Micro Channel Adapter Handbook

ALTERA

Micro Channel Adapter Handbook

April 1990

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
(408) 984-2800
Applications Information:
(408) 984-2805 ext. 102
Marketing Information:
(408) 984-2805 ext. 101

About This Handbook

This handbook contains two data sheets, two application notes, and one application brief about Altera's user-configurable adapter interface chips for the PS/2 Micro Channel. The handbook is a comprehensive design guide for interfacing a PS/2 adapter card and the Micro Channel, and for providing Micro Channel Direct Memory Access (DMA) arbitration.

EPB2001: Adapter Card Interface Chip for PS/2 Micro Channel

Describes the general-purpose interface EPB2001 device, including bus control, POS register, and chip-select logic. It also describes how to use the MCMap development software, and how to order a development system. AC characteristics and timing diagrams are also provided.

EPB2002A: DMA Bus Arbitration Interface Chip for PS/2 Micro Channel

Describes the DMA interface EPB2002A device, including POS register and arbitration logic. AC characteristics and timing diagrams are also provided.

Application Note 14: IBM PS/2 Adapter Card Interfacing with the EPB2001 and EPB2002A

Illustrates use of the EPB2001 and EPB2002A with the IBM PS/2 Micro Channel and offers a set of general design tips. This Application Note complements *Application Note 15*.

Application Note 15: IBM PS/2 Adapter Card Software Design

Describes the software design criteria for an adapter card for the Micro Channel Bus architecture described in *Application Note 14*. It discusses installation and configuration of adapter cards, and provides a utility to interrogate each adapter slot and read back the adapter ID and POS register contents.

Application Brief 72: IBM PS/2 Master and Slave Adapter Design

Shows how the EPB2002A may be used as a building block to construct a bus master peripheral adapter for the IBM Micro Channel.

For additional information, please contact Altera Applications at (408) 984-2805 ext. 102.

Contents

Application Note 14 (continued)

Application Note 15: IBM PS/2 Adapter Card Software Design

Application Brief 72: IBM PS/2 Master and Slave Adapter Design

EPB2001

Adapter Card Interface Chip for PS/2 Micro Channel

April 1990, Rev. 1 **Data Sheet**

Features

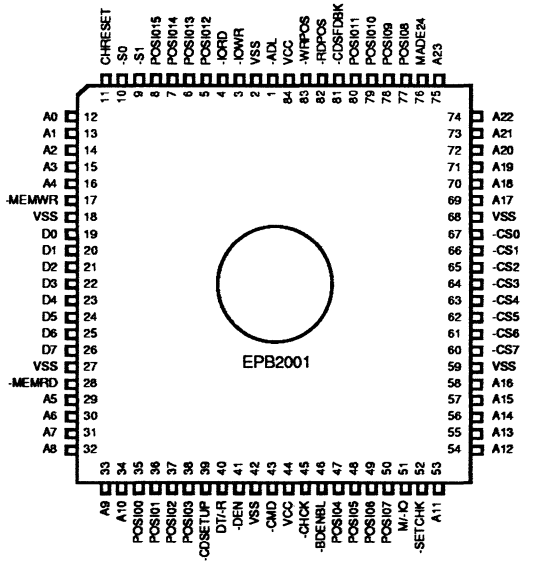
- User-configurable, single-chip interface for PS/2 Micro Channel Bus adapter cards
- Integration of basic interface functions into a single 84-lead device
  - Programmable Option Select (POS) registers 0102-0105 with 16 programmable card-interface I/O lines
  - 2-byte programmable card ID EPROM (POS registers 0100 and 0101)
  - 24-mA, byte-wide Micro Channel data bus port
  - Card address remapping/chip-select decode function provided by 8 chip-select blocks, each having 8 programmable address ranges
  - Card control lines for control of card memory, I/O, and transceivers
  - Support of channel-check and card-enable functions
- 100% compatible with Micro Channel AC timing and DC output drive specifications
- EPROM Security Bit to protect proprietary designs
- Quick PC-based design entry with MCMap design software

General Description

The Altera EPB2001 (Figure 1) is an 84-lead, function-specific Erasable Programmable Logic Device (EPLD). It provides—in a single chip—all essential functions to interface a PS/2 add-on card (called *adapter* by IBM)

Figure 1. EPB2001 Package Pin-Out Diagram

Note: The EPB2001LC is identical but has no window.

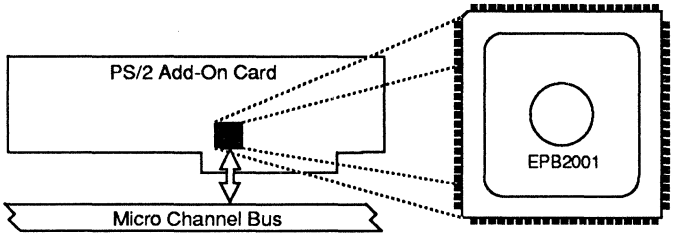


and the Micro Channel Bus (MC Bus). The EPB2001 is an ideal chip for manufacturers of IBM PS/2 adapter cards based on Micro Channel architecture, since it allows specific card characteristics to be programmed for a specific application. The EPB2001's integrated functions can replace 18 or more MSI/TTL and standard PLDs.

Although the EPB2001 occupies less than two square inches on the PC card (see Figure 2), its CMOS EPROM technology provides non-volatile storage of card ID, chip-select ranges, and POS I/O selection for reduced component count and added design security.

Figure 2. EPB2001 Single-Chip, Small-Footprint Interface

The EPB2001 occupies less than two square inches on the PS/2 card.

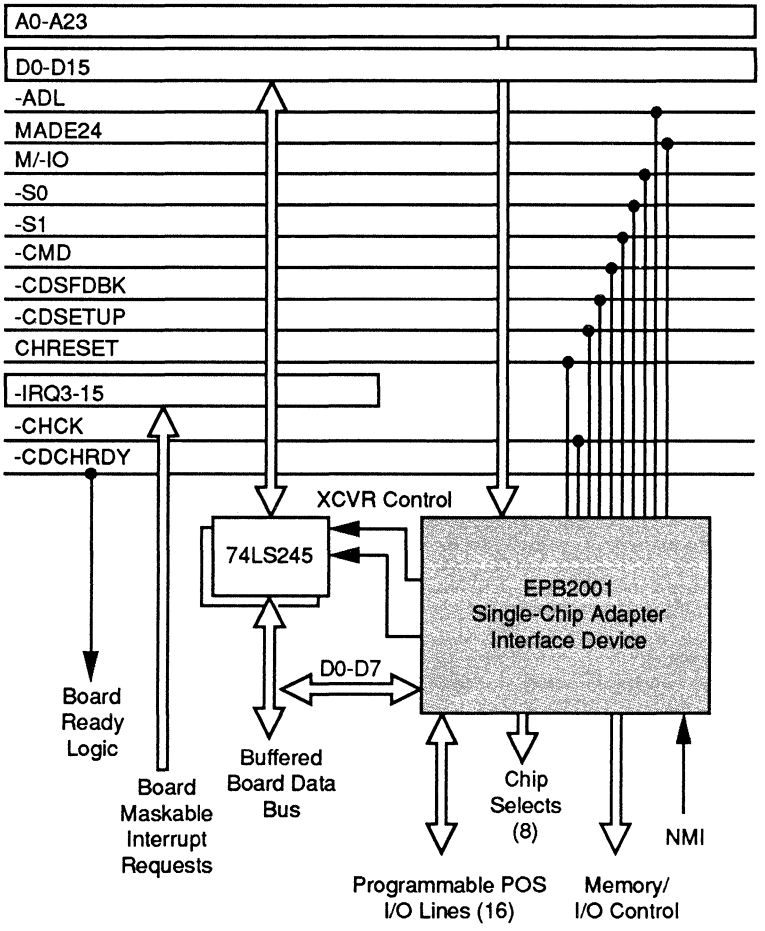


More specifically, the EPB2001 provides the following general-purpose interface functions: POS registers 0100-0105 (including card ID), access to POS register contents on card-accessible I/O lines (replacing jumpers and DIP switches on IBM PC and compatible adapter cards), card address remapping via programmable chip-select logic, and card control signals (e.g., **-MEMWR**, **-IORD**). Figure 3 shows the EPB2001 integrated into the Micro Channel environment.

The EPB2001 is available in both one-time-programmable plastic and erasable/reprogrammable ceramic J-lead chip carrier packages.

Figure 3. Micro Channel Interface Connection

The EPB2001 provides a convenient interface to the Micro Channel Bus.



MCMap, Altera's PC-based development software, makes designing with the EPB2001 quick and easy. This table-driven software guides the user through a series of menus to generate the desired design and convert it to a JEDEC file. MCMap also generates an Adapter Description File (ADF) for address decodes controlled by POS register bits. Finally, the converted design is programmed into the EPB2001 in seconds with LogicMap software, the LP4, LP5, or LP6 Programming Card, the PLE3-12A Master Programming Unit, and the PLEJ2001 Programming Adapter.

Figure 4 shows the EPB2001 block diagram, with Micro Channel interface signals on the left and card interface signals on the right.

Figure 4. EPB2001 Block Diagram

The programmable elements of the EPB2001 (shaded) allow it to be customized for a wide variety of applications.

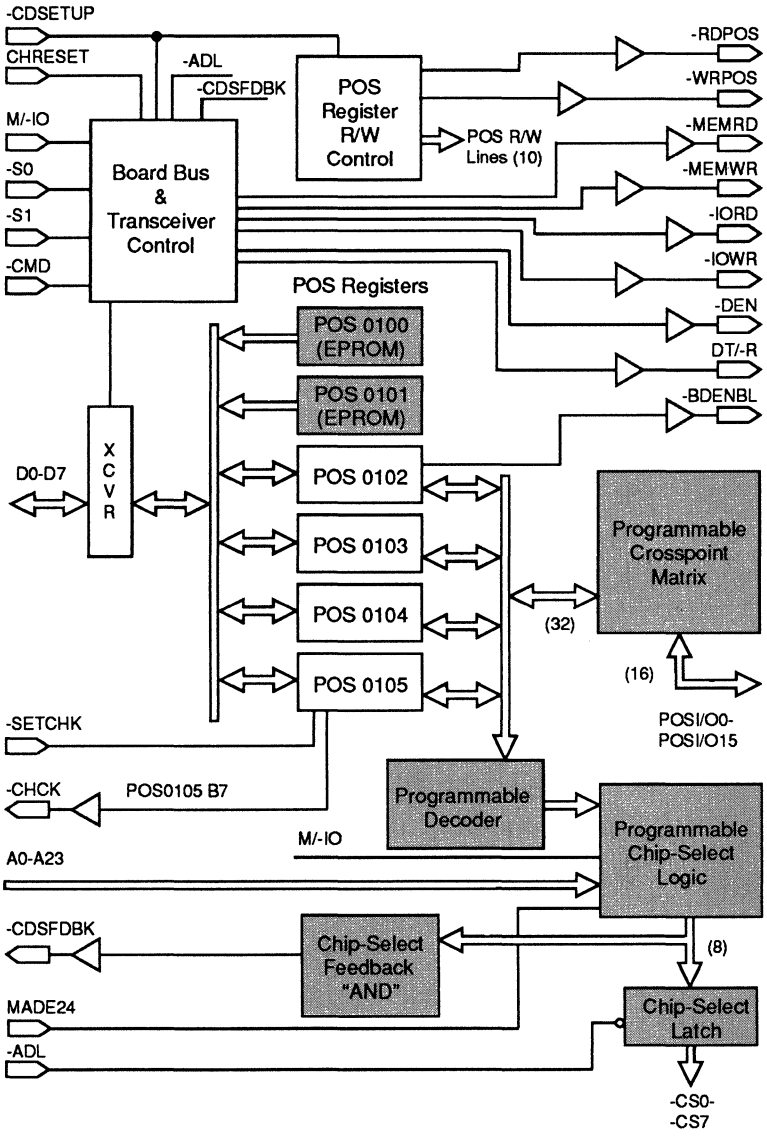


Table 1 summarizes the functions of EPB2001 pins. Active-low signals are prefixed with a dash (–).

Table 1. EPB2001 Pin Descriptions

Signal	Type*	Output Drive (mA) I_{OH}	Output Drive (mA) I_{OL}	Description
–CDSFDBK	TP	2	6	Active-low bus-cycle acknowledge output generated by the EPB2001 for any I/O or memory cycle that activates one of the –CS outputs. Derived as a decode of unlatched chip selects.
–CHCK	OD		24	Active-low channel-check output used to signal Non-Maskable Interrupt errors. Reflects the state of POS register 0105, bit 7. Activated by an active-low input pulse on the –SETCHK line.
D0-D7	TS	4	24	Tri-state bidirectional data bus lines. POS register read/write data access path. Enabled to the MC Bus only during a valid I/O read cycle qualified by –CDSETUP.
–CDSETUP	I			Active-low set-up input. Signals a POST read or write cycle to the EPB2001's POS registers.
M/–IO	I			Memory/I/O cycle input from the MC Bus. High for memory cycles, low for I/O cycles.
–S0, –S1	I			Bus-cycle status input lines from the MC Bus. (Codings for various cycles are shown in Table 2.) Together with –CMD, used to generate card control lines (–MEMWR, –DEN, etc.).
–CMD	I			Active-low bus-cycle strobe input. Used to time data transfers during read and write operations.
CHRESET	I			Active-high channel-reset input. The EPB2001 deasserts all active outputs a short time after CHRESET rises. POS register 0102, bit 0 is also reset by this input, deactivating –BDENBL.
MADE24	I			Active-high input indicating a 24-bit address is present on the MC Bus for the current cycle. When low, indicates that an extended address (32 bits) is present.
A0-A23	I			MC Bus address inputs. Valid while –ADL is low.
–ADL	I			Active-low address-latch input. Leading edge of this signal is used to latch addresses and (optionally) chip-select lines.
–MEMRD	TP	4	24	Active-low memory-read strobe output. Generated as a decode of –S0=1, –S1=0, and M/–IO=1, timed by –CMD.
–MEMWR	TP	4	24	Active-low memory-write strobe output. Generated as a decode of –S0=0, –S1=1, and M/–IO=1, timed by –CMD.
–IORD	TP	4	24	Active-low I/O-read strobe output. Generated as a decode of –S0=1,–S1=0, and M/–IO=0, timed by –CMD.
–IOWR	TP	4	24	Active-low I/O-write strobe output. Generated as a decode of –S0=0,–S1=1, and M/–IO=0, timed by –CMD.
–DEN	TP	4	6	Active-low transceiver-enable output. Low during data transfer portion of selected MC Bus cycles.
DT/–R	TP	4	6	Data transceiver direction-control output. High during MC Bus read cycles and low during bus write cycles.
–CS0-7	TP	4	6	Active-low chip-select outputs. Derived as a decode of addresses, M/–IO, and MADE24. May be individually latched by –ADL. Eight user-defined address ranges per output, enabled by groups of POS register bits.

* Signal Types:
I = Input TP = Totem-pole (push-pull) output OD = Open-drain output TS = Bidirectional tri-state I/O

Table 1. EPB2001 Pin Descriptions (continued)

Signal	Type*	Output Drive (mA) I_{OH}	Output Drive (mA) I_{OL}	Description
POSI/O0 - POSI/O15	I/OD		6	Bidirectional POS I/O lines. Each open-drain output is driven by a user-defined POS register bit. State of POS I/O pin is reflected when corresponding POS bit is read through MC Bus port, and POS register contents equal 1 (default).
–WRPOS	TP	4	6	Active-low POS register write strobe. Active for POS set-up cycle. Used to control optional POS functions external to the EPB2001.
–RDPOS	TP	4	6	Active-low POS register read strobe. Active for POS set-up cycle. Used to control optional POS functions external to the EPB2001.
–BDENBL	OD		24	Active-low card-enable output. Open-drain output reflects the state of POS register 0102, bit 0. Active low when this register bit is set to a 1. Deactivated by CHRESET.
–SETCHK	I			Active-low set channel-check input. A low pulse on this input resets POS register 0105, bit 7, and thereby activates the –CHCK output to the MC Bus.
V_{CC} (2 pins)				+5 V power supply.
V_{SS} (6 pins)				Ground.

* Signal Types:
I = Input TP = Totem-pole (push-pull) output OD = Open-drain output TS = Bidirectional tri-state I/O

Bus Control Section

The EPB2001 contains bus control logic, shown in the upper portion of Figure 4, which generates read and write signals for the internal POS registers, as well as card control signals such as **–MEMWR** and **–IORD**. The bus control section interfaces with the **–CDSETUP**, **–S0**, **–S1**, **–CMD**, and **M/–IO** inputs on the Micro Channel side, and the **–DEN**, **DT/–R**, **–IOWR**, **–IORD**, **–MEMWR**, **–MEMRD**, **–RDPOS**, and **–WRPOS** outputs on the card side.

This section is activated by either an active **–CDSETUP** line together with an I/O read or write cycle from the processor (indicating a POS setup configuration cycle), or a valid bus cycle (i.e., I/O read or write, memory read or write). An active **–CDSFDBK** output indicates a bus cycle for the adapter card, unless it occurs during setup. The card must have previously been enabled to activate the outputs of this bus control section. (See "POS Register Section" later in this data sheet.)

The bus control section interprets combinations of the **–S0**, **–S1**, and **M/–IO** signals as MC Bus cycles. The interpretation of each possible signal combination is shown in Table 2.

The states of the **–S0**, **–S1**, and **M/–IO** lines, like those of **–CDSETUP** and addresses **A0–A23**, are latched by the falling edge of **–ADL** for the duration of the cycle.

Table 2. Coding for MC Bus Cycles

M/-IO	-S0	-S1	Cycle Type
0	0	0	No Operation
0	0	1	I/O Write
0	1	0	I/O Read
0	1	1	No Operation
1	0	0	No Operation
1	0	1	Memory Write
1	1	0	Memory Read
1	1	1	No Operation

The **-CMD** signal, acting as a command strobe, times the generation of the appropriate control lines. Therefore, **-MEMRD**, **-MEMWR**, **-IORD**, and **-IOWR** are about as long as the **-CMD** signal.

DT/-R controls the direction of data flow through an external data transceiver. It changes after **-ADL** falls, and remains latched for the duration of the cycle. It is low for all write cycles.

-DEN controls external data transceiver output enables. It is active during a valid read or write cycle for about the same time as **-CMD**.

To minimize MC Bus loading, the EPB2001 data bus pins (**D0-D7**) should be connected to the adapter card's buffered data bus rather than directly to the Micro Channel (see Figure 3). By locating the data bus pins behind the card bus transceiver, only one load per data pin is given to the Micro Channel. To support this operating mode, the EPB2001 **-DEN** output is active during **-CDSETUP** cycles. If, however, EPB2001 data pins are directly connected to the Micro Channel, the **-DEN** output *must* be externally disabled during setup operations to ensure that there is no contention between the EPB2001 data pins and the card data transceiver.

The adapter card setup procedure, which is part of the Power-On System Test (POST), can occur only when **-CDSETUP** is active on the rising edge of **-ADL** followed by an I/O read or write cycle. The falling edge of **-ADL** may be used to latch addresses for any type of cycle, and is used during setup to latch **A0-A2** to select the correct POS registers.

The **-RDPOS** and **-WRPOS** signals control optional external POS register functions. They are active for any POS read or write operation accompanied by **-CDSETUP**. Timing for these signals is similar to that of **-IOWR** and **-IORD**.

If **CHRESET** is asserted, any bus cycle in progress is immediately halted, and the **D0–D7** outputs of the EPB2001 chip are tri-stated. In addition, the adapter-card control lines immediately become inactive.

The **–MEMRD**, **–MEMWR**, **–IORD**, and **–IOWR** outputs have 24-mA push-pull output drivers. The **–DEN**, **DT/–R**, **–RDPOS**, and **–WRPOS** outputs have 6-mA push-pull drivers.

The EPB2001's internal transceiver (connected to **D0–D7**) is only enabled to the MC Bus during an I/O read qualified by **–CDSETUP**.

POS Register Section

The POS registers, shown in the middle of Figure 4, are accessible through the dedicated transceiver associated with pins **D0–D7** on the EPB2001. Data is transferred to the selected POS register in a write operation while **–CMD** is low. The rising edge of **–CMD** then latches the input data into the register. Data is read from the POS registers while **–CMD** is low, and becomes valid at the **D0–D7** pins after the **–CMD** falling edge. (See "EPB2001 Waveforms" later in this data sheet for **–CMD** timing information.)

The required POS registers reside in a block at I/O addresses 0100-0105H for all cards. All registers are byte-wide. Locations 0100 and 0101, which are read-only, non-volatile EPROM locations, contain the card ID. POS registers 0102-0105 are user-defined, with the exception of the following three bit locations:

Bit 0 of register 0102: This bit is used as an adapter-card enable bit. It is reset by **CHRESET** or by the processor writing a 0 to this bit during a **–CDSETUP** cycle. When the bit is set to 0, the EPB2001 (and card) does not respond to any normal bus cycles. Only setup reads and writes are allowed. When the processor sets the bit to 1, the card is enabled. This card-enable bit cannot be written to location 0102 by normal I/O write operations. The **–BDENBL** signal on the card interface reflects the state of this bit for on-card use. The **–BDENBL** pin uses a 24-mA open-drain output structure.

Bit 7 of register 0105: This bit is used as a channel-check flag. A card reports Non-Maskable Interrupts (NMIs) to the processor by asserting the **–CHCK** (channel check) line, which is wire-ORed to all cards. On the EPB2001, a pulse on the **–SETCHK** input resets this bit to a **–CHCK** output on the MC Bus. This bit may be reset by a write to 0105 with 0 in the bit 7 location. The channel-check flag bit is set by a **CHRESET** or by writing a 1 to the bit 7 position.

Bit 6 of register 0105: This bit may be used if bit 7 is set to 1. It flags availability of channel-check exception status in optional POS registers 0106 and 0107. (These registers, if used, are typically implemented in components such as 74LS374s.) If channel-check exception status is provided in POS registers 0106 and 0107, a 0 is found in the bit 6 location; if not, a 1

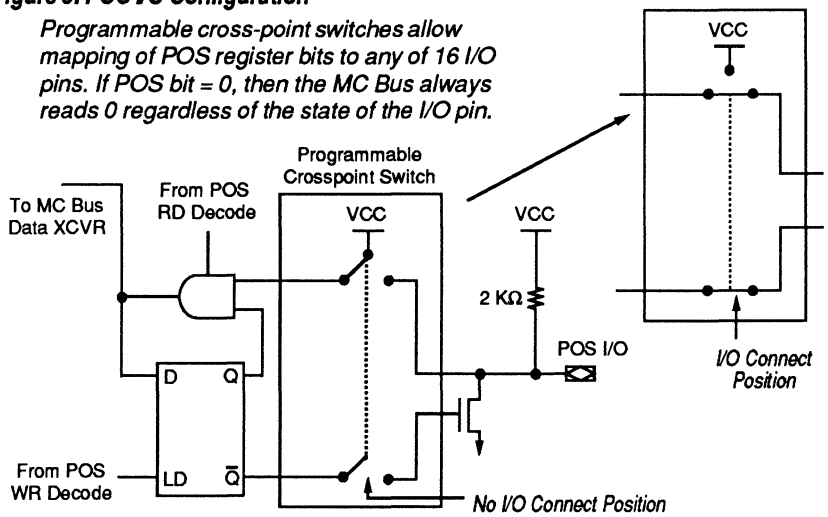
is found. If bit 6 is used, one of the programmable POS I/O pins on the adapter card interface may be used to force the appropriate value.

All remaining bits are user-definable. These bits may be used for address remapping control (i.e., software-controlled "jumpers") or simply for general input or output port functions on the card. Each POS I/O pin can be used for input or output, and may be assigned to any POS register bit. The remapping function is discussed in "Chip-Select Logic."

The connection of any of the 31 POS register bits (locations 0102-0105, exclusive 0102 bit 0) with the 16 dedicated POS I/O pins on the adapter card interface is controlled by a user-programmable cross-point switch arrangement (see Figure 5). Each POS I/O pin has a 6-mA open-drain output structure as well as an input path. On the output side, a programmable matrix takes the output of any of the POS register bits and assigns it to any of the 16 output lines. Since the pins are open drain, if a 1 is written to a given POS register bit from the MC Bus, the associated I/O pin is not driven. The I/O pin can therefore be driven by an external signal source, and its value may be read through the corresponding POS register bit location. However, forcing a value from the POS I/O pins does not change the value in the POS register location. All POS I/O pins should either be driven by an external signal source or connected to external pull-up resistors. Resistors should have a value between 2 K and 10 K ohms.

Figure 5. POS I/O Configuration

Programmable cross-point switches allow mapping of POS register bits to any of 16 I/O pins. If POS bit = 0, then the MC Bus always reads 0 regardless of the state of the I/O pin.



