the Logical Alternative

Micro Channel Adapter Design Handbook

August, 1988

MICRO CHANNEL ADAPTER DESIGN HANDBOOK

ALTERA CORPORATION

AUGUST, 1988

TABLE OF CONTENTS

1. Application Note #14:

   IBM PS/2 Add-On Card Interfacing with the EPB2001 & EPB2002

2. Application Note #15:

   IBM PS/2 Add-On Card Software Design

3. Application Brief #72:

   IBM PS/2 Master and Slave Adapter Design

4. Data Sheet:

   EPB2001 / EPB2002 User-Configurable Adapter Interface Chips for PS/2 Micro Channel

IBM PS/2 ADD-ON CARD INTERFACING WITH THE EPB2001 & EPB2002

INTRODUCTION

The IBM Personal System/2 (PS/2) architecture announced in early 1987 included a new add-on card bus architecture called the Micro Channel Bus. This architecture, which supports both 16- and 32-bit processors, includes the concepts of high-performance multi-device bus arbitration, level-sensitive interrupts, and Programmable Option Select (POS) registers. The POS registers are intended to replace DIP switches and jumpers on the add-on card, and allow software-configuration of card features. Simultaneously, in order to reduce the overall size of the PS/2 system, the physical size of add-on cards has been reduced by some 40% to only 33 square inches. The net result is a higher-performance, more reliable scheme for the attachment of add-on cards to the PS/2. The cost of these benefits, however, is a more complex interface design problem for the add-on card designer requiring a highly-integrated VLSI approach.

This Application Note illustrates the use of Altera's user- configurable adapter interface chips for the IBM Personal System/2 (PS/2) Micro Channel. A typical application is desribed, consisting of a Multi-Function card including 32K words of static CMOS RAM, a general-purpose output port and two serial ports. The Micro Channel interface EPLDs, the EPB2001 and EPB2002, provide all essential control interfaces between the Micro Channel Bus (MC Bus) and a PS/2 feature adapter (add-on board). Implementation of optional add-on card features, such as wait- state generation, output port and bit-rate prescaler, is also shown using a BUSTER EPB1400 EPLD. User-configurability allows these highly-integrated devices to provide functions typically requiring multiple components.

POS I/O pins from the EPB2001 are used to control typical board-level functions: wait-state duration and DMA channel selection and enabling. Board control lines (-DEN, DT/-R, -IOWR, -MEMRD, etc.) are used to control memory and I/O accesses. The EPB2001's chip select block provides all address decoding for board functions. The overall versatility of the EPB2001/2002 is illustrated in a real application.

In addition to this specific application example, a set of general design tips for the EPB2001/2002 is presented at the end of this Note. These tips show ways that the programmable nature of these chips can be exploited to provide optional, board- specific functions, such as latched addresses or card data-size feedback. 32-bit system design considerations are also discussed.

This Application Note assumes the reader is familiar with the basic concepts of the IBM MC Bus. The reader is directed to the references listed at the end of this Note for further information on the basic characteristics of the Micro Channel. Comprehensive functional and parametric information on the EPB2001/EPB2002 may be found in the EPB2001/EPB2002 Data Sheet, available from Altera.

