	1	1	0	Yes

The protocol\_error\_mask bits are defined as follows:

Bits 15-4 are not used.

These bits must be set to zero.

Buffer\_length\_exceeded (bit 3)

- 1 Receive frames which exceed the negotiated N201 value are accepted. If N201\_value is even, then N201 data bytes are stored in the receive buffer. If N201\_value is odd, then N201-1 data bytes are stored in the receive buffer. Any remaining bytes are discarded.
- 0 Receive frames which exceed the negotiated N201 value are discarded.

RxFIFO\_overrun (bit 2)

- 1 Receive frames which cause a receive FIFO overrun condition are accepted. The data length in the receive frame descriptor is the number of data bytes that were placed in the data buffer before the overrun occurred. For the definition of an overrun condition see **2.1.3.1 RxFIFO OVERRUN THRESHOLD**.
- 0 Receive frames which cause a receive FIFO overrun condition are discarded.

Abort/nonoctet (bit 1)

- 1 Receive frames which are nonoctet aligned or receive frames which are aborted are accepted. The data is placed into the memory buffer as it is received. The last word, which contains the residual data bits, is not transferred to memory. The MLAPD does not give any indication as to the number of received residual bits.
- 0 Receive frames which are nonoctet aligned or receive frames which are aborted are discarded.

CRC\_error (bit 0)

- 1 Receive frames with CRC error are accepted.
- 0 Receive frames with CRC error are discarded.

**2.1.2.19 NONPROTOCOL ERROR MASK.** As a frame is received for a nonprotocol link, the MLAPD places the entire frame into a receive buffer. If a receive error is detected during reception, the MLAPD usually discards the frame by overwriting any bytes in the receive buffer with a subsequently received frame. Alternately, to allow the user to save these erroneous frames for inspection, the MLAPD supports an error mask.

When a receive frame contains multiple errors, the MLAPD checks the error mask for the errors detected according to the following order:

- 1) RxFIFO overrun
- 2) Abort/nonoctet
- 3) CRC error
- 4) Buffer length exceeded (N201 error)

If one or more errors is detected and one of the corresponding error mask bits is set, then the MLAPD saves the frame and does not decrement the corresponding error counters. If



one or more errors is detected and none of the corresponding error mask bits is set, then the MLAPD discards the frame and decrements the corresponding error counters. All errors are reported in the receive frame descriptor error\_code\_entry. Table 2-1 provides examples of the logic used by the MLAPD when saving a frame.

The least significant four bits of this 16-bit GCB entry contain a mask for four possible receive errors. This mask is used by the MLAPD during frame reception when two conditions are met:

- 1) The error\_mask\_valid bit in the addressed link's LLT pool\_number entry is set to one, and
- 2) The addressed link is assigned to nonprotocol mode.

The nonprotocol\_error\_mask bits are defined as follows:

Bits 15-4 are not used.

These bits must be set to zero.

Buffer\_length\_exceeded (bit 3)

- 1 When the GCB multibuffer\_select bit is set to zero, receive frames which exceed the specified N201 value are accepted. N201 bytes are stored in the receive buffer. When multibuffer\_select is set to one and the receive pool becomes empty while receiving a frame, the partially received frame is placed into the receive queue and both the first and last frame descriptors for this frame are marked with the buffer\_length\_exceeded encoding.
- 0 When the GCB multibuffer\_select bit is set to zero, receive frames which exceed the specified N201 value are discarded. When multibuffer\_select is set to one and the receive pool becomes empty while receiving a frame, the partially received frame is not placed in the receive queue. Its buffers are returned to the receive pool.

RxFIFO\_overrun (bit 2)

- 1 Receive frames which cause a receive FIFO overrun condition are accepted. The data length in the receive frame descriptor is the number of bytes that were placed in the receive buffer before the overrun occurred. For the definition of an overrun condition see **2.1.3.1 RxFIFO OVERRUN THRESHOLD**.
- 0 Receive frames which cause a receive FIFO overrun condition are discarded.

Abort/nonoctet (bit 1)

- 1 Receive frames which are nonoctet aligned or receive frames which are aborted are accepted. The information between the opening and closing flag is placed into the memory buffer as it is received. The data\_length entry in the frame descriptor includes the two bytes of CRC (or, in other words, the last two bytes of the frame). The last word or byte contains the residual bits. The MLAPD does not give any indication as to the number of received residual bits. Also, the MLAPD does not pad out the last word or byte with a predictable bit pattern.

- 0 Receive frames which are nonoctet aligned or receive frames which are aborted are discarded.

CRC\_error (bit 0)

- 1 Receive frames with CRC error are accepted.
- 0 Receive frames with CRC error are discarded.

**2.1.2.20 FILTER MASK AND FILTER MATCH.** These 32-bit GCB entries are meaningful only when the promiscuous\_receive\_select bit is set to one and the filter\_select bit is set to one. Otherwise, these entries are not used and should be set to zero. The filter\_mask entry combined with the filter\_match entry allow the user to selectively accept frames from the link based upon the first 32-bits of each frame. See **9.2.2 Filter Mode**.

### 2.1.3 Statistics Threshold Area

The length of this area is 32 bytes from '60' (hex) to '7F' (hex). The MLAPD reads these entries at initialization (INIT command) and stores them internally. The MLAPD has an internal register with the same name that corresponds to each entry and is used as a down counter. When the counter reaches zero, an interrupt is generated, and an entry is written into the Interrupt Queue. The internal register is then reinitialized to the threshold value. The user can change any of these thresholds while the MLAPD is in the on-line state by issuing a PRESET\_STATISTICS command to cause the MLAPD to reinitialize each counter to the new threshold values specified in the GCB. The user can view the status of these error counters by issuing a DUMP\_STATISTICS command.

**2.1.3.1 RxFIFO OVERRUN THRESHOLD.** This 16-bit GCB entry contains the number of receive FIFO (RxFIFO) overrun conditions that must occur before the MLAPD generates an interrupt. An overrun condition occurs when the MLAPD attempts to write to its RxFIFO during frame reception and the FIFO is full. An overrun indicates that the MLAPD was unable to DMA (direct memory access) information from the RxFIFO into memory buffers quickly enough to keep up with the serial bit rate. If this condition occurs frequently, it indicates that the MLAPD is not receiving sufficient system bus bandwidth. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.2 TxFIFO UNDERRUN THRESHOLD.** This 16-bit GCB entry contains the number of transmit FIFO (TxFIFO) underrun conditions that must occur before the MLAPD generates an interrupt. An underrun condition occurs when the MLAPD transmit task attempts to read from an empty TxFIFO during frame transmission. The MLAPD sends an abort sequence (seven consecutive ones) to terminate the frame. An underrun condition indicates that the MLAPD was unable to DMA information from the transmit memory buffer into the TxFIFO quickly enough to keep up with the serial bit rate. If this condition occurs frequently, it indicates that the MLAPD is not receiving sufficient system bus bandwidth. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.3 INACTIVE DLCI THRESHOLD.** This 16-bit GCB entry contains the number of frames that must be received with an inactive Data-Link-Connection Identifier (DLCI) before the MLAPD generates an interrupt. The MLAPD determines whether a DLCI is active by accessing the external Match Table in expanded-system operation mode or by accessing the internal content addressable memory (CAM) in on-chip system operation mode. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.4 INVALID ADDRESS THRESHOLD.** This 16-bit GCB entry contains the number of frames that must be received with an invalid address structure before the MLAPD generates an interrupt. This address structure only applies to links assigned to LAPD operation. An invalid address structure is defined as any address structure other than the following:

15	14	13	12	11	10	9	8
X	X	X	X	X	X	X	1
7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	0

X = don't care

When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.5 DISCARDED FRAME THRESHOLD.** This 16-bit GCB entry contains the number of frames that must be discarded (due to the lack of receive data buffers) before the MLAPD generates an interrupt. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.6 SHORT FRAME THRESHOLD.** This 16-bit GCB entry contains the number of LAPD frames with less than five bytes between flags that must be received before the MLAPD generates an interrupt. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.7 CRC ERROR THRESHOLD.** This 16-bit GCB entry contains the number of frames received with cyclical redundancy check (CRC) error that are required to generate an interrupt. Because the DLCI of a frame with a CRC error may not be correct, it is meaningless to post CRC errors to individual DLCIs. Therefore, the counting of frames with CRC errors is handled on a global basis. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

**2.1.3.8 ABORT OR NONOCTET THRESHOLD.** This 16-bit GCB entry contains the number of aborted frames or nonoctet aligned frames that must be received before the MLAPD generates an interrupt. When this entry is set to all zeros, the MLAPD considers the threshold to be  $2^{16}$ .

## 2.1.4 Command Arguments Area

The length of this area is 16 bytes from '80' (hex) to '8F' (hex). Some MLAPD commands require one or more arguments. The host writes the argument(s) into this area before issuing the command. Each MLAPD command individually defines the usage of these three arguments, if required.

**2.1.4.1 COMMAND ARGUMENT 1.** This GCB entry may contain a 16-bit argument associated with a host command.

**2.1.4.2 COMMAND ARGUMENT 2.** This GCB entry may contain a 32-bit pointer argument associated with a host command.

**2.1.4.3 COMMAND ARGUMENT 3.** This GCB entry may contain a 32-bit pointer argument associated with a host command.

## 2.1.5 User Area

The user may choose to reserve a user area located just beyond the command arguments area of the GCB. The values to be stored in this area and the length of this area are defined by the user. The MLAPD does not read or write to the user area. The following user pointers are required to interface with the MLAPD, and the user may opt to store these pointers in this area.

**2.1.5.1 USER INTERRUPT QUEUE READ POINTER.** This 32-bit pointer indicates the next entry to be handled by the host in the Interrupt Queue.

**2.1.5.2 USER GLOBAL XID/UI Tx NEXT CONFIRM POINTER.** This 32-bit pointer indicates the next frame to be collected by the host from the Global XID/UI Queue.

**2.1.5.3 USER GLOBAL XID/UI Tx LAST QUEUED POINTER.** This 32-bit pointer indicates the last frame queued by the host for transmission on the Global XID/UI Queue.

**2.1.5.4 USER XID/UL<sub>0</sub> Tx NEXT CONFIRM POINTER.** This 32-bit pointer indicates the next frame to be collected by the host from the XID/UI Queue<sub>0</sub>.

**2.1.5.5 USER XID/UL<sub>0</sub> Tx LAST QUEUED POINTER.** This 32-bit pointer indicates the last frame queued by the host for transmission on the XID/UI Queue<sub>0</sub>.

**2.1.5.6 USER XID/UL\_1 Tx NEXT CONFIRM POINTER.** This 32-bit pointer indicates the next frame to be collected by the host from the XID/UI Queue\_1.

**2.1.5.7 USER XID/UL\_1 Tx LAST QUEUED POINTER.** This 32-bit pointer indicates the last frame queued by the host for transmission on the XID/UI Queue\_1.

## 2.2 MATCH TABLE

The Match Table is accessed in the expanded operation mode only. The start of the table is defined by the `match_table_pointer` in the GCB. The MLAPD uses this table to obtain the Logical-Link Identification (LLID) number of the logical link whose DLCI is contained in an incoming frame. The incoming DLCI is used as an index into this table.

The host must zero this table before initializing the MLAPD, defining all entries as invalid. After initialization by the host, the MLAPD device is responsible for updating this table as logical links are assigned or removed. During the assignment procedure, the MLAPD copies the LLID, `nonprotocol_select` bit and `user/network_select` bit from the link's LLT into the appropriate Match Table entry. Each 16-bit Match Table entry is organized as follows:

Valid\_DLCI (bit 15)

- 1 This DLCI has been assigned to a logical link. The associated LLID is contained in this Match Table entry.
- 0 This DLCI has not been assigned to a logical link. This Match Table entry is not valid.

Nonprotocol\_select (bit 14)

- 1 This logical link is assigned to nonprotocol mode. The MLAPD will not implement the LAPD procedures for this logical link.
- 0 The MLAPD implements the LAPD protocol for this logical link.

User/network (bit 13)

- 1 This logical link is defined as a "user" station. The MLAPD will set the C/R bit to zero in transmitted command frames and to one in transmitted response frames.
- 0 This logical link is defined as a "network" station. The MLAPD will set the C/R bit to one in transmitted command frames and to zero in transmitted response frames.

LLID (bits 12-0)

These bits contain the LLID associated with this DLCI.

The length of the Match Table is fixed by the number of possible DLCIs. Therefore, the Match Table length is 8192 words. The Match Table format is described in Figure 2-8.

	15	12	0	
BASE + 0	1	1	U/N	LLID ASSOCIATED WITH DLCI = 0
BASE + 2	0	X	X	LLID ASSOCIATED WITH DLCI = 1
BASE + 4	1	0	U/N	LLID ASSOCIATED WITH DLCI = 2
	⋮			⋮
BASE + 16K - 4	0	X	X	LLID ASSOCIATED WITH DLCI = 8190
BASE + 16K - 2	0	X	X	LLID ASSOCIATED WITH DLCI = 8191

X = DON'T CARE Condition

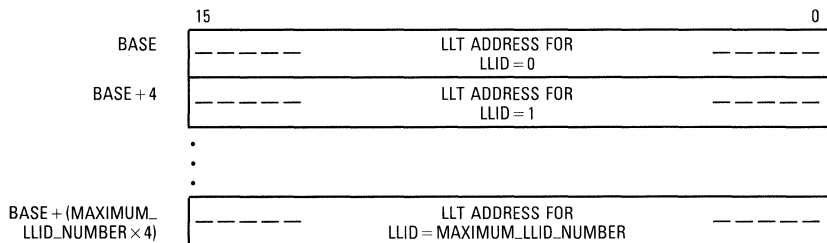
NOTE: In this example, only DLCI '0' and '2' are assigned to active logical links. The MLAPD implements the LAPD protocol only for the link with DLCI = '2'.

**Figure 2-8. Match Table Format**

### 2.3 LLID-LLT TABLE

The Logical-Link Identification to Logical-Link Table (LLID-LLT) Table is maintained by the host. The MLAPD only reads information from this table. The LLID-LLT Table identifies the address of a logical link's LLT, using the link's LLID as an index. The start of the table is defined by the LLID-LLT\_table\_pointer in the GCB. This table must accommodate a 32-bit entry for each LLID that may be assigned to a logical link by higher level software.

Before issuing an MDL\_ASSIGN\_REQUEST, the host is responsible for creating a LLT for the logical link and storing the address of the LLT in the appropriate LLID-LLT Table entry. When a logical link is removed from service, the host may remove the corresponding LLID-LLT entry if desired. In any event, the MLAPD will not access this entry after link removal since the DLCI will be marked invalid in the Match Table. The LLID-LLT Table format is described in Figure 2-9.



**Figure 2-9. LLID-LLT Table**

## 2.4 LOGICAL-LINK TABLE

A Logical-Link Table (LLT) is associated with each active logical link. The LLT contains user-defined constants and the variables and pointers managed by the MLAPD for Level 2 handling. The host may not change the LLT entries after a link is activated via an MDL\_ASSIGN\_REQUEST or ACTIVATE\_LL command, except as noted in the description of each entry. The MLAPD reads the constant and variable entries in a link's LLT to initialize link-specific registers when the link is to be serviced. During and after a particular link service task is completed, the MLAPD updates the variable entries in the LLT.

After preparing the LLT, the host need not access this table again, except for diagnostic purposes. However, the host may wish to reserve additional words beyond the end of the LLT for use by higher levels. The two additional entries required by the higher level software to interface with the MLAPD are indicated at the end of the LLT. These entries may actually reside anywhere in memory, as the MLAPD does not access them. The LLT format is shown in Figure 2-10.

HEX DISPLACEMENT	15	8	7	0	
0	TRANSMIT STATUS				
2	CONFIGURATION	DLCI			
4	MAXIMUM OUTSTANDING I FRAMES				
6	V(A)				
8	V(S)	0	V(R)	0	
A	LINK STATUS				
C	Tx I QUEUE NUMBER		T200 VALUE		
E	N201 VALUE				
10	EMV*	RECEIVE POOL NUMBER			
12	-----		Tx NEXT POINTER		-----
14					
16	-----		Tx NEXT ACKNOWLEDGE POINTER		-----
18					
1A	-----		Tx NEXT LLT POINTER		-----
1C					
1E	REJ Tx COUNT		RNR Tx COUNT		
20	REJ Rx COUNT		RNR Rx COUNT		
22					
24	-----		USER Tx NEXT CONFIRM POINTER		-----
26					
28	-----		USER Tx LAST QUEUED POINTER		-----

\*Error Mask Valid

**Figure 2-10. Logical-Link Table Format**

## 2.4.1 Transmit Status

This 16-bit LLT entry contains the logical link's Transmit Queue status. The host zeros this entry before the LLT is first presented to the MLAPD. Thereafter, the MLAPD sets and resets the bits. The transmit-status bits are defined as follows:

Bits 15–5 are not used.

These bits are set to zero by the host.

### Stop-Tx (bit 4)

- 1 The MLAPD will not transmit any remaining I frames for this logical link due to one of the following conditions:
  - A STOP\_TX\_I command has been issued by the host for this logical link.
  - A DL\_RELEASE\_REQUEST, DEACTIVATE\_LL, MDL\_REMOVE\_REQUEST, or DL\_ESTABLISH\_REQUEST command has been issued by the host.
  - A link reset operation has been initiated by the remote station (DISC or SABME frame was received while in the connected state).
  - A link reset operation has occurred due to any other situation, according to the LAPD protocol.

The MLAPD immediately sets the stop-Tx bit in the transmit-status to indicate that this LLT must be removed from its assigned I frame queue. When the MLAPD transmit task next services this logical link as part of the transmit servicing scheme, the link is removed from the I frame queue, and its active bit is set to zero (if it is not already zero). The transmit servicing scheme is explained in **5.1 TRANSMIT SERVICING SCHEME**.

- 0 The logical link associated with this LLT has no stop-Tx condition and may or may not be linked to its assigned transmit I frame queue.

### Active (bit 3)

- 1 The Transmit Queue associated with this LLT is linked to the appropriate I frame queue. Transmission of frames for this logical link will be performed as part of the transmit servicing scheme.
- 0 The Transmit Queue associated with this LLT is not linked to its I frame queue. This condition may be static (no pending transmit activity) or temporary (current link conditions do not permit transmit activity). In either case, the LLT will be relinked to the appropriate I frame queue when a host command or a link event enables transmit activity.

### Acknowledge (bit 2)

- 1 This logical link has one or more frames waiting for acknowledgement.
- 0 All the frames in this logical link's Transmit Queue have been transmitted and acknowledged. Before the MLAPD sets this bit to zero, the MLAPD again checks the link's Transmit Queue to verify that no additional frames have been added to the queue for transmission. If additional frames are found, the MLAPD again places this link's LLT in its assigned I frame queue and sets the transmit and active bits to one.



### Transmit (bit 1)

- 1 The MLAPD has more frames to transmit for this logical link. This logical link's Transmit Queue may or may not be linked to its I frame queue depending on the link's status information.
- 0 This logical link has no transmit I frames queued for transmission.

### Last\_LLT (bit 0)

- 1 This LLT is the last LLT linked to this I frame queue. The Tx\_next\_LLT\_pointer entry in this LLT is not valid.
- 0 At least one more LLT is linked to this I frame queue, and the Tx\_next\_LLT\_pointer entry in this LLT is valid.

The transition table for the transmit\_status bits is shown in Figure 2-11.

Transmit Status Bits					Input Conditions									
	Stop_Tx	Active	Acknowledge	Transmit	DL_DATA_REQUEST	RELINK_REQUEST	STOP_TX_I	MDL_REMOVE_REQUEST DL_RELEASE_REQUEST DEACTIVATE_LL DL_ESTABLISH_REQUEST	All Frames Transmitted	All Transmitted Frames Acknowledged and Finds No Additional Frames for Transmission	All Transmitted Frames Acknowledged and Finds Additional Frames for Transmission	P Flag/Remote Busy/ K Limit/Timer Recovery/ Remote Status Request	Condition Clears Relink to Queue	Link Reset
S0 =	0	0	0	0	S7	ILL	ILL	S8	X	X	X	X	X	S0
S2 =	0	0	1	0	ILL	S7	ILL	S8	X	S7	S0	X	X	S0
S3 =	0	0	1	1	ILL	S3	SA	S8	X	X	X	S3	S7	S0
S7 =	0	1	1	1	ILL	S7	SE	SC	S2	X	X	S3	X	S0
S8 =	1	0	0	0	S7	ILL	ILL	S8	X	X	X	X	X	S0
SA =	1	0	1	0	ILL	ILL	ILL	S8	X	S8	S8	X	X	S0
SC =	1	1	0	0	S7	ILL	ILL	SC	X	X	X	X	X	S0
SE =	1	1	1	0	ILL	ILL	ILL	SC	X	S8	S8	X	X	S0

ILL - Illegal Command (MOL\_error\_indication argument = '000000' and MLAPD takes no action)  
X - Impossible or DON'T CARE Condition

**Figure 2-11. Transmit Status Bits Transition Table**

## 2.4.2 DLCI and Configuration Bits

This 16-bit entry is written by the host and read by the MLAPD. This entry contains the assigned DLCI for the logical link and is considered a constant. The MLAPD assumes that this entry is not modified after an ACTIVATE\_LL or MDL\_ASSIGN\_REQUEST command is issued until a DEACTIVATE\_LL command is given. The DLCI is contained in the low-order 13 bits. The remaining bits are defined as follows:

Bit 15 is zeroed by the host.

Nonprotocol\_select (bit 14)

- 1 This logical link is assigned to nonprotocol mode. The MLAPD will not implement the LAPD protocol for this link. See **9.1 NONPROTOCOL LINKS**.
- 0 The MLAPD implements the LAPD protocol for this logical link.

User/network\_select (bit 13)

- 1 The logical link associated with this LLT is defined as a "user" station. The MLAPD will set the C/R bit to zero in transmitted command frames and to one in transmitted response frames.
- 0 The logical link associated with this LLT is defined as a "network" station. The MLAPD will set the C/R bit to one in transmitted command frames and to zero in transmitted response frames.

## 2.4.3 Maximum Number of Outstanding I Frames (K)

This entry is written by the host and read by the MLAPD. The host may only change this entry when no transmit queue is active for the link. That is, a DL\_data\_confirmation interrupt is issued, or both the active bit and the acknowledge bit in the transmit\_status bits are set to zero. The most significant seven bits of this 16-bit LLT entry define the maximum number of I frames which can be sent on this logical link before an acknowledgement is required from the remote station. The MLAPD transmits numbered frames for the link as long as  $V(S) - V(A)$  is less than K. When K is set to zero, no numbered frames can be transmitted. The maximum value of K is 127 for modulo 128 operation. Bit 8 must be set to zero.

## 2.4.4 V(A)

This entry is written and read by the MLAPD. The most significant seven bits of this 16-bit LLT entry contain the acknowledge\_state\_variable, V(A). V(A) identifies the next frame to be acknowledged by the remote station. The MLAPD initializes V(A) to zero following link establishment. Bit 8 is always set to zero.

## 2.4.5 V(S)/V(R)

This entry is written and read by the MLAPD. This 16-bit LLT entry contains the link's send and receive state variables, which are specified in the numbered information frame format.

The 7-bit `send_state_variable`, V(S), denotes the sequence number of the next in-sequence I frame to be transmitted. This sequence number is stored in bits 15-9. Bit 8 is always set to zero. The 7-bit `receive_state_variable`, V(R), denotes the expected sequence number of the next in-sequence I frame to be received. This sequence number is stored in bits 7-1. Bit 0 is always set to zero. The V(S)/V(R) entry is initialized to zero by the MLAPD during link establishment. When a particular link service task is completed, e.g., a frame is transmitted, the V(S)/V(R) value is written back into the LLT if it has changed.

### 2.4.6 Link Status

Each active logical link is operated according to the finite-state machine defined by the LAPD protocol. This 16-bit LLT entry contains the current state of the link's finite-state-machine and other status bits that define the link's state. The host must initialize this word to all zeroes before presenting the LLT to the MLAPD. The MLAPD will thereafter modify this status word based upon events occurring on this logical link.

Bits 15 and 14 are not used.

These bits are always set to zero by the host.

`Remote_status_check` (bit 13)

- 1 The host has requested updated status information for this link via a `REMOTE_STATUS_REQUEST` command. A remote status response is expected.
- 0 A remote status response is not expected.

`Remote_busy` (bit 12)

- 1 The remote station is in a busy condition.
- 0 The remote station is not in a busy condition.

`Local_busy` (bit 11)

- 1 The host has issued a `SET_LOCAL_BUSY` command. No frames with a data field may be received for this link.
- 0 The host has not disabled reception of frames with data fields for this link.

`XID_flag` (bit 10)

- 1 An XID command frame has been sent, and the MLAPD is waiting for an XID response frame from the remote station.
- 0 An XID response frame is not expected.

`Status_inquiry` (bit 9)

- 1 T203 expired, and the MLAPD transmitted a S frame with the poll bit set to one.
- 0 A remote status response is not expected.

`P_flag` (bit 8)

- 1 The MLAPD transmitted a command frame with the poll (P) bit set to one, and a response frame with the final (F) bit set to one is expected.
- 0 A response frame with the F bit set to one is not expected.

Bits 7 and 6 are not used.

These bits are always set to zero by the host.

2

#### Busy\_reject (bits 5,4)

When the logical link is in the MF\_EST\_BUSY or TM\_REC\_BUSY state and cannot receive I frames, this flag determines whether the MLAPD must send a REJ response frame when the link exits the busy state.

- 00 No received I frame has been discarded. Therefore, a REJ frame is not necessary.
- 01 An I frame has been discarded and a REJ frame must be sent.
- 10 A REJ frame has already been sent.
- 11 Not a valid encoding.

#### Link\_state (bits 3-0)

These four bits encode the current state (in binary) of the link's finite-state-machine. The encodings with numeric state values 7.0, 7.1, and 7.2 are the multiple-frame established states. (For a listing of the link states, see Table 8-1.) The encodings with numeric state values of 8.0, 8.1, and 8.2 are the timer recovery states. The state encodings are shown as follows:

- 0000 Uninitialized state. This value is programmed in the link\_status word by the host before this link is presented to the MLAPD via an MDL\_ASSIGN\_REQUEST or ACTIVATE\_LL command.
- 0001 TEL\_UNASSIGN (numeric state value 1)
- 0010 Undefined. The MLAPD will not write this bit encoding.  
The ASSIGN\_WAIT\_TEI state (numeric state value 2) is not implemented. When an XID/UI frame transmission request cannot be performed because a TEI value has not been assigned, the MLAPD will not store any information field associated with the XID/UI frame. Storage is not possible because the MLAPD does not support a UI queue per link. If a DL\_UI frame transmission is requested for a link in TEL\_UNASSIGN, the link will not change states, and the MLAPD will return a negative confirmation for the frame.
- 0011 EST\_WAIT\_TEI (numeric state value 3)
- 0100 TEL\_ASSIGNED (numeric state value 4)
- 0101 AWAIT\_EST (numeric state value 5)
- 0110 AWAIT\_REL (numeric state value 6)
- 0111 Undefined. The MLAPD will not write this bit encoding.
- 1000 MF\_EST\_NORM (numeric state value 7.0)
- 1001 MF\_EST\_REJ (numeric state value 7.1)
- 1010 MF\_EST\_BUSY (numeric state value 7.2)
- 1011 Undefined. The MLAPD will not write this bit encoding.
- 1100 TM\_REC\_NORM (numeric state value 8.0)
- 1101 TM\_REC\_REJ (numeric state value 8.1)
- 1110 TM\_REC\_BUSY (numeric state value 8.2)
- 1111 Undefined. The MLAPD will not write this bit encoding.

### 2.4.7 Tx I Queue Number

This entry is written by the host and read by the MLAPD. The host may only change this entry when no transmit queue is active for the link. That is, a DL\_data\_confirmation interrupt is issued, or both the active bit and the acknowledge bit in the transmit\_status bits are set to zero. Bits 10 and 9 of this 7-bit LLT entry specify the transmit frame queue assigned by the host for all I frame transmission on this link. The MLAPD queues I frames for this link to one of the four I frame queues (0-3) based on the value of this LLT entry. Bits 15–11 of this entry must be zeroed by the host.

### 2.4.8 T200 Value

This entry is written by the host and read by the MLAPD. The host may change this entry at any time, and the MLAPD will use the new T200 value the next time that timer T200 is started. This 9-bit LLT entry (bits 8-0) defines the timeout value for the acknowledgement timer (T200). The CCITT default for T200 is one second. If F is the system clock frequency and T is the T200 timer value in seconds, then the value to be loaded in the T200\_value entry in the LLT is calculated as follows:

$$T200\_value = T \times F / 2^{20}$$

Example 1: T = 1 sec, F = 10 MHz

$$1 \times 10^7 / 2^{20} = 9.53$$

T200\_value = 9 or 10 decimal

Example 2: T = 25 sec, F = 16.67 MHz

$$25 \times 16.67 \times 10^6 / 2^{20} = 397.36$$

T200\_value = 397 or 398 decimal

### 2.4.9 N201 Value

This entry is written by the host and read by the MLAPD. This entry may be changed by the host following a SET\_LOCAL\_BUSY command. The 15 low-order bits of this LLT entry are used to specify the N201\_value. The function of the N201\_value entry for protocol, nonprotocol, and promiscuous receive operation is described in the following paragraphs. This entry can range from 0 to (32K-1) bytes. When a zero is programmed, data can not be transferred. Bit 15 of this entry must be set to zero by the host.

For a logical link operating in protocol mode, this LLT entry defines the maximum number of octets allowed in an information field of a frame. The CCITT default value is 260 bytes. The minimum frame size that can be received is 5 bytes for unnumbered frames and 6 bytes for numbered frames (information and supervisory frames). The minimum buffer size is 0 bytes.

For a logical link operating in nonprotocol mode with the `multibuffer_select` bit set to zero, this LLT entry defines the maximum number of bytes that can be written to a data buffer and must be specified as an even number. The minimum frame size that can be received is 5 bytes for nonprotocol links. The user must provide a minimum of 6 bytes for each memory buffer.

For a logical link operating in nonprotocol mode with the `multibuffer_select` bit set to one, this LLT entry defines the maximum number of bytes that will be stored in a single receive buffer and must be an even number. If the number of bytes between the opening flag and closing flag of the frame is greater than the link's `N201_value`, then the MLAPD will fetch additional buffers as needed. Except for the last buffer, each buffer will contain  $(N201\_value - 2)$  bytes. The user must provide a minimum of 6 bytes for each memory buffer.

The MLAPD verifies that incoming receive frames satisfy the `N201_value` requirement. The user should allocate enough memory space for each receive data buffer to contain at least `N201` bytes. Table 2-2 lists the `N201_value` range, minimum frame size, maximum data length of the frame, the buffer size, and the maximum number of bytes written to each data buffer in protocol or nonprotocol. The minimum frame size is the minimum number octets between the opening and closing flags. Frames shorter than the minimum frame size will be discarded. Frames longer than the maximum frame size will be treated as long frames, and all of the extra bytes will be discarded.

For transmission, the data length is specified by the user in each transmit frame descriptor. The MLAPD does not check that the transmit data length is less than or equal to `N201`.

**Table 2-2. Receive Frame Parameter Summary**

Mode	N201_value Range	Minimum Frame Size	Maximum Data Length†	Buffer Size	Maximum Number of Bytes Written to Buffer
Protocol	0 to $(32K - 1)$	5†† 6††	N201	$\geq N201$ and Even	N201
Nonprotocol (S)	6 to $(32K - 2)$ and even	5	N201	N201	N201
Nonprotocol (M)	6 to $(32K - 2)$ and even	5	infinite	N201 N201	$N201 - 2$ ††† $N201$ †††

NOTES:

- † — For a protocol link, the maximum data length is the maximum number of octets in the information field. For a nonprotocol link, the maximum data length is the maximum number of octets between the opening and the closing flags.
  - †† — The minimum size for protocol frames is 5 bytes for unnumbered frames and 6 bytes for numbered frames (Information and Supervisory frames).
  - ††† — In multibuffer mode,  $N201 - 2$  bytes is the maximum number of bytes written to all of the buffers except the last buffer. `N201` is the maximum number of bytes written to the last buffer.
  - (S) — Single Buffer Mode
  - (M) — Multibuffer Mode
- All sizes are given in bytes unless otherwise specified.

#### 2.4.10 Receive Pool Number and Error Mask Valid (EMV)

This entry is written by the host and read by the MLAPD. This entry may be changed by the host following a SET\_LOCAL\_BUSY command. The 13 low-order bits of this 16-bit LLT entry identify the receive pool assigned to this link. During frame reception, the 13-bit receive\_pool\_number is used as an offset into the Receive Pool Pointers Table to locate the next receive frame descriptor to be fetched. Refer to **6.1.1 Receive Pool Format**. The length of each data buffer in a receive pool must be equal to or greater than the largest N201 of all logical links which share this pool.

This 16-bit LLT entry also contains the error\_mask\_valid bit. The remaining bits in this entry are defined as follows:

Error\_mask\_valid (bit 15)

- 1 This logical link accepts erroneous receive frames according to the protocol\_error\_mask entry in the GCB if this link implements the LAPD procedures or according to the nonprotocol\_error\_mask entry in the GCB if this link is assigned to nonprotocol operation.
- 0 This logical link does not accept erroneous receive frames.

Bits 14 and 13 are not used.

These bits must be set to zero by the host.

#### 2.4.11 Tx Next Pointer

This entry is written and read by the MLAPD. This 32-bit pointer contains either the address of the I frame descriptor with the data buffer currently being transmitted or the address of the I frame descriptor with the data buffer that will be transmitted when the link is next serviced. This pointer is initialized by the MLAPD to point to the first transmit I frame descriptor in the link's Transmit Queue based on the command argument associated with a DL\_DATA\_REQUEST.

#### 2.4.12 Tx Next Acknowledge Pointer

This entry is written and read by the MLAPD. This 32-bit pointer contains the address of the first transmit I frame descriptor that is not yet acknowledged. A DL\_DATA\_REQUEST command causes the MLAPD to initialize this pointer to point to the first transmit I frame descriptor in the link's Transmit Queue. The MLAPD then updates this entry as transmitted frames are acknowledged.

#### 2.4.13 Tx Next LLT Pointer

This entry is written and read by the MLAPD. This 32-bit pointer is the linking mechanism for placing a logical link's Transmit Queue into its assigned I frame queue. When the

last-LLT bit in the link's transmit-status bits is set to zero, then this entry indicates the address of the LLT for the next logical link that is linked to the same transmit I frame queue.

#### 2.4.14 REJ Tx Counter

This entry is written and read by the MLAPD. The MLAPD counts the number of REJ frames transmitted on each link when link statistics are enabled by the user. The user enables link statistics by setting the link-statistics-select bit to one in the option-bits-2 entry of the GCB.

This 8-bit LLT entry is the REJ-transmit counter for this link. The counter is initialized by the MLAPD to the REJ-transmit-threshold during link assignment. Thereafter, the counter is decremented each time a REJ frame is transmitted on this logical link. When the counter reaches zero, the MLAPD informs the host via the Interrupt Queue, and the counter is reinitialized to the threshold value.

No method exists to allow the user to immediately reinitialize a link's REJ-transmit-counter to the threshold value defined in the GCB. The user may change the threshold value in the GCB and issue a RELOAD command. Then each link's counter is reinitialized to the new threshold value when it expires.

#### 2.4.15 RNR Tx Counter

This entry is written and read by the MLAPD. The MLAPD counts the number of RNR frames transmitted on each link when link statistics are enabled by the user. The user enables link statistics by setting the link-statistics-select bit to one in the option-bits-2 entry of the GCB.

This 8-bit LLT entry is the RNR-transmit counter for this link. The counter is initialized by the MLAPD to the RNR-transmit-threshold during link assignment. Thereafter, the counter is decremented each time a RNR frame is transmitted on this logical link. When the counter reaches zero, the MLAPD informs the host via the Interrupt Queue, and the counter is reinitialized to the threshold value.

No method exists to allow the user to immediately reinitialize a link's RNR-transmit-counter to the threshold value defined in the GCB. The user may change the threshold value in the GCB and issue a RELOAD command. Then each link's counter is reinitialized to the new threshold value when it expires.

#### 2.4.16 REJ Rx Counter

This entry is written and read by the MLAPD. The MLAPD counts the number of REJ frames received on each link when link statistics are enabled by the user. The user enables link



statistics by setting the `link_statistics_select` bit to one in the `option_bits_2` entry of the GCB.

This 8-bit LLT entry is the `REJ_receive_counter` for this link. The counter is initialized by the MLAPD to the `REJ_received_threshold` during link assignment. Thereafter, the counter is decremented each time a REJ frame is received on this logical link. When the counter reaches zero, the MLAPD informs the host via the Interrupt Queue, and the counter is reinitialized to the threshold value.

No method exists to allow the user to immediately reinitialize a link's `REJ_receive_counter` to the threshold value defined in the GCB. The user may change the threshold value in the GCB and issue a RELOAD command. Then each link's counter is reinitialized to the new threshold value when it expires.

#### **2.4.17 RNR Rx Counter**

This entry is written and read by the MLAPD. The MLAPD counts the number of RNR frames received on each link when link statistics are enabled by the user. The user enables link statistics by setting the `link_statistics_select` bit to one in the `option_bits_2` entry of the GCB.

This 8-bit LLT entry is the `RNR_receive_counter` for this link. The counter is initialized by the MLAPD to the `RNR_received_threshold` during link assignment. Thereafter, the counter is decremented each time a RNR frame is received on this logical link. When the counter reaches zero, the MLAPD informs the host via the Interrupt Queue, and the counter is reinitialized to the threshold value.

No method exists to allow the user to immediately reinitialize a link's `RNR_receive_counter` to the threshold value defined in the GCB. The user may change the threshold value in the GCB and issue a RELOAD command. Then each link's counter is reinitialized to the new threshold value when it expires.

#### **2.4.18 User Tx Next Confirm Pointer**

This entry is written and read by the host. This 32-bit pointer contains the address of the next frame descriptor in the link's Transmit Queue to be confirmed by the MLAPD. The host updates this pointer after each frame descriptor is removed.

#### **2.4.19 User Tx Last Queued Pointer**

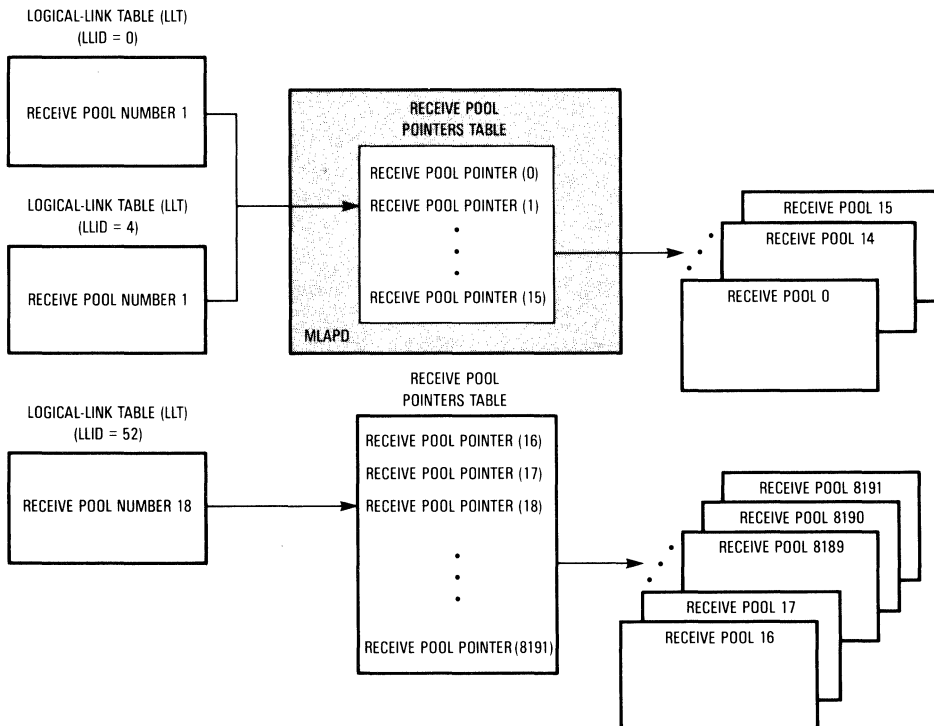
This entry is written and read by the host. This 32-bit pointer contains the address of the last frame descriptor that the host queued to the link's Transmit Queue. The host updates this pointer when new frames are added to the link's Transmit Queue for transmission.

## 2.5 RECEIVE POOL POINTERS TABLE

The Receive Pool Pointers Table is written and read by the MLAPD. A portion of this table is located on-chip, and the remainder of this table is located in shared memory. The address of the table in shared memory is defined by the `pool_table_pointer` entry in the GCB.

The Receive Pool Pointers Table allows the host to define up to 8192 receive pools. The address of the first receive frame descriptor in each receive pool is identified by the corresponding `receive_pool_pointer`. The Receive Pool Pointers Table is a sequential list of `receive_pool_pointers` for pools 16 to 8191. The MLAPD loads a `receive_pool_pointer` into this table as the result of a `ASSIGN_POOL_POINTER` command.

The first 16 `receive_pool_pointers` (0 to 15) are stored on-chip, and any additional `receive_pool_pointers` (16 to 8191) are stored in the Receive Pool Pointers Table in shared memory. When a maximum of 16 receive pools are defined for all active logical links, the MLAPD need not access the external Receive Pool Pointers Table, which provides improved performance. The Receive Pool Pointers Table is shown in Figure 2-12.



NOTE: Logical-link tables shown are for example only.

Figure 2-12. Receive Pool Pointers Table

Any number of logical links may share a receive pool. The host assigns each logical link to a receive pool using the `receive_pool_number` entry in the link's LLT. During the reception process, the MLAPD uses the `receive_pool_number` as an offset into the Receive Pool Pointers Table to locate a free receive buffer. After a frame descriptor is used for frame reception, the MLAPD reads the `next_Rx_frame_descriptor_pointer` entry in the receive frame descriptor and updates the `receive_pool_pointer` entry for this pool in the Receive Pool Pointers Table.

## 2.6 TRANSMIT FRAME DESCRIPTOR

An I or XID/UI frame consists of a frame descriptor and an associated data buffer. The MLAPD also provides the option of configuring the frame as a frame descriptor with an associated header buffer and an associated data buffer. A transmit frame is shown in Figure 2-13.

The transmit frame descriptor format is described in Figure 2-14. In some cases the definition of the various bits in the frame descriptor differ for I frames and XID/UI frames. Also, some entries in the frame descriptor are not used, depending on the frame type.

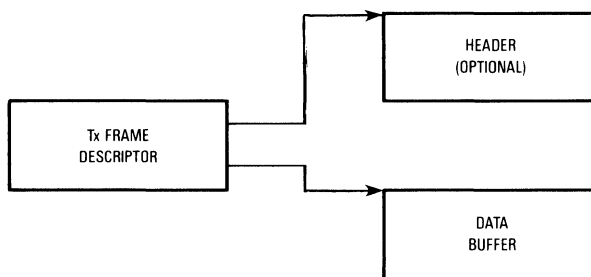


Figure 2-13. Transmit Frame Configuration

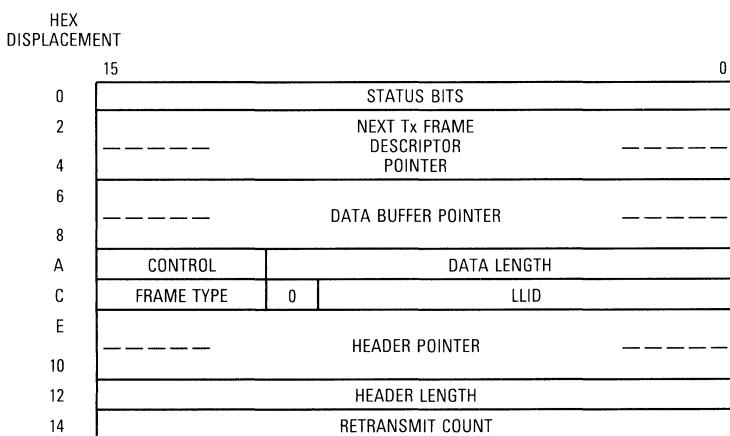


Figure 2-14. Transmit Frame Descriptor

### 2.6.1 Status Bits

The status\_bits entry is written by the MLAPD and read by the host. This word is initialized to zero by the host. The meaning of the status\_bits is slightly different depending on whether the frame descriptor describes an I frame, an XID/UI frame, or a frame for a nonprotocol link.

For an I frame or any frame for a nonprotocol link this word is defined as follows:

Bits 15-3 are not used.

These bits are set to zero by the MLAPD.

Empty (bit 2)

- 1 This link's Transmit Queue has been processed by the MLAPD. This queue contains no frames awaiting transmission. The bit is set in the last frame descriptor when all transmitted frames are acknowledged for protocol links or when all frames are transmitted for nonprotocol links.
- 0 This link's Transmit Queue is still being processed by the MLAPD. Frames are awaiting transmission.

Confirmation (bit 1)

This bit has meaning only for protocol links. For nonprotocol links, this bit is not used and is set to zero by the MLAPD.

- 1 The buffer(s) associated with this frame descriptor has (have) been acknowledged.
- 0 The buffer(s) associated with this frame descriptor has (have) not been acknowledged.

Transmit (bit 0)

- 1 The data in the buffer(s) associated with this frame descriptor has been read and placed into the Tx FIFO for transmission.
- 0 The data in the buffer(s) associated with this frame descriptor has not been read and placed into the Tx FIFO for transmission.

For an XID/UI transmit frame, this word is defined as follows:

Bits 15-3 are not used.

These bits are set to zero by the MLAPD.

Empty (bit 2)

- 1 This XID/UI transmit queue has been handled by the MLAPD. This queue contains no frames awaiting transmission.
- 0 This XID/UI transmit queue is still being processed by the MLAPD. Frames are awaiting transmission.

#### Transmit\_bits (bits 1,0)

- 00 The data in the buffer(s) associated with this frame descriptor has not been read and placed into the Tx FIFO for transmission.
- 01 The data in the buffer(s) associated with this frame descriptor has been read and placed into the Tx FIFO for transmission. This encoding is a positive confirmation.
- 10 This frame has been not transmitted because a DL\_UI, XID, or nonstandard\_control frame may not be transmitted while the logical link is in its present link state (TEI\_UNASSIGN or EST\_WAIT\_TEI). This encoding is a negative confirmation.
- 11 Not a valid encoding.

### 2.6.2 Next Tx Frame Descriptor Pointer

This entry is written by the host and read by the MLAPD. This 32-bit entry contains the address of the next transmit frame descriptor in the linked list.

### 2.6.3 Data Buffer Pointer

This entry is written by the host and read by the MLAPD. This 32-bit entry contains the beginning address of the data buffer. The data buffer may begin on either an odd- or even-byte boundary. The MLAPD will always read both the even byte and the odd byte from the first memory location of the data buffer. The even data byte is discarded when the buffer begins on an odd-byte boundary.

### 2.6.4 Control Bits and Data Length

This entry is written by the host and read by the MLAPD. This word is defined as follows:

#### Last (bit 15)

- 1 This frame descriptor is the last one in this transmit queue.
- 0 This frame descriptor is not the last one in this transmit queue.

#### Header\_valid (bit 14)

- 1 This frame descriptor contains a header buffer, which is transmitted before the data buffer.
- 0 This frame descriptor only contains a data buffer. The header\_pointer entry is not valid.

#### Data\_length (Bits 13-0)

This field contains the length of the associated data buffer in bytes. The data\_length may be an odd number of bytes or an even number of bytes. When the data ends on an odd-byte boundary, the last MLAPD access will be a byte read. The MLAPD does not check the data\_length against the N201\_value associated with this logical link. The host must construct data buffers which satisfy the N201 requirement that was established through parameter negotiation (i.e., header\_length + data\_length ≤ N201). The transmit data\_length may range from 0 to (16K - 1) bytes.

## 2.6.5 Frame Type and LLID

This entry is written by the host and read by the MLAPD. The meaning of the bits in this entry are different depending on whether the frame descriptor belongs to an XID/UI queue or to a Transmit Queue for a nonprotocol link. This entry is not used in I frame descriptors that belong to protocol links.

For a frame in the XID/UI queue, this word is defined as follows:

Frame\_type (bits 15, 14, 13)

- 000 This is a MDL UI frame.
- 001 This is a DL UI frame.
- 010 This is an XID\_command frame.
- 011 This is an XID\_response frame.
- 100 This is a nonstandard\_control\_command frame. The control field encoding is defined by the nonstandard\_control GCB entry, and the P/F bit is set to zero.
- 101 This is a nonstandard\_control\_command frame. The control field encoding is defined by the nonstandard\_control GCB entry, and the P/F bit is set to one.
- 110 This is a nonstandard\_control\_response frame. The control field encoding is defined by the nonstandard\_control GCB entry, and the P/F bit is set to zero.
- 111 This is a nonstandard\_control\_response frame. The control field encoding is defined by the nonstandard\_control GCB entry, and the P/F bit is set to one.

LLID (bits 12-0)

These bits contain the LLID of the link on which the frame is to be transmitted. This link must be assigned as a protocol link. Otherwise, the MLAPD will not transmit the frame and will return a negative\_confirmation.

For any frame that belongs to a nonprotocol link, this word is defined as follows:

Transmit\_CRC\_disable (bit 15)

- 1 The MLAPD will not append a CRC to the end of this frame
- 0 The MLAPD will append a CRC to the end of this frame transmission.

Transmit\_address\_disable (bit 14)

- 1 The MLAPD will not append a DLCI to the beginning of this frame transmission.
- 0 The MLAPD will append the DLCI associated with this logical link to the beginning of this frame transmission. Bit 0 is set to zero and bit 8 is set to one. Bit 1 is set to the inverse of the user/network\_select bit in the DLCI entry in the link's LLT (i.e., treated as a command frame).

Bits 13-0 are not used.

These bits must be set to zero by the host.

### 2.6.6 Header Pointer

The 32-bit header\_pointer is written by the host and read by the MLAPD. When the header\_valid bit is set to one in the control\_bits/data\_length entry, this location contains the address of a header buffer. Otherwise, this entry is not used.

The MLAPD will transmit the header buffer before transmitting the data buffer specified by this frame descriptor. The header buffer must begin on an even-byte boundary.

### 2.6.7 Header Length

When the header\_valid bit is set to one in the control\_bits/data\_length entry, the least significant four bits of this entry contain the length of the associated header buffer. Otherwise, this entry is not used. The header\_length entry is written by the host and read by the MLAPD. Bits 15 to 4 must be set to zero by the host.

The header\_length may be an odd number of bytes or an even number of bytes. When the header\_length is odd, the MLAPD reads both the even and the odd bytes of the last memory location. The even byte is discarded. The MLAPD does not check whether the header\_length plus the data\_length exceeds the N201 value associated with this logical link. The host must construct data buffers which satisfy the N201 requirement that was established through parameter negotiation. The header\_length may range from 0 to 15 bytes. When a header\_length of zero is programmed, the MLAPD will transmit information from the data buffer only.

### 2.6.8 Retransmit Count

This entry is not used for XID/UI frames or for any frames transmitted by nonprotocol links. The retransmit\_count entry in I frame descriptors is written by the MLAPD only when the retransmit\_statistics\_select bit in the option\_bits\_2 GCB entry is set to one. Otherwise, this entry is not used for I frames either.

This 16-bit entry is a wrap around up-counter. The host must initialize this entry to zero. Each time an I frame is retransmitted, the MLAPD increments the retransmit\_count entry by one. If this entry becomes equal to the retransmit\_threshold, the MLAPD places an interrupt on the Interrupt Queue (if this interrupt is not masked). Additional interrupts will occur every (Hex) '10000' retransmissions of the frame.

## 2.7 RECEIVE FRAME DESCRIPTOR

The MLAPD reports information about a received frame via its receive frame descriptor. The format of a receive frame descriptor is described in Figure 2-15.

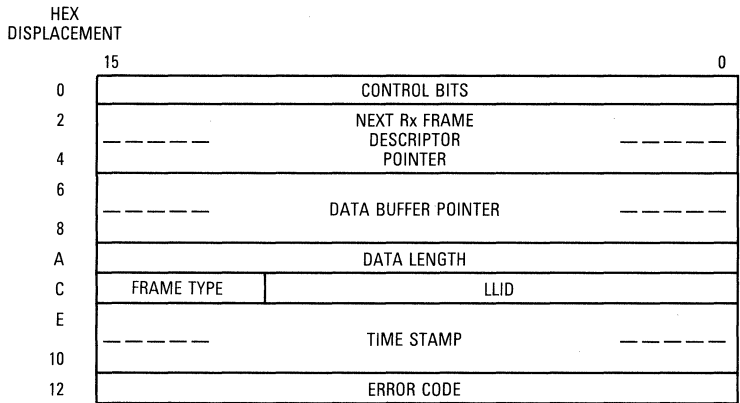


Figure 2-15. Receive Frame Descriptor

### 2.7.1 Control Bits

The control\_bits entry is written by the host and read by the MLAPD. This word is defined as follows:

Bits 15-3 are not used.

These bits must be set to zero by the host.

Data\_indication (bit 2)

- 1 An interrupt is generated when this frame descriptor is used for frame reception. The host may choose to set this bit in the first frame descriptor in the receive pool, so that an interrupt is generated when the receive queue is not empty. The host may alternatively choose to set this bit in each frame descriptor in the receive pool, so that an interrupt is generated each time a frame is received. If the host wishes to service a receive queue after some number of frames have been received, this bit can be set in frame descriptors in the receive pool that are multiples of this number.
- 0 No interrupt indication is given when this frame descriptor is used for frame reception.

Red\_line (bit 1)

This bit can be programmed by the host during the preparation of the receive pool or dynamically after this frame descriptor is already in the pool. During frame reception, the MLAPD reads the next\_receive\_frame\_descriptor\_pointer and checks the red\_line bit in this identified frame descriptor. If this bit is set to one, the MLAPD issues an interrupt. An interrupt is only issued the first time the red\_line bit is checked in a frame descriptor.

The red\_line indication allows the host to avoid a busy condition on a logical link by dynamically adding frame descriptors. The red\_line indication can alternatively



indicate that a frame has been received to an empty queue. See **6.3 COLLECTING RECEIVED FRAMES**.

- 1 The MLAPD issues a red-line interrupt if a frame has been successfully stored into the previous frame descriptor and the red-line bit is set in this frame descriptor, which is now the head of the receive pool. When the MLAPD again accesses this frame descriptor to store a frame, the MLAPD will not issue another red-line interrupt.
- 0 No red-line indication is enabled.

Last-in-pool (bit 0)

- 1 This is the last frame descriptor (the pool-dummy frame descriptor) in the receive pool. The MLAPD will not use this frame descriptor's associated buffer to store data from an incoming frame. The MLAPD will not remove the pool-dummy frame descriptor from the receive pool.
- 0 More frame descriptors are in the receive pool. A buffer is associated with this frame descriptor and can be used by the MLAPD for frame reception.

### 2.7.2 Next Rx Frame Descriptor Pointer

This 32-bit entry is written by the host and read by the MLAPD. This entry points to the next receive frame descriptor in the receive pool. When collecting receive data, the host reads the next\_receive\_frame\_descriptor\_pointer to traverse the receive queue.

### 2.7.3 Data Buffer Pointer

This entry is written by the host and read by the MLAPD. This 32-bit pointer contains the beginning address of the data buffer. Restrictions are placed on receive data buffers: 1) a data buffer must begin on an even-byte boundary; and 2) a data buffer must contain an even number of bytes, even when N201 is odd. In the case of a frame containing an odd number of bytes in the data field, the MLAPD will perform a word write for a 16-bit data bus or two byte writes for an 8-bit data bus to place the last data byte into the data buffer. The data written to the even byte of this last memory location will be invalid and the data\_length entry will not include this even byte as data.

### 2.7.4 Data Length

This 16-bit entry is written by the MLAPD and read by the host. After the associated data buffer is filled with data from an incoming frame, the MLAPD writes the number of bytes used in the data buffer into this data\_length entry. This entry will be less than or equal to the N201\_value associated with this logical link. The host must ensure that the length of each data buffer in a receive pool is equal to or greater than the largest N201 value negotiated for all logical links which share the pool.

## 2.7.5 Frame Type and LLID

This 16-bit entry is written by the MLAPD and read by the host. This entry is initialized by the host when the frame descriptor is added to the receive pool. This entry must be initialized to zero for frame descriptors that will be used by protocol links. The host software then tests for a nonzero value to know that this buffer now contains receive data. For frame descriptors to be used by nonprotocol links, the entry initialization may be handled in one of two ways. If the host initializes this entry to zero, a test for a nonzero value will indicate that the buffer now contains receive data. However, in the case of multibuffer operation, this test will mask the receive data stored in memory from a multibuffer frame that is not completely received. To allow recognition of receive data before a multibuffer frame has ended, the host can initialize this entry to an undefined encoding, such as 010XXXXXXXXXXXXX.

The definition of the bits in this entry differ depending on whether the addressed logical link is assigned to protocol or nonprotocol mode.

For links assigned to protocol mode, these bits are defined as follows:

### Frame\_type (bits 15-13)

- 000 The buffer associated with this frame descriptor does not contain data from an incoming frame. This frame descriptor belongs to the receive pool.
- 001 The data buffer associated with this frame descriptor contains data from an XID command frame. This frame descriptor belongs to the receive queue.
- 010 The data buffer associated with this frame descriptor contains data from an XID response frame. This frame descriptor belongs to the receive queue.
- 011 The data buffer associated with this frame descriptor contains data from an UI frame. This frame descriptor belongs to the receive queue.
- 100 The data buffer associated with this frame descriptor contains data from a numbered I frame. This frame descriptor belongs to the receive queue.
- 101 The data buffer associated with this frame descriptor contains data from a FRMR frame. This frame descriptor belongs to the receive queue.
- 110 The data buffer associated with this frame descriptor was used for reception of a command frame containing the nonstandard\_control field specified in the GCB entry. The buffer contains all information following the control field of the frame and before the closing flag. This frame descriptor belongs to the receive queue.
- 111 The data buffer associated with this frame descriptor was used for reception of a response frame containing the nonstandard\_control field specified in the GCB entry. The buffer contains all information following the control field of the frame and before the closing flag. This frame descriptor belongs to the receive queue.

### LLID (bits 12-0)

These bits specify the LLID associated with the received frame.

For links assigned to nonprotocol mode with the `multibuffer_select` bit in the `option_bits_2` GCB entry set to zero, these bits are defined as follows:

Frame\_type (bits 15-13)

- 000 Undefined. The MLAPD will not write this encoding.
- 001 Undefined. The MLAPD will not write this encoding.
- 010 Undefined. The MLAPD will not write this encoding.
- 011 Undefined. The MLAPD will not write this encoding.
- 100 Undefined. The MLAPD will not write this encoding.
- 101 Undefined. The MLAPD will not write this encoding.
- 110 Undefined. The MLAPD will not write this encoding.
- 111 The data buffer associated with this frame descriptor has been used for reception of an incoming frame. This frame descriptor belongs to the receive queue.

LLID (bits 12-0)

These bits specify the LLID associated with the received frame.

For links assigned to nonprotocol mode with the `multibuffer_select` bit in the `option_bits_2` GCB entry set to one, these bits are defined as follows:

Frame\_type (bits 15-13)

- 000 The data buffer associated with this frame descriptor has been used for reception of an incoming frame. When the MLAPD receives a frame for a nonprotocol link which will not fit into a single receive buffer, the MLAPD marks the buffer 000. As the frame continues to be stored in memory, all additional buffers are marked 000. When the entire frame is received, the last buffer used is marked 011. At that time, the MLAPD returns to the first buffer used for receiving this frame and marks it 100. All other intermediate buffers used for receiving this frame remain marked 000. The data\_length for buffers marked 000 is equal to  $N201\_value - 2$ . (When the MLAPD is operating in line monitor mode. No concept of a frame exists; therefore, all used receive buffers will be marked with this encoding, and the data length will be  $N201\_value - 2$ . See **9.4 LINE MONITOR**).
- 001 Undefined. The MLAPD will not write this encoding.
- 010 Undefined. The MLAPD will not write this encoding.
- 011 This data buffer is the last buffer of a multibuffer frame. The data\_length will be less than or equal to  $N201\_value$ .
- 100 This data buffer is the first buffer of a multibuffer frame. The data\_length will be equal to  $N201\_value - 2$ .
- 101 Undefined. The MLAPD will not write this encoding.
- 110 Undefined. The MLAPD will not write this encoding.
- 111 This data buffer contains an entire frame. The data\_length will be less than or equal to  $N201\_value$ .