Chip-Select Logic

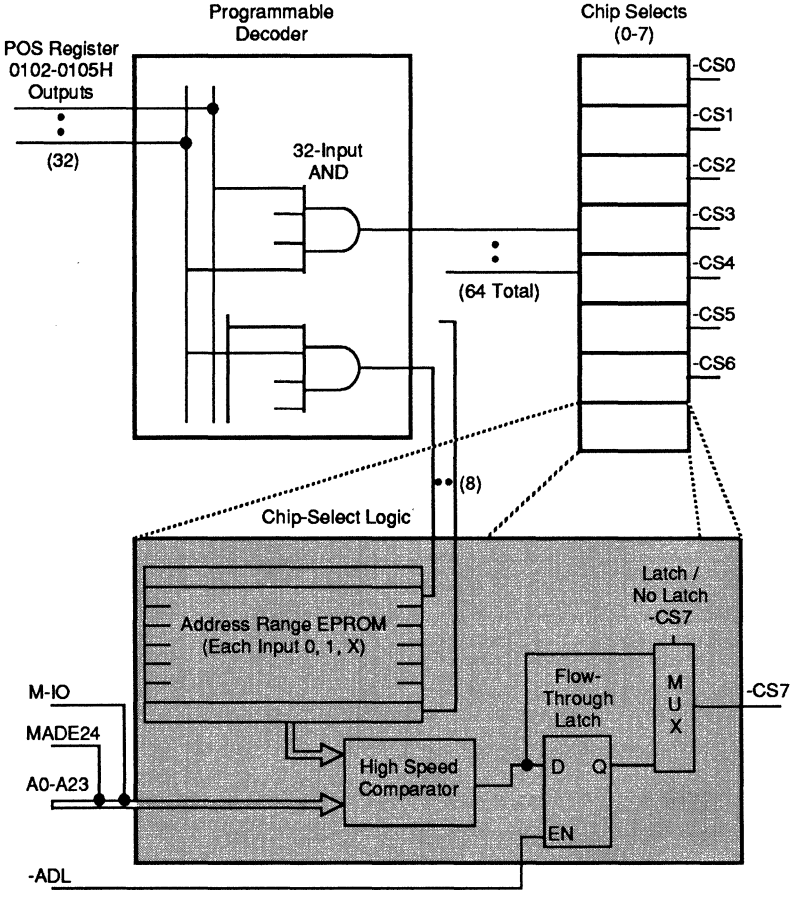
The chip-select logic on the EPB2001 (Figure 6) provides up to 8 user-programmable chip selects. Each chip-select output (**-CS0** through **-CS7**) has a 6-mA push-pull driver and is active-low. Each may also have up to 8 pre-programmed address ranges over which it is active. The granularity of these chip selects may range from one location to the entire 24-bit

(16-Mbyte) physical address range. Each may be defined for either memory or I/O mapping. All 24 MC Bus addresses and the **M/-IO** input enter the programmable chip-select logic.

An additional input to the programmable chip-select arrays may act as an enable for the chip selects. Typically, this input is connected to the **MADE24** MC Bus signal to qualify chip selects when 32-bit addressing is involved.

Figure 6. Chip-Select Logic

EPROM arrays allow programmable address range decoding.



Normally, chip-select outputs are not latched, and are valid only during a valid combination of address and **M/-IO** on the bus. Optionally, the chip-select outputs may be individually latched by user-programmable, flow-through latches enabled by **-ADL**. This latching causes the affected chip-select output(s) to become active a short time after **-ADL** has become active-low (the **A0-A23** address lines and **M/-IO** input having stabilized well before **-ADL** falls). The outputs are latched on the **-ADL** falling edge and remain active until the next bus cycle, when **-ADL** goes low again. Latched/non-latched operation for each chip-select output is determined by the user when the device is programmed.

The chip-select logic is implemented as 8 distinct logic blocks, with 1 block per chip select. Each block consists of an 8-word-by-52-bit programmable memory (416 bits) feeding a comparator and the address and other required inputs. Each word corresponds to a desired chip-select active range. Any input bit may be compared for 0, 1, or "don't care" in determining an address match. Thus, 2 bits per input (26 inputs x 2 = 52 bits) are needed to encode these possibilities.

The selection of one of the eight available chip-select ranges for a particular chip-select output (corresponding to one of the eight words in each of the blocks) is determined by user-defined combinations of POS register bits. The user may define the POS register bits and bit combinations that activate a given chip-select range. This information is then coded into a programmable decoder on the device that generates an enable for each range. The PS/2 operating system may remap address ranges during the POST setup if there is a conflict, i.e., two or more cards responding to the same address range. Address ranges are remapped by changing the POS register bits controlling the chip selects, enabling a new address range.

All of the unlatched, active-low chip-select outputs may be logically ANDed to form the **-CDSFDBK** signal presented to the MC Bus. MCMap allows the user to choose any combination of the eight chip selects to include for **-CDSFDBK**. The resulting output, acting as a "cycle acknowledge" line, signals a valid bus cycle for the card to the MC Bus. **-CDSFDBK** is active-low and has a 6-mA push-pull driver.

If **CHRESET** is active, all chip-select latches are immediately cleared to the inactive (high) state.

Design Guidelines

The EPB2001 will be permanently damaged if it is operated under conditions that surpass those listed under "Absolute Maximum Ratings." This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability. The EPB2001 contains circuitry to protect device pins from high static voltages or electric fields; however, normal precautions

should be taken to avoid application of any voltage higher than maximum rated voltages.

For proper operation, input and output pins must be constrained to the range GND < (V_{IN} or V_{OUT}) < V_{CC}. Unused inputs must always be tied to an appropriate logic level (either V_{CC} or GND). Each set of V_{CC} and GND pins must be connected directly at the device. Power-supply decoupling capacitors of at least 0.2 µF must be connected between V_{CC} and GND. For the most effective decoupling, each V_{CC} pin should be separately decoupled to GND, directly at the device.

Latch-Up & ESD Protection

The EPB2001 input, output, and I/O pins are designed to resist electrostatic discharge (ESD) and latch-up damage. Each device pin will withstand voltage energy levels exceeding those specified by MIL-STD-883C. Pins will not latch up for input voltages between -1 V and V_{CC} + 1 V with currents up to 100 mA. During transitions, the inputs may undershoot to -2.0 V for periods less than 20 ns. Additionally, the programming pin is designed to resist latch-up to the 14.0 V maximum device limit.

Device Erasure

The EPB2001 begins to erase when exposed to light wavelengths shorter than 4000 Å. Since fluorescent lighting and sunlight fall into this range, opaque labels should be placed over the EPB2001 window to ensure long-term reliability. (Plastic packaged devices are, of course, protected). Constant exposure to room-level fluorescent lighting could erase an EPB2001 in about three years, and direct sunlight could erase it in about one week.

To ensure proper erasure, the EPB2001 must be exposed to ultraviolet light with a wavelength of 2537 Å. The integrated erasure dose should be a minimum of 30 Wsec/cm^2. The erasure time with this dosage is about one hour using an ultraviolet lamp with a 12,000 µW/cm^2 power rating. During erasure, the EPB2001 should be placed within 1 inch of the lamp tubes. The maximum integrated exposure dose for an EPB2001 is 7,000 Wsec/cm^2. This dose is equivalent to one week at 12,000 µW/cm^2. Exposure of the windowed EPB2001 to high-intensity ultraviolet light for long periods of time may cause permanent damage.

The EPB2001 may be erased and reprogrammed as often as necessary within the limits described and using the recommended procedure.

Design Security

The EPB2001 contains a programmable Security Bit that controls access to programmed information. If this Security Bit is used, the custom pattern in the device is secured from external interrogation and possible reverse engineering. The Security Bit, which is set by the user during design entry, may be erased with ultraviolet light as described in "Device Erasure."

MCMap Development System

Altera provides MCMap, a PC-based design development system, to support efficient design and use of the EPB2001. It is an interactive, table-driven software package. The designer is prompted for information concerning the programmable portions of a design: card ID, chip-select ranges, POS register-bit combinations used as enables, etc. Real-time error checking reports any errors as they are entered. When entry is complete, MCMap compiles a JEDEC programming file for the EPB2001 in seconds. MCMap also generates an Adapter Description File (ADF) for address decodes controlled by POS register bits.

The JEDEC file may then be submitted to LogicMap version 6.5 (or a later version) to program the EPB2001 on the PC. Altera's LP4, LP5, or LP6 programming card, PLE3-12A Master Programming Unit, and PLEJ2001 Programming Adapter are the required hardware. The PLEJ2001 provides an interface between the LP4 and PLE3-12A and the 84-lead EPB2001 chip carrier package. For more information about development systems, please contact Altera's Marketing Department at (408) 984-2805 ext. 101.

The recommended PC system requirements for Altera's MCMap software and hardware are:

- ❒ IBM XT, AT, or compatible PC
- ❒ EGA,VGA, or Hercules Graphics Adapter
- ❒ 640 KBytes RAM
- ❒ 10-MByte hard disk and 5.25 inch floppy drive
- ❒ DOS Version 3.3 or a later version

Ordering Information

Development Systems

Part Number	Description
PLDS-MCMAP	Stand-alone Programmable Logic Development System. Contains MCMap programmable logic development software and documentation, PL-ASAP programming hardware, PLEJ2001 adapter, and sample EPB2001JC.
PLS-MCKIT	Software only from PLDS-MCMAP system. Also contains PLEJ2001 adapter and sample EPB2001JC.

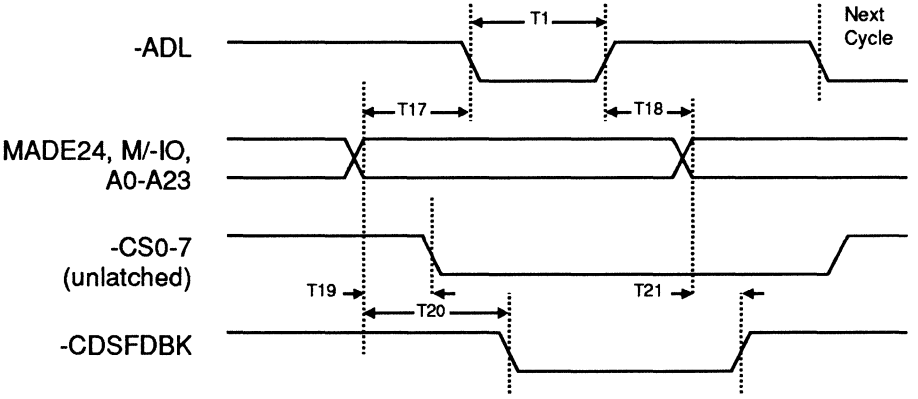
Components

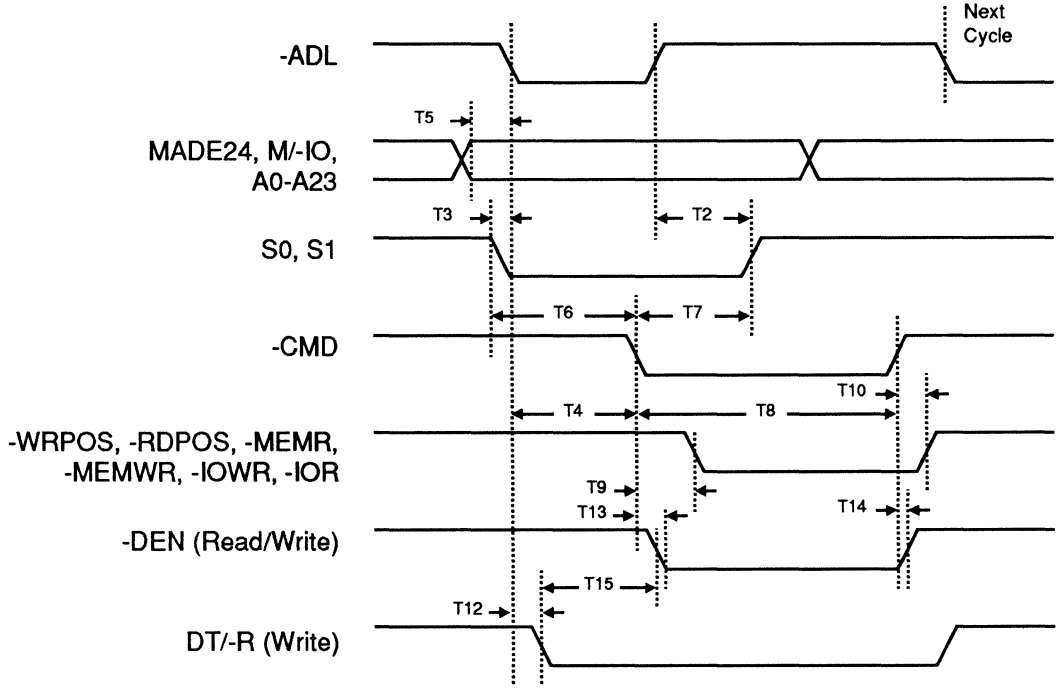
Part Number	Description
EPB2001JC	Windowed ceramic J-lead chip carrier
EPB2001LC	One-time-programmable plastic J-lead chip carrier

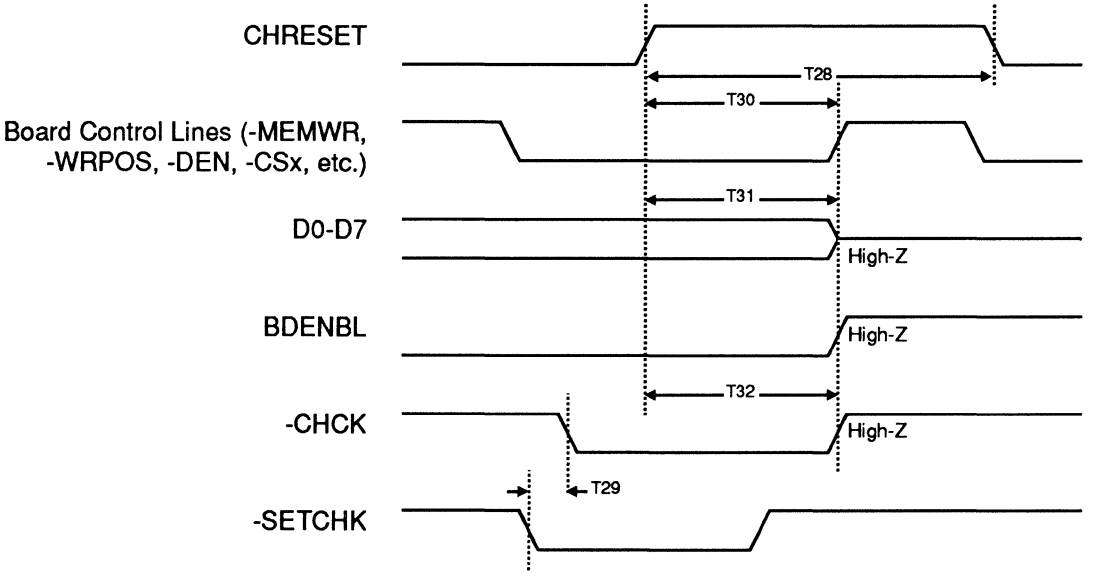
Reference

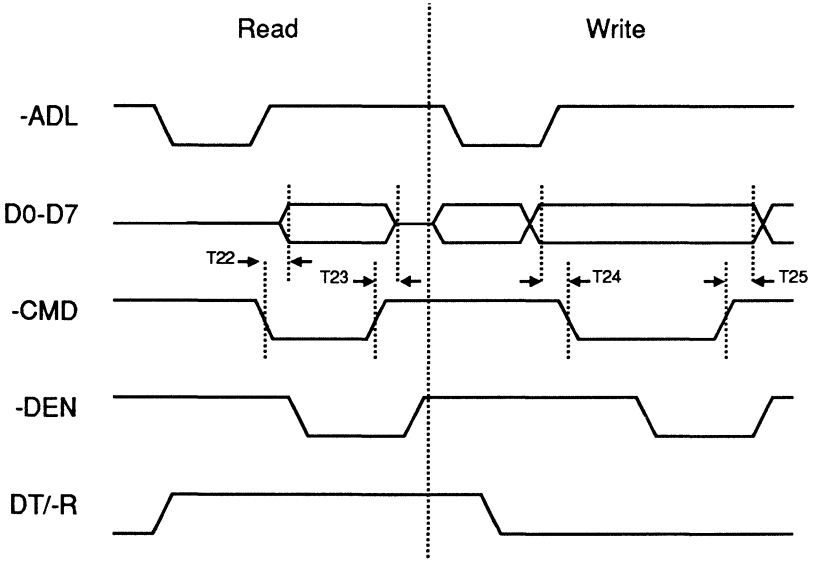
"Hands-on Experience Paves the Way for Future MCA Designs." EDN (November 9, 1989): 233.

EPB2001
Waveforms






CHRESET
Board Control Lines (-MEMWR, -WRPOS, -DEN, -CSx, etc.)
D0-D7
BDENBL
-CHCK
-SETCHK
T28
T30
T31
T32
T29
High-Z
High-Z
High-Z


Read
Write
-ADL
D0-D7
-CMD
-DEN
DT/-R
T22
T23
T24
T25

Absolute Maximum Ratings

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	-2.0	7.0	V
V_{PP}	Programming supply voltage	*See Note (1)*	-2.0	14.0	V
V_I	DC input voltage	*See Note (1)*	-2.0	V_{CC} + 1.0	V
I_{MAX}	DC V_{CC} or GND current		-500	+500	mA
I_{OUT}	DC output current, per pin		-50	+50	mA
P_D	Power dissipation			1000	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature		0	+70	° C
t_R	Input rise time			250	ns
t_F	Input fall time			250	ns

DC Operating Characteristics

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High-level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low-level input voltage		-0.3		0.8	V
V_{OH}	High-level TTL output voltage	See tables above	2.4			V
V_{OL}	Low-level output voltage	See tables above			0.50	V
I_I	Input leakage current	V_I = V_{CC} or GND	-10		+10	µA
I_{OZ}	Output high-Z leakage current	V_O = V_{CC} or GND	-10		+10	µA
I_{CC}	V_{CC} supply current (standby)	V_I = V_{CC} or GND, No load, *See Note (2)*		20	30	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{OUT} = 0 V *See Note (3)*		15	pF
C_{IO}	I/O capacitance			15	pF
C_{OD}	Output capacitance			15	pF

AC Operating Characteristics

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C

Symbol	Parameter	Conditions	Min	Max	Unit
T1	–ADL width		40		ns
T2	–ADL high to –S0, –S1 high		25		ns
T3	–S0, –S1 low to –ADL low		12		ns
T4	–ADL low to –CMD low		40		ns
T5	MADE24, M/–IO, A0-A23 valid to –S0, –S1 low		10		ns
T6	–S0, –S1 low to –CMD low		55		ns
T7	–CMD low to –S0, –S1 high		30		ns
T8	–CMD width		90		ns
T9	–CMD low to –MEMRD, –MEMWR, –IORD, –IOWR, –RDPOS, –WRPOS low			18	ns
T10	– CMD high to –MEMRD, –MEMWR, –IORD, –IOWR, –RDPOS, –WRPOS high			18	ns
T11	–ADL low to DT/–R high			20	ns
T12	–ADL low to DT/–R low			20	ns
T13	–CMD low to –DEN low			15	ns
T14	–CMD high to –DEN high			15	ns
T15	DT/–R low to –DEN low		20		ns
T16	–ADL low to latched –CS0-7 low			20	ns
T17	MADE24, M/–IO, A0-A23 valid to –ADL low		45		ns
T18	MADE24, M/–IO, A0-A23 hold from –ADL high		25		ns
T19	MADE24, M/–IO, A0-A23 valid to –CS0-7 low			25	ns
T20	MADE24, M/–IO, A0-A23 valid to –CDSFDBK low			40	ns
T21	MADE24, M/–IO, A0-A23 invalid to –CDSFDBK high			45	ns
T22	–CMD low to D0-D7 valid (read)			60	ns
T23	–CMD high to D0-D7 high-Z			40	ns
T24	D0-D7 valid to –CMD low (write)		0		ns
T25	D0-D7 hold from –CMD high (write)		30		ns
T26	POS I/O input valid to POS data valid			175	ns
T27	–CMD low to POS I/O valid			130	ns
T28	CHRESET width		100		ns
T29	–SETCHCK low to –CHK low			30	ns
T30	CHRESET high to –DEN , –MEMWR , –MEMRD, –IOWR, –IORD high			30	ns
T31	CHRESET high to D0-D7 high-Z			30	ns
T32	CHRESET high to –CHCK, –BDENBL high-Z			30	ns

AC Output Capacitance Loadings

Output Pins	Load Capacitance
–CS0-7, –DEN, POSI/O0-15, –WRPOS, –RDPOS	50 pF
–BDENBL, –MEMRD, –MEMWR, –IORD, –IOWR	200 pF
–CDSFDBK, –CHCK, D0-D7	240 pF

Notes to tables:

(1) Minimum DC input is -0.3 V. During transitions, inputs may undershoot to -2.0 V for periods less than 20 ns.
(2) Typical values are for $T_A = 25°$ C and $V_{CC} = 5$ V.
(3) Pin 39 (high-voltage pin during programming) has capacitance of 35 pF max.
(4) Capacitances measured at 25° C. Sample tested only.

EPB2002A

DMA Bus Arbitration Interface Chip for PS/2 Micro Channel

April 1990 **Data Sheet**

Features

- The EPB2002A CMOS device integrates all DMA interface/arbitration functions into a single 28-lead device.
  - Supports Micro Channel arbitration protocol for slave DMA or bus master adapters
  - Provides user-mappable POS register bits for arbitration level and fairness
  - Supports single-transfer and burst-cycle modes
- The EPB2002A is 100% compatible with Micro Channel AC timing and DC output drive specifications

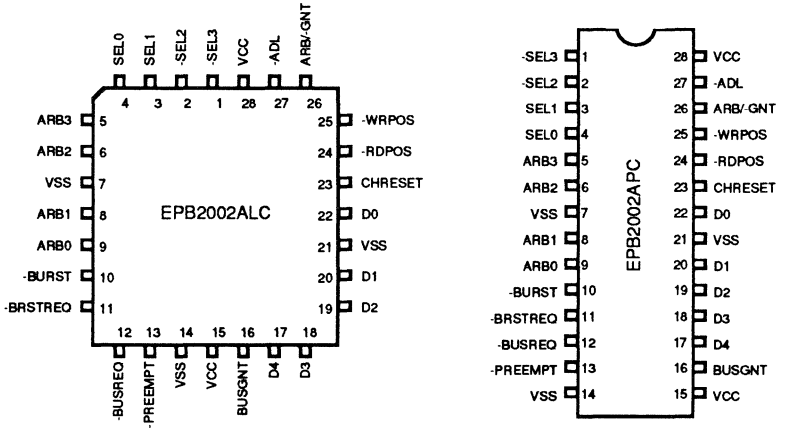
General Description

The Altera EPB2002A (Figure 1) is a 28-lead, function-specific, CMOS device that integrates the support functions for Micro Channel Direct Memory Access (DMA) bus arbitration capability.

The EPB2002A provides full support for the Micro Channel arbitration protocol, including single- or burst-mode transfers, as well as arbitration level and fairness Programmable Option Select (POS) bits. Pin-strapping options allow mapping of the POS bits into any valid locations. When the EPB2002A is used together with the EPB2001, no additional components are required to provide an adapter slave interface.

The EPB2002A is available in plastic 300-mil DIP and plastic J-lead packages.

Figure 1. EPB2002A Package Pin-Out Diagram

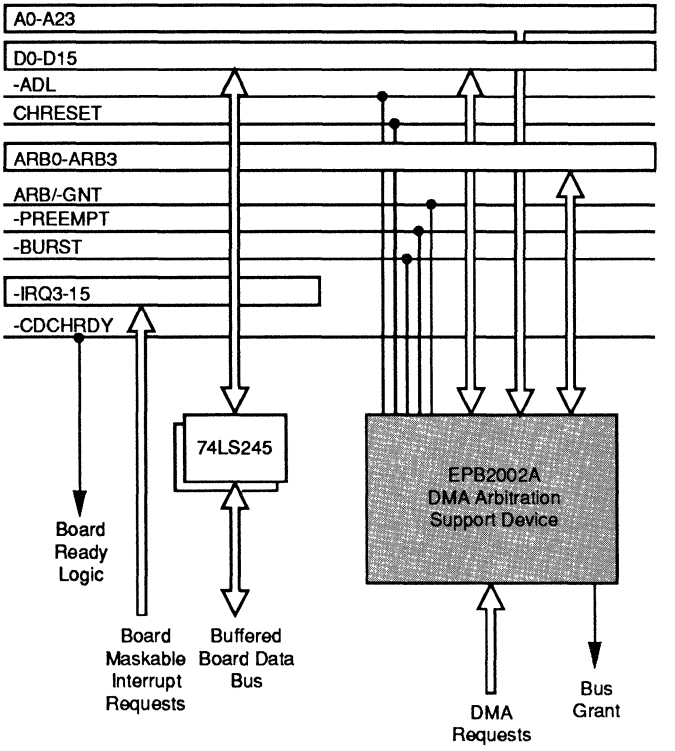


Functional Description

The EPB2002A performs the bus arbitration functions for the Micro Channel interface. There are two primary sections of the chip: the POS register function and the bus arbitration state machine. Figure 2 shows the EPB2002A as part of the Micro Channel environment.

Figure 2. Micro Channel Interface Connection

The EPB2002A provides a convenient interface to the Micro Channel Bus.