MULTI-FUNCTION CARD APPLICATION OVERVIEW

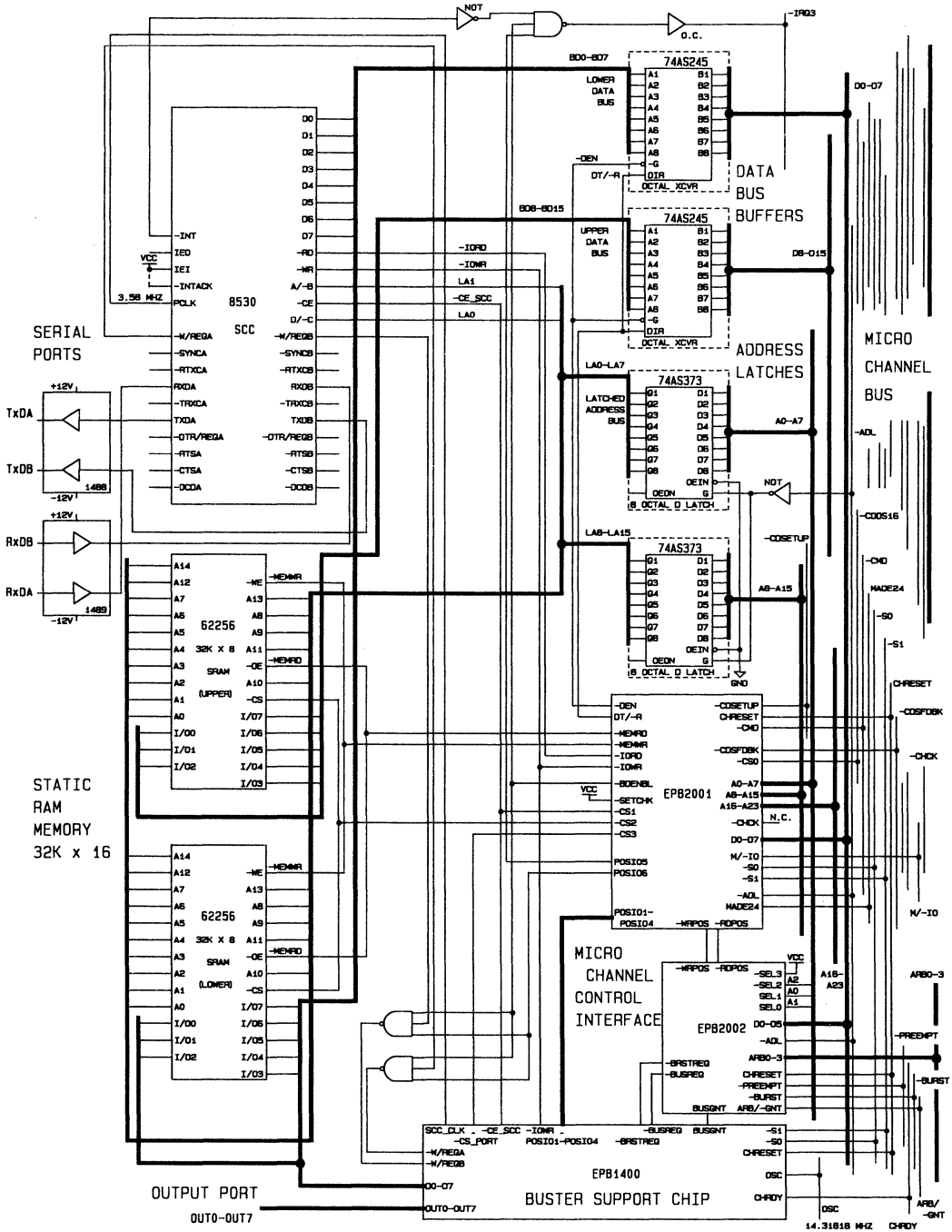
The block diagram in Figure 1 shows the overall structure of the Multi-Function card design. Shown on the right are the MC Bus lines which run to all add-on card slots. The EPB2001 provides mandatory POS register functions required by the specification in any add-on card interface. In addition, board control logic, board I.D. storage and address decoding are integrated onto the chip. The EPB2002, shown at lower right, provides DMA arbitration support. Not all add-on designs require DMA, but where it is needed, the EPB2002 fully supports the IBM bus exchange protocols.

This board design appears to the PS/2 as either an 8-bit I/O peripheral (for accesses to the Z8530 Serial Communications Controller (SCC)), or as a 16-bit memory extension (32K words). The two 74AS245 transceivers shown at upper right buffer data transfers between the Micro Channel and I/O or memory. Memory will always be accessed as words (no byte read/write), and as a result, selective enabling of the buffers using -SBHE from the Micro Channel is not required. 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 be latched since memory capacity is 64K (2**16). The EPB2001's chip select logic will handle upper-order address decoding for the various I/O and memory chips on the board.

The Z8530 SCC provides two 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 which interface to the EPB2002. The EPB2002 requests use of the MC Bus via the -PREEMPT