LLID (bits 12-0)

These bits specify the LLID associated with the received frame.

### 2.7.6 Time Stamp

This 32-bit entry is written by the MLAPD only when the `promiscuous_receive_select` bit in the `option_bits_1` GCB entry is set to one. Otherwise, this entry is not used. The MLAPD writes the arrival time of the frame into this entry. The time stamp is based on an internal 32-bit counter with a resolution of 16 system clock cycles.

### 2.7.7 Error Code

The `error_code` entry is written by the MLAPD and read by the host. The `error_code` word must be initialized to zero by the host before the frame descriptor is linked to the receive pool. The MLAPD will receive frames with errors for logical links which have the `error_mask_valid` bit set to one in the link's LLT. Frames are received based upon the `protocol_error_mask` GCB entry for links operating in protocol mode, while frames are received based upon the `nonprotocol_error_mask` for links operating in nonprotocol mode. For an erroneous nonprotocol frame, the `error_code` is written in both the first and last buffers of the frame when the `multibuffer_select` option bit is set. If the receive pool becomes empty before a multibuffer frame ends, the `buffer_length_exceeded` error bit is set. The error codes are defined as follows:

Bits 15-4 are not used.

These bits must be set to zero by the host.

`Buffer_length_exceeded` (bit 3)

- 1 This frame violated the N201 value negotiated for this logical link. For nonprotocol links with the multibuffer option enabled, this bit is set if the `receive_pool` becomes empty during reception of this frame.
- 0 This frame did not violate the N201 value for this logical link.

`RxFIFO_overflow` (bit 2)

- 1 This frame caused a receive FIFO overrun.
- 0 This frame did not cause a receive FIFO overrun.

`Abort/nonoctet` (bit 1)

- 1 This frame ended with an abort or this frame was nonoctet aligned.
- 0 This frame ended normally.

`CRC_error` (bit 0)

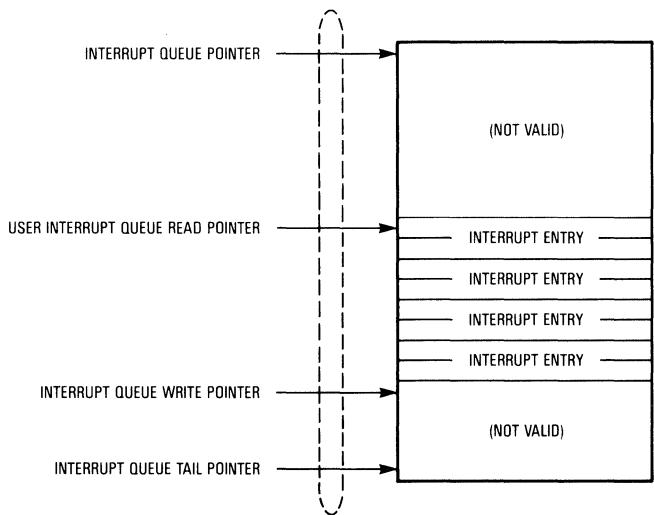
- 1 This frame had a CRC error.
- 0 This frame did not have a CRC error.

## 2.8 INTERRUPT QUEUE

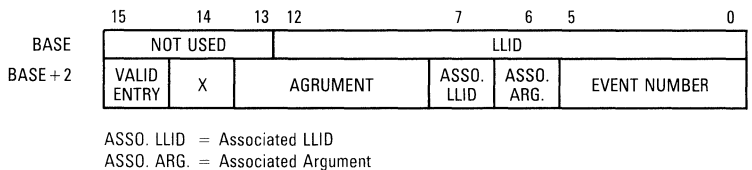
The Interrupt Queue is located in shared memory. The host specifies the head of the queue and the number of available entries via the GCB during initialization. The host must zero

the Interrupt Queue before presenting the area to the MLAPD. The Interrupt Queue structure is shown in Figure 2-16.

The Interrupt Queue provides detailed information on interrupting events to the host. Each entry in the queue consists of two words. The format of an interrupt entry is shown in Figure 2-17.



**Figure 2-16. Interrupt Queue Structure**



**Figure 2-17. Interrupt Queue Entry Format**

The first word of the interrupt entry contains the corresponding LLID, if the interrupt is associated with a specific logical link. Otherwise, the first word of the interrupt entry is not valid. This word is read by the host and written by the MLAPD. The format of the first word is defined as follows:

Bits 15, 14, and 13 are not used.

These bits are set to zero by the MLAPD, if associated\_LLID is set to one. If associated\_LLID is set to zero, bits 15, 14, and 13 are don't care since the entire word is not valid.

LLID (bits 12-0)

These bits contain the LLID associated with the interrupting condition, if applicable.

The second word in an interrupt entry indicates whether the entry contains a valid interrupt condition and, if so, the cause of the pending interrupt.

Valid\_entry (bit 15)

Set to valid by MLAPD, reset to invalid by the host.

- 1 This Interrupt Queue entry defines an interrupt condition not yet handled by the host.
- 0 This Interrupt Queue entry is free.

Bit 14

This bit is a don't care.

Interrupt\_argument (bits 13-8)

For two interrupting conditions, this field is encoded to provide additional information about the cause of the interrupt. See **2.8.2 Interrupt Arguments**.

Associated\_LLID (bit 7)

- 1 The LLID associated with this interrupt is contained in the first word of this interrupt entry.
- 0 No LLID is associated with this interrupt entry. The first word of this interrupt entry is not used.

Associated\_argument (bit 6)

- 1 The argument field in this word is valid.
- 0 The argument field in this word is not valid.

Interrupt\_event\_number (bits 5-0)

This field contains a number which identifies the cause of the interrupt. This number corresponds to the bit location of the interrupt in the 32-bit interrupt\_mask. The list of interrupt conditions is given in **2.8.1 Interrupt Events**.

## 2.8.1 Interrupt Events

The following list specifies all interrupting events and their corresponding interrupt\_event\_number (in binary), which is stored in the Interrupt Queue entry.

- 000000 RxFIFO\_overnrun\_threshold\_reached  
This interrupt indicates that the system bus bandwidth allocated to the MLAPD is not sufficient to handle the receive data activity. This interrupt has no associated LLID.
- 000001 TxFIFO\_underrun\_threshold\_reached  
This interrupt indicates that the system bus bandwidth allocated to the MLAPD for providing data to the transmitter is not sufficient. This interrupt has no associated LLID.
- 000010 Inactive\_DLCL\_threshold\_reached  
This interrupt indicates that the inactive\_DLCL\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 000011 Invalid\_address\_threshold\_reached  
This interrupt indicates that the invalid\_address\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 000100 Discarded\_frame\_threshold\_reached  
This interrupt indicates that the discarded\_frame\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 000101 Short\_frame\_threshold\_reached  
This interrupt indicates that the short\_frame\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 000110 CRC\_error\_threshold\_reached  
This interrupt indicates that the CRC\_error\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 000111 Abort/nonoctet\_threshold\_reached  
This interrupt indicates that the abort/nonoctet\_threshold (specified in the GCB) has been reached. This interrupt has no associated LLID.
- 001000 Link\_error\_threshold\_reached  
This interrupt indicates that a link error threshold (specified in the GCB) has been reached. This interrupt has an associated LLID. The argument field of this interrupt defines the specific link error.
- 001001 L\_frame\_retransmit\_threshold\_reached  
This interrupt indicates that the L\_frame\_retransmit\_threshold (specified in the GCB) has been reached for an L frame. The retransmit\_count entry in the transmit

frame descriptor of the affected I frame is equal to the retransmit-threshold. This interrupt has an associated LLID.

- 001010 CTS-timeout-threshold-reached  
This interrupt indicates that the MLAPD asserted  $\overline{\text{RTS}}$  and waited for  $(\text{CTS-timeout-threshold} \times 2048)$  TxCLK cycles for  $\overline{\text{CTS}}$  to be asserted. Alternatively, this interrupt indicates that the MLAPD has waited for this time interval since the last time this interrupt was issued. This interrupt has no associated LLID entry.
- 001011 CTS-lost  
This interrupt indicates that the  $\overline{\text{CTS}}$  pin was negated during a frame transmission for more than one TxCLK cycle. TxD is three-stated while  $\overline{\text{CTS}}$  is not active. This interrupt is issued only once per frame, regardless of the number of bit times where  $\overline{\text{CTS}}$  is negated. This interrupt has no associated LLID.
- 001100 Pool red-line  
This interrupt indicates that a red line condition has been reached for the receive pool associated with the specified logical link. See **2.7 RECEIVE FRAME DESCRIPTOR**. This interrupt has an associated LLID.
- 001101 CAM-overflow  
This interrupt indicates that the host attempted to activate more than 16 logical links in on-chip system operation mode. This interrupt has no associated LLID.
- 001110 L2-queue-overflow  
This interrupt indicates that the memory allocated to the external Level 2 Queue was not sufficient for storing all the Level 2 frames generated as the result of host commands or received frames. This interrupt has no associated LLID.
- 001111 Undefined/reserved
- 010000 MDL-assign-indication  
This interrupt indicates that the MLAPD is waiting for a DLCI assignment for the specified link for which a service has been requested. This interrupt has an associated LLID.
- 010001 Remote-status-confirmation  
This interrupt indicates that a frame with F set to one has been received in response to a S frame transmitted by the MLAPD with P set to one following a **REMOTE-STATUS-REQUEST** command. Refer to **4.5.3 REMOTE-STATUS-REQUEST Command**. This interrupt has an associated LLID.
- 010010 Local-busy  
This interrupt indicates that a frame was received for the specified logical link, while this logical link was in the local busy condition. See **8.10.1 Local Busy Condition**. This interrupt has an associated LLID.



- 010011 Data\_indication  
This interrupt indicates that a frame was added to a receive queue for the specified link. For a protocol link, specification of the frame type is found in the receive frame descriptor. This interrupt has an associated LLID.
- 010100 DL\_data\_confirmation  
This interrupt indicates that the last frame in the specified logical link's Transmit Queue was transmitted and acknowledged. This interrupt has an associated LLID.
- 010101 XID/UI-Queue\_0\_confirmation  
This interrupt indicates that the last frame in the XID/UI Queue\_0 was transmitted. This interrupt has no associated LLID.
- 010110 XID/UI-Queue\_1\_confirmation  
This interrupt indicates that the last frame in the XID/UI Queue\_1 was transmitted. This interrupt has no associated LLID.
- 010111 Global-XID/UI\_confirmation  
This interrupt indicates that the last frame in the Global XID/UI Queue was transmitted. This interrupt has no associated LLID.
- 011000 DL\_establish\_confirmation  
This interrupt indicates that the specified link was established as the result of a DL\_ESTABLISH\_REQUEST command. This interrupt has an associated LLID.
- 011001 DL\_establish\_indication  
This interrupt indicates that the specified link was established as the result of a SABME frame reception. This interrupt has an associated LLID.
- 011010 DL\_release\_confirmation  
This interrupt indicates that the specified logical link was released as the result of a DL\_RELEASE\_REQUEST command. This interrupt has an associated LLID.
- 011011 DL\_release\_indication  
This interrupt indicates that a DISC command frame was received for the specified link, that the link was released for timeout reasons, or that the link was released as the result of a MDL\_REMOVE\_REQUEST command. This interrupt has an associated LLID.
- 011100 Undefined/reserved
- 011101 MDL\_error\_indication  
This interrupt indicates that an error condition occurred on the specified link. This interrupt has an associated LLID and argument field.

- 011110 Undefined\_host\_command  
This interrupt indicates that the host issued an unimplemented command. This interrupt has no associated LLID.
- 011111 thru 111101 Undefined/reserved
- 111110 Interrupt\_queue\_overflow  
This interrupt indicates that the memory allocated for the Interrupt Queue was not sufficient for storing all pending interrupt requests. This interrupt has no associated LLID. This interrupt is not maskable.
- 111111 Bus/address\_error  
A bus/address error occurred on an MLAPD DMA cycle. Bus/address error operation is described in **7.2 BUS ERROR OPERATION**. This interrupt has no associated LLID. This interrupt is not maskable.  $\overline{IRQ}$  (INTR) will be asserted regardless of the user-programmed value of the polling\_select bit.

## 2.8.2 Interrupt Arguments

Two interrupt conditions, MDL\_error\_indication and link\_error\_threshold\_reached, encode the argument field to provide additional information about the interrupt cause.

**2.8.2.1 MDL ERROR INDICATION ARGUMENTS.** The valid encodings of the interrupt\_argument field and the corresponding approved CCITT error codes are listed below. The indicated link state numbers correspond to the numeric state values defined in Table 8-1.

- 000000 Reception of a host command which is not permitted in the link's current state. (no code defined)
- 000001 Reception of a S frame with F set to one when not allowed. (code A)
- 000010 Reception of a DM frame when not allowed. (code B)
- 000011 Reception of a UA frame when link state is 4, 7, or 8. (code C)
- 000100 Reception of a UA frame with F set to zero when link state is 5 or 6. (code D)
- 000101 Undefined/reserved
- 000110 Reception of a SABME frame when not allowed. (code F)
- 000111 SABME retransmission limit reached. (code G)
- 001000 DISC retransmission limit reached. (code H)

- 001001 Status inquiry retransmission limit reached as part of normal timer recovery. (code I1)
- 001010 Status inquiry retransmission limit reached as part of the T203 status inquiry process. (code I2)
- 001011 N(R) error. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code J)
- 001100 Reception of FRMR when link state is 7 or 8. (code K)
- 001101 Reception of an unimplemented frame. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code L)
- 001110 Reception of an I field when not permitted. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code M)
- 001111 Undefined/reserved
- 010000 Reception of an I frame with N201 error. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code O1)
- 010001 Reception of a XID command frame with N201 error. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code O2)
- 010010 Reception of a XID response frame with N201 error. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code O3)
- 010011 Reception of a UI command frame or a user-defined U frame with N201 error. A FRMR frame was transmitted, if enabled by option\_bits\_1 entry in GCB. (code O4) Or reception of a frame with an N201 error to a nonprotocol link.
- 010100 Reception of a XID command frame in one of the following cases:
1. When the link state is 7.2
  2. When the link state is 8.2
  3. When the link state is 4, 5, or 6 AND the link is in local busy
- In all cases the received frame is discarded. (code P2)
- 010101 Reception of a XID response frame in one of the following cases:
1. When the link state is 7.2
  2. When the link state is 8.2
  3. When the link state is 4, 5, or 6 AND the link is in local busy
- In all cases the received frame is discarded. (code P3)

010110 Reception of a UI command frame or a user-defined U frame in one of the following cases:

1. When the link state is 7.2
2. When the link state is 8.2
3. When the link state is 1, 3, 4, 5, or 6 AND the link is in local busy. Also, reception of a frame for a nonprotocol link while the link is in local busy. In all cases the received frame is discarded. (code P4)

010111 Reception of a FRMR response frame in one of the following cases:

1. When the link state is 7.2
2. When the link state is 8.2

According to the LAPD protocol, FRMR is considered a bad frame in states 1, 3, 4, 5, and 6. Also, issuing a SET\_LOCAL\_BUSY command in any connected state places the link in state 7.2 or 8.2.

In all cases the received frame is discarded. (code P5)

011000 thru 111111 Undefined/reserved

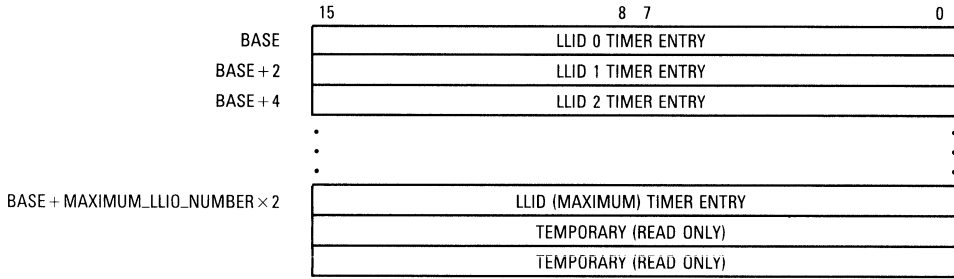
**2.8.2.2 LINK COUNTER THRESHOLD REACHED ARGUMENTS.** The valid encodings of the interrupt\_argument field for the link\_error\_threshold\_reached interrupt are as follows:

- 000000 REJ transmit counter reached its threshold.
- 000001 RNR transmit counter reached its threshold.
- 000010 REJ receive counter reached its threshold.
- 000011 RNR receive counter reached its threshold.
- 000100 thru 111111 Undefined/reserved.

## 2.9 TIMER TABLE

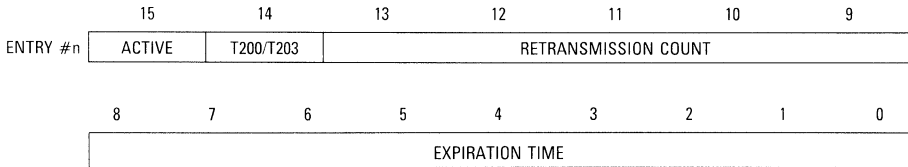
The Timer Table is read and written by the MLAPD. The MLAPD uses this table together with two free-running on-chip counters to implement the Level 2 timers: T200 and T203. Only one timer per logical link is active at any given time. The timers are maintained for protocol links in AWAIT\_EST (state 5), AWAIT\_REL (state 6), MF\_EST\_NORM (state 7.0), MF\_EST\_REJ (state 7.1), MF\_EST\_BUSY (state 7.2), TM\_REC\_NORM (state 8.0), TM\_REC\_REJ (state 8.1), and TM\_REC\_BUSY (state 8.2). Refer to the LAPD state tables for details of when each timer is (re)started and stopped. Timers are not maintained for non-protocol links. The Timer Table format is shown in Figure 2-18.

The Timer Table begins at the address specified by the timer\_table\_pointer. The host must allocate enough memory for this table to accommodate a 16-bit entry for each LLID that may be assigned by the host plus two additional words, which are used by the MLAPD to store temporary values. The host must zero the Timer Table before presenting the area to the MLAPD. The LLID multiplied by two is the index into the table.



**Figure 2-18. Timer Table Format**

Each entry in the Timer Table contains the timer status for a specific logical link. The format of each entry is defined as follows:



**Active (bit 15)**

- 1 This entry contains an active timer.
- 0 This entry has no active timer.

**T200/T203 (bit 14)**

- 1 This logical link timer is T200. (acknowledgement)
- 0 This logical link timer is T203. (idle)

**Retransmission\_count (bits 13-9)**

Stores the number of retransmissions of the current transmit frame for this logical link. The MLAPD increments the retransmission\_count according to the LAPD protocol and compares this value to the GCB parameter, N200\_value.

**Expiration\_time (bits 8-0)**

These bits store the "time", with respect to an on-chip timer, when this logical link's timer expires.

**2.9.1 Timer Operations**

Three timer operations are performed on a per link basis to implement the T200 and T203 protocol functions: disable timer, (re)start timer, and timeout recognition.

The disable and (re)start operations are performed frequently, according to the link's activity. These operations require up to three memory cycles. To disable a timer, the MLAPD sets the active bit to zero in the appropriate entry. To start a timer for a logical link, the present value of the corresponding on-chip counter is added to the link's timeout period. This expiration\_time is written in the appropriate table entry, along with the timer type (T200/T203). The retransmission\_count is also zeroed as specified by the LAPD protocol in conjunction with this timer operation, and the active bit is set to one. The T203 timeout period is a global parameter contained in the T203\_value entry in the GCB. The T200 timeout is specified on a per link basis and is contained in the T200\_value entry in the link's LLT.

A timeout recognition operation is performed at every time step. The following example shows the time step for T200 and T203, based upon the system clock frequency:

$$\begin{aligned} \text{T200 time step} &= (2^{20})/F \\ \text{T203 time step} &= (2^{20} \times 10)/F \\ F &\text{ is the system clock frequency.} \end{aligned}$$

F (MHz)	T200 Time Step (milliseconds)	T203 Time Step (milliseconds)
16.67	62.91	629.1
12.5	83.88	838.8
10.0	104.85	1048.5
8.0	131.07	1310.7

During timeout recognition, all active timer entries are read from memory and compared to the present value of the corresponding on-chip counter. If the expiration\_time entry for a logical link and the on-chip counter have the same value, then the logical link's timer has expired. When a retransmission occurs as result of timeout recognition, the retransmission\_count is incremented by one and compared to the maximum number of retransmissions (N200\_value).

The MLAPD keeps track of the highest active LLID assigned by the host. During a timeout recognition operation, the MLAPD only checks the Timer Table entries up to the highest active LLID. By assigning the lowest available LLID to a link during link establishment, the system bus loading for timer handling is minimized. Also, because a complete Timer Table check is performed once per timer resolution (70 to 130 milliseconds depending on the system clock frequency), the recognition of a timeout may be delayed by almost a full time step when many LLIDs are active.

During a timer recognition operation two entries are read during each DMA burst. For 8K active logical links, the microcode is busy with the timer recognition operation approximately 40-45% of the time. In the remaining 55-60%, the microcode is available to handle command execution and service the serial link. The amount of microcode time for timer maintenance increases and decreases linearly with respect to the number of active LAPD links.

## 2.10 LEVEL 2 QUEUE

The MLAPD generates all Level 2 LAPD control frames (see Figure 2-19). As these frames are generated, the MLAPD queues them for transmission on the Level 2 Transmit Queue. Level 2 frames must be queued for transmission, since the MLAPD is not always able to immediately begin transmission. The first 32 (on-chip operation mode) or 48 (expanded operation mode) words of the Level 2 Queue are located on-chip, and additional storage is allocated by the host in external memory. When an overflow condition occurs on the internal queue, the MLAPD starts filling the external queue. Once the external Level 2 Queue is emptied, the MLAPD resumes use of the internal queue.

The Level 2 Queue is completely managed by the MLAPD and is transparent to the user. The host is notified, though, when the MLAPD overflows the allocated external Level 2 queue area. It is recommended that this external area be at least 128 words in length to avoid a Level 2 Queue overflow condition. However, the LAPD procedure allows for easy recovery from an overflow since this situation equates to the MLAPD's transmission of Level 2 frames, which were lost on the link.

During initialization the MLAPD fetches the L2-queue-pointer and the L2-queue-length from the GCB. The beginning address of the external queue is stored in the 32-bit L2-queue-pointer register. The value  $(L2\text{-queue-length} - 5) \times 2$  are stored in the L2-tail-displacement register. The L2-tail-displacement register then indicates to the last entry that can be used for a Level 2 frame, since the largest Level 2 Queue entry consists of five words (FRMR frame). The MLAPD manages this external queue by using the L2-read-displacement register and the L2-write-displacement register, which are 16-bit offsets from the L2-queue-pointer register. The internal queue is managed in the same manner as the external Level 2 Queue, using the L2-int-read-displacement register and the L2-int-write-displacement register. These six registers may be inspected for diagnostic purposes by issuing a DUMP command.

The Level 2 Queue structure is shown in Figure 2-20. Each entry in the Level 2 Queue consists of the control field of the frame followed by the DLCI. If the Level 2 frame is an FRMR frame, the DLCI is then followed by the FRMR I field.

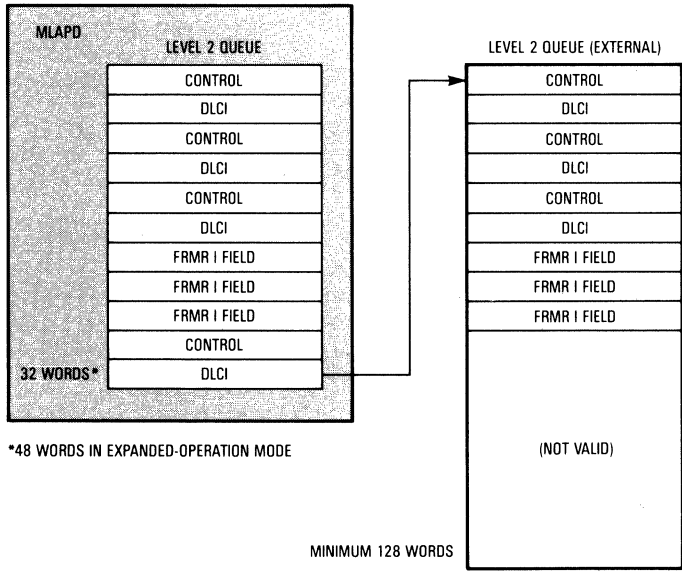
### SUPERVISORY (S) FRAMES

RECEIVE READY (RR)  
RECEIVE-NOT-READY (RNR)  
REJECT (REJ)

### UNNUMBERED (U) FRAMES

SET ASYNCHRONOUS BALANCED MODE EXTENDED (SABME)  
DISCONNECT (DISC)  
DISCONNECTED MODE (DM)  
UNNUMBERED ACKNOWLEDGEMENT (UA)  
FRAME REJECT (FRMR)

**Figure 2-19. MLAPD Generated Level 2 Frames**



\*48 WORDS IN EXPANDED-OPERATION MODE

Figure 2-20. Level 2 Queue Structure



## SECTION 3 INTERNAL REGISTERS

The MLAPD has 7 functional blocks: serial, direct memory access (DMA), microcontroller, arithmetic logic unit (ALU), register file, random access memory (RAM), and content addressable memory (CAM). Each of these sections contain user visible and nonvisible registers that define and control the operation of the MLAPD. Some of these registers contain global information for all logical links. This information permanently resides on-chip after initialization. Other registers contain information for the specific link currently receiving service by the MLAPD receiver or transmitter. The information in these registers is moved into and out of the chip each time the in-service link changes.

Because the MLAPD communicates with the host primarily through shared memory structures, a minimum number of host directly accessible registers are required. Most of the MLAPD registers are visible to the user but are not directly accessible. Commands issued to the MLAPD cause the MLAPD to load internal registers from the Global Configuration Block (GCB) and Logical-Link Tables (LLTs) or to write the updated register values into the GCB and LLTs. Most of the internal registers are cleared (set to zero) during the reset sequence. The indirectly addressed registers may be global or per link, and constant or variable.

### 3.1 DIRECTLY ACCESSIBLE REGISTERS

The directly accessible registers include the command register (CR), semaphore register (SR), data register (DR), and interrupt-vector register (IVR). A register map is shown in Table 3-1 for Motorola configuration and in Table 3-2 for Intel-compatible configuration.

#### 3.1.1 Command Register

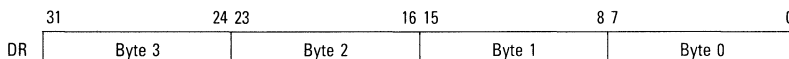
The control interface between the MLAPD and the host processor is the CR. This 8-bit register is written by the host processor to issue commands to the MLAPD. Once a command is issued, it must be completed (or accepted) by the MLAPD before the next command can be written to the CR. The SR indicates when a new command may be issued. Passing the MLAPD an undefined command causes an undefined-host-command interrupt to be placed on the Interrupt Queue.

The command set is detailed in **SECTION 4 COMMAND SET**. Some commands need additional arguments, which are written into the argument fields in the GCB by the host prior to issuing a command.

**Table 3-1. Motorola Bus Selection of Directly Accessible Registers**

A2	A1	UDS/A0	LDS/DS	D8-D15	D0-D7
<b>16-Bit Bus</b>					
0	0	X	0	X	CR, SR
0	1	X	0	X	IVR
1	0	0	0	DR (High Word)	
1	0	0	1	DR Byte 3	X
1	0	1	0	X	DR Byte 2
1	1	0	0	DR (Low Word)	
1	1	0	1	DR Byte 1	X
1	1	1	0	X	DR Byte 0
<b>8-Bit Bus</b>					
0	0	X	0	X	CR, SR
0	1	X	0	X	IVR
1	0	0	0	X	DR Byte 3
1	0	1	0	X	DR Byte 2
1	1	0	0	X	DR Byte 1
1	1	1	0	X	DR Byte 0

X-DON'T CARE Condition



### 3.1.2 Semaphore Register

The MLAPD uses the 8-bit SR to indicate execution of the current host command. When a command is written to the CR, the MLAPD indicates a busy condition by setting this register to the hex value 'FE'. Upon completion (or acceptance) of the command, the SR is set to the hex value 'FF'. The host processor must read the SR to ensure that it is hex 'FF' before modifying the command\_argument entries or issuing the next command. Failure to do so will cause unpredictable results. As an exception to the above statement, a software RESET command can be issued even when the SR is hex 'FE'.

The MLAPD also reports the completion of a hardware reset by setting the SR to the hex value 'FF'. The worst-case delay for a command to begin execution (other than RESET, INIT, and DUMP) is approximately 400 system clock cycles.

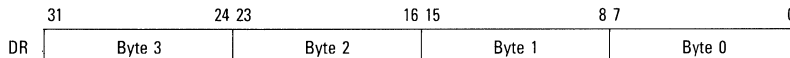
### 3.1.3 Interrupt Vector Register

The IVR contains the 8-bit interrupt vector number presented to the system during an interrupt acknowledge cycle. The upper seven bits of the vector are user programmable. The least significant bit is generated internally by the MLAPD to provide a unique interrupt

**Table 3-2. Intel-Compatible Bus Selection of Directly Accessible Registers**

A2	A1	A0	$\overline{\text{BHE}}$	D8-D15	D0-D7
<b>16-Bit Bus</b>					
0	0	0	X	X	CR, SR
0	1	0	X	X	IVR
1	0	0	0	DR (Low Word)	
1	0	0	1	X	DR Byte 0
1	0	1	0	DR Byte 1	X
1	1	0	0	DR (High Word)	
1	1	0	1	X	DR Byte 2
1	1	1	0	DR Byte 3	X
<b>8-Bit Bus</b>					
0	0	0	X	X	CR, SR
0	1	0	X	X	IVR
1	0	0	X	X	DR Byte 0
1	0	1	X	X	DR Byte 1
1	1	0	X	X	DR Byte 2
1	1	1	X	X	DR Byte 3

X—DON'T CARE Condition



vector number corresponding to the interrupt type. Normal and severe interrupt sources are defined in **2.8.1 Interrupt Events**. The interrupt conditions which cause a severe interrupt are bus error and address error. The interrupt vector encodings are shown in the following table.

**IVR (BIT 0)**

Normal Interrupt	0
Severe Interrupt	1

The MLAPD exits reset in the interrupt-driven mode with the IVR set to hex '0F'. The hex '0F' encoding is defined in the M68000 architecture as an uninitialized interrupt vector. In an Intel-compatible configuration, this encoding may be defined as a system reserved vector. Therefore, the IVR must be loaded during the initialization procedure with a user specified value to avoid a possible conflict of system bus interrupt usage.

**3.1.4 Data Register**

The 32-bit DR is written by the host during initialization with the address of the GCB. As part of the INIT command, the MLAPD transfers the value in the DR into the global\_configuration\_block\_pointer register.

## 3.2 INDIRECTLY ACCESSIBLE REGISTERS

The DUMP command allows the user to view many indirectly accessible, internal MLAPD registers. During execution of the DUMP command, the MLAPD writes these internal registers and CAM into a user-specified dump area in memory. The format of the dump area is described in **4.2.2 DUMP Command**. The definitions of these internal registers and CAM are contained in this section. The ordering of this section directly corresponds to the ordering of the dump area map. Some locations in the dump area contain temporary values. These temporary values are not defined. The dump area also contains the internal Level 2 Queue which is described in **2.10 LEVEL 2 QUEUE**. The dump area also contains receive pool pointers 0-15 which are described in **2.5 RECEIVE POOL POINTERS TABLE**.

### 3.2.1 CAM

The on-chip content addressable memory (CAM) has 16 entries, where each entry stores a DLCI-LLID pair. The CAM is used only when the on-chip system operation mode is selected. All the entries are cleared by hardware or software reset. A new entry is written into the CAM as part of the MDL\_ASSIGN\_REQUEST command, if a free entry exists. If there is no free entry, a CAM\_overflow interrupt is generated. An entry is removed from the CAM by the MLAPD as part of the DEACTIVATE\_LL command execution.

The CAM identifies the Logical-Link Identification (LLID) number associated with the Data-Link-Connection Identifier (DLCI) in the incoming frame. If the CAM is enabled, an incoming DLCI is presented to the CAM and a *hit* or *miss* indication is generated. In the case of a hit, the LLID is returned. A miss indicates that the DLCI is unassigned, and the frame is ignored.

Each CAM entry consists of two words. The first word of the word pair is defined as follows:

15	14	13	12	0
VALID CAM ENTRY	LAST CAM ENTRY	0	DLCI	

Valid\_CAM\_entry (bit 15)

- 1 A valid DLCI-LLID pair is stored in this entry.
- 0 This CAM entry is free. No DLCI-LLID pair is stored in this entry.

Last\_CAM\_entry (bit 14)

- 1 This is the last entry in the CAM.
- 0 This CAM entry is not the last one (the sixteenth entry).

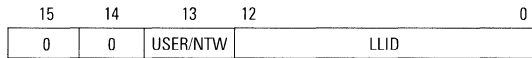
Bit 13

This bit is always set to zero.

DLCI (bits 12-0)

This field contains a valid DLCI when the valid\_CAM\_entry bit is set to one.

The second word of the CAM word pair is defined as follows:



Bits 15 and 14

These bits are always set to zero.

User/Network (bit 13)

- 1 The logical link associated with this LLT is on the “network” side. The C/R bit is set to one in transmitted command frames and to zero in transmitted response frames.
- 0 The logical link associated with this LLT is on the “user” side. The C/R bit is set to zero in transmitted command frames and to one in transmitted response frames.

LLID (bits 12-0)

This field contains the LLID associated with the DLCI in this CAM entry when the valid\_CAM\_entry bit is set to one.

### 3.2.2 N200 Value

The five bits (13–9) of this 16-bit register contain the user-specified number of retransmission attempts for a supervisory (S) command frame or an unnumbered (U) command frame (N200). Bits 15 and 14 are set to one. Bits 8 and 0 are set to zero. This register is initialized during INIT execution with the corresponding GCB value and is a global definition for all logical links. The MLAPD will update this register during execution of a RELOAD command with a new user-defined value.

### 3.2.3 T203 Value

Bits 8-0 of this 16-bit register define the maximum time that an established link may be inactive (carry no frames). This register is initialized during INIT execution with the corresponding GCB value, which is a global definition for all links. The MLAPD will update this register during execution of a RELOAD command with a new user-defined value.

### 3.2.4 Highest Active LLID

The least significant 13 bits of this 16-bit register contain the highest LLID which the host has assigned to any LAPD link in numeric link state 5 or higher. This register is initialized to zero during INIT execution. The highest\_active\_LLID register is used by the MLAPD when performing a timeout recognition operation. The MLAPD updates this register as necessary when a link makes a transition from a numeric link state less than 5 to numeric link state 5 or higher and when a link makes a transition from numeric link state 5 or higher to a numeric link state less than five.

### 3.2.5 T200 Counter

The MLAPD uses this 9-bit counter (bits 8–0) together with the external Timer Table to implement the individual T200 link acknowledgment timers for each LAPD link in numeric link states 5 and higher. Bits 15–9 are set to 1100000. The counter is set to zero after a hardware or software reset. Every  $2^{20}$  master clocks, the T200-counter is incremented. So for a 16.67 MHz clock, this counter is incremented every 63 milliseconds with a maximum count of 32 seconds.

### 3.2.6 T203 Counter

The MLAPD uses this 9-bit counter (bits 8–0) together with the external Timer Table to implement the activity timer T203 for each active logical link. Bits 15–9 are set to 1000000. The counter is set to zero after a hardware or software reset. Every  $10 \times 2^{20}$  master clocks, the T203-counter is incremented. For example, with a 16.67 MHz clock, this counter is incremented every 0.63 seconds, with a maximum count of 322 seconds. The T200 counter is scaled by 10 to give the proper range for the T203 counter.

### 3.2.7 Pool Number

This 16-bit register contains the receive-pool-number for the link addressed in the current/last receive frame. The MLAPD uses this number to fetch the corresponding receive-pool-pointer from the Receive Pool Pointers Table during a receive operation. This register is initialized to zero during INIT execution. This register is updated as a frame is received.

### 3.2.8 L2 Tail Displacement

This 16-bit register is an offset from the Level 2 (L2)-queue-pointer register. The L2-tail-displacement register identifies the last entry which may be used by the MLAPD in the external Level 2 Queue. This memory location has the address  $(L2\text{-queue-pointer} + (L2\text{-queue-length} - 5) \times 2)$ . This register is initialized during INIT execution.

### 3.2.9 External L2 Queue Displacement Registers

The MLAPD uses two 16-bit registers to manage the external Level 2 Queue: the L2-read-displacement register and the L2-write-displacement register. These registers are byte offsets from the L2-queue-pointer register. The L2-read-displacement register points to the next Level 2 frame to be transmitted, and the L2-write-displacement register points to the next location where a Level 2 frame will be stored. When these displacements are equal, the external Level 2 Queue is empty, and the MLAPD begins using the internal Level 2 Queue area. These registers are initialized to zero during INIT execution.

### 3.2.10 Tx XID/UI LLID

The 13 least significant bits of this 16-bit register contain the LLID of the current/last logical link serviced by the MLAPD transmitter while processing frames from either the Global XIC/UI Queue, XID/UI Queue<sub>0</sub>, or XID/UI Queue<sub>1</sub>. Bits 15-13 are not used and always set to zero. This register is initialized to zero during INIT execution. This register is updated as frames are transmitted from one of the XID/UI queues.

### 3.2.11 Tx XID/UI DLCI

This 16-bit register contains the DLCI of the current/last logical link serviced by the MLAPD transmitter while processing the Global XID/UI Queue. The format of this register is as follows:

- Bits 15-13 Always set to zero
- Bits 12-7 SAPI/DLCI's upper field
- Bits 6-0 TEI/DLCI's lower field

This register is initialized to zero during INIT execution. This register is updated as frames are transmitted from the Global XID/UI Queue.

### 3.2.12 Internal L2 Queue Displacement Registers

The MLAPD uses two 16-bit registers to manage the internal Level 2 Queue: the L2\_int\_write\_displacement and the L2\_int\_read\_displacement. These registers are offsets from the beginning of the internal Level 2 Queue area. These registers range from 0–31 in on-chip system operation mode and from 0–47 in expanded-system operation mode. The L2\_int\_read\_displacement register points to the next Level 2 frame to be transmitted, and the L2\_int\_write\_displacement register points to the next location where a Level 2 frame will be stored. When these pointers are equal, the internal Level 2 Queue is empty. When the L2\_int\_write\_displacement is greater than the L2\_int\_read\_displacement, then the internal Level 2 Queue is full, and the MLAPD will begin to use the external Level 2 Queue. These registers are initialized to zero during INIT execution.

### 3.2.13 Tx FD Status

This 16-bit register contains the status word of the current or last transmit frame descriptor (FD) serviced by the MLAPD transmitter.

### 3.2.14 Tx V(S)/V(R)

This 16-bit register contains the send and receive state variables for the current or last logical link serviced by the MLAPD transmitter. The send state variable, V(S), denotes the

sequence number of the next information (I) frame to be transmitted. This variable is stored in bits 9–15. Bit 8 is always set to zero. The receive state variable,  $V(R)$ , denotes the expected sequence number of the next I frame to be received. This variable is stored in bits 7-1. Bit 0 is always set to zero. The  $Tx\_V(S)/V(R)$  value is initialized by the MLAPD during link establishment. The LLT entry is updated whenever this value changes.

### 3.2.15 CTS Timeout Counter

The least significant 8 bits of this 16-bit counter are used by the MLAPD to measure the time from when request-to-send ( $\overline{RTS}$ ) is asserted until clear-to-send ( $\overline{CTS}$ ) is asserted. This counter is initialized during INIT execution with the corresponding GCB threshold entry and is reinitialized each time  $\overline{RTS}$  is asserted. Following the assertion of  $\overline{RTS}$ , the MLAPD decrements this counter every 2048 transmit clock (TxCLK) cycles. When this counter reaches zero, the MLAPD issues a  $CTS\_timeout\_threshold\_reached$  interrupt, reloads the counter from the  $CTS\_timeout\_threshold$  register, and begins counting again. Once  $\overline{CTS}$  is asserted, the MLAPD reinitializes the counter to its threshold value.

### 3.2.16 Tx Maximum Number of Outstanding I Frames (K)

The lower byte of this 16-bit register defines the maximum number of I frames that can be transmitted for the current/last link serviced by the MLAPD transmit task before an acknowledgment is required. Bits 15-8 are not used and set to zero. This register is initialized to zero during INIT execution. This register is updated when the link in-service changes.

### 3.2.17 Tx V(A)

The most significant 7 bits of this 16-bit register contain the acknowledge state variable,  $V(A)$ , for the current or last logical link serviced by the MLAPD transmitter.  $V(A)$  identifies the last I frame that has been acknowledged. Bits 8-0 are always set to zero. This register is initialized to zero during link establishment and is updated as each valid acknowledgment is received for the associated link. The link's LLT entry is updated whenever this value changes.

### 3.2.18 Tx Link Status

This 16-bit register contains the  $link\_status$  LLT entry of the current/last logical link serviced by the MLAPD transmitter. This register is initialized to zero during INIT execution and is updated as each frame is transmitted, if necessary. The corresponding LLT entry is updated whenever the link status changes.

### 3.2.19 Current Queue In-Service

The three low-order bits of this 16-bit register specify which of the six lowest-priority transmit queues is to be serviced next by the MLAPD transmit task. This register is incremented modulo 6. This register is initialized to zero during hardware or software reset;



therefore after reset, I frame Queue\_0 is serviced before the other queues (but after the Level 2 Queue and the Global XID/UI Queue, as usual). The current\_queue-in-service register encodings correspond to the six queues as follows:

000	I Queue_0
001	I Queue_1
010	I Queue_2
011	I Queue_3
100	XID/UI Queue_0
101	XID/UI Queue_1
110	Not used
111	Not used

### 3.2.20 Scan Length Counter

This 16-bit counter counts the number of frames to be transmitted from the XID/UI or I frame queue currently receiving service by the MLAPD transmitter. The counter is set to the user specified scan\_length for the current transmit queue when service begins and is decremented as each frame is transmitted. When the last frame in the current transmit queue is transmitted or when the scan\_length\_counter reaches zero, the MLAPD automatically begins servicing the next XID/UI or I frame queue according to the transmit servicing scheme explained in **5.1 TRANSMIT SERVICING SCHEME**.

### 3.2.21 Tx I LLID

The 13 least significant bits of this 16-bit register contain the LLID of the current/last logical link which received I frame service by the MLAPD transmitter. Bits 15-13 are not used and always set to zero. This register is initialized to zero during INIT execution. This register is updated when the link receiving I frame transmit service changes.

### 3.2.22 Tx I DLCI

This 16-bit register contains the DLCI of the current/last logical link which received I frame service by the MLAPD transmitter (right justified). The format of this register is as follows:

- Bits 15-13 Always set to zero
- Bits 12-7 SAPI/DLCI's upper field
- Bits 6-0 TEI/DLCI's lower field

### 3.2.23 Retransmit Threshold

Bits 7-0 of this 16-bit register contain the user-specified retransmission limit for I frames which causes an interrupt to be generated. This register is initialized during INIT execution with the corresponding GCB value. The MLAPD will update this register during execution of a RELOAD command with a new user-defined value.

### 3.2.24 Rx Address

This 16-bit register contains the address field for the current/last received frame. This register is initialized to zero during INIT execution; it is updated as each incoming frame is received.

### 3.2.25 Rx N201

This 16-bit register contains the maximum data field length (N201) allowed for the current/last received frame. This register is initialized to zero during INIT execution. This register is updated from the addressed link's LLT as each incoming frame is received.

### 3.2.26 Rx Link Status

This 16-bit register contains the link\_status LLT entry for the logical link addressed by the current/last received frame. This register is initialized to zero during INIT execution and is updated on each incoming frame, if necessary. The corresponding LLT entry is updated whenever the link status changes.

### 3.2.27 Rx LLID

The 13 low-order bits of this 16-bit register contain the LLID of the current/last logical link that received a frame. If bit 15 is zeroed, the logical link specified by this LLID is still in an Bits 7-0 of this 16-bit register contain the user-specified retransmission limit for I frames last link to receive a frame, and no other logical link was in the process of receiving a frame at the time that the DUMP command was performed. This register is initialized to zero during INIT execution. This register is updated as each incoming frame is received.

### 3.2.28 Rx DLCI

The 13 low-order bits of this 16-bit register contain the DLCI of the current/last receive frame (right justified). The format of this register is as follows:

- Bits 15-13 Always set to zero
- Bits 12-7 SAPI/DLCI's upper field
- Bits 8-0 TEI/DLCI's lower field

The Rx-DLCI is used in the DLCI-LLID match operation with the internal CAM in the on-chip operation mode and with the external Match Table in the expanded operation mode. This register is initialized to zero during INIT execution and is updated as each incoming frame is received.

### 3.2.29 Rx Frame ID

This 16-bit register contains the frame\_type and LLID for the current/last receive frame which is written into the frame\_type and LLID entry of the receive frame descriptor. This

register is initialized to zero during INIT execution. This register is updated as each frame is received.

### 3.2.30 Tx Address

This 16-bit register contains the address field for the current/last transmitted frame. This register is initialized to zero during INIT execution and is updated as each frame is transmitted.

### 3.2.31 Time Stamp

This 16-bit register contains the most significant 16-bits of a 32-bit counter. This counter is initialized to zero during a hardware or software reset. This 32-bit counter is incremented every  $2^{20}$  system clock cycles.

### 3.2.32 RxFIFO Overrun Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a receive first-in first-out (RxFIFO) overrun occurs, this counter is decremented. Upon reaching zero, a `RxFIFO_overrun_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a `PRESET_STATISTICS` command.

### 3.2.33 TxFIFO Underrun Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a transmit FIFO (TxFIFO) underrun occurs, this counter is decremented. Upon reaching zero, a `TxFIFO_underrun_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a `PRESET_STATISTICS` command.

### 3.2.34 Invalid Address Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a LAPD frame is received with an invalid address, this counter is decremented. Upon reaching zero, an `invalid_address_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a `PRESET_STATISTICS` command.

### 3.2.35 Inactive DLCI Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a frame is received for an inactive DLCI, this counter is decremented.

Upon reaching zero, an inactive-DLCL-threshold-reached interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a PRESET-STATISTICS command.

### 3.2.36 Discarded Frame Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a receive frame is discarded due to lack of receive buffers, this counter is decremented. Upon reaching zero, a discarded-frame-threshold-reached interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a PRESET-STATISTICS command.

### 3.2.37 Short Frame Counter

This 16-bit counter is initialized to the corresponding GCB entry during execution of the INIT command. When a LAPD frame is received with less than five bytes between flags, this counter is decremented. Upon reaching zero, a short-frame-threshold-reached interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a PRESET-STATISTICS command.

### 3.2.38 CRC Error Counter

This 16-bit counter is initialized to the CRC-error-threshold during execution of the INIT command. When a frame is received with a cyclical redundancy check (CRC) error, this counter is decremented. Upon reaching zero, a CRC-error-threshold-reached interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a PRESET-STATISTICS command.

### 3.2.39 Abort/Nonoctet Counter

This 16-bit counter is initialized to the abort/nonoctet-threshold during execution of the INIT command. When a receive frame contains an abort or is nonoctet aligned, this counter is decremented. Upon reaching zero, an abort/nonoctet-error-threshold-reached interrupt is placed on the Interrupt Queue, and the counter is reinitialized. This counter is also reinitialized as the result of a PRESET-STATISTICS command.

### 3.2.40 Tx REJ/RNR Counter

The upper 8 bits of this 16-bit register count the number of transmitted reject (REJ) frames for the current/last link receiving service by the MLAPD transmit task. This counter is initialized to zero during INIT execution. The counter is loaded with the current link's

corresponding LLT entry when transmit servicing begins and is decremented when a REJ frame is transmitted. Upon reaching zero, a `link_error_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized with the `REJ_transmit_threshold` GCB entry. The LLT entry is updated when this register value changes.

The lower 8 bits of this 16-bit register count the number of transmitted receive-not-ready (RNR) frames for the current/last link receiving service by the MLAPD transmit task. This counter is initialized to zero during INIT execution; it is loaded with the current link's corresponding LLT entry when transmit servicing begins. This counter is decremented when a REJ frame is transmitted. Upon reaching zero, a `link_error_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized with the `RNR_transmit_threshold` GCB entry. The LLT entry is updated when this register value changes.

### 3.2.41 Rx REJ/RNR Counter

The upper 8 bits of this 16-bit register count the number of received REJ frames for the current/last link addressed by an incoming receive frame. The counter is initialized to zero during INIT execution and is loaded with the addressed link's corresponding LLT entry as an incoming frame is received. This counter is decremented when a REJ frame is received. Upon reaching zero, a `link_error_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized with the `REJ_receive_threshold` GCB entry. The LLT entry is updated when this register value changes.

The lower 8 bits of this 16-bit register count the number of received RNR frames for the current/last link addressed by an incoming receive frame. The counter is initialized to zero during INIT execution and is loaded with the addressed link's corresponding LLT entry as an incoming frame is received. This counter is decremented when a RNR frame is received. Upon reaching zero, a `link_error_threshold_reached` interrupt is placed on the Interrupt Queue, and the counter is reinitialized with the `RNR_receive_threshold` GCB entry. The LLT entry is updated when this register value changes.

### 3.2.42 CTS Timeout Threshold

The least significant 8 bits of this 16-bit register allow the user to program the time period, after asserting `RTS`, that the MLAPD waits for `CTS` to be asserted before issuing an interrupt. The time period is determined by  $(\text{CTS\_timeout\_threshold} \times 2048)$  TxCLK cycles. The register is initialized during INIT execution to the corresponding GCB value. The MLAPD updates the register during execution of a RELOAD command with a new user-defined value.

### 3.2.43 Scan Length I Queues 0-3

These four 16-bit registers contain the maximum number of I frames to be transmitted from the corresponding I frame Queues\_0-3 before switching to service the next queue

according to the transmit servicing scheme. These registers are initialized during INIT execution with the corresponding GCB value. The MLAPD updates these registers during execution of a RELOAD command with the new user-defined values.

#### **3.2.44 Scan Length XID/UI Queues 0, 1**

These two 16-bit registers contain the maximum number of XID/UI frames to be transmitted from the corresponding XID/UI Queues 0 and 1 before switching to service the next queue according to the transmit servicing scheme. These registers are initialized during INIT execution with the corresponding GCB value. The MLAPD updates these registers during execution of a RELOAD command with the new user-defined values.

#### **3.2.45 Interrupt Mask**

This 32-bit register allows the user to selectively disable interrupts for specific events. Clearing the appropriate bit in this mask prevents the specific event or condition from being reported via the Interrupt Queue. Figure 2-7 describes the interrupt mask bits. This register is initialized during INIT execution with the corresponding GCB value. The MLAPD updates the register during execution of a RELOAD command with the new user-defined value.

#### **3.2.46 Protocol Error Mask**

This 16-bit register contains the user-specified receive error mask for LAPD links. The host may individually enable reception of frames with any of the four invalid frame conditions on a per logical link basis. The register is initialized during INIT execution with the corresponding GCB value. The MLAPD will update this register during execution of a RELOAD command with the new user-defined value.

#### **3.2.47 Nonprotocol Error Mask**

This 16-bit register contains the user-specified receive error mask for nonprotocol links. The host may individually enable reception of frames with any of the four invalid frame conditions on a per logical-link basis. The register is initialized during INIT execution with the corresponding GCB value. The MLAPD will update this register during execution of a RELOAD command with the new user-defined value.

#### **3.2.48 Rx Maximum Number of Outstanding Frames or Filter Mask (Word 1)**

If `promiscuous_receive_select` and `filter_select` bits in `option_bits_1` are set to one, then this entry contains the GCB `filter_mask` word 1. In this case, the register is initialized during

INIT execution to filter\_mask word 1. Otherwise, this 16-bit register contains the maximum number of outstanding frames (K) for the logical link addressed by the incoming receive frame. The register is initialized during INIT execution to zero and is updated for each receive frame from the link's LLT entry.

### 3.2.49 Rx V(A) or Filter Mask (Word 2)

If promiscuous\_receive\_select and filter\_select bits in option\_bits\_1 are set to one, then this entry contains the GCB filter\_mask word 2. In this case, the register is initialized during INIT execution to filter\_mask word 2. Otherwise, the most significant 7 bits of this 16-bit register contain the acknowledge state variable, V(A), for the logical link addressed by the current/last incoming frame. V(A) identifies the last frame that has been acknowledged. In this case, bits 8-0 are always set to zero. This register is updated as each valid acknowledgment is received for the associated link. The link's LLT entry is updated whenever this value changes. This register is initialized to zero during INIT execution.

### 3.2.50 Rx V(S)/V(R) or Filter Match (Word 1)

If promiscuous\_receive\_select and filter\_select bits in option\_bits\_1 are set to one, then this entry contains the GCB filter\_match word 1. In this case, the register is initialized during INIT execution to filter\_match word 1. Otherwise, this 16-bit register contains the send and receive state variables, for the current or last logical link addressed by an incoming frame. The send state variable, V(S), denotes the sequence number of the next frame to be transmitted. This variable is stored in bits 15-9. Bit 8 is always zero. The receive state variable, V(R), denotes the expected sequence number of the next I frame to be received. This variable is stored in bits 7-1. Bit 0 is always zero. This register is initialized to zero during INIT execution. The corresponding LLT entry is updated whenever this value changes.

### 3.2.51 Rx Control or Filter Match (Word 2)

If promiscuous\_receive\_select and filter\_select bits in option\_bits\_1 are set to one, then this entry contains the GCB filter\_match word 2. In this case, this register is initialized during INIT execution to filter\_match word 2. Otherwise, this 16-bit register contains the control field of an incoming receive frame. The register is initialized to zero during INIT execution and is updated as each incoming frame is received.

### 3.2.52 Tx LLT Transmit Status

This 16-bit register contains the transmit\_status LLT entry for the current/last logical link serviced by the MLAPD transmitter. The register is initialized to zero during INIT execution and is updated whenever the link's transmit status changes. The corresponding LLT entry is also updated whenever this value changes.

### 3.2.53 Nonstandard Control Field

This 16-bit register contains the user-defined nonstandard\_control field, which is loaded from the corresponding GCB entry during INIT execution. This register is updated during RELOAD execution with the new user-defined value.

### 3.2.54 Revision Number

This 16-bit register contains the revision number of this silicon. Bits 15–8 are not used and set to zero. Bits 7–0 are encoded to indicate the silicon revision. This register is initialized during INIT execution.

### 3.2.55 Tx Next Pointer

This register contains 1) the 32-bit address of the transmit I frame descriptor with the data buffer currently being transmitted or 2) the address of the transmit I frame descriptor with the data buffer that will be transmitted the next time the logical link is serviced. This register is initialized to zero during INIT execution and is updated as each I frame is transmitted. The MLAPD also updates the corresponding LLT entry when this register changes.

### 3.2.56 Tx LLT Pointer

This 32-bit register stores the address of the LLT for the current/last link receiving I frame service by the MLAPD transmit task. This register is initialized to zero during INIT execution and is updated when the link receiving I frame service by the MLAPD transmit task changes.

### 3.2.57 Global XID/UI Head Pointer

This 32-bit pointer identifies 1) the frame descriptor in the Global XID/UI Queue with the data buffer currently being transmitted or 2) the frame descriptor with the data buffer that will be transmitted when the Global XID/UI Queue is next serviced. This register is initialized to zero during INIT execution and is updated when a GLOBAL\_XID/UI\_REQUEST command is accepted and after a queued frame descriptor is handled.

### 3.2.58 I Queue 0-3 Head Pointer Registers

Each of the four 32-bit I\_Queue\_0-3\_head\_pointer registers identify a linked list of logical links' Transmit Queues that have I frames to be transmitted. The I\_Queue\_0-3\_head\_pointer registers actually contain the address of the LLT of the first logical link in the corresponding queue which has pending I frames. These registers are initialized to zero during INIT execution. A register is updated when the identified link's Transmit Queue servicing is completed and the link's Transmit Queue is removed from its I frame queue.



### 3.2.59 I Queue 0-3 Tail Pointer Registers

The end of the four transmit I frame Queues 0-3 are identified by the four 32-bit L-Queue-0-3-tail-pointer registers, respectively. The L-Queue-0-3-tail-pointer registers actually contain the address of the LLT of the last logical link in the corresponding queue which has pending I frames. These registers are initialized to zero during INIT execution. When a DL\_DATA\_REQUEST command is executed, the MLAPD updates the appropriate tail pointer register to link the logical link into its I frame queue, unless a link condition (such as P-flag, remote-busy, etc.) prevents the linking. In this case, the register would be updated when the link condition clears. Also, the register may be updated at any time when a logical link in an I frame queue must be removed due to a link condition and then later relinked to the queue.

### 3.2.60 L2 Queue Pointer

This 32-bit register contains the beginning address of the external Level 2 Queue. This register is initialized during INIT execution with the corresponding GCB value and is considered a constant.

### 3.2.61 Match Table Pointer

The 32-bit match-table-pointer register contains the address of the Match Table in shared memory. This register is used only in expanded-system operation mode and is initialized during INIT execution with the corresponding GCB value. The register is considered a constant.

### 3.2.62 LLID-LLT Table Pointer

This 32-bit pointer identifies the LLID-LLT Table which contains the address of the LLT associated with each active link. This register is initialized during INIT execution with the corresponding GCB value and is considered a constant.

### 3.2.63 Timer Table Pointer

This 32-bit register contains the address of the first word of the Timer Table in shared memory. The register, which is considered a constant, is initialized during INIT execution with the corresponding GCB value.

### 3.2.64 Interrupt Queue Tail Pointer

This 32-bit register points to the last entry in the Interrupt Queue. Each Interrupt Queue entry is two words in length. During execution of the INIT command the value of

{interrupt\_queue\_pointer + (4 × interrupt\_queue\_length) – 2 words} is loaded into the interrupt\_queue\_tail\_pointer register. The register is considered a constant.

### 3.2.65 Interrupt Queue Pointer

3

This 32-bit pointer identifies the beginning address of the Interrupt Queue in shared memory. Together with the interrupt\_queue\_length, this register defines the limits of the circular queue which is used to pass activity information from the MLAPD to the host. Initialized during INIT execution with the corresponding GCB value, this register is considered a constant.

### 3.2.66 Interrupt Queue Write Pointer

This 32-bit pointer indicates the next entry in the Interrupt Queue to be written by the MLAPD. This register contains the address of the second word of the interrupt queue entry that was last written. When the interrupt queue entry located at the end of the Interrupt Queue has been written, identified by the interrupt\_queue\_tail\_pointer, and another interrupt condition occurs, the interrupt\_queue\_write\_pointer register is loaded with the beginning Interrupt Queue address to implement a circular queue. The register is initialized to the interrupt\_queue\_tail\_pointer during INIT execution.

### 3.2.67 First Rx FD Pointer

This 32-bit register contains the address of the first receive frame descriptor used to store an incoming frame in multibuffer operation. The register is initialized to zero during INIT execution. This register is updated as an incoming frame is received for a nonprotocol link for which multibuffer operation is enabled by the multibuffer\_select bit in the option\_bits\_2 GCB entry.

### 3.2.68 GCB Pointer

This 32-bit register contains the address of the Global Configuration Block (GCB). This register is loaded from the data register during execution of the INIT command and is considered a constant.

### 3.2.69 Temporary Rx FD Pointer

This 32-bit register contains the address of the receive frame descriptor used to store an incoming frame in multibuffer operation. This register is initialized to zero during INIT execution. This register is updated as an incoming frame is received for a nonprotocol link for which multibuffer operation is enabled by the multibuffer\_select bit in the option\_bits\_2 GCB entry.

### 3.2.70 Rx LLT Pointer

This 32-bit register contains the address of the LLT for the link addressed by the current/last incoming receive frame. This register is initialized to zero during INIT execution; it is updated on each incoming receive frame, if necessary.

### 3.2.71 Rx Pool Pointer

This 32-bit register holds the address of the receive\_pool\_pointer entry for the link addressed by the current/last incoming receive frame. The MLAPD uses this pointer to locate the first free data buffer in the pool. This register is initialized to zero during INIT execution and is updated as a frame is received, if necessary.