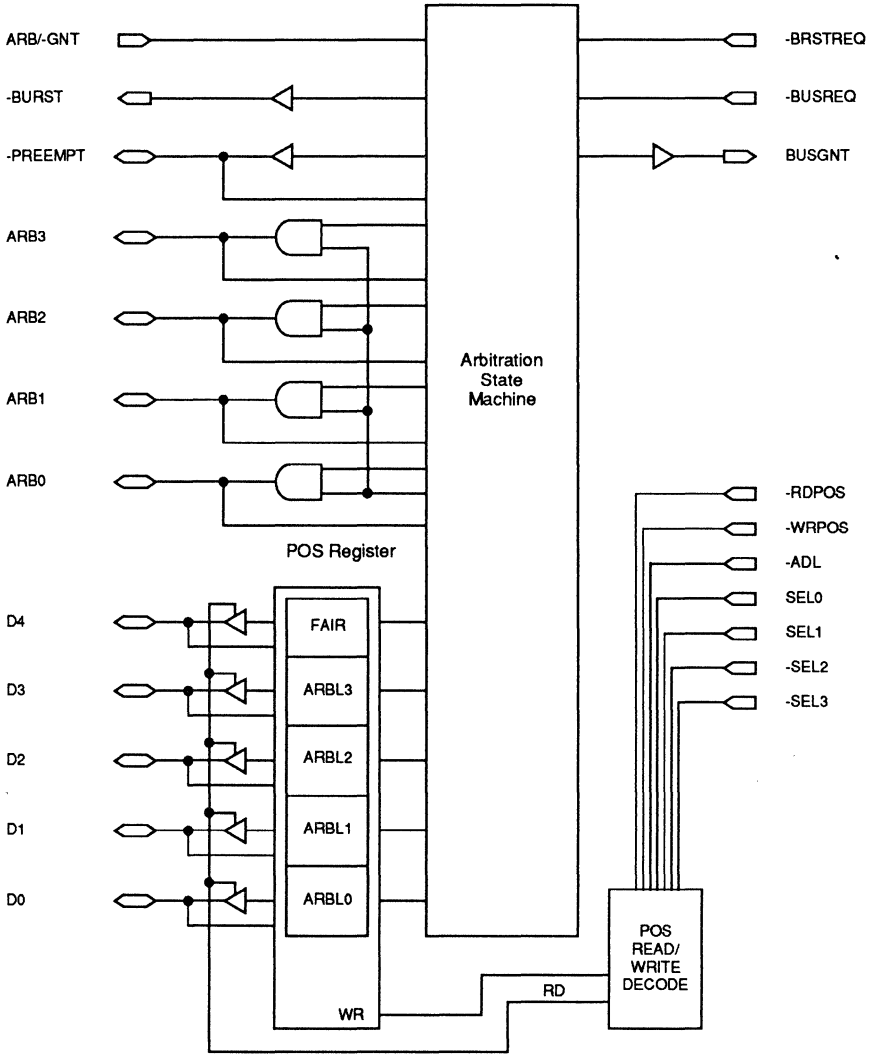


POS Register

Micro Channel Bus (MC Bus) specifications require that each adapter interface supporting DMA functions must have a DMA arbitration/priority level (4 bits) and 1 arbitration fairness bit, all programmable by the PS/2 system's Central Arbitration Control Point (CACP). These 5 bits configure a given board's DMA function at system power-up. They may be placed anywhere in the 30 available POS register bits.

The EPB2002A has five bidirectional data bus lines (**D0** to **D4**) that drive the MC Bus. The pins can be arbitrarily connected to any MC Bus lines (e.g., **D0** pin could be connected to MC Bus **D7**); thus any of these bits may be mapped into any arbitrary set of five POS register bit positions. The bit position is programmable through the connections to the EPB2002A in the printed-circuit board. Figure 3 shows a block diagram of the EPB2002A.

Figure 3. EPB2002A Block Diagram



The **SEL0-3** lines determine in which POS register byte the five bits will reside. **SEL0-1** are active-high, while **-SEL2-3** are active-low. By connecting the **SEL** lines to the appropriate MC Bus address lines, the registers may be mapped into any of the four valid POS register positions shown in Table 1. *All five bits must be in the same POS register byte.*

The 0 and 1 in the "EPB2002A Connection SEL" column are equivalent to hard-strapping these pins to either GND or V_{CC} to obtain the corresponding mapping.

Table 1. Valid POS Register Positions

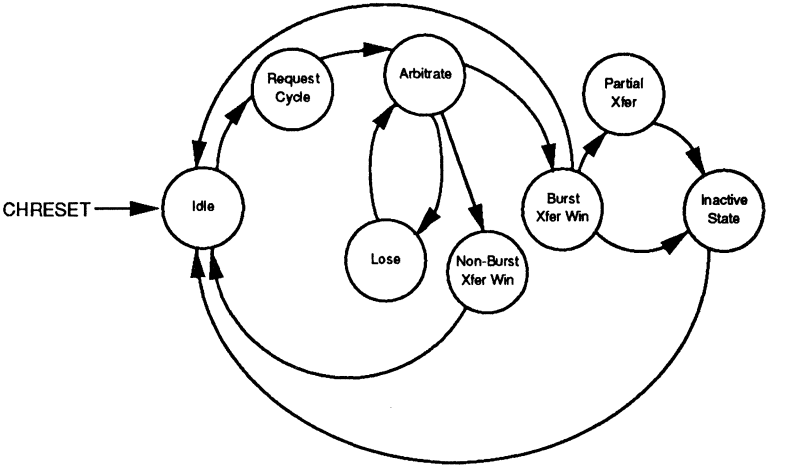
	MC Bus Address			EPB2002A Connection SEL			
REGISTER	A2	A1	A0	-SEL3	-SEL2	SEL1	SEL0
0102H	0	1	0	A2	A0	A1	1
0103H	0	1	1	A2	0	A1	A0
0104H	1	0	0	A1	A0	A2	1
0105H	1	0	1	A1	0	A2	A0

The **-RDPOS** and **-WRPOS** signals provided by the EPB2001 chip act as read and write strobes for the EPB2002A chip. Since the EPB2002A has only one five-bit register, any **-RDPOS** (combined with valid inputs on **SEL0-3**) reads the EPB2002A register to the **D0-4** pins, and any **-WRPOS** signal with appropriate **SEL** inputs writes the EPB2002A register. **-ADL** latches the **SEL** inputs (as it does with addresses on the EPB2001).

Arbitration Logic

If several adapters are vying for control of the MC Bus, the system's Central Arbitration Control Point (CACP) must decide which adapter will be given control first. This process is called arbitration. See Figure 4. Concepts necessary for understanding arbitration logic are explained here.

Figure 4. EPB2002A Arbitration Logic



Arbitration

Arbitration for the MC Bus occurs during defined periods, centrally coordinated by the CACP. The **ARB/-GNT** line is the system control line that signals the periods during which arbitration may occur. When this line is high, an arbitration cycle is in progress. An adapter requests control of the bus to initiate an arbitration cycle by driving the **-PREEMPT** signal low. When **ARB/-GNT** goes high in response to this signal, all adapters

requesting bus control participate in the bus arbitration process (with one exception, as discussed below under "Fairness").

Arbitration Level

Arbitration levels are the means by which the system configures the priority of a given adapter's DMA requests. The 4 bits in the EPB2002A's registers— defined by Micro Channel specifications—allow 16 different arbitration levels or priorities. The lower the number, the higher the priority; so 0000 would have highest priority and 1111 would have lowest. During arbitration, the adapter assigned the highest priority (i.e., the one with the lowest arbitration level) will win the next bus cycle.

The arbitration process is straightforward. When **ARB/-GNT** goes high, all adapters requesting control of the bus place their arbitration levels on the **ARB0-3** lines (all adapter outputs are wire-ORed together). If an adapter detects a value lower than its own arbitration level on the **ARB0-3** lines, it realizes that a higher-priority adapter is requesting the bus. As a result, it withdraws the low-order bits of its arbitration level, allowing the arbitration level of the highest-priority requestor to appear on the bus after some settling time. When **ARB/-GNT** goes low, the highest-priority adapter takes control of the bus (for one cycle only in the case of non-burst mode). Adapters that lose the arbitration cycle continue to assert **-PREEMPT** until their requests are satisfied.

Should **ARB/-GNT** go to the arbitrate state unexpectedly, the adapters are expected to reenter arbitration immediately, even if control of the next bus cycle has already been granted.

The EPB2002A supports this arbitration process. If the **-BUSREQ** line is asserted, the EPB2002A asserts **-PREEMPT** and arbitrates for the next cycle, using the arbitration level programmed into its register. When the bus is granted, **BUSGNT** is asserted until **-BUSREQ** becomes inactive high, thus signalling end of the transfer.

Burst

The burst operation allows an adapter to hold the bus for multiple contiguous cycles, thus increasing transfer efficiency by eliminating the need for arbitrating every single bus cycle.

A device enters a burst transfer by arbitrating for the bus and then asserting **-BURST** when the bus is granted. The **-BURST** line is asserted as long as the burst transfer lasts. The CACP does not honor a **-PREEMPT** bus request while **-BURST** is low, therefore suspending arbitration cycles. To prevent an adapter from wresting control away from other adapters, the adapter in control of the bus monitors the common **-PREEMPT** line. If **-PREEMPT** is asserted during its burst transfer, the adapter must complete its transfer within a certain time and then release **-BURST**. Once **-BURST**

is released, the CACP is free to run a new arbitration cycle. During this arbitration cycle, the adapter releasing the bus does not participate.

EPB2002A supports burst operation with a **-BRSTREQ** input. If **-BRSTREQ** is asserted, the adapter requests a bus cycle, and when granted the request, **-BURST** is asserted. **-BRSTREQ** is asserted as long as the burst transfer lasts. If the burst transfer is preempted, **-BUSGNT** is deasserted immediately. When **-BRSTREQ** is deasserted, **-BURST** is deasserted. The EPB2002A arbiter does not arbitrate again until **-PREEMPT** goes high. Once it goes high, a **-BRSTREQ** input initiates a new arbitration cycle. In burst-transfer mode, the adapter must run at least one cycle after it gains control of the bus. ***-BUSREQ** should not be asserted during requests for burst-mode transfers.*

Fairness

The obvious drawback to the burst process is that an adapter may seize ("hog") the bus and prevent other adapters from gaining control of it. Therefore, MC Bus specifications allow the following fairness mechanism: if a burst transfer is preempted, the "bumped" adapter must wait until all pending arbitration requests have been satisfied (signaled by **-PREEMPT** going inactive high) before it can re-enter arbitration. The adapters waiting for **-PREEMPT** to go high are said to be in the "hog pen," i.e., inactive.

The fairness mode is used when the fairness bit in the EPB2002A control register is programmed to be 1. However, if this bit is programmed to be 0, the arbitration protocol reflects linear priority, and a bursting adapter that is preempted *may* re-enter arbitration on the next available cycle. Obviously, this process greatly increases the risk of "hogging" the bus. The EPB2002A employs the fairness algorithm by default.

Table 2 summarizes functions of EPB2002A pins. Active-low signals are prefixed with a dash (-).

Design Guidelines

Operation of the EPB2002A with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability. The EPB2002A contains circuitry to protect device pins from high static voltages or electric fields; however, normal precautions should be taken to avoid application of higher than maximum-rated voltage .

For proper operation, input and output pins must be constrained to the range GND < (V_{IN} or V_{OUT}) < V_{CC}. Unused inputs must always be tied to an appropriate logic level (either V_{CC} or GND). Each set of V_{CC} and GND pins must be connected together directly at the device. Power-supply decoupling capacitors of at least 0.2 µF must be connected between V_{CC} and GND. For the most effective decoupling, each V_{CC} pin should be separately decoupled to GND, directly at the device.

Table 2. EPB2002 Pin Descriptions

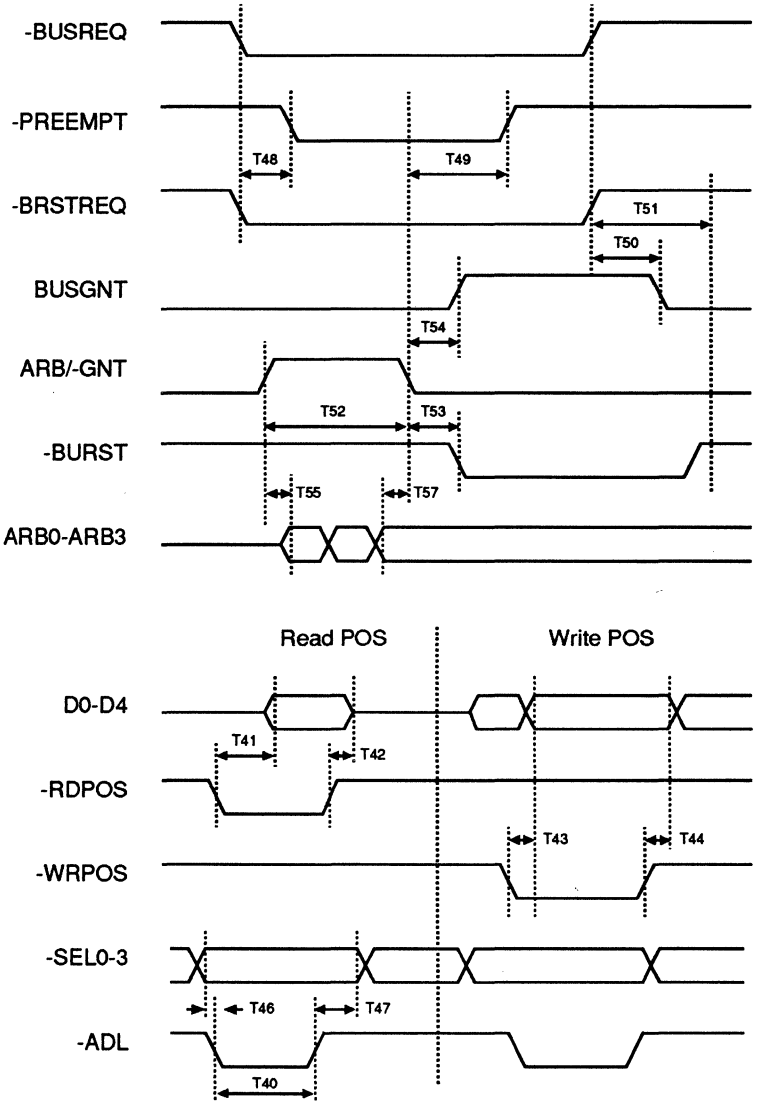
Signal	Type*	Output Drive (mA) I_{OH}	Output Drive (mA) I_{OL}	Description
ARB/–GNT	I			Arbitration cycle strobe input from MC Bus. When high, arbitration occurs. If EPB2002A has pending DMA request of either type, ARB0-ARB3 are enabled during interval to compete for the bus.
–BURST	OD		24	Active-low burst DMA cycle output. Asserted during a burst DMA transfer if the bus is granted as result of active –BRSTREQ. Deasserted on CHRESET.
–PREEMPT	I/OD		24	Bidirectional bus preempt output and input. Driven low by EPB2002A upon –BUSREQ or –BRSTREQ input to request bus arbitration cycle. Released when the bus is granted. Assertion during burst DMA transfer indicates request from another adapter, and transfers. Must be terminated in 7.8 µs. Condition is signalled by deasserting BUSGNT; –BURST remains asserted. Deasserted on CHRESET.
D0-D4 (5x)	TS	4	24	Tri-state bidirectional data bus lines. POS register read/write data access path. Enabled to MC Bus when SEL inputs select device and –RDPOS is asserted. D0-D3 are associated with POS register bits ARB0-ARB3, respectively; D4 with fairness bit.
ARB0-ARB3 (4x)	I/OD		24	Bidirectional arbitration bus lines. Adapter's arbitration level is output initially when ARB/–GNT is high and DMA request is pending. Final bus value matches highest priority request. If value on bus matches adapter's arbitration level when ARB/–GNT returns low, adapter has won the bus.
–ADL	I			Active–low address latch input. Leading edge of this signal is used to latch SEL inputs.
SEL0, SEL1, –SEL2, –SEL3	I			POS register select inputs. SEL0 and SEL1 must be high while –SEL2 and –SEL3 are low to access the POS register. Latched by –ADL on its falling edge.
CHRESET	I			Active-high channel reset input. The EPB2002A deasserts all active outputs a short time after CHRESET rises, and the arbitration state machine returns to idle.
–BRSTREQ	I			Active-low burst DMA transfer request. Unlatched input must remain valid during cycle request and subsequent transfer. Must be deasserted during last DMA bus cycle to terminate bus control.
–BUSREQ	I			Active-low single-transfer DMA request input. Unlatched input must remain valid during the request until BUSGNT is asserted. Must be deasserted following BUSGNT to terminate bus control.
BUSGNT	TP	4	6	Active-high bus grant output. Driven active when bus control is won through arbitration cycle. Stays active until DMA request is removed or –PREEMPT is detected EPB2002A responds to external –PREEMPT by deasserting BUSGNT, but –BURST is asserted until –BRSTREQ is removed. Goes inactive on CHRESET.
–WRPOS	I			Active-low POS register write strobe input. Data is written from D0 D7 to the register when chip is selected and –WRPOS is low.
–RDPOS	I			Active-low POS register read strobe. POS register data is presented on D0-D5 when device is selected and –RDPOS is low.
V_{CC} (2x)				+5 V power supply
V_{SS} (3x)				Ground

* Signal Types:
I = Input TP = Totem-pole (push-pull) output OD = Open-drain output TS = Bidirectional tri-state I/O
I/OD = Bidirectional open-drain output I/O

Latch-Up & ESD Protection

The EPB2002A input, output, and I/O pins are designed to resist electrostatic discharge (ESD) and latch-up damage. Each device pin will withstand voltage energy levels exceeding those specified by MIL-STD-883C. Pins will not latch up for input voltages between -1 V and V_{CC}+ 1 V with currents up to 100 mA. During transitions, the inputs may undershoot to -2.0 V for periods less than 20 ns.

EPB2002A Waveforms



Absolute Maximum Ratings

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND *See Note (1)*	-2.0	7.0	V
V_I	DC input voltage	*See Note (1)*	-2.0	V_{CC} + 1.0	V
I_{MAX}	DC V_{CC} or GND current			+500	mA
I_{OUT}	DC output current, per pin		-50	+50	mA
P_D	Power dissipation			1000	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature		0	+70	° C
t_R	Input rise time			250	ns
t_F	Input fall time			250	ns

DC Operating Characteristics

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High-level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low-level input voltage		-0.3		0.8	V
V_{OH}	High-level TTL output voltage	See Tables above	2.4			V
V_{OL}	Low -level output voltage	See Tables above			0.45	V
I_I	Input leakage current	V_I = V_{CC} or GND	-10		+10	µA
I_{OZ}	Output high-Z leakage current	V_O = V_{CC} or GND	-10		+10	µA
I_{CC}	V_{CC} supply current (standby)	V_I = V_{CC} or GND, No load		8	50	µA
I_{CC}	V_{CC} supply current (active)	V_I = V_{CC} or GND, No load, f = 5 MHz		3	5	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{OUT} = 0 V *See Note (3)*		10	pF
C_{IO}	I/O capacitance			20	pF
C_{OD}	Output capacitance			20	pF

AC Output Capacitance Loadings

Output Pins	Load Capacitance
–BURST, –PREEMPT, ARB0-ARB3	200 pF
D0-D4	240 pF
BUSGNT	50 pF

AC Output Capacitance Loadings

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T40	–ADL width		40			ns
T41	–RDPOS low to D0-D4 valid				50	ns
T42	D0-D4 tri-state from –RDPOS high				20	ns
T43	D0-D4 setup to –WRPOS low		0			ns
T44	D0-D4 hold from –WRPOS high		20			ns
T46	SEL0-1, –SEL2-3 setup to –ADL ow		10			ns
T47	SEL0-1, –SEL2-3 hold from –ADL high		25			ns
T48	–BUSREQ, –BRSTREQ to –PREEMPT low	*See Note (4)*			70	ns
T49	ARB/–GNT low to –PREEMPT tri-state				50	ns
T50	–BUSREQ, –BRSTREQ high to BUSGNT low				40	ns
T51	–BRSTREQ high to –BURST high				30	ns
T52	ARB/–GNT width			300		ns
T53	ARB/–GNT low to –BURST low				50	ns
T54	ARB/–GNT low to BUSGNT high				60	ns
T55	ARB/–GNT high to ARB0-ARB3 driver turn-on delay				50	ns
T57	ARB0-ARB3 setup to ARB/–GNT low		10			ns
T58	CHRESET high to BUSGNT, –BURST, –PREEMPT, ARB0-ARB3 inactive				50	ns

Notes to tables:

(1) Minimum DC input is -0.3 V. During transitions, inputs may undershoot to -2.0 V for periods less than 20 ns.
(2) Typical values are for T_A = 25° C and V_{CC} = 5 V.
(3) Capacitances measured at 25° C. Sample tested only.
(4) If ARB/-GNT is not in ARBITRATE state, i.e., high.

Ordering Information

Part Number	Description
EPB2002A LC	Plastic J-lead chip carrier.
EPB2002A PC	Plastic DIP carrier.

IBM PS/2 Adapter Card

Interfacing with the EPB2001 and EPB2002A

April 1990 **Application Note 14**

Introduction

The IBM PS/2 includes an adapter card bus architecture called the Micro Channel Bus (MC Bus). This architecture, which addresses both 16- and 32-bit processors, supports high-performance multi-device bus arbitration, level-sensitive interrupts, and Programmable Option Select (POS) registers. The POS registers replace DIP switches and jumpers on the adapter card and allow software configuration of card features. In addition, to decrease the overall size of the PS/2 system, the physical size of adapter cards has been reduced by about 40% to only 33 square inches. The net result is higher performance and a more reliable scheme for the attachment of adapter cards to the PS/2. These benefits, however, do not come without a price: the interface is more complex and requires a highly integrated VLSI approach.

This Application Note illustrates the use of Altera's user-configurable adapter interface chips for the PS/2 Micro Channel. It describes a typical application consisting of a multi-function card that includes 32 K words of static CMOS RAM, a general-purpose output port, and two serial ports. The Micro Channel interface EPLDs—EPB2001 and EPB2002A—provide all essential control interfaces between the MC Bus and a PS/2 adapter card. In addition, this Application Note shows how to implement optional adapter card features—such as wait-state generation, output port, and bit-rate prescaler—with an EPM5064 EPLD. Since these highly integrated devices are user-configurable, they can provide functions typically requiring multiple components.

The POS I/O pins from the EPB2001 control typical card-level functions: wait-state duration and Direct Memory Access (DMA) channel selection and enabling; board-control lines (**-DEN**, **DT/-R**, **-IOWR**, **-MEMRD**, etc.) control memory and I/O accesses. The EPB2001's chip-select block provides all address decoding for board functions.

This Application Note not only illustrates the great versatility of the EPB2001 and EPB2002A, but also provides a list of design considerations for the 32-bit system and a set of general design tips for the EPB2001/EPB2002A devices. The designer will learn how to use the programmability of these chips to provide optional, card-specific functions such as latched addresses or card data-size feedback.

This Application Note assumes that the reader is familiar with the basic concepts of the IBM MC Bus. Additional information may be found in the references listed at the end of this document. Comprehensive functional and parametric information on the EPB2001 and EPB2002A is available in the *EPB2001* and *EPB2002A* data sheets.

Multi-Function Card

Figure 1 shows a block diagram of the multi-function card. Shown on the right are the MC Bus lines that run to all adapter card slots. The EPB2001 provides mandatory POS register functions required for any adapter card interface. In addition, board control logic, board ID storage, and address decoding are integrated into the chip. The EPB2002A, shown at the bottom of Figure 1, provides DMA arbitration support. Not all add-on designs require DMA, but when it is needed, the EPB2002A fully supports the IBM bus exchange protocols.

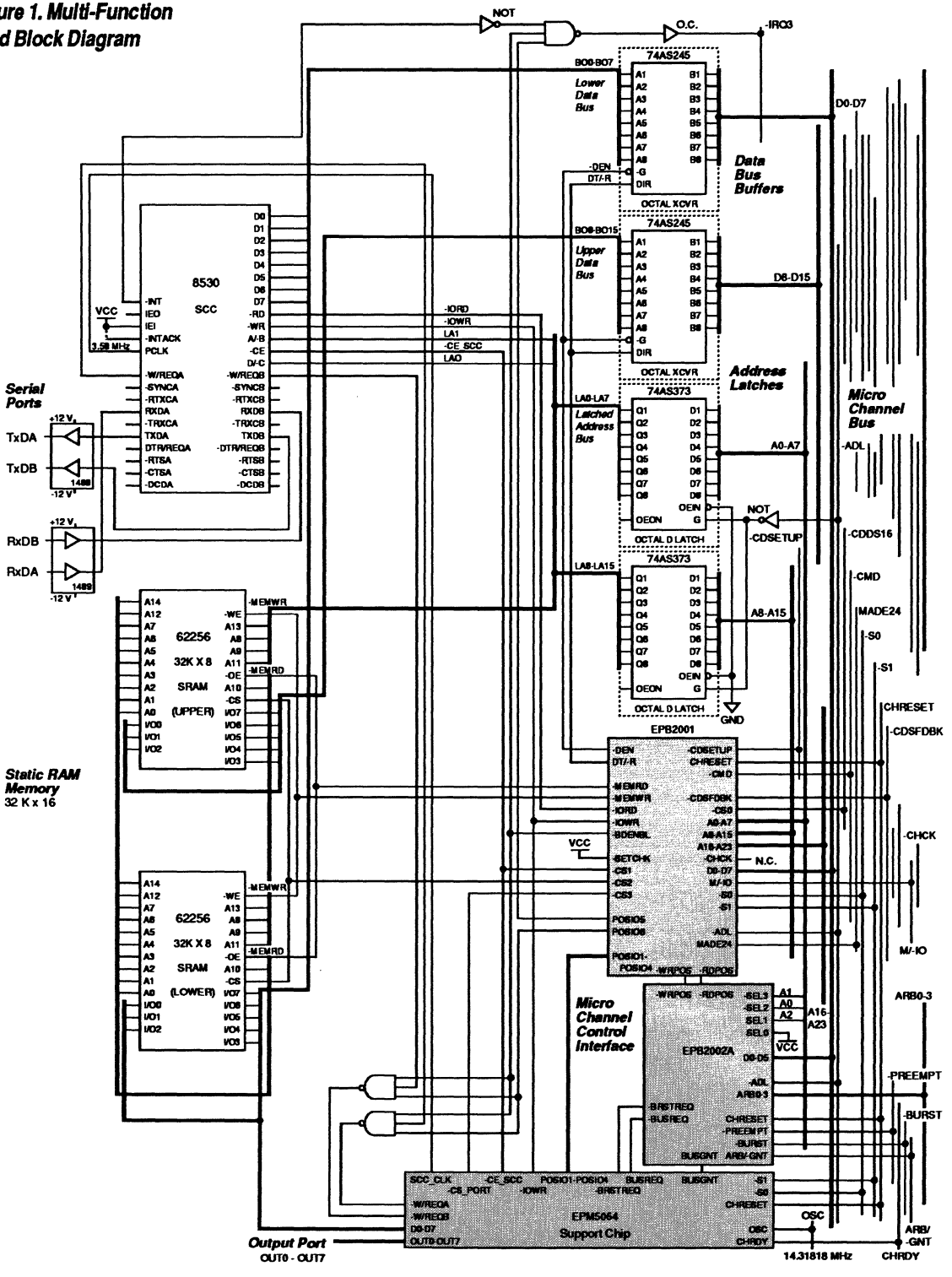
This card design appears to the PS/2 as either an 8-bit I/O peripheral for access to the Z8530 Serial Communications Controller (SCC), or as a 16-bit memory extension (32 K words). The two 74AS245 transceivers, shown at the top right, buffer data transfers between the Micro Channel and I/O or memory. Memory is always accessed as words (no byte read/write), thus eliminating selective enabling of buffers with **-SBHE** from the Micro Channel. Two 74AS373 flow-through latches are used to latch addresses during bus transfers. The **-ADL** line from the MC Bus controls the latching. Only the 16 low-order address bits need to be latched since memory capacity is 64 K (2^{16}). The EPB2001's chip-select logic handles upper-order address decoding for the I/O and memory chips on the board.

The Z8530 SCC provides 2 serial channels in a single 40-pin device. This chip supports a variety of synchronous and asynchronous communication modes, on-chip data buffering, and an integral baud-rate generator. The Z8530 includes handshake lines for DMA request/acknowledge that interface to the EPB2002A. In response to these requests, the EPB2002A asks for permission to use the MC Bus via the **-PREEMPT** line.

The clock for the SCC is provided by a divide-by-four counter, driven by the **OSC** line on the MC Bus. This clock divider is implemented in a small portion of an EPM5064 EPLD. The **OSC** line provides a precision clock input of 14.31818 MHz. The clock divider generates a 3.58-MHz output for the SCC, thus eliminating the need for a local oscillator for the SCC.

Wait-state logic associated with the Z8530 is also designed into the EPM5064. In a Micro Channel default bus cycle, the width of the **-CMD** bus transfer strobe may be as narrow as 90 ns. However, the 4-MHz Z8530 used in this design requires a 250-ns minimum width on **-RD** and **-WR** to the device, so the bus cycle must be "stretched" by at least 160 ns. This stretching is achieved by inserting wait-states to extend the cycle. (This logic is described in detail under "EPM5064 Support Logic.")

Figure 1. Multi-Function Card Block Diagram



The two CMOS SRAM chips used in this design have 100-ns address access times and require no wait-states. This memory is always accessed as 16-bit words. The 32-K block can be used as a general-purpose memory extension. A more elaborate design can use this memory as a transmit/receive buffer if an additional local transfer controller is added to move data between buffer and SCC without external intervention.

The two pairs of serial data lines (**RxD** and **TxD**, channels **A** and **B**) from the SCC are connected to 1488/1489 line driver/receivers that, in turn, form RS-232-compatible interfaces. In this application, the full DCD/CTS/RTS handshake is not shown, although it can be easily added by placing buffers between the RS-232 links and the SCC modem control pins. The +12 V/ -12 V supply required by these drivers is obtained from the Micro Channel edge connector.

EPB2001 Interface

In this multi-function application, the EPB2001 provides the primary control interface. The main functional blocks in the EPB2001 and the specific functions provided for this application include the following items:

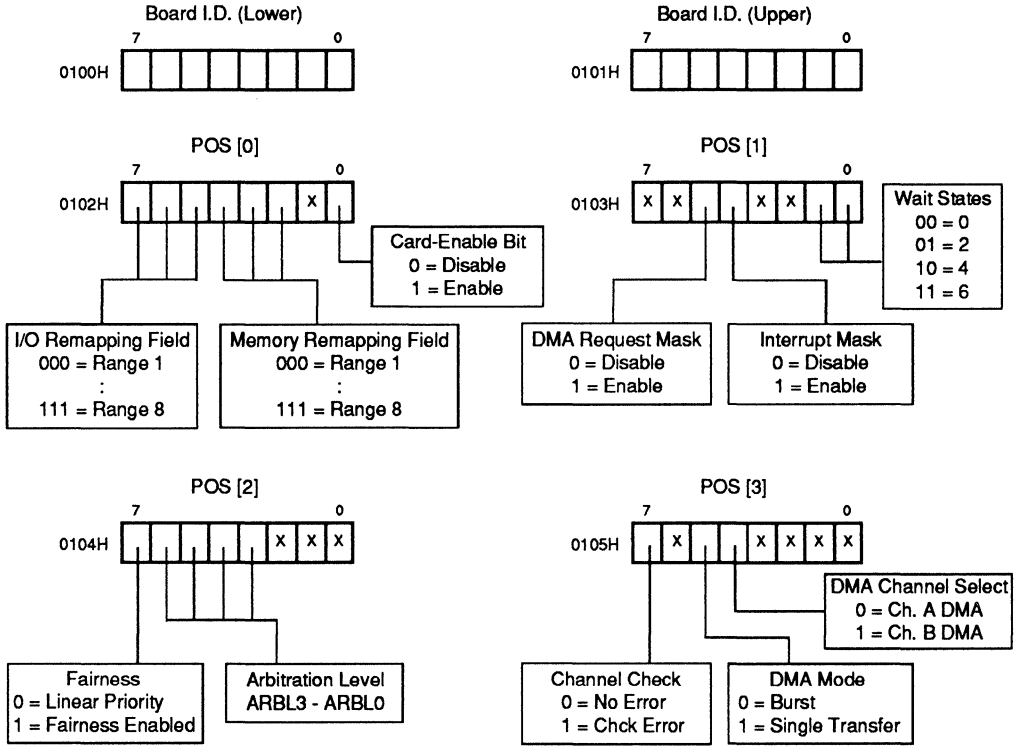
Board ID: The EPB2001 has two CMOS EPROM bytes at locations 0100-0101H for the board ID, which are required for every MC Bus adapter card. These are read-only locations accessible from the MC Bus. An ID is unique to a given board design and is allocated by IBM to registered Independent Developers. The developer must contact IBM directly for such IDs.

POS Registers: This application uses the four required POS registers, 0102-0105H, to control I/O remapping, memory remapping, number of I/O wait-states, DMA request source selection and masking, interrupt masking, and DMA arbitration level and fairness. The DMA arbitration level and fairness control functions are also mapped into the satellite POS register bits on the EPB2002A. Figure 2 shows the bit-mapping of these functions. POS bits that are required outside the EPB2001 are brought out via the POS I/O pins. This mapping is also shown in Figure 2.

Board Control: The board-control logic on the EPB2001 provides all required transceiver control (**-DEN**, **DT/-R**), as well as I/O and memory control strobes (**-IOWR**,**-IORD**,**-MEMWR**,**-MEMRD**). These are generated as bus cycle status decodes that are timed by the MC Bus **-CMD** transfer strobe. These signals directly drive the Z8530, SRAM, and EPM5064 read and write strobes.

In this application, the board enable (**-BDENBL**) output of the EPB2001, which reflects POS register 0102H bit 0, is used to mask DMA requests and interrupts before enabling the card at system power-up. In addition, this bit is internally factored into the **-CSx** enable functions to disable chip-selects when the board is disabled and to suppress generation of the **-CDSFDBK** line before card setup.

Figure 2. Multi-Function POS Register Control Functions



POS I/O Connections

Register	Bit	Pin
0103	0	POSI/01
0103	1	POSI/02
0105	5	POSI/03
0105	4	POSI/04
0103	4	POSI/05
0103	5	POSI/06

Chip-Select Logic: The EPB2001 chip-select block provides chip-select outputs for the CMOS SRAM, SCC, and EPM5064 output port. Also, one of the **-CSx** outputs drives the **-CDDS16** (card data size 16) output to the MC Bus whenever the SRAM (organized as words) is accessed. In this example, the first three chip-select outputs are latched by **-ADL**; the **-CDDS16** output is not latched.

The **-CDSFDBK** (card-select feedback) line to the MC Bus is an unlatched OR function of appropriate chip selects. All **-CSx** functions with the exception of **-CDDS16** are factored into this signal. This selective ORing is a user-programmable feature of the EPB2001 architecture. Latching of these lines is also programmable. Remaining **-CSx** outputs can be used to drive other peripheral or memory chips, or to supply design-specific board logic as described later.

Each of the chip selects has eight pre-programmed ranges controlled by the address remapping fields specified in the POS registers. The programmable POS chip-select enable decoder provides a unique mechanism for linking the POS registers to the chip-select decode block. This mechanism allows the PS/2 to control address response ranges for the card as well as associated memory and I/O resources, as it configures the system. This feature eliminates address conflicts between cards without the error-prone mechanical setting of DIP switches or jumpers. In fact, use of such mechanical arrangements violates the Micro Channel specification.

The structure of the chip-select block as defined in the *EPB2001* and *EPB2002A* data sheets requires that an address block of 2^N locations sit on a 2^N address boundary. In other words, as the block size increases, the allowable number of base address positions decreases. For example, a 256-Kbyte RAM chip-select must have a base address that is a multiple of 256 Kbytes. This restriction is not too severe in most practical applications, but should be kept in mind as alternate address locations are selected.

EPB2002A Interface

The EPB2002A, which does not require as much programmability as the EPB2001, implements the bus arbitration protocol defined for the Micro Channel in an asynchronous state machine. POS register bits, which can be read and written from the MC Bus, are included. These bits control the four-bit arbitration level for the card (arbitration priority) and enable fairness. (Fairness is a feature specified by IBM that eliminates bus "hogging" by bursting DMA devices. Both arbitration and fairness are discussed in detail in the *EPB2002A* data sheet and in the IBM documentation listed at the end of this document.) The EPB2002A provides direct AC and DC compatibility with the interface signals required.

The designer determines the bit positions and POS register location to which the bits are mapped. MC Bus specifications do not define required POS locations. Any of POS registers 0102-0105H may be used for this purpose. The EPB2002A allows bits to be remapped by appropriate connections to the **SELx** inputs on the chip (see the *EPB2002A* data sheet). In this application, the **SELx** inputs, shown in Figure 1, are connected so the bits are mapped into POS register 0104H, bit locations D3-D7 on the bus.

The electrical interface of the EPB2002A to the EPM5064 and SCC components is described in "EPM5064 Support Logic."

Z8530 Interface

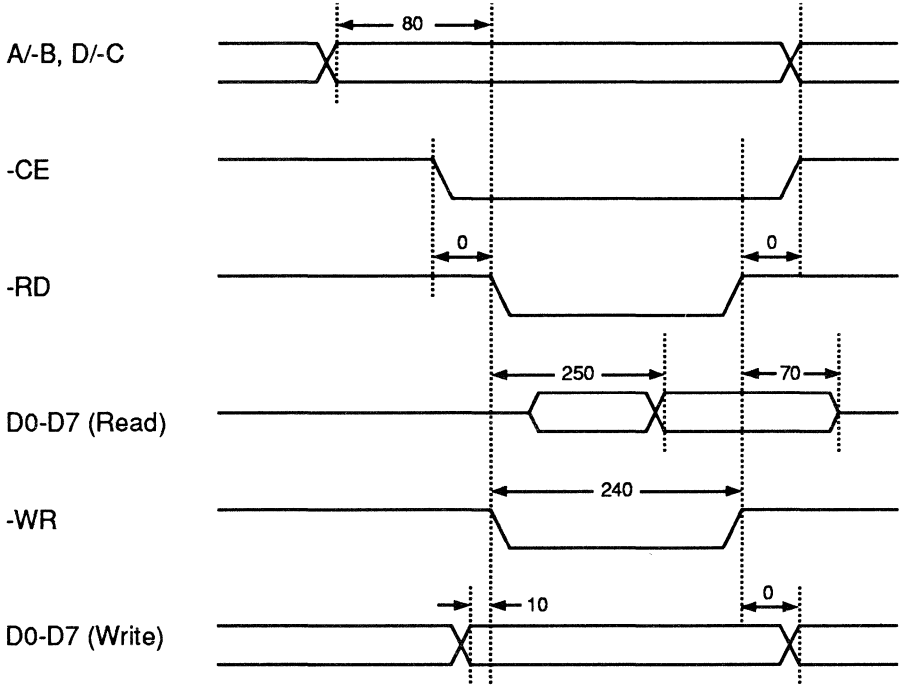
The Z8530 interface consists of the following parts:

- System Interface
- Interrupt Interface
- DMA Interface
- Serial Interface

System Interface

Signals for interfacing the Z8530 to the MC Bus are shown in Figure 3. They include **-CE** (chip enable), **A/-B** (channel A/B designator), **D/-C** (data/command designator), **-RD** (read strobe), **-WR** (write strobe), and the 8-bit data bus **D0-D7**. (The IBM MC Bus specification convention uses a leading hyphen to indicate an active-low signal name.) The Z8530 includes 9 readable and 15 writable registers to set device configuration, report status, and access data. To reduce address input pin requirements, the Z8530 uses an address pointer register: any command register access (read or write) must be preceded by a write to the SCC with **D/-C** low and the address of the register that is accessed on the next cycle as the associated data. After the next access, the pointer register is automatically reset to the address pointer register. Every command register access is thus a two-bus-cycle process: (1) write address pointer, (2) write or read selected register.

Figure 3. Z8530/SCC Timing Requirements (4 MHz)



The waveforms shown are for data transfer cycles. Timing for a command cycle is similar, but requires the additional cycle mentioned above. Timing data is for the 4-MHz Z8530. In general, the timing is straightforward: the Z8530 requires that **-CE**, **D/-C**, and **A/-B** be set up before the strobe (**-RD** or **-WR**) goes active-low. Setup time for **-CE** setup time is 0 ns, while **D/-C** and **A/-B** must be set up 80 ns prior to the strobe edge. All signals have a 0 ns hold time to the rising (inactive) edge of the strobe.

Data for a write cycle must be set up 10 ns before the active **-WR** edge. In a read cycle, the Z8530 places valid data on its **D0-D7** pins within 250 ns of the **-RD** active edge.

Figure 4 shows the signals provided by the EPB2001 chip-select and board-control logic. In this design, the **D/-C** and **A/-B** inputs to the SCC are connected to the latched address inputs provided by the 74AS373 devices. The MC Bus provides a minimum setup of 85 ns from address valid on the MC Bus to **-CMD** falling. Using 74AS373s, the delay from input (MC Bus) to output (latched board address bus) is 5 ns. The address is consequently valid at the Z8530 more than 80 ns before the strobe edge, since the strobe is triggered by the **-CMD** line going active.

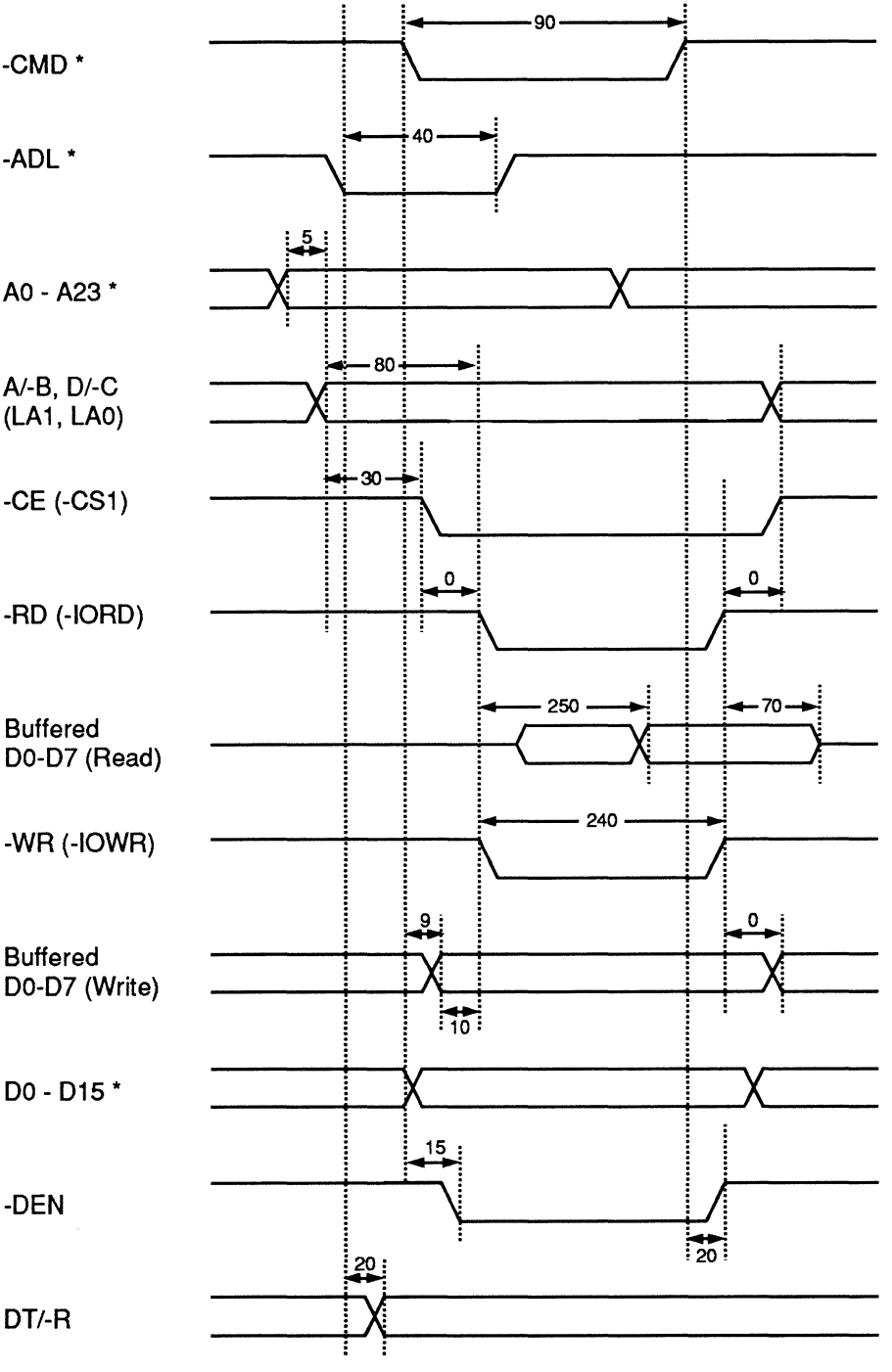
The delay from address valid on the **A0-A23** inputs to the EPB2001 **-CSx** outputs valid is 25 ns. The Z8530 requires only 0 ns; therefore, over 60 ns of timing margin is available on the **-CE** path. The EPB2001 latches the **-CE** line with **-ADL** for the entire bus cycle.

The address ranges that the EPB2001 decodes for the **-CE** input to the SCC consist of blocks of four locations. The EPB2001 allows the designer to specify up to eight such blocks for each chip select. The PS/2 POST (Power-On Self Test) routines can then relocate the multi-function card's resources to eliminate address conflicts with other adapters. Any I/O base address can be selected for the block that resides on a four-byte boundary.

The EPB2001 asserts the **-DEN** line for data transceivers within 20 ns of the **DT/-R** line falling (write cycle to board). This signal falls within 20 ns of **-ADL** going active. The MC Bus specification has a minimum of 40 ns from **-ADL** active to **-CMD** active. **-CMD** triggers **-IOWR**, and **-IOWR** has a delay of up to 20 ns, which tracks parametrically with the **-DEN** delay. As a result, there is a minimum of 20 ns between **-DEN** falling and **-IOWR** falling. The 74AS245s have a 9-ns delay from enable input to outputs valid. Data is therefore present at the SCC inputs 11 ns before the **-IOWR** (**-WR** strobe) input and supports data setup time requirements of the Z8530.

MC Bus specifications require drivers to tri-state within 30 ns of **-CMD** going inactive. The 74AS245's 9-ns tri-state delay, coupled with the 20-ns **-CMD** to **-DEN** delay on the trailing edge, is therefore a good match with the bus specifications.

Figure 4. EPB2001/Z8530 Micro Channel Timing



* Micro Channel Signal

Interrupt Interface

The SCC may be programmed to run in either interrupt-driven or DMA transfer mode. On this particular board, hardware has been provided to support either mode, allowing the board software driver to select the data transfer mechanism. The Z8530 can be programmed to issue an interrupt request each time a byte of data is received, when the transmit buffer is empty, or when one of a number of error conditions occurs. (See the *Z8030/ Z8530 Data Sheet* for further details.)

To support interrupt-driven operation, an open-collector 24-mA driver is used to connect the SCC's **-INT** output to the MC Bus interrupt **-IRQ3**. While the SCC has an open-drain output, its drive is insufficient to handle the MC Bus I_{OL} requirements and must be re-buffered. In this example, several prioritized **-IRQx** lines are available on the MC Bus, and selection of the **-IRQ3** line fixes the SCC's priority at the fourth lowest level in the hierarchy, which is a relatively low level. (It is the alternate serial-port level defined for the PS/2.)

The Z8530 specification requires that the dedicated interrupt acknowledge **-INTACK** become valid 250 ns before **-RD** becomes active. This requirement is not feasible in a single bus cycle and complicates logic design. However, the Z8530 also supports interrupt servicing by providing a register (Read Register 2) that contains an interrupt vector and interrupt reset bits in Write Register 0. Interrupt acknowledge can therefore be treated as standard I/O read and write operations without special timing.

This multi-function card design does not use the channel-check non-maskable interrupt protocol defined for the MC Bus. The **-CHCK** line, like the other interrupt lines on the MC Bus, is a level-sensitive, shared, open-collector arrangement. A channel-check request by any add-on card is wire-ORed onto this line. Adapter logic can activate this line by an active-low pulse on the **-SETCHK** input to the EPB2001, which also resets POS register 0105H, bit 7 (when inactive, it is a logic 1). By interrogating this location on each adapter, the PS/2 channel-check handler can determine which card has issued the channel-check request.

DMA Interface

In this application, the Z8530 handshakes with the EPB2002A when DMA transfers are enabled. Due to constraints in the basic DMA structure of the Micro Channel and PS/2, only one DMA channel per adapter is allowed. As a result, only one SCC channel can issue DMA requests at a time. To make this feature software-configurable, a DMA request source selector has been built into the EPM5064. It has the two **-W/REQx** outputs of the SCC as inputs, and generates either a **-BRSTREQ** or **-BUSREQ** line for the

EPB2002A. The DMA request mode is controlled by a POS register bit as outlined earlier. A POS register bit selects which channel is the DMA requestor. DMA requests can also be permanently disabled by another POS bit for interrupt-driven mode on both channels. Interrupt-driven operation may be used on one channel while DMA operation is employed on the other.

The Z8530 may be programmed to assert **-W/REQx** on a character-receive or transmit-buffer-empty condition. This signal is routed to either the EPB2002A **-BRSTREQ** or **-BUSREQ** input and causes the EPB2002A to assert **-PREEMPT**. Timing for the assertion is not critical.

However, timing for the removal of the request is important. The Z8530 deasserts **-W/REQx** line 240 ns after the corresponding strobe goes active. This signal experiences a 35-ns delay in going through the DMA request selector in the EPM5064 . In the burst transfer mode, the EPB2002A deasserts **-BURST** approximately 50 ns after this signal is deasserted, and at the same time **BUSGNT** is deasserted. To properly terminate the cycle, **-BURST** must be inactive 35 ns before **-CMD** rises. Therefore, in order to guarantee correct burst operation, it is necessary that the strobe width (and therefore **-CMD** active width) be 240 ns + 35 ns + 50 ns + 35 ns = 360 ns.