### 3.2.72 Rx Next Acknowledge Pointer

This register contains the 32-bit address of the next transmit I frame descriptor to be acknowledged for the logical link addressed by the incoming receive frame. This register is initialized to zero during INIT execution and is updated when an acknowledgement is received. The MLAPD also updates the corresponding LLT entry when this register changes.

### 3.2.73 Rx Next Tx Pointer

This register contains the 32-bit address of the next transmit I frame descriptor to be transmitted for the logical link addressed by the incoming receive frame. This register is initialized to zero during INIT execution and is updated when an incoming frame is received. The MLAPD also updates the corresponding LLT entry when this register changes.

### 3.2.74 Rx Current FD Pointer

This 32-bit register contains the address of the current/last receive frame descriptor used by the MLAPD for incoming data. This register is initialized to zero during INIT execution and is updated as each incoming frame is received.

### 3.2.75 Rx Next FD Pointer

This 32-bit register contains the address of the next receive frame descriptor in the linked receive pool following the frame descriptor identified by the Rx\_current\_FD register. This register is initialized to zero during INIT execution and is updated as each incoming frame is received.

### 3.2.76 Pool Table Pointer

This 32-bit register contains the address of the first `receive_pool_pointer` in the Receive Pool Pointers Table in shared memory. The MLAPD stores the `receive_pool_pointers` for receive pools 0–15 on chip. The first `receive_pool_pointer` in the external memory table corresponds to pool 16. This register is initialized during INIT execution with the corresponding GCB value and is considered a constant.

### 3.2.77 XID/UI Queue 0 Head Pointer

This 32-bit pointer identifies 1) the frame descriptor in XID/UI Queue\_0 with the data buffer currently being transmitted or 2) the frame descriptor with the data buffer that will be transmitted when XID/UI Queue\_0 is next serviced. This register is initialized to zero during INIT execution and is updated when a XID/UI\_QUEUE\_0\_REQUEST command is accepted and when a queued frame descriptor is handled.

### 3.2.78 XID/UI Queue 1 Head Pointer

This 32-bit pointer identifies 1) the frame descriptor in XID/UI Queue\_1 with the data buffer currently being transmitted or 2) the frame descriptor with the data buffer that will be transmitted when XID/UI Queue\_1 is next serviced. The register is initialized to zero during INIT execution. This register is updated when a XID/UI\_QUEUE\_1\_REQUEST command is accepted and when a queued frame descriptor is handled.

### 3.2.79 Tx XID/UI LLT Pointer

This 32-bit register contains the address of the LLT associated with the current/last logical link to receive XID/UI frame service by the MLAPD transmitter from XID/UI Queue\_0 or \_1. This register is initialized to zero during INIT execution and is updated as each frame is transmitted, if necessary.

### 3.2.80 Tx XID/UI FD Pointer

This 32-bit register contains the address of the frame descriptor associated with the current/last logical link to receive XID/UI frame service by the MLAPD transmitter from XID/UI Queue\_0 or \_1. This register is initialized to zero during INIT execution and is updated as each frame is transmitted.

### 3.2.81 Bus/Address Error Pointers

When a bus/address error occurs, the MLAPD stores the `DMA_Rx_data_pointer` register, `DMA_Tx_data_pointer` register, and the two `DMA_general_purpose_pointer` registers into

the corresponding bus/address error pointer registers. No indication is provided as to which of these registers contains the address driven on the address bus during the bus/address error cycle. These register values may be larger by one (8-bit data bus) or by two (16-bit data bus) than the actual address which was driven on the address bus, due to an automatic post-incrementing mechanism. These registers are initialized to zero during INIT execution.

### **3.2.82 DMA Rx Data Pointer**

This 32-bit register contains the addresses used by the MLAPD to write receive data buffers. When this register is written to the dump area, the register value may be larger by one (8-bit data bus) or by two (16-bit data bus) than the actual address which was last driven on the address bus, due to an automatic post-incrementing mechanism. This register is not initialized.

### **3.2.83 DMA Tx Data Pointer**

This 32-bit register contains the addresses used by the MLAPD to read transmit data buffers. When this register is written to the dump area, the register value may be larger by one (8-bit data bus) or by two (16-bit data bus) than the actual address which was last driven on the address bus, due to an automatic post-incrementing mechanism. This register is not initialized.

### **3.2.84 DMA Rx Data Counter**

This 16-bit register counts the number of bytes written to the location beginning with the DMA\_Rx\_data\_pointer register. This register is not initialized.

### **3.2.85 DMA Tx Data Counter**

This 16-bit register counts the number of bytes read from the location beginning with the DMA\_Tx\_data\_pointer register. This register is not initialized.

### **3.2.86 DMA General-Purpose Pointers**

These two 32-bit registers contain the addresses used by the MLAPD to write the various shared memory tables and frame descriptors. These registers are not initialized.



## SECTION 4 COMMAND SET

The host processor instructs the MLAPD to perform various operations by writing a command into the MLAPD command register (CR). Some commands also require command arguments, which must be written into the command\_argument fields in the Global Configuration Block (GCB) before the command is issued.

The commands fall into the following five categories:

- 1) Initialization
- 2) Test/Diagnostics
- 3) Host – MLAPD Interface
- 4) Protocol
- 5) Protocol Extension

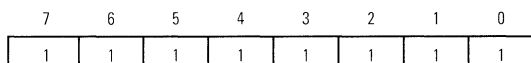
Upon reception of a command from the host processor, the MLAPD sets the semaphore register (SR) to hex 'FE'. After command execution or command acceptance, depending on the specific command, the MLAPD sets the SR to hex 'FF', indicating that it is ready to receive the next command from the host. The host processor must read the SR before writing the next command or modifying any command argument to ensure that the new command will not interfere with the execution of the previous command. When the MLAPD receives an undefined command, an undefined\_host\_command interrupt is issued if this interrupt is not masked.

### 4.1 INITIALIZATION

Initialization commands configure the MLAPD for operation after a hardware or software reset. The initialization commands specify certain system attributes and the location of the GCB in memory. These commands must only be issued as part of the MLAPD initialization procedure following reset.

#### 4.1.1 RESET Command

The format of the RESET command is shown below:



The RESET command or hardware reset causes the following actions:

The receive (Rx) channel is reset.

The transmit (Tx) channel is reset, request-to-send ( $\overline{\text{RTS}}$ ) is negated, and transmit data (TxD) is three-stated.

The system bus is relinquished immediately.

The interrupt\_vector register is set to 'OF' hex.

The data bus width is set to eight bits.

The SR is set to 'FF' hex when reset is completed.

#### 4.1.2 SET\_BUS\_WIDTH\_8 Command

The format of the SET\_BUS\_WIDTH\_8 command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0

The SET\_BUS\_WIDTH\_8 command sets the data bus width to 8 bits. The SR is set to 'FF' hex when the command is completed.

#### 4.1.3 SET\_BUS\_WIDTH\_16 Command

The format of the SET\_BUS\_WIDTH\_16 command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

The SET\_BUS\_WIDTH\_16 command sets the data bus width to 16 bits. The SR is set to 'FF' hex when the command is completed.

#### 4.1.4 INIT Command

The format of the initialization (INIT) command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0

The user issues this command after RESET and the appropriate SET\_BUS\_WIDTH command. The INIT command assumes that the host has previously written the address of the GCB into the data register (DR). During INIT execution, the MLAPD moves the value in the DR into the global\_configuration\_block\_pointer register. The MLAPD then loads its internal



registers corresponding to the entries in the constants area, reloadable variables area, and statistics threshold area of the GCB. In addition, the INIT command causes the MLAPD to:

- Clear all content addressable memory (CAM) entries,
- Clear all other internal registers, and
- Set the highest\_active\_LLID (Logical-Link Identification number) register to zero.

The SR is set to 'FF' hex when the command is completed.

## 4.2 TEST/DIAGNOSTIC

These commands allow the user to test, debug, and diagnose the system. The user must only issue the test/diagnostic commands when the MLAPD is in the off-line state. If these commands are issued in the on-line state, the results are unpredictable. If these commands are issued in the bus/address error state, they are ignored.

### 4.2.1 DMA\_TEST Command

The format of the (direct memory access) DMA\_TEST command is shown below:

7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	0

The DMA\_TEST command requires a length value (odd or even) in the command\_argument\_1 field, a source address in the command\_argument\_2 field, and a destination address in the command\_argument\_3 field in the GCB. For a 16-bit data bus, the MLAPD requires an even source and destination address. For an 8-bit data bus, no restriction is made on the source and destination address.

The DMA\_TEST command tests the handling of parallel data in the logical configuration shown in Figure 4-1. The MLAPD reads data beginning at the memory location specified by the source address. The number of data bytes to be read is specified by the length value. The MLAPD then writes this data back into memory, beginning at the location specified by the destination address. The MLAPD transfers this data via the DR, without using the internal receive or transmit FIFOs (RxFIFO or TxFIFO). The serial link is not affected by this operation, except that no DMA services are provided for the serial link until the DMA transfer is completed.

Upon complete of the data transfer, the MLAPD sets the SR to hex 'FF'. To determine whether the DMA\_TEST was successful, the user should compare the source data with the destination data.

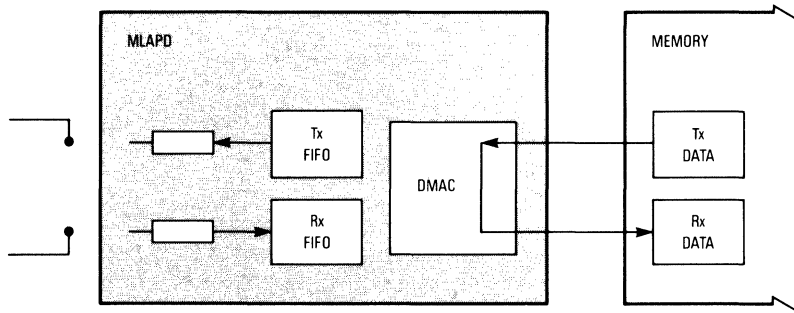


Figure 4-1. DMA Transfer Configuration

### 4.2.2 DUMP Command

The format of the DUMP command is shown below:

7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	0

The DUMP command requires a 32-bit dump area address to be specified in the command\_argument\_2 field of the GCB.

The DUMP command writes internal MLAPD registers and CAM into the dump area in external memory. The location of the dump area is specified by the user in the command argument. The dump area is 650 bytes in length. The ordering of the registers in the dump memory area is shown in Figure 4-2. Detailed descriptions of the registers and the CAM format are found in **3.2 INDIRECTLY ACCESSIBLE REGISTERS**. The SR is set to hex 'FF' when the command is completed.

HEX DISPLACEMENT	8	7	0
0	INTERNAL LEVEL 2 QUEUE		
•			
•			
•			
40	CAM — (DLCIs) OR INTERNAL LEVEL 2 QUEUE		
•	N200 VALUE		
60			
62	T203 VALUE		
64	HIGHEST ACTIVE LLID		
66	TEMPORARY		
68	TEMPORARY		
6A	TEMPORARY		

Figure 4-2. Dump Registers Memory Map (Sheet 1 of 5)

HEX DISPLACEMENT	15	8	7	0
6C				T200 COUNTER
6E				T203 COUNTER
70				POOL NUMBER
72				L2 TAIL DISPLACEMENT (EXTERNAL)
74				L2 WRITE DISPLACEMENT (EXTERNAL)
76				L2 READ DISPLACEMENT (EXTERNAL)
78				Tx XID/UI LLID
7A				Tx XID/UI DLCI
7C				INTERNAL L2 WRITE DISPLACEMENT
7E				INTERNAL L2 READ DISPLACEMENT
80				Tx FD STATUS
82				TEMPORARY
84				CAM — (LLIDs)
.				
.				
A4				Tx V(S)/V(R)
A6				CTS TIMEOUT COUNTER
A8				Tx MAXIMUM NUMBER OF OUTSTANDING FRAMES (K)
AA				Tx V(A)
AC				Tx LINK STATUS
AE				CURRENT QUEUE IN-SERVICE
B0				SCAN LENGTH COUNTER
B2				Tx I LLID
B4				Tx I DLCI
B6				RETRANSMIT THRESHOLD
B8				Rx ADDRESS
BA				Rx N201
BC				Rx LINK STATUS
BE				Rx LLID
C0				Rx DLCI
C2				Rx FRAME ID
C4				Tx ADDRESS
C6				TIME STAMP (HIGH WORD)
C8				RxFIFO OVERRUN COUNTER
CA				TxFIFO UNDERRUN COUNTER
CC				INVALID ADDRESS COUNTER
CE				INACTIVE DLCI COUNTER
D0				DISCARDED FRAME COUNTER
D2				SHORT FRAME COUNTER
D4				CRC ERROR COUNTER
D6				ABORT/NOOCTET COUNTER
D8				Tx REJ/RNR COUNTER
DA				Rx REJ/RNR COUNTER
DC				CTS TIMEOUT THRESHOLD
DE				SCAN LENGTH I QUEUE_0
E0				SCAN LENGTH I QUEUE_1
E2				SCAN LENGTH I QUEUE_2
E4				SCAN LENGTH I QUEUE_3
E6				SCAN LENGTH XID/UI QUEUE_0

Figure 4-2. Dump Registers Memory Map (Sheet 2 of 5)

HEX  
DISPLACEMENT

8 7

0

E8	SCAN LENGTH XID/UI QUEUE_1	
EA	-----	-----
EC	INTERRUPT MASK	
EE	PROTOCOL ERROR MASK	
F0	NONPROTOCOL ERROR MASK	
F2	Rx MAXIMUM NUMBER OF OUTSTANDING FRAMES (K) OR FILTER MASK (WORD 1)	
F4	Rx V(A) OR FILTER MASK (WORD 2)	
F6	Rx V(S)/V(R) OR FILTER MATCH (WORD 1)	
F8	Rx CONTROL OR FILTER MATCH (WORD 2)	
FA	Tx LLT TRANSMIT STATUS	
FC	NONSTANDARD CONTROL	
FE	REVISION NUMBER	
100	TEMPORARY	
.		
.		
114	-----	-----
	Tx NEXT POINTER	
116		
118	-----	-----
	Tx LLT POINTER	
11A		
11C	-----	-----
	GLOBAL XID/UI HEAD POINTER	
11E		
120	-----	-----
	I QUEUE_0 HEAD POINTER	
122		
124	-----	-----
	I QUEUE_0 TAIL POINTER	
126		
128	-----	-----
	I QUEUE_1 HEAD POINTER	
12A		
12C	-----	-----
	I QUEUE_1 TAIL POINTER	
12E		
130	-----	-----
	I QUEUE_2 HEAD POINTER	
132		
134	-----	-----
	I QUEUE_2 TAIL POINTER	
136		
138	-----	-----
	I QUEUE_3 HEAD POINTER	
13A		
13C	-----	-----
	I QUEUE_3 TAIL POINTER	
13E		
140	-----	-----
	L2 QUEUE POINTER	
142		
144	-----	-----
	MATCH TABLE POINTER	
146		

Figure 4-2. Dump Registers Memory Map (Sheet 3 of 5)

HEX DISPLACEMENT	15	8 7	0
148	-----	LLID-LLT TABLE POINTER	-----
14A	-----	TIMER TABLE POINTER	-----
14C	-----	TIMER TABLE POINTER	-----
14E	-----	INTERRUPT QUEUE TAIL POINTER	-----
150	-----	INTERRUPT QUEUE TAIL POINTER	-----
152	-----	INTERRUPT QUEUE POINTER	-----
154	-----	INTERRUPT QUEUE POINTER	-----
156	-----	INTERRUPT QUEUE WRITE POINTER	-----
158	-----	INTERRUPT QUEUE WRITE POINTER	-----
15A	-----	INTERRUPT QUEUE WRITE POINTER	-----
15C	-----	FIRST Rx FD POINTER (MULTIBUFFER OPERATION)	-----
15E	-----	FIRST Rx FD POINTER (MULTIBUFFER OPERATION)	-----
160	-----	GCB POINTER	-----
162	-----	GCB POINTER	-----
164	-----	TEMPORARY Rx FD POINTER (MULTIBUFFER OPERATION)	-----
166	-----	TEMPORARY Rx FD POINTER (MULTIBUFFER OPERATION)	-----
168	-----	Rx LLT POINTER	-----
16A	-----	Rx LLT POINTER	-----
16C	-----	Rx POOL POINTER	-----
16E	-----	Rx POOL POINTER	-----
170	-----	Rx NEXT ACKNOWLEDGE POINTER	-----
172	-----	Rx NEXT ACKNOWLEDGE POINTER	-----
174	-----	Rx NEXT Tx POINTER	-----
176	-----	Rx NEXT Tx POINTER	-----
178	-----	Rx CURRENT FD POINTER	-----
17A	-----	Rx CURRENT FD POINTER	-----
17C	-----	Rx NEXT FD POINTER	-----
17E	-----	Rx NEXT FD POINTER	-----
180	•	Rx POOL POINTER TABLE (RECEIVE POOL POINTERS 0-15)	•
	•		•
	•		•
	•		•
1C0	-----	POOL TABLE POINTER	-----
1C2	-----	POOL TABLE POINTER	-----
1C4	-----	XID/UI QUEUE_0 HEAD POINTER	-----
1C6	-----	XID/UI QUEUE_0 HEAD POINTER	-----
1C8	-----	XID/UI QUEUE_1 HEAD POINTER	-----
1CA	-----	XID/UI QUEUE_1 HEAD POINTER	-----
1CC	-----	Tx XID/UI LLT POINTER	-----
1CE	-----	Tx XID/UI LLT POINTER	-----

Figure 4-2. Dump Registers Memory Map (Sheet 4 of 5)

HEX DISPLACEMENT	15	8	7	0
1D0	-----	TEMPORARY		-----
1D2				
1D4	-----	Tx XID/UI FD POINTER		-----
1D6				
1D8	-----	BUS/ADDRESS ERROR GENERAL-PURPOSE POINTER		-----
1DA				
1DC	-----	BUS/ADDRESS ERROR GENERAL-PURPOSE POINTER		-----
1DE				
1E0	-----	BUS/ADDRESS ERROR DMA Rx DATA POINTER		-----
1E2				
1E4	-----	BUS/ADDRESS ERROR DMA Tx DATA POINTER		-----
1E6				
1E8		TEMPORARY		
.				
.				
.				
1FE				
200	-----	DMA Rx DATA POINTER		-----
202				
204	-----	DMA Tx DATA POINTER		-----
206				
208		DMA Rx DATA COUNTER		
20A		DMA Tx DATA COUNTER		
20C				
20E	-----	DMA GENERAL-PURPOSE POINTER		-----
210				
212	-----	DMA GENERAL-PURPOSE POINTER		-----
214				
.				
.				
27E		TEMPORARY REGISTERS		

Figure 4-2. Dump Registers Memory Map (Sheet 5 of 5)

### 4.3 HOST — MLAPD INTERFACE

These commands are issued by the host to instruct the MLAPD to perform operations not explicitly named in the link access procedure (LAPD) protocol, but which are required by the host to interface with the MLAPD.

#### 4.3.1 OFF-LINE Command

The format of the OFF-LINE command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0

The OFF-LINE command transfers the MLAPD to the off-line state in which the MLAPD will only execute commands. All other MLAPD activities, (e.g., transmitting, receiving, timeout checking, etc.) are disabled.

When the MLAPD is in the on-line state, this command places the MLAPD in a state appropriate for debugging. When the MLAPD is in the bus/address error state (after a bus/address error has occurred), this command notifies the MLAPD that the host has recognized the error. The SR is set to hex 'FF' when the command is completed.

#### 4.3.2 ON-LINE Command

The format of the ON-LINE command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	1

The ON-LINE command transfers the MLAPD to the on-line state. In this state, the MLAPD will execute host commands, receive frames, and transmit frames. This command is used to transfer the MLAPD from the off-line state to a state where the full functionality of the MLAPD is available for processing the serial bit stream. The SR is set to hex 'FF' when the command is completed.

#### 4.3.3 RELOAD Command

The format of the RELOAD command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0

The RELOAD command causes the MLAPD to load its internal registers from the corresponding entries in the reloadable variables area of the GCB. This command allows the host to dynamically change parameters, such as scan\_length or interrupt\_mask, without stopping the MLAPD on-line operation. The SR is set to hex 'FF' when the command is accepted. The actual reloading of each parameter occurs the next time the parameter is used by the MLAPD. The following list details when the new value is loaded.

Pad Time Select

Before the next frame transmission begins

Nonstandard Control

The next time a nonstandard\_control frame is transmitted or received

T203 Value

The next time the T203 timer is (re)started for any link

I Frame Retransmit Threshold

The next time an information (I) frame is transmitted

N200 Value

The next time a supervisory (S) or unnumbered (U) frame is retransmitted

REJ Transmit Threshold

After this threshold is reached for any logical link

RNR Transmit Threshold

After this threshold is reached for any logical link

REJ Received Threshold

After this threshold is reached for any logical link

RNR Received Threshold

After this threshold is reached for any logical link

CTS Timeout Threshold

The next time clear-to-send ( $\overline{\text{CTS}}$ ) is negated

Scan Length I Queue 0

The next time this queue is serviced

Scan Length I Queue 1

The next time this queue is serviced

Scan Length I Queue 2

The next time this queue is serviced



Scan Length I Queue 3  
 The next time this queue is serviced

Scan Length XID/UI Queue 0  
 The next time this queue is serviced

Scan Length XID/UI Queue 1  
 The next time this queue is serviced

Interrupt Mask  
 The next time an interrupt occurs

Protocol Error Mask  
 The next time an error occurs on a protocol link

Nonprotocol Error Mask  
 The next time an error occurs on a nonprotocol link

Filter Mask  
 The next receive frame

Filter Match  
 The next receive frame

**4.3.4 DUMP\_STATISTICS Command**

The format of the DUMP\_STATISTICS command is shown below:

7	6	5	4	3	2	1	0
0	0	1	0	0	0	1	1

This command requires a dump address in the command\_argument\_2 field of the GCB.

The DUMP\_STATISTICS command causes the MLAPD to write the internal global statistics counters to the memory location specified by the pointer in command\_argument\_2. The order in which these counters are written into memory is the same order as their corresponding threshold values are specified in the statistics threshold area of the GCB (refer to Figure 2-4). The SR is set to hex 'FF' when the command is completed.

**4.3.5 PRESET\_STATISTICS Command**

The format of the PRESET\_STATISTICS command is shown below:

7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1

The PRESET\_STATISTICS command causes the MLAPD to immediately load its internal global statistics counters from the corresponding entries in the GCB statistics threshold area. This command allows the host to dynamically change threshold values or to dynamically reinitialize the counters without stopping the MLAPD on-line operation. The SR is set to hex 'FF' when the command is completed.

#### 4.3.6 ENABLE\_IRQ Command

The format of the ENABLE\_IRQ (interrupt request) command is shown below:

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0

The ENABLE\_IRQ command is issued by the host to notify the MLAPD that all pending entries in the Interrupt Queue have been handled. In response, the MLAPD deactivates the IRQ (INTR) pin and checks that the last pending interrupt entry in the Interrupt Queue has been handled. If not, the MLAPD asserts a new interrupt request via IRQ (INTR).

The SR is set to hex 'FF' when the command is completed.

#### 4.3.7 ASSIGN\_POOL\_POINTER Command

The format of the ASSIGN\_POOL\_POINTER command is shown below:

7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	1

This command requires a receive\_pool\_number in the command\_argument\_1 field and the address of the first frame descriptor in the command\_argument\_2 field located in the GCB.

The host issues an ASSIGN\_POOL\_POINTER command to establish a new receive pool. In response, the MLAPD uses the receive\_pool\_number as an index into the Receive Pool Pointers Table. The MLAPD then stores the address of the first frame descriptor in this receive\_pool\_pointer entry. The receive\_pool\_pointers for receive pools 0–15 are stored on chip, and pointers for receive pools 16–8191 are stored in the Receive Pool Pointers Table located in shared memory. The SR is set to hex 'FF' when the command is completed.

### 4.4 PROTOCOL

These commands are the primitives explicitly defined in the LAPD protocol. The protocol commands allow the host to setup and resume link operation as well as to control the flow

of frames on the link. With the exception of DL\_DATA\_REQUEST, the protocol commands are only valid for links operating according to the LAPD protocol.

The short description of the MLAPD's response to each of the protocol commands assumes that the command is issued when the associated logical link's current status allows the command to be handled. For detailed operation descriptions, refer to CCITT Recommendation Q.920/Q9.21.

**4.4.1 MDL\_ASSIGN\_REQUEST Command**

The format of the MDL\_ASSIGN\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	0

This command requires an LLID in the command\_argument\_1 field in the GCB.

In response to an MDL\_ASSIGN\_REQUEST command, the MLAPD fetches the Logical-Link Table (LLT) address from the LLID-LLT Table and reads the Data-Link Connection Identifier (DLCI) and link\_status entries.

If this LLID is not active, then in on-chip operation mode, the MLAPD stores the DLCI and associated LLID in a free CAM entry. If no free entry is found, a CAM\_overflow interrupt is generated. In expanded operation mode, the LLID is stored in the external Match Table. The MLAPD also sets the state of the specified logical link to (terminal endpoint identifier) TEL\_ASSIGN.

If this LLID is already active, then the logical link's state is changed as follows:

TEL\_UNASSIGN (STATE 1) ♦ TEL\_ASSIGN (STATE 4)  
 or  
 EST\_WAIT\_TEI (STATE 3) ♦ AWAIT\_EST (STATE 5)

If the link's state is changed to AWAIT\_EST, a set asynchronous balanced mode extended (SABME) frame is transmitted, T200 is started for this LLID, and the highest\_active\_LLID register is updated, if necessary. The SR indicates that the command has been completed by returning hex 'FF'.

**4.4.2 DL\_ESTABLISH\_REQUEST Command**

The format of the DL\_ESTABLISH\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	1

This command requires an LLID in the command\_argument\_1 field in the GCB.

In response to this command, the MLAPD changes the link's state as follows:

New link (STATE 0) ♦ EST\_WAIT\_TEI (STATE 3)  
or  
TEI\_UNASSIGN (STATE 1) ♦ EST\_WAIT\_TEI (STATE 3)  
or  
TEI\_ASSIGN (STATE 4) ♦ AWAIT\_EST (SPACE 5)

4

If the link's state is changed to AWAIT\_EST, a SABME frame is transmitted, T200 is started for this LLID, and the highest\_active\_LLID register is updated, if necessary. The SR is set to hex 'FF' when the command is completed.

#### 4.4.3 DL\_DATA\_REQUEST Command

The format of the DL\_DATA\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0

This command requires an LLID in the command\_argument\_1 field and an address of the first transmit I frame descriptor in the command\_argument\_2 field located in the GCB.

DL\_DATA\_REQUEST is issued by the host to enable transmission after queuing at least one frame to the specified logical link's Transmit Queue. The pointer associated with the command identifies the head of the Transmit Queue. In response, the MLAPD places the specified logical link into its associated I frame queue by writing the address of this link's LLT in the Tx\_next\_LLT\_pointer entry of the last LLT in the I frame queue. The MLAPD then initializes the following LLT entries for the specified link:

- transmit\_status (initialized to '000F' hex)
- Tx\_next\_pointer (initialized to value in command\_argument\_2)
- Tx\_next\_acknowledge\_pointer (initialized to value in command\_argument\_2)

This command is only accepted by the MLAPD when the current Transmit Queue for this logical link has been transmitted and all frames have been acknowledged. Otherwise, an MDL\_error\_indication interrupt is issued. When frames are added to a nonempty Transmit Queue, no command is required to enable their transmission.

A DL\_DATA\_REQUEST is also issued to resume handling of a logical link's Transmit Queue after the MLAPD has stopped transmission for the specified link due to the receipt of a STOP\_TX-I command.

The SR is set to hex 'FF' when the link's Transmit Queue is placed in its I frame queue. A DL\_data\_confirmation interrupt indicates that the command has been completed.

#### 4.4.4 RELINK\_REQUEST Command

The format of the RELINK\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	1

This command requires an LLID in the command\_argument\_1 field located in the GCB.

The host issues a RELINK\_REQUEST command when the host wants to queue at least one frame to the specified logical link's Transmit Queue and all frames in the link's current Transmit Queue have been transmitted, but not all frames are acknowledged. The MLAPD's response depends upon the two following conditions:

- 1) A real need for relinking exists — meaning that all frames in the queue were transmitted, but not all frames were acknowledged, when the host dynamically added the frame(s) to the queue, and
- 2) No reason exists to prevent the relinking action (e.g., P\_flag = 1).

If these two conditions are met, the RELINK\_REQUEST command is similar to a DL\_DATA\_REQUEST in that the MLAPD: places the link's LLT in its assigned I frame queue; updates the queue's tail pointer; and, if this queue is empty, updates the queue's head pointer also.

If these two conditions are not met, the command may be considered illegal, may be ignored, or may cause the relinking to occur after link conditions clear. For specific information on the affect of a RELINK\_REQUEST command see Figure 2-11. The SR is set to hex 'FF' when the command is accepted.

#### 4.4.5 GLOBAL\_XID/UI\_REQUEST Command

The format of the GLOBAL\_XID/UI\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	0	1	0	0	0	1	0

This host command requires the 32-bit address of the first (exchange identification/un-numbered information) XID/UI transmit frame descriptor in the command\_argument\_2 field of the GCB.

The host issues this command after queuing at least one frame to the Global XID/UI Queue to enable transmission. In response, the MLAPD initializes the Global\_XID/UI\_header\_pointer register to point to the first frame descriptor in the queue. The MLAPD then places the Global XID/UI Queue in the transmit servicing scheme.

The MLAPD accepts this command when all XID/UI frames from a previous GLOBAL\_XID/UI\_REQUEST have been transmitted. Otherwise, an MDL\_error\_indication interrupt is issued. When frames are added to a nonempty Global XID/UI Queue, no command is required to enable their transmission.

This command is also issued to resume handling of the Global XID/UI Transmit Queue after the MLAPD has stopped transmission due to the receipt of a STOP\_GLOBAL\_XID/UI command.

The SR is set to hex 'FF' when this command is accepted. A Global\_XID/UI\_confirmation interrupt is issued when all of this queue is handled by the MLAPD.

#### 4.4.6 XID/UI\_QUEUE\_0\_REQUEST Command

The format of the XID/UI\_QUEUE\_0\_REQUEST is shown below:

7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	0

This command requires the 32-bit address of the first XID/UI Queue\_0 frame descriptor in the command\_argument\_2 field of the GCB.

The host issues this command after queuing at least one frame to the XID/UI Queue\_0 to enable transmission. In response, the MLAPD initializes the XID/UI\_Queue\_0\_head\_pointer register to point to the first frame descriptor in the queue. The MLAPD then places the XID/UI Queue\_0 in the transmit servicing scheme. The MLAPD accepts this command when all XID/UI frames from a previous XID/UI\_QUEUE\_0\_REQUEST have been transmitted. Otherwise, an MDL\_error\_indication interrupt is issued. When frames are added to a nonempty XID/UI Queue\_0, no command is required to enable their transmission.

This command is also issued to resume handling of the XID/UI Queue\_0 after the MLAPD has stopped transmission due to the receipt of a STOP\_XID/UI\_QUEUE\_0 command. The SR is set to hex 'FF' when the command is accepted. An XID/UI\_Queue\_0\_confirmation interrupt indicates that the MLAPD has completed handling this queue.

#### 4.4.7 XID/UI\_QUEUE\_1\_REQUEST Command

The format of the XID/UI\_QUEUE\_1\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	1

This command requires the 32-bit address of the first XID/UI Queue-1 frame descriptor in the command-argument-2 field in the GCB.

After queuing at least one frame to the XID/UI Queue-1 to enable transmission, this command is issued by the host. In response, the MLAPD initializes the XID/UI-Queue-1-head-pointer register to point to the first frame descriptor in the queue. The MLAPD then places the XID/UI Queue-1 in the transmit servicing scheme. The command is accepted by the MLAPD when all XID/UI frames from a previous XID/UI-QUEUE-1-REQUEST have been transmitted. Otherwise, an MDL\_error-indication interrupt is issued. When frames are added to a nonempty XID/UI Queue-1, no command is required to enable their transmission.

This command is also issued to resume handling of the XID/UI Queue-1 after the MLAPD has stopped transmission due to the receipt of a STOP-XID/UI-QUEUE-1 command. The SR is set to hex 'FF' when the command is accepted. An XID/UI-Queue-1-confirmation interrupt indicates that the MLAPD had completed handling this queue.

#### 4.4.8 MDL\_ERROR\_RESPONSE Command

The format of the MDL\_ERROR\_RESPONSE command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0

This command requires an LLID in the command-argument-1 field in the GCB.

The command is issued when T202 expiration has been recognized by the management entity for Layer 2 and before a DLCI has been assigned by the peer management. This command only causes a state change for the specified logical link back to the state TEL\_UNASSIGN. Timer 202 is the minimum time between the transmission of TEL identity request messages. The SR is set to hex 'FF' when the command is completed.

#### 4.4.9 DL\_RELEASE\_REQUEST Command

The format of the DL\_RELEASE\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	0

This command requires an LLID in the command-argument-1 field in the GCB.

If a DL\_RELEASE\_REQUEST is issued under normal link conditions, the MLAPD responds to this command by generating a disconnect (DISC) frame for the specified logical link. The SR is set to hex 'FF' when the command is completed.

#### 4.4.10 MDL\_REMOVE\_REQUEST Command

The format of the MDL\_REMOVE\_REQUEST command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	0

This command requires an LLID in the command\_argument\_1 field in the GCB.

4

In response to an MDL\_REMOVE\_REQUEST, the MLAPD changes the state of the specified logical link to TEL\_UNASSIGN. The highest\_active\_LLID register is updated, if necessary.

The SR indicates that this command has been accepted by returning hex 'FF'. If the LLT of this link is in its assigned I frame queue, the command is actually completed by the transmit task that removes the LLT from its queue. Removal occurs when the LLT reaches the head of its I frame queue. The LLT can be reused only after the link's active bit in its LLT has been set to zero by the MLAPD.

If the LLT of this link is not in its assigned I frame queue, the semaphore value 'FF' hex indicates the completion of the command. In either case, when the active bit in the transmit\_status entry in the link's LLT is set to zero, the command has been completed.

### 4.5 PROTOCOL EXTENSION

These commands are not specified by the LAPD protocol. Because they allow the host to interfere in the normal flow of the protocol processing, the host must carefully issue these commands to the MLAPD. The SET\_LOCAL\_BUSY and CLEAR\_LOCAL\_BUSY commands may also be used as a part of normal protocol processing. The protocol extension commands that are valid for nonprotocol links are: ACTIVATE\_LL, DEACTIVATE\_LL, SET\_LOCAL\_BUSY, CLEAR\_LOCAL\_BUSY, and STOP\_TX-I.

The short description of the MLAPD response to each command assumes that the command is issued when the associated logical link's current state allows the command to be handled.

#### 4.5.1 ACTIVATE\_LL Command

The format of the ACTIVATE\_LL (logical link) command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0

This command requires an LLID in the command\_argument\_1 field in the GCB. The ACTIVATE\_LL command may be issued by the host only after the LLID-LLT Table has a valid



entry for the specified LLID and after the following entries in the corresponding LLT have a valid value:

- DLCI and configuration bits
- N201\_value
- Receive\_pool\_number

The nonprotocol\_select bit in the LLT configuration bits determines whether this newly active link will operate according to the LAPD protocol or whether this link will operate in nonprotocol mode. The ACTIVATE\_LL command configures a LAPD logical link to support unnumbered information (UI) exchange from the management entity. As an example, this command can be used to configure the broadcast logical link.

In response to an ACTIVATE\_LL command, the MLAPD fetches the LLT address from the LLID-LLT Table and reads the DLCI entry. If the requested LLID or DLCI is already active, an MDL\_error\_indication (argument = '000000') interrupt is placed on the Interrupt Queue. Otherwise, in one-chip operation mode, the MLAPD stores the DLCI and the associated LLID in a free CAM entry. If no free entry is found, a CAM\_overflow interrupt is generated. In expanded operation mode, the LLID is stored in the external Match Table. For a protocol link, the MLAPD also sets the state of the specified logical link to TEI\_UNASSIGN and updates the highest\_active\_LLID register, if necessary. When the SR has a value of hex 'FF', the specified logical link is ready for operation.

#### 4.5.2 DEACTIVATE\_LL Command

The format of the DEACTIVATE\_LL command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	0

This command requires an LLID in the command\_argument\_1 field in the GCB.

In response to a DEACTIVATE\_LL command, the MLAPD sets the transmit\_status bits as follows: stop\_TX = 1; active = unchanged; acknowledge = 0; and transmit = 0. The MLAPD also deletes the specified logical link's DLCI-LLID entry in the CAM when in the on-chip operation mode. In expanded-system operation mode, the DLCI-LLID entry is deleted from the external Match Table by the MLAPD. The highest\_active\_LLID register is also updated, if necessary.

The SR indicates command acceptance by returning hex 'FF', and, at that time, the host may issue another command. If this link's LLT is in its assigned I frame queue, the command is actually completed by the transmit task that removes the LLT from the queue. Removal occurs when the LLT reaches the head of its I frame queue. The LLT can be reused only after the link's active bit in its LLT is set to zero by the MLAPD.

If the LLT of this link is not in its assigned I frame Queue, the semaphore value 'FF' hex indicates completion of the command. In either case, when the active bit in the transmit-status entry is set to zero, the command has been completed.

### 4.5.3 REMOTE\_STATUS\_REQUEST Command

The format of the REMOTE\_STATUS\_REQUEST is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1

This command requires an LLID in the command\_argument\_1 field in the GCB.

The MLAPD responds to this command as follows:

- If the P\_flag is set for this logical link (i.e., a supervisory frame has been sent with the poll (P) bit set to one, and no response frame has been received with the final (F) bit set to one), the MLAPD sets the remote\_status\_check flag indicating that a REMOTE\_STATUS\_REQUEST has been issued for this logical link.
- If the P\_flag is not set for this logical link, the MLAPD transmits a S frame, appropriate for link's current state, with the P bit set to one and sets the remote\_status\_check flag and the P\_flag.

In both cases, when a response frame is received with F set to one, the MLAPD issues a Remote\_status\_confirmation interrupt and clears both the remote\_status\_check flag and the P\_flag. The SR is set to hex 'FF' when the command is accepted.

### 4.5.4 SET\_LOCAL\_BUSY Command

The format of the SET\_LOCAL\_BUSY command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1

This command requires an LLID in the command\_argument\_1 field located in the GCB.

A SET\_LOCAL\_BUSY command forces the receive busy situation, in which no free buffers in the receive pool are associated with the specified LLID. Higher level software may use this command for immediate flow control by allowing the data-link layer to initiate flow control rather than asserting flow control at a higher level. In addition, a remote station could request this busy condition through dialogue between the remote and local management layers. In this case, the command can be used to check the operation of the station

in a local busy condition for testing or certification purposes. The SR indicates that the local-busy flag in the appropriate LLT's link-status word is set by returning hex 'FF'.

#### 4.5.5 CLEAR\_LOCAL\_BUSY Command

The format of the CLEAR\_LOCAL\_BUSY command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0

This command requires an LLID in the command\_argument\_1 field located in the GCB.

When the local-busy flag was previously set by a SET\_LOCAL\_BUSY command, the host issues a CLEAR\_LOCAL\_BUSY command to clear the flag and return the link to normal operation. The command will also clear the flag and return the link to normal operation. The command will also clear a true receive busy condition caused by lack of receive buffers. After adding buffers to an empty receive pool, the host can issue this command to return to normal operation on this link. The SR is set to hex 'FF' when this command is completed.

#### 4.5.6 STOP\_TX\_I Command

The format of the STOP\_TX\_I command is shown below:

7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	1

This command requires an LLID in the command\_argument\_1 field in the GCB.

The host issues a STOP\_TX\_I command to instruct the MLAPD not to initiate any new transmission service for this specified logical link's Transmit Queue. The SR is set to hex 'FF' when this command is accepted. If the MLAPD receives this command while transmitting an I frame for the link, then the MLAPD aborts the frame. Next, the MLAPD waits for all outstanding I frames for the specified logical link to be acknowledged and then places a DL\_data\_confirmation interrupt on the Interrupt Queue indicating that the STOP\_TX\_I command has been completed.

#### 4.5.7 STOP\_GLOBAL\_XID/UI Command

The format of the STOP\_GLOBAL\_XID/UI command is shown below:

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	1

A STOP\_GLOBAL\_XID/UI command is issued by the host to instruct the MLAPD not to initiate any new transmission service for the Global XID/UI Queue. If the MLAPD receives this command while transmitting an XID/UI frame from this queue, then MLAPD aborts this frame. The SR is set to hex 'FF' when this command is completed.

#### 4.5.8 STOP\_XID/UI\_QUEUE\_0 Command

The format of the STOP\_XID/UI\_QUEUE\_0 command is shown below:

7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0

The host issues STOP\_XID/UI\_QUEUE\_0 command to instruct the MLAPD not to initiate any new transmission service for the XID/UI Queue\_0. If the MLAPD receives this command while transmitting an XID/UI frame from this queue, then the MLAPD aborts this frame. The SR is set to hex 'FF' when this command is completed.

#### 4.5.9 STOP\_XID/UI\_QUEUE\_1 Command

The format of the STOP\_XID/UI\_QUEUE\_1 command is shown below:

7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1

A STOP\_XID/UI\_QUEUE\_1 command is issued by the host to instruct the MLAPD not to initiate any new transmission service for the XID/UI Queue\_1. If the MLAPD receives the command while transmitting an XID/UI frame from this queue, then the MLAPD aborts this frame. The SR is set to hex 'FF' when this command is completed.

### 4.6 COMMAND SUMMARY

Table 4-1 briefly defines the MLAPD commands.

Table 4-2 summarizes the MLAPD command set.

**Table 4-1. MLAPD Command Definitions**

Command Group	Command	Functional Description
Initialization	RESET SET_BUS_WIDTH_8 SET_BUS_WIDTH_16 INIT	Software reset. Data bus width is 8 bits. Data bus width is 16 bits. MLAPD loads registers from the GCB.
Host/MLAPD Interface	OFF-LINE ON-LINE RELOAD DUMP_STATISTICS PRESET_STATISTICS ENABLE_IRQ  ASSIGN_POOL_POINTER	MLAPD only executes host commands. MLAPD executes commands, receives frames, and transmits frames. MLAPD loads registers from reloadable area of GCB. MLAPD writes global counters to specified memory area. MLAPD loads global counters from statistics threshold area of GCB. All pending interrupt entries handled. Enable external interrupt signal. Host has created a new receive pool.
Protocol	MDL_ASSIGN_REQUEST DL_ESTABLISH_REQUEST DL_DATA_REQUEST  RELINK REQUEST  GLOBAL_XID/UI_REQUEST XID/UI_QUEUE_0_REQUEST XID/UI_QUEUE_1_REQUEST MDL_ERROR_RESPONSE  DL_RELEASE_REQUEST MDL_REMOVE_REQUEST	Place specified logical link in the TEL_ASSIGN state. Setup link. Link enters multiple-frame established mode. Request transmission of an I frame queue for the specified logical link. Frames were added to a queue in which all frames were 'Tx awaiting ACK'. Request transmission of XID/UI frames in the Global XID/UI Queue. Request transmission of XID/UI frames in XID/UI_Queue_0. Request transmission of XID/UI frames in XID/UI_Queue_1. Indication that no DLCI assigned. The logical link is in TEL_UNASSIGN state. Release link from the multiple-frame established mode. Place logical link in TEL_UNASSIGN state.
Protocol Extension	ACTIVATE_LL  DEACTIVATE_LL REMOTE_STATUS_REQUEST  SET_LOCAL_BUSY CLEAR_LOCAL_BUSY STOP_TX_I STOP_GLOBAL_XID/UI STOP_XID/UI_QUEUE_0 STOP_XID/UI_QUEUE_1	Place specified logical link in the TEL_UNASSIGN state. Remove DLCI-LLID pair. Remove specified logical link from LAPD service. Send frame with poll bit set to check remote station status for specified link. Specified logical link enters the local busy condition. Specified logical link exits the local busy condition. Stop transmission of the specified logical link's I frame queue. Stop transmission of frames in the GLOBAL XID/UI Queue. Stop transmission of frames in the XID/UI_Queue_0. Stop transmission of frames in the XID/UI_Queue_1.
Test Diagnostic	DMA TEST DUMP	Test DMA operation. Dump all internal MLAPD registers and CAM.

**Table 4-2. MLAPD Command Summary**

Code	Command	Command Argument_1	Command Argument_2	Command Argument_3
<b>Initialization Command</b>				
FF	RESET	—	—	—
40	SET_BUS_WIDTH_8	—	—	—
41	SET_BUS_WIDTH_16	—	—	—
42	INIT (GCB address is in data register)	—	—	—
<b>Host Interface Commands</b>				
44	OFF-LINE	—	—	—
45	ON-LINE	—	—	—
46	RELOAD	—	—	—
23	DUMP STATISTICS	—	Pointer	—
47	PRESET_STATISTICS	—	—	—
88	ENABLE_IRQ	—	—	—
A1	ASSIGN_POOL_POINTER	Pool_Number	Pool_Pointer	—
<b>Protocol Commands</b>				
66	MDL_ASSIGN_REQUEST	LLID	—	—
63	DL_ESTABLISH_REQUEST	LLID	—	—
A0	DL_DATA_REQUEST	LLID	Queue_Pointer	—
6B	RELINK_REQUEST	LLID	—	—
22	GLOBAL_XID/UI_REQUEST	—	Queue_Pointer	—
26	XID/UI_QUEUE_0 REQUEST	—	Queue_Pointer	—
27	XID/UI_QUEUE_1 REQUEST	—	Queue_Pointer	—
68	MDL_ERROR_RESPONSE	LLID	—	—
64	DL_RELEASE_REQUEST	LLID	—	—
6C	MDL_REMOVE_REQUEST	LLID	—	—
<b>Protocol Extension Commands</b>				
62	ACTIVATE_LL	LLID	—	—
6A	DEACTIVATE_LL	LLID	—	—
69	REMOTE_STATUS_REQUEST	LLID	—	—
61	SET_LOCAL_BUSY	LLID	—	—
60	CLEAR_LOCAL_BUSY	LLID	—	—
67	STOP_TX_I	LLID	—	—
89	STOP_GLOBAL_XID/UI	—	—	—
8A	STOP_XID/UI_QUEUE_0	—	—	—
8B	STOP_XID/UI_QUEUE_1	—	—	—
<b>Test/Diagnostics Commands</b>				
A2	DMA_TEST	Data_Length	Source_Pointer	Destination_Pointer
20	DUMP	—	Pointer	—

## SECTION 5 TRANSMIT PROCESS

### 5.1 TRANSMIT SERVICING SCHEME

The MLAPD services several transmit queues. All frames in a single queue belong to one of the following categories: Level 2 (MLAPD) generated frames, exchange identification/unnumbered information (XID/UI) frames (including nonstandard-control frames), or numbered information (I) frames. A fixed internal servicing scheme determines when each transmit queue is handled.

The highest priority transmit queue is the Level 2 Transmit Queue. When the MLAPD services the Level 2 Queue, all queued frames are transmitted before servicing the next lower priority queue. The Level 2 Queue contains frames generated by the MLAPD in response to received frames and host commands. These frames, unnumbered (U) and supervisory (S), are referred to as *Level 2 frames* throughout this specification. A breakdown of Level 2 frames is given in Figure 2-19. Since these Level 2 frames are generated independently by the MLAPD and this queue is managed transparently to the host, the Level 2 Queue is discussed in a separate section. (See **2.10 LEVEL 2 QUEUE**).

The next lower priority transmit queue is the Global XID/UI Queue. When the MLAPD services the Global XID/UI Queue, all queued frames are transmitted before servicing the lowest priority queues. The Global XID/UI Queue contains XID and UI (also nonstandard-control) frames generated by the host for transmission on the specified logical link. System implementations may dedicate the Global XID/UI Queue for frames generated by the Level 3 management entity.

The lowest priority transmit queues are the user-defined servicing queues, which contain I frames and XID/UI (also nonstandard-control) frames. A total of six queues implement user-defined servicing: four transmit I frame queues (I frame Queues\_0–3) and two XID/UI queues (XID/UI Queues\_0, 1). When the MLAPD services these queues, all queued frames are NOT transmitted before servicing the next I or XID/UI queue. Instead, the user defines the maximum number of frames to be transmitted from each queue before switching to the next queue. This variable is programmed into the scan-length fields in the Global Configuration Block (GCB). When a scan-length of zero is programmed, the corresponding queue is never serviced; thus, the user can opt to implement from zero up to all four queues for I frame transmission and from zero up to two queues for XID/UI (and nonstandard control) frame transmission. The scan-length for any queue may be changed dynamically by writing the appropriate GCB entry and issuing a RELOAD command.

While the scan-length does not implement an absolute priority scheme, the scan-length does allow the user to define the relative servicing of transmit frames in each queue that

is to be provided by the MLAPD transmit task, eliminating the problem of queue starvation. In an absolute priority scheme, frames in a queue with a higher scan-length would always be transmitted before frames in a queue with a lower scan-length. In a relative priority scheme, the MLAPD transmit task services frames in a queue with a higher scan length more often than frames in a queue with a lower scan-length although the queue with the lower scan-length is guaranteed to receive a percentage of the transmit service.

According to the fixed priority scheme, the MLAPD transmits all frames in the Level 2 Queue and all frames in the Global XID/UI Queue before servicing the lower priority I frame queues and XID/UI queues. The MLAPD services the lowest priority queues in round robin fashion as follows: I frame Queue\_0, I frame Queue\_1, I frame Queue\_2, I frame Queue\_3, XID/UI Queue\_0, XID/UI Queue\_1, I frame Queue\_0, etc. After each frame, transmission from the lowest priority queues may be interrupted to maintain priority to the Level 2 and Global XID/UI Queues. Upon completion, the MLAPD will resume servicing the lowest priority queues at the point of interruption. The MLAPD transmit servicing scheme is shown pictorially in Figure 5-1 and in flowchart form in Figure 5-2.

## 5.2 TRANSMIT I FRAME PROCESS

The host assigns each logical link to one of four I frame queues for all of its I frame transmission. When a link has I frames pending transmission, the MLAPD places the link in its assigned queue. Each I frame queue is serviced according to the transmit servicing scheme. The MLAPD maintains pointers to control the process flow. The following paragraphs provide an overview of the transmission process and also explanations of the various procedures and capabilities.

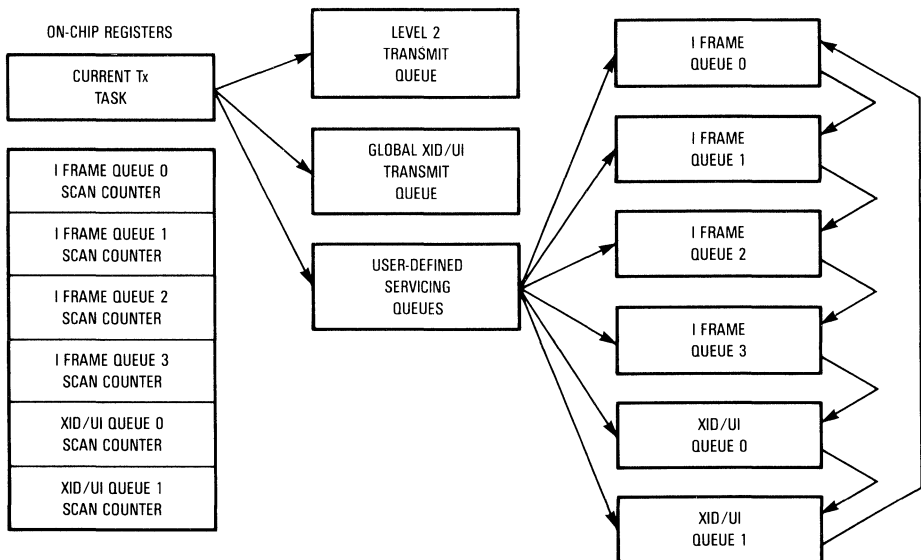


Figure 5-1. Transmit Servicing Scheme



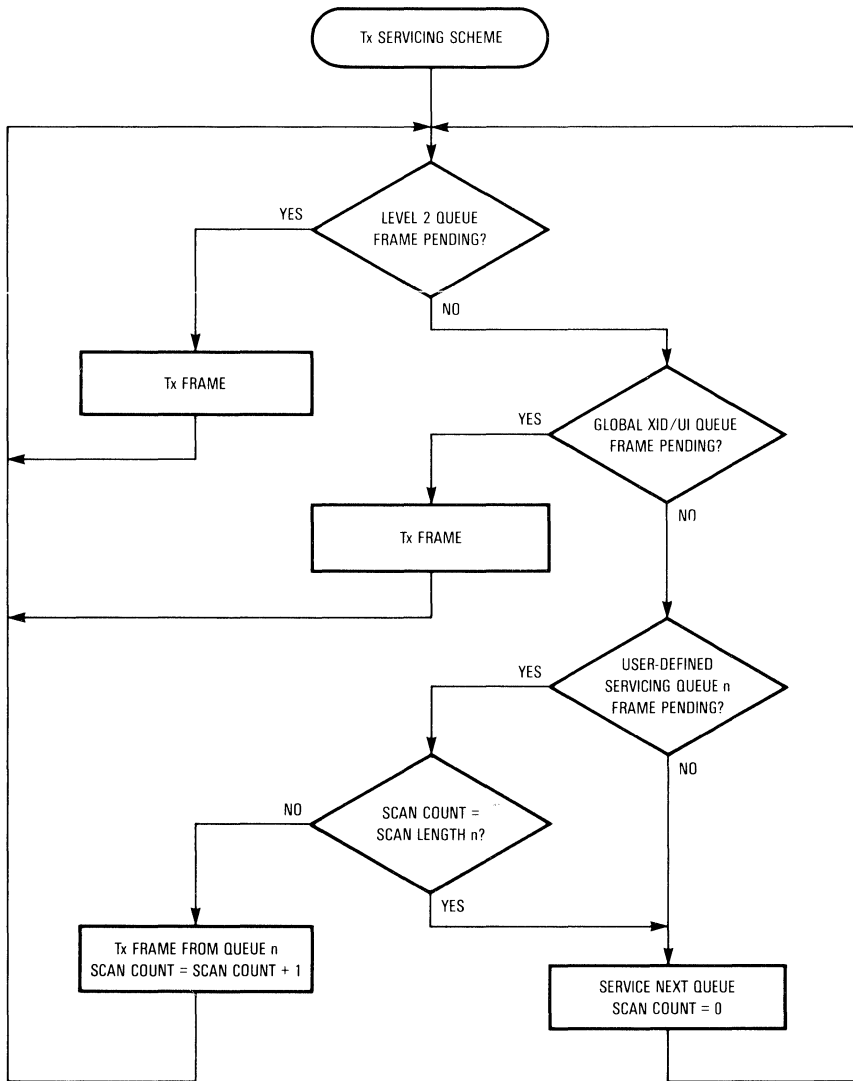
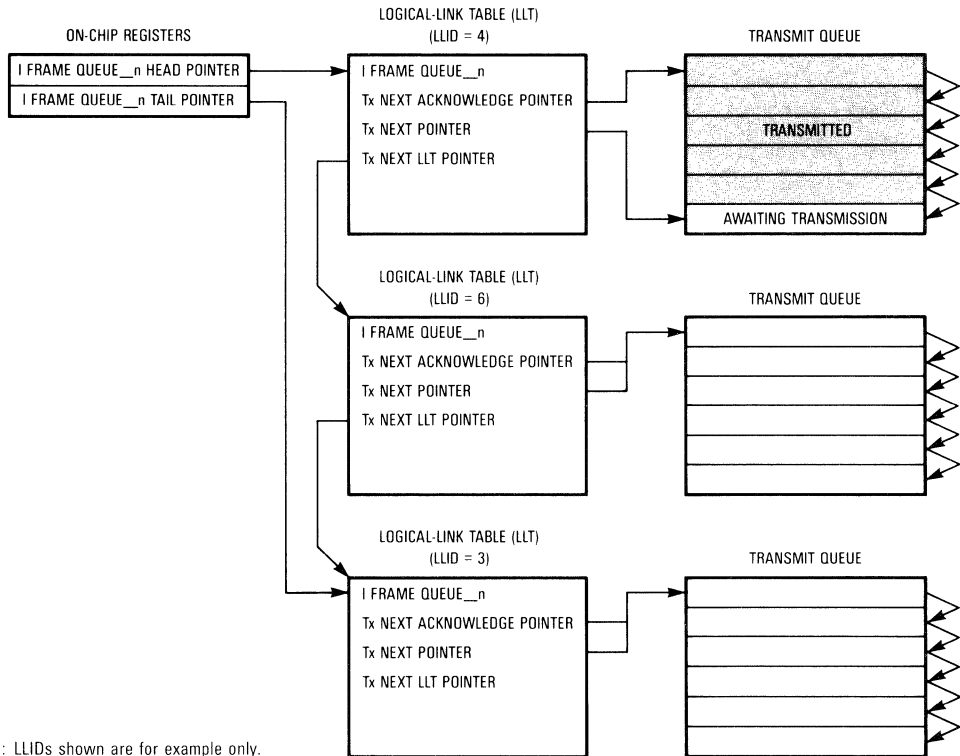


Figure 5-2. Transmit Servicing Flowchart

### 5.2.1 I Frame Queue Structure

An I frame queue consists of linked LLTs belonging to links with I frames awaiting transmission. Each LLT contains three pointers maintained by the MLAPD, which identify the next frame to be acknowledged, the next frame to be transmitted, and the next logical link in the I frame queue to be serviced. The structure of an I frame queue is shown in Figure 5-3.



NOTE: LLIDs shown are for example only.

**Figure 5-3. Structure of Each I Frame Queue**

The MLAPD maintains a pointer to the tail and head of each transmit I frame queue. In response to a DL\_DATA\_REQUEST command, the MLAPD places the link's LLT into the assigned I frame queue using the corresponding tail pointer to locate the last LLT belonging to the queue. When the MLAPD is ready to service an I frame queue, the MLAPD uses the corresponding head pointer register to locate the first LLT in the queue.

The MLAPD is exclusively responsible for handling each I frame queue. A logical link's Transmit Queue is linked to its I frame queue if and only if the MLAPD is able to provide transmission services for the link. Inability to provide transmission services may result from either local or remote link conditions and host commands. The MLAPD relinks the logical link's Transmit Queue to its I frame queue when the link conditions inhibiting transmission servicing clear. For more details see **5.2.5 MLAPD I Frame Queue Processing**.

### 5.2.2 Logical-Link Transmit Queue Structure

To minimize context switching, each logical link has its own Transmit Queue for I frames. A Transmit Queue is a linked list of frame descriptors. The format of a transmit frame

descriptor is discussed in **2.6 TRANSMIT FRAME DESCRIPTOR**. A Transmit Queue may be conceptually divided into two portions: a portion which is inactive and a portion which is active. In the inactive portion, all frames have been transmitted and acknowledged. The active portion contains frames awaiting acknowledgment (frames which have been transmitted) and frames awaiting transmission. The inactive portion of the Transmit Queue has been returned to the host, and these frame structures may be collected and reused. The active portion is still under the control of the MLAPD. The structure of a logical link's Transmit Queue is shown in Figure 5-4. To enable transmission of I frames in a Transmit Queue, this queue must be linked to one of the four user-defined I frame queues. The host assigns each logical link to one of these queues for all of its I frame activity.