To guarantee that there is no DMA overrun (i.e., erroneous extra transfers), the DMA transfer cycle must be extended via wait states. Since **-BURST** is not used in single transfer mode, this timing constraint need not be considered. However, for simplicity's sake, single transfer and burst wait-states are the same in this design.

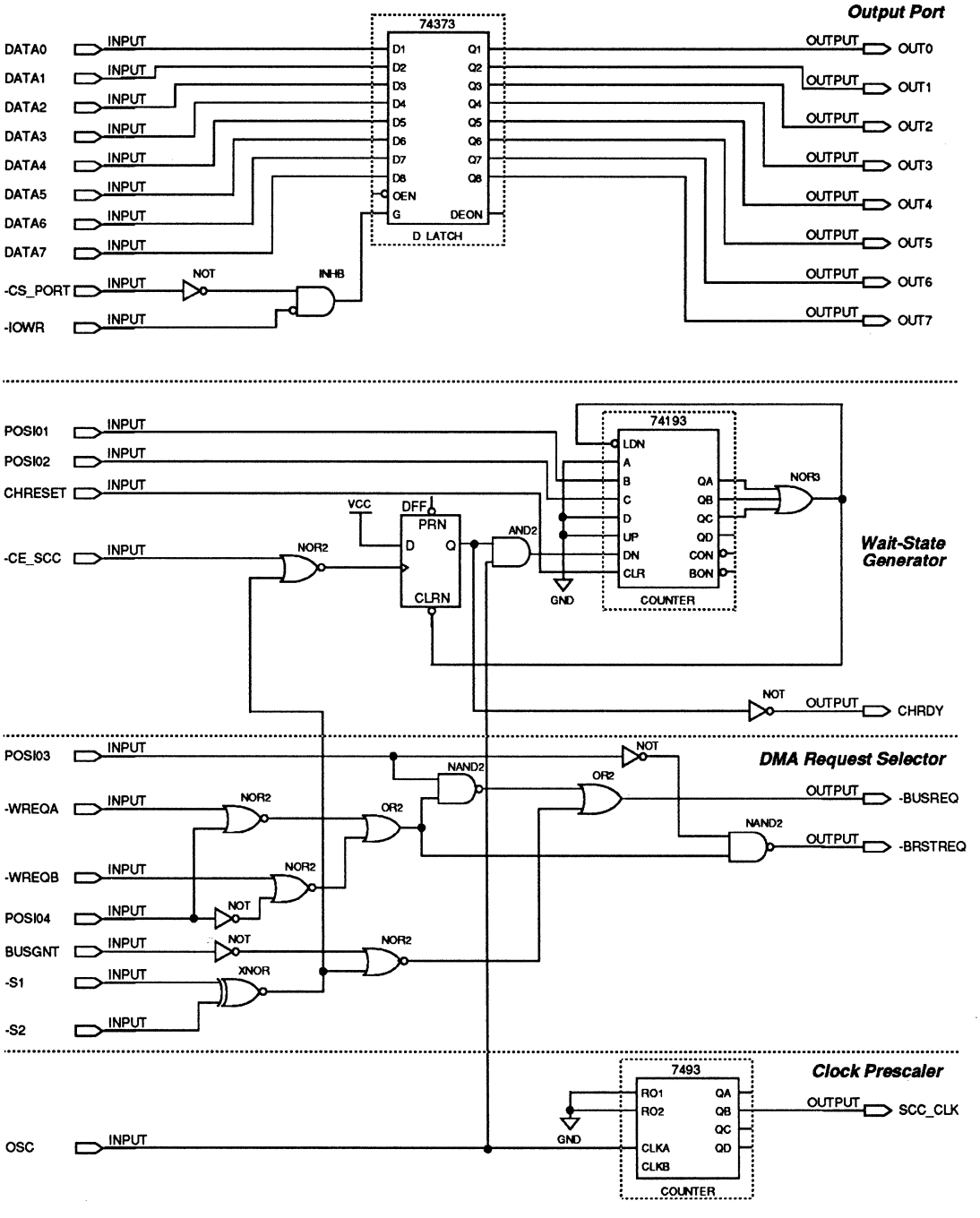
In single-transfer mode, the **-BUSREQ** line must be deasserted on the occurrence of **BUSGNT** active and the **-S0** and **-S1** status lines going to the active state (indicating the start of the DMA transfer). The logic shown in the lower left of the EPM5064 schematic (Figure 5) performs this function.

For burst transfers, the **BUSGNT** signal is not used on the adapter. Since the PS/2's resident DMA channels on the motherboard are being used, **BUSGNT** is not needed to communicate bus ownership. As long as **-BURST** is active, the DMA channels execute transfer cycles. DMA acknowledge for both cases consists of the DMA channel actually accessing (reading/writing) the SCC data buffer. No dedicated **DMACK** lines appear on the MC Bus. In other applications, **BUSGNT** might be used to reset a request transfer flip-flop or indicate to board logic that the bus has been granted.

Serial Interface

Minimal serial interfaces for channels **A** and **B** are shown in this design. Modem control signals (**-RTS**, **-CTS**, **-DCD**) are not used to control data

Figure 5. EPM5064 Support Chip



transfer over the RS-232 links, although this feature can easily be added by programming the SCC for such an operation and adding the appropriate buffers. With the arrangement shown, only the **RxD**, **TxD**, and **GND** lines for each RS-232 line need to be connected.

EPM5064 Support Logic

EPM5064 support logic consists of wait-state logic and a general-purpose output port.

Wait-State Logic

Overall strobe (**-RD** or **-WR**) width required by the SCC is 250 ns. Since the default MC Bus cycle provides only a 90 ns **-CMD** width, the cycle must be extended by a wait-state generator, which, in this case, is designed into an EPM5064 EPLD. The timing source for the wait-state generator, i.e., the **OSC** signal from the MC Bus, lasts about 70 ns. The wait-state generator pulls **CHRDY** on the MC Bus inactive (low) for a number of **OSC** clock edges determined by a value written into a POS register bit field.

The number of wait-states is programmed with parameters in the adapter card's Adapter Description File (ADF). The ADF, required for every adapter card, describes to the PS/2 the available address response ranges and configuration options for a given card. On power-up, the system reads the board ID for each card in its backplane. The board ID is used to find the associated ADF on the system's disk, and the information there is used to configure the adapter and/or to eliminate address range conflicts with other cards.

The EPM5064's 40-MHz operating frequency supports the wait-state generator and clock divider functions with considerable margin given the 14-MHz clock inputs. The number of wait states can be varied from 0 to 6. The minimum number of clock edges to support the 250-ns width required by the 4-MHz SCC is 280 ns (4 x 70). However, if DMA is used as described earlier, 420 ns (6 x 70) must be used. The generator is triggered by an active **-CE** input to the SCC, indicating an Z8530 access to/from the MC Bus.

The actual wait-state generator consists of a loadable down-counter shown in Figure 6. The counter is automatically reloaded each time it reaches 0. During counting, **CHRDY** is held low.

Output Port

As an added feature, a general-purpose output port is constructed in a portion of the EPM5064 EPLD (see Figure 5). Its eight lines can control whatever output functions are required. The output port is entered with Altera's MAX+PLUS Graphic Editor and TTL macrofunctions. The integral Central Arbitration Control Point (CACP) data port on the EPB2001 simplifies the MC Bus interface.

This output port uses another of the EPB2001's **-CSx** outputs during write operations. The **-WS** input to EPM5064 is connected to the **-IOWR** output of the EPB2001. Since the output port is the only EPM5064 resource mapped into the I/O address space, no address inputs are needed.

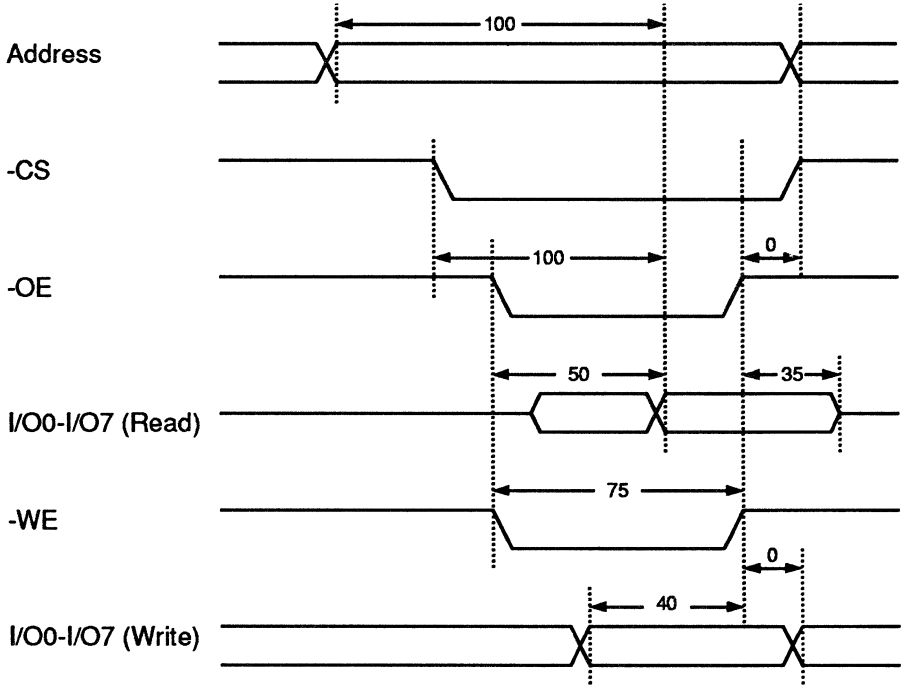
EPM5064 write-control interface timing is simple: a minimum write strobe (**-WS**) low width of 20 ns ensures correct writing of the output port when accompanied by valid data and chip select. To be precise, data must be valid 7 ns prior to **-WS** input falling, and the chip select must be valid 10 ns prior to **-WS** falling. Since MC Bus data is valid at **-CMD**'s leading edge (prior to the leading edge of **-IOWR**), and the EPB2001's **-CSx** lines are valid at least 55 ns prior to **-CMD** falling, the minimum overlaps are easily satisfied. Minimum strobe width of 90 ns is also more than adequate.

Static RAM Interface

The 32 K word static RAM on the multi-function card is implemented with two 32 K x 8 CMOS SRAM devices (62256LP-10 RAMs) in 28-lead, 600-mil DIP packages. They have 15 address inputs (**A0-A14**), 8 bidirectional data lines (**I/O0-I/O7**), a chip select (**-CS**), output enable (**-OE**) and write enable (**-WE**).

Figure 6 shows that access time is 100 ns from address or chip select for these chips. Output-enable delay from **-OE** to valid read data is 50 ns.

Figure 6. 62256LP-10 Timing Requirements



Minimum write-enable width (**-WE**) is 75 ns, with a 40-ns setup time for valid write data to the **-WE** trailing edge.

As shown in the multi-function board schematic (Figure 1), addresses from the 74AS373s drive the SRAM address inputs. SRAM I/Os drive the upper and lower bytes of the board data bus. The **-CE** inputs are driven by a **-CSx** output of the EPB2001, while **-OE** is connected to **-MEMRD** and **-OE** to **-MEMWR**.

As discussed earlier, the MC Bus guarantees a 85-ns setup time of addresses to **-CMD**. A 5-ns delay through the 74AS373s gives greater than 80 ns of the address to **-MEMRD** or **-MEMWR** setup at SRAM devices. Even assuming minimum strobe width of 90 ns (minimum **-CMD** width on the MC Bus), 170 ns of address access time are available at the SRAMs. Chip-select access is reduced by the chip-select decode delay on the EPB2001 (25 ns), but is still 145 ns.

Since the output-enable delay on the SRAMs is 50 ns—even with a 9-ns delay through the 74AS245 transceivers—data is valid on the bus in time to meet the 60-ns MC Bus maximum delay for read data.

The SRAMs require 40 ns of valid write data/**-WE** overlap to guarantee correct writing of the selected locations. The MC Bus specifies 0 ns of write-data setup to the leading edge of **-CMD** and a minimum 90 ns **-CMD** width, resulting in an overlap greater than 80 ns, which is more than adequate even with the 9-ns buffer delay.

EPB2001/ EPB2002A Design Tips

EPB2001 Chip-Select Logic in 32-Bit Systems

The decoding inputs to the EPB2001 are labeled **A0-A23**. They allow full decoding of 24-bit MC Bus addresses down to the byte level. Even though 32 address inputs are not provided, the device is still applicable to 32-bit machines, because the granularity required in chip-select ranges is often quite coarse.

A typical adapter may be memory only, or I/O only, or some mix of memory and I/O. **A0-A2** on the EPB2001 must always be connected to the **A0-A2** pins on the MC Bus to ensure correct access of the POS registers and board ID. If the **A3-A23** inputs to the EPB2001 are connected to the **A11-A31** lines on the MC Bus, the chip-select logic can decode addresses down to block sizes of 2 Kbytes over the full 32-bit address range. Since most RAM chips in use today have a density greater than 2 Kbytes, this number is usually sufficient for memory addressing and memory-only applications.

Mixed memory and I/O adapters can also use this scheme, although it implies that I/O peripherals are placed on 2-Kbyte boundaries, which is

somewhat wasteful of the address space. Only 32 such blocks are available. Further decoding of **A3–A10** external to the EPB2001 can reduce or eliminate this waste.

Pure I/O applications require only **A0–A15**. In these cases, more than enough address inputs are available on the EPB2001 for 16- or 32-bit applications.

Alternative Address Input

A design for an I/O-only adapter or an adapter specifically for 16-bit applications may not require all 24 address inputs and **MADE24**. In such cases, these inputs may be used to provide additional board-specific functions. For example, it may be desirable to generate read or write strobes for specific board ICs. These strobes may be generated as an address decode logically ANDed with a transfer strobe. The **–CMD** signal on the MC Bus typically functions as such a strobe. By connecting an unused **Ax** pin to **–CMD**, it can be factored into appropriate read or write strobes.

Alternative Uses for the EPB2001 Chip-Select Logic Block

The chip-select programmable logic block on the EPB2001 is primarily used to generate adapter chip-select signals. Some other uses for this logic include:

- *Latched Addresses* **–CSx** outputs of the EPB2001 may be used as latched-address outputs for general use on the add-on card. In this case, the programmable chip-select block is programmed to drive the corresponding output low whenever the desired address input is low. The output is latched by **–ADL**. When used for this purpose, the corresponding **–CSx** output is not factored into the **–CDSFDBK** line to the Micro Channel (an EPB2001 programmable feature).

- *Additional POS Register Bit Access* **–CSx** outputs of the EPB2001 may access POS register bits (output only), increasing POS register output pins from 16 to 24 POS I/O lines dedicated for this purpose.

- *Data Size Feedback* **–CSx** outputs have sufficient drive to directly drive the **–CDDS32** and **–CDDS16** lines on the Micro Channel. These signals indicate that a device's data bus width is 32 or 16 bits, respectively. They are derived as unlatched decodes of appropriate address ranges.

  As described in the *EPB2001* and *EPB2002A* data sheets, chip-select address ranges may be enabled by any combination of POS register bits. Typically, a bit field is designated within a POS register to act as

an address relocation control field. It may be desirable to factor certain chip-select enabling functions into special bits such as POS 0102 bit 0 (card enable) or POS 0105 bit 7 (channel check). In this way, chip-selects may be selectively disabled when the card is disabled or while error-recovery (channel-check) routines are being executed.

MCMap Software Support

The MCMap Micro Channel Interface Assembler, available from Altera, provides an easy table-driven entry mechanism for specifying EPB2001/EPB2002A designs. All programmable options are easily specified with this PC-based package. Once design entry is finished, the design is compiled in less than a minute and the resulting JEDEC file may be used to program the EPB2001. The part is programmed with Altera's LogicMap version 6.5 (or a later version) and PC-based LP4, LP5, or LP6 Programming Card, PLE3-12 Master Programming Unit, and PLEJ2001 Programming Adapter. The actual device is programmed and ready for use on the board in seconds.

For additional information about MCMap software and programming hardware, contact Altera Marketing at (408) 984-2805, ext. 101.

Conclusion

The EPB2001 and EPB2002A Micro Channel interface devices provide all required interface functions between an IBM PS/2 add-on board and the system bus. The programmable nature of the EPB2001 provides many possibilities for implementing adapter-specific functions. When other EPLDs such as the EPM5064 are used for unique application features, the user can rapidly design and implement extremely efficient interfaces for PS/2 adapters.

Where to Get More Information

Information on IBM's Independent Developer Assistance Program may be obtained by calling (800) 426-3333. Registration in this program is a prerequisite for obtaining technical assistance and board ID assignments from IBM.

Specifications and technical reference manuals may be obtained from IBM by calling IBM Technical Directory at (800) 426-7282.

Reference

IBM Personal System/2 Model 50,60,80 Micro Channel Architecture. Rev. 5.3. International Business Machines Corporation, 1987.

IBM Personal System/2 Model 50/60/80 Technical Reference. International Business Machines Corporation, 1987.

IBM Personal System/2 Model 80 Technical Reference. International Business Machines Corporation, 1987.

"Z8030/Z8530 Data Sheet (H)." In *MOS Microprocessors and Peripherals Data Book.* Advanced Micro Devices, Inc., 1987.

IBM PS/2 Adapter Card
Software Design

April 1990 **Application Note 15**

Introduction

This Application Note describes software design requirements for an adapter card for the IBM Micro Channel Bus architecture as described in *Application Note 14: PS/2 Adapter Card Interfacing with the EPB2001 and EPB2002A*, and provides configuration and installation information for Micro Channel-based adapter cards. Included also is a utility to interrogate each adapter slot and read back the adapter ID and POS register contents and a description of the Altera multi-function adapter card developed for IBM PS/2 Models 50/60/80. The Adapter Description File (ADF) and a software driver written for this adapter illustrate the software design issues involved. This Application Note, together with *Application Note 14*, provides tested hardware and software design examples that use Altera's user-configurable solutions to speed up and simplify IBM PS/2 add-on card design.

The card is designed with Altera's Micro Channel interface and Direct Memory Access (DMA) arbitration chips, EPB2001 and EPB2002A. These chips handle interface requirements for a PS/2 adapter that range from POS register implementation and address range definition for chip selects to bus arbitration for single-cycle or burst DMA transfers. (See *Application Note 14* for a description, including timing information, of the multi-function adapter card.)

Programmable Option Select (POS)

The Programmable Option Select (POS) feature specified in the MC Bus architecture for IBM PS/2 Models 50/60/80 eliminates configuration switches on adapter cards. POS registers are implemented through programmable registers residing at I/O addresses 0100H through 0107H on each card. POS also permits installation of multiple identical feature cards, positive identification of any card by slot, and resolution of resource assignment conflicts. Each adapter card type must be identifiable by a unique 16-bit ID provided by two read-only registers at addresses 0100H (for the low-order byte) and 0101H (for the high-order byte).

IBM has published the following guidelines for adapter IDs:

0000	Device not ready
0001-0FFF	Bus Master
5000-5FFF	DMA devices
6000-6FFF	Direct program control or memory-mapped I/O
7000-7FFF	Storage or multi-function cards
8000-8FFF	Video

A card developer must register with the IBM Independent Developer Assistance Program (IDAP). Registration in this program is a prerequisite for technical assistance, including information on how to obtain adapter IDs. Information about this program may be obtained by calling (800) 426-3333.

Since the Micro Channel specification is quite complex, the developer should review the IBM ***Technical Reference Manual*** for the specific PS/2 Model (50/60/80). The EPB2001 and EPB2002A greatly simplify the Micro Channel interface portion of an adapter card design and ensure compatibility with the Micro Channel specification.

Adapter Installation

Before installing an adapter card, the user must create an Adapter Description File (ADF) that can be used for the installation.

Adapter Description File (ADF)

After the adapter card hardware design has been completed (based on the POS register bit assignment) the developer must use a text editor to create an ASCII ADF, and place this ADF on a 3 1/2-inch "option diskette." Each adapter must have a separate ADF with the filename **@CARDID** and the extension **.ADF**. For example, an adapter that uses the ID ABCD Hex (the POS register at address 0100H contains the byte CD and the POS register at address 0101H contains the byte AB) must be accompanied by an ADF named **@ABCD.ADF** on a 3 1/2-inch option diskette.

The ADF provides information about POS settings, usage of resources for automatic configuration, input for the System Configuration Utilities (these are packaged with every system on a 3 1/2-inch Reference Diskette), as well as help screens and prompts. The ADF also provides information to the PS/2 configuration utilities about the available address response ranges and configuration options for a given card. When the system is powered up, it reads the board ID for each card in its backplane and compares it to the information stored in its CMOS RAM. When a new adapter is installed, the board ID is used to find the associated ADF on the configuration/option diskette, and the information there is used to configure the adapter and/or to eliminate address-range conflicts with other cards. (The ADF syntax is available in the IBM ***Technical Reference Manual*** for any of the Micro Channel-based models.)

The System Configuration Utilities automatically create configuration data using the ADFs supplied for each adapter card by matching the unique adapter ID with an ADF. Configuration data and the adapter IDs are subsequently stored on battery-backed CMOS RAM.

Creating an ADF

When creating an ADF, the following considerations are important:

- The syntax for creating an ADF is available in the IBM *Technical Reference Manual* for the respective PS/2 model. Syntax rules must be followed strictly since error messages generated by the system during configuration are terse and do not point specifically to syntax rule violations. (For example, if the Help field in a **NamedItem** structure is omitted, a seemingly unrelated divide-overflow exception error is flagged.) The ADF syntax is not case-sensitive.

- The following notation is used to refer to the POS registers in the ADF:

  pos[0] is the POS register located at I/O address 0102H
  pos[1] is the POS register located at I/O address 0103H
  pos[2] is the POS register located at I/O address 0104H
  pos[3] is the POS register located at I/O address 0105H

- Bit 0 in **pos[0]** and bit 7 in **pos[3]** are *reserved* for card-enable and **-CHCK** (channel-check) functions. Therefore, to avoid writing to those two bits, an "X" must be specified in the respective bit positions in the ADF for configuration data for POS registers 0 and 3.

- The **NamedItem** structure specifies a field that provides resource or configuration options. The text within quotes following the keyword **PROMPT** appears in the View/Change Configuration screen when Set Configuration under the System Configuration Utilities is invoked. This text field must contain a brief description of the resource or option being controlled.

- The text within quotes following the keyword **CHOICE** associates the bit pattern in the POS register with one of the resource's configuration choices. This text is then displayed in the Configuration screen. The **[Next]** and **[Previous]** function keys guide the user through the choices to which the resource may be configured. The actual POS bits are therefore transparent to the user. The last portion of the **CHOICE** structure is an optional resource qualifier (e.g., **arb** for arbitration level, **mem** for memory address space, **i/o** for I/O address space). If the resource does not fall into one of the categories specified in the syntax rules, this field may be omitted.

- To customize the configuration, the user invokes the System Configuration Utilities and selects Change Configuration in the Set Configuration menu. If a particular choice conflicts with a previously assigned configuration for the same resource, a * **Conflicts** flag appears on the screen and an asterisk (*) appears next to the field that

caused the conflict. Conflicts can be resolved by selecting a different resource either for the new adapter or for the existing adapter. In each field, the user may press the **<F1>** function key for help.

Installation

This procedure updates the system configuration to support the newly installed adapter card:

1. Make a backup copy of the Reference Diskette (the original diskette is read-only).

2. Re-boot the system with the original Reference Diskette in drive A. The System Configuration Utilities screen is displayed.

3. Press **<Enter>** to select Set Configuration.

4. Select the Copy an Option Diskette utility.

5. When prompted for the new option diskette, insert the backup or the option diskette that contains the ADF for the new adapter card in drive A. (This ADF for the adapter being installed may either be on a separate option diskette or on the back-up diskette.) The new adapter configuration information in the ADF or option file is then read and merged with the existing information.

6. Select Back-up Configuration to save the new configuration data on the backup diskette. *This diskette should be used for all future updates to the system configuration.*

7. Remove the backup diskette from drive A and re-boot the computer. The system boots with the updated configuration stored in the battery-backed CMOS RAM.

8. To view the updated configuration, insert the backup diskette in drive A. Type **A: sc <Enter>** to select the Set Configuration menu within the System Configuration Utilities. While in this menu, the user may simply view the current configuration, or select another utility to change the configuration.

Power-On Self-Test (POST)

The Power-On Self-Test (POST) process automatically configures the system. POST verifies whether the configuration has changed by reading each adapter ID and comparing it to the values stored in CMOS RAM for that slot. If the configuration has changed, System Configuration Utilities on the Reference Diskette must be run again. A configuration error has

been identified if the number 165 appears on the screen at power-up. For example, a change in configuration occurs if a new card is installed or if a card that was part of the existing configuration has been removed.

If an adapter is turned off (e.g., when an external drive or the terminal is turned off) or is not working properly, error number 162 is displayed. In this case, the user must reboot the system with the back-up Reference Diskette in drive A, and follow the prompts in the System Configuration Utilities to verify that the adapters and devices attached to the adapter cards have power. If cards are moved to different slots, POST flags error 165, and the user must follow the prompts for running Automatic Configuration. POST automatically reconfigures the system and updates the CMOS RAM.

Figure 1 contains the C-language source code for a utility that interrogates each slot in the system. It runs setup cycles to each adapter to read back the contents of I/O addresses 0100H to 0105H. This routine has been successfully used on a PS/2 Model 80 to read back POS register contents of adapters in all the eight slots. It may be further tailored to suit the system under consideration, e.g., registers **pos[4]** and **pos[5]** at addresses 0106H and 0107H, respectively, may be read back to obtain the low and high bytes of the subaddress extension. The number of slots being interrogated may also vary depending on the model. The utility confirms the POS bit settings slot by slot, for the various **NamedItem**(s) in the ADF.

This source file is available on the Altera Bulletin Board Service under the file name **RDPOS.C**.

Figure 1. Utility for Reading Adapter ID and POS Register Contents (Part 1 of 2)

```
/* ***************  (c) 1988 Altera Corp.  ***************** */
/*                                                            */
/* File Name "rdpos.c", available on the Altera Bulletin      */
/* Board Service.                                             */
/*                                                            */
/* This program runs 'setup' cycles for each slot in order    */
/* to obtain setup information such as the CARD I.D. and      */
/* contents of the POS Registers. This involves I/O oper-     */
/* ations to the Channel Position Select Register residing    */
/* at port 0096H. Details available in the IBM Technical      */
/* Reference Manual. Microsoft "C" Ver. 5.0 utilized.         */
/*                                                            */
/* ********************************************************** */

#include <stdio.h>
#include <conio.h>

#define cardidl 0x0100  /* Low byte of cardid read at 0100H       */
#define cardidh 0x0101  /* Upper byte of cardid read at 0101H     */
#define posio0 0x0102   /* I/O address for POS Register 0 = 0102H */
#define posio1 0x0103   /* I/O address for POS Register 1 = 0103H */
#define posio2 0x0104   /* I/O address for POS Register 2 = 0104H */
#define posio3 0x0105   /* I/O address for POS Register 3 = 0105H */
```

Figure 1. Utility for Reading Adapter ID and POS Register Contents (Part 2 of 2)

```
main()
{
  int  data, i;
  int port96, reset;
  int val[10];
  struct adapt {
               int crdidl;
               int crdidh;
               int pos0;
               int pos1;
               int pos2;
               int pos3;
               } adapter[8];
                /* for the eight slots in a Model 80 PS/2 */

   port96 = 0x0096;/*  Channel Position Select Register at 0096H */

  /* val[] contains bit patterns to select the channel position */
  /* where the subsequent setup cycle (low active pulse on     */
  /* -CD SETUP(n) ) may be run.                                 */

   val[7] = 0x0F;  /* Bit pattern 00001111B for Slot position 8 */
   val[6] = 0x0E;
   val[5] = 0x0D;
   val[4] = 0x0C;
   val[3] = 0x0B;
   val[2] = 0x0A;
   val[1] = 0x09;
   val[0] = 0x08;

   reset = 0x00;

   for (i=0 ; i<= 7; i++){
           outp(port96,val[i]); /* Select Channel Position */

           adapter[i].crdidl = inp(cardidl);
           adapter[i].crdidh = inp(cardidh); /* Read info. */
           adapter[i].pos0   = inp(posio0);  /* at I/O     */
           adapter[i].pos1   = inp(posio1);  /* addresses  */
           adapter[i].pos2   = inp(posio2);  /* 0100H to   */
           adapter[i].pos3   = inp(posio3);  /*      0105H */

           printf ("\n\n\n The CARD ID at Slot #  ");
           printf ("%d", i+1);
           printf (" is %X%X \n", adapter[i].crdidh, adapter[i].crdidl);

           printf ("\n POS Register Contents for this Adapter are:");

           printf ("\n POS Reg. ");
           printf ("%02X     %02X", posio0, adapter[i].pos0);

           printf ("\n POS Reg. ");
           printf ("%02X     %02X", posio1, adapter[i].pos1);

           printf ("\n POS Reg. ");
           printf ("%02X     %02X", posio2, adapter[i].pos2);

           printf ("\n POS Reg. ");
           printf ("%02X     %02X", posio3, adapter[i].pos3);


   } /* end for loop */

   /* Reset Channel Position Select Register */

           outp(port96,reset);     /* No op. value = 00000XXXB */

}  /* end main*/
```

Figure 2 contains the output file generated by running the utility on a PS/2 Model 80 with a disk controller adapter card (**CARDID = DFFDH**) in slot 8 and the Altera multi-function card (**CARDID=6789H**) in slot 6. The configuration is as follows: slots 1 - 5 are empty; slot 6 is for the Altera multi-function card; slot 7 is empty; and slot 8 is for the hard disk.

The empty slots read back as FF Hex at the POS I/O addresses.

Figure 2. Utility Output File

```
The CARD ID at Slot #  1 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  2 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  3 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  4 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  5 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  6 is 6789

POS Register Contents for this Adapter are:
POS Reg. 102      75
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      A3

The CARD ID at Slot #  7 is FFFF

POS Register Contents for this Adapter are:
POS Reg. 102      FF
POS Reg. 103      FF
POS Reg. 104      FF
POS Reg. 105      FF

The CARD ID at Slot #  8 is DFFD

POS Register Contents for this Adapter are:
POS Reg. 102      09
POS Reg. 103      F3
POS Reg. 104      FF
POS Reg. 105      FF
```

Adapter Hardware

The block diagram in Figure 3 shows the multi-function card design. The board provides 32 K words x 16 memory, a serial communications chip (AmZ8530), the Altera MC Bus interface, as well as the DMA arbitration chips EPB2001 and EPB2002A. The common MC Bus lines are shown on the right. The EPB2001 provides the POS register functions specified by IBM for any add-on card interface. In addition, card control logic, card ID storage, and address decoding are integrated into the chip. The EPB2002A provides DMA arbitration and fully supports the IBM bus-exchange protocols.

The Altera multi-function adapter card appears to the PS/2 as either an 8-bit I/O peripheral (to access the Z8530 Serial Communications Controller), or as a 16-bit memory extension (32 K words).

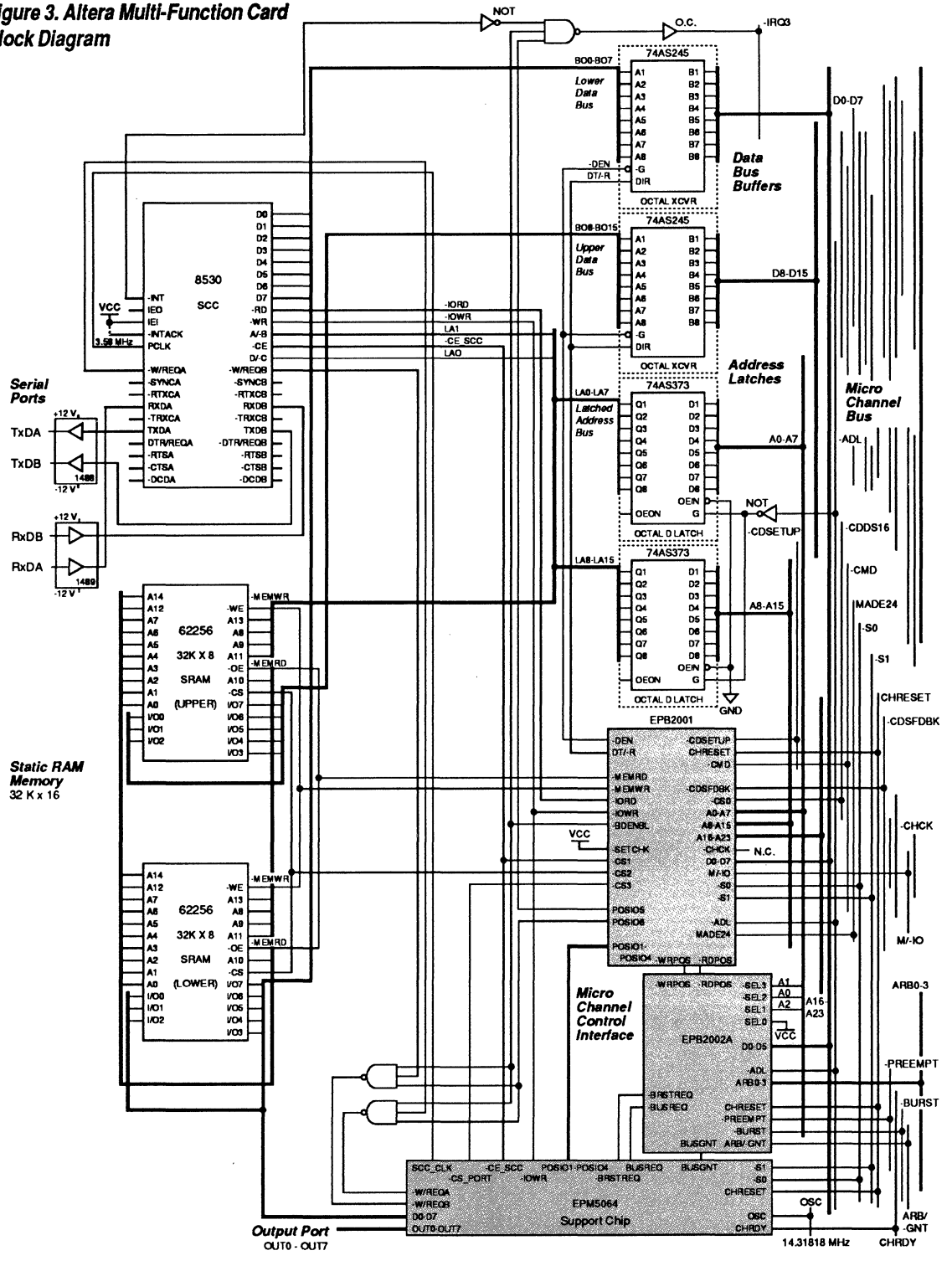
The two 74AS245 transceivers buffer data during transfers between the Micro Channel and I/O or memory. Two 74AS373 flow-through latches latch addresses during bus transfers. The **-ADL** line from the MC Bus controls the latching. Only the 16 low-order address bits need to be latched since memory capacity is 64 K (2^{16}). The EPB2001's chip-select logic handles upper-order address decoding for the various I/O and memory chips on the board. A separate write-only output port designed with an EPM5064 is also available.

The Z8530 Serial Communications Controller (SCC) chip provides 2 serial channels in a single 40-pin device. The chip supports a variety of synchronous and asynchronous communication modes, on-chip data buffering, and an integral baud-rate generator. The Z8530 includes handshake lines for DMA request/acknowledge that interface to the EPB2002A. The EPB2002A requests use of the MC Bus via the **-PREEMPT** line in response to requests from the SCC. The clock for the SCC is provided by a divide-by-four clock derived from the **OSC** line on the MC Bus. The clock divider is implemented in a portion of an EPM5064. The **OSC** line provides a precise 14.31818-MHz clock. The clock divider generates 3.58-MHz output for the SCC.

Wait-state logic associated with the Z8530 is also designed into the same EPM5064. In a Micro Channel default bus cycle, the width of the **-CMD** bus transfer strobe may be as narrow as 90 ns. The 4-MHz Z8530 used in this design requires a 250-ns minimum width on **-RD** and **-WR** to the device. Therefore, wait states must be inserted so that the bus cycle can be "stretched" to at least 160 ns. (This logic is described in detail in *Application Note 14*.) The two CMOS SRAM chips used in this design have 100-ns address access times. Access to these chips requires no wait states.

The two pairs of serial data lines (**RxD** and **TxD**, channels **A** and **B**) from the SCC are connected to 1488/1489 line driver/receivers that form

Figure 3. Altera Multi-Function Card Block Diagram



RS-232-compatible interfaces. In this application, the full DCD/CTS/RTS handshake is not shown. However, it can be implemented by adding buffers between the RS-232 links and the SCC modem control pins. The +12V/-12V supply required by these drivers is obtained from the Micro Channel edge connector.

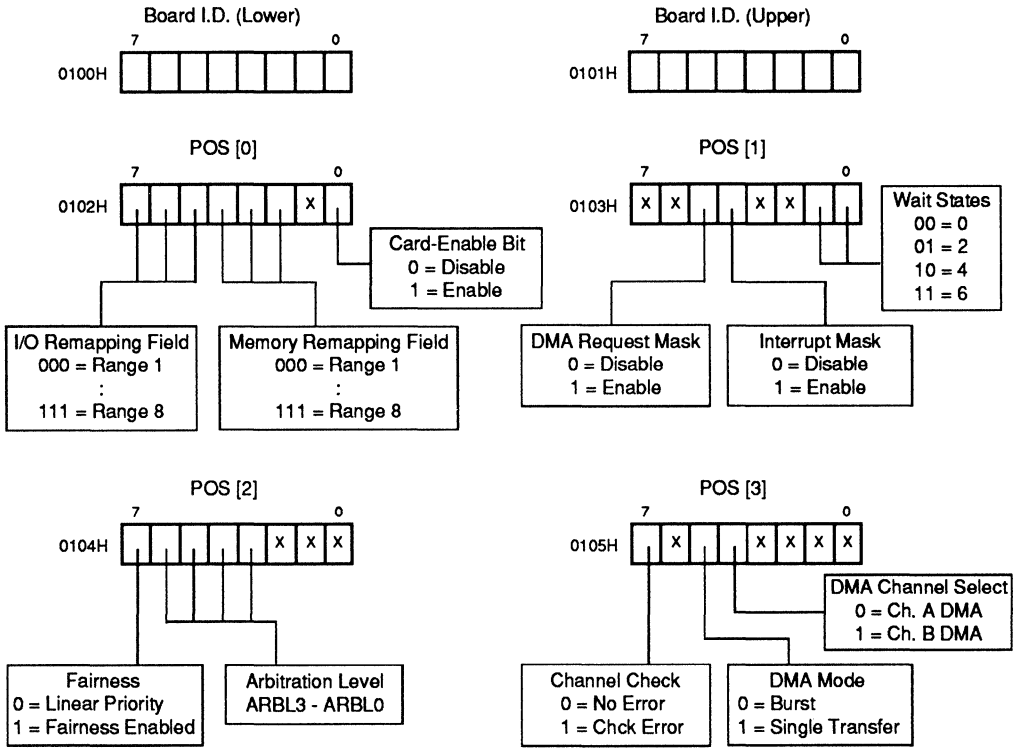
Multi-Function Card POS Functions

In this multi-function application, the EPB2001 provides the primary control interface. As required for any MC Bus add-on card, the EPB2001 needs two CMOS EPROM bytes at board ID locations 0100-0101H that serve as read-only locations accessible from the MC Bus. These IDs are unique to a given board design and are allocated by IBM to registered Independent Developers. Four read/write POS registers located at addresses 0102-0105H are used in this application to control I/O remapping, memory remapping, number of I/O wait-states, DMA request source selection, bus (single-cycle) or burst request, and DMA arbitration level and fairness (fairness is a method devised by IBM to eliminate bus "hogging;" see the *EPB2002A* data sheet for more details). The DMA arbitration level and the fairness bit are also mapped into the satellite POS register bits on the EPB2002A. Bit-mapping of these functions is illustrated in Figure 4. POS bits required outside the EPB2001 are brought out via the POS I/O pins.

Each of the chip selects has eight pre-programmed ranges controlled by the address remapping fields specified in the POS registers. The programmable POS chip-select enable decoder provides a unique mechanism for linking the POS registers to the chip-select decode block so that the PS/2 can control address response ranges for the card as well as associated memory and I/O resources as it configures the system. This feature eliminates address conflicts between cards without requiring mechanical setting of DIP switches or jumpers. The EPB2002A implements the bus arbitration protocol defined for the Micro Channel in an asynchronous state machine. POS register bits that are readable and writable from the MC Bus are included. These bits control the four-bit arbitration level for the board (arbitration priority) and enable fairness.

The board designer determines the bit positions and POS register location to which these bits are mapped. The MC Bus specification does not define required POS locations. Any of POS registers 0102H-0105H may be used, except bit 0 in POS register at 0102H and bit 7 in POS register at 0105H. The EPB2002A allows remapping of these bits by appropriate connections to the **SELx** inputs to the chip. (This function is described in detail in the *EPB2002A* data sheet.) By connecting the **SELx** inputs as shown in Figure 3, these bits are mapped into POS register 0104H, bit locations **D3-D7** on the data bus.

Figure 4. Multi-Function Card POS Register Bit Maps



POS I/O Connections

Register	Bit	Pin
0103	0	POSI/01
0103	1	POSI/02
0105	5	POSI/03
0105	4	POSI/04
0103	4	POSI/05
0103	5	POSI/06

Z8530 Interface

The signals used to interface the Z8530 to the MC Bus are **-CE** (chip enable), **A/-B** (channel A/B designator), **D/-C** (data/control select), **-RD** (read strobe), **-WR** (write strobe), and the 8-bit data bus **D0-D7.** (A leading hyphen indicates an active-low signal name.) The Z8530 includes 9 readable and 15 writable registers to set device configuration, report status, and access data. To reduce address input pin requirements, the Z8530 uses an address pointer register: any command register access (read or write) must

be preceded by a write to the SCC with **D/-C** low and the address of the register to be accessed on the next cycle as the associated data. After the access, the pointer register is automatically reset to the address pointer register. Every command register access is thus a two-bus-cycle process: (1) write address pointer, (2) write or read selected register. The receive and transmit data registers only require a single bus cycle with the **D/-C** input held high.

DMA Interface

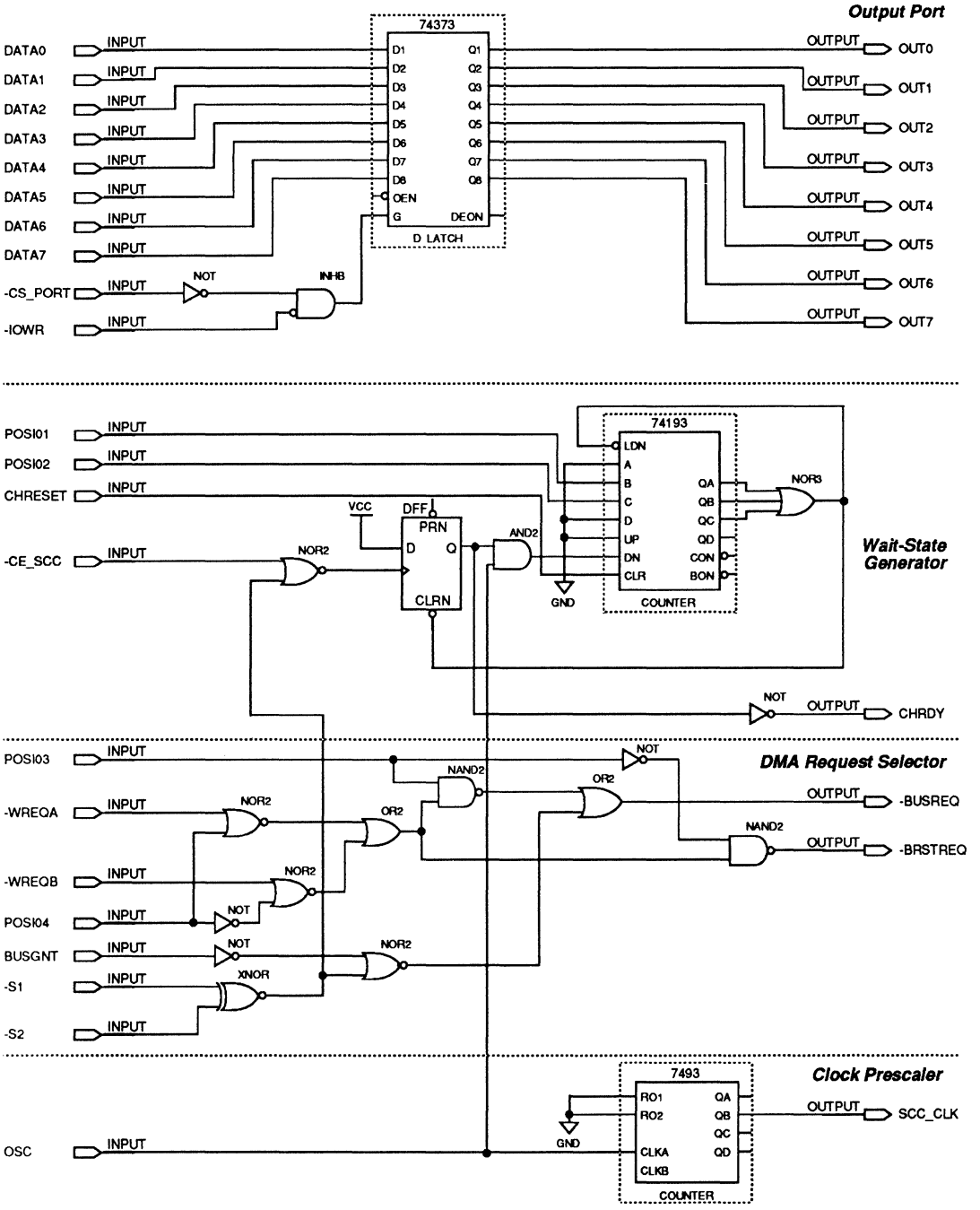
In this application, the Z8530 handshakes with the EPB2002A when DMA transfers are enabled. Constraints in the DMA structure of the Micro Channel and PS/2 allow only one DMA channel per adapter. As a result, only one SCC channel can issue DMA requests at a time. To make this feature software-configurable, a DMA request source selector has been built into the EPM5064. It has the two **-W/REQx** outputs of the SCC as inputs, and generates either a **-BRSTREQ** or **-BUSREQ** line for the EPB2002A. The DMA request mode is controlled by a POS register bit that determines which channel is the DMA requestor. DMA requests can also be permanently disabled by another POS bit for interrupt-driven mode on both channels. Interrupt-driven operation may be used on one channel while DMA operation is employed on the other.

The Z8530 may be programmed to assert **-W/REQx** on a "character receive" or "transmit buffer empty" condition. This signal is routed either to the **-BRSTREQ** or **-BUSREQ** input on the EPB2002A and causes the EPB2002A to assert **-PREEMPT**. Minimal serial interfaces for channels **A** and **B** are shown in this design. Modem-control signals (**-RTS**, **CTS**, and **-DCD**) are not used to control data transfer over the RS-232 links. However, this feature could easily be added by programming the SCC for such operation and adding the appropriate buffers. With the arrangement shown, only the **RxD**, **TxD**, and **GND** lines for each RS-232 line need to be connected.

Output Port (EPM5064)

As an added feature, an output port for general-purpose use has been included in a portion of the EPM5064. These output port lines (shown in Figure 5) may be used to control output functions in software. The port is designed with Altera's MAX+PLUS Graphic Editor and TTL macrofunction elements. The integral system Central Arbitration Control Point (CACP) data port on the EPB2001 simplifies the MC Bus interface. This output port uses another of the EPB2001's **-CSx** outputs that is selected during write operations. The **-WS** input to the EPM5064 is connected to the **-IOWR** output of the EPB2001. Since the output port is the only resource on the EPM5064 mapped into the I/O address space, no address inputs are needed.

Figure 5. EPM5064 Multi-Function Support EPLD Design



Altera Adapter Card ADF

The Adapter Description File (ADF) for the Altera add-on card is shown in Figure 6. The bit assignment indicated in Figure 4 is specified through the **NamedItem** structure in the ADF. POS registers 0, 1, 2, and 3 are implemented in this multi-function interface with **NumBytes=4** in the ADF.

The syntax for generating an ADF is described in the IBM *Technical Reference Manual* for Micro Channel-based PS/2 models. The file **@6789.ADF** for the Altera multi-function adapter card illustrates how the ADF is constructed once the POS register bits are assigned to implement specific control functions. The **CARDID**, which is implemented in the read-only POS registers at addresses 0101H and 0100H, is 6789H.