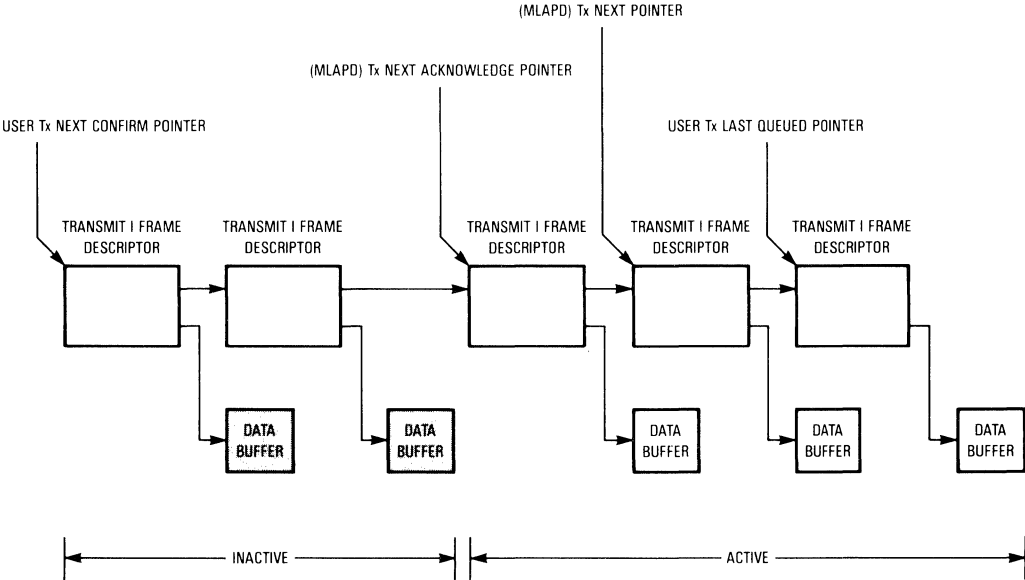


Figure 5-4. Logical-Link Transmit Queue Structure

5.2.3 I Frame Transmission Queueing

To queue I frames for transmission, the host first creates a Transmit Queue for a logical link and then instructs the MLAPD to add the queue to its assigned I frame queue by issuing a DL\_DATA\_REQUEST command. (See **4.4.3 DL\_DATA\_REQUEST Command**). The host must maintain the user Tx\_next\_confirm\_pointer and the user Tx\_last\_queued\_pointer for each logical link's Transmit Queue. These pointers are required for removing acknowledged frame descriptors and/or for adding more frame descriptors for transmission.

The host may add frames to the Transmit Queue while frames are still awaiting transmission. When a frame is ready to be added, the host sets the last bit in the new frame descriptor and links the new frame descriptor to the last frame descriptor in the existing

Transmit Queue. The last frame descriptor is identified by the user `Tx-last-queued-pointer`. The host then clears the last bit in the frame descriptor that was previously the last frame descriptor in the queue and updates the user `Tx-last-queued-pointer`. Upon receiving a `DL-data-confirmation` interrupt, the host must verify that the last frame added to the queue was indeed transmitted. If not, then the addition operation was performed after the MLAPD had checked the last bit in the frame descriptor, which the host intended to clear. In this case, the host must present the remaining frames to the MLAPD as a new Transmit Queue and issue a `DL-DATA-REQUEST`.

The host may also add frames to the Transmit Queue after all frames have been transmitted but before all frames have been acknowledged. In this case, the host must issue a `RELINK-REQUEST` command after the frames are added to the queue to enable transmission.

#### 5.2.4 MLAPD I Frame Queue Processing

Upon receiving a `DL-DATA-REQUEST` command, the MLAPD adds the logical link's Transmit Queue to the appropriate I frame queue for transmission, using the queue's tail pointer register. The `Tx-next-pointer` and `Tx-next-acknowledge-pointer` LLT entries for this link are initialized to point to the first frame descriptor in this link's Transmit Queue. The `stop-Tx`, `active`, `acknowledge`, and `transmit` bits in this link's `transmit-status` entry are set to 0111, respectively. The `last-LLT` bit in the `transmit-status` entry is set to one since this LLT is now the last one in the I frame queue.

After loading the link's state, (or context), information from the LLT, the MLAPD locates the link's Transmit Queue using the `Tx-next-pointer`. The MLAPD then begins transmitting the link's queued frames. Advancing the `Tx-next-pointer`, the MLAPD continues to transmit frames from the link's Transmit Queue until the queue is emptied or until the `scan-length` associated with this I frame queue is exhausted. If the link's Transmit Queue is emptied before the `scan-length` is exhausted, the MLAPD locates the next LLT in the I frame queue using the linking mechanism, the `Tx-next-LLT` entry in the current link's LLT. The MLAPD then loads the next link's context and begins transmitting frames from its Transmit Queue. This process continues until the `scan-length` for this I frame queue is satisfied.

The MLAPD may temporarily remove this link's Transmit Queue from its I frame queue as the result of link conditions and host commands. The link conditions that cause removal are:

- Remote busy condition,
- Outstanding remote status request (`P-flag` or `remote-status-check-`),
- Outstanding frames limit (`K`),
- Link reset, and
- Timer recovery condition.

The remote busy condition, the outstanding remote status request, and the timer recovery condition are indicated by the `link-status` LLT entry. The outstanding frames limit condition

is determined by the value of  $V(S) - V(A) \geq K$ . Send\_state\_variable  $V(S)$ , acknowledge\_state\_variable  $V(A)$ , and  $K$  are entries in the link's LLT. The link reset condition is reported via the Interrupt Queue. The host commands that cause removal are:

STOP\_TX\_I,  
MDL\_REMOVE\_REQUEST,  
DEACTIVATE\_LL,  
DL\_RELEASE\_REQUEST, and  
REMOTE\_STATUS\_REQUEST.

The MLAPD also removes the link's Transmit Queue from its I frame queue after all frames have been transmitted. Later, when all frames have been acknowledged, the MLAPD checks whether any transmit frames have been added to the link's Transmit Queue. If so, the link's Transmit Queue is automatically relinked to the tail of its I frame queue by the MLAPD.

5

When a transmitted frame is acknowledged, the MLAPD updates the Tx\_next\_acknowledge pointer to the next frame awaiting confirmation.

After all frames have been transmitted and acknowledged for this link, a DL\_data\_confirmation interrupt is issued by the MLAPD to notify the host. At this time, the host may request transmission of a new queue of frames for this logical link by issuing another DL\_DATA\_REQUEST command.

### 5.2.5 Stopping I Frame Transmission

At any time, the host may instruct the MLAPD to immediately suspend all I frame transmit activity for a specific logical link by issuing a STOP\_TX\_I command. (See **4.5.6 STOP\_TX\_I Command**.) Upon receiving this command, the MLAPD aborts any current frame transmission for this link and does not advance the Tx\_next\_pointer past its current position. The host can immediately manipulate any transmit frames following the last frame in which the transmit status bit is set. Later, the host is notified via a DL\_data\_confirmation interrupt that all outstanding acknowledgments have been received. At this point, the host is free to manipulate the entire Transmit Queue as desired since this queue is inactive. The host reactivates the handling of I frame transmission for this link via the usual DL\_DATA\_REQUEST command. This stop mechanism may be used to provide faster servicing of certain I frames or may be necessary to implement various types of error recovery.

To stop transmit activity for a link without the possibility of an abort transmission, the host can set the last bit in the first two frame descriptors in the link's Transmit Queue which have not been transmitted. This action ensures that the MLAPD will transmit only these two frames at the most before stopping the link's transmission. The host can immediately manipulate any transmit frames following the frames in which the last bit was set. Later, the host is notified via a DL\_data\_confirmation interrupt that all outstanding acknowledgments have been received. At this point, the host is free to manipulate the entire Transmit Queue as desired since this queue is inactive.

## 5.2.6 Collecting Acknowledged Frames

To collect acknowledged frames, the host reads the status\_bits in each frame descriptor beginning with the frame descriptor indicated by the user\_Tx\_next\_confirm\_pointer. Once the associated data buffer has been transmitted and acknowledged, the confirmation bit is set to one by the MLAP. Acknowledged frames may be collected by the host dynamically (while other frames await acknowledgment) or after being notified by the MLAPD that all frames in the queue are acknowledged via a DL\_data\_confirmation interrupt. The host should update the user\_Tx\_next\_confirm\_pointer after each frame is collected.

When the host has dynamically added frames to a link's Transmit Queue, the host must determine whether the additional frames were handled by the MLAPD. Verification is made by inspecting the frame descriptor which was the last frame descriptor in the queue before the additional frames were added. If the empty bit is set and the last bit is cleared in this frame descriptor, then the frame addition was not successful. The host must issue a DL\_DATA\_REQUEST with this link's LLID in command\_argument\_1 to enable transmission. The address of the next frame descriptor, the unsuccessful frame descriptor added previously, should be placed in command\_argument\_2.

5

## 5.3 TRANSMIT XID/UI FRAME PROCESSING

XID/UI (also nonstandard\_control) transmit frames are contained in both the Global XID/UI Transmit Queue and the user-defined servicing queues, XID/UI Queues\_0, 1. Each Transmit Queue consists of frame descriptors with associated data buffers. In order to maintain process flow, pointers are used to identify the next frame for transmission. The following paragraphs provide an overview of the transmission process and also explanations of the various procedures and capabilities.

### 5.3.1 XID/UI Queue Structure

The Global XID/UI Queue and the XID/UI Queues\_0,1 are linked lists of frame descriptors. These queues may contain frames to be transmitted on one or more logical links. Each frame descriptor specifies the logical link associated with the XID, UI, or nonstandard\_control frame. The format of a transmit frame descriptor is discussed in **2.6 TRANSMIT FRAME DESCRIPTOR**. The structure of the XID/UI queues is shown in Figure 5-5.

The XID/UI queue may be conceptually divided into two portions: a portion which is inactive (frames which have been transmitted) and a portion which is active (frames awaiting transmission). The inactive portion of the Transmit Queue has been returned to the host and these frame structures may be collected and reused. The active portion is still under the control of the MLAPD.

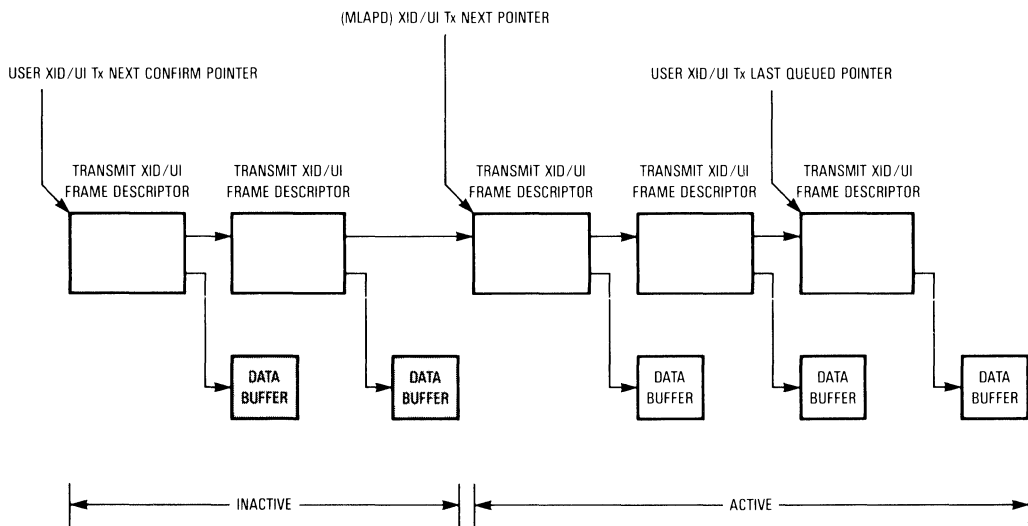


Figure 5-5. XID/UI Transmit Queue Structure

### 5.3.2 XID/UI Transmit Queue Usage

XID/UI queues contain XID frames, UI frames, and user-defined nonstandard-control frames. Infrequent use of XID frames and nonstandard-control frames by Level 3 entities is anticipated. In applications, which primarily use numbered information frames for data transfer, the infrequent use of UI frames is anticipated. However, some applications, e.g., bridges which interface to connectionless-oriented networks, may use UI frames extensively for data transfer.

To address the varying usage of XID/UI (and nonstandard-control) frames, the MLAPD provides two methods of servicing XID/UI queues. In the first method, when frames are queued to the Global XID/UI Queue, the MLAPD will transmit all pending frames each time this queue is serviced. After the Level 2 Queue, the Global XID/UI Queue has second highest transmit priority. In the second method, when frames are queued to either XID/UI Queue\_0 or XID/UI Queue\_1, the MLAPD will transmit only the number of frames specified by the host in the corresponding scan-length GCB entry. In this way, the host defines the servicing to be provided by the MLAPD transmit task for the queued frames.

The host may choose to use only the Global XID/UI Queue, only the XID/UI Queues\_0 and/ or 1, or a combination of these queues depending on the application. (Of course, the host may opt not to use XID, UI, or nonstandard-control frames at all.)

### 5.3.3 XID/UI Frame Transmission Queuing

To begin XID/UI frame transmission, the host first creates a transmit queue consisting of any number of XID/UI (and nonstandard-control) frames to be transmitted on various

logical links. Then the host instructs the MLAPD to add this queue to the transmit servicing scheme by issuing an `XID/UI-QUEUE-0-REQUEST`, an `XID/UI-QUEUE-1-REQUEST`, or a `GLOBAL-XID/UI-REQUEST`, depending on the servicing desired by the host for these frames.

The host must also maintain a `user-XID/UI-Tx-next-confirm-pointer` and a `user-XID/UI-Tx-last-queued-pointer` for these queues. These pointers, which may be stored in the user area of the GCB, are required for removing transmitted frame descriptors and/or for adding frame descriptors for transmission.

The host may add XID/UI (and nonstandard-control) frames to an existing XID/UI queue while frames are still awaiting transmission. When a frame is ready to be added to a queue, the host sets the last bit in the new frame descriptor and links the new frame descriptor to the last frame descriptor in the existing queue. The last frame descriptor is indicated by the `user-XID/UI-Tx-last-queued-pointer` associated with the XID/UI queue. The host then clears the last bit in the frame descriptor that was previously the last frame descriptor in the queue and updates the associated `user-XID/UI-Tx-last-queued-pointer`. Upon receiving an XID/UI confirmation interrupt for this queue, the host must verify that the last frame added to the queue was handled by the MLAPD. If not, then the addition operation was performed after the MLAPD had checked the last bit in the frame descriptor that the host intended to clear. In this case, the host must present the remaining frames by issuing an appropriate `GLOBAL-XID/UI-REQUEST`, `XID/UI-QUEUE-0-REQUEST`, or `XID/UI-QUEUE-1-REQUEST` command.

### 5.3.4 MLAPD XID/UI Queue Processing

The MLAPD maintains a pointer to the head of each XID/UI queue. These three pointers are: 1) the Global XID/UI-head pointer, 2) the XID/UI Queue-0-head pointer, and 3) the XID/UI Queue-1-head pointer. Each pointer is contained in the corresponding register. The MLAPD is given the address of the head of the queue in the `command-argument-2` field when a `XID/UI-request` command is issued.

When the MLAPD is ready to service the Global XID/UI Queue, the MLAPD uses the `Global-XID/UI-Tx-next-FD` register to locate the first frame descriptor in the queue. The MLAPD transmits XID, UI, and nonstandard-control frames from the queue, until the queue is emptied. The MLAPD then issues a `Global-XID/UI-confirmation` interrupt. At this time, the host may request transmission of a new queue of XID/UI (and nonstandard-control field) frames by again issuing a `GLOBAL-XID/UI-request` command.

When the MLAPD is ready to service XID/UI Queue-0 or 1, the MLAPD uses the corresponding `XID/UI-Tx-next-FD` register to locate the first frame descriptor in the queue. The MLAPD transmits XID, UI, and nonstandard-control frames from the queue until the queue is emptied or until the `scan-length` associated with this XID/UI queue is exhausted. If the `scan-length` is exhausted before the queue is emptied, the MLAPD moves on to service the next XID/UI queue or I frame queue in the transmit servicing scheme. When the XID/UI queue is eventually emptied, the MLAPD issues a confirmation interrupt. At this time,



the host may request transmission of a new queue of XID/UI (and nonstandard\_control) frames by again issuing a XID/UI\_QUEUE\_0\_REQUEST or an XID/UI\_QUEUE\_1\_REQUEST.

### 5.3.5 Stopping XID/UI Frame Transmission

At any time, the host may instruct the MLAPD to immediately suspend all transmit activity for an XID/UI queue by issuing a STOP\_GLOBAL\_XID/UI, STOP\_XID/UI\_QUEUE\_0 or STOP\_XID/UI\_QUEUE\_1 command. Upon receiving one of these commands, the MLAPD aborts any current XID/UI (or nonstandard\_control) frame transmission for this queue and does not service additional XID/UI frame descriptors in the specified queue. At this point, the host is free to manipulate the queue as desired since this queue is inactive. The host reactivates frame transmission for the queue via the appropriate command (GLOBAL\_XID/UI\_REQUEST, XID/UI\_QUEUE\_0\_REQUEST, or XID/UI\_QUEUE\_1\_REQUEST). This stop mechanism may be used to provide faster servicing of certain XID/UI (or nonstandard\_control) frames for a particular logical link or may be necessary to implement various types of error recovery.

To stop transmit activity for an XID/UI queue without the possibility of abort transmission, the host can set the last bit in the first two frame descriptors in the XID/UI queue which have not been transmitted. This action ensures that the MLAPD will transmit only these two frames at the most before stopping frame transmission for this queue.

### 5.3.6 Collecting Transmitted XID/UI Frames

The process of collecting transmitted XID/UI frames is the same for all three XID/UI queues. To collect transmitted frames, the host reads the status\_bits in each frame descriptor beginning with the frame descriptor indicated by the user\_XID/UI\_Tx\_next\_confirm\_pointer associated with the queue. When either the positive\_confirmation bit or the negative\_confirmation bit is set, the associated data buffer has been handled by the MLAPD. If the last bit is set, then this frame descriptor is the last frame descriptor in this XID/UI queue.

The positive\_confirmation bit indicates that the associated data buffer has been transmitted, while the negative\_confirmation bit indicates that the associated data buffer cannot be transmitted. A negative indication is generated whenever frame transmission is requested for a logical link in conflict with the services available in the current link state as defined by the LAPD protocol state tables. (DL\_UI, XID, and nonstandard\_control frame transmission service is not permitted when a logical link is in the TEI\_UNASSIGN state or EST\_WAIT\_TEI state. Requests to transmit such frames for this logical link are not honored by the MLAPD and the negative\_confirmation bit is set.) When frame transmission is denied, the host may later resubmit the frame for transmission by adding the frame descriptor to an XID/UI queue.

Transmitted frames may be collected by the host dynamically (while other frames await transmission) or after being notified by the MLAPD that all frames in the queue have been

sent via an `XID/UI_confirmation` interrupt. After each frame is collected from an `XID/UI_queue`, the host should update the associated `user_XID/UI_Tx_next_confirm_pointer`.

If the host has dynamically added frames to an `XID/UI_transmit_queue`, the host must determine if the additional frames were handled by the MLAPD. The host can easily check for this situation by inspecting the frame descriptor which was the last frame descriptor in the queue before the additional frames were added. If the empty bit is set and the last bit in this frame descriptor is cleared, then the addition was not successful. The host must issue an appropriate `XID/UI_request` command with the address of the next frame descriptor, the unsuccessful frame descriptor added previously, in the `command_argument_2` to enable transmission.

## 5

### 5.4 ERRORS DURING FRAME TRANSMISSION

Two possible errors may occur during frame transmission:

- Transmit FIFO (TxFIFO) underrun
- Clear-to-send (CTS) lost

If the TxFIFO is emptied during frame transmission, the next first-in first-out (FIFO) read by the transmitter causes the transmit channel to enter the underrun state. In this state, the current frame transmission is aborted, and the MLAPD later retransmits the same frame. If another underrun occurs while this frame is being retransmitted, the MLAPD will try again until successful. No limit is placed on the number of retries since underrun implies that the problem concerns the system bus bandwidth allocated to the MLAPD. Also, an underrun counter is incremented, and when it reaches the underrun threshold defined in the GCB, a `Tx_underrun_threshold_reached` interrupt is issued by the MLAPD.

If the  $\overline{\text{CTS}}$  pin is negated during frame transmission for more than one transmit clock cycle, the transmitted frame will be corrupted. For each bit time that  $\overline{\text{CTS}}$  is negated, the transmit data pin is three-stated. So, the transmitted frame will most probably be received by the remote station with cyclical redundancy check (CRC) error. Although a `CTS_lost` interrupt is also issued, the frame transmission continues.

Another error concerning  $\overline{\text{CTS}}$  operation occurs when  $\overline{\text{CTS}}$  is not asserted within a reasonable time period after the MLAPD has asserted `request_to_send (RTS)`. This time period is defined by the `CTS_timeout_threshold` entry in the GCB. When this time period expires, the MLAPD issues a `CTS_timeout_threshold_reached` interrupt. Additional interrupts continue to be issued whenever the time period expires, until  $\overline{\text{CTS}}$  is asserted.

## SECTION 6 RECEIVE PROCESS

The host maintains receive pools that contain free receive frame descriptors with associated data buffers. The host can create up to 8192 receive pools and can assign any number of logical links to the same receive pool. The data buffers in a pool must have a length at least equal to the largest N201-value specified for all the logical links assigned to the pool.

Before the MLAPD can receive frames for a particular logical link, the host must prepare a receive pool and assign the link to the pool via the receive\_pool\_number entry in the link's Logical-Link Table (LLT). Then, as a frame is received for the logical link, the MLAPD fetches the first available receive frame descriptor from the link's receive pool to store the incoming data. When the frame is successfully received, the MLAPD indicates the frame type and stores the associated Logical-Link Identification (LLID) number. Although the linkage to the receive pool is not broken, the frame can now be considered part of a receive queue. All logical links that share the same receive pool will also share a common receive queue. Higher level software is responsible for removing valid frames from the receive queue and eventually returning the frame descriptors to the receive pool.

6

### 6.1 RECEIVE DATA STRUCTURES

The overall receive data structure is shown in Figure 6-1. The receive data structure is logically divided into two portions: a receive queue and a receive pool. When a receive pool is initially created, the receive queue is null. The receive\_pool\_pointer and user\_Rx\_next\_pointer address the first frame descriptor in the pool. When this frame descriptor is used for frame reception, the receive\_pool\_pointer is advanced to point to the next frame descriptor in the pool. At this point, a receive queue is created containing the filled (used) frame descriptor. This frame descriptor is available for collection by the host using the user\_Rx\_next\_pointer.

#### 6.1.1 Receive Pool Format

A receive pool is organized as a linked list of receive frame descriptors with associated data buffers. A data buffer must begin on an even-byte boundary and must contain an even number of bytes, even when N201 is odd. (In the case of a frame containing an odd number of bytes in the data field, the MLAPD performs a word write for a 16-bit data bus to place the last data byte into the data buffer. For an 8-bit data bus, the MLAPD performs two byte writes to place the last data byte into the data buffer. The data written to the even byte of this last memory location will be invalid, and the data\_length word will not include

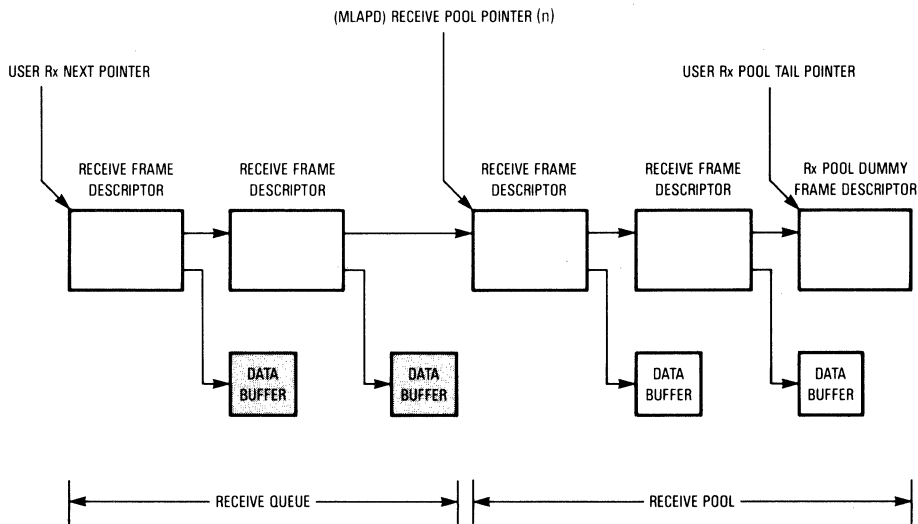


Figure 6-1. Receive Data Structure

this even byte as data.) The format of the receive frame descriptor is described in **2.7 RECEIVE FRAME DESCRIPTOR**.

Each receive pool is assigned an identification number by the host. This identification number is referred to as the `receive_pool_number` in this specification. The `receive_pool_number` ranges from 0 to 8191.

After creating a receive pool, the host specifies the `receive_pool_pointer`, which is the address of the first frame descriptor in the receive pool, and issues an `ASIGN_POOL_POINTER` command to cause the MLAPD to load the `receive_pool_pointer` into the Receive Pool Pointers Table. The MLAPD stores the first sixteen `receive_pool_pointers` (0 to 15) on-chip and stores any additional `receive_pool_pointers` (16 to 8191) in the Receive Pool Pointers Table in shared memory. The address of the memory table is specified by the host in the Global Configuration Block (GCB).

A receive pool must always contain at least one frame descriptor. The last frame descriptor in the pool is the `pool_dummy` frame descriptor, and the `last_in_pool` bit in this frame descriptor's `control_bits` entry is set. When the `receive_pool_pointer` points to this `pool_dummy` frame descriptor, the receive pool is defined to be empty.

The host can add frame descriptors to the receive pool dynamically by using the `user_Rx_pool_tail_pointer` for the receive pool. The `user_Rx_pool_tail_pointer` always points to the `pool_dummy` frame descriptor. The MLAPD does not access this pointer.

To aid in the management of the receive pool, the host may choose to use the `red_line` feature. This feature allows the host to avoid a busy condition on a logical link by dynamically adding frame descriptors when the MLAPD issues a `red_line` interrupt. To implement

this pool management tool, the host sets the red\_line bit in a receive frame descriptor far enough away from the pool's end to retain sufficient receive frame descriptors for acceptance of all incoming information (I) frames in transmission.

### 6.1.2 Receive Queue Format

The receive data structure shown in Figure 6-1 serves as both a receive pool and a receive queue. When a frame descriptor in a receive pool is used for frame reception, the MLAPD updates the pool's receive\_pool\_pointer entry in the Receive Pool Pointers Table. At this point, the used frame descriptor is logically removed from the receive pool and placed in the receive queue, even though no unlinking/linking operation is performed. The head of the receive queue is identified by the user\_Rx\_next\_pointer.

### 6.1.3 User Receive Pointers

The host must maintain two pointers to manipulate each receive data structure: the user\_Rx\_next\_pointer and the user\_Rx\_pool\_tail\_pointer. The host may locate these user pointers in contiguous memory or throughout memory as the MLAPD does not access them. A suggested format for storing the user receive pointers is shown in Figure 6-2.

The 32-bit user\_Rx\_next\_pointer points to the next frame descriptor to be collected by the the host from the receive queue. When the frame\_type field in this frame descriptor is '000', the receive queue is empty.

The 32-bit user\_Rx\_pool\_tail\_pointer points to the last frame descriptor in the receive pool. This frame descriptor is the pool\_dummy frame descriptor. The host dynamically adds frame descriptors to the receive pool starting at the user\_Rx\_pool\_tail\_pointer.

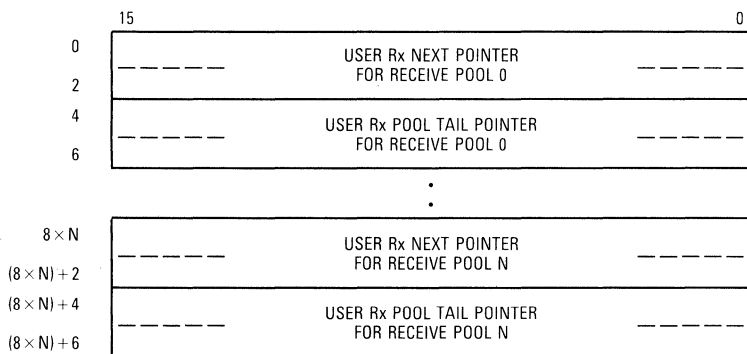


Figure 6-2. User Receive Pointers Table

## 6.2 MLAPD RECEIVE PROCESSING

When the Data-Link-Connection Identifier (DLCI) of an incoming frame identifies an active LLID, the MLAPD performs a look-up process to determine the address of the logical link's LLT. This process is described in detail in **1.5.4 Frame Reception Overview**. Once the MLAPD has located the appropriate LLT, the MLAPD accesses this table to obtain the receive pool\_number. Using this number, the MLAPD fetches the receive\_pool\_pointer from the Receive Pool Pointers Table which identifies the first free frame descriptor in this link's receive pool. The data in the incoming frame is transferred to the frame descriptor's associated data buffer.

After the frame is received successfully, the MLAPD performs the following operations:

- The receive\_pool\_pointer is updated to point to the next free frame descriptor in the pool.
- The data\_length entry is written to indicate the length (in bytes) of the information field of the incoming frame.
- The LLID of the addressed logical link is written into the LLID entry of the frame descriptor.
- The frame\_type entry is set to indicate whether the data is an I frame, an unnumbered information (UI) frame, an exchange identification (XID) command frame, an XID response frame, a frame reject (FRMR) response frame, a nonstandard\_control command frame, or nonstandard\_control response frame.
- The error\_code entry is written if the MLAPD detected a receive frame error that was allowed by the protocol\_receive\_error\_mask.
- If the data\_indication bit is set to one in this frame descriptor, which describes the newly filled buffer, a data\_indication interrupt is generated to inform the host that this receive queue now contains a frame available for collection.

## 6.3 COLLECTING RECEIVED FRAMES

The host uses the user\_Rx\_next\_pointer to collect received frames by traversing the linked list of frame descriptors until a frame descriptor is reached with the frame\_type bits set to 000. This encoding indicates that this frame descriptor still belongs to the receive pool and has not been used for frame reception. After collecting receive frames, the host must update the user\_Rx\_next\_pointer.

The host may set the data\_indication bit in the first frame descriptor in the receive pool to request an interrupt when a receive frame is stored in this frame descriptor's data buffer. The host must set the data\_indication bit before checking the frame\_type field to guarantee that an interrupt is issued when a frame is added to an empty receive queue.

Rather than using the data\_indication bit, the host may choose to use the red\_line bit to provide an indication that a frame has been placed in an empty receive queue. This may save processing time for a system in which the software needs to be notified that a frame

has been received to an empty queue, but frames received subsequent to that do not need notification (until the queue has been re-emptied by the user). In this case, a data\_indication interrupt does not need to be enabled when each receive frame descriptor is filled. Once the host is informed that the queue is not empty, the software empties the receive queue of all frames that have been received.

The receive queue begins at the frame descriptor identified by the user\_next\_Rx\_pointer. When creating a receive pool, the host must set the red\_line bit in the second frame descriptor in the receive queue. When adding a frame to an empty pool, the red\_line bit is set in the dummy frame descriptor. The red\_line bit need not be set in an empty receive pool.

When collecting received frames, the user must perform two operations for each frame checked. First, the red\_line bit in the frame descriptor following the head-of-receive queue is set, then the frame\_type in the head-of-receive queue is checked. Two scenarios are possible:

1. If the current frame is filled before the red\_line bit of the following frame descriptor has been set, the software will recognize the frame\_type of the current frame as one of a received frame and collect it. A red\_line interrupt will not be generated.
2. If the MLAPD receives the next frame after the red\_line bit is set, a red\_line interrupt will be issued for that queue. This is interpreted by the software as meaning that the receive queue is no longer empty.

The data\_indication bit is not needed when the red\_line bit is used to inform the host when the receive queue is not empty. The data\_indication bit cannot be used to provide the "receive-to-empty queue" feature. This inability exists because of differences in the implementation of this bit and the red\_line bit by the MLAPD. Using the data\_indication bit as described above could lead to a situation where the user requests an interrupt for a receive queue believed to be empty and the MLAPD has received a frame to that queue for which no data\_indication interrupt has been requested.





## SECTION 7

### EXCEPTION PROCESSING

When the MLAPD determines that a nonmasked interrupt condition has occurred, an interrupt indication is written into the Interrupt Queue in shared memory. An external interrupt request signal is also asserted when enabled by the host. The host is responsible for collecting interrupt events expeditiously to avoid an interrupt queue overflow.

#### 7.1 INTERRUPT MECHANISM

According to LAPD procedures, the MLAPD must provide event indication and confirmation. Additionally, the MLAPD must have the ability to report events concerning its system environment, such as the MLAPD-to-host interface and the MLAPD-to-memory interface. To support these requirements, the MLAPD implements an interrupt mechanism consisting of an Interrupt Queue, interrupt/polling mode selection, interrupt vector number, interrupt-queue-mask, and interrupt-queue-write-pointer.

7

##### 7.1.1 Interrupt Queue Handling

The interrupt queue is described in **2.8 INTERRUPT QUEUE**. The interrupt-queue-write-pointer is used by the MLAPD to manage the Interrupt Queue. This pointer is stored on-chip and is not accessed by the host. As an entry is filled, the pointer is advanced. Since the Interrupt Queue is cyclical, the pointer is reinitialized to point to the top of the queue after the last entry in the queue is filled.

Upon detection of an interrupting event, the MLAPD checks the interrupt source against the interrupt-mask entry in the Global Configuration Block (GCB). If this condition is not masked, the MLAPD enters the interrupt indication into the Interrupt Queue. If the interrupt-queue-write-pointer register points to an entry with the valid-entry bit set, indicating that it has not yet been removed by the host, the Interrupt Queue has overflowed. (See **7.1.4 Interrupt Queue Overflow**. In this case, the previous entry is overwritten with an interrupt-queue-overflow interrupt indication. If the entry is free, the MLAPD writes the LLID (if applicable) and the cause word with the valid-entry bit set to one. Then, in the interrupt-driven mode, the MLAPD asserts  $\overline{\text{IRQ}}$  (INTR), if it is not already asserted due to a prior interrupting event.

##### 7.1.2 Selecting Polled or Interrupt-Driven Operation

The MLAPD interrupt scheme supports systems which use a polling method and systems which use an interrupt-driven method for interrupt handling. The MLAPD exits reset in

the interrupt-driven mode. The host then specifies interrupt-driven or polling operation for interrupt handling as part of the initialization sequence. (See **2.1.1.1 OPTION BITS 1**). When the interrupt-driven mode is selected and vectored interrupts are to be implemented, the host must also initialize the interrupt\_vector register (IVR).

In a polled system, interrupt indications are written into the Interrupt Queue in shared memory, and the host services this queue periodically. The MLAPD does not request interrupt service via an external bus signal.

In an interrupt-driven system, the MLAPD writes the interrupt indication into the Interrupt Queue and also issues an interrupt request by asserting  $\overline{\text{IRQ}}$  (INTR). If an interrupt acknowledge cycle is initiated by the host, the MLAPD responds with an interrupt vector number which specifies the interrupt type. The most significant bits, 7–1, of the interrupt vector number are programmed by the host, while the least significant bit is encoded by the MLAPD to indicate a normal or severe interrupt. The interrupt handshake signals are  $\overline{\text{IRQ}}$  and  $\overline{\text{IACK}}$  (INTR and  $\overline{\text{INTA}}$ ). The interrupt acknowledge cycle is described in detail for Motorola systems in **11.1.2 Interrupt Acknowledge Cycle** and for Intel compatible systems in **11.2.2 Interrupt Acknowledge Cycle**.

## 7

### 7.1.3 Collecting Interrupt Events

A user\_interrupt\_queue\_read pointer must be maintained by the host to store the next Interrupt Queue entry to be read. This pointer is initialized with the address of the beginning of the Interrupt Queue.

To collect interrupt information from the Interrupt Queue, the host first determines that an entry is valid by inspecting the valid\_entry bit in the second word of the interrupt entry. After each valid entry is read, the host zeroes the valid\_entry bit allowing the MLAPD to reuse the entry to report new interrupting conditions. The host then updates the user\_interrupt\_queue\_read\_pointer. Once the host reads a valid\_entry bit that is set to zero, then all pending interrupts have been serviced.

In the interrupt-driven mode, the host must issue an ENABLE\_IRQ command to cause  $\overline{\text{IRQ}}$  (INTR) to be negated. The possibility exists that some additional interrupt indications may have been written into the Interrupt Queue by the MLAPD after the host determined that the queue was empty but before it issued the ENABLE\_IRQ command. To avoid missing these interrupting events, the MLAPD inspects the valid\_entry bit in the last entry which it wrote into the queue on reception of the ENABLE\_IRQ command. If this entry has not been handled, the MLAPD reissues an interrupt request by asserting  $\overline{\text{IRQ}}$  (or INTR).

### 7.1.4 Interrupt Queue Overflow

Information passed to the host through the interrupt mechanism is not stored explicitly in any other location. The host is responsible for retrieving this information at a rate compatible with the memory resources allocated to the Interrupt Queue. If interrupt information is not removed quickly enough, new interrupt information may be lost.

If the Interrupt Queue overflows, the MLAPD notifies the host, by overwriting the last entry in the queue with an interrupt-queue-overflow interrupt indication. In the unusual case that the Interrupt Queue overflows and that the interrupt event causing the overflow was a bus/address error, the MLAPD notifies the host by overwriting the last entry in the queue with a bus-error interrupt indication (not an interrupt-queue-overflow interrupt indication).

## 7.2 BUS ERROR OPERATION

The MLAPD enters the bus error condition when the  $\overline{\text{BERR}}$  pin is asserted. The MLAPD asserts the  $\overline{\text{IRQ}}$  (INTR) signal, regardless of the programmed value of the polling-select bit in the option-bits-1 GCB entry. (An external interrupt indication is always provided for a bus error condition because the bus error may have occurred during an access to the Interrupt Queue. So, no guarantee can be given to ensure that a bus error interrupt entry can always be reported via the Interrupt Queue.) The MLAPD also places a nonmaskable bus/address-error interrupt on the Interrupt Queue. If an interrupt acknowledge cycle is then initiated by the host, the MLAPD reports a severe interrupt request by passing an interrupt vector number with the least significant bit set to one.

Internally, the MLAPD enters the bus/address error state in which it ignores all commands except an OFF-LINE or RESET command. When the host issues any command other than OFF-LINE or RESET, the MLAPD immediately sets the semaphore register (SR) to hex 'FF' without executing this command.

7

After issuing an OFF-LINE command, the host can determine the address which caused the access error, together with other relevant arguments by issuing a DUMP command. Recommendation is made that the user then inspect the MLAPD shared memory structures to gather additional information before issuing INIT.

## 7.3 ADDRESS ERROR OPERATION

An address error occurs when either  $\overline{\text{CS}}$  or  $\overline{\text{IACK}}$  ( $\overline{\text{INTA}}$ ) is asserted during an MLAPD bus master cycle. The MLAPD exception processing for an address error is the same as described for a bus error. See **7.2 BUS ERROR OPERATION**.



## SECTION 8

### MLAPD IMPLEMENTATION OF LAPD

This section describes the interaction between the host and the MLAPD in implementing a LAPD node. Following the MLAPD initialization process, details are provided on logical link initialization, setup, and release. The remainder of this section details the MLAPD implementation of the LAPD procedures for frame transmission and reception. The reader should refer to the CCITT Q.920/Q.921 recommendation for a complete description of the LAPD procedures. Table 8-1 identifies the LAPD link states referenced in the following paragraphs.

**Table 8-1. List of Link States**

Numeric State Value	State Name	TEI Assignment
1	TEL_UNASSIGN	TEI Unassigned State
2	ASSIGN_WAIT_TEI	Assign Waiting TEI State
3	EST_WAIT_TEI	Establish Waiting TEI (to be assigned) State
Numeric State Value	State Name	Multiframe Definition
4	TEL_ASSIGNED	TEI Assigned State
5	AWAIT_EST	Awaiting Establishment State
6	AWAIT_REL	Awaiting Release State
7.0	MF_EST_NORM	Multiple Frame Established (Normal State)
7.1	MF_EST_REJ	Multiple Frame Established (Reject State)
7.2	MF_EST_BUSY	Multiple Frame Established (Busy State)
8.0	TM_REC_NORM	Timer Recovery (Normal State)
8.1	TM_REC_REJ	Timer Recovery (Reject State)
8.2	TM_REC_BUSY	Timer Recovery (Busy State)

The MF\_EST\_NORM, MF\_EST\_BUSY, MF\_EST\_REJ, TM\_REC\_NORM, TM\_REC\_BUSY, and TM\_REC\_REJ states are referred to as the connected states.

#### 8.1 MLAPD INITIALIZATION PROCEDURE

During initialization, system configuration information, the MLAPD interrupt vector number, and the Global Configuration Block (GCB) address are loaded by the MLAPD under the direction of the host, as shown in the sample program below. Internal registers directly accessed during the initialization procedure are the command register (CR), semaphore register (SR), interrupt\_vector register (IVR), and data register.

1. RESET
  - 2a. Repeat : Read SR until it is the hex value 'FF'
  - 2b. Write CR : SET\_BUS\_WIDTH\_16,(8)
  3. Write IVR : Interrupt vector number
  - 4a. Repeat : Read SR until it is the hex value 'FF'
  - 4b. Write DR : GCB address
  - 4c. Write CR : INIT

After initialization, the MLAPD is in the on-line state. No logical link is active when the initialization process is completed. The host can write any command to the MLAPD. However, it is always necessary for the host to check the SR for the hex value 'FF' to ensure that the MLAPD is ready to accept the command.

## 8.2 INITIALIZATION OF LOGICAL LINKS

After powerup, the MLAPD has no association with logical addresses for any link. To support an ISDN environment for signaling and packet-mode bearer services, a broadcast channel must first be established. This broadcast channel supports the negotiation procedure by which a logical address may be assigned for the signaling channel. Once the signaling channel has a Data-Link-Connection Identifier (DLCI) assigned, and an establishment procedure has established a logical connection, the user-to-network signaling negotiation will determine the DLCIs to be used for the assignment of the bearer logical-link connections. Figure 8-1 shows a simple overview to the steps of address assignment.

### 8.2.1 Broadcast Link Initialization

To establish a broadcast channel, the host prepares a Logical-Link Table (LLT) and chooses a Logical-Link Identification (LLID) number that will be associated with this link. (Level 3

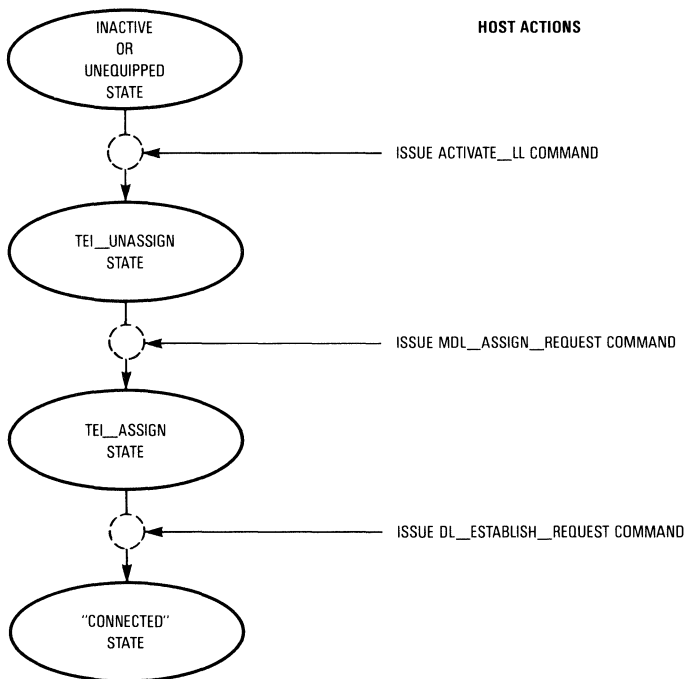


Figure 8-1. MLAPD Protocol States

defines a 13-bit LLID corresponding to each active DLCI. Translation to an LLID, which is only of local significance to the link-level process, serves to reduce the external memory requirements.) The host records the address of the LLT in the appropriate entry of the LLID-LLT Table. This entry is determined by using the LLID as a displacement from the beginning LLID-LLT Table address. The values assigned to parameters in the LLT (N201, K, etc.) are those defined as the default parameters for a signaling logical link. The DLCI entry in the LLT is programmed as 8191, the address assigned to the broadcast link by the LAPD protocol. The host then prepares a receive pool and issues an ASSIGN\_POOL\_POINTER command.

Next, the host issues the ACTIVATE\_LL command. In response, the MLAPD fetches the LLT address from the LLID-LLT Table and reads the DLCI entry in the link's LLT. If the MLAPD is operating in the on-chip system operation mode, the MLAPD stores the DLCI and associated LLID into a free content addressable memory (CAM) location. Alternatively, if the MLAPD is operating in the expanded-system operation mode, the MLAPD stores the LLID in the external Match Table. This Match Table entry is identified by using the DLCI as an offset into the table (refer to **2.2 MATCH TABLE**). Finally, the MLAPD places the link in the TEI\_UNASSIGN state (link state 1). The broadcast link is now in a protocol-defined state which allows the exchange of connection management information using the un-numbered information (UI) frame as part of a terminal endpoint identifier (TEI) assignment procedure.

**8.2.1.1 BROADCAST LINK INITIALIZATION PROCEDURE.** The following steps are for broadcast address assignment:

- 5a. Repeat: Read SR until it is the hex value 'FF'
- 5b. Write arguments (receive pool number and receive pool pointer) into the command\_arguments area in the GCB.
- 5c. Write CR : ASSIGN\_POOL\_POINTER
- 6a. Repeat : Read SR until it is the hex value 'FF'
- 6b. Write argument (LLID) into the command\_arguments area in the GCB.
- 6c. Write CR : ACTIVATE\_LL

## 8.2.2 Signaling Link Initialization

Once a broadcast channel is active, the negotiation procedure begins for the assignment of a logical address for the signaling channel. As part of the negotiation procedure, the connection management entity sends an identity request message to its peer network management entity. This message is placed into a data buffer and linked to a transmit frame descriptor. The host encodes the frame\_type field of the frame descriptor to indicate a MDL\_UI frame. Next, the host issues a GLOBAL\_XID/UI\_REQUEST command to enable transmission of this frame. As the MLAPD processes this transmit queue, the frame is sent as a UI frame.

After receiving this identity request message, the peer network management entity responds with an identity-assigned message, containing the DLCI to be assigned to the signaling link. This message is also transmitted as a UI frame over the broadcast channel. When the originating MLAPD receives the UI frame, the frame is stored in a receive data

buffer identified by the link's assigned Receive Pool Pointer. The frame\_type field of the receive frame descriptor is encoded as a UI frame, and the MLAPD issues a data\_indication interrupt.

The management entity then extracts the DLCI value contained in this identity-assigned message, creates a LLT for the signaling link, associates an LLID with this DLCI, and stores the address of the LLT in the LLID-LLT Table. Another receive pool may optionally be created to service this link. Finally, the host issues an MDL\_ASSIGN\_REQUEST command. In response, the MLAPD stores the LLID in the Match Table or stores the DLCI-LLID pair in the on-chip CAM, depending on the system operation mode. This link's state is set to TEI\_ASSIGNED (state 4).

The host may now follow the link setup procedure to establish the signaling connection. See **8.4 LINK SETUP PROCEDURE**. Once the signaling link enters the connect mode, negotiations can occur to assign DLCIs for various bearer logical-link connections. When a DLCI is assigned for a logical link, the host performs an assignment procedure to activate the link.

### 8.3 ASSIGNMENT PROCEDURE

Before any frame can be transmitted via a logical link, an assignment process must take place which associates a specific DLCI with the logical link. When the assigned address is known, the host prepares a Logical-Link Table (LLT) for the link and records the LLT pointer in the appropriate entry of the LLID-LLT Table. Then, the host prepares a receive pool and issues an ASSIGN\_POOL\_POINTER command to the MLAPD. Finally, the host issues an MDL\_ASSIGN\_REQUEST command for the logical link.

When in the expanded-system operation mode, the MLAPD then responds by writing the LLID into the corresponding entry of the Match Table. When in the on-chip system operation mode, the MLAPD responds to the MDL\_ASSIGN\_REQUEST by loading the DLCI and its associated LLID into the on-chip CAM. If no entry is available in the on-chip CAM, the MLAPD issues a CAM\_over\_flow interrupt. Figure 1-9 illustrates the above operation. Finally, if the link was in the TEI\_UNASSIGN state (link state 1), the MLAPD changes the logical link's internal state from the TEI\_UNASSIGN state to the TEI\_ASSIGNED state (link state 4).

A DL\_ESTABLISH\_REQUEST may be issued for a new link before that link was assigned by an MDL\_ASSIGN\_REQUEST command, according to the LAPD protocol. For more details see **8.9 FRAME TYPES ALLOWED IN EACH LINK STATE**.

### 8.4 LINK SETUP PROCEDURE

A link setup, or establishment, procedure is performed for a previously assigned link to enter connect mode for multiple frame operation. A successful link setup consists of the exchange of set asynchronous balanced mode extended (SABME) and unnumbered acknowledgement (UA) frames, while an unsuccessful link setup consists of the exchange



of SABME and disconnected mode (DM) frames. A link setup is initiated in one of two ways:

- The MLAPD receives a SABME command frame from the remote station; or
- The MLAPD executes a DL-ESTABLISH-REQUEST command from the host.

When the MLAPD receives a SABME command frame from the remote station and the logical link can enter the information transfer phase, the MLAPD transmits a UA response frame. The MLAPD zeroes the retransmission\_count, the remote\_busy flag, and the link variables: send\_state\_variable, V(S), receive\_state\_variable, V(R), and acknowledge\_state\_variable, V(A). Then, the MLAPD restarts data link activity timer T203. The MLAPD updates its internal highest\_active\_LLID register, if necessary. The MLAPD moves the logical link to the MF-EST-NORM state and issues a DL-establish-indication interrupt for the link using the interrupt mechanism described in **7.1 INTERRUPT MECHANISM**. If a SABME command is received while the logical link is in the AWAIT-REL state (link state 6), then the logical link cannot enter the MF-EST-NORM state, and the MLAPD transmits a DM response frame.

When the MLAPD executes a DL-ESTABLISH-REQUEST command from the host, a SABME command frame is sent to the remote station by the MLAPD. The acknowledgement timer, T200, is started to determine when retransmission of the SABME frame should be initiated if no response is received. The MLAPD updates its internal highest\_active\_LLID register, if necessary, and the MLAPD zeroes the retransmission\_count. Then, the logical link changes its internal protocol state to the AWAIT-EST state (link state 5).

When a UA response frame is received while in this AWAIT-EST state, the logical link enters the MF-EST-NORM state, and the MLAPD issues a DL-establish-confirmation interrupt for this link using the interrupt mechanism. The MLAPD zeroes the remote\_busy flag and the link variables: V(S), V(R), and V(A). Alternatively, if a DM response frame is received, the MLAPD stops T200 and issues a DL-release-indication interrupt. All frames other than UA, DM, SABME, and disconnect (DISC) are ignored while in the AWAIT-EST state. The reception of a DISC or SABME frame is a collision of unnumbered command frames as discussed in **8.6 COLLISION OF UNNUMBERED COMMAND FRAMES**.

If T200 expires before a response frame is received, the MLAPD retransmits the SABME frame, restarts T200, and increments the retransmission\_count in the link's timer entry. When the number of retransmission attempts reaches the N200\_value, the MLAPD stops T200 and issues a DL-release-indication and an MDL-error-indication interrupt for this link.

The commands MDL-ASSIGN-REQUEST and DL-ESTABLISH-REQUEST may be issued in any state to place a new link into the MF-EST-NORM state.

## 8.5 LINK RELEASE PROCEDURE

A link release procedure is performed for a previously connected logical link to terminate multiple frame operation. A link release procedure consists of the exchange of DISC and

UA frames or DISC and DM frames. A link release procedure is initiated in one of two ways:

- The MLAPD receives a DISC command frame from the remote station; or
- The MLAPD executes a DL-RELEASE-REQUEST command from the host.

When the MLAPD receives a DISC command frame from the remote station, the MLAPD aborts any reception or transmission of information (I) frames for this link and transmits a UA response frame. The MLAPD stops any running timer, issues a DL-release-indication interrupt, and changes the link state to TEI-ASSIGNED (link state 4). If the DISC command frame is received while the logical link is in the AWAIT-EST state (link state 5), the MLAPD transmits a DM response frame without any change in the link state.

When the MLAPD executes a DL-RELEASE-REQUEST command from the host, a DISC command frame is sent to the remote station by the MLAPD. The acknowledgement timer, T200, is started to determine when retransmission of the frame should be initiated if no response is received. The MLAPD updates its internal highest-active-LLID register, if necessary, and the MLAPD zeroes the retransmission-count. Then, the logical link changes its internal protocol state to the AWAIT-REL state (link state 6).

When a UA response frame or a DM response frame is received while in this AWAIT-REL state, the logical link enters the TEI-ASSIGNED state. The MLAPD stops T200 and issues a DL-release-confirmation interrupt. Frames other than UA, DM, SABME, and DISC are ignored while in the AWAIT-REL state. The reception of a DISC or SABME frame is a collision of unnumbered commands as discussed in **8.6 COLLISION OF UNNUMBERED COMMAND FRAMES**.

If T200 expires before a response frame is received, the MLAPD retransmits a DISC frame, restarts T200, and increments the retransmission-count in the link's timer entry. When the number of retransmission attempts reaches the N200-value, the MLAPD stops T200 and issues a DL-release-indication and an MDL-error-indication interrupt for this link.

Once in the TEI-ASSIGNED state, the link can receive and transmit user frames as described in **8.9 FRAME TYPES ALLOWED IN EACH LINK STATE**. To further limit link activities, the host issues an MDL-REMOVE-REQUEST command to transfer the link to the TELUNASSIGN state. To stop all link activities regardless of the link's current LAPD protocol state, the host issues a DEACTIVATE-LL command. Following this command, the SR value must return to hex 'FF', and the active bit in the transmit-status LLT entry must be set to zero by the MLAPD before the memory space allocated for the link's LLT is free for host manipulation.

A link can also exit the connect mode:

- By an MDL-REMOVE-REQUEST command, which changes the link-state to TELUNASSIGN.
- By a DEACTIVATE-LL command, which changes the link-state to uninitialized (0000).

In normal operation, recommendation is made not to allow a link to exit the connect state via the MDL-REMOVE-REQUEST or DEACTIVATE-LL command as the remote link has no way to know that its peer link is no longer in the connect mode. For more details see **SECTION 4 COMMAND SET**.

## 8.6 COLLISION OF UNNUMBERED COMMAND FRAMES

If the sent and received unnumbered command frames are the same (SABME/SABME or DISC/DISC), the MLAPD and the remote station send a UA response at the earliest opportunity. The logical link then enters the requested operational mode after the MLAPD receives the UA response from the remote station.

If the sent and received unnumbered command frames are different (SABME/DISC or DISC/SABME), the MLAPD and the remote station transmit a DM response at the earliest opportunity. If the MLAPD receives a DM frame with the final (F) bit set to one after transmitting a DISC frame, the MLAPD issues a DL-release-confirmation interrupt and moves the link from the AWAIT-REL state (link state 6) to the TEL-ASSIGNED state (link state 4). If a DM frame with the F bit set to one is received after transmitting a SABME frame, the MLAPD issues a DL-release-indication interrupt and places the logical link in the TEL-ASSIGNED state.

## 8.7 TRANSMIT PROCEDURE

Frame preparation and transmission procedures are outlined in this segment. Procedures are defined for I frames and XID/UI (nonstandard-control) frames.

### 8.7.1 Information Frame Preparation

When the host has data ready for transmission on a logical link, the host prepares one or more frame descriptors which contain information required by the MLAPD to transmit the associated frame(s). The host links the frame descriptors to the desired logical link's Transmit Queue. If the logical link's Transmit Queue is empty when the new frame descriptor(s) are linked to the queue, the host must issue a DL-DATA-REQUEST for this logical link and specify the head of the Transmit Queue. When this DL-DATA-REQUEST is issued, the MLAPD links the corresponding Transmit Queue to its assigned I frame queue. If frames are waiting for transmission for the logical link when the new frame descriptor(s) are added, the linking of the frame descriptors is sufficient for the associated frame(s) to be transmitted. If all the frames for this logical link have been transmitted but some frames are still awaiting acknowledgement, the host must issue a RELINK-REQUEST command for this logical link to enable transmission of the new frame(s).

### 8.7.2 Information Frame Transmission

The MLAPD transmits I frames according to the transmit servicing scheme described in **5.1 TRANSMIT SERVICING SCHEME**. When an information frame is to be transmitted, the MLAPD first fetches the logical link's context information from the link's LLT. This information is used to build the DLCI and control fields for the frame, which are then placed in the transmit first-in first-out (TxFIFO) buffer. Next, the MLAPD uses information contained in the frame descriptor to locate the data to be sent and then transfers this information into its TxFIFO. The MLAPD transmits the frame and attaches a frame check sequence to complete the frame. Zeros are inserted when necessary for transparency, if the

zero-insertion-select bit in the GCB is set to one. After frame transmission, V(S) is updated, and T200 is started if it is not already running.

Transmission begins when six data bytes are present in the TxFIFO, in addition to the DLCI and control field, or when the entire frame is present in the TxFIFO (when the entire frame is less than six bytes). Between frames the MLAPD transmits the user selected number of pad flags. Additional flags may be transmitted until the requirements for start of transmission are met. While transmitting an information frame, the MLAPD requests use of the system bus when at least four empty bytes are available in the TxFIFO. (The TxFIFO contains 20 words).

The MLAPD aborts an I frame when:

- A TxFIFO underrun occurs;
- A STOP\_TX-I command is executed and an I frame is in-transmission for the specified link;
- A REJ frame is received and an I frame in-transmission for this link;
- The MLAPD resets a link (due to a link condition, host command, etc.) and an I frame is in-transmission for this link; or
- A bus error occurs (the transmitter is reset,  $\overline{RTS}$  is negated, and TxD is three-stated).

The MLAPD continues transmitting frames for the same logical link until one of the following events occur:

- The logical link's Transmit Queue is exhausted;
- The maximum number of outstanding I frames (K) for this logical link is reached;
- The MLAPD transmitter switches to service another queue according to the transmit servicing scheme;
- A receive-not-ready (RNR) frame is received from the remote station;
- A supervisory (S) frame with poll (P) bit set to one is transmitted by this logical link;
- A DL\_data\_confirmation is issued when all frames in a logical link's Transmit Queue have been acknowledged.

### 8.7.3 XID/UI/Nonstandard\_control Frame Preparation

When the host has XID/UI (or nonstandard\_control) information for transmission on a logical link, the host prepares one or more frame descriptors which contain the information required by the MLAPD to transmit the frame(s). The host links the frame descriptors to the Global XID/UI Transmit Queue, to XID/UI Queue\_0, or to XID/UI Queue\_1. The host sets the frame\_type bits in the frame descriptor to specify whether the frame is to be transmitted as an MDL\_UI, DL\_UI, XID command, XID response, nonstandard\_control command, or nonstandard\_control response frame. If the queue is empty when the new frame descriptor(s) are linked to the queue the host must issue an appropriate XID/UI\_REQUEST command with the pointer to the first frame descriptor stored in the command\_argument\_2 field of the GCB. When this request is issued, the MLAPD then places the specified XID/UI Queue into the transmit servicing mechanism. If any frames are waiting for transmission when the new XID/UI transmit frame descriptor(s) are added, the linking of the frame descriptor(s) is sufficient for the associated frame(s) to be transmitted.

### 8.7.4 XID/UI/Nonstandard\_control Frame Transmission

The MLAPD services each XID/UI queue according to the transmit priority mechanism. The MLAPD locates the link's LLT, using the LLID entry in the frame descriptor to index into the LLID-LLT Table, to obtain the appropriate DLCI for the XID/UI (or nonstandard\_control field) frame. The MLAPD places this DLCI and the correct control field into its TxFIFO. Next, the address of the data buffer is read from the frame descriptor, and the MLAPD transfers this data into the TxFIFO. As the frame is transmitted, zeros are inserted when necessary for transparency. The cyclical redundancy check (CRC) is calculated and appended to complete the frame.

Transmission begins when six data bytes are present in the TxFIFO, in addition to the DLCI and control field, or when the entire frame is present in the TxFIFO (when the entire frame is less than six bytes). Between frames, the MLAPD transmits the user selected number of pad flags. Additional flags may be transmitted until the requirements for start of transmission are met. While transmitting a frame, the MLAPD requests the system bus when at least four empty bytes are available in the TxFIFO. (The TxFIFO contains 20 words.)

The MLAPD aborts a XID/UI (or nonstandard\_control) frame when:

- A TxFIFO underrun occurs;
- A STOP\_GLOBAL\_XID/UI, STOP\_XID/UI\_QUEUE\_0, or STOP\_XID/UI\_QUEUE\_1 command is executed and a frame is in-transmission for the specified queue; or
- A bus error occurs (the transmitter is reset,  $\overline{\text{RTS}}$  is negated, and TxD is three-stated).

A Global\_XID/UI\_confirmation, XID/UI\_Queue\_0\_confirmation, or XID/UI\_Queue\_1\_confirmation interrupt is issued when all frames from the specified queue have been transmitted.

## 8.8 RECEPTION PROCEDURE

Procedures are outlined for reception of I frames and XID/UI (and nonstandard\_control) frames. Also, receive errors are defined, including invalid frame conditions and frame reject conditions.

### 8.8.1 Information Frame Reception

Information frame reception is enabled for a specific logical link by performing the link setup procedure. Once the link is established, the MLAPD is ready to receive frames containing the associated DLCI.

For information frames that are not treated as "invalid frames" (see following section on "Invalid Frames Reception") or as "bad frames" (see section on "Bad Frames Reception"), the MLAPD proceeds to locate a free buffer for the frame's information field using the link's receive\_pool\_number to index into the Receive Pool Pointers Table. If the receive pool associated with this logical link is not empty and the link's local\_busy flag is not set, then the information field is transferred through the receive FIFO into the receive buffer. Finally,

the MLAPD updates the LLT state variable V(R) and acknowledges the frame reception by queuing a RR frame on the Level 2 Queue. The MLAPD uses S frames to acknowledge information frames rather than piggy-backing acknowledgments because it is impossible to know when the next information frame belonging to this logical link will be transmitted.

If the receive pool associated with this logical link is empty or the link's local\_busy flag is set, then a Level 2 frame (RNR) is queued for transmission and the information field is ignored. In this case, the logical link will be in the MF\_EST\_BUSY state or TM\_REC\_BUSY state.

Zero deletion is performed throughout the reception process based on the zero\_deletion\_select bit in the option bits 1. The MLAPD requests use of the system bus when at least four bytes are in the RxFIFO. Frames are received in sequence as long as memory buffers are available and adequate bandwidth is provided for DMA on the system bus.

### 8.8.2 Invalid Frames Reception

An invalid frame is a frame which:

- a) is not properly bounded by two flags
- b) has fewer than 6 octets between flags of frames with sequence numbers (I frames and S frames), or fewer than 5 octets between flags of frames without sequence numbers (U frames)
- c) does not consist of an integral number of octets after zero deletion
- d) contains a CRC error

The MLAPD also treats the following frames as invalid:

- Frames for which a RX\_FIFO overflow occurred during reception
- Frames with inactive DLCI
- Frames with invalid address (error in the address field extension bits)

In compliance with the LAPD protocol, the invalid frame is discarded without notification to the sender. No protocol action is taken as the result of this frame.

Note that the error\_mask\_valid bit entry in each LLT allows the host to selectively receive invalid frames for a logical link, according to the protocol\_error\_mask entry in the GCB. Also the MLAPD provides individual error counters for some types of invalid receive frames.

The following types of invalid frames can be received:

- Aborted frames
- Frames with CRC error
- Frames in which a RX\_FIFO overflow occurred during reception

The following types of invalid frames have an error counter associated with them:

- RX\_FIFO overrun
- Aborted frames
- Frames with a CRC error
- Short frames
- Inactive DLCI
- Invalid address

### 8.8.3 Bad Frames Reception

A bad frame is defined by the LAPD protocol as an error-free frame (a frame that is not an invalid frame) that contains:

- An undefined control field
- An information field which is not permitted (U or S frame)
- An information field length which is not equal to 5 in FRMR frame
- An invalid N(R)
- An information field which is longer than N201 (I, UI, or XID frames)

In compliance with the LAPD protocol, the bad frame is discarded. The MLAPD then executes the link setup procedure:

- queues a FRMR response frame (depending on the Tx\_FRMR\_select bit in the option\_bits\_1 entry of the GCB)
- queues a SABME frame
- changes the logical link's state to AWAIT\_EST
- issues an MDL\_error\_indication interrupt to the host with the appropriate argument

A frame with an information field longer than N201 can be received if the error\_mask\_valid bit entry in the LLT and the N201 bit in the protocol\_error\_mask entry in the GCB are set. In that case, only the first N201 bytes of the frame will be written to the data buffer and the rest of the frame will be discarded.

A frame with an invalid N(R) and a correct N(S) will be received if the CCITT/DMI mode bit in the option\_bits\_2 entry of the GCB is set to one.

### 8.8.4 Frame Reject Mode

The MLAPD checks the frame reject mode on reception of a bad frame (see sub chapter "Bad Frames Reception"). The MLAPD queues a FRMR response frame only if the Tx\_FRMR\_select bit in the option\_bits\_1 entry of the GCB is set to one. All the other protocol actions (see sub chapter "Bad Frames Reception") are always executed.

### 8.8.5 XID/UI/Nonstandard\_control Frame Reception

XID, UI, and nonstandard\_control field frame reception is the same as I frame reception. Received XID, UI, and nonstandard\_control frames are placed in the logical link's receive queue. To differentiate between types of received frames in the receive queue, the MLAPD encodes the frame\_type bits in the associated receive frame descriptor. (Refer to Figure 2-15.)

### 8.8.6 Receiving an Out-Of-Sequence I Frame

When the MLAPD receives a valid I frame with N(S) not equal to the LLT state variable V(R), the information field is discarded. If the addressed logical link is in the MF\_EST\_NORM state or TM\_REC\_NORM state, the MLAPD queues a REJ frame to the Level 2 frame queue

and enters the REJ condition for the affected link. A REJ condition is cleared when the requested I frame is received. If the addressed logical link is in the MF\_EST\_BUSY or TM\_REC\_BUSY, the MLAPD places a RNR frame on the Level 2 Queue, and a REJ frame is sent when the link changes to MF\_EST\_NORM or TM\_REC\_NORM.

### 8.8.7 Receiving a FRMR Frame

Upon reception of an FRMR frame, the MLAPD issues an MDL\_error\_indication interrupt to the host (with argument code K). The information field of the FRMR frame is written into the receive queue. The MLAPD also writes the LLID and sets the appropriate status\_bits. However, if the link's receive pool is empty, the MLAPD will be unable to transfer the information field to the host. In this case, the MLAPD issues an MDL\_error\_indication interrupt to the host (with argument code P5). In response to a FRMR, the MLAPD sends a SABME command frame with the P bit set to one, and changes the state of the logical link to AWAIT\_EST.

## 8.9 FRAME TYPES ALLOWED IN EACH LINK STATE

The following sections list the frame types that are allowed by the LAPD protocol in each link state. The method for transition between LAPD states is also noted.

# 8

### 8.9.1 TEL\_UNASSIGN and EST\_WAIT\_TEI States

The user assigns a new link to TEL\_UNASSIGN (state 1) by issuing an ACTIVATE\_LL command. In this state, a DL\_ESTABLISH\_REQUEST command causes a transition to EST\_WAIT\_TEI (state 3).

Also, the user can assign a new link to EST\_WAIT\_TEI (state 3) by a DL\_ESTABLISH\_REQUEST command. The TEL\_UNASSIGN state is used only for management links. In these two states, the link can transmit the following frames:

- MDL\_UI frames
- Nonstandard\_control frames

In these two states, the link can receive the following frames:

- UI frames
- Nonstandard\_control frames

### 8.9.2 TEL\_ASSIGNED, AWAIT\_EST, and AWAIT\_REL States

The user assigns a new link to TEL\_ASSIGN (state 4) by either issuing an MDL\_ASSIGN\_REQUEST command or by issuing an ACTIVATE\_LL command followed by an MDL\_ASSIGN\_REQUEST command.



The user assigns a new link to Awaiting state (state 5) by issuing commands as follows:

- 1) MDL\_ASSIGN\_REQUEST and DL\_ESTABLISH\_REQUEST, in this order;
- 2) DL\_ESTABLISH\_REQUEST and MDL\_ASSIGN\_REQUEST, in this order; or
- 3) ACTIVATE\_LL, DL\_ESTABLISH\_REQUEST and MDL\_ASSIGN\_REQUEST, in this order.

The user can cause a previously connected link (i.e., in connect mode) to transition to the Awaiting-Release (state 6) by issuing an MDL\_RELEASE\_REQUEST command.

In these three states, the link can transmit the following frames:

- MDL-UI frames
- DL-UI frames
- Nonstandard-control frames
- XID frames

In these three states, the link can receive the following frames:

- UI frames
- Nonstandard-control frames
- XID frames

### 8.9.3 Connected States

To assign a new link to MF\_EST-NORM (state 7.0), the link must first enter the Awaiting state (state 5). Then, after reception of a UA response frame from the remote station, the link enters the MF\_EST-NORM state. MF\_EST-NORM is the entry point into the multiple frame established mode. From MF\_EST-NORM, the link may eventually change to the other connected states: MF\_EST-REJ (state 7.1), MF\_EST-BUSY (state 7.2), TM\_REC-NORM (state 8.0), TM\_REC-REJ (state 8.1), or TM\_REC-BUSY (state 8.2).

While in connected mode, the link can transmit the following frames:

- MDL-UI frames
- DL-UI frames
- Nonstandard-control frames
- XID frames
- I frames

While in connected mode (except states 7.2 and 8.2), the link can receive the following frames:

- UI frames
- Nonstandard-control frames
- XID frames
- I frames

### 8.10 FLOW CONTROL AND ERROR CONTROL PROCEDURES

Supervisory (S) frames are used in flow and error control. These frames provide receiver status information and acknowledge received I frames.

### 8.10.1 Local Busy Condition

A busy condition is defined as a temporary inability to accept additional incoming I frames. A logical link enters the local busy condition when one of the following occurs:

- A SET\_LOCAL\_BUSY command is issued by the host for this specific logical link, or the receive pool associated with this logical link has no free buffers.

When a SET\_LOCAL\_BUSY command is issued by the host, the MLAPD sets the local\_busy flag in the specified link's link\_status entry of its LLT and reacts to subsequent receive frames as though no free buffers are available in the link's receive pool. A RNR frame is queued to the Level 2 Queue and the logical link enters the MF\_EST\_BUSY state or the TM\_REC\_BUSY state. The host clears the busy condition by issuing a CLEAR\_LOCAL\_BUSY command. In response, the MLAPD queues a RR or REJ frame to the Level 2 Queue and resumes normal multiple frame operation.

When the receive pool has no free buffers, only the pool\_dummy frame descriptor remains in the pool. When a frame is received for a logical link that is assigned to this receive pool, the MLAPD recognizes the last\_in\_pool bit in the pool\_dummy frame descriptor, and this logical link enters the local busy condition. The MLAPD issues a local\_busy interrupt for this link and queues a RR or REJ frame to the Level 2 Queue. The logical link enters the MF\_EST\_BUSY state or the TM\_REC\_BUSY state.

The MLAPD recognizes that the host has added free buffers to a receive pool when:

- The MLAPD first receives a frame addressed to a link assigned to the receive pool;
- The MLAPD responds to a RR or RNR frame addressed to the link with the P bit set to one; or
- The MLAPD responds to a T203 or T200 timeout condition for this link.

To determine if a specific logical link is in the busy condition, the MLAPD reads the first receive frame descriptor in its associated receive pool. Based upon whether this frame descriptor is the pool\_dummy frame descriptor or a valid frame descriptor, the logical link remains in the local busy condition or exits this condition. When the logical link exits the busy condition, the MLAPD queues a RR or REJ frame to the Level 2 Queue.

### 8.10.2 Awaiting Acknowledgement

The MLAPD starts timer T200 after a frame has been transmitted from a logical link's Transmit Queue to ensure that an acknowledgment for the frame is received before the programmed timeout value is reached. If T200 expires while this logical link is in the MF\_EST\_NORM state (link state 7.0), the MLAPD changes the state of this logical link to the TM\_REC\_NORM state (link state 8.0). If this logical link is in MF\_EST\_REJ state (link state 7.1), the MLAPD changes the state of this logical link to the TM\_REC\_REJ state (link state 8.1). The MLAPD then sends a RR frame with the P bit set to one if receive buffers are available from the buffers pool associated with this logical link. Alternatively, the MLAPD sends a RNR frame with the P bit set to one if no receive buffers are available and changes the state of this logical link to the TM\_REC\_BUSY state (link state 8.2). The MLAPD then increments the retransmission\_count in this link's timer entry and restarts T200.

The MLAPD clears the timer recovery condition when a S frame is received with the F bit set to one and with N(R) within the range of the logical link's current V(A) to V(S), inclusive. The MLAPD then stops T200, updates its send state variable V(S), updates the Tx\_next\_pointer in the LLT, and begins transmission or retransmission, as appropriate.

If T200 expires while the logical link is in the TM\_REC\_xxx state, the MLAPD increments the retransmission\_count for this logical link and transmits the appropriate S command with the P bit set to one. If the retransmission\_count reaches the N200\_value, the MLAPD will initiate a link resetting procedure as described in **8.4 LINK SETUP PROCEDURE** and will issue an MDL\_error\_indication interrupt.

### 8.10.3 Receiving Acknowledgement

When the MLAPD correctly receives an I frame or a S frame for a given logical link, the N(R) contained in this frame acknowledges all I frames previously transmitted with an N(S) up to and including the received N(R) - 1. The MLAPD stops T200 when it receives an N(R) higher than the last received N(R) (acknowledging some I frames) or when the MLAPD receives a REJ frame with the N(R) equal to the last received N(R). If T200 is stopped by the reception of an I, RR, or RNR frame and outstanding I frames are still unacknowledged, the MLAPD will restart T200.

### 8.10.4 Receiving a REJ Frame

When a REJ frame is received for a logical link, the MLAPD sets V(S) and V(A) for this logical link to the value of the N(R) contained in the frame. The MLAPD then updates the Tx\_next\_pointer in the link's LLT, so that the frame(s) to be retransmitted are relinked into the transmit servicing mechanism.

If the REJ frame has the P bit set to one, the MLAPD will send an RR, RNR, or REJ response with the F bit set to one before retransmitting the requested I frame(s).

### 8.10.5 Receiving a RNR Frame

When a RNR frame is received for a logical link, the MLAPD updates the remote\_busy flag and V(A) for this logical link. Until a RR frame is received, MLAPD does not transmit any I frames. UI, XID, and undefined\_control frames can be transmitted. The MLAPD then restarts T200 and, upon its expiration, sends a S frame with P bit set to one. If the remote station responds with a RNR frame with the F bit set to one, indicating the continuance of the busy condition, the MLAPD repeats the above sequence. If the remote station responds with an RR or REJ frame, indicating the clearance of the busy condition, the MLAPD stops T200, clears the remote\_busy flag, and begins (re)transmission as appropriate.

## 8.11 ERROR HANDLING OPTIONS

Options are provided for handling error conditions. The MLAPD will optionally transmit an FRMR frame when a frame reject condition is determined. The user may also select either the CCITT or the DMI error mode.

### 8.11.1 Frame Reject Mode

The MLAPD enters the frame reject mode for a logical link on reception of an error-free frame containing:

- An undefined command or response control field;
- An information field which is not permitted (U or S frame) or a U or S frame with incorrect length (e.g., FRMR with a three-byte I field instead of a five-byte I field);
- An invalid N(R), or
- An information field longer than N201.

The `error_mask_valid` entry in each LLT allows the host to selectively receive invalid frames for a logical link with an information field longer than N201, based on the `buffer_length_exceeded` bit of the `protocol_error_mask` in the GCB. Also the MLAPD allows the user to define a `nonstandard_control` field which can be received without causing a FRMR condition.

When a FRMR condition is detected and the `Tx_FRMR_select` bit in the `option_bits_1` entry of the GCB is set to one, the MLAPD queues a FRMR response frame and immediately proceeds to the link setup procedure without waiting for acknowledgement of the FRMR frame. Regardless of the `Tx_FRMR_select` bit value, the MLAPD issues an `MDL_error_indication` interrupt to the host with the appropriate argument, queues a SABME frame, and changes the logical link's state to `AWAIT_EST`.

### 8.11.2 CCITT/DMI Mode

There are some minor differences between the DMI and CCITT state tables with respect to error handling. The user can configure the MLAPD to implement the DMI or CCITT state table via the `DMI/CCITT_select` bit in the GCB. The differences are as follows:

- 1) Reception of a S response frame with unexpected F bit set to one and N(R) is correct:
 

CCITT	Use the received N(R) to acknowledge the transmitted I frames, Report MDL_error_indication code A
DMI	The N(R) field is verified for N(R) error but is not used to acknowledge the transmitted I frames Report MDL_error_indication code A
- 2) Reception of an I frame with N(R) error and N(S) correct:
 

CCITT	If the receiver is not in a busy condition, Deliver the I field to Level 3 Report MDL_error_indication code J Send RR response frame Link reset (FRMR and SABME)
	If the receiver is in a busy condition, Report MDL_error_indication code J Send RNR response frame Link reset (FRMR and SABME)
DMI	Report MDL_error_indication code J Link reset (FRMR and SABME)

- 3) Reception of an I frame with N(R) error and N(S) error:
  - CCITT If the receiver is in normal condition,
    - Report MDL\_error\_indication code J
    - Send REJ response frame with F equal to P
    - Link reset (FRMR and SABME)
  - If the receiver is in reject recovery condition,
    - Report MDL\_error\_indication code J
    - Send RR response frame with F equal to P
    - Link reset (FRMR and SABME)
  - If the receive is in a busy condition,
    - Report MDL\_error\_indication code J
    - Send RNR response frmae with F equal to P
    - Link reset (FRMR and SABME)
  - DMI Report MDL\_error\_indication code J
    - Link reset (FRMR and SABME)
- 4) Reception of DM response with F equal to 1 in connect states:
  - CCITT No change in state
    - Report MDL\_error\_indication code B
  - DMI Report MDL\_error\_indication code B
    - Link reset (SABME)
- 5) DL\_RELEASE\_REQUEST or MDL\_REMOVE\_REQUEST command issued when the link is in EST\_WAIT\_TEI state:
  - CCITT No change in state
  - DMI Change to TEI\_UNASSIGN
    - MDL\_error\_indication



## SECTION 9

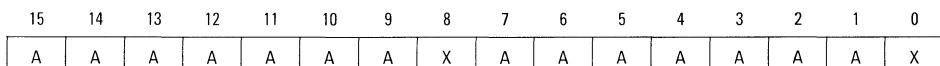
### MLAPD IMPLEMENTATION OF SPECIAL MODES

The MLAPD's primary function is to perform the LAPD procedures on frames entering/exiting its serial, nonchannelized physical level interface. However, the MLAPD can be configured for several alternative modes, which affect the processing of frames or its physical level interface. The nonprotocol, promiscuous receive, and line monitor modes limit the protocol processing performed on received frames. Parallel assist and memory-to-memory modes support alternative interfaces to the physical level. This section describes the purpose of each mode, how the mode is evoked, and the details of each mode's operation.

#### 9.1 NONPROTOCOL LINKS

The host defines a link to be either a protocol or a nonprotocol link. The MLAPD can simultaneously support both types of links. For a protocol, or LAPD link, the MLAPD implements the elements of procedure for link management and guarantees that all frames meet the LAPD frame format. For a nonprotocol link, any elements of procedure are implemented by the host in software as the MLAPD does not apply the LAPD procedures. The frame format is generally considered to contain a 16-bit address field and a 16-bit CCITT-CRC.

Programmable options and MLAPD variations in frame processing minimize the effect of these frame format assumptions. For the receive operation, all receive frames must be at least five bytes in length. The MLAPD uses the first 16-bits of the frame as usual to index into the Match Table (or the on-chip CAM) to determine whether the link is active and whether the link is defined to be a protocol or nonprotocol link. Then, for a nonprotocol link, these first 16 bits are stored in memory along with the remainder of the frame. Bits 0, 1, and 8 are don't care bits and are not required to meet the specified LAPD values. (The address field structure for a nonprotocol link is shown below.) Finally, even though the MLAPD checks the last 16-bits of the frame against a cyclical redundancy check (CRC), the host can opt to receive frames with CRC errors. For the transmit operation, the host programs whether the MLAPD is to insert the link's Data-Link-Connection Identifier (DLCI) into the frame address field and whether the MLAPD is to append a CRC to the frame. Thus, the host may provide the entire transmit frame if desired.



A = Address Bit  
X = Don't Care

Nonprotocol operation uses the same structures as protocol operation. Any differences in bit definitions or feature usage is indicated in the relevant sections of this manual. The following sections describe the interaction between the host and MLAPD for nonprotocol operation and the specific MLAPD processing flows.

### 9.1.1 Setup Procedure

Before any frames may be received or transmitted for a nonprotocol link, an assignment process must take place which associates a specific Data-Link-Connection Identifier (DLCI) with the logical link. First, the host prepares a Logical-Link Table (LLT) for the link. The host specifies that the link will operate in nonprotocol mode by programming the nonprotocol-select bit in the configuration-bits of the link's LLT. Next, the host places the address of the LLT in the appropriate entry of the LLID-LLT Table. This entry is determined by using the Logical-Link Identification (LLID) number as a displacement from the beginning LLID-LLT Table address. (Level 3 defines a 13-bit LLID corresponding to each active DLCI. The LLID is used during link-level processing and is only of local significance. The LLID serves to reduce the external memory requirements for Level 2.) Then the host prepares a receive pool and issues an ASSIGN-POOL-POINTER command to the MLAPD. Finally, the host issues an ACTIVATE-LL command for the link.

### 9.1.2 Release Procedure

At any time, the host may terminate activity on a nonprotocol link by issuing a DEACTIVATE-LL command for the link. The semaphore register (SR) is set to hex 'FF' when the command is accepted. From this point on, no receive or transmit activities are performed for this link. Any current frame transmission for this link is aborted, and any current frame reception for this link is discarded when the command is received.

If the LLT of this link is in its assigned information (I) frame queue, this command is actually completed by the MLAPD transmit task that removes the LLT from the queue. This removal occurs when the LLT reaches the head of its I frame queue. The LLT for this link can be reused by the host only after the link's active bit in its LLT is set to zero by the MLAPD.

If the LLT of this link is not in its assigned I frame queue, the semaphore value hex 'FF' indicates the completion of the command. In either case, when the active bit in the transmit-status entry is set to zero, the command has been completed.

### 9.1.3 Queuing Frames For Transmission

To queue frames for transmission on a nonprotocol link, the host first creates a Transmit Queue. The structure of this Transmit Queue is identical to an I frame Transmit Queue which is used for protocol links. See Figure 5-4. The definition of the fields in a transmit frame descriptor for a nonprotocol link are found in **2.6 TRANSMIT FRAME DESCRIPTOR**.



To enable transmission of the Transmit Queue for a nonprotocol link, the host issues a DL\_DATA\_REQUEST command. The host must maintain the user\_Tx\_next\_confirm\_pointer and the user\_Tx\_last\_queued\_pointer for each logical link's Transmit Queue. These pointers are required for removing transmitted frame descriptors and/or for adding more frame descriptors for transmission.

The host may add frames to the Transmit Queue while frames are still awaiting transmission. When a frame is ready to be added, the host sets the last bit in the new frame descriptor and links the new frame descriptor to the last frame descriptor in the existing Transmit Queue. The last frame descriptor is identified by the user\_Tx\_last\_queued\_pointer. The host then clears the last bit in the frame descriptor that was previously the last frame descriptor in the queue and updates the user\_Tx\_last\_queued\_pointer. Upon receiving a DL\_data\_confirmation interrupt, the host must verify that the last frame added to the queue was indeed transmitted. If it was not, then the addition operation was performed after the MLAPD had checked the last bit in the frame descriptor, which the host intended to clear. In this case, the host must present the remaining frames to the MLAPD as a new Transmit Queue and issue another DL\_DATA\_REQUEST.

For nonprotocol links, the transmit frame descriptor acknowledge bit has no meaning, and so the RELINK\_REQUEST command is meaningless.

#### 9.1.4 MLAPD Transmit Queue Processing

Upon receiving a DL\_DATA\_REQUEST command, the MLAPD adds the logical link's Transmit Queue to the appropriate I frame queue for transmission, using the queue's tail pointer register. The stop\_Tx, active, acknowledge, and transmit bits in this link's transmit\_status entry are set to 0101, respectively, by the MLAPD. The last\_LLT bit in the transmit\_status entry is set to one, since this LLT is now the last one in the I frame queue. The MLAPD will not remove this link's Transmit Queue from its I frame queue until all frames are transmitted, except as the result of a STOP\_TX-I command.

Once the MLAPD begins handling this link's Transmit Queue, the MLAPD will continue servicing the queue until all frames are transmitted or until the scan\_length associated with the I frame queue is exhausted. If the scan\_length is exhausted before the Transmit Queue is emptied, the MLAPD will move on to service the next queue in the transmit servicing scheme. The next time this I frame queue is serviced again, the transmission of frames for this nonprotocol link will continue. When all frames in the Transmit Queue are transmitted, the MLAPD issues a DL\_data\_confirmation interrupt for the link.

#### 9.1.5 MLAPD Frame Transmission

When a frame is to be transmitted for a nonprotocol link, the MLAPD first fetches the link's context from its LLT by reading LLT entries 0 to 14. Then, based upon the transmit\_address\_disable bit in the frame descriptor, the MLAPD may or may not prefix

the DLCI associated with this logical link to the beginning of the transmit frame. Next, the MLAPD uses the address contained in the frame descriptor to locate the transmit memory buffer, and then transfers this information into its transmit first-in first-out (TxFIFO) buffer. The MLAPD transmits the frame inserting zeros when necessary if the zero-insertion-select is enabled. After the transmit buffer is transmitted, the MLAPD may or may not append a cyclical redundancy check (CRC) to the frame, based upon the transmit-CRC-disable bit in the frame descriptor.

Transmission begins when ten bytes are present in the TxFIFO or when the entire frame is present in the TxFIFO. Between frames, the MLAPD transmits the user selected number of pad flags. Additional flags may be transmitted, until the requirements for start of transmission are met. While transmitting a frame, the MLAPD requests use of the system bus when six to eight empty bytes are available in the TxFIFO.

### 9.1.6 Stopping Frame Transmission

At any time, the host may instruct the MLAPD to immediately suspend all transmit activity on a nonprotocol link by issuing a STOP\_TX\_I command. (See **4.5.6 STOP\_TX\_I Command**). Upon receiving this command, the MLAPD aborts any current frame transmission for this link and does not service additional frame descriptors. The Tx\_next\_pointer is not advanced past its current position. At this point, the host is free to manipulate the Transmit Queue as desired, since this queue is inactive. The host reactivates the handling of frame transmission for this link via the usual DL\_DATA\_REQUEST command. This stop mechanism may be used to provide faster servicing of certain frames for this link, or this mechanism may be necessary to perform various types of error recovery.

9

To stop transmit activity for a link without the possibility of an abort transmission, the host can set the last bit in the first two frame descriptors in the link's Transmit Queue, which have not been transmitted. This action ensures that the MLAPD will transmit only these two frames at the most, before stopping this link's transmission.

### 9.1.7 Collecting Transmitted Frames

To collect transmitted frames, the host reads the status\_bits in each frame descriptor, beginning with the frame descriptor indicated by the user\_Tx\_next\_confirm\_pointer. When the transmit bit is set, the associated data buffer has been transmitted. If the last bit is set, this frame descriptor is the last frame descriptor in the queue. Transmitted frames may be collected by the host dynamically (while other frames await transmission) or after being notified by the MLAPD that all frames in the queue are transmitted via a DL\_data\_confirmation interrupt. The host should update the user\_Tx\_next\_confirm\_pointer after each frame is collected.

When the host has dynamically added frames to a link's Transmit Queue, the host must determine whether the additional frames were handled by the LAPD. Verification is made

by inspecting the frame descriptor, which was the last frame descriptor in the queue before the additional frames were added. When the empty bit is set and the last bit is cleared in this frame descriptor, then the frame addition was not successful. The host must issue a DL\_DATA\_REQUEST with this link's LLID in command\_argument\_1 to enable transmission. The address of the frame descriptor, which was not successfully added, should be placed in command\_argument\_2.

### 9.1.8 Receive Structures

The receive structures for a nonprotocol link are identical to the structures for a protocol link. See Figure 6-1. A receive pool is a linked list of receive frame descriptors. The format of a receive frame descriptor is defined in **2.7 RECEIVE FRAME DESCRIPTOR**. When a receive buffer is filled with information from an incoming frame, the frame descriptor is written by the MLAPD to provide information about the received frame. At this point, the frame descriptor is logically removed from the receive pool although no unlinking/linking operation is performed. So, the receive structure serves as both a receive pool and a receive queue.

A nonprotocol link is assigned to a receive pool by the host using the receive\_pool\_number entry in the link's LLT. Any number of links may share a receive pool although it is anticipated that most system implementations will not assign protocol and nonprotocol links to the same receive pool since both links have different receive characteristics. The minimum length of the data buffers in a receive pool must be equal to the largest N201\_value specified for all links that share the pool. The N201\_value specified for a nonprotocol link defines the receive buffer length and must be an even number. Each buffer in a receive pool must accommodate all information between the opening and closing flag of a frame, except when multibuffer mode is enabled. See **9.2.1 Multibuffer Mode**.

### 9.1.9 MLAPD Frame Reception

Frame reception is enabled for a nonprotocol link by performing the link setup procedure described in **9.1.1 Setup Procedure**. As an incoming frame is received in on-chip system operation mode, the DLCI field in the frame is compared to the on-chip content addressable memory (CAM) to identify the corresponding logical link, if any. In expanded-system operation mode, the MLAPD accesses the external Match Table to determine the associated logical link. If no DLCI match is found, then the incoming DLCI has not been assigned to a logical link, and the frame is ignored.

If a DLCI match is found, the corresponding entry in the Match Table or CAM specifies both the LLID and whether the link is assigned to protocol or nonprotocol operation. If the frame is for a nonprotocol link, the MLAPD proceeds to locate a free buffer to store the information between the opening and closing flag of the frame. The MLAPD locates the link's receive pool by using the link's receive\_pool\_number LLT entry to index into the Receive Pool Pointers Table. If the receive pool associated with the logical link is empty, then the frame is ignored, and a local\_busy interrupt is generated.

Otherwise, the MLAPD begins to transfer the frame into the memory buffer. Throughout frame reception, the MLAPD performs zero deletion if enabled by the zero\_deletion\_select bit in the Global Configuration Block (GCB). The MLAPD also checks for receive errors and calculates the CRC. For every receive frame, five error conditions are checked using the following order of priority:

- 1) Inactive DLCL error detected;
- 2) Frame ended with abort or nonoctet aligned;
- 3) CRC error detected;
- 4) Data length error detected (information field exceeds N201).

Frame was too short (less than five octets between flags) when an error is detected during the receive process, the MLAPD does not advance the receive\_pool\_pointer. The next frame received for a link that shares this receive pool will be stored over the erroneous frame. However, the host may save these erroneous frames for inspection by setting the error\_mask\_valid bit in the link's LLT. The nonprotocol\_error\_mask entry allows the host to individually enable reception of frames with various error conditions.

For nonprotocol links, the host may also choose to have the MLAPD access multiple receive buffers as needed to store receive frames. If the multibuffer\_select bit in the option\_bits\_2 GCB entry is set to one and the frame is longer than N201, the chip will continue data reception into the next receive buffer in the pool. Any number of receive buffers may be used by the MLAPD in this mode for storing one incoming frame. Refer to **2.7.5 Frame Type and LLID** for more details.

### 9.1.10 Restrictions

The nonprotocol restrictions are:

- 1) N201\_value in LLT must be an even number greater than five.
- 2) The only valid commands for a nonprotocol link are:
  - ACTIVATE\_LL Establish a new link
  - DEACTIVATE\_LL Terminate the link activity
  - DL\_DATA\_REQUEST Start/continue transmission
  - STOP\_TX\_I Stop transmission
  - SET\_LOCAL\_BUSY Stop reception
  - CLEAR\_LOCAL\_BUSY Continue reception

Although these listed commands are considered valid for nonprotocol links, the MLAPD will not issue an undefined\_host\_command interrupt when commands not listed are issued by the host.

## 9.2 PROMISCUOUS RECEIVE MODE

Promiscuous receive mode is a special case of the nonprotocol link operation in which no address matching is performed by the MLAPD. The host instructs the chip to enter this

mode by setting the `promiscuous_receive_select` bit in the `option_bits_1` GCB entry and by assigning the logical link associated with DLCl equal to zero to nonprotocol operation. This mode only affects the MLAPD receiver operation. The MLAPD transmitter operation is not affected.

When promiscuous receive mode is selected, the MLAPD does not perform LAPD frame processing on the incoming serial bit stream. The MLAPD strips flags and performs zero deletion if enabled by the `zero_deletion_select` bit in the GCB. Otherwise, the  $\overline{\text{RSTART}}$  signal provides frame delineation. For every receive frame, three error conditions are checked using the following order of priority:

- 1) Frame ended with abort or nonoctet aligned;
- 2) CRC error detected;
- 3) Data length error detected (an information field exceeds N201) when not multibuffer.

No address matching or control field analysis is performed. The entire frame is stored in memory.

All incoming frames are transferred to one receive pool — the pool assigned by the host to the logical link with DLCl equal to zero. As each receive buffer is filled, the MLAPD writes a time stamp into the associated frame descriptor. This time stamp indicates the “time”, with respect to an internal 32-bit timer value, when the MLAPD has transferred the incoming frame to the specific receive buffer. This internal timer is incremented every 16 system clock cycles, which provides an accuracy of approximately one hour (for 16.67 MHz: one hour and eight minutes). This timer is zeroed during the INIT command execution.

The host may choose to accept invalid frames based upon the `nonprotocol_error_mask`. When a frame is received with one of these receive errors, the `error_code` entry in the frame descriptor defines the error detected by the MLAPD.

### 9.2.1 Multibuffer Mode

When the `multibuffer_select` bit in the `option_bits_1` GCB entry is zero, one receive buffer is used per frame. If the received frame is longer than the preassigned buffer, then, based upon the `buffer_length_exceeded` bit in the `nonprotocol_error_mask`, the frame is either 1) discarded and the same buffer is overwritten by the next incoming frame or 2) N201 bytes are stored and the remainder of the frame is discarded. On the other hand, when the `multibuffer_select` bit is set to one, the MLAPD will use as many receive buffers as required to store the received frame. For each frame longer than N201, the chip will use the subsequent receive buffer(s) in the receive pool until the whole frame is stored. The MLAPD writes the time stamp into each receive buffer as it is filled.

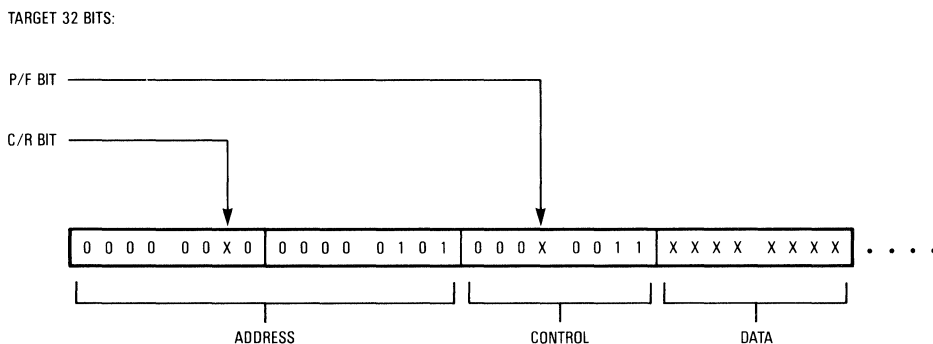
### 9.2.2 Filter Mode

This mode is an extension of the promiscuous receive mode in which only a portion of the received frames are stored in memory based upon a filtering mechanism. When the

MLAPD is in promiscuous receive mode, the host may enable a filtering option by setting the filter\_select bit in the options\_bits\_1 entry to one. This filter option allows the host to selectively receive frames from the serial link based upon the first 32-bits of each frame.

In this mode, the host defines two 32-bit words in the GCB: filter\_mask and filter\_match. During frame reception, the MLAPD performs a logical AND operation between the first 32 bits of each incoming frame and the GCB filter\_mask entry. The result of this operation is then compared to the result of (GCB filter\_match entry AND GCB filter\_mask entry). If equal, then the MLAPD places the information between the opening and closing flags of this frame into a receive memory buffer (including the address, control, information, and CRC fields).

For example, to receive all the unnumbered information (UI) frames from DLCI '2' (hex), the user programs the GCB entries as follows:



X = DON'T CARE Condition

Mask Word High = FFFD (hex) to mask C/R bit  
 Mask Word Low = 00EF (hex) to mask P/F bit and data byte  
 Match Word High = 0500 (hex)  
 Match Word Low = 0003 (hex)

MLAPD filter operation:  
 Match AND Mask = ? = (FIRST32) AND Mask  
 05000003 AND FFFD00EF = ? = (FIRST32) AND FFFD00EF

### 9.2.3 Restrictions

The two following restrictions apply to the promiscuous receive mode: 1) the logical link with DLCI equal to zero must be assigned as a *nonprotocol* link, and, 2) N201\_value in its LLT must be an even number greater than five.

## 9.3 PARALLEL ASSIST MODE

The purpose of this mode is to minimize the external logic required to interface the MLAPD to a physical level, which is based upon a parallel interface. This mode also allows the

MLAPD to be a controller for non-HDLC based serial communications. The full functionality of the MLAPD is available to support both protocol links and nonprotocol links when parallel assist mode is selected.

The MLAPD enters the parallel assist mode when the host sets the zero\_insertion\_select and zero\_deletion\_select bits in the option\_bits\_1 GCB entry to one. Since HDLC flags cannot serve as frame delimiters when zero insertion/deletion is disabled, two of the MLAPD serial pins are used to delineate the frames.

On the transmit side, the request-to-send ( $\overline{\text{RTS}}$ ) pin functions as transmit start ( $\overline{\text{TSTART}}$ ) to indicate that the data on the transmit data (TxD) output pin is valid. The  $\overline{\text{TSTART}}$  pin becomes active as the first bit of the serial frame is transmitted and stays active until the last bit of the frame is transmitted. On the receive side, the clear-to-send ( $\overline{\text{CTS}}$ ) pin functions as receive start ( $\overline{\text{RSTART}}$ ). The MLAPD considers the data on receive data (RxD) to be valid when  $\overline{\text{RSTART}}$  is asserted. The MLAPD continues receiving the frame until  $\overline{\text{RSTART}}$  is negated. The receiver checks the CRC for each frame although the host may choose to ignore the results of the CRC check by using the protocol and/or nonprotocol\_error\_masks.

All frames must be octet aligned. A nonoctet aligned frame is treated as though the frame ended with an abort. When the transmit machine needs to terminate the current frame with an abort due to an underrun or STOP command, then the  $\overline{\text{TSTART}}$  signal negation does not occur on an octet boundary.

Since this mode affects only the serial portion of the chip, LAPD protocol processing can continue normally. A possible application of this mode would be to provide LAPD protocol processing on data coming from a non HDLC-like physical media. A physical level device could provide the frames directly or via memory buffers to and from the MLAPD.

The parallel assist mode can also be used to assist the implementation of any protocol in software (not just HDLC-type protocols) since HDLC framing is not performed by the MLAPD. By assigning a link to nonprotocol operation, the frame is entirely user-defined. On the receive side, the MLAPD may be assigned to promiscuous receive mode. Thus, the MLAPD transmits and receives frames without regard to the imposed protocol.

## 9.4 LINE MONITOR

This mode is a combination of some of the previously defined operating modes. In line monitor, the user can monitor all the bits on the data link. To enter this mode, the host sets zero\_deletion\_select to one, multibuffer\_select to one, and promiscuous\_receive\_select to one. The host then assigns the logical link with DLCI equal to zero to nonprotocol operation.

When the  $\overline{\text{RSTART}}$  pin is asserted, the receive operation begins. As long as  $\overline{\text{RSTART}}$  remains asserted, the chip will pack all received bits to words and write them into receive data buffers from the receive pool specified for the logical link with DLCI equal to zero. Since no processing is performed on the serial bit stream, each bit appearing on RxD is dumped into successive memory buffers using the multibuffer receive function.

This mode may be very useful in analyzing line problems, bit errors, zero insertion errors, and special bit patterns such as flags, abort, and idle sequences that are not normally dumped into memory.

## 9.5 SYSTEM LOOPBACK TESTING

The host may wish to test the MC68606-based system without disturbing the network. However, a logical link cannot enter the connect state according to the LAPD protocol if it receives its own transmit frames (MLAPD transmitter connected to MLAPD receiver). To allow the MLAPD to perform the LAPD procedures during a system loopback test, the MLAPD provides a *flip* option. When the host enables the flip option by setting the flip\_select bit in the option\_bits\_2 GCB entry, the MLAPD will invert the least significant bit of the DLCI field for each received frame.

To implement this mode, the host activates pairs of logical links with DLCIs that differ only in the least significant bit position. For each pair, one link is assigned as network and the other link is assigned as user. Then when a frame is transmitted for one logical link, the frame is received for the other logical link; thus, the MLAPD can implement the LAPD procedures.

The loopback can be internal or external. When internal loopback is selected, the host may also specify that any bits received on RxD should be echoed back to the network on TxD. To enable this option, the host sets the echo\_select bit in the option\_bits\_1 GCB entry to one.

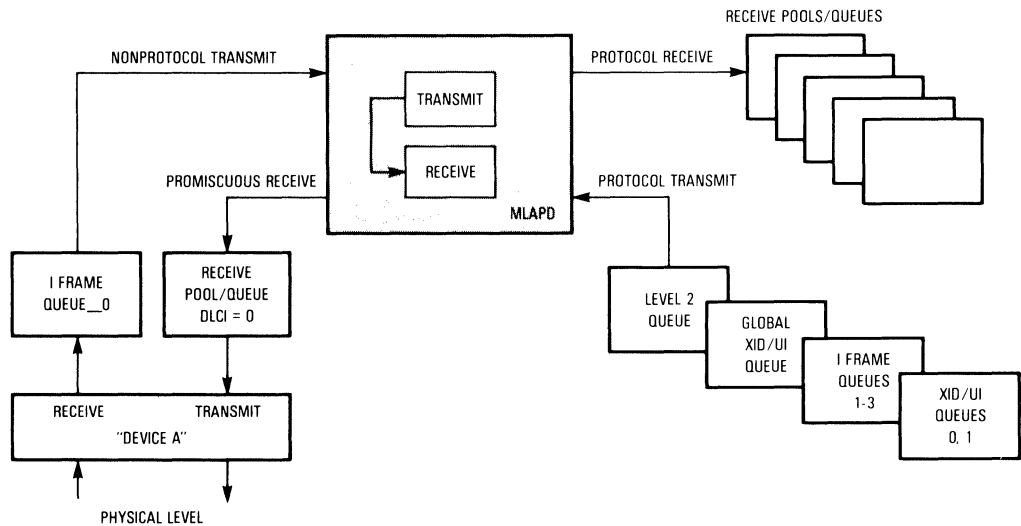
When the flip\_select option is not enabled, the host may only operate nonprotocol links under the internal or external loopback configuration.

## 9.6 MEMORY-TO-MEMORY OPERATION

The MLAPD can be configured for operation in systems with a channelized serial interface such as T1 networks or local area networks (LANs). The memory-to-memory operation mode allows the system designer to use the MLAPD to implement the LAPD procedures in a manner independent of the physical level characteristics of the system. When using the MLAPD in this manner, it is assumed that a device in the system (*Device A* in Figure 9-1) directly interfaces to the physical level. Received frames are placed into memory, and transmit frames are placed into Device A using either direct memory access (DMA) or host input/output (I/O) capabilities. In memory-to-memory operation, the MLAPD performs the LAPD procedures on the memory-resident receive and transmit frames. Frames for non-protocol links are also supported. Since the memory-to-memory operation mode uses the receive actions described for promiscuous receive operation and the transmit actions described for nonprotocol links, the designer should refer to **9.2 PROMISCUOUS RECEIVE MODE** and **9.1 NONPROTOCOL LINKS**.

To enable memory-to-memory operation, the host must set the memory-to-memory\_select, zero\_insertion\_disable, and zero\_deletion\_disable bits in the option\_bits\_1 entry of the GCB. In this configuration, the MLAPD transmitter and receiver are internally connected; the RxD and TxD pins are not used. The MLAPD on-chip DMA controller feeds the receive





**Figure 9-1. Memory-to-Memory Operation**

and transmit machines via their associated FIFOs. The host then activates a logical link with DLCI equal to zero as a nonprotocol link and assigns this link to I frame Queue\_0. I frame Queue\_0 should be reserved for use only by this logical link associated with DLCI equal to zero.

### 9.6.1 Handling Received Frames

As frames are received from the physical level, the host is responsible for placing these frames into the memory format defined for a MLAPD Transmit Queue. Each transmit buffer contains an entire received frame. The host then issues a DL\_DATA\_REQUEST command with both the LLID associated with DLCI equal to zero as command\_argument\_1 and the address of the first transmit frame descriptor as command\_argument\_2. In response, the MLAPD begins transmitting frames from I frame Queue\_0 according to the MLAPD transmit servicing scheme.

Since the logical link associated with DLCI equal to zero is assigned to nonprotocol operation, the MLAPD acts as a simple HDLC framer while servicing I frame Queue\_0. The MLAPD may optionally add the frame CRC, based upon the transmit\_CRC\_disable bit in each transmit frame descriptor. The frame address should already be contained in the transmit data buffer (passed through from the physical level).

When the frame is received via the internal loopback path, the MLAPD analyzes the address field, determines whether the addressed logical link is assigned to LAPD or nonprotocol operation, and then handles the frame accordingly. The MLAPD continues to transmit frames from I frame Queue\_0 until the scan\_length specified for this queue is exhausted or until all queued frames are transmitted. A DL\_data\_confirmation interrupt is issued to the host when all frames are transmitted. This interrupt indicates that all frames received

from the physical level have been handled by the MLAPD. Level 3 collects the received I, exchange identification (XID), unnumbered information (UI), and nonstandard-control frames for the various logical links from their associated receive queues. All frames received for nonprotocol links are collected from their associated receive queues and processed by the host.

### 9.6.2 Handling Transmit Frames

The memory-to-memory operation mode does not affect the procedure for passing transmit frames from Level 3 to the MLAPD for Level 2 handling. Level 3 queues I, XID, UI, and nonstandard-control frames to the Global XID/UI Queue, I frame Queues-1-3, and XID/UI Queues-0,1. (I frame Queue-0 is not available for passing transmit frames since it is dedicated for passing received frames to the MLAPD for processing.) The MLAPD independently generates unnumbered (U) and supervisory (S) frames and places them on the Level 2 Queue.

The transmit servicing scheme for these transmit queues and the protocol actions provided by the MLAPD are unchanged. During the transmit process, the MLAPD prefixes the appropriate DLCI, control field, and CRC to queued transmit frames for LAPD links. The MLAPD may optionally append address and CRC for nonprotocol links.

As each transmitted frame is received via the internal loopback path, the MLAPD receiver handles the frame as for promiscuous receive mode. The entire frame is stored in memory. The MLAPD uses the receive pool assigned by the host to the logical link associated with DLCI equal to zero. The host is responsible for enabling transmission of these frames by the physical level.

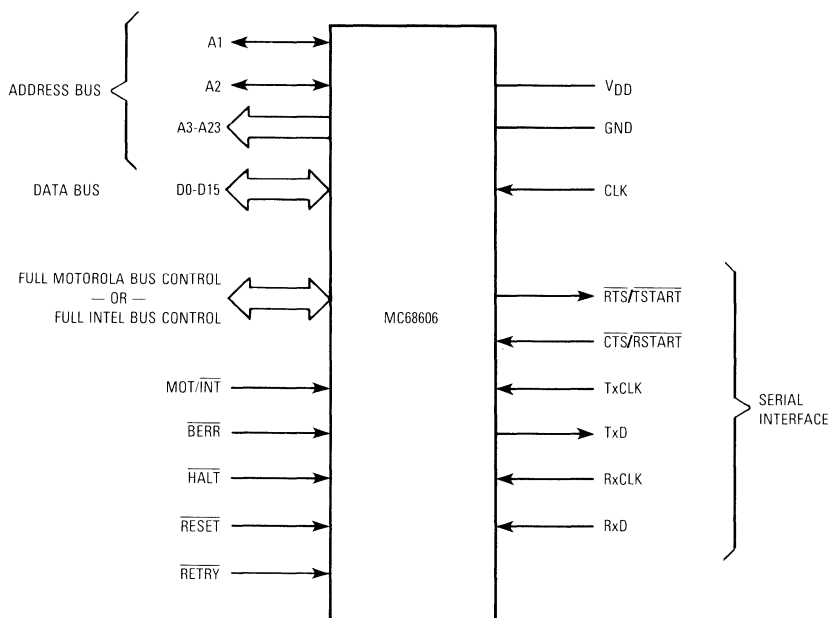
## SECTION 10 SIGNAL DESCRIPTION

The input and output signals of the MLAPD can be functionally viewed as shown in Figure 10-1. The MC68606 implements a 24-bit address bus and supports either an 8- or 16-bit data bus. The level on the MOT/ $\overline{\text{INT}}$  pin determines whether the bus interface signals operate according to Motorola or Intel-compatible bus specifications. In either case, the four bus exception signals,  $\overline{\text{BERR}}$ ,  $\overline{\text{HALT}}$ ,  $\overline{\text{RESET}}$ , and  $\overline{\text{RETRY}}$  are available. The physical-level connection is implemented as a conventional 6-signal serial interface. The system clock is independent of the receive and transmit serial clocks. Additionally the internal serial logic block is static, allowing the receive and transmit clocks to be stopped if necessary.

This section contains a brief description of the input and output signals of the MLAPD. Reference is given (if applicable) to other paragraphs containing more information about the function being performed.

### NOTE

The terms **assertion** and **negation** will be used extensively to avoid confusion when dealing with a mixture of "active low" and "active high" signals. The terms assert



**Figure 10-1. MC68606 Signals**

and assertion are used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The terms negate and negation are used to indicate that a signal is inactive or false.

## 10.1 SERIAL INTERFACE

### 10.1.1 Modem Control Signals

The following paragraphs describe the modem control signals.

**10.1.1.1 REQUEST-TO-SEND ( $\overline{\text{RTS}}$ ) OR TRANSMIT START ( $\overline{\text{TSTART}}$ ).** This output pin is defined as  $\overline{\text{RTS}}$  when the zero-insertion-select bit in option-bits-1 Global Configuration Block (GCB) entry is set to zero. This pin is defined as  $\overline{\text{TSTART}}$  when the zero-insertion-select bit is set to one.

The MLAPD negates the  $\overline{\text{RTS}}$  pin for the following conditions:

- Hardware or software reset,
- DMA-TEST command, and
- Loopback-select enabled and echo-select disabled.

When  $\overline{\text{RTS}}$  is negated, transmit data (TxD) is three-stated.

The MLAPD asserts the  $\overline{\text{RTS}}$  output pin either continually or intermittently after executing the INIT command based upon the RTS-select bit in the option-bits-1 GCB entry. If RTS-select is set to one,  $\overline{\text{RTS}}$  will be asserted when the first transmit frame is pending following INIT and will remain asserted until one of the above conditions occur, which cause the MLAPD to negate  $\overline{\text{RTS}}$ . When no transmit frames are pending in this mode, the MLAPD will transmit flags on the TxD line.

If RTS-select is set to zero,  $\overline{\text{RTS}}$  will be asserted whenever a frame is waiting for transmission and negated whenever no transmit frames are pending. When no transmit frames are pending in this mode, the TxD line is three-stated. This provides a ones fill when a pullup resistor is connected to TxD, satisfying the basic rate ISDN requirement.

For either setting of the RTS-select bit, when  $\overline{\text{RTS}}$  and clear-to-send ( $\overline{\text{CTS}}$ ) are both asserted, the MLAPD only transmits pad flags and frames.

The MLAPD asserts the  $\overline{\text{TSTART}}$  when the first bit of a frame is presented on the TxD pin and negates this signal after the last bit of the frame is presented on TxD.

**10.1.1.2 CLEAR-TO-SEND ( $\overline{\text{CTS}}$ ) OR RECEIVE START ( $\overline{\text{RSTART}}$ ).** This input pin is defined as  $\overline{\text{CTS}}$  when the zero-deletion-select bit in the option-bits-1 GCB entry is set to zero. This pin is defined as  $\overline{\text{RSTART}}$  when the zero-deletion-select bit is set to one.

Following the assertion of  $\overline{\text{RTS}}$ , the MLAPD monitors  $\overline{\text{CTS}}$ . If  $\overline{\text{CTS}}$  is not asserted within the time period defined by the  $\text{CTS\_timeout\_threshold}$  value multiplied by 2048 transmit clock (TxCLK) cycles, a  $\text{CTS\_timeout\_threshold}$  interrupt is placed on the Interrupt Queue.

If the  $\overline{\text{CTS}}$  signal is negated for more than one TxCLK cycle during a frame transmission, the MLAPD writes a  $\text{CTS\_lost}$  indication to the Interrupt Queue. Whenever  $\overline{\text{CTS}}$  is negated, TxD is in three-state. By connecting a pullup resistor to TxD, ones are sent on the TxD line while  $\overline{\text{CTS}}$  is lost. If  $\overline{\text{CTS}}$  remains negated for seven TxCLK cycles during frame transmission, the effect is as if an abort has been generated for the frame. If the  $\overline{\text{CTS}}$  is reasserted in the same frame, the MLAPD transmitter is allowed to drive TxD with the bit that would have been transmitted at that bit time if  $\overline{\text{CTS}}$  had not been lost. However, a cyclical redundancy check (CRC) error should be detected at the receiving station due to the ones which were inserted in the bit stream.

When  $\overline{\text{RSTART}}$  is asserted, the receiver starts receiving the data on receive data (RxD). When  $\overline{\text{RSTART}}$  is negated the receiver stops sampling RxD and checks the CRC. A frame is defined as all the bits received on RxD while  $\overline{\text{RSTART}}$  is asserted.

## 10.1.2 Transmit Signals

The following paragraphs describe the transmit signals.

**10.1.2.1 TRANSMIT CLOCK (TxCLK).** The MLAPD synchronizes the transmit data to this input clock. This clock is also used to synchronize both the receive and transmit data during internal serial loopback. The MLAPD was designed to perform the full LAPD procedures with a serial clock to system clock ratio of 1:6. Operation at ratios less than 1:6 may result in performance and throughput degradation. The limiting serial clock to system clock ratio is 1:1.

**10.1.2.2 TRANSMIT DATA (TxD).** This output pin is used to send the serial bit stream. The data is encoded in nonreturn-to-zero (NRZ). TxD can be driven by two sources: the transmitter and RxD. The  $\text{echo\_select}$  bit in the  $\text{option\_bits\_1}$  GCB entry selects between these two transmit sources.

The control of the three-state logic for TxD is a function of four option bits, an input pin, and an output pin (see Table 10-1).

## 10.1.3 Receive Signals

The following paragraphs describe the receive signals.

**10.1.3.1 RECEIVE CLOCK (RxCLK).** The MLAPD synchronizes the receive data to this input clock. The MLAPD was designed to perform the full LAPD procedures with a serial clock

**Table 10-1. TxD Three-State Logic Control**

Echo	Loop Back	Zero Insert Select	Zero Delete Select	$\overline{\text{RTS}}$	$\overline{\text{CTS}}$	Output in Three-State?
1	X	X	X	X	X	No (RxD is the transmit source)
0	1	X	X	X	X	Yes
0	0	1	X	X	X	No (transmit queues are the transmit source)
0	0	X	1	X	X	No (transmit queues are the transmit source)
0	0	0	0	0	0	No (transmit queues are the transmit source)
0	0	0	0	0	1	Yes
0	0	0	0	1	0	Yes
0	0	0	0	1	1	Yes

to system clock ratio of 1:6. Operation at ratios less than 1:6 may result in performance and throughput degradation. The limiting serial clock to system clock ratio is 1:1.

**10.1.3.2 RECEIVE DATA (RxD).** This input line receives the serial bit stream from the communications link synchronized to the receive clock. The data is encoded in NRZ.

## 10.2 MOTOROLA BUS INTERFACE

On the system bus, the MLAPD operates as a bus master and as a slave device. When in the master mode, the MLAPD assumes mastership of the system bus and performs memory reads and writes using its on-chip direct memory access (DMA) capability. The MLAPD enters slave mode whenever  $\overline{\text{CS}}$  or  $\overline{\text{IACK}}$  ( $\overline{\text{INTA}}$ ) is asserted. In this mode, the MLAPD accepts data from or places data on D0-D15, according to the level on the R/W (RD, WR) pin. Therefore, many MLAPD system bus signals are bidirectional, with the pin's status as an input or an output determined by the current master or slave operation mode.

The Motorola bus interface signals are discussed in the following paragraphs.

### 10.2.1 Motorola/Intel Mode (MOT/ $\overline{\text{INT}}$ )

This input signal determines whether the MLAPD operates according to the Motorola asynchronous-system bus specifications or the Intel synchronous-system bus specifications. When MOT/ $\overline{\text{INT}}$  is high, the MLAPD bus signals function as Motorola bus signals; when MOT/ $\overline{\text{INT}}$  is low, the MLAPD bus signals function as Intel bus signals.

### 10.2.2 Address Bus (A1–A23)

This is a 24-bit (when combined with the  $\overline{\text{UDS/A0}}$  signal), unidirectional (with the exception of A1 and A2), three-state bus capable of addressing up to 16 Mbytes of memory. A1 and

A2 are bidirectional three-state lines that address internal MLAPD registers in the slave mode and that provide the lower two address outputs in the master mode.

### 10.2.3 Data Bus (D0–D15)

The MLAPD has a 16-bit, bidirectional, three-state bus for general-purpose data transfer. The MLAPD can be configured to interface to an 8-bit or a 16-bit data bus. The data bus is used for data input during a host processor write or MLAPD read cycle, and for data output during a host processor read or MLAPD write cycle.

### 10.2.4 Bus Control Signals

The following paragraphs describe the bus control signals.

**10.2.4.1 CHIP SELECT ( $\overline{CS}$ ).** This input pin selects the MLAPD for a host processor bus cycle. When  $\overline{CS}$  is asserted, the address on A1, A2, and the data strobes select the internal MLAPD register that will be involved in the transfer.  $\overline{CS}$  should be generated by qualifying an address decode signal with address strobe.

**10.2.4.2 ADDRESS STROBE ( $\overline{AS}$ ).** This bidirectional three-state signal is an output in the direct memory access (DMA) mode which indicates that a valid address is present on the address bus. In slave mode,  $\overline{AS}$  is an input that is monitored to determine when the MLAPD can take control of the bus (after the MLAPD has requested and has been granted use of the system bus).

**10.2.4.3 READ/WRITE ( $R/\overline{W}$ ).** This bidirectional three-state signal indicates the direction of data transfer during a bus cycle. The  $R/\overline{W}$  pin is an input in the slave mode. A high level indicates that the transfer is from the MLAPD to the data bus, and a low level indicates that the transfer is from the data bus to the MLAPD.

The  $R/\overline{W}$  pin is an output in the master mode. A high level indicates that the transfer is from the data bus to the MLAPD, and a low level indicates that the transfer is from the MLAPD to the data bus.

**10.2.4.4 UPPER DATA STROBE ( $\overline{UDS}/A0$ ) AND LOWER DATA STROBE ( $\overline{LDS}/\overline{DS}$ ).** These bidirectional three-state signals control the flow of data on the data bus. When using a 16-bit data bus, these pins function as  $\overline{UDS}$  and  $\overline{LDS}$ . During any bus cycle,  $\overline{UDS}$  is asserted if data is to be transferred over data lines D8–D15, and  $\overline{LDS}$  is asserted if data is to be transferred over data lines D0–D7.  $\overline{UDS}$  and  $\overline{LDS}$  are controlled by the MLAPD when operating in the master mode and by the host when operating in the slave mode.

When using an 8-bit data bus, these pins function as A0 and  $\overline{DS}$ . A0 is an extension to the lower address lines to provide the address of a byte in the address map and is valid when A1–A23 are valid.  $\overline{DS}$  is the data strobe that enables external data buffers and indicates that valid data is on the bus during a write cycle. See Table 10-2.