Figure 6. Multi-Function Card File @6789.ADF (Part 1 of 4)

```
AdapterID 06789h

          AdapterName  "Altera Multi-Function Card"

          NumBytes 4

          NamedItem

                 Prompt  "I/O Remapping"

                    Choice "I/O Range 1"
                      pos[0]=000XXXXXb
                      io 6000h-600Fh
                    Choice "I/O Range 2"
                      pos[0]=001XXXXXb
                      io 6400h-640Fh
                    Choice "I/O Range 3"
                      pos[0]=010XXXXXb
                      io 6800h-680Fh
                    Choice "I/O Range 4"
                      pos[0]=011XXXXXb
                      io 6C00h-6C0Fh
                    Choice "I/O Range 5"
                      pos[0]=100XXXXXb
                      io 7000h-700Fh
                    Choice "I/O Range 6"
                      pos[0]=101XXXXXb
                      io 7400h-740Fh
                    Choice "I/O Range 7"
                      pos[0]=110XXXXXb
                      io 7800h-780Fh
                    Choice "I/O Range 8"
                      pos[0]=111XXXXXb
                      io 7C00h-7C0Fh

          Help
                  "This field assigns an address range in which
                   the SCC registers may be accessed."


          NamedItem

                  Prompt  "Memory Remapping"

                    Choice "Memory Range 1"
                      pos[0]=XXX000XXb
                      mem 200000h-20ffffh
                    Choice "Memory Range 2"
                      pos[0]=XXX001XXb
                      mem 210000h-21ffffh
```

Figure 6. Multi-Function Card File @6789.ADF (Part 2 of 4)

```
            Choice "Memory Range 3"
              pos[0]=XXX010XXb
              mem 220000h-22ffffh
            Choice "Memory Range 4"
              pos[0]=XXX011XXb
              mem 230000h-23ffffh
            Choice "Memory Range 5"
              pos[0]=XXX100XXb
              mem 240000h-24ffffh
            Choice "Memory Range 6"
              pos[0]=XXX101XXb
              mem 250000h-25ffffh
            Choice "Memory Range 7"
              pos[0]=XXX110XXb
              mem 260000h-26ffffh
            Choice "Memory Range 8"
              pos[0]=XXX111XXb
              mem 270000h-27ffffh

Help
        "To configure this adapter you must choose
      the base address for  the memory that the
      adapter will use for buffering data. There are
      eight memory ranges that can be selected.
      Under normal circumstances, select <Segment
      200000>."


NamedItem

          Prompt  "DMA Request Mask"
            Choice  " Disable DMA "
              pos[1]=XX0XXXXXb
            Choice  " Enable DMA "
              pos[1]=XX1XXXXXb

Help
          "This field lets you enable or disable
           DMA requests."


NamedItem

          Prompt  "Interrupt Mask"
            Choice  " Disable Interrupt "
              pos[1]=XXX0XXXXb
            Choice  " Enable Interrupt "
              pos[1]=XXX1XXXXb

Help
          "This bit field lets you enable or disable
           Interrupts."


NamedItem

          Prompt  "Number of Wait States"
            Choice  " 0 Wait States"
              pos[1]=XXXXXX00b
            Choice  " 2 Wait States"
              pos[1]=XXXXXX01b
            Choice  " 4 Wait States"
              pos[1]=XXXXXX10b
            Choice  " 6 Wait States"
              pos[1]=XXXXXX11b
Help
        " This determines the number of wait states
      introduced.
          Under normal circumstances, select <6 Wait
      States>."
```

Figure 6. Multi-Function Card File @6789.ADF (Part 3 of 4)

```
NamedItem
        Prompt "Fairness Option"
           Choice "Fair"
           pos[2]=1XXXXXXXb
           Choice "Linear Priority"
           pos[2]=0XXXXXXXb

Help
        "This bit-fields selects the way in which
         the device competes during arbitration.
         Under normal circumstances, select <Fair>."


NamedItem
        Prompt  "DMA Arbitration Level"
          Choice  "Level 0"
            pos[2]=X0000XXXb
            arb 0
          Choice  "Level 1"
            pos[2]=X0001XXXb
            arb 1
          Choice  "Level 2"
            pos[2]=X0010XXXb
            arb 2
          Choice  "Level 3"
            pos[2]=X0011XXXb
            arb 3
          Choice  "Level 4"
            pos[2]=X0100XXXb
            arb 4
          Choice  "Level 5"
            pos[2]=X0101XXXb
            arb 5
          Choice  "Level 6"
            pos[2]=X0110XXXb
            arb 6
          Choice  "Level 7"
            pos[2]=X0111XXXb
            arb 7
          Choice  "Level 8"
            pos[2]=X1000XXXb
            arb 8
          Choice  "Level 9"
            pos[2]=X1001XXXb
            arb 9
          Choice  "Level 10"
            pos[2]=X1010XXXb
            arb 10
          Choice  "Level 11"
            pos[2]=X1011XXXb
            arb 11
          Choice  "Level 12"
            pos[2]=X1100XXXb
            arb 12
          Choice  "Level 13"
            pos[2]=X1101XXXb
            arb 13
          Choice  "Level 14"
            pos[2]=X1110XXXb
            arb 14

        Help
        "This selects the adapter arbitration level."
```

Figure 6. Multi-Function Card File @6789.ADF (Part 4 of 4)

```
NamedItem
        Prompt  "DMA Mode"
          Choice  "Single Transfer"
            pos[3]=XX1XXXXXb
          Choice  "Burst Transfer"
            pos[3]=XX0XXXXXb
        Help
        "This field simulates the adapter's mode of
         request for the bus."


NamedItem
        Prompt  "SCC Channel Select for DMA"
          Choice  "CHANNEL A "
            pos[3]=XXX0XXXXb
          Choice  "CHANNEL B "
            pos[3]=XXX1XXXXb
        Help
        "Determines which SCC channel requests will
     be honored."
```

Features on the multi-function card—such as number of wait states, SCC channel select (**A** or **B**), mode of DMA request (bus or burst), and the arbitration level—are software-programmable and can be selected in the Change Configuration screen of the System Configuration Utilities. The Change Configuration screen for **@6789.ADF** is shown in Figure 7.

Each time the configuration is changed, System Configuration Utilities must be run as described under "Adapter Installation."

Figure 7. Sample Change Configuration Screen

```
┌──────────────────────────────────────────────────────────────────────┐
│    Change Configuration                                * Conflicts   │
└──────────────────────────────────────────────────────────────────────┘
     Total System Memory
           .
           .

     Built In Features
           .
           .

           .

     Slot1 - Empty

     Slot2 - Empty

     Slot3 - Empty

     Slot4 - Empty

     Slot5 - Empty

     Slot6 - Altera Multi-Function Card
        I/O Remapping ..................... [I/O Range 1]
        Memory Remapping .................. [Memory Range 2]
        DMA Request Mask .................. [Disable DMA]
        Interrupt Mask .................... [Enable Interrupt]
        Number of Wait States ............. [0 Wait States]
        Fairness Option ................... [Fair]
        DMA Arbitration Level ............. [Level 10]
        DMA Mode .......................... [Single Transfer]
        SCC Channel Select for DMA ........ [Channel B]

     Slot7 - Empty

     Slot8 - IBM Fixed Disk Adapter
        Type of first drive ............... [ --]
        Type of second drive .............. [ --]
        Arbitration Level ................. [Level_10]  *
┌──────────────────────────────────────────────────────────────────────┐
│    THIS WINDOW DISPLAYS FUNCTION KEYS FOR CHANGE CONFIGURATION       │
└──────────────────────────────────────────────────────────────────────┘
```

Sample Software Driver

Figure 8 shows the driver for the Altera serial port adapter that has been developed with Microsoft C, version 5.0. The source code is documented here to show how the Z8530 Serial Communications Controller (SCC) and the DMA controller are configured for the application. Detailed technical information about the SCC may be found in the *Z8030/Z8530 Data Sheet*. Information about programming the DMA controller (8237) is available in the IBM *Technical Reference Manual* for the respective PS/2 model.

Specifications for the software driver are as follows:

File name : DMALPB.C (DMA transfer, SCC local loopback). The source code is included and is also available on Altera's Bulletin Board Service.

Function: To provide a software loopback test for the SCC using a PS/2 DMA channel on the motherboard.

Hardware Setup: On the adapter board the **Tx** data pin for SCC channel **A** is connected directly to the **Rx** data pin for channel **B**.

Software Setup:

SCC Channel A: configured for transmission at 1200 baud, 8 data bits, 1 stop bit, and no parity. **Tx** clock is sourced by the on-chip baud rate generator.

SCC Channel B: configured for receiving at 1200 baud, 8 data bits, 1 stop bit, and no parity. **Rx** clock is also sourced by the baud-rate generator. Because of timing constraints, a wait loop has been inserted in the driver between successive read/write operations to the SCC control registers. (See the *Z8030/Z8530 Data Sheet.*)

Using a system call in C (**segread**), the data segment register contents are read and the actual run-time address assigned to an array is calculated. This address in memory is used by the DMA controller for data writes.

System DMA Controller (8237): programmed with the DMA extended mode. This mode allows the DMA controller to be programmed with a function register and I/O addresses 0018H and 001AH, so that additional read/write ports become accessible without increasing I/O space requirements. The controller is programmed for an 8-bit write transfer operation to the address specified in the 3-byte DMA memory address register. The DMA I/O address register is programmed with the SCC channel **B** data register address (6001H).

Figure 8. Software Driver Listing (Part 1 of 8)

```
/*-------------------------------------------------------------*/
/*                                                             */
/*   Ch. A on the SCC is configured to Tx & Channel B to Rx.   */
/*                                                             */
/* This program is set up to receive Data on Ch. B from an     */
/* external source and the block of data stored in an array.   */
/*                                                             */
/*  The DMA controller is then set up to perform memory write*/
/*  transfers, to transfer the received block of data into     */
/*  memory.                                                    */
/*                                                             */
/*-------------------------------------------------------------*/


   #include <stdio.h>
   #include <conio.h>
   #include <dos.h>

   #define wait_count 2
   #define array_size 64
   #define xfer_count 15

   main()

       {

       int scc_addr_chbc = 0x6000L;        /* SCC address for Ch. B control */
       int scc_addr_chbd = 0x6001L;        /* SCC address for Ch. B data    */
       int scc_addr_chac = 0x6002L;        /* SCC address for Ch. A control */
       int scc_addr_chad = 0x6003L;        /* SCC address for Ch. A data    */

       int port_addr = 0x8000L; /* Output Port address */

       int i, data, ndata, dummy_cntr, count, flag ;
       int cntr = 0;
               struct SREGS segregs;

       int array[array_size];    /* This array contains the block of data */
                           /* to be transferred by DMA                  */

       int rx_array[array_size]; /* This array holds data rxed by the SCC */


       unsigned long rx_mem_addr;
       int *rxarray;
       int   rx_mem_addr_0, rx_mem_addr_1, rx_mem_addr_2;

       unsigned int ds;

       int function_reg = 0x0018;
       int exec_fcn_reg = 0x001A;

/* ------------------------------------------------------------------- */
/* ***************** CONFIGURE SCC CHANNEL A TO TX ******************** */
/* ------------------------------------------------------------------- */

/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                          */
                 i= inp(scc_addr_chac);  /* Reset pointer bits to 0 */
/*                                                          */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                          */
                 outp(scc_addr_chac, 9);/*  Point to WR9*/
/*                                                          */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                          */
                 outp(scc_addr_chac, 0xC0); /* Reset both Channels  */

/*                                                          */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                          */
                 i=inp(scc_addr_chac);

/* WRITE  AND READ REGISTER 4 */
/*                                                          */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                          */
                 outp(scc_addr_chac, 4);
/*                                                          */
```

Figure 8. Software Driver Listing (Part 2 of 8)

```
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0x46);/* Set up Register 4*/
                          /* x16 Clock Mode           */
                          /* 1 Stop bit/char          */
                          /* Even Parity, Disabled    */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 4);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                i = inp(scc_addr_chac); /* Read back Status of Register 4= RR0 */
                printf("\n Contents of RR4=RR0  = %X\n", i);

/* WRITE  TO REGISTER 3 */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 3);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0xC0);/* Set up Write Register 3*/

/* WRITE  TO REGISTER 5 - Tx Control */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 5);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0xE4);/* Set up Write Register 5*/

/* WRITE  TO REGISTER 9 - Interrupt Control*/
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 9);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0x17);/* Set up Write Register 9*/

/* WRITE  TO REGISTER 11 - Clock Control */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 11);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0x56);/* Set up Write Register 11.*/

/* WRITE AND READ REGISTER 12 - Baud Rate Genr. Time Constant low byte */

                i = inp(scc_addr_chac);       /* Reset pointer bits */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0x0C);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chac, 0x5B);/ *Write low byte of time const.*/

/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                i = inp(scc_addr_chac);       /* Reset pointer bits */

/* WRITE  REGISTER 13 - Baud Genr. Time Constant high byte */
```

Figure 8. Software Driver Listing (Part 3 of 8)

```
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 0x0D);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 0x00);/*Write upper byte of time const.*/
                 i = inp(scc_addr_chac); /* Reset pointer bits */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 0x0D);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 0x00);/*Rewrite upper byte of time const.*/
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac,0x0D);
/*                                                    */
/* Wait  */ for (i=0; i<=5; i++);
/*                                                    */
                 i = inp(scc_addr_chac); /* Read back Register 13 */

                 printf("\n Contents of RR13 = %X\n", i);

/*   Read Reg. 12                                     */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 0x0C);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 i = inp(scc_addr_chac);   /* Read back Register 12 */

                 printf("\n Contents of RR12 = %X\n", i);

/* WRITE  TO REGISTER 14 - Misc.Control, enable Baud Rate Generator */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 14);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chac, 03);/* Set up Write Register 14.*/

/* ------------------------------------------------------------------ */
/* ******************** Channel A Set-up Complete  ******************** */
/* ------------------------------------------------------------------ */
/* ------------------------------------------------------------------ */
/* ******************** CONFIGURE CHANNEL B TO RX ********************** */
/* ------------------------------------------------------------------ */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 i= inp(scc_addr_chbc); /* Reset pointer bits to 0 */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 9);/* Point to WR9*/
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 0x40); /* Reset Channel B only */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 i=inp(scc_addr_chbc);

/* WRITE REGISTER 4 */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 4);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 0x46);/* Set up Register 4.     */
                           /* x16 Clock Mode          */
                           /* 1 Stop bit/char         */
                           /* Even Parity, Disabled */
```

Figure 8. Software Driver Listing (Part 4 of 8)

```
/* WRITE  TO REGISTER 3 */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 3);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0xC0);/* Set up Write Reg.3    */
                          /* Rx 8 bits/char.          */

/* WRITE  TO REGISTER 1 */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 1);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x60);/* Set up Write Register 1.  */
                          /*  Wait/Req. not enabled    */

/* WRITE  TO REGISTER 9 - Interrupt Control*/
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 9);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x17);/* Set up Write Register 9.*/

/* WRITE  TO REGISTER 11 - Clock Control */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 11);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x56);   /* Rx clock = BRG clk */
                             /* Tx clock = BRG clk */

/* WRITE  TO REGISTER 12 - Baud Rate Genr. Time Const. low byte */

/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                i = inp(scc_addr_chbc); /* Reset pointer bits */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x0C);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x5B);/*Write low byte of time const.*/

                i = inp(scc_addr_chbc); /* Reset pointer bits */

/* WRITE  TO REGISTER 13 - Baud Genr. Time Constant high byte */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x0D);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x00);/*Write upper byte of time const.*/

                i = inp(scc_addr_chbc);      /* Reset pointer bits */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x0D);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x00);/* Re write upper byte of time const.*/
```

Figure 8. Software Driver Listing (Part 5 of 8)

```
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc,12);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                i = inp(scc_addr_chbc); /* Read back Register 12 */

                printf("\n Contents of RR12 Ch. B = %X\n", i);

/*   Read Reg. 13                                     */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc,13);
/*                                                    */
/* Wait  */ for (i=0; i<=5; i++);
/*                                                    */
                i = inp(scc_addr_chbc); /* Read back Register 13 */

                printf("\n Contents of RR13  Ch. B = %X\n", i);
/* WRITE  TO REGISTER 14 - Misc.Control */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 14);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x02);/* Set up Write Register 14.*/

/* WRITE  TO REGISTER 14 - Misc.Control, enable Baud Rate Generator */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 14);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                outp(scc_addr_chbc, 0x03);/* Set up Write Register 14.*/
                        /* Baud Rate Genr. Enabled  */

/* ------------------------------------------------------------------ */
/* ***************** Channel B Set-up Complete ********************** */
/* ------------------------------------------------------------------ */

/* Initialize the tx array with data to be xmitted */

      array[0] = 0xCB;
      array[1] = 0xDD;
      array[2] = 0xCB;
      array[3] = 0xED;
      array[4] = 0xAB;
      array[5] = 0xDE;
      array[6] = 0xEB;
      array[7] = 0xAD;
      array[8] = 0xFE;
      array[9] = 0xCD;
      array[10] = 0xAC;
      array[11] = 0xBD;
      array[12] = 0xDA;
      array[13] = 0xA1;
      array[14] = 0xB0;
      array[15] = 0xBB;

/* Initialize the rx array locations to 0 */

    for (i=0; i<= (xfer_count); i++)
        rx_array[i] = 0;
```

Figure 8. Software Driver Listing (Part 6 of 8)

```
/* ------------------------------------------------------------------ */
/* ***************** Reserve Memory for DMA Write Xfer *************** */
/* ------------------------------------------------------------------ */

    rxarray = &rx_array[0];

 /* Allocate space for array_size integers */

     segread(&segregs); /* Read segment register values */

     ds = segregs.ds;   /* Extract Data Seg. Reg. Contents */
     printf("\n DATA SEGMENT Register contents = %X", ds);
     rx_mem_addr = (((long)ds)<<4) + ((long)rxarray & 0x0ffffL);
     printf("\n rx_array address = %08lX", (long)rxarray);
     printf("\n DATA SEGMENT addr shifted L 4 + array[0] = %lX", rx_mem_addr);


         rx_mem_addr_2 =  ((rx_mem_addr>>16) & 0x0FF) ;
   printf ("\n Memory Address for DMA High byte = %X", rx_mem_addr_2);
        rx_mem_addr_1  = ((rx_mem_addr>>8) & 0x0FF) ;
   printf ("\n Memory Address for DMA Middle byte = %X", rx_mem_addr_1);

         rx_mem_addr_0 = rx_mem_addr & 0x0FF;
   printf ("\n Memory Address for DMA Low byte = %X", rx_mem_addr_0);


    for (i=0; i<= xfer_count; i++)
        rx_array[i] = 0;

/* ------------------------------------------------------------------ */
/* ********** Configure the DMA Channel for Mem. Write Xfer ********* */
/* ------------------------------------------------------------------ */


   outp (function_reg, 0xD4);
                        /* Master Clear */

   outp (function_reg, 0x94); /* Set Mask bit, DMA channel 4 */


   outp (function_reg, 0x84); /* Set up  channel #4 */
   outp (exec_fcn_reg, 0xFA); /* Assign Arb Level 10 */

   outp (function_reg, 0x04); /*Write SCC data port addr. to I/O addr. reg.*/
   outp (exec_fcn_reg, 0x01);
   outp (exec_fcn_reg, 0x60); /* 6001H is the SCC Ch. B data reg. addr.    */

   outp (function_reg,0x24);/* Mem. Address write */

   outp (exec_fcn_reg, rx_mem_addr_0);        /* Lower  Byte of address */
   outp (exec_fcn_reg, rx_mem_addr_1);        /* Middle Byte of address */
   outp (exec_fcn_reg, rx_mem_addr_2);        /* Upper  Byte of address */

   outp (function_reg, 0x44);
   outp (exec_fcn_reg, xfer_count);               /* Transfer Count */
   outp (exec_fcn_reg, 0x00);

   outp (function_reg, 0x54);         /* Read the Transfer Count Reg.*/
   i = inp (exec_fcn_reg);            /* Transfer Count */
   printf("\n Low  byte of the xfer count = %X",i);

   i = inp (exec_fcn_reg);              /* Transfer Count */
   printf("\n High byte of the xfer count = %X",i);

   outp (function_reg, 0x74); /* Extended Mode Reg.  */

   outp (exec_fcn_reg, 0x0D); /*Write, 8-bit Xfer, Programmed I/O addr */


/* Write to Channel A Register 5 to enable  Tx       */
                outp(scc_addr_chac, 5);
/*                                                   */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                   */
                outp(scc_addr_chac, 0xEC);/* Set up Write Register 5.*/

                outp (function_reg, 0xA4); /* Clear DMA Channel mask Bit */
```

Figure 8. Software Driver Listing (Part 7 of 8)

```
/* ************** Write to PORT0  ****************  */
   outp(port_addr, 0x00);

/* Wait 2 us. */ for (i=0; i<=45; i++);

   outp(port_addr, 0x01);   /* Enable the SCC Requests */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 1);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 0xE0);
                      /*Enable DMA Req. on Channel B  */


/* Write to Register 3 Channel B, to enable  Rx       */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc, 3);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 outp(scc_addr_chbc,0xC1); /* Rx enabled             */

/* ------------------------------------------------------------------ */
/* ****************** Monitor the Tx Buffer Status ****************** */
/* ------------------------------------------------------------------ */
    flag = 1; /* Initialize flag */
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */

   for (count=0; count<=xfer_count; count++)
   {
   flag = 1;
         /* reset the flag to 1 for next xfer, "0" implies Tx buffer empty*/

   while (flag)
         {
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
       i = inp(scc_addr_chac);

/* Test  Channel A RR0 bit 2 */

       data = (i & 0x04);
       if (data == 0x04)  /* Bit 2 = 1 => Tx Buffer Empty              */
   {
    outp(scc_addr_chad, array[count]);
    flag=0;
   }
   else
printf ("\n Flag = %d i.e Tx Buffer at count = %d is not empty ", flag,
count);
         } /* end while */


    } /* end of "for count" */
   printf("\n\n\n Number of data bytes received = %d ", (count));


/* ------------------------------------------------------------------ */
/* ***********************Txfer complete ? ********************** */
/* ------------------------------------------------------------------ */
                 again: outp(scc_addr_chac, 0x01);
/*                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                    */
                 i = inp(scc_addr_chac); /* Read back Register 1 */
                 printf("\n Contents of RR1 = %X\n", i);
```

Figure 8. Software Driver Listing (Part 8 of 8)

```
ndata = (i & 0x21);
      printf("\n Masked Contents of RR1 = %X\n", ndata);
if ((ndata == 0x20) | (ndata== 0x00)) {
                goto again;
               }

/* Disable Rx after all data has been sent */

/* Wait 2 us. */ for (i=0; i<=45; i++);
                outp(scc_addr_chbc, 3);
/* Wait 2 us. */ for (i=0; i<=45; i++);
                outp(scc_addr_chbc, 0xC0);/* Disable Rx */

/* ------------------------------------------------------------------ */
/* **** Disable SCC Req., Tx and Set DMA mask bit before leaving *** */
/* ------------------------------------------------------------------ */

                outp(scc_addr_chac, 0x01);
/*                                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                                    */
                outp(scc_addr_chac, 0x40); /* Disable Wait/Req  */

/* WRITE  TO REGISTER 5 to disable  Tx  */
/*                                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                                    */
                outp(scc_addr_chac, 5);
/*                                                                    */
/* Wait 2 us. */ for (i=0; i<=45; i++);
/*                                                                    */
                outp(scc_addr_chac, 0xE4);/* Set up Write Register 5*/

                outp (function_reg, 0x94); /* Set Mask bit, DMA channel 4 */

printf ("\n Disabled Tx, Rx and cleared the mask bit ");

cntr = 0;
for (i=0; i<= (xfer_count); i++){
printf ("\n\n tx_array[%d] = %X ", i, array[i]);
printf ("     rx_array[%d] = %X ", i, rx_array[i]);
                                  }



  }    /* end main*/
```

Conclusion

The EPB2001 and EPB2002A Micro Channel interface devices provide all required interface functions between an IBM PS/2 adapter card and the system bus. The programmable nature of the EPB2001 offers many possibilities for implementing adapter-specific functions. When other EPLDs, such as the EPM5064, are used for unique application features, the user can quickly design and implement extremely efficient interfaces for PS/2 adapters.

Where to Get More Information

The source code for the driver is available under the file name **DMALPB.C** on the Altera Bulletin Board Service. The code is also shown in Figure 8 in this Application Note.

Information about IBM's Independent Developer Assistance Program may be obtained by calling (800) 426-3333. Registration in this program is a prerequisite for obtaining technical assistance and board ID assignments from IBM.

Specifications and technical reference manuals may be obtained from IBM through the IBM Technical Directory at (800) 426-7282.

Reference

IBM Personal System/2 Model 50,60,80 Micro Channel Architecture. Rev. 5.3. International Business Machines Corporation, 1987.

IBM Personal System/2 Model 50/60/80 Technical Reference. International Business Machines Corporation, 1987.

IBM Personal System/2 Model 80 Technical Reference. International Business Machines Corporation, 1987.

"Z8030/Z8530 Data Sheet (H)." In *MOS Microprocessors and Peripherals Data Book.* Advanced Micro Devices, Inc., 1987.

IBM PS/2
Master and Slave Adapter Design

April 1990 **Application Brief 72**

Introduction

"Bus master" and "bus slave" are concepts applicable to a wide variety of microprocessor bus protocols. The IBM Micro Channel Bus (MC Bus) specification, in particular, defines an enhanced bus arbitration mechanism so that multiple bus masters may effectively share the Micro Channel for maximum system throughput. Altera's EPB2002A Direct Memory Access (DMA) arbitration support device for the Micro Channel implements all bus arbitration protocols required for a bus master or bus slave. This Application Brief describes how the EPB2002A may be used as an essential building block in a bus master peripheral adapter for the Micro Channel.

Bus Master vs. Bus Slave

Adapter (i.e., add-on) cards on the PS/2 Micro Channel may be non-DMA bus slaves, DMA bus slaves, or bus masters. Interface requirements for a bus master adapter on the Micro Channel are a superset of those required by a DMA bus slave. It is important, therefore, to define the terms bus master and bus slave before highlighting their specific Micro Channel differences.

Bus Slave: A bus slave is a block of logic that provides data to or receives data from a given bus under the control of a bus master. Figure 1 illustrates a typical bus slave interface. Common bus slaves are add-on memory cards or low- to medium-performance peripheral cards. With a bus slave (whether DMA or non-DMA), address and control lines from the bus are used purely as inputs. A bus slave does not drive these signals and cannot independently transfer data across the bus, but it is always dependent on a bus master for data transfers across the bus.

Bus Master: A bus master can independently transfer data across a bus once it is granted control of the bus. While a system must have at least one bus master, called the system Central Arbitration Control Point (CACP), additional bus masters are possible, depending on the bus architecture. These additional masters may be resident either on the main system motherboard or on a feature add-on card called a bus master adapter. A common example of such a bus master on a PS/2 motherboard is the system DMA controller. A high-performance hard-disk controller might be a bus master storage adapter.