**Table 10-2. Motorola Data Strobe Control of Data Bus**

$\overline{UDS}/A0$	$\overline{LDS}/\overline{DS}$	$R/\overline{W}$	D8-D15	D0-D7
<b>16-Bit Bus</b>				
High	High	X	No Valid Data	No Valid Data
Low	Low	X	Valid Data	Valid Data
High	Low	Low	X	Valid Data
High	Low	High	X	Valid Data
Low	High	Low	Valid Data	X
Low	High	High	Valid Data	X
<b>8-Bit Bus</b>				
X	Low	Low	No Valid Data	Valid Data
X	Low	High	X	Valid Data
X	High	X	No Valid Data	No Valid Data

X-DON'T CARE Condition

**10.2.4.5 DATA TRANSFER ACKNOWLEDGE ( $\overline{DTACK}$ ).** This bidirectional three-state line signals that the asynchronous bus cycle may be terminated. In the slave mode, this output indicates that the MLAPD has accepted data from the host or placed data on the system bus for the host. In the master mode, this input is monitored by the MLAPD to determine when to terminate the bus cycle. As long as  $\overline{DTACK}$  remains negated, the MLAPD will insert wait cycles into the bus cycle. The system should provide the DMA timeout function by asserting bus error ( $\overline{BERR}$ ) or  $\overline{RETRY}$  to terminate the cycle. When  $\overline{DTACK}$  is asserted, the bus cycle will be terminated.

## 10.2.5 Bus Arbitration Signals

10

The three signals discussed in the following paragraphs form a bus arbitration circuit that determines which device in a system is the current bus master.

**10.2.5.1 BUS REQUEST ( $\overline{BR}$ ).** This open-drain output pin is asserted by the MLAPD to request control of the bus.  $\overline{BR}$  is wire-ORed with all other devices that may be bus masters.

**10.2.5.2 BUS GRANT ( $\overline{BG}$ ).** This input is asserted by the CPU or an external bus arbiter to inform the MLAPD that it may assume bus mastership as soon as the current bus cycle is completed.

**10.2.5.3 BUS GRANT ACKNOWLEDGE ( $\overline{BGACK}$ ).** This bidirectional three-state signal is asserted by the MLAPD to indicate that it is the current system bus master.  $\overline{BGACK}$  is monitored as an input to determine when the MLAPD can become bus master.  $\overline{BGACK}$  is not asserted as an output until the following conditions are met:

- $\overline{BR}$  is asserted,
- $\overline{BG}$  is asserted,



$\overline{AS}$  is inactive, indicating that the current bus cycle has ended,  $\overline{BGACK}$  is inactive, indicating that no other device is claiming bus mastership, and  $\overline{HALT}$ ,  $\overline{RETRY}$ , and  $\overline{BERR}$  are negated.

## 10.2.6 Interrupt Control Signals

The two signals discussed in the following paragraphs perform an interrupt request/acknowledge handshake with the host processor.

**10.2.6.1 INTERRUPT REQUEST ( $\overline{IRQ}$ ).** This open-drain output is asserted by the MLAPD to request service from the host.

**10.2.6.2 INTERRUPT ACKNOWLEDGE ( $\overline{IACK}$ ).** This input is asserted by the host to acknowledge that it has received an interrupt request from the MLAPD. In response to the assertion of  $\overline{IACK}$ , the MLAPD places a vector on D0-D7 that is used by the host to fetch the proper MLAPD interrupt handler routine.

## 10.2.7 Bus Exception Signals

The following paragraphs describe the bus exception signals.

**10.2.7.1 RESET ( $\overline{RESET}$ ).** When this input signal is asserted, the MLAPD executes an internal reset sequence.  $\overline{RESET}$  should be asserted for at least 10 system clock cycles. TxCLK is not required for completion of reset. The semaphore register (SR) is set to hex 'FF' when the reset sequence is completed.

**10.2.7.2 HALT ( $\overline{HALT}$ ).** When this input signal is asserted, the MLAPD halts after the current bus cycle is terminated by  $\overline{DTACK}$ . The MLAPD releases ownership of the bus and enters the idle state until all the exception signals are negated. At this time, the MLAPD will rearbtrate for the system bus and continue DMA operations, if necessary.

**10.2.7.3 BUS ERROR ( $\overline{BERR}$ ).** When the  $\overline{BERR}$  input signal is asserted, the MLAPD aborts the current bus cycle and releases control of the bus, regardless of the state of the  $\overline{HALT}$  or  $\overline{RETRY}$  signals. After  $\overline{BERR}$  is negated, the MLAPD will rearbtrate for the bus as part of its bus error handling operation. The MLAPD reports the bus error via the Interrupt Queue.

**10.2.7.4 RETRY ( $\overline{RETRY}$ ).** When the  $\overline{RETRY}$  input signal is asserted, the MLAPD terminates the current bus cycle and enters a waiting mode. After all the exception signals are negated by external logic, the MLAPD will rerun the previous cycle. If  $\overline{HALT}$ ,  $\overline{RETRY}$ , and  $\overline{DTACK}$  signal are asserted simultaneously, the MLAPD enters the relinquish and retry sequence.

## 10.2.8 System Clock (CLK) Signal

This input signal is the MLAPD master clock. This signal ranges from 8 to 16.67 MHz. The MLAPD can operate with a CLK signal that is synchronous or asynchronous with respect to the host clock in the Motorola mode, as long as the bus requirements are satisfied.

## 10.2.9 Motorola Signal Summary

Table 10-3 is a summary of the Motorola signals definitions.

**Table 10-3. Motorola Signal Summary**

Signal Name	Mnemonic	Input/Output	Active State	Driver Type
Address Bus	A1-A2	Input/Output	NA	Three-State
Address Bus	A3-A23	Output	NA	Three-State
Data Bus	D0-D15	Input/Output	NA	Three-State
Upper Data Strobe	$\overline{UDS}/A0$	Input/Output	Low/NA	Three-State*
Lower Data Strobe	$\overline{LDS}/\overline{DS}$	Input/Output	Low	Three-State*
Address Strobe	$\overline{AS}$	Input/Output	Low	Three-State*
Read/Write	$R/\overline{W}$	Input/Output	High/Low	Three-State*
Chip Select	$\overline{CS}$	Input	Low	
Data Transfer Acknowledge	$\overline{DTACK}$	Input/Output	Low	Three-State*
Bus Request	$\overline{BR}$	Output	Low	Open-Drain
Bus Grant	$\overline{BG}$	Input	Low	
Bus Grant Acknowledge	$\overline{BGACK}$	Input/Output	Low	Three-State*
Reset	$\overline{RESET}$	Input	Low	
Halt	$\overline{HALT}$	Input	Low	
Bus Error	$\overline{BERR}$	Input	Low	
Retry	$\overline{RETRY}$	Input	Low	
Motorola/Intel	$\overline{MOT}/\overline{INT}$	Input	High/Low	
Request-To-Send/Transmit Start	$\overline{RTS}/\overline{TSTART}$	Output	Low	Normal
Clear-To-Send/Receive Start	$\overline{CTS}/\overline{RSTART}$	Input	Low	
Transmit Clock	TxCLK	Input	NA	
Transmit Data	TxD	Output	N/A	Three-State*
Receive Clock	RxCLK	Input	NA	
Receive Data	RxD	Input	N/A	
Interrupt Request	$\overline{IRQ}$	Output	Low	Open-Drain
Interrupt Acknowledge	$\overline{IACK}$	Input	Low	
Clock	CLK	Input	NA	

\*These signals require a pullup resistor to maintain a high voltage when in the high-impedance or negated state. However, when these signals go to the high-impedance or negated state, they will first drive the pin high momentarily to reduce the signal rise time.

## 10.3 INTEL-COMPATIBLE BUS INTERFACE

On the system bus, the MLAPD operates as a bus master and as a slave device. When in the master mode, the MLAPD assumes mastership of the system bus and performs memory reads and writes using its on-chip direct memory access (DMA) capability. The MLAPD enters slave mode whenever  $\overline{CS}$  or  $\overline{IACK}$  ( $\overline{INTA}$ ) is asserted. In this mode, the MLAPD accepts data from or places data on D0-D15, according to the level on the R/W ( $\overline{RD}$ ,  $\overline{WR}$ ) pin. Therefore, many MLAPD system bus signals are bidirectional, with the pin's status as an input or an output determined by the current master or slave operation mode.

The Intel-compatible bus interface signals are discussed in the following paragraphs.

### 10.3.1 Motorola/Intel Mode ( $\overline{MOT/INT}$ )

This input signal determines whether the MLAPD operates according to the Motorola asynchronous system bus specifications or the Intel synchronous system bus specifications. When  $\overline{MOT/INT}$  is high, the MLAPD bus signals function as Motorola bus signals; when  $\overline{MOT/INT}$  is low, the MLAPD bus signals function as Intel-compatible bus signals.

### 10.3.2 Address Bus (A0–A23)

This is a 24-bit, unidirectional (with the exception of A0, A1, and A2), three-state bus capable of addressing up to 16 Mbytes of memory. A1 and A2 are bidirectional three-state lines that address internal MLAPD registers in the slave mode and that provide the lower address output lines in the master mode. A0 is a bidirectional, three-state line that enables data transfer onto the lower byte of the data bus (D7-D0).

### 10.3.3 Data Bus (D0–D15)

The MLAPD has a 16-bit, bidirectional, three-state bus for general-purpose data transfers. The MLAPD can be configured to interface to an 8-bit or a 16-bit data bus. The data bus is used for data input during a host processor write or MLAPD read cycle and for data output during a host processor read or MLAPD write cycle.

### 10.3.4 Bus Control Signals

The following paragraphs describe the bus control signals.

**10.3.4.1 CHIP SELECT ( $\overline{CS}$ ).** This input pin selects the MLAPD for a host processor bus cycle. When  $\overline{CS}$  is asserted, the address on A0, A1, A2 and the  $\overline{BHE}$  selects the internal MLAPD register that will be involved in the transfer.

**10.3.4.2 ADDRESS STROBE ( $\overline{AS}$ ).** In master mode, this three-state output signal indicates that a valid address is present on the address bus.

**10.3.4.3 READ ( $\overline{RD}$ ).** This bidirectional three-state signal indicates the direction of the data transfer during a bus cycle. In master mode,  $\overline{RD}$  is driven low by the MLAPD when a read cycle is to be performed. In slave mode,  $\overline{RD}$  is an input signal monitored by the MLAPD. When the  $\overline{RD}$  input is low, the MLAPD transfers data from the selected register onto the data bus.

**10.3.4.4 WRITE ( $\overline{WR}$ ).** This bidirectional three-state signal indicates the direction of data transfer during a bus cycle. In master mode,  $\overline{WR}$  is driven low by the MLAPD when a write cycle is to be performed. In slave mode,  $\overline{WR}$  is an input signal monitored by the MLAPD. When the  $\overline{WR}$  input is low, the MLAPD transfers data from the data bus into the selected register.

**10.3.4.5 BUS HIGH ENABLE ( $\overline{BHE}$ ).** This bidirectional three-state line is used to enable transfer of data onto the most significant byte of the data bus (D15–D8). In master mode,  $\overline{BHE}$  is an output signal that is combined with A0 to control the flow of data as shown in the following table. In the slave mode,  $\overline{BHE}$  is an input signal that is combined with A0 to control the data flow on the data bus. The  $\overline{BHE}$  and A0 decoding is shown in Table 10-4.

**10.3.4.6 READY (READY).** This three-state input/output line is used to extend data transfer cycles as necessary. In master mode, READY is an input signal monitored by the MLAPD to determine when to terminate the current bus cycle. While READY is negated, the MLAPD will insert wait states. When READY is asserted, the MLAPD will terminate the bus cycle.

**Table 10-4. Intel-Compatible Byte Addressing**

$\overline{BHE}$	A0	$\overline{RD}$	$\overline{WR}$	D8-D15	D0-D7
<b>16-Bit Bus</b>					
X	X	High	High	No Valid Data	No Valid Data
High	High	X	X	No Valid Data	No Valid Data
Low	Low	Low	High	Valid Data	Valid Data
Low	Low	High	Low	Valid Data	Valid Data
High	Low	Low	High	X	Valid Data
High	Low	High	Low	X	Valid Data
Low	High	Low	High	Valid Data	X
Low	High	High	Low	Valid Data	X
<b>8-Bit Bus</b>					
X	X	High	High	No Valid Data	No Valid Data
X	High	Low	High	X	Valid Data
X	High	High	Low	No Valid Data	Valid Data
X	Low	Low	High	X	Valid Data
X	Low	High	Low	No Valid Data	Valid Data

X—DON'T CARE Condition

In slave mode, READY is an output signal asserted by the MLAPD indicating to the host that the MLAPD is ready to perform the data transfer on the next clock high transition.

### 10.3.5 Bus Arbitration Signals

The two signals discussed in the following paragraphs are used for requesting and acknowledging bus mastership.

**10.3.5.1 HOLD REQUEST (HRQ).** This output signal is asserted to request control of the system bus. This signal remains asserted after bus mastership has been granted, as long as the MLAPD controls the system bus.

**10.3.5.2 HOLD ACKNOWLEDGE (HOLDA).** This input signal is asserted by the CPU or an external bus arbiter indicating that the MLAPD has been given mastership of the system bus. This signal is negated when HRQ is negated, acknowledging that the current bus master has released control of the system bus. If HOLDA is negated before HRQ negation, the MLAPD will execute a HALT sequence. The MLAPD releases ownership of the bus after the current bus cycle is terminated by READY and then negates HRQ.

### 10.3.6 Interrupt Control Signals

The two signals discussed in the following paragraphs perform an interrupt request/acknowledge handshake with the host processor.

**10.3.6.1 INTERRUPT REQUEST (INTR).** This output signal is asserted by the MLAPD to request service from the host.

**10.3.6.2 INTERRUPT ACKNOWLEDGE ( $\overline{\text{INTA}}$ ).** This input signal is asserted by the host to acknowledge that it has received an interrupt request from the MLAPD. In response to the assertion of  $\overline{\text{INTA}}$ , the MLAPD places a vector on D0–D7, which is used by the host to fetch the address of the proper MLAPD interrupt handler routine, and negates INTR.  $\overline{\text{INTA}}$  may be asserted once or twice, depending on the interrupt logic implementation. The interrupt vector will be driven at every  $\overline{\text{INTA}}$  assertion.

### 10.3.7 Bus Exception Signals

The following paragraphs describe the bus exception signals.

**10.3.7.1 RESET ( $\overline{\text{RESET}}$ ).** When this input signal is asserted, the MLAPD executes an internal reset sequence. RESET should be asserted for at least 10 clock cycles. TxCLK is not required for completion of reset. The SR is set to hex 'FF' when the reset sequence is completed.

**10.3.7.2 HALT ( $\overline{\text{HALT}}$ ).** When this input signal is asserted, the MLAPD halts after the current bus cycle is terminated by  $\overline{\text{READY}}$ . The MLAPD releases ownership of the bus and enters the idle state until all the exception signals are negated. At this time, the MLAPD will rearbtrate for the system bus and continue DMA operations, if necessary.

**10.3.7.3 BUS ERROR ( $\overline{\text{BERR}}$ ).** When the  $\overline{\text{BERR}}$  input signal is asserted, the MLAPD aborts the current bus cycle and releases control of the bus, regardless of the state of the  $\overline{\text{HALT}}$  or  $\overline{\text{RETRY}}$  signals. After  $\overline{\text{BERR}}$  is negated, the MLAPD will rearbtrate for the bus as part of its bus error handling operation. The MLAPD reports the bus error via the Interrupt Queue.

**10.3.7.4 RETRY ( $\overline{\text{RETRY}}$ ).** When the  $\overline{\text{RETRY}}$  input signal is asserted, the MLAPD terminates the current bus cycle and enters a waiting mode. After all the exception signals are negated by external logic, the MLAPD will rerun the previous cycle. If  $\overline{\text{HALT}}$ ,  $\overline{\text{RETRY}}$ , and  $\overline{\text{READY}}$  are asserted, the MLAPD enters the relinquish and retry sequence.

### 10.3.8 System Clock (CLK) Signal

This input signal is the MLAPD master clock. This signal can range from 8 to 16.67 MHz.

### 10.3.9 Intel Signal Summary

Table 10-5 is a summary of the Intel signal definitions.

**Table 10-5. Intel-Compatible Signal Summary**

Signal Name	Mnemonic	Input/Output	Active State	Driver Type
Address Bus	A0	Input/Output	NA	Three-State*
Address Bus	A1-A2	Input/Output	NA	Three-State*
Address Bus	A3-A23	Output	NA	Three-State
Data Bus	D0-D15	Input/Output	NA	Three-State
Address Strobe	$\overline{AS}$	Output	Low	Three-State*
Bus High Enable	$\overline{BHE}$	Input/Output	Low	Three-State*
Read	$\overline{RD}$	Input/Output	Low	Three-State*
Write	$\overline{WR}$	Input/Output	Low	Three-State*
Chip Select	$\overline{CS}$	Input	Low	
Ready	READY	Input/Output	High	Three-State**
Hold Request	HRQ	Output	High	Normal
Hold Acknowledge	HOLDA	Input	High	
Reset	$\overline{RESET}$	Input	Low	
Halt	$\overline{HALT}$	Input	Low	
Bus Error	$\overline{BERR}$	Input	Low	
Retry	$\overline{RETRY}$	Input	Low	
Motorola/Intel	$\overline{MOT/INT}$	Input	High/Low	
Request-To-Send/Transmit Start	$\overline{RTS/TSTART}$	Output	Low	Normal
Clear-To-Send/Receive Start	$\overline{CTS/RSTART}$	Input	Low	
Transmit Clock	TxCLK	Input	NA	
Transmit Data	TxD	Output	NA	Three-State*
Receive Clock	RxCLK	Input	NA	
Receive Data	RxD	Input	NA	
Interrupt Request	INTR	Output	High	Normal
Interrupt Acknowledge	$\overline{INTA}$	Input	Low	
Clock	CLK	Input	NA	

\*These signals require a pullup resistor to maintain a high voltage when in the high-impedance or negated state. However, when these signals go to the high-impedance or negated state, they will first drive the pin high momentarily to reduce the signal rise time.

\*\*This signal requires a pulldown resistor to maintain a low voltage when in the high-impedance or negated state. However, when this signal goes to the high-impedance or negated state, it will first drive the pin low momentarily to reduce the signal fall time.





## SECTION 11 BUS OPERATION

The following section describes the bus signal operation of the MLAPD during data transfer operations (slave and master operation modes), bus arbitration, and bus exceptions. Separate sections describe Motorola bus operation and Intel-compatible bus operation. Functional timing diagrams are included to assist in the definition of signal timing; however, these diagrams are not intended as parametric timing definitions. For detailed timing relationships refer to **SECTION 12 ELECTRICAL SPECIFICATIONS**.

### 11.1 MOTOROLA BUS OPERATION

#### 11.1.1 Slave Operation Mode

In the slave operation mode, the MLAPD is a peripheral slave to the bus master. The MLAPD enters the slave operation mode when  $\overline{CS}$  or  $\overline{IACK}$  is asserted. During slave mode operations, the MLAPD accepts data from or places data on the data bus, according to the level on the  $R/\overline{W}$  pin. The data transferred will either be written into or read from the internal register that is selected by the encoding of A1 and A2 and by the data strobes. Refer to Table 10-2.

This mode of operation is used during the initialization sequence to load the interrupt-vector register (IVR) and the Global Configuration Block (GCB) address into the data register (DR). The slave operation mode is used during normal operation to read the semaphore register (SR) and to write commands into the MLAPD command register (CR).

In slave mode, the MLAPD can operate with a clock input which is synchronous or asynchronous with respect to the host clock, as long as the bus requirements are satisfied. In the functional diagrams showing host operations, the bus master is assumed to be a MC68000, MC68008, or MC68010 with a clock signal identical to the MLAPD clock signal. The state numbers (S0, S1, etc.) refer to the numbering convention for those processors.

**11.1.1.1 HOST PROCESSOR READ CYCLE.** During a host processor read cycle, the MLAPD places data on the data bus and asserts  $\overline{DTACK}$  to indicate that the data is valid. Figure 11-1 shows the functional timing for a byte read cycle on a 16-bit data bus. The timings for even- and odd-byte host reads on a 16-bit data bus or any host read on an 8-bit bus are identical, with the encoding of  $\overline{UDS}/A0$  and  $\overline{LDS}/DS$  selecting the proper byte. The 8-bit SR is always selected during a host processor read cycle, regardless of the A1/A2 encoding, as this register is the only MLAPD register directly readable by the host processor.

If the upper data byte is selected during a host processor read cycle, the MLAPD drives D8–D15 to hex 'FF'.

The MLAPD begins a host read cycle when  $\overline{CS}$  is asserted and the  $R/\overline{W}$  line is high. The MLAPD responds to  $\overline{CS}$  by driving the data bus bytes selected by  $\overline{UDS}/A0$  and  $\overline{LDS}/\overline{DS}$  and by asserting  $\overline{DTACK}$ . Next, the MLAPD waits until both  $\overline{UDS}/A0$  and  $\overline{LDS}/\overline{DS}$  are negated or until  $\overline{CS}$  is negated; then three-states the data lines, and last, negates  $\overline{DTACK}$ .

**11.1.1.2 HOST PROCESSOR WRITE CYCLE.** During a host processor write cycle, the MLAPD accepts data from the data bus and asserts  $\overline{DTACK}$  indicating to the bus master that the data has been loaded into the selected register. The only MLAPD registers that are directly writable by the host processor are the IVR, DR, and CR.

A host processor write cycle begins when  $\overline{CS}$  is asserted and  $R/\overline{W}$  is low. The MLAPD responds by decoding A1, A2,  $\overline{UDS}/A0$ , and  $\overline{LDS}/\overline{DS}$  signals. When a valid register is selected, the MLAPD accepts data from the data bus, places the data into the selected register, and asserts  $\overline{DTACK}$ . Next, the MLAPD waits until both  $\overline{LDS}/\overline{DS}$  and  $\overline{UDS}/A0$  are negated or until  $\overline{CS}$  is negated and then negates  $\overline{DTACK}$ . The timing for this operation is shown in Figure 11-2.

### 11.1.2 Interrupt Acknowledge Cycle

During an interrupt acknowledge cycle, the host processor is responding to an interrupt request from the MLAPD. The timing of an interrupt acknowledge cycle is identical to an odd-byte read cycle, except that it is started by the assertion of an  $\overline{IACK}$  signal rather than  $\overline{CS}$ .  $\overline{CS}$  and  $\overline{IACK}$  are mutually exclusive signals and should not be asserted simultaneously. If  $\overline{IACK}$  is asserted when the MLAPD is bus master, an address error is generated.

The interrupt acknowledge cycle is started by the MLAPD when  $\overline{IACK}$  is asserted and  $\overline{LDS}/\overline{DS}$  is asserted. The MLAPD responds by placing a vector number on D0-D7 and asserting

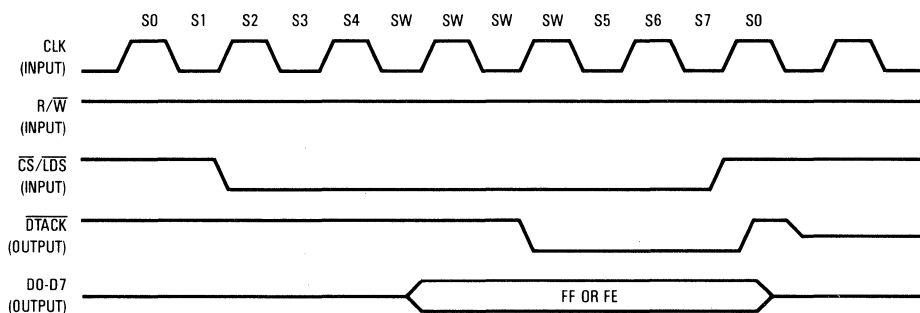
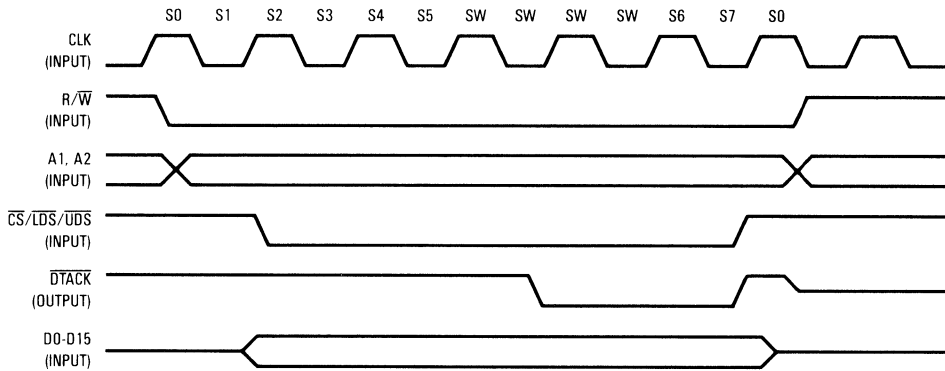


Figure 11-1. Motorola Typical Host Byte Read Cycle Timing Diagram



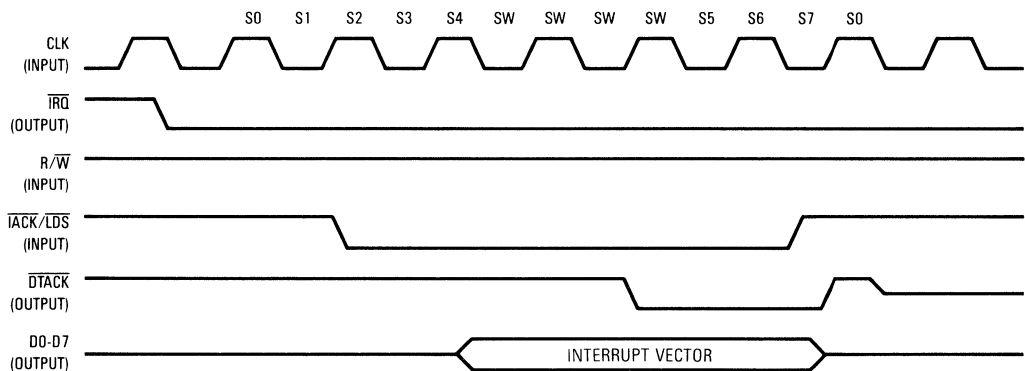
**Figure 11-2. Motorola Typical Host Word Write Cycle Timing Diagram**

$\overline{DTACK}$ . The vector number remains valid on the data bus until  $\overline{IACK}$  or  $\overline{LDS}$  is negated by the host processor. At this time the MLAPD three-states the data lines and negates  $\overline{DTACK}$ . The timing is shown in Figure 11-3.

### 11.1.3 Master Operation

In the master operation mode, the MLAPD is the bus master and performs memory read and write operations. The MLAPD can operate in either an 8-bit or a 16-bit bus configuration.

**11.1.3.1 DMA PRIORITY SCHEME.** The MLAPD has four direct memory access (DMA) channels which are used to access the receive memory buffers, the transmit memory buffers, and the various shared memory tables. During one DMA burst, the MLAPD may access one or more of these memory structures. Commands from the host that are received



**Figure 11-3. Motorola Typical Interrupt Acknowledge Cycle Timing Diagram**

between DMA bursts are handled by the MLAPD microcode and have higher priority than the receiver and transmitter. For each memory access, an internal arbiter determines which section of the MLAPD will be serviced: the microcode controller, the receiver, or the transmitter. The priority scheme used in the online state is listed below.

- 1) Microcode controller, if execution of microcode requires a memory access
- 2) Receiver, if the RxFIFO has more than 15 full words
- 3) Transmitter, if the TxFIFO has more than 15 empty words
- 4) Receiver, if the RxFIFO has more than four full words or the RxFIFO contains a complete frame
- 5) Transmitter, if the TxFIFO has more than four empty words
- 6) Receiver, if the RxFIFO is not empty and the MLAPD is already bus master
- 7) Transmitter, if the TxFIFO is not full and the MLAPD is already bus master

As the state of the receive first-in first-out (RxFIFO) buffer and transmit first-in first-out (TxFIFO) buffer changes, the priority of the required memory accesses for the receiver and transmitter changes. For example: If RxFIFO has two full words and TxFIFO has one empty word, then no DMA activity will be initiated until the microcode requires a DMA cycle. Thus the sequence will be:

- (rule 1, RxFIFO = 2, TxFIFO = 1) microcode
- (rule 6, RxFIFO = 2, TxFIFO = 1) Rx, Rx
- (rule 7, RxFIFO = 0, TxFIFO = 1) Tx

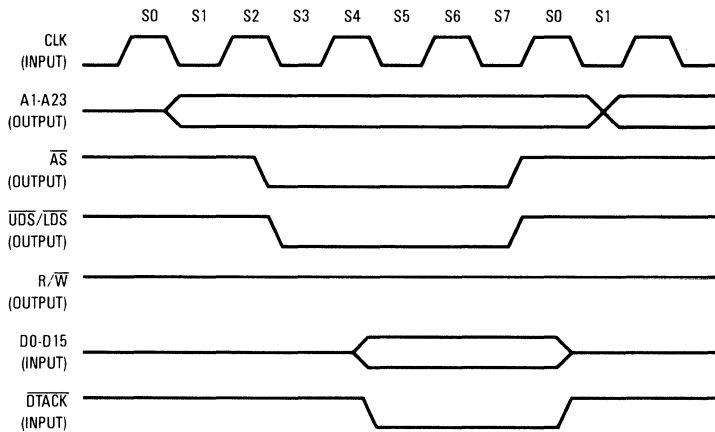
Both receive (Rx) and transmit (Tx) requests are handled during the microcode-initiated DMA burst.

For example: If the RxFIFO has 10 full words, the TxFIFO has 17 empty words, and no commands have been received from the host, then the sequential memory cycles are:

- (rule 3, RxFIFO = 10, TxFIFO = 17) Tx, Tx
- (rule 4, RxFIFO = 10, TxFIFO = 15) Rx, Rx, Rx, Rx, Rx, Rx
- (rule 5, RxFIFO = 4, TxFIFO = 15) Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx
- (rule 6, RxFIFO = 4, TxFIFO = 4) Rx, Rx, Rx, Rx
- (rule 7, RxFIFO = 0, TxFIFO = 4) Tx, Tx, Tx, Tx

**11.1.3.2 MLAPD READ CYCLES.** During a DMA read operation, the MLAPD controls the transfer of data from memory into the MLAPD. The functional timing for a DMA read operation is shown in Figure 11-4. The timing for an even- or odd-byte read on a 16-bit data bus or any read on an 8-bit data bus is identical, with the encoding of  $\overline{\text{UDS/A0}}$  and  $\overline{\text{LDS/DS}}$  selecting the proper byte.

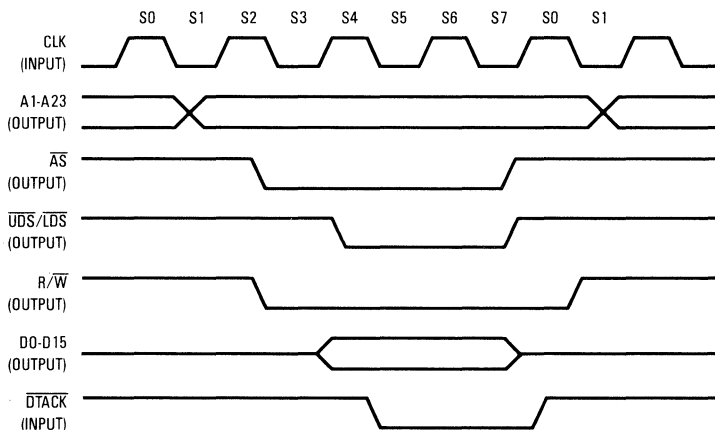
The MLAPD drives the A1–A23 pins with the address of the memory location to be read. Then  $\overline{\text{R/W}}$  is driven high, and  $\overline{\text{AS}}$ ,  $\overline{\text{UDS/A0}}$ , and  $\overline{\text{LDS/DS}}$  are asserted.  $\overline{\text{DTACK}}$  is asserted by memory when valid data is on lines D0–D15. If using an 8-bit bus, only the data on lines D0–D7 is assumed to be valid. When  $\overline{\text{DTACK}}$  is asserted, the data is latched by the MLAPD from the data bus, and the bus cycle is terminated.



**Figure 11-4. Motorola Typical DMA Read Cycle Timing Diagram**

**11.1.3.3 MLAPD WRITE CYCLES.** During a DMA write operation, the MLAPD controls the transfer of data to memory from the MLAPD. The functional timing for this operation is shown in Figure 11-5. The timing for an even- or odd-byte write on a 16-bit data bus or any write on an 8-bit data bus is identical, with the encoding of  $\overline{UDS/A0}$  and  $\overline{LDS/DS}$  selecting the proper byte.

The MLAPD drives the A1–A23 pins with the address of the memory location to be written. Then  $R/\overline{W}$  is driven low,  $\overline{AS}$  is asserted, and, depending on data size,  $\overline{UDS/A0}$  and/or  $\overline{LDS/DS}$  are asserted. Data to be written to memory is placed on the bus, and when  $\overline{DTACK}$  is asserted, the cycle is terminated. On a slow write, the bus cycle is extended because  $\overline{DTACK}$  is not asserted by the end of S4.



**Figure 11-5. Motorola Typical DMA Write Cycle Timing Diagram**

### 11.1.4 Bus Arbitration

Once the host has initialized the MLAPD, the MLAPD uses the following bus arbitration protocol to request bus mastership before entering the master operation mode. The bus arbitration timing is shown in Figure 11-6.

The MLAPD requests control of the system bus by asserting  $\overline{BR}$  when a command is issued that requires a DMA operation or when data needs to be moved to or from the internal FIFOs.  $\overline{BG}$  is asserted by an external bus arbiter to indicate that the MLAPD may assume bus mastership as soon as the current bus master has released the bus.

The MLAPD waits until  $\overline{AS}$  and  $\overline{BGACK}$  are negated and until any bus exceptions clear, before asserting  $\overline{BGACK}$ . The negation of  $\overline{AS}$  indicates that the previous master has completed its bus cycle, and the negation of  $\overline{BGACK}$  indicates that the previous master has released control of the system bus. After the MLAPD asserts  $\overline{BGACK}$ ,  $\overline{BR}$  is negated to allow the external bus arbiter to begin arbitration for the next bus master.

The MLAPD maintains control of the system bus for up to eight bus cycles or until all data transfers have been serviced, based upon the burst-control bit in the option-bits entry of the GCB.  $\overline{BGACK}$  is negated after the last bus cycle is completed. Bus mastership is terminated at the negation of  $\overline{BGACK}$ .

### 11.1.5 Bus Exception Control Functions

To fully support the M68000 bus architecture, the MLAPD has four bus exception inputs: RESET, HALT, BERR, and RETRY. RESET is always recognized by the MLAPD. HALT, RETRY, and BERR are only recognized when the MLAPD has asserted  $\overline{BR}$  to become the bus master or when the MLAPD is the current bus master. When an exception is asserted, the MLAPD terminates the current bus cycle and waits for the exception signal to be negated. Each of the bus exceptions are discussed in detail in the following paragraphs.

Three possible cases for bus exceptions exists. In case one, a very early bus exception occurs when the bus exception is asserted more than one clock cycle before DTACK is

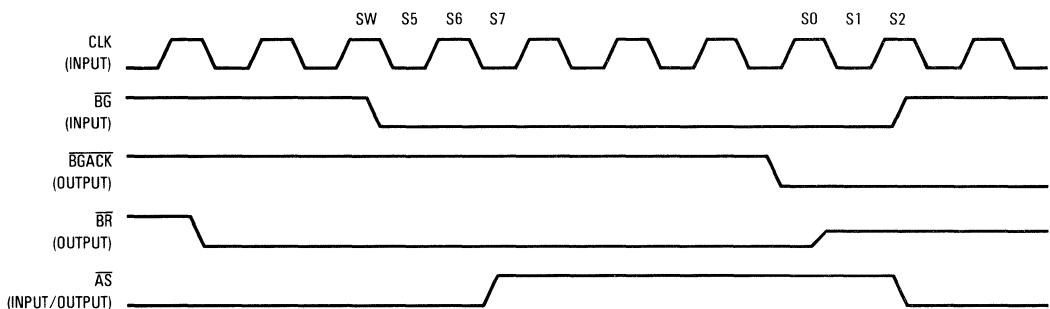


Figure 11-6. Motorola Typical Bus Arbitration Timing Diagram

asserted. In case two, which is the typical case, the bus exception occurs in the same clock cycle as  $\overline{DTACK}$ . In case three, the bus exception signal occurs in the clock cycle after  $\overline{DTACK}$  has been asserted. In all cases, the bus exception is acted on without any delay by the MLAPD, and the bus cycle is terminated.

A late bus exception, which does not meet electrical specification (58), may cause improper behavior of the MLAPD, including system bus lockup.

**11.1.5.1 HALT.** A low level on the  $\overline{HALT}$  pin halts the MLAPD, and the current bus cycle is terminated by the assertion of  $\overline{DTACK}$ . The MLAPD releases ownership of the bus and enters the idle state until the  $\overline{HALT}$  pin returns to a one level. The halt timing diagram is shown in Figure 11-7. The MLAPD rearbiterates for the bus and continues DMA operations, if necessary.

**11.1.5.2 BUS ERROR.** When a low level is detected on the  $\overline{BERR}$  pin, the MLAPD aborts the current bus cycle and releases bus ownership. After the  $\overline{BERR}$  signal returns to a one level, the MLAPD rearbiterates for the bus to report the bus error to the host processor as described in **7.2 BUS ERROR OPERATION**. A bus error condition is shown in Figure 11-8.

**11.1.5.3 RETRY.** A low level on the  $\overline{RETRY}$  pin terminates the current bus cycle and places the MLAPD into a waiting mode. The MLAPD retains bus mastership by keeping  $\overline{BGACK}$

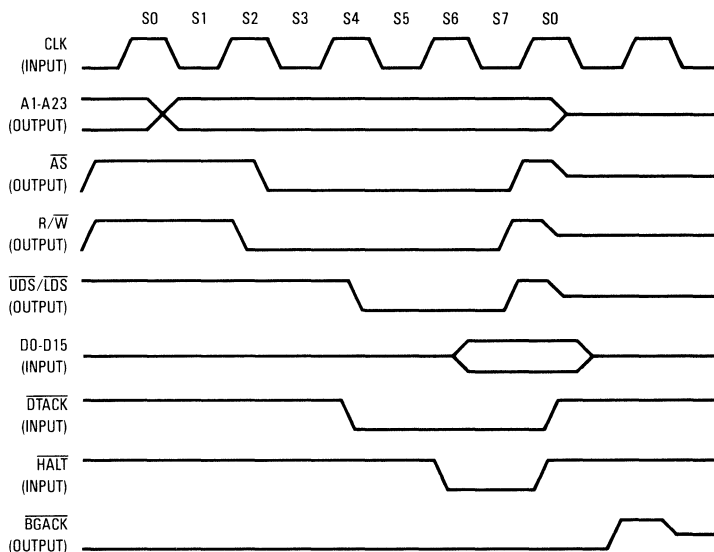
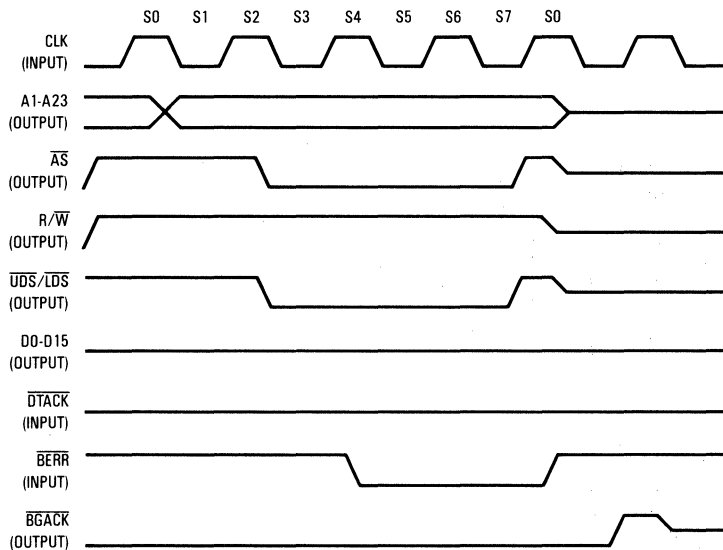


Figure 11-7. Motorola Typical Write Cycle with  $\overline{HALT}$  Timing Diagram



**Figure 11-8. Motorola Typical Read Cycle with  $\overline{\text{BERR}}$  Timing Diagram**

asserted. When the  $\overline{\text{RETRY}}$  signal returns to a one level, the MLAPD reruns the same bus cycle using the same address. Figure 11-9 shows the timing diagram for a retry operation.

**11.1.5.4 RESET.** The MLAPD is reset either by a host processor command or by a hardware reset from an external device. The host processor can issue a reset by passing the MLAPD a RESET command (hex 'FF'). The hardware reset is accomplished by driving the  $\overline{\text{RESET}}$  pin low for at least 10 clock cycles. Normally after either a hardware or software reset, the MLAPD requires four clock cycles before it is ready to accept a subsequent command.

# 11

## 11.2 INTEL-COMPATIBLE BUS OPERATION

### 11.2.1 Slave Operation Mode

In the slave operation mode, the MLAPD is a slave to the bus master. The MLAPD enters the slave operation mode when  $\overline{\text{CS}}$  is asserted. During slave mode operation, the MLAPD accepts data from or places data on the data bus, according to the level on the  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  pins. The data transferred is either written into or read from the internal register that is selected by the encoding of A0, A1, A2 and  $\overline{\text{BHE}}$ .

The slave operation mode is used during the initialization sequence to load the IVR and GCB address into the DR. The slave operation mode is used during normal operation to read the SR and to write commands to the CR. Each memory transfer consists of at least four clock cycles which are referred to as T1, T2, T3, and T4.



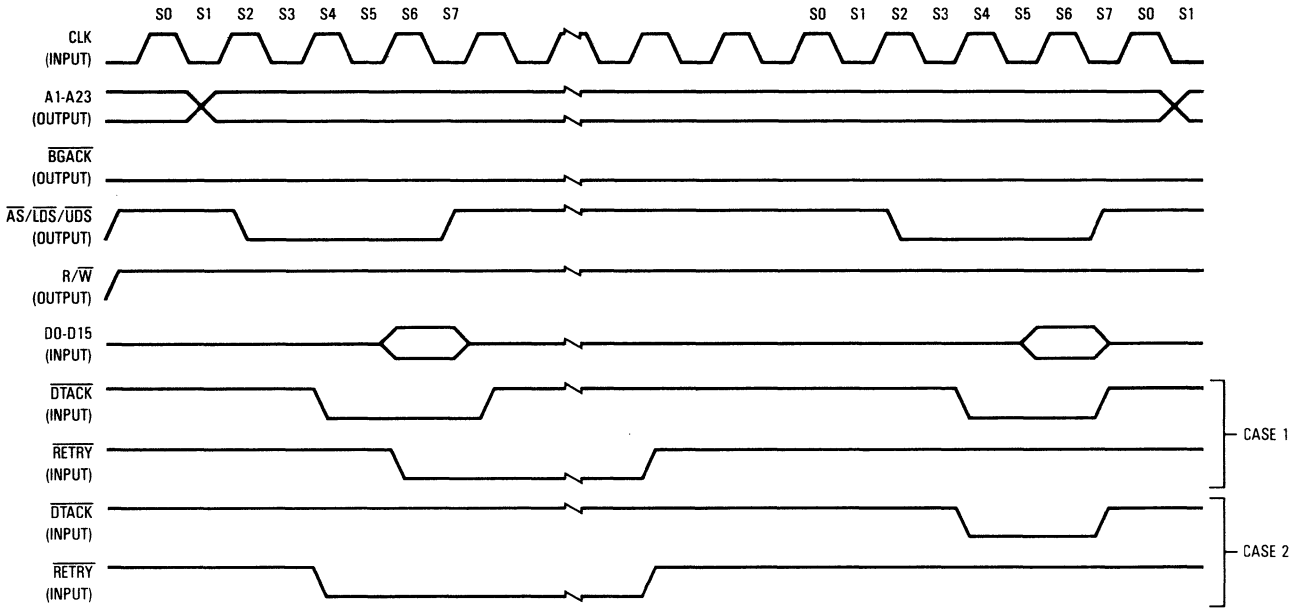


Figure 11-9. Motorola Typical Read Cycle with  $\overline{\text{RETRY}}$  Timing Diagram

**11.2.1.1 HOST PROCESSOR READ CYCLE.** During host processor read cycles, the MLAPD places data on the data bus and asserts READY to indicate that the data is valid on the data bus. Figure 11-10 shows the functional timing for a byte read cycle on a 16-bit data bus. The timing for even- and odd-byte host reads on a 16-bit data bus or any host read on an 8-bit data bus are identical, with the encoding of  $\overline{BHE}$  and A0 selecting the proper byte. The 8-bit SR is always selected during a host processor read cycle, regardless of the A1, A2 encoding, as this register is the only MLAPD register directly readable by the host. When the upper data byte is selected during a host read cycle, the MLAPD drives D8–D15 to hex 'FF'.

The MLAPD begins a host read cycle when  $\overline{CS}$  is asserted and  $\overline{RD}$  is asserted. During the T1 clock cycle, the MLAPD monitors the pins A0, A1, A2, and  $\overline{BHE}$ . The MLAPD drives the appropriate data lines after  $\overline{RD}$  is asserted. When the MLAPD asserts READY, the data may be latched at the next T4 clock cycle, according to set up and hold time specifications.

**11.2.1.2 HOST PROCESSOR WRITE CYCLE.** During host processor write cycles, the MLAPD accepts data from the data bus and asserts READY to indicate that the data has been loaded into the selected register. The only MLAPD registers that are directly writable by the host are the CR, IVR, and the DR. The timing is identical for even- and odd-byte host processor writes to a 16-bit data bus or any host processor write to an 8-bit data bus.

A host processor write cycle begins when  $\overline{CS}$  is asserted and  $\overline{WR}$  is asserted. During the T1 clock cycle, the MLAPD monitors the pins A0, A1, A2, and  $\overline{BHE}$ . These pins select the internal MLAPD register and the data byte(s) to be written. When the MLAPD asserts READY, the data has been latched, and the host may terminate this bus cycle. Data is latched by the falling edge of T3 or Tw prior to the negation of  $\overline{WR}$ . See Figure 11-11.

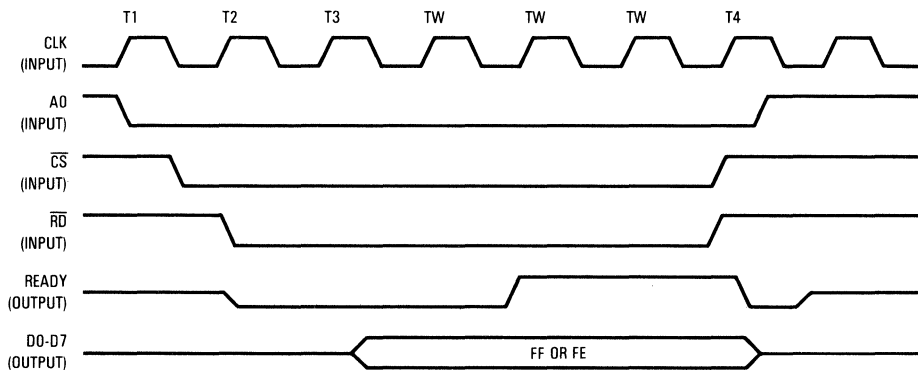
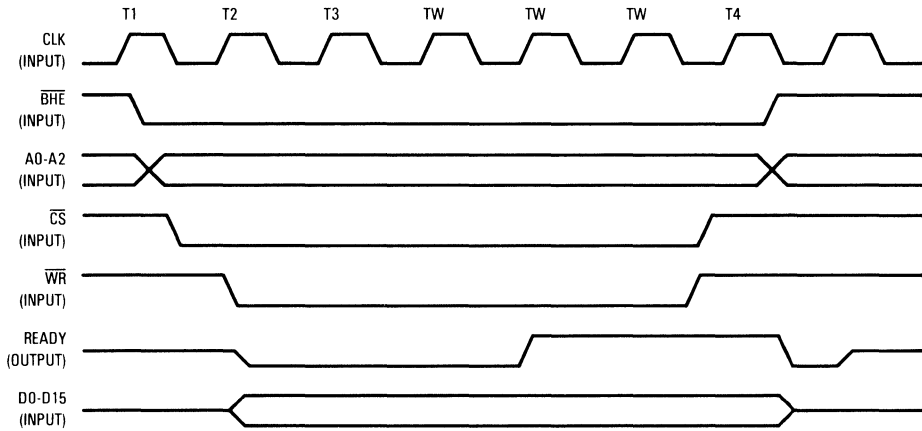


Figure 11-10. Intel-Compatible Typical Host Byte Read Cycle Timing Diagram

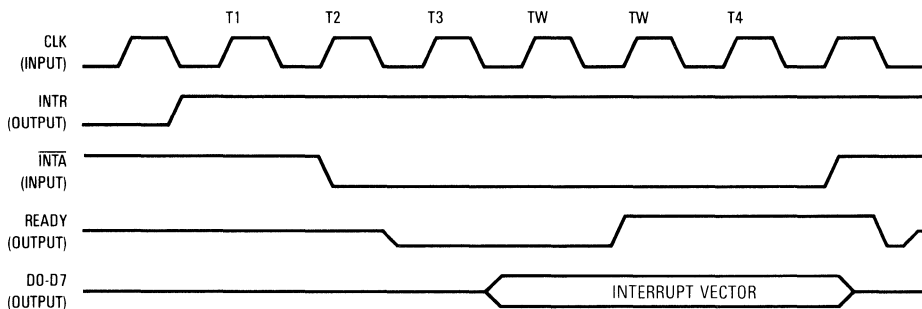


**Figure 11-11. Intel-Compatible Typical Host Word Write Cycle Timing Diagram**

### 11.2.2 Interrupt Acknowledge Cycle

During an interrupt acknowledge cycle, the host processor is responding to an interrupt request from the MLAPD. The timing of an interrupt acknowledge cycle is identical to an even-byte read cycle, except that it is started by the assertion of the  $\overline{\text{INTA}}$  signal rather than  $\overline{\text{CS}}$ .  $\overline{\text{CS}}$  and  $\overline{\text{INTA}}$  are mutually exclusive signals and should not be asserted simultaneously. If  $\overline{\text{INTA}}$  is asserted when the MLAPD is bus master, an address error is generated.

Whenever the MLAPD has an interrupting condition and an external interrupt indication is enabled via the polling\_select GCB entry, INTR is asserted. The host processor acknowledges the interrupt by asserting  $\overline{\text{INTA}}$  during T2. The MLAPD responds to  $\overline{\text{INTA}}$  by placing a vector number on D0–D7 and asserting READY. The vector number remains valid on the data bus until  $\overline{\text{INTA}}$  is negated by the host processor. See Figure 11-12.



**Figure 11-12. Intel-Compatible Typical Interrupt Acknowledge Cycle Timing Diagram**

### 11.2.3 Master Operation

In master operation mode, the MLAPD is the system bus master and performs memory read and write operations. The MLAPD can operate in either an 8-bit or a 16-bit bus configuration.

**11.2.3.1 DMA PRIORITY SCHEME.** The MLAPD has four DMA channels which are used to access the receive memory buffers, the transmit memory buffers, and the various shared memory tables. During one DMA burst, the MLAPD may access one or more of these memory structures. Commands from the host that are received between DMA bursts are handled by the MLAPD microcode and have higher priority than the receiver and transmitter. For each memory access, an internal arbiter determines which section of the MLAPD will be serviced: the microcode controller, the receiver, or the transmitter. The priority scheme used in the online state is listed below.

- 1) Microcode controller, if execution of microcode requires a memory access
- 2) Receiver, if the RxFIFO has more than 15 full words
- 3) Transmitter, if the TxFIFO has more than 15 empty words
- 4) Receiver, if the RxFIFO has more than four full words or the RxFIFO contains a complete frame
- 5) Transmitter, if the TxFIFO has more than four empty words
- 6) Receiver, if the RxFIFO is not empty and the MLAPD is already bus master
- 7) Transmitter, if the TxFIFO is not full and the MLAPD is already bus master

As the state of the RxFIFO and TxFIFO changes, the priority of the required memory accesses for the receiver and transmitter changes. For example: If RxFIFO has two full words and TxFIFO has one empty word, then no DMA activity will be seen until the microcode requires a DMA cycle. Then the sequence will be:

- (rule 1, RxFIFO=2, TxFIFO=1) microcode
- (rule 6, RxFIFO=2, TxFIFO=1) Rx, Rx
- (rule 7, RxFIFO=0, TxFIFO=1) Tx

11

Both Rx and Tx requests are handled during the microcode-initiated DMA burst.

For example: If the RxFIFO has 10 full words, the TxFIFO has 17 empty words, and no commands have been received from the host, then the sequential memory cycles are:

- (rule 3, RxFIFO=10, TxFIFO=17) Tx, Tx
- (rule 4, RxFIFO=10, TxFIFO=15) Rx, Rx, Rx, Rx, Rx, Rx
- (rule 5, RxFIFO=4, TxFIFO=15) Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx, Tx
- (rule 6, RxFIFO=4, TxFIFO=4) Rx, Rx, Rx, Rx
- (rule 7, RxFIFO=0, TxFIFO=4) Tx, Tx, Tx, Tx

**11.2.3.2 MLAPD READ CYCLE.** During a DMA read operation, the MLAPD controls the transfer of data from memory into the MLAPD. The functional timing for a DMA read operation is shown in Figure 11-13. The timing for an even- or odd-byte read on a 16-bit

data bus or any read on an 8-bit data bus is identical, with the encoding of  $\overline{BHE}$  and A0 selecting the proper byte.

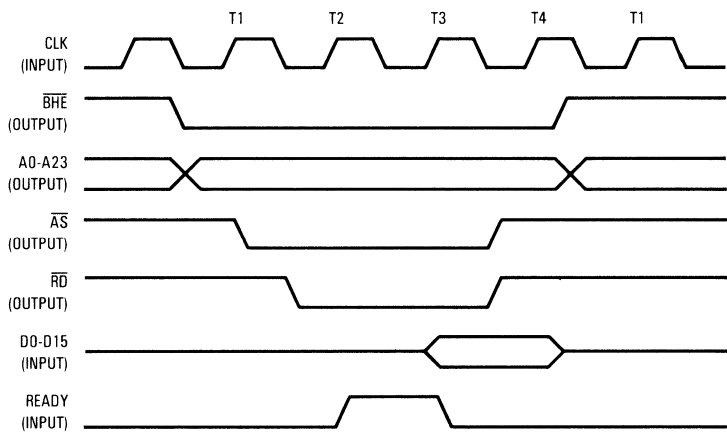
The MLAPD read cycle begins with the generation of an address. The memory  $\overline{RD}$  signal is asserted at T2.  $\overline{RD}$  causes the addressed device to enable its system data bus drivers. When data is valid on the bus, the READY line is asserted. After  $\overline{RD}$  is negated by the MLAPD, the addressed device again three-states its data bus drivers. If READY is not asserted during T3 by the addressed device, the MLAPD inserts wait states between T3 and T4, and data is latched on the falling edge of the last wait cycle.  $\overline{RD}$  is negated during T4.

**11.2.3.3 MLAPD WRITE CYCLE.** During a DMA write operation, the MLAPD controls the transfer of data to memory from the MLAPD. The functional timing for this operation is shown in Figure 11-14. The timing for an even- or odd-byte write on a 16-bit data bus or any write to an 8-bit data bus is identical, with the encoding of  $\overline{BHE}$  and A0 selecting the proper byte.

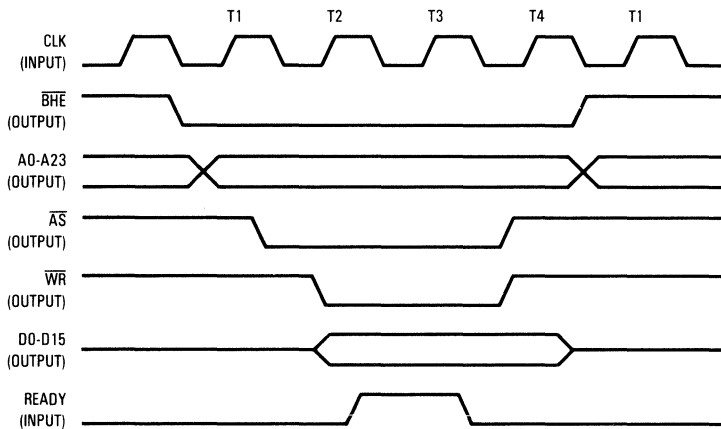
The MLAPD write cycle begins with the generation of an address. At T3 the MLAPD drives the data onto the data bus. This data remains valid until the middle of T4. The  $\overline{WR}$  signal is asserted during T3. The  $\overline{WR}$  signal remains asserted during any Tw states. When READY is asserted, the cycle is terminated.

**11.2.4 Bus Arbitration**

Once the host has initialized the MLAPD, the MLAPD uses the following bus arbitration protocol to request bus mastership before entering the DMA operation mode. The bus arbitration timing is shown in Figure 11-15.



**Figure 11-13. Intel-Compatible Typical DMA Read Cycle Timing Diagram**



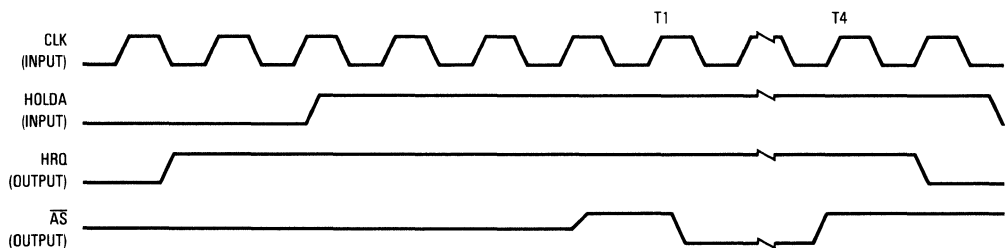
**Figure 11-14. Intel-Compatible Typical DMA Write Cycle Timing Diagram**

The MLAPD requests control of the system bus by asserting HRQ when a command is issued that requires a DMA operation or when data needs to be moved to or from the internal FIFOs. HOLDA is asserted by an external bus arbiter to indicate that the MLAPD may assume bus mastership. The MLAPD maintains control of the system bus for up to eight bus cycles or until all data transfers have been serviced, based upon the burst-control bit in the option-bits entry in the GCB. HRQ is negated only after the MLAPD has completed its last transfer. After HRQ is negated, the external arbiter negates HOLDA. If HOLDA is negated before HRQ, the MLAPD will act as if HALT has been activated. The current bus cycle is terminated by the assertion of READY. The MLAPD releases ownership of the bus but still asserts HRQ. The MLAPD will rearbtrate for the bus and continue DMA operations, if necessary.

### 11.2.5 Bus Exception Control Functions

11

The MLAPD has four bus exception inputs:  $\overline{\text{RESET}}$ ,  $\overline{\text{HALT}}$ ,  $\overline{\text{BERR}}$ , and  $\overline{\text{RETRY}}$ .  $\overline{\text{RESET}}$  is always recognized by the MLAPD.  $\overline{\text{HALT}}$ ,  $\overline{\text{RETRY}}$  and  $\overline{\text{BERR}}$  are only recognized when the



**Figure 11-15. Intel-Compatible Typical Bus Arbitration Timing Diagram**

MLAPD has asserted HRQ to become the bus master or when the MLAPD is the current bus master. When an exception is asserted, the MLAPD will terminate the current bus cycle and wait for the exception signal to be negated. Each of the possible conditions is discussed in detail in the following paragraphs.

Three possible cases for bus exceptions exist. In case one, a very early bus exception occurs when the bus exception is asserted more than one clock cycle before READY is asserted. In case two, which is the typical case, the bus exception occurs in the same clock cycle as READY. In case three, the bus exception signal occurs in the clock cycle after READY has been asserted. In all cases, the bus exception is acted on without any delay by the MLAPD, and the bus cycle is terminated.

A late bus exception, which does not meet electrical specification (58), may cause improper behavior of the MLAPD, including system bus lockup.

**11.2.5.1 HALT.** A low level on the  $\overline{\text{HALT}}$  pin halts the MLAPD, and the current bus cycle is terminated by the assertion of READY. The MLAPD releases ownership of the bus and enters the idle state until the  $\overline{\text{HALT}}$  pin returns to a one level. The halt timing diagram is shown in Figure 11-16. The MLAPD rearbiterates for the bus and continues DMA operations, if necessary.

**11.2.5.2 BUS ERROR.** When a low level is detected on the  $\overline{\text{BERR}}$  pin, the MLAPD aborts the current bus cycle and releases bus ownership. After the  $\overline{\text{BERR}}$  signal returns to a one

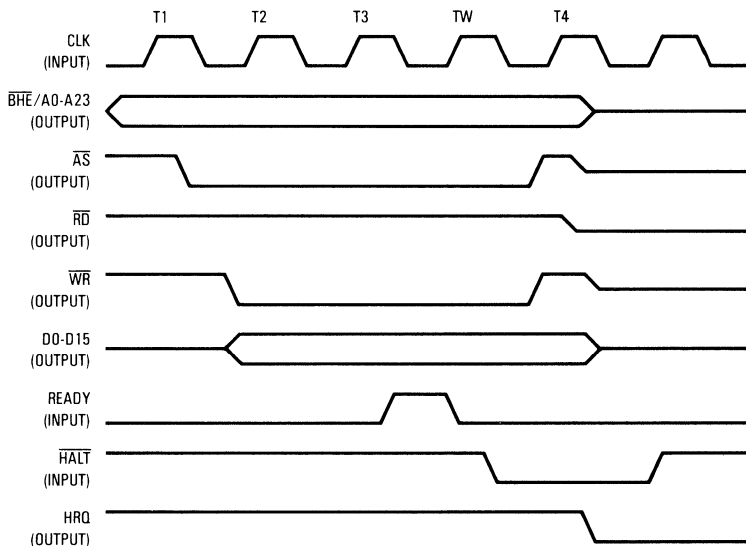


Figure 11-16. Intel-Compatible Typical Write Cycle with  $\overline{\text{HALT}}$  Timing Diagram

level, the MLAPD rearbitrates for the bus to report the bus error to the host processor as described in **7.2 BUS ERROR OPERATION**. A bus error condition is shown in Figure 11-17.

**11.2.5.3 RETRY.** A low level on the  $\overline{\text{RETRY}}$  pin terminates the current bus cycle and places the MLAPD into a waiting mode. The MLAPD retains bus mastership by keeping  $\overline{\text{HRQ}}$  asserted. When the  $\overline{\text{RETRY}}$  signal returns to a one level, the MLAPD reruns the same bus cycle using the same address. Figure 11-18 shows the timing diagram for a retry operation.

**11.2.5.4 RESET.** The MLAPD is reset either by a host processor command or by a hardware RESET from an external device. The host processor can issue a reset by passing the MLAPD a RESET command (hex 'FF'). The hardware reset is accomplished by driving the  $\overline{\text{RESET}}$  pin low for at least 10 clock cycles. Normally after either a hardware or software reset, the MLAPD requires four clock cycles before it is ready to accept a subsequent command.

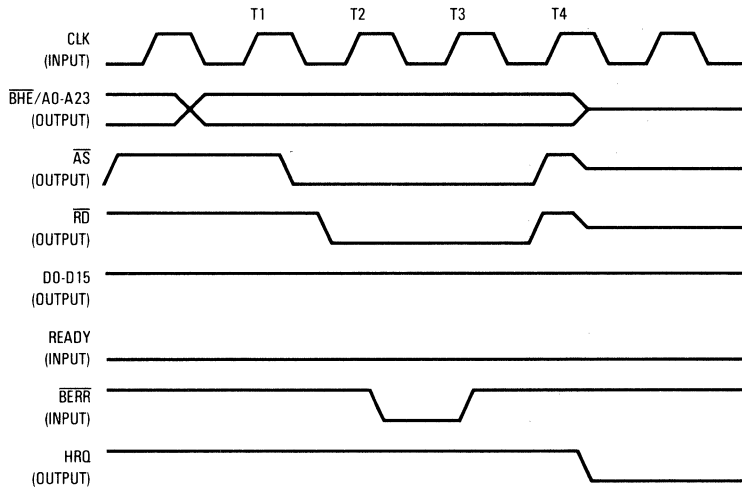


Figure 11-17. Intel-Compatible Typical Read Cycle with  $\overline{\text{BERR}}$  Timing Diagram



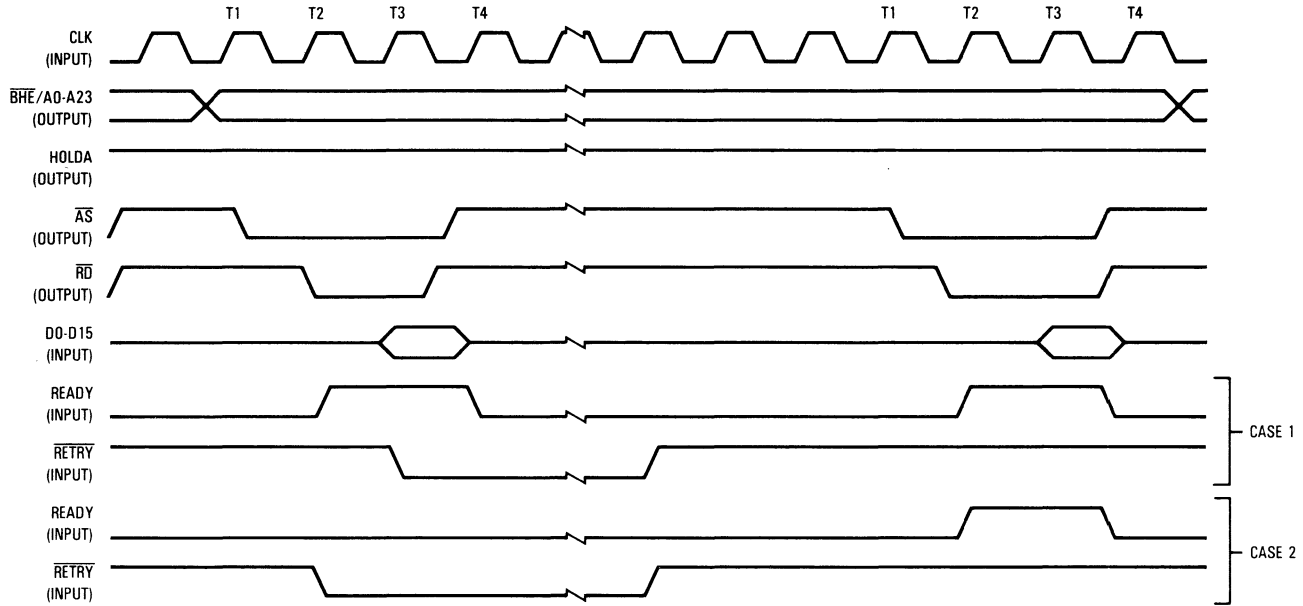


Figure 11-18. Intel-Compatible Typical Read Cycle with  $\overline{\text{RETRY}}$  Timing Diagram



## SECTION 12 ELECTRICAL SPECIFICATIONS

This section contains the electrical specifications and associated timing information for the MC68606 MLAPD.

### 12.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V <sub>DD</sub>	-0.3 to +7.0	V
Input Voltage	V <sub>in</sub>	-0.3 to +7.0	V
Operating Temperature Range	T <sub>A</sub>	0 to 70	°C
Storage Temperature Range	T <sub>stg</sub>	-55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than the maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g. either ground or V<sub>cc</sub>).

### 12.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance PGA	θ <sub>JA</sub>	33	°C/W

$$T_J = T_A + (P_D \times \theta_{JA})$$

$$P_D = (V_{DD} \times I_{DD}) + P_{I/O}$$

where:

P<sub>I/O</sub> is the lower dissipation on pins (user determined) which can be neglected in most cases.

For T<sub>A</sub> = 70°C and P<sub>D</sub> = 0.55 Watts at 12.5 MHz

$$T_J = 88^\circ\text{C}$$

### 12.3 POWER CONSIDERATIONS

The average chip-junction temperature, T<sub>J</sub>, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T<sub>A</sub> = Ambient Temperature, °C

θ<sub>JA</sub> = Package Thermal Resistance, Junction-to-Ambient, °C/W

P<sub>D</sub> = P<sub>INT</sub> + P<sub>I/O</sub>

P<sub>INT</sub> = I<sub>DD</sub> × V<sub>DD</sub>, Watts — Chip Internal Power

P<sub>I/O</sub> = Power Dissipation on Input and Output Pins, Watts — User Determined

For most applications P<sub>I/O</sub> < P<sub>INT</sub> and can be neglected.

The following is an approximate relationship between P<sub>D</sub> and T<sub>J</sub> (if P<sub>I/O</sub> is neglected):

$$P_D = K \div (T_J + 273^\circ\text{C}) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273^\circ\text{C}) + \theta_{JA} \cdot P_D^2$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring  $P_D$  (at equilibrium) for a known  $T_A$ . Using this value of K, the values of  $P_D$  and  $T_J$  can be obtained by solving equations (1) and (2) iteratively for any value of  $T_A$ .

## 12.4 DC ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	Min	Max	Unit
Input High Voltage (Except System Clock)	$V_{IH}$	2.0	$V_{DD}$	V
Input Low Voltage (Except System Clock)	$V_{IL}$	$V_{SS} - 0.3$	0.8	V
Input High Voltage (System Clock)	$V_{CIH}$	2.4	$V_{DD}$	V
Input Low Voltage (System Clock)	$V_{CIL}$	$V_{SS} - 0.3$	0.5	V
Input Leakage Current	$I_{in}$	—	20	$\mu\text{A}$
Input Capacitance	$C_{in}$	—	13	pF
Three-State Leakage Current (2.4/0.5 V)	$I_{TSI}$	—	20	mA
Open-Drain Leakage Current (2.4 V)	$I_{OD}$	—	20	V
Output High Voltage ( $I_{OH} = 400 \mu\text{A}$ )	$V_{OH}$	2.4	—	V
Output Low Voltage ( $I_{OL} = 3.2 \text{ mA}$ ) A1-A31, $\overline{\text{RTS}}$ , $\overline{\text{TSTART}}$ , $\overline{\text{TxD}}$ , A0, $\overline{\text{UDS/A0}}$ as A0 ( $I_{OL} = 5.3 \text{ mA}$ ) D0-D15, AS, BHE, LDS, A0, $\overline{\text{UDS/A0}}$ as $\overline{\text{UDS}}$ , READY, $\overline{\text{DTACK}}$ , RD, WR, R/W, $\overline{\text{BGACK}}$ $I_{OL} = 8.9 \text{ mA}$ ) HRO, BR, INTR, $\overline{\text{IRQ}}$	$V_{OL}$	—	0.5	V
Power Dissipation ( $\alpha$ 12.5 MHz, 0°C) ( $\alpha$ 16.67 MHz, 0°C)	$P_D$	—	0.55 0.75	W

## 12.5 AC ELECTRICAL CHARACTERISTICS

This segment contains the parametric numbers and timing diagrams for both Motorola and Intel bus operations. Each reference is standalone.

### 12.5.1 Motorola AC Electrical Characteristics

The MLAPD is characterized at two rated frequencies, 12.5 MHz and 16.67 MHz. To use the MLAPD at system clock frequencies between 8 MHz and the rated clock frequency, the designer should use the electrical characteristics for the rated frequency of the MLAPD. The characteristics that are specified in nanoseconds are guaranteed for system clock frequencies between 8 MHz and the rated frequency. The characteristics that are specified as clock periods refer to the period of actual system clock.

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
1	Asynchronous Input Set-up Time	10	—	10	—	ns
2	$\overline{UDS}$ , $\overline{LDS}$ Inactive to $\overline{CS}$ , $\overline{IACK}$ Inactive	0	80	0	60	ns
3	CLK Low (on which $\overline{UDS}$ or $\overline{LDS}$ and $\overline{CS}$ or $\overline{IACK}$ are recognized) to Data-Out Valid (see note 5)	—	80	—	60	ns
4	$\overline{CS}$ or $\overline{IACK}$ High to Data-Out High-Impedance (see note 8)	—	40	—	30	ns
5	$\overline{LDS}$ , $\overline{DS}$ High to Data-Out Invalid	0	—	0	—	ns
6	$\overline{CS}$ or $\overline{IACK}$ Low to $\overline{DTACK}$ High (Driving Three-State $\overline{DTACK}$ High)	—	40	—	30	ns
7	CLK Low (on which $\overline{UDS}$ or $\overline{LDS}$ and $\overline{CS}$ or $\overline{IACK}$ are recognized) to $\overline{DTACK}$ Low (see note 5)	— —	2+ 50	— —	2+ 45	Clk. Per. ns
8	CLK Low to $\overline{DTACK}$ Low	—	50	—	45	ns
9	Data-Out Valid to $\overline{DTACK}$ Low	20	—	20	—	ns
10	$\overline{DTACK}$ Low to $\overline{UDS}$ , $\overline{LDS}$ , $\overline{CS}$ , or $\overline{IACK}$ High (earliest one)	0	—	0	—	ns
11	$\overline{CS}$ or $\overline{IACK}$ or Data Strokes (Earliest One) High to $\overline{DTACK}$ High (see note 9)	—	40	—	30	ns
12	$\overline{DTACK}$ High to $\overline{DTACK}$ High-Impedance (at the end of bus cycle after $\overline{CS}/\overline{IACK}$ High).	—	40	—	30	ns
13	$\overline{UDS}$ , $\overline{LDS}$ Inactive Time	1	—	1	—	Clk. Per.
14	$\overline{CS}$ , $\overline{IACK}$ Inactive Time	0	—	0	—	ns
15	A1-A2 Valid to $\overline{UDS}/\overline{LDS}/\overline{CS}$ (latest one) Low (Write)	20	—	10	—	ns
16	$\overline{DTACK}$ Low to Data-In Hold Time	0	—	0	—	ns
17	$\overline{UDS}$ or $\overline{LDS}$ , $\overline{CS}$ (latest one) Low to Data-In Valid	—	50	—	40	ns
18	R/ $\overline{W}$ Valid to $\overline{UDS}$ or $\overline{LDS}$ , $\overline{CS}$ or $\overline{IACK}$ (latest one) Low	20	—	10	—	ns
19	$\overline{UDS}$ , $\overline{LDS}$ High to R/ $\overline{W}$ High, A1-A2 Invalid	0	—	0	—	ns
20	CLK High to $\overline{IRQ}$ Low	—	80	—	60	ns
21	Reserved	—	—	—	—	—
22	Reserved	—	—	—	—	—
23	CLK High to $\overline{BR}$ Low	—	40	—	30	ns
24	CLK High to $\overline{BR}$ High-Impedance	—	40	—	30	ns
25	$\overline{BGACK}$ Low to $\overline{BR}$ High-Impedance	20	—	10	—	ns
26	Reserved	—	—	—	—	—
27	CLK Low to $\overline{BGACK}$ Low	—	50	—	40	ns
28	CLK High to $\overline{BGACK}$ High-Impedance	—	40	—	30	ns
29	$\overline{AS}$ and $\overline{BGACK}$ Input High (latest one) to $\overline{BGACK}$ Output Low (when $\overline{BG}$ is previously asserted)	2+ 15	3+ 55	2+ 10	3+ 40	Clk. Per. ns
30	$\overline{BG}$ Low to $\overline{BGACK}$ Low (No Other Bus Master)	2+ 15	3+ 55	2+ 10	3+ 40	Clk. Per. ns
31	$\overline{BR}$ High-Impedance to $\overline{BG}$ High	0	—	0	—	ns
32	CLK Low on which $\overline{BGACK}$ Low to CLK High on which $\overline{AS}$ Low	1.5	1.5	1.5	1.5	Clk. Per.
33	CLK Low to $\overline{BGACK}$ High	—	40	—	30	ns

Continued

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
34	CLK on which $\overline{BR}$ Low to CLK on which $\overline{BGACK}$ Low (Assuming that $\overline{BG}$ is Active and $\overline{AS}$ and $\overline{BGACK}$ are Inactive for at least 2 Clk. Per.)	— —	1+ 55	— —	1+ 45	Clk. Per. ns
35	Clock on which $\overline{AS}$ is High to Clock on which $\overline{BGACK}$ is High	—	1	—	1	Clk. Per.
37	CLK High to Address High-Impedance	—	50	—	40	ns
38	CLK High to Address Valid	—	80	—	60	ns
39	Address Valid to $\overline{AS}$ Low	15	—	10	—	ns
40	CLK High to $\overline{AS}$ , $\overline{UDS}$ , $\overline{LDS}$ Low	—	40	—	30	ns
41	CLK to $\overline{AS}$ , $\overline{UDS}$ , $\overline{LDS}$ High (see note 7)	—	40	—	30	ns
42	$\overline{AS}$ High to Address Invalid	15	—	10	—	ns
43	CLK High to $\overline{AS}$ , $\overline{UDS}$ , $\overline{LDS}$ High Impedance	—	40	—	30	ns
44	CLK to $R/\overline{W}$ High (see Note 4)	—	40	—	30	ns
45	CLK Low to $R/\overline{W}$ High-Impedance	—	40	—	30	ns
46	$\overline{UDS}$ , $\overline{LDS}$ High to Data-In Invalid	0	—	0	—	ns
47	$\overline{AS}$ , $\overline{UDS}$ , $\overline{LDS}$ High to $\overline{DTACK}$ High (the earliest of $\overline{AS}$ , $\overline{UDS}$ , or $\overline{LDS}$ )	0	80	0	60	ns
48	Data-In to CLK Low Required when $\overline{DTACK}$ Satisfies (1) (see note 1)	10	—	5	—	ns
49	$\overline{DTACK}$ Low to Data-In Valid Required when $\overline{DTACK}$ Does Not Satisfy (1) (see notes 1 and 2)	—	50	—	40	ns
50	CLK High to $R/\overline{W}$ Low	—	40	—	30	ns
51	$\overline{AS}$ Low to Data-Out Valid (Write)	— —	0.5+ 40	— —	0.5+ 30	Clk. Per. ns
52	CLK Low to Data-Out Valid	—	45	—	35	ns
53	Data-Out Valid to $\overline{UDS}$ , $\overline{LDS}$ Low	10	—	10	—	ns
54	$\overline{UDS}$ , $\overline{LDS}$ High to Data-Out Invalid	10	—	10	—	ns
55	CLK High to Data-Out Invalid	0	80	0	60	ns
56	No Exception to $\overline{BR}$ Low	1.5+ 10	2.5+ 50	1.5+ 10	2.5+ 40	Clk. Per. ns
57	$\overline{DTACK}$ Low to Asynchronous Exception Active Required when $\overline{DTACK}$ Does Not Satisfy (1) (see note 2)	—	40	—	30	ns
58	Exception Active to CLK Low Setup Time Synchronous Input ("late exception") Required when $\overline{DTACK}$ Satisfies (1) (see note 1)	30	—	20	—	ns
59	Exception Active to CLK Low Setup Time Asynchronous Input (Required when $\overline{DTACK}$ is Absent) (see note 3)	10	—	10	—	ns
60	$\overline{AS}$ , $\overline{UDS}$ , $\overline{LDS}$ High to Exception Inactive	0	—	0	—	ns
61	Reserved	—	—	—	—	—
62	Reserved	—	—	—	—	—
63	$\overline{RESET}$ Width	10	—	10	—	Clk. Per.
64	CLK Frequency	8	12.5	8	16.66	MHz

- Continued

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
65	CLK Period	80	125	60	125	ns
66	CLK Width High (see note 6)	35	62.5	25	62.5	ns
67	CLK Rise/Fall Time (see note 6)	—	5	—	5	ns
68	CLK Width Low (see note 6)	35	62.5	25	62.5	ns
69	RxCLK, TxCLK Frequency	0	12.5	0	16.67	MHz
70	RxD, $\overline{\text{RSTART}}$ Valid to RxCLK High Setup Time	25	—	25	—	ns
71	RxCLK High to RxD, $\overline{\text{RSTART}}$ Hold Time	15	—	15	—	ns
72	RxCLK, TxCLK Rise/Fall Time	5	—	5	—	ns
73	RxCLK, TxCLK Width Low	35	—	25	—	ns
74	RxCLK, TxCLK Width High	35	—	25	—	ns
75	RxCLK, TxCLK Period	80	—	60	—	ns
76	TxCLK Low to TxD, $\overline{\text{RTS}}$ , $\overline{\text{TSTART}}$ Valid	10	40	10	40	ns
77	Reserved	—	—	—	—	—
78	$\overline{\text{CTS}}$ Low to TxCLK High Set-up Time	25	—	25	—	ns

NOTES:

1. If  $\overline{\text{DTACK}}$  satisfies the asynchronous set-up time (1), then (48) is required for the data-in set-up time and (58) for the synchronous exception setup time. Erroneous behavior may occur if (58) is not satisfied.
2. If  $\overline{\text{DTACK}}$  does not satisfy (1), then (49) is required for data-in and (57) for the exception. Erroneous behavior may occur if (57) is not satisfied.
3. Active exception when  $\overline{\text{DTACK}}$  is absent must satisfy the asynchronous set-up time (59).
4.  $\overline{\text{R/W}}$  rises at the end of a write cycle, on the high clock edge following S7. When the MLAPD acquires the bus,  $\overline{\text{R/W}}$  is three-stated until S1 and rises on that low clock.
5. Data (3) and  $\overline{\text{DTACK}}$  (7) will be timed from the latest of  $\overline{\text{CS}}$  and either data strobe during an MPU cycle. Data (3) and  $\overline{\text{DTACK}}$  (7) will be timed from the latest of  $\overline{\text{IACK}}$  and either data strobe during an IACK cycle.
6. These specifications reflect tolerances on the CLK circuit. However, a 50% duty cycle CLK input should be provided.
7.  $\overline{\text{AS}}$  rises at the end of a cycle on the low clock of S7. When the MLAPD acquires the bus,  $\overline{\text{AS}}$  is three-stated until S0 and rises on that high clock.
8. If  $\overline{\text{CS}}$  or  $\overline{\text{IACK}}$  is negated before  $\overline{\text{UDS/LDS}}$ , the data bus will be three-stated (4) possibly before  $\overline{\text{UDS/LDS}}$  negation.

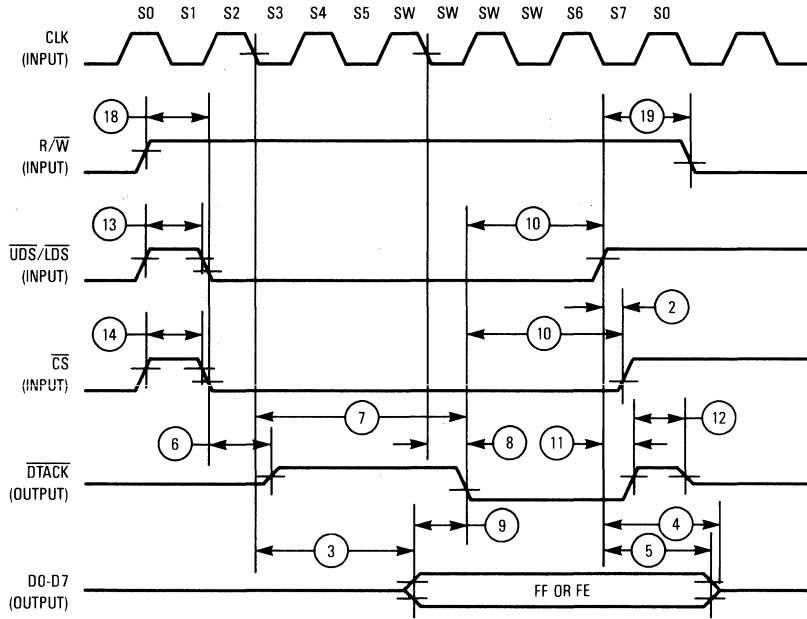


Figure 12-1. Motorola Host Processor Read Cycle Timing Diagram



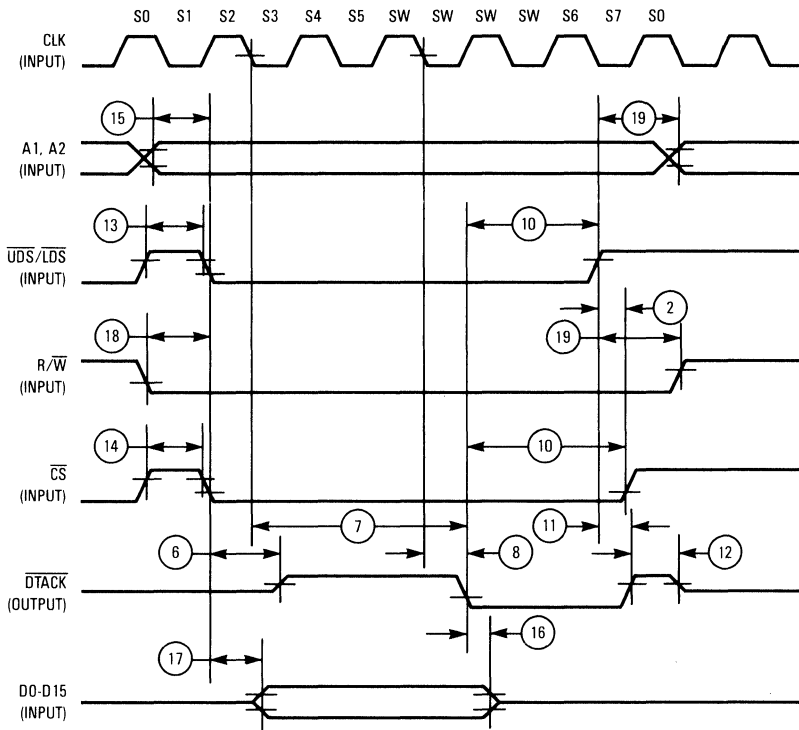


Figure 12-2. Motorola Host Processor Write Cycle Timing Diagram

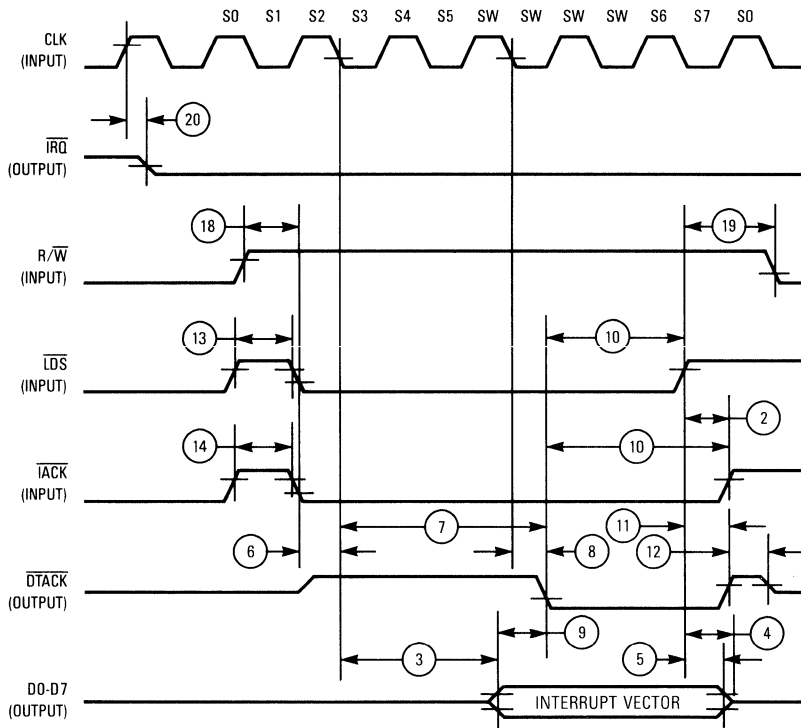


Figure 12-3. Motorola Interrupt Acknowledge Cycle Timing Diagram

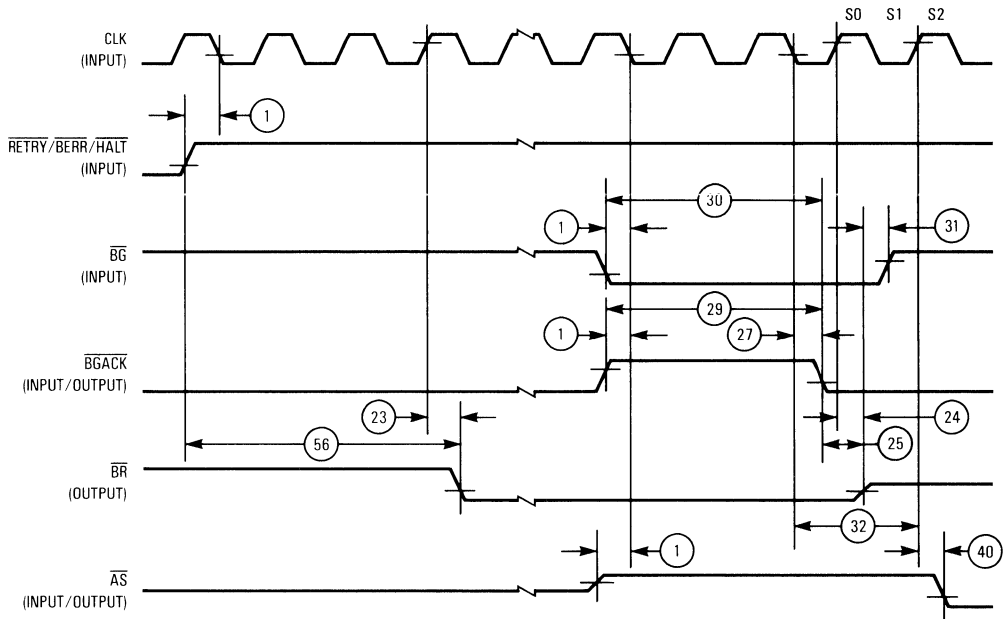


Figure 12-4. Motorola Bus Arbitration Timing Diagram

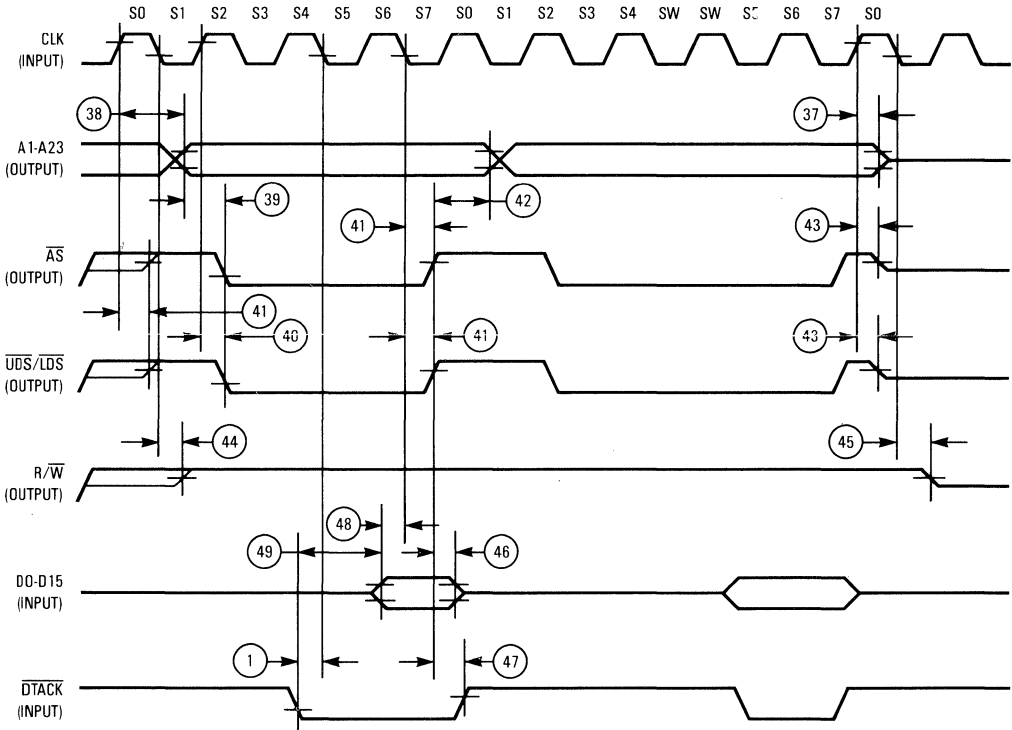


Figure 12-5. Motorola DMA Read Cycle and Slow Read Cycle Timing Diagram

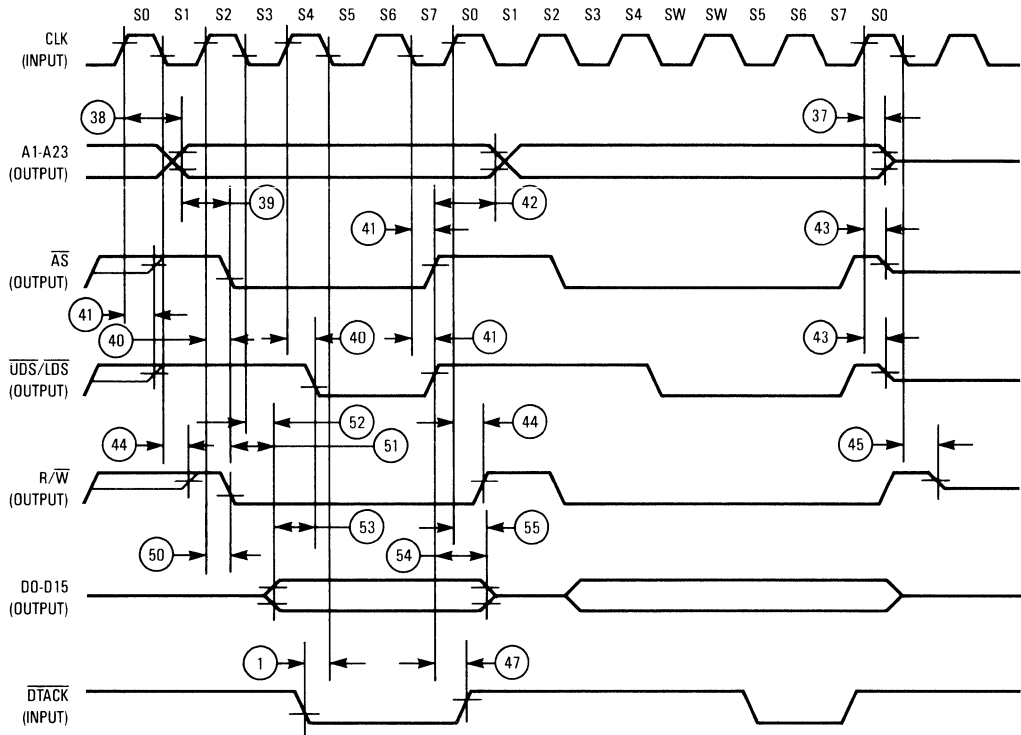


Figure 12-6. Motorola DMA Write Cycle and Slow Write Cycle Timing Diagram

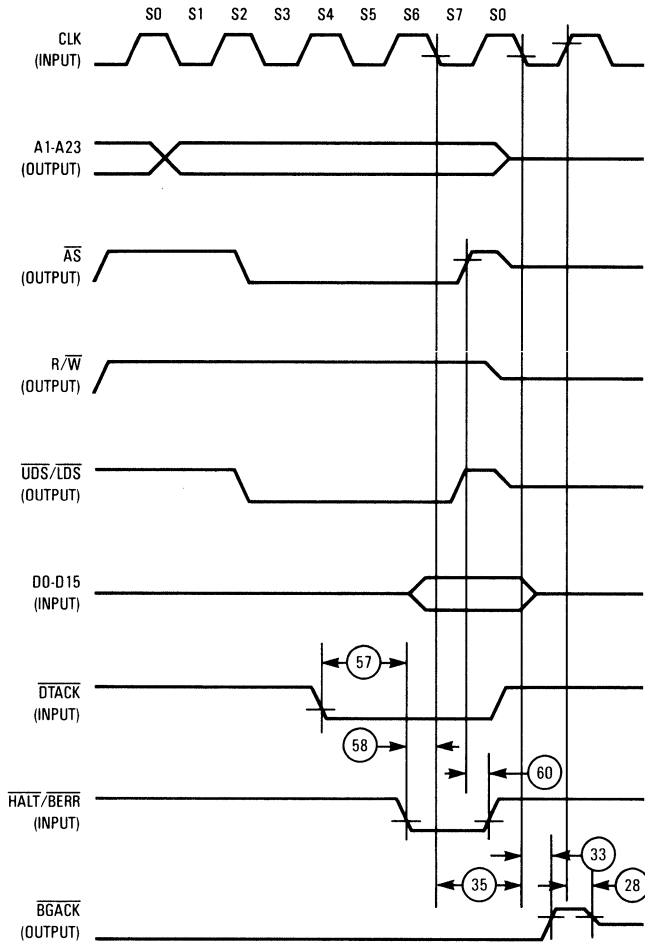


Figure 12-7. Motorola Late Synchronous Exception  $\overline{DTACK}$  Active Timing Diagram

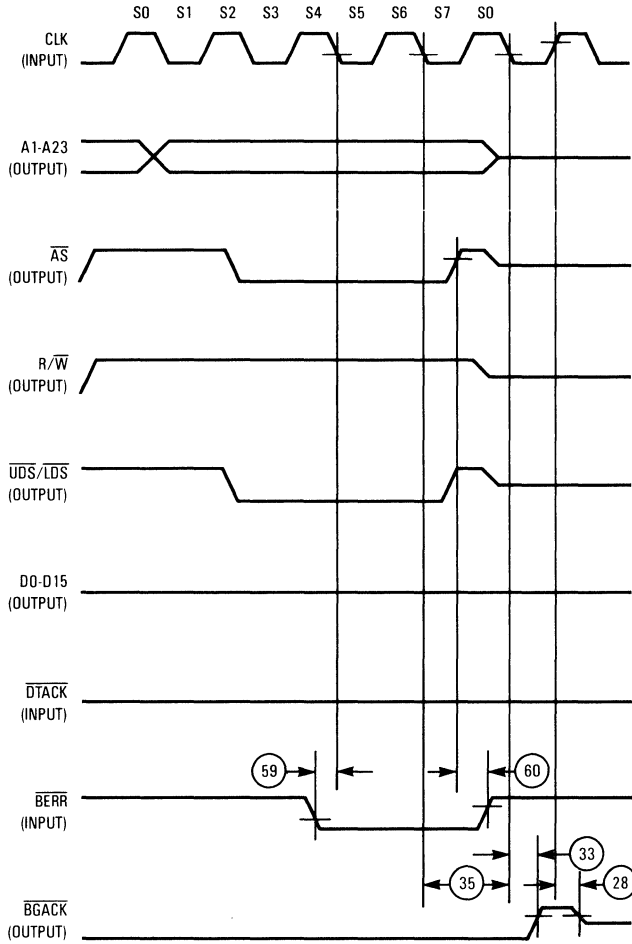


Figure 12-8. Motorola Early Synchronous Exception DTACK Not Active Timing Diagram

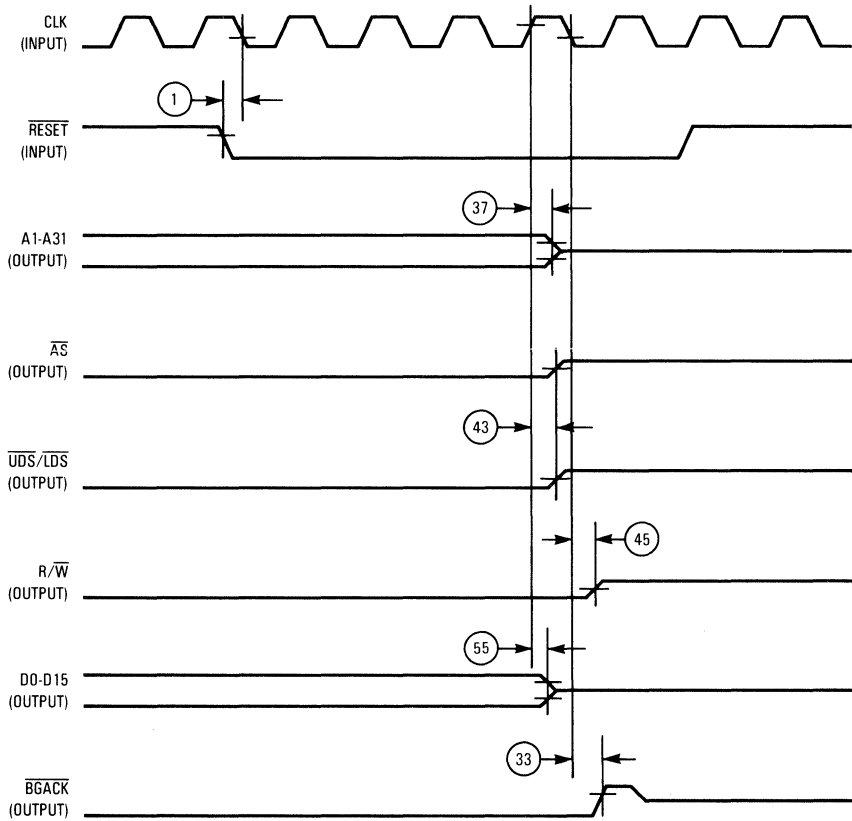


Figure 12-9. Motorola Hardware RESET Timing Diagram



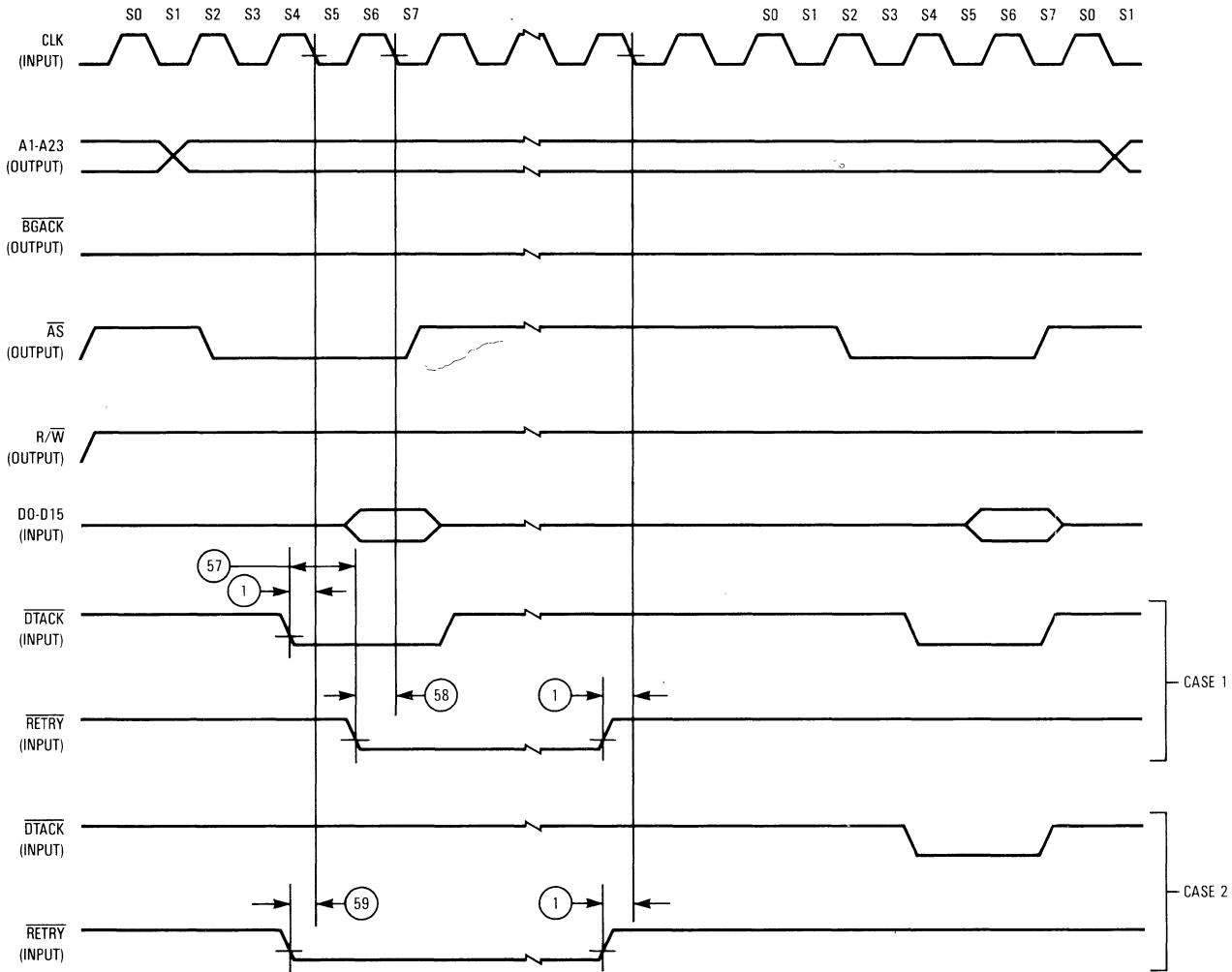
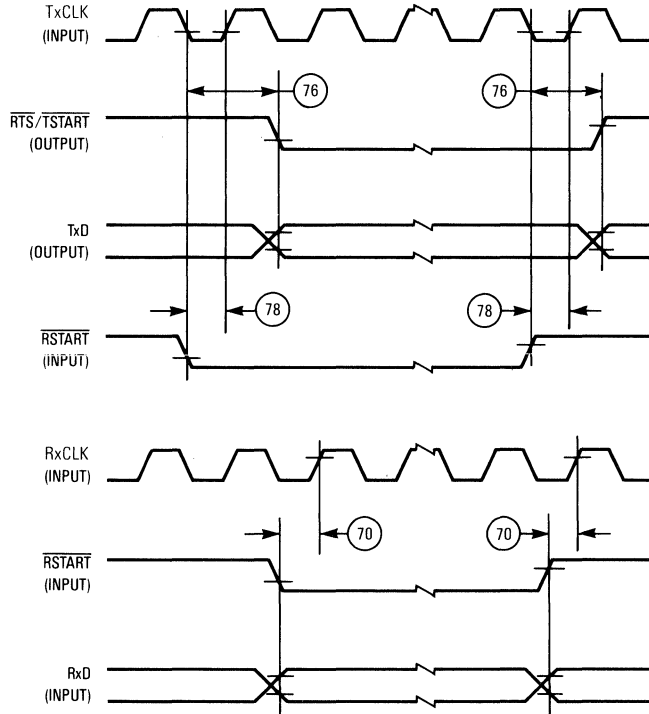


Figure 12-10. Motorola Read Cycle with RETRY Timing Diagram



NOTE: The MLAPD was designed to perform the full LAPD procedures with a serial clock to system clock ratio of 1:6. Operation at ratios less than 1:6 may result in performance and throughput degradation.

**Figure 12-11. Serial Data and Serial Clocks Timing Diagram**

## 12.5.2 Intel-Compatible AC Electrical Characteristics

The MLAPD is characterized at two rated frequencies, 12.5 MHz and 16.67 MHz. To use the MLAPD at system clock frequencies between 8 MHz and the rated clock frequency, the designer should use the electrical characteristics for the rated frequency of the MLAPD. The characteristics that are specified in nanoseconds are guaranteed for system clock frequencies between 8 MHz and the rated frequency. The characteristics that are specified as clock periods refer to the period of actual system clock.

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
1	Asynchronous Input Setup Time	10	—	10	—	ns
1A	Asynchronous Input Hold Time	15	—	15	—	ns
2	$\overline{RD}$ , $\overline{WR}$ Inactive to $\overline{CS}$ Inactive	0	80	0	60	ns
3	$\overline{RD}$ and $\overline{CS}$ or $\overline{INTA}$ to Data-Out Valid	—	80	—	60	ns
4	$\overline{RD}$ and $\overline{CS}$ or $\overline{INTA}$ High to Data-Out High-Impedance (see note 2)	—	40	—	30	ns
5	$\overline{RD}$ and $\overline{CS}$ or $\overline{INTA}$ High to Data-Out Invalid	0	—	0	—	ns
6	$\overline{CS}$ or $\overline{INTA}$ Low to READY LOW (Driving Three-State READY Low)	—	40	—	30	ns
7	CLK Low (on which $\overline{RD}$ or $\overline{WR}$ Low) to READY High (see note 4)	—	2+ 50	—	2+ 45	Clk. Per. ns
8	CLK Low to READY High	—	50	—	45	ns
9	Data-Out Valid to READY High	20	—	20	—	ns
10	READY High to $\overline{RD}$ or $\overline{WR}$ , $\overline{CS}$ or $\overline{INTA}$ High (earliest one)	0	—	0	—	ns
11	$\overline{RD}$ or $\overline{WR}$ , $\overline{CS}$ or $\overline{INTA}$ High to READY Low	—	40	—	30	ns
12	READY Low to READY High-Impedance (At the end of bus cycle after $\overline{CS}/\overline{INTA}$ High)	—	40	—	30	ns
13	$\overline{RD}$ or $\overline{WR}$ , $\overline{INTA}$ Inactive Time	1	—	1	—	Clk. Per.
14	$\overline{CS}$ , $\overline{INTA}$ Inactive Time	0	—	0	—	ns
15	$\overline{BHE}$ , A0-A2 to $\overline{RD}$ or $\overline{WR}$ , $\overline{CS}$ (latest one) Low (Write)	20	—	10	—	ns
16	READY High to Data-In Invalid	0	—	0	—	ns
17	$\overline{WR}$ , $\overline{CS}$ (latest one) Low to Data-In Valid	—	50	—	40	ns
18	Reserved	—	—	—	—	—
19	$\overline{RD}$ or $\overline{WR}$ High to $\overline{BHE}$ , A0-A2 Invalid	0	—	0	—	ns
20	CLK High to INTR High	—	80	—	60	ns
21	Reserved	—	—	—	—	—
22	Reserved	—	—	—	—	—
23	CLK High to HRQ Valid	—	40	—	30	ns
24	Reserved	—	—	—	—	—
25	Reserved	—	—	—	—	—
26	Reserved	—	—	—	—	—
27	Reserved	—	—	—	—	—

- Continued

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
28	Reserved	—	—	—	—	—
29	Reserved	—	—	—	—	—
30	Reserved	—	—	—	—	—
31	HRQ Negated to HOLDA Negated	0	—	0	—	ns
32	CLK Low on which HOLDA High to CLK High on which $\overline{AS}$ Low	1.5	1.5	1.5	1.5	Clk. Per.
33	Reserved	—	—	—	—	—
34	Reserved	—	—	—	—	—
35	Clock on which $\overline{AS}$ is High to Clock on which HRQ is Negated	—	0.5	—	0.5	Clk. Per.
36	Reserved	—	—	—	—	—
37	CLK High to Address High-Impedance	—	50	—	40	ns
38	CLK High to Address/ $\overline{BHE}$ Valid	—	80	—	60	ns
39	Reserved	—	—	—	—	—
40	CLK High to $\overline{AS}$ Low	—	40	—	30	ns
41	CLK to $\overline{AS}$ High (see note 3)	—	40	—	30	ns
42	Reserved	—	—	—	—	—
43	CLK High to $\overline{AS}$ , $\overline{RD}$ or $\overline{WR}$ High-Impedance	—	40	—	30	ns
44	CLK to $\overline{RD}$ or $\overline{WR}$ High (see note 3)	—	40	—	30	ns
45	Reserved	—	—	—	—	—
46	CLK Low to Data-In Invalid	15	—	15	—	ns
47	CLK Low to READY Low	15	—	15	—	ns
48	Data-In Valid to CLK (on which READY recognized) Low	—	10	—	5	ns
49	Reserved	—	—	—	—	—
50	CLK Low to $\overline{RD}$ or $\overline{WR}$ Low	—	40	—	30	ns
51	Reserved	—	—	—	—	—
52	CLK Low to Data-Out Valid	—	45	—	35	ns
53	Reserved	—	—	—	—	—
54	Reserved	—	—	—	—	—
55	CLK High to Data-Out Invalid	0	80	0	60	ns
56	No Exception to HRQ Asserted	1.5+ 10	2.5+ 50	1.5+ 10	2.5+ 40	Clk. Per. ns
57	Reserved	—	—	—	—	—
58	Exception Active to CLK Low	30	—	20	—	ns
59	Exception Active to CLK Low (Early Asynchronous Input Exception Required when READY is Absent)	10	—	10	—	ns
60	Reserved	—	—	—	—	—
61	CLK Low to Exception Inactive	15	—	15	—	ns
62	Reserved	—	—	—	—	—
63	RESET Width	10	—	10	—	Clk. Per.

- Continued

Number	Characteristic	12.5 MHz		16.67 MHz		Unit
		Min	Max	Min	Max	
64	CLK Frequency	8	12.5	8	16.66	MHz
65	CLK Period	80	125	60	125	ns
66	CLK Width High (see note 5)	35	62.5	25	62.5	ns
67	CLK Rise/Fall Time (see note 5)	—	5	—	5	ns
68	CLK Width Low (see note 5)	35	62.5	25	62.5	ns
69	RxCLK, TxCLK Frequency	0	12.5	0	16.67	MHz
70	RxD, $\overline{\text{RSTART}}$ Valid to RxCLK High Set-up Time	25	—	25	—	ns
71	RxCLK High to RxD, $\overline{\text{RSTART}}$ Hold Time	15	—	15	—	ns
72	RxCLK, TxCLK Rise/Fall Time	5	—	5	—	ns
73	RxCLK, TxCLK Width Low	35	—	25	—	ns
74	RxCLK, TxCLK Width High	35	—	25	—	ns
75	RxCLK, TxCLK Period	80	—	60	—	ns
76	TxCLK Low to TxD, $\overline{\text{RTS}}$ , $\overline{\text{TSTART}}$ Valid	10	40	10	40	ns
77	Reserved	—	—	—	—	—
78	$\overline{\text{CTS}}$ Low to TxCLK High Set-up Time	25	—	25	—	ns

NOTE

1. If  $\overline{\text{READY}}$  satisfies the asynchronous set-up time (1), then (48) is required for the data-in set-up time and (58) for the synchronous exception set-up time. Erroneous behavior may occur if (57) is not satisfied.
2. If  $\overline{\text{CS}}$  is negated before  $\overline{\text{RD}}$ , the data bus will be three-stated (4) possibly before  $\overline{\text{RD}}$  negation.
3.  $\overline{\text{RD}}/\overline{\text{WR}}$  and  $\overline{\text{AS}}$  rise on the end of a write cycle on the low phase of T4. When the MLAPD acquires the bus,  $\overline{\text{RD}}/\overline{\text{WR}}$  and  $\overline{\text{AS}}$  are three-stated until the rising edge before T1, and they rise on that high clock.
4. Data (3) and  $\overline{\text{READY}}$  (7) will be timed from the latest of  $\overline{\text{CS}}$  and either  $\overline{\text{RD}}/\overline{\text{WR}}$  during an MPU cycle. Data (3) and  $\overline{\text{READY}}$  (7) will be timed from the latest of  $\overline{\text{INTA}}$  and  $\overline{\text{RD}}$  during an INTA cycle.
5. If  $\overline{\text{CS}}$  or  $\overline{\text{IACK}}$  is negated before  $\overline{\text{UDS}}/\overline{\text{LDS}}$ , the data bus will be three-stated (4) possibly before  $\overline{\text{UDS}}/\overline{\text{LDS}}$  negation.

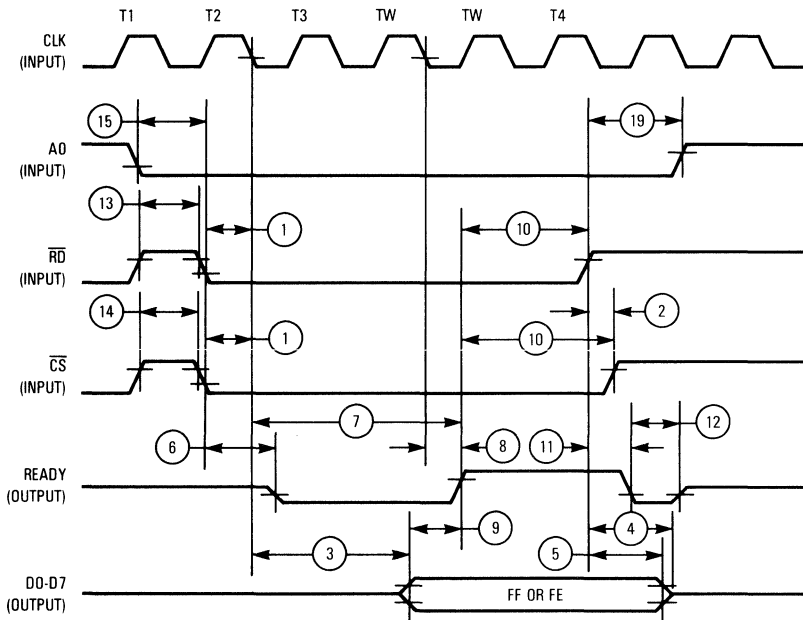


Figure 12-12. Intel-Compatible Host Processor Read Cycle Timing Diagram

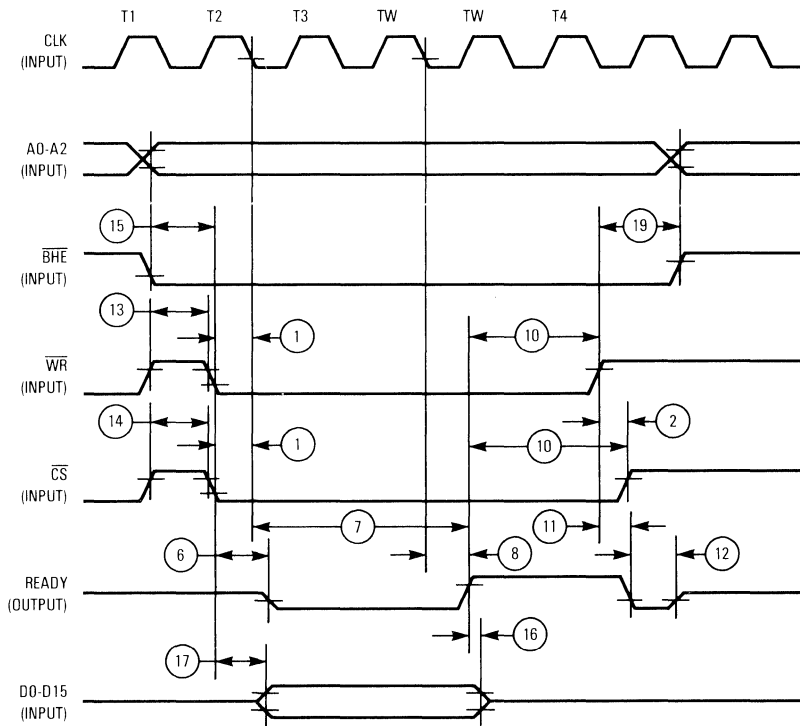
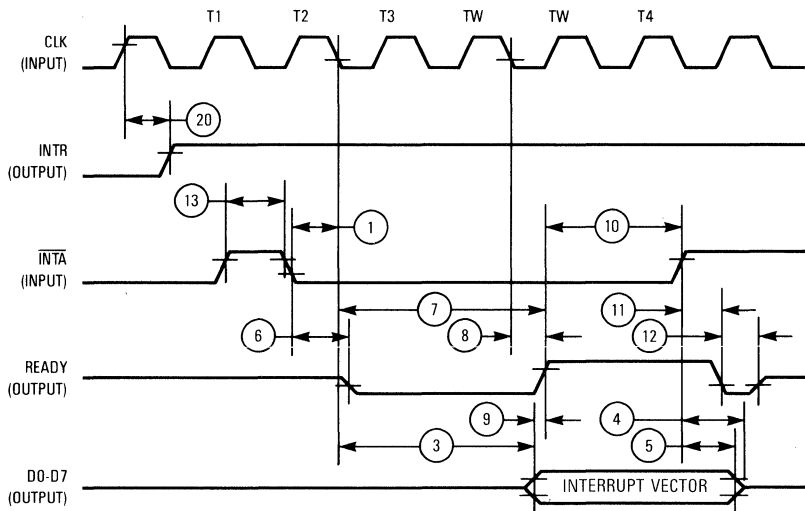
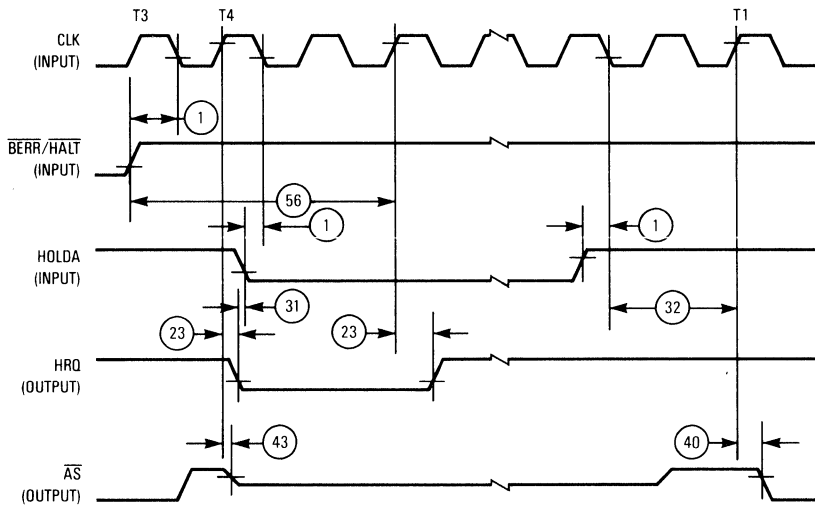


Figure 12-13. Intel-Compatible Host Processor Write Cycle Timing Diagram



**Figure 12-14. Intel-Compatible Interrupt Acknowledge Cycle Timing Diagram**





**Figure 12-15. Intel-Compatible Bus Arbitration Timing Diagram**

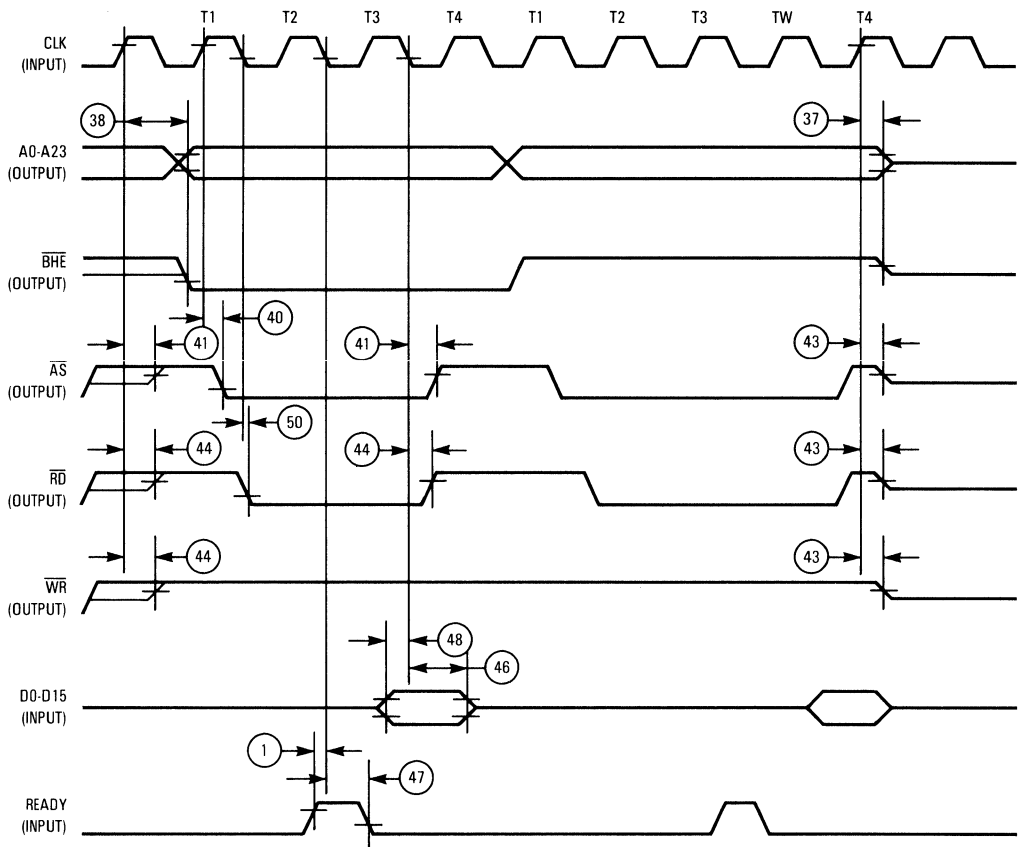


Figure 12-16. Intel-Compatible DMA Read Cycle and Slow Read Cycle Timing Diagram

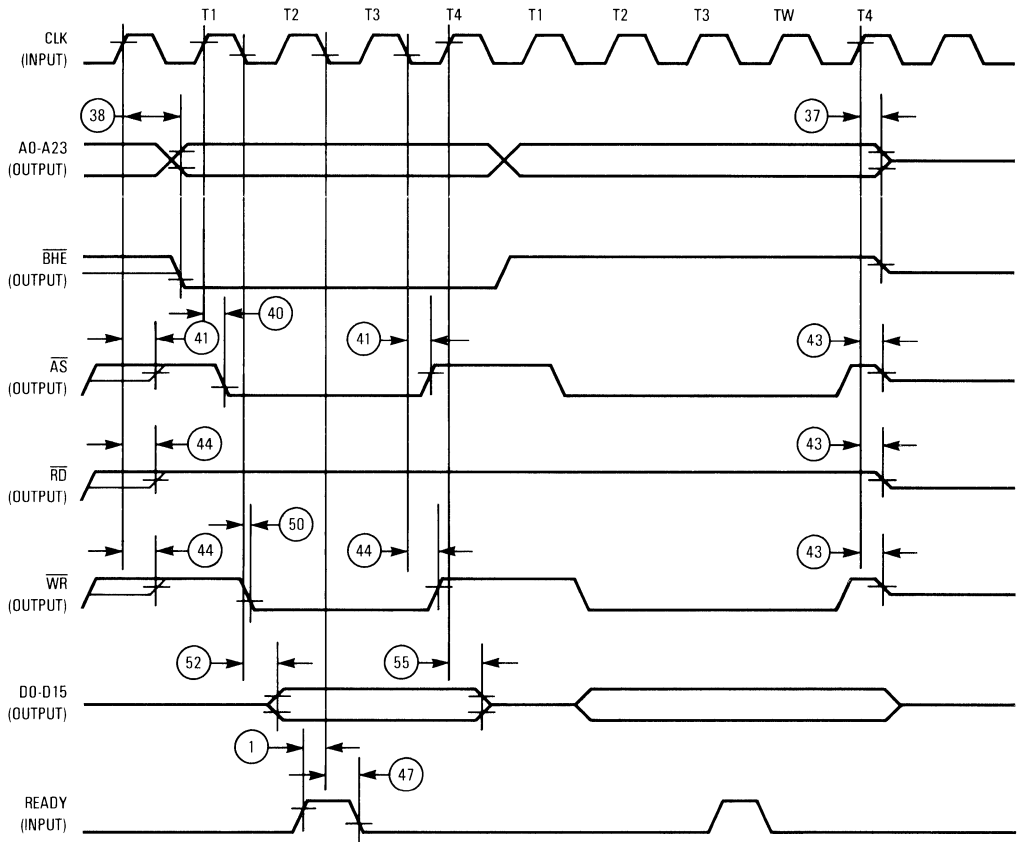


Figure 12-17. Intel-Compatible DMA Write Cycle and Slow Write Cycle Timing Diagram

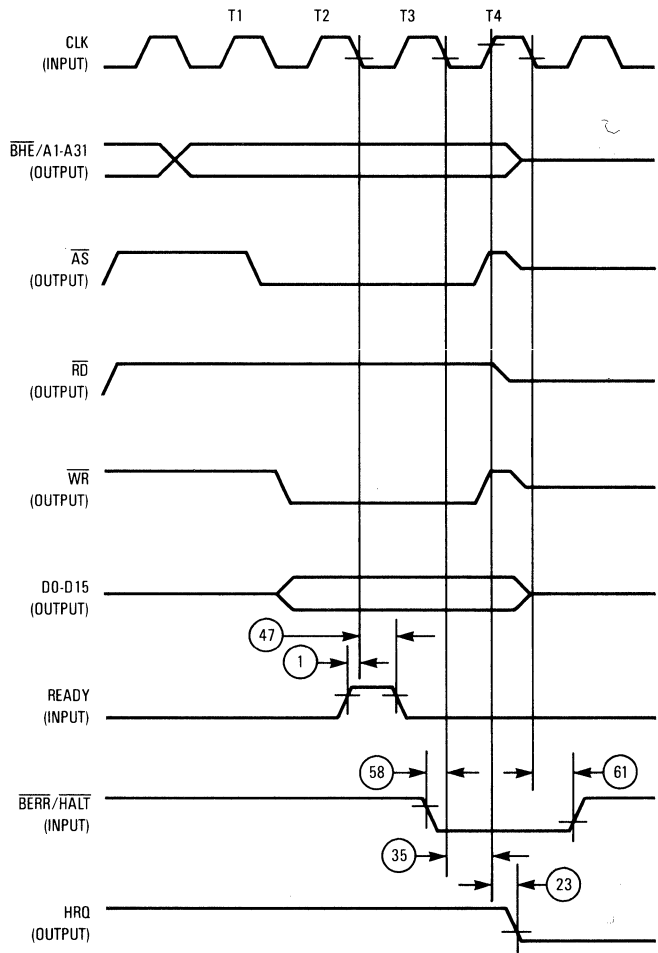
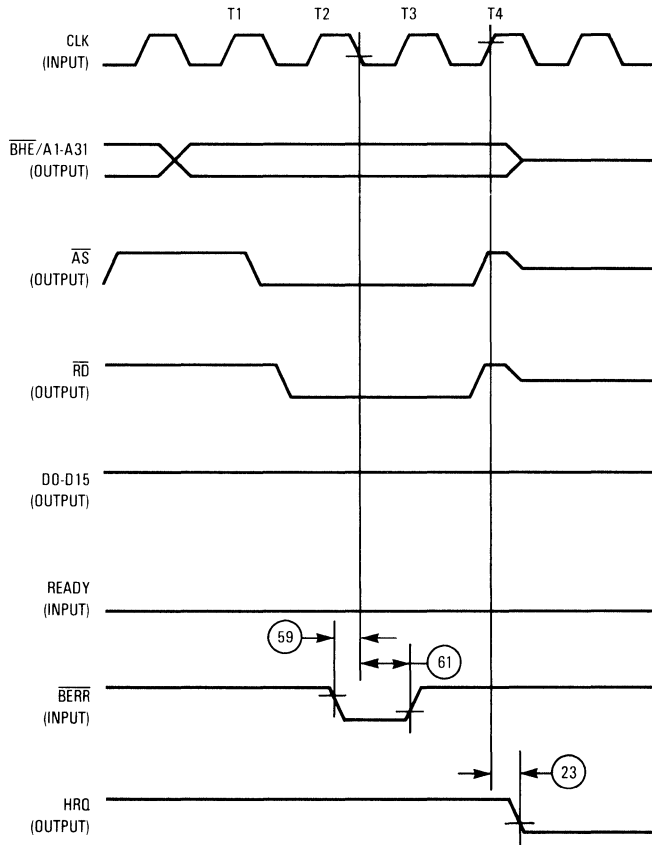


Figure 12-18. Intel-Compatible Late Synchronous Exception READY Active Timing Diagram



**Figure 12-19. Intel-Compatible Early Synchronous Exception READY Not Active Timing Diagram**

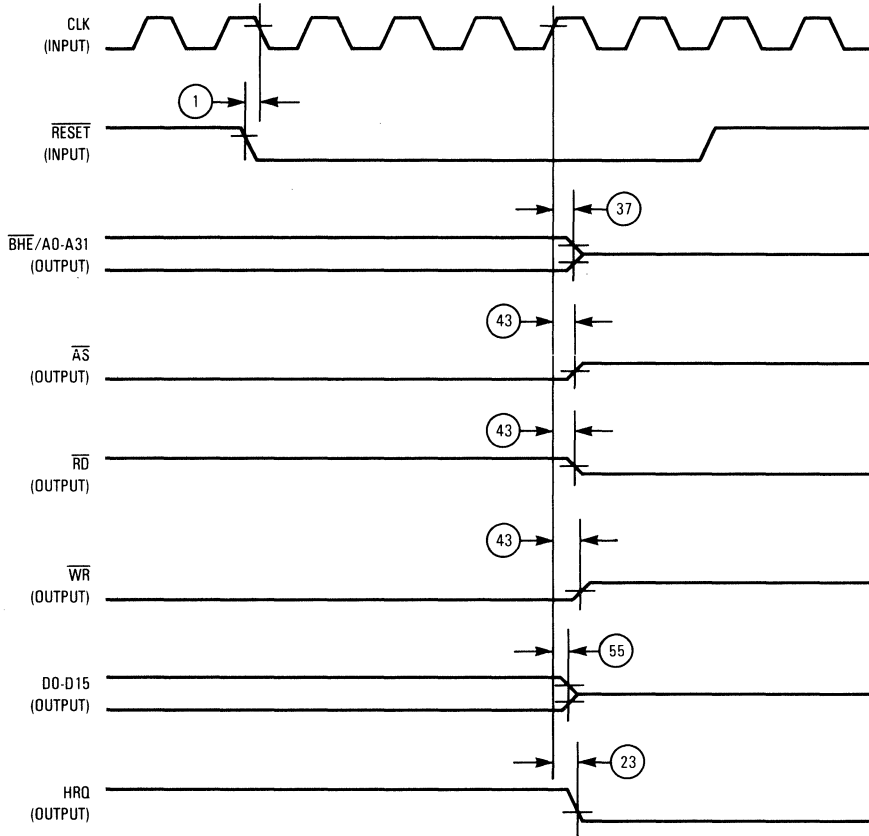


Figure 12-20. Intel-Compatible Hardware RESET Timing Diagram

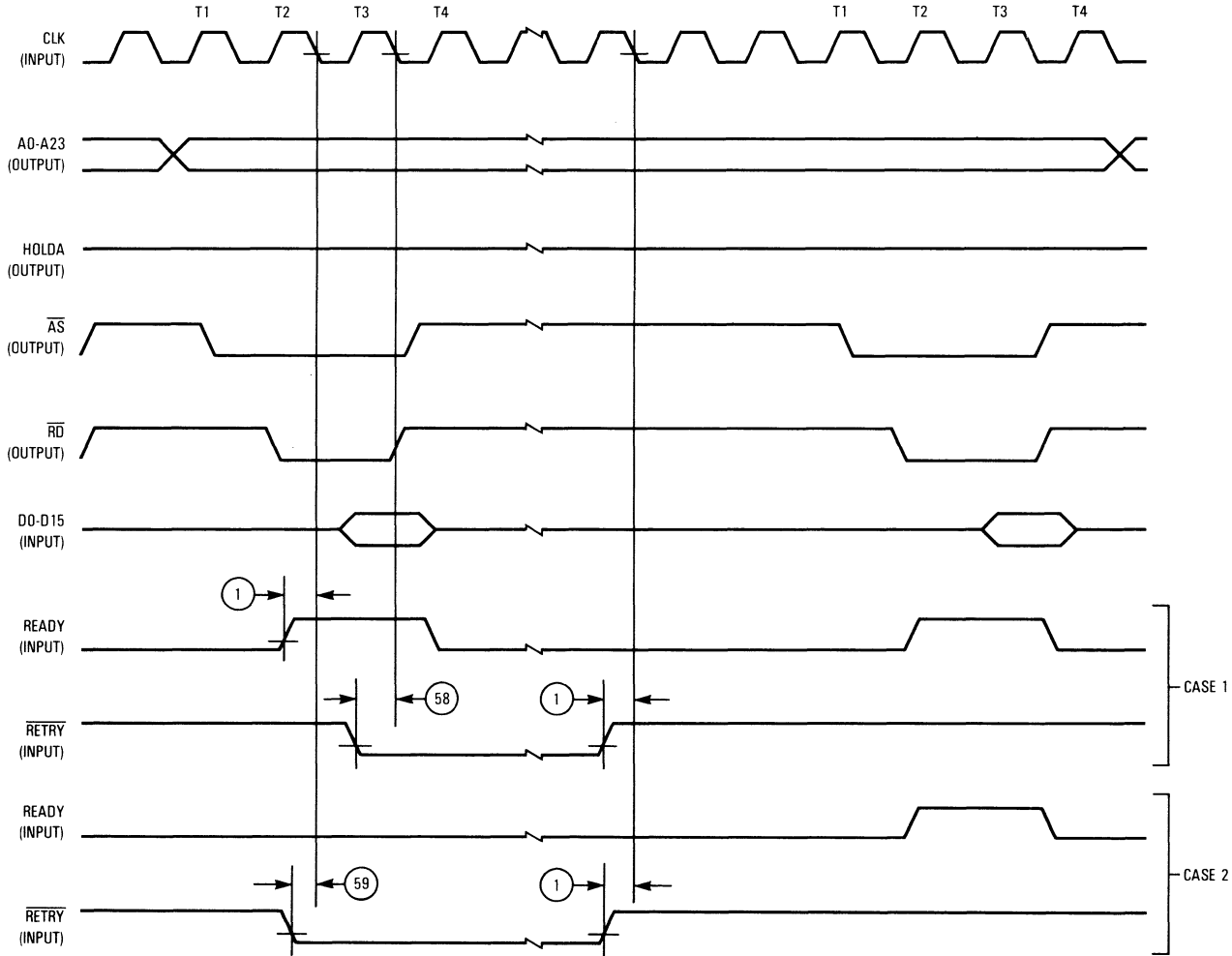
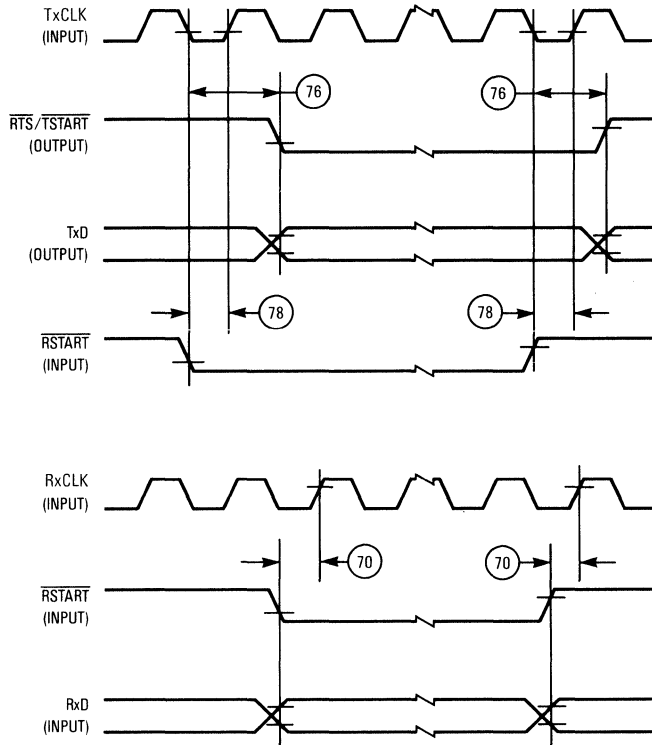


Figure 12-21. Intel-Compatible Read Cycle with  $\overline{RETRY}$  Timing Diagram



NOTE: The MLAPD was designed to perform the full LAPD procedures with a serial clock to system clock ratio of 1:6. Operation at ratios less than 1:6 may result in performance and throughput degradation.

Figure 12-22. Serial Data and Serial Clocks Timing Diagram



## SECTION 13 MECHANICAL DATA

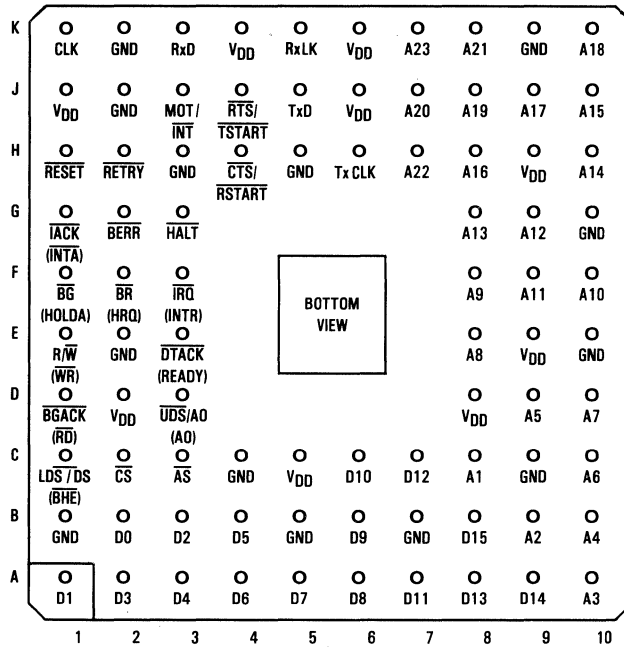
This section contains the pin assignments and package dimensions for the PGA (pin grid array) MC68606RC and for the PLCC (plastic leaded chip carrier) MC68606FN.

### 13.1 PACKAGE TYPES

Suffix	Package Type	Comments
RC	Pin Grid Array (PGA) Ceramic	Depopulated Center Pins Gold Lead Finish No Standoffs
FN	Plastic Leaded Chip Carrier (PLCC)	Suitable for Socketing or Surface Mounting

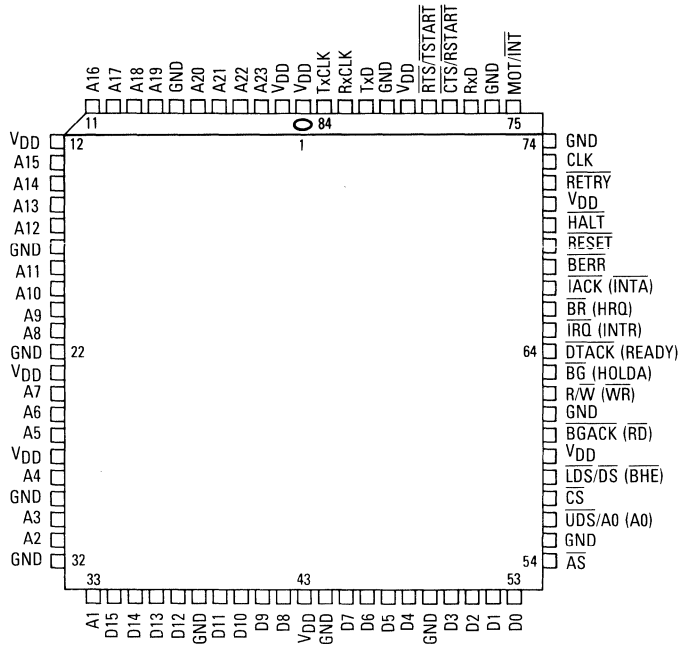
### 13.2 PIN ASSIGNMENTS

#### PIN GRID ARRAY



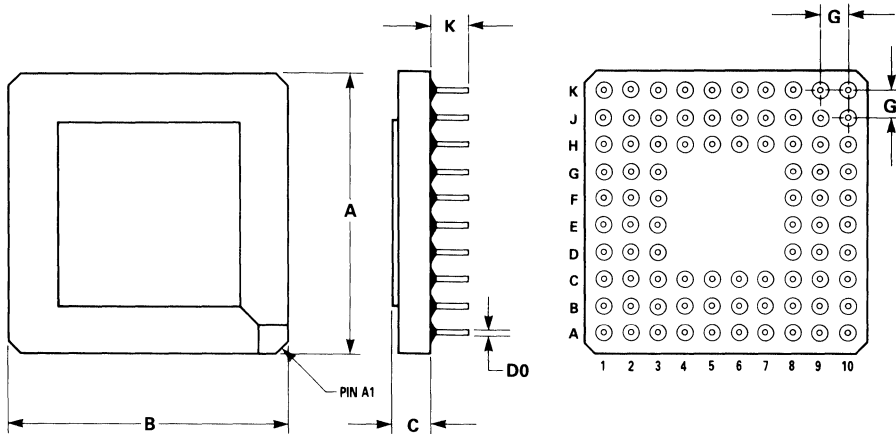
NOTE: Parenthetical signals apply to Intel assignment.

PLASTIC LEADED CHIP CARRIER



### 13.3 PACKAGE DIMENSIONS

RC SUFFIX  
PIN GRID ARRAY  
CASE 793-02

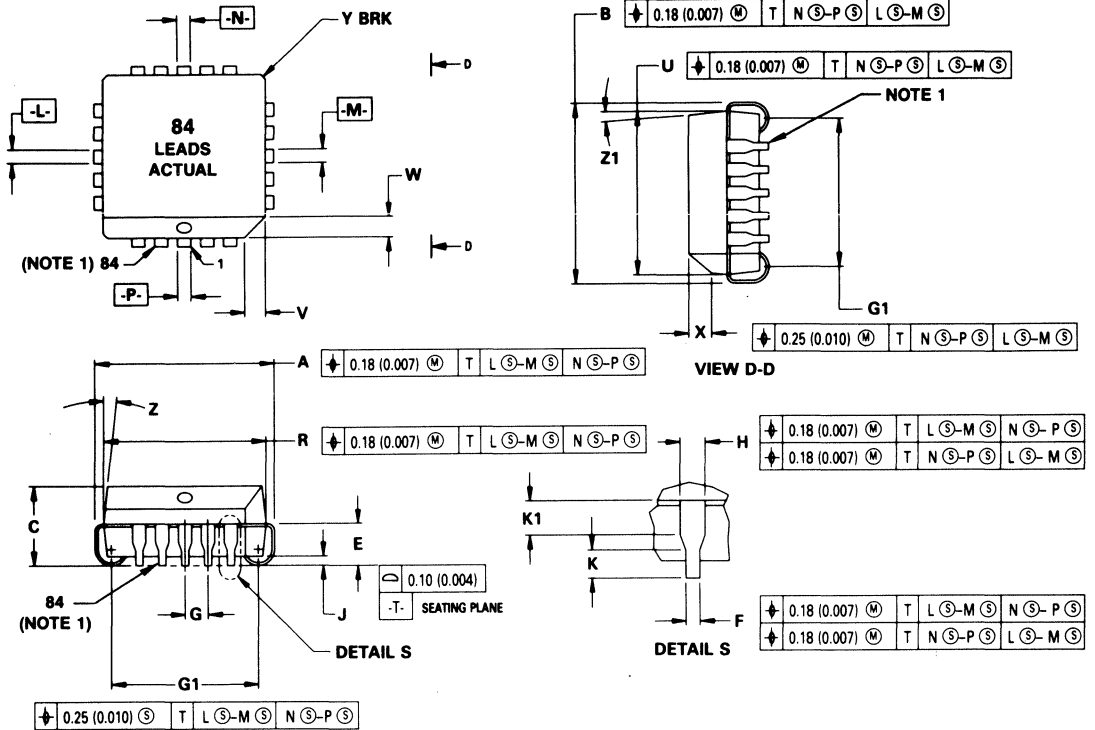


DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	—	27.43	—	1.080
B	—	27.43	—	1.080
C	2.03	2.67	0.080	0.105
D	0.43	0.61	0.017	0.024
G	2.54 BSC		0.100 BSC	
K	3.56	4.95	0.140	0.195

NOTES:

1. DIMENSIONS A AND B ARE DATUMS AND T IS A DATUM SURFACE.
2. POSITIONAL TOLERANCE FOR LEADS: (84 PL)  
 $\phi 0.13 (0.005) \text{ L T A } \textcircled{S} \text{ B } \textcircled{S}$
3. DIMENSIONING AND TOLERANCING PER Y14.5M, 1982.
4. CONTROLLING DIMENSION: INCH.

FN SUFFIX  
 PLASTIC LEADED  
 CHIP CARRIER  
 CASE 780-01



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	30.10	30.35	1.185	1.195
B	30.10	30.35	1.185	1.195
C	4.20	4.57	0.165	0.180
E	2.29	2.79	0.090	0.110
F	0.33	0.48	0.013	0.019
G	1.27 BSC		0.050 BSC	
H	0.66	0.81	0.026	0.032
J	0.51	—	0.020	—
K	0.64	—	0.025	—
R	29.21	29.36	1.150	1.156
U	29.21	29.36	1.150	1.156
V	1.07	1.21	0.042	0.048
W	1.07	1.21	0.042	0.048
X	1.07	1.42	0.042	0.056
Y	—	0.50	—	0.020
Z	2°	10°	2°	10°
G1	28.20	28.70	1.110	1.130
K1	1.02	—	0.040	—
Z1	2°	10°	2°	10°

NOTES:

1. DUE TO SPACE LIMITATION, CASE 780-01 SHALL BE REPRESENTED BY A GENERAL (SMALLER) CASE OUTLINE DRAWING RATHER THAN SHOWING ALL 84 LEADS.
2. DATUMS -L-, -M-, -N-, AND -P- DETERMINED WHERE TOP OF LEAD SHOULDER EXIT PLASTIC BODY AT MOLD PARTING LINE.
3. DIM G1, TRUE POSITION TO BE MEASURED AT DATUM -T-, SEATING PLANE.
4. DIM R AND U DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.25 (0.010) PER SIDE.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
6. CONTROLLING DIMENSION: INCH.

# APPENDIX A

## SOFTWARE INTERFACE FLOWS FOR THE MLAPD

### A.1 Nomenclature:

1. Comments are enclosed by /\* and \*/
2. Modules written as routines with associated arguments
3. Setting and zeroing bits affects only the bits concerned (logical AND, OR operations)
4. Pointer-> field is used to refer to the "field" belonging to a structure (C) or record (PASCAL) which is pointed to by "pointer".
5. Table( . n .) is used to signify the n-th element of array "table".
6. All bits in various fields of the FD's referenced in these "routine" are mentioned below. This is not a complete listing of bits that may appear in FD's.

### A.2 Frame Descriptor (FD) bits used in User-MLAPD Interface

#### A.2.1 Tx I FD-> status

ACK	The I frame has been acknowledged (confirmation for I-frame).
TX	The I frame has been transmitted.
EMPTY	No outstanding frames for Tx or Acknowledge

#### A.2.2 Tx U FD-> status

POS	The frame was successfully transmitted
NEG	The frame could not be transmitted.
EMPTY	No outstanding frames for Tx

#### A.2.3 Tx FD-> length (both U and I frames)

HV (header valid)	The current FD contains a valid header buffer
LAST	The current FD is the last in this transmit queue.

## A.2.4 Tx U FD->llid

**TX\_FRAME\_TYPE** These are three bits that may take on one of the values:  
DL\_UNIDATA, MDL\_UNIDATA, XID\_REQ, XID\_RSP  
NON\_STANDARD\_CMD(0 or 1) and NON\_STANDARD\_RSP(0 or 1)  
for non-protocol links,  
DISABLE\_CRC, DISABLE\_DLCl

## A.2.5 Rx FD-> control

**FIRST** This will be the first frame in the receive queue when filled with a received frame (i.e. the receive queue is now empty). MLAPD will interrupt when this is filled with a received frame.

**LASTP** Last frame in receive buffer pool.

**RED\_FLAG** The current FD has a red flag set. MLAPD will interrupt when this FD becomes the head of the receive pool (i.e. the FD preceding it has been used for frame reception).

## A.2.6 Rx FD-> llid

**RX\_FRAME\_TYPE** These are three bits that may take on one of the following values: UN-USED, UNDEFINED\_COMMAND, UNDEFINED\_RESPONSE, UI, I, FRMR, XID\_COMMAND, XID\_RESPONSE

## A.2.7 Rx FD-> error bits

**CRC** This frame has a CRC error.

**ABR** This frame ended in an abort sequence or not on an octet boundary.

**OVR** This frame caused a fifo overrun.

**BUF** This frame length exceeded N201 for its logical link.

## A.2.8 Interrupt queue entry bits

**LLID** The mask for reading the llid for the current entry

**VE (valid entry)** This entry defines a valid (and pending) interrupt condition.

**ARG** Argument bits (relevant for certain interrupts only).

**EVENT\_NUMBER** The number of the interrupt event (see interrupt mask).

**LV** The associated llid number is valid.

## A.3 Flows

### A.3.1 Issue command

```
/*
 * ROUTINE 1.
 * issue a command to MLAPD with
 * accompanied arguments
 * SR is the semaphore register
 * CR is the command register
 */

issue_command ( command , arg1 , arg2 , arg3 )

/*
 * wait till SR is FF
 */
while ( SR < > FF )
    IF looped more than MAXIMUM times
    THEN
        abort routine
        perform reset and initialization procedures

/* load arguments in GCB */
for each argument, argi
    GCB->argi = argi

write command to CR

/* if expected, wait for result */

END
```

### A.3.2 Build Receive Pool

```
/*
 * ROUTINE 13.
 * build a pool of receive buffers.
 */
BUILD_RX_POOL ( pool_number , pool_size )

    fd = allocate a frame
    set LASTP bit in FD-> control
    pool_descriptor( . pool_number .)->Next_Rx_FD_Pointer = FD
    pool_descriptor( . pool_number .)->User_Buffer_Tail_Pointer
        = FD

/*
 * instruct chip to receive frames to the new pool
 */
issue_command ( ASSIGN_POOL_POINTER , pool_number , fd )

ADD_TO_POOL ( pool_size frames )

END
```



### A.3.3 Build LL

```
/*
 * ROUTINE 2.
 * build a new logical link
 * we assume that the DLCI for the
 * new link has already been determined
 * and is passed here as an argument, as
 * well as the new LLID and the pool number
 * for the link
 */

BUILD LL ( LLID , DLCI , pool_index , queue_no )

    LLT = allocate memory for the LL configuration block
    fill in LLT (status, DLCI, pool number and protocol arguments)

    /*
     * set dlcI configuration bit
     */
    zero nonprotocol bit for protocol links
    set user/network bit as required

    /* build the receive pool if necessary */
    if( pool number pool_index does not exist )
        BUILD RX POOL( pool_index )

    /* insert pointer in LLID-LLT table */
    LLID_LLTT[ LLID ] = LLT

    /*
     * the broadcast LL is given a special command
     * to be activated while remaining in the
     * TEI-UNASSIGNED state
     */
    IF ( ( DLCI = BROADCAST_DLCI ) OR ( link is non-protocol ) )
    THEN
        issue_command ( ACTIVATE_LL , LLID )
    ELSE
        issue_command ( MDL_ASSIGN_REQUEST , LLID )

END
```

### A.3.4 Initialize MLAPD

```
/*
 * ROUTINE 3.
 * initialization sequence
 */

INITIALIZE MLAPD

/* global configuration block */
gcb = allocate memory for GCB
initialize fields of GCB

/* match table */
IF high_end operation mode is desired
THEN
    match = allocate memory for match table (8k words)
    initialize match table (all entries marked invalid)

/* layer 2 queue */
l2 = allocate memory for layer 2 queue ( layer_2_length )
zero layer 2 queue memory
set pointer to l2 in gcb

/*
 * timers table: the user is responsible for
 * providing enough memory for the timers table.
 * The MLAPD requires one word for every value of
 * llid between 0 and MAXIMUM_LLID plus two additional
 * words for housekeeping.
 */
timers = allocate memory for timers
zero timers memory
set pointer to timers in gcb

/* LLID-LLT table */
LLID_LL_T = allocate memory for LLID to LLT table
set pointer in gcb

/* interrupt queue */
iq = allocate memory for interrupt queue (interrupt_queue_length)
zero interrupt queue memory
set pointer in gcb

/*
 * the following sequence instructs the
 * MLAPD to begin its normal functions
 */
issue_command ( RESET )
write interrupt vector number to IVR /* not a MLAPD command */
issue_command ( SET_BUS_WIDTH_16,(8) ) /* depend on system bus*/
/*
 * writing the GCBPR is not a command to the MLAPD.
 * it should immediately precede the INIT command
 */
write gcb in GCBPR
issue_command ( INIT )

/* broadcast LL */
```

BUILD LL ( BROADCAST\_LLID , BROADCAST\_DLCI )

END

### A.3.5 Handle\_Interrupts

```
/*
 * there are 2 interrupt routines, corresponding
 * to the 2 generic classes of interrupts issued
 * by MLAPD: normal (intr0) and severe (intr1)
 */

/*
 * ROUTINE 4a.
 * interrupt routine: severe interrupt
 */
INTR1 ( )

/*
 * severe interrupt routine
 * note that the MLAPD will not accept any instruction
 * now other than OFFLINE.
 * this is only a guideline for possible handling
 * of severe interrupts
 */

issue_command ( OFFLINE )
issue_command ( DUMP )
test memory

To continue working the user must reinitialize the MLAPD
END
```

```

/*
 * ROUTINE 4b.
 * interrupt: normal interrupt
 * this routine may be reached following an interrupt
 * or called directly by the polling task
 * each element of the interrupt queue is referenced as a long
 * word (32 bit).
 */
INTRO ( )

    WHILE ( *next_intr & VE is set )
    DO

        /*
         * if the LV bit is raised, the lld associated
         * with the interrupt is in (*next_intr & LLID)
         * if the AV bit is raised, an argument associated
         * with the interrupt is in (*next_intr & ARG)
         */

        IF ( *next_intr & AV )
        THEN
            /*
             * MDL_error_indications and link_counter thresholds
             * both have an associated argument field
             */

            IF ( *next_intr & EVENT_NUMBER = MDL_error_indication )
            THEN
                *next_intr & ARG contains encoded explanation of error

            IF ( *next_intr & EVENT_NUMBER = link_counter )
            THEN
                *next_intr & ARG contains coded name of counter

            IF ( *next_intr & EVENT_NUMBER = bus_error )
            THEN
                handle as in severe interrupt routine

        /*
         * note that if the interrupt was INTERRUPT_QUEUE_OVERFLOW,
         * at least one interrupt has been lost
         */

        handle interrupt as required

        zero VE bit in *next_intr

        advance next_intr (MODULO interrupt_queue_length)

    END OF WHILE

    /* enable further interrupts if necessary */
    IF NOT polling mode
    THEN
        issue_command ( ENABLE_IRQ )
END

```

### A.3.6 Add Tx I frames

```
/*
 * ROUTINE 5.
 * add linked list of Tx I FD_a -> ... -> FD_n to Tx queue in LLT
 * (each FD's Last bit is zero)
 *
 * It is assumed that this routine is reached when frames are to be
 * added for transmission either before or after all outstanding acknow-
 * nments have arrived, and possibly following a STOP_TRANSMIT command.
 * All frames have been prepared for transmission: header buffer (if
 * exists), length, etc.
 *
 * Note that the tx queue of each LL may be marked empty (in
 * transmit and confirm routines). This may be done by zeroing either
 * the user_tx_last_queued_ptr or the user_tx_next_confirm_ptr.
 * or by raising and lowering bits in a flag which the user
 * creates for each logical link.
 */
ADD I FRAME(S) FOR TX ( LLT, FD_a, FD_n )

    set Last bit in FD_n->length
    /*
     * add to transmit queue
     * by updating user entries in LLT
     */
    IF tx queue is marked empty /* a user flag for this LLT */
    THEN                          /* Tx queue is empty */

        User_Tx_Last_Queued_ptr = FD_n
        User_Tx_Next_Confirm_ptr = FD_a

    ELSE                          /* Tx queue not empty */
        /* add to tail of queue */
        User_Tx_Last_Queued_ptr->Next_I_Frame_Descriptor_Pointer = FD_a
        zero Last bit in User_Tx_Last_Queued_ptr->length
        temp = User_Tx_Last_Queued_ptr
        User_Tx_Last_Queued_ptr = FD_n

    /*
     * if the queue is not stopped,
     * see if a RELINK command is needed.
     */
    IF ( LL is not stopped )
    THEN
        IF ( Transmit bit in temp->status is set )
        THEN
            issue_command ( RELINK , LLID )
            return

    /*
     * if tx queue was either stopped or empty
     * we must issue a DATA_REQUEST command
     * NOTE: the user should only raise these flags
     * after receiving a DL_data_confirm interrupt
     * for this link.
     */
    IF ( LL is marked as stopped or empty )
    THEN                          /* give command, setting chip's tx pointer */
```

```
mark LLT tx queue as not empty and not stopped
set fd to first non-transmitted frame
issue_command ( DL_DATA_REQUEST , LLT, fd )
END
```

### A.3.7 Add XID/UI Tx frames

```
/*
 * ROUTINE 6.
 * add linked list of Tx XID/UI FD_a -> ... -> FD_n
 * to any of the three XID/UI Tx queues
 * Note that the command given must be chosen with
 * respect to which unnumbered queue is being used.
 */
ADD FRAME(S) FOR XID/UI TX ( FD_a, FD_n )

    FOR EACH frame to be transmitted
        set Last bit in FD->length to zero
        set FD->llid to required llid
        set FD->llid & TX_FRAME_TYPE to desired type of frame
            /* UI, XID_REQUEST, XID_RESPONSE */
            /* or NON_STANDARD_CMD/RSP */

        set Last bit in FD_n->length

        /* add to transmit queue */

        IF ( XID/UI_tx queue is empty )
            THEN /* queue is empty */

                User_U_Tx_Last_Queued_ptr = FD_n
                User_U_Tx_Next_Confirm_ptr = FD_a
            ELSE /* queue not empty */

                /* add to tail of queue */
                User_U_Tx_Last_Queued_ptr->Next_U_Frame_Descriptor_Pointer
                    = FD_a
                temp = User_U_Tx_Last_Queued_ptr
                User_U_Tx_Last_Queued_ptr = FD_n
                zero Last bit in temp->length

        /*
        * a command must be issued if the XID/UI tx queue
        * is either empty or stopped
        */
        IF ( XID/UI_tx is empty or stopped )
            THEN
                fd = first non-transmitted FD
                mark UI_tx queue as not empty and not stopped
                issue_command ( XID/UI_REQUEST , fd )
    END
```



### A.3.8 Interrupt DL\_Data\_Confirmation

```
/*
 * ROUTINE 7.
 * interrupt routine performed on
 * "all-I-frames-confirmed interrupt".
 * check that all frames added to the Tx queue
 * have been handled by the MLAPD
 * the check here should be redundant if done in
 * "collect_next_confirmed_frame" and is included
 * here only for completeness
 */
INTERRUPT DL_Data_Confirmation ( LLT )
  IF ( LL is stopped )
    THEN
      return

  fd = User_Tx_Next_confirm_ptr

  WHILE ( LAST bit in fd->length = 0 )

    IF ( ACK bit in fd->status = 1 AND
        EMPTY bit in fd->status = 1 )
    THEN      /* chip "missed" the last frames added */
      issue_command ( DL_DATA_REQUEST , LLT , fd->next )
      return

    fd = fd->next

/*
 * presumably some indication will be made here to
 * tell the software to perform GET NEXT CONFIRMED FRAMES
 * in the near future
 */
END
```

### A.3.9 Collect Next Confirmed Frame on LLT

```
/*
 * ROUTINE 8.
 * get next confirmed frame on LLT
 * for further layer 3 processing
 */

GET NEXT CONFIRMED FRAME ( llT )

IF ( ACK bit is set in User_Tx_Next_Confirm_ptr->status )
THEN
    temp = User_Tx_Next_Confirm_ptr    /* save current position */

    IF ( Last bit is not set in User_Tx_Next_Confirm_ptr->length )
    THEN                                /* more frames in queue */
        temp = User_Tx_Next_Confirm_ptr
        IF Empty bit in temp->status is set
        THEN
            issue_command ( DL_DATA_REQUEST , llT , temp->next )

            User_Tx_Next_Confirm_ptr =
                User_Tx_Next_Confirm_ptr->
                    Next_I_Frame_Descriptor_Pointer

    ELSE
        mark tx queue as empty    /* checked in ADD I FRAMES
        FOR TX */

        return ( temp ) /* to layer 3 */

    ELSE
        return ( NULL )

END
```

### A.3.10 Interrupt XID/UI Confirmation

```
/*
 * ROUTINE 9.
 * interrupt routine performed on
 * XID/UI_Confirmation interrupt".
 * check on last frame added to the XID/UI Tx queue
 * that the whole queue has been handled
 * if the check is done in "collect_next_tx_xid_ui_frame"
 * it is redundant here. it is included only for completeness.
 */
INTERRUPT XID/UI_Confirmation

/*
 * note that this interrupt can only be received
 * when the XID/UI tx queue is not stopped
 */

fd = User_U_Tx_Last_Queued_ptr

WHILE ( LAST bit in fd->length = 0 )

    IF ( EMPTY bit in fd->status = 1 )

        THEN /* chip "missed" frames following this one */
            issue_command ( XID/UI_REQUEST , fd->next )
            return

        fd = fd->next

/*
 * presumably some indication will be made here to
 * tell the software to perform COLLECT NEXT TRANSMITTED
 * XID/UI FRAMES
 * in the near future
 */
END
```

### A.3.11 Collect Next Tx XID/UI Frame

```
/*
 * ROUTINE 10.
 * collect confirmed XID/UI frames on LLT
 * for further layer 3/management processing
 */

COLLECT NEXT TRANSMITTED XID/UI FRAME

IF ( (User_U_Tx_Next_ptr->status & POS) is non-zero
    OR (User_U_Tx_Next_ptr->status & NEG) is non-zero )
THEN
    temp = User_U_Tx_Next_ptr      /* save current position */

    IF Last bit is not set in User_U_Tx_Next_ptr->length
    THEN      /* more frames in queue */
        If EMPTY bit in temp->status is set
        THEN
            issue_command( XID/UI_REQUEST , temp->next )

            User_U_Tx_Next_ptr =
                User_U_Tx_Next_PTR->Next_U_Frame_Descriptor_Pointer
            return ( temp )      /* to layer 3 */

        ELSE
            mark XID/UI tx queue empty /* checked when transmittin*/
            return ( temp )

    ELSE
        return ( NULL )      /* no confirmed frames */

END
```

### A.3.12 Collect Next Frame Received on pool

```
/*
 * ROUTINE 11.
 * get next frame received to the desired receive pool
 */

GET NEXT RX FRAME ( pool )

/*
 * don't take the last frame from pool
 * RX_FRAME_TYPE non-zero means a frame has been received
 * with that frame descriptor
 */
IF Last_In_Pool bit in User_Rx_Next_ptr->control is zero
 AND RX_FRAME_TYPE in USER_Rx_Next_ptr->llid is non-zero
 THEN
   temp = User_Rx_Next_Ptr
   User_Rx_Next_Ptr = User_Rx_Next_Ptr->
                       Next_Rx_Frame_Descriptor_Pointer
/*
 * note that temp->llid contains the llid for which the
 * frame was received and bits which indicate that the frame
 * is either I, UI or XID (REQ or RSP).
 */
   return ( temp )
ELSE /* Rx queue is empty */
   return( null )
END
```

### A.3.13 Add to Receive Pool

```
/*
 * ROUTINE 12.
 * add the given frame to the receive buffer pool.
 * The argument "pool" points to the pool descriptor
 * for the pool concerned.
 */

ADD TO POOL ( frame, pool )

    temp = pool->User_Buffer_Pool_Tail_Pointer

    /* add frame to end of pool */
    temp->Next_Frame_Descriptor_Pointer = frame

    /* frame is now the last in the pool */
    raise LASTP bit in frame->control

    /* update tail pointer and previous last bit */
    pool->User_Buffer_Pool_Tail_Pointer = frame
    zero LASTP bit in temp->control

END
```

### A.3.14 Stop Transmit of I Frames

```
/*
 * ROUTINE 14.
 * stop transmission on a logical link
 * as soon as possible
 */

STOP TX ( LLT )

/*****
version 1:
The STOP_TX command is issued. This means that if the MLAPD
is currently transmitting a frame from this link, it will stop
the transmission of the frame and generate an ABORT sequence.
*****/

issue_command ( STOP_Transmit , LLID )

/*
 * only when all acknowledgements have arrived can the
 * queue be manipulated
 * NOTE: the routine add_frames_for_tx should not be
 * called until all I-frames have been confirmed.
 */
wait for interrupt on I-frames confirmed

mark LL as stopped /* checked in add_frames_for_tx */

/*
 * MLAPD will now cease accessing the queue,
 * leaving the user free to manipulate the
 * transmission queue of "LLT" at will.
...
 * eventually, the user may prepare FD_a-> .. ->FD_n
 * for transmission
 */
ADD I FRAME(S) FOR TX ( FD_a , FD_n , LLID )

/*****
version 2:
The STOP_TX command is not issued.
The transmit queue is effectively shortened by raising the
LAST bit in the first two non-transmitted FD
*****/

/*
 * Find first non-transmitted frame
 */
fd = first fd in ll tx queue
while TX bit in fd->status is set
  fd = fd->next

/*
 * raise LAST bit in the two FD's at the end of the tx queue
 * this must be done in a loop to avoid race conditions
 * that can occur if the MLAPD is currently transmitting
 * these frames.
 * (the bit is set before the condition is checked)

```

```
*/
set LAST bit in fd->next->length
do
  fd = fd->next
  set LAST bit in fd->next->length
while TX bit in fd->status is set

END
```



### A.3.15 Stop Transmit for XID/UI Frames

```
/*
 * ROUTINE 15.
 * stop transmission of XID/UI frames
 * as soon as possible
 */
STOP XID/UI TRANSMIT

    mark XID/UI_tx as stopped /* checked in confirm interrupt */
    issue_command ( STOP_XID/UI_TRANSMIT )

/*
 * only when the chip has handled the STOP XID/UI command
 * may the queue be manipulated. This is true when the
 * semaphore register returns to the value of FF.
 */
wait until semaphore register = FF

/*
 * MLAPD will now cease accessing the queue.
 * leaving the user free to manipulate the XID/UI
 * transmission queue at will.
 *
 * ...
 * eventually, the user may prepare FD_a-> .. ->FD_n
 * for transmission
 */
ADD FRAME(S) FOR XID/UI_TRANSMIT ( FD_a , FD_n )
```

### A.3.16 DMA test

```
/*  
 * ROUTINE 16  
 *  
 * DMA_TEST  
 * three arguments are passed to this routine:  
 * ptr1, ptr2 and length  
 */  
  
DMA_TEST ( ptr1, ptr2, length )  
  
perform INIT sequence  
issue_command ( OFFLINE )  
clear length bytes of memory at ptr2  
  
issue_command ( DMA_TEST, length, ptr1, ptr2 )  
wait tiii semaphore register = FF  
  
check that the memory at ptr2 is identical to ptr1  
  
END
```

### A.3.17 Serial loopback test

```
/*
 * ROUTINE 17
 * perform a serial loopback test
 * three scenarios are presented:
 * 1. the chip is in normal mode and non-protocol links are used
 * 2. the chip is in normal mode and the FLIP bit is used
 * 3. the chip is in promiscuous_rx mode
 */

SERIAL_LOOPBACK_TEST ( test_fds )

.....
scenario 1: chip is in normal mode, test_link is non-protocol
.....

/*
 * in performing the INIT sequence for a serial loopback test,
 * the WRAP bit in gcb->option bits should be set and the
 * non_protocol_error_mask set to receive all frames
 * NOTE: The data buffer size of frames in the receive pool
 * should be two bytes longer than the user data presented.
 * This is to compensate for the dlci which may be added
 * in transmission (optional). Also note that in this
 * scenario, the user data should include the 16-bit CRC.
 */
perform INIT sequence with test_option_bits
prepare a receive pool test_link_pool_no

/*
 * the error mask is valid for this link so that we may check
 * for CRC or ABORT errors in receiving the data
 */
set NON_PROTOCOL bit for test_link
set ERROR_MASK_VALID bit for test_link
build II ( test_link , test_link_pool_no )

lower DISABLE_DLCL bit for all test_fds
lower DISABLE_CRC bit for all test_fds

/*
 * the following routine issues the DL_DATA_REQUEST command
 */
add frame(s) for tx ( test_link , fd )
wait until frame(s) in receive pool are not-empty

/*
 * the dlci has been received by the chip in the first 2
 * bytes of the receive buffer: it should be ignored.
 * the last two bytes of data are the CRC
 */
check for ABORT or CRC error in received frame(s)
check received frames match transmitted frames
```

```
.....
scenario 2: chip is in normal mode, FLIP bit is used
.....
```

```
/*
 * for this scenario, both the FLIP bit and
 * the WRAP bit in gcb->option bits should be set.
 */
perform INIT sequence with test_option_bits
prepare receive pool(s)
/*
 * prepare two dlcis that differ only on the "flip" bit
 * (e.g. 100 and 101)
 */
dlci2 = dlci1 with FLIP bit raised
zero user/network bit in dlci1
set user/network bit in dlci2
set nonprotocol bit for test_link1, test_link2

build ll ( test_link1 , dlci1 , test_link_pool_no1 )
build ll ( test_link2 , dlci2 , test_link_pool_no2 )

/*
 * the following routine issues the DL_DATA_REQUEST command
 */
add frame(s) for tx ( test_link1 , fd )
wait frames in receive pool are not-empty

/*
 * supervisory and data frames should be exchanged by
 * the two links as dictated by the protocol.
 */
check that the data frames received on test_link2
      match those transmitted on test_link1
```

```
.....
scenario 3: chip is in promiscuous_rx mode
.....
```

```
/*
 * in performing the INIT sequence for a serial loopback test,
 * the WRAP bit in gcb->option bits should be set and the
 * non_protocol_error_mask set to receive all frames
 * in this scenario, the PROMISCUOUS mode is chosen
 * NOTE: The data buffer size of frames in the receive pool
 * should be four bytes longer than the user data presented.
 * This is to compensate for the dlci and CRC which may be
 * added in transmission (optional). These fields are
 * received and stored in the data buffer when the MLAPD is
 * in promiscuous mode.
 */
perform INIT sequence with test_option_bits
prepare a receive pool test_link_pool_no

/*
 * create a link for transmitting the frames
 */
set NON_PROTOCOL bit for tx_link
build ll ( tx_link )
```

**A**

```

/*
 * the error mask is valid for the rx_link so that we may check
 * for CRC or ABORT errors in receiving the data.
 * note that in promiscuous mode, all frames are received in
 * the pool for which link dlcI = 0 is associated.
 */
set ERROR_MASK_VALID bit for rx_link
set NON_PROTOCOL bit for rx_link
build II ( rx_link , dlcI = 0, test_pool_no )

raise DISABLE_DLCI bit for all test_fds
lower DISABLE_CRC bit for all test_fds

/*
 * the following routine issues the DL_DATA_REQUEST command
 */
add frame(s) for tx ( tx_link , fd )
wait until frame(s) in receive pool are not-empty

/*
 * the last two bytes of data are the CRC
 */
check for ABORT or CRC error in received frame(s)
check received frames match transmitted frames

```

END



### A.3.18 Serial loopback test

```
/*
 * ROUTINE 18.
 *
 * PERFORM LAYER 2 PROCESSING (MEMORY TO MEMORY)
 *
 * This routine indicates how an application may use the MLAPD
 * to perform layer 2 (LAPD) protocol processing where the
 * serial interface is not non-channelized wide band.
 * The interface to the physical level is achieved via another
 * device, referred to as device A.
 * The MLAPD communicates with device A via shared memory
 * (which may necessitate some data structure translation by
 * a processor).
 */

LAYER_2_PROCESSING

/*
 * initialize the MLAPD. Note that the INTERNAL_LOOPBACK
 * option bit is NOT set
 */
perform INIT sequence with MEMORY_TO_MEMORY bit set

/*
 * create a link for dlci = 0 (on I_QUEUE_0) for handling
 * frames received by device A.
 * this link must be put in the NON_PROTOCOL mode
 * and is used to interface the physical layer device (A).
 */
set NON_PROTOCOL bit for physical_llid
build ll ( physical_llid, dlci = 0, physical_pool_no, queue_no = 0 )

/*
 * build logical links required by the application
 * (say appl_llid). None of these links may be assigned
 * to I_QUEUE_0.
 */
build ll (appl_llid, appl_dlci, appl_pool_no, appl_queue_no )

/*
 * to receive a frame from device A, the following
 * steps are taken
 */

gather frames from device_A receive queue

/*
 * the following may involve translating device A's receive
 * data structures to the MLAPD's transmit data structures
 */
link frames for transmission to tx queue of physical_llid

/*
 * CRC may be generated by the MLAPD if stripped off by
 * device A, or not, if the received CRC remains in memory.
 * assume the latter is the case.
 */
```

```

raise DISABLE_CRC bit for all physical_fds
add frames for tx ( llid = 0, physical_fds)

on receiving DL_DATA_CONFIRMATION interrupt for dlc_i = 0:
/*
 * all frames have been transmitted. Free them, or return them
 * to a device A receive pool, as necessary.
 */
release transmitted frames

/*
 * the frames received from device A are transmitted
 * by the non-protocol link which serves as an HDLC framer.
 * Layer 2 address and control fields are already in the
 * transmitted data. The frames are internally looped to the
 * receiver and received for whichever link they were destined,
 * (generating a DL_DATA_INDICATION interrupt if appropriate).
 * Protocol processing, and user interface from this point onwards
 * is identical to that of normal MLAPD functions.
 *
 *
 * In order to transmit frames on the physical layer (on appl_llid)
 * the following steps should be taken.
 * (U or I-frames may be transmitted)
 */
add frames for tx ( appl_llid, tx_fds)

/*
 * these transmitted frames are looped internally, and received
 * on physical_llid (dlci = 0).
 * the MLAPD will also automatically transmit any Supervisor frames
 * in accordance with the protocol.
 */
FOR each frame received on physical_pool_no

    collect next frame received on physical_pool_no

    /*
     * the following step translates the MLAPD receive-data structure
     * to device A's transmit data structure.
     */
    transmit frame on device A

    /*
     * device A will notify user when to return these frames
     * back to physical_pool_no
     */

```

END

A



## APPENDIX B ABBREVIATIONS

<b>CAM</b>	Content Addressable Memory
<b>CR</b>	Command Register
<b>DISC</b>	DISConnect
<b>DLCI</b>	Data Link Connection Identifier
<b>DL</b>	Data Link entity
<b>DM</b>	Disconnected Mode
<b>DMI</b>	Digital Multiplexed Interface
<b>F</b>	Final
<b>FD</b>	Frame Descriptor
<b>FRMR</b>	FRaMe Reject
<b>GCB</b>	Global Configuration Block
<b>GCBP</b>	Global Configuration Block Pointer
<b>GCBPR</b>	Global Configuration Block Pointer Register
<b>I</b>	Information
<b>ISDN</b>	Integrated Services Digital Network
<b>IVR</b>	Interrupt Vector Register
<b>LAPD</b>	Link Access Procedure on D-channel
<b>LL</b>	Logical Link
<b>LLID</b>	Logical Link IDentification
<b>LLT</b>	Logical Link Table
<b>MDL</b>	Management of Data Link entity
<b>MLAPD</b>	Multi-link LAPD controller
<b>P</b>	Poll
<b>P/F</b>	Poll/Final
<b>REJ</b>	REJect
<b>RNR</b>	Receiver Not Ready
<b>RR</b>	Receiver Ready

<b>Rx</b>	Receive
<b>S</b>	Supervisory
<b>SABME</b>	Set Asynchronous Balanced Mode Extended
<b>SR</b>	Semaphore Register
<b>Tx</b>	Transmit
<b>U</b>	Unnumbered
<b>UA</b>	Unnumbered Acknowledge
<b>UI</b>	Unnumbered Information

INTRODUCTION	1
MEMORY STRUCTURES	2
INTERNAL REGISTERS	3
COMMAND SET	4
TRANSMIT PROCESS	5
RECEIVE PROCESS	6
EXCEPTION PROCESSING	7
MLAPD IMPLEMENTATION OF LAPD	8
MLAPD IMPLEMENTATION OF SPECIAL MODES	9
SIGNAL DESCRIPTION	10
BUS OPERATION	11
ELECTRICAL SPECIFICATIONS	12
MECHANICAL DATA	13
SOFTWARE INTERFACE FLOWS FOR THE MLAPD	A
ABBREVIATIONS	B

- 1 INTRODUCTION**
- 2 MEMORY STRUCTURES**
- 3 INTERNAL REGISTERS**
- 4 COMMAND SET**
- 5 TRANSMIT PROCESS**
- 6 RECEIVE PROCESS**
- 7 EXCEPTION PROCESSING**
- 8 MLAPD IMPLEMENTATION OF LAPD**
- 9 MLAPD IMPLEMENTATION OF SPECIAL MODES**
- 10 SIGNAL DESCRIPTION**
- 11 BUS OPERATION**
- 12 ELECTRICAL SPECIFICATIONS**
- 13 MECHANICAL DATA**
- A SOFTWARE INTERFACE FLOWS FOR THE MLAPD**
- B ABBREVIATIONS**





**Literature Distribution Centers:**

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; P.O. Box 80300; Cheung Sha Wan Post Office; Kowloon Hong Kong.

JAPAN: Nippon Motorola Ltd.; 3-20-1 Minamiazabu, Minato-ku, Tokyo 106 Japan.



**MOTOROLA**