Every bus master must be able to generate addresses for data source and destination pointers. In addition, many bus masters can increment memory addresses automatically as part of a block transfer to or from memory, and provide a byte counter to track the required number of transfers.

Figure 1. Bus Masters and Slaves

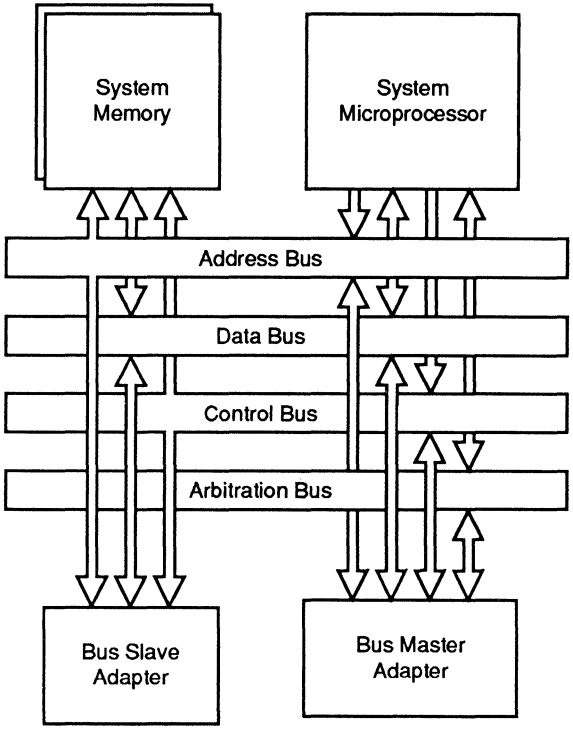


Figure 1 illustrates a typical bus master adapter interface. The address and control lines, unlike those of a bus slave, must function as both inputs and outputs, depending on where bus ownership resides. Additional bus masters are usually initialized by the system CACP for a given operation, and then enabled to complete the task independently. The system CACP writes source and destination address registers, byte counter, and other transfer configuration information into the additional master. During this process, the bus master is used by the CACP as a bus slave. Bus arbitration logic is used to request the bus, resolve priority, and award bus control.

A bus master can interact with a bus slave in a number of ways. In a single bus master system, the CACP can poll status registers in a non-DMA bus slave or respond to its interrupts. Data transfers occur when software instructions are executed. Such transfers are called program I/O in the case of a slave peripheral device. DMA bus slaves issue DMA request signals to the bus master to coordinate data transfers. However, the bus master ultimately retains control of the system. Data transfers are efficiently interleaved with normal CACP instruction-driven bus cycles.

Multiple bus masters offer two benefits: (1) they provide increased data transfer efficiency and thus higher performance because bus-master cards do not have to wait for service from another master, and can initiate and complete transfers more quickly; (2) they allow bus cycles to be "stolen" without disturbing the CACP's program flow, thus decoupling I/O or other operations from the CACP's primary tasks.

To support multiple bus masters, a bus arbitration protocol must be able to pass control unambiguously among different bus masters. Most bus masters use the bus infrequently, but require quick response when the bus *is* requested. For example, a disk-controller bus master typically transfers a block of data across the main bus in microseconds by seeking, reading, and buffering multiple bytes from the disk. Therefore, the bus arbitration scheme must be able to prioritize different bus-master requests to handle simultaneous requests for use of the bus and avoid stacking.

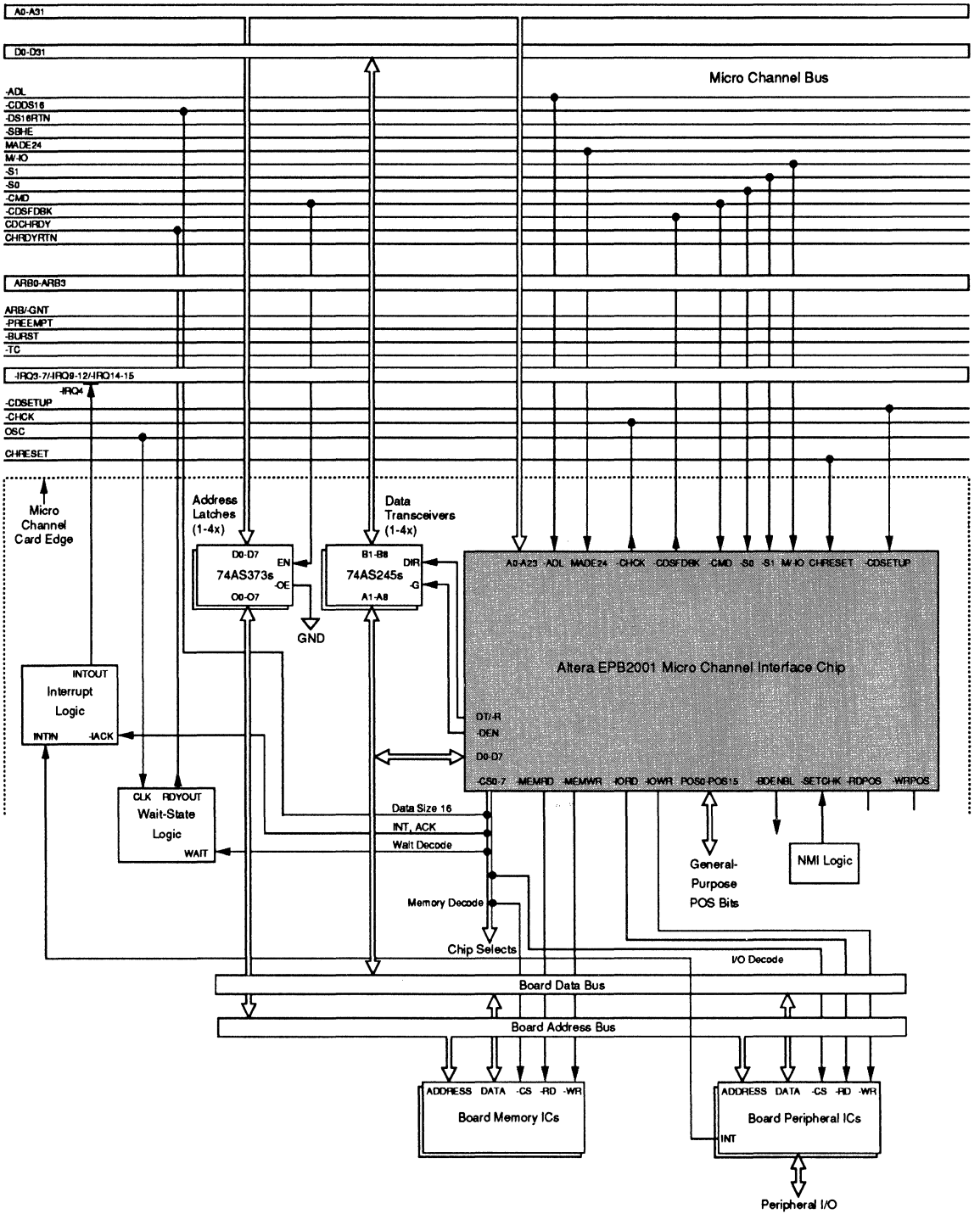
A bus master such as a motherboard DMA controller can execute data transfers at the request of a DMA bus slave adapter card. A DMA controller can, in fact, service multiple slave cards by assigning each to a DMA channel. Each DMA channel, in turn, is assigned independent source and destination pointers, configuration information, as well as an arbitration priority. To coordinate the operation, the bus slave must alert the DMA controller that it is ready for a data transfer. This task is typically accomplished by routing a DMA request signal from slave to DMA controller, or by having the slave request use of the bus, as is done in the Micro Channel via the common **-PREEMPT** line. (See also "Micro Channel Master Interface.")

PS/2 Non-DMA Slave Interface

Figure 2 shows a typical Micro Channel non-DMA bus slave interface. The peripheral function shown on the adapter card may be a communications, graphics, storage, or other type of controller, or it may encompass a mix of functions.

The EPB2001—in this instance used without the EPB2002A—provides the control interface between Micro Channel and adapter logic, including generation of control strobes and chip-select signals. The board ID is stored in the EPB2001, along with Programmable Option Select (POS) configuration information. (For information on POS, refer to the Altera *EPB2001* and *EPB2002A* data sheets.) The 74AS245s buffer the board databus from the Micro Channel, while the 74AS373s latch addresses from the MC Bus. If only a few latched addresses are needed, as is often the case with an I/O-only card, two or three of the EPB2001's **CS** outputs can be used (refer to *Application Note 14*). The latched addresses, together with the **CS** outputs assigned to the memory and peripheral controller(s), allow selection of specific command, status registers, or memory locations. Consequently, the CACP can configure and interrogate the controller and access memory as needed.

Figure 2. Non-DMA Bus Slave Adapter Interface



Also shown in the diagram is wait-state logic, which provides a delayed **CHRDY** signal to the MC Bus for bus cycles that use the peripheral controller. Typically, this logic is implemented in a general-purpose EPLD as a counter triggered by the appropriate **CS** signal(s). By delaying **CHRDY**, the bus cycle time can be lengthened to accommodate the longer-than-minimum access time required by the peripheral's registers.

The interface between the CACP and bus slave occurs via the interrupt line **-IRQ4**. When the bus slave requires a data transfer, it interrupts the CACP, which then moves the data as part of its interrupt service routine. Although this process is relatively inefficient due to interrupt latency, it presents no problem for this low-data-rate peripheral.

The Non-Maskable Interrupt (NMI) logic interfaces to the EPB2001 and any error-generating logic blocks on the board. Typical NMI conditions are memory parity-error or power-fail conditions. The nature of the logic is unique within a given design and best implemented in a general-purpose EPLD. By pulsing **-SETCHK** low, this logic asserts a channel-check condition to the MC Bus via the EPB2001. If this NMI capability is not used, the NMI block can be eliminated, and the **-SETCHK** input to the EPB2001 can be permanently tied high.

PS/2 DMA Slave Interface

Figure 3 shows a Micro Channel DMA bus-slave interface using the EPB2001 and EPB2002A. The PS/2 motherboard DMA controller performs data transfers. Three features have been added to the slave adapter:

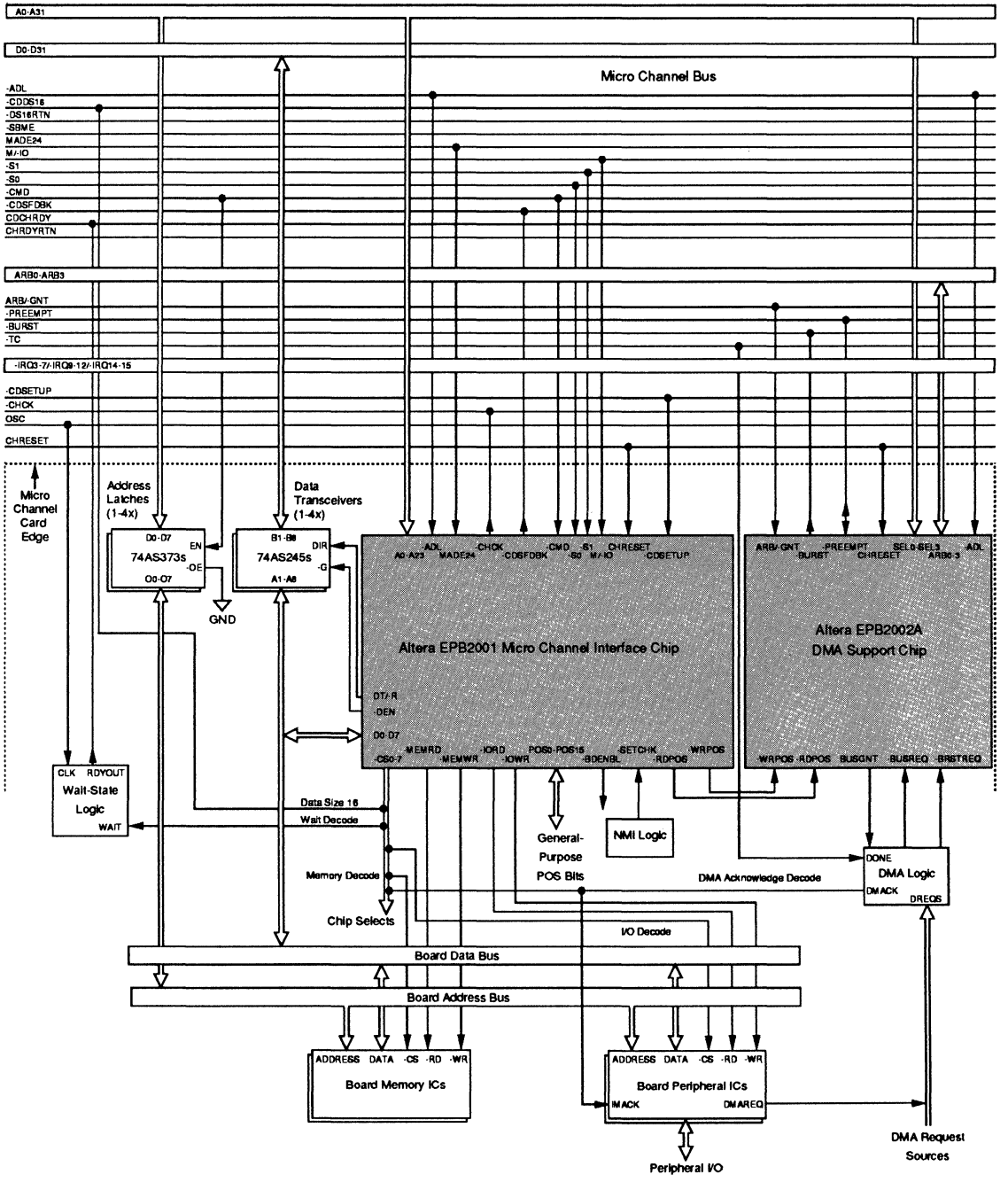
EPB2002A: It provides the DMA arbitration functions for the adapter card. Since transfer coordination occurs over the arbitration bus, the interrupt interface is not needed.

DMA logic: The PS/2's DMA controller on the system motherboard services the transfer requests of the peripheral function.

Connections to the Peripheral: The Micro Channel provides no dedicated DMA request lines from the adapters to the motherboard. A bus slave must signal to the motherboard DMA channels that a transfer is needed by arbitrating for the bus. The DMA controller monitors every bus arbitration cycle. If the bus slave arbitrates for and wins the bus, the DMA controller detects the priority code on the arbitration bus (**ARB0-3**) associated with the bus slave. It interprets this code as a transfer request from the bus slave and initiates the data transfer for which it has been configured. When transfers are complete, the slave releases the bus.

This scheme works only if the bus slave/EPB2002A and motherboard DMA controller are configured appropriately. The Adapter Description File (ADF) for the bus slave card must contain its assigned arbitration

Figure 3. DMA Bus Slave Adapter Interface



level, which is loaded into the EPB2002A's POS register bits at system power-up. The same arbitration level must be loaded into the PS/2 DMA controller's command registers for the channel assigned to the card. In addition, source and destination addresses, direction of transfer, byte count, and other transfer parameters must be initialized by writing the appropriate control registers (refer to *Application Brief 15*).

When control of the bus is granted through the arbitration process, the PS/2 DMA controller transfers data in two cycles: (1) the first cycle fetches data from either memory or I/O and places it into a holding register; (2) the second cycle, run on the Micro Channel, moves the data from the holding register to its destination. This process is much more efficient than interrupt-driven transfers, although single-cycle transfers that move directly from data source to destination are possible and offer twice the bus transfer rates. (The bus master adapter, described below under "Micro Channel Master Interface," provides the benefits of single-cycle transfers.)

The DMA logic, shown in Figure 3, can include selector logic for burst or non-burst transfers and request-enabling logic, data-transfer counters, or other useful glue functions to interface the peripheral DMA functions to the EPB2002A. The **-TC** input from the MC Bus signals the end of a block transfer. It can be used by the DMA logic to relate the request inputs to the EPB2002A and to release the bus.

In general, use of the EPB2002A is not contingent on use of the EPB2001. Since the interface between the two chips consists only of **-RDPOS** and **-WRPOS**, a PLD/TTL-based or ASIC Micro Channel interface can be used with an EPB2002A that provides bus arbitration support. The interface need only generate **-RDPOS** and **-WRPOS** for the EPB2002A. These signals can be generated as latched decodes of **-CDSETUP** and I/O-read or I/O-write bus cycles, respectively, strobed by **-CMD**, which would result in the correct read and write strobes for the EPB2002A POS register bits (refer to the *EPB2001* and *EPB2002A* data sheets). Thus, DMA support can easily be added to an existing design with the EPB2002A and non-DMA interface.

Micro Channel Master Interface

The Micro Channel architecture supports bus master attachments or adapters. A bus master adapter differs from a slave adapter in four important ways:

1. A master must drive the address, data transfer control, and arbitration channels.

2. Signals such as **CHRDYRTN** and **-DS 16 RTN** (**-DS 32 RTN** for 32-bit) must be monitored by the bus master adapter. The address channel must be held stable until these signals are returned from the supported slave devices.

3. A master arbitrates for the channel. Once it is granted bus control, it controls bus operation for one or more channel cycles. This feature significantly reduces the bandwidth and burden on the processor, increases functionality, and generally improves performance.

4. Improved functionality, however, is accompanied by higher board cost, greater complexity, timing constraints, and increased power consumption.

A bus master adapter thus offers an alternative to the PS/2 motherboard DMA controller by making direct, single-cycle transfers possible for higher data rates.

Figure 4 shows an interface that uses an EPB2001 and EPB2002A. It includes the logic used for the DMA bus slave, a local adapter DMA controller that generates addresses during data transfers, and drivers (74AS244s) for the Micro Channel address bus and control bus (**-ADL**, **-S0**, **-S1**, **M/-IO**, **MADE24**, **-CMD**, and **-SBHE**; for additional information, refer to *Application Note 15*). The adapter bus master takes in additional signals from the Micro Channel such as **CHRDYRTN** (channel ready return) and **-DS16RTN** (data size 16 return) to coordinate bus cycles just like the CACP or motherboard DMA controller.

The functions provided by the EPB2001 and EPB2002A are essential building blocks in the design of the bus master. The adapter DMA controller may be a custom design based on PLDs and/or TTL, or a standard VLSI controller combined with PLDs to generate the Micro Channel control interface.

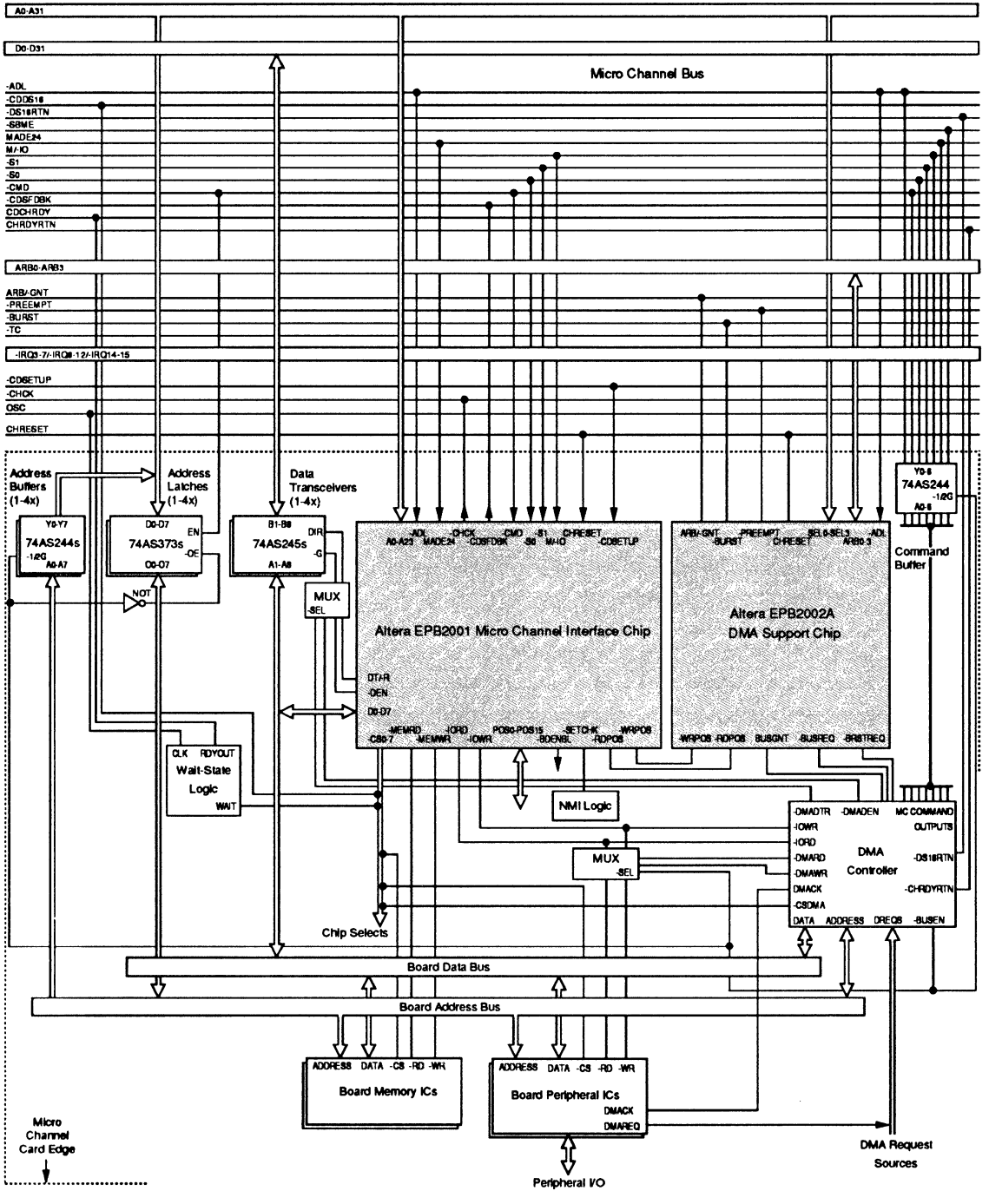
The DMA controller still has **-IORD** and **-IOWR** as inputs from the EPB2001. These inputs, together with the **-CSDMA** chip select, allow the system CACP to configure the DMA controller for transfers as well as read status.

The DMA controller must also generate control strobes for the peripheral and data transceivers. The multiplexers shown in Figure 4 switch control from the EPB2001 to the DMA controller when the **-BUSEN** signal goes low. The **-BUSEN** signal is asserted when **BUSGNT** is generated from a successful arbitration cycle. **-BUSEN** also enables the DMA controller's addresses and control signals onto the MC Bus to execute the DMA transfer(s).

Conclusion

The added logic-interface functions required in a bus master increase overall card cost. Most cards do not require the performance enhancement of a bus master. However, in those applications where performance is a must, a bus master can give exceptional performance. As illustrated in Figure 4, it is easy to upgrade to a bus master design by using the EPB2002A, DMA control, and buffer logic.

Figure 4. Bus Master Adapter Interface



Reference

IBM Personal System/2 Model 80 Technical Reference. International Business Machines Corporation, 1987.

Sales Offices

CALIFORNIA (Headquarters)
ALTERA CORPORATION
2610 Orchard Parkway
San Jose, CA 95134-2020
Telephone: (408) 984-2800
FAX: (408) 248-7097

SOUTHERN CALIFORNIA
ALTERA CORPORATION
17100 Gillette Avenue
Irvine, CA 92714
Telephone: (714) 474-9616
FAX: (714) 474-7355

ILLINOIS
ALTERA CORPORATION
200 W. Higgins Road
Suite 322
Schaumburg, IL 60195
Telephone: (708) 310-8522
FAX: (708) 310-0909

MASSACHUSETTS
ALTERA CORPORATION
945 Concord Street
Framingham, MA 01701
Telephone: (508) 626-0181
FAX: (508) 879-0698

GEORGIA
ALTERA CORPORATION
1080 Holcomb Bridge Road
Suite 300, Bldg. 100
Roswell, GA 30076
Telephone: (404) 594-7621
FAX: (404) 998-9830

TEXAS
ALTERA CORPORATION
Signature Place
14785 Preston Road
Suite 550
Dallas, TX 75240
Telephone: (214) 233-1491
FAX: (214) 233-1493

NEW JERSEY
ALTERA CORPORATION
Office Gallery
981 U.S. Hwy. 22
Suite 2000
Bridgewater, NJ 08807
Telephone: (201) 526-9400
FAX: (201) 526-5471

EUROPE
ALTERA CORPORATION
25, Av Beaulieu
B-1160 Bruxelles
Belgium
Telephone: 02 660 2077
FAX: 02 660 5225

ALTERA CORPORATION
21 Broadway
Maidenhead,
Berkshire SL6 1JK
England
Telephone: 0628 32516
FAX: 0628 770892

ALTERA FRANCE
72-78 Grande Rue
F-92310 Sèvres
France
Telephone: 1 4534 3787
FAX: 1 4534 0109

ALTERA GMBH
Ismaninger Straße 21
D-8000 München 80
West Germany
Telephone: 089/41300614
FAX: 089 4706284

For a complete list of Sales Representatives and Distributors, please call Altera Marketing at (408) 984-2805 ext. 101.

Notes:

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
(408)984-2800 Telex 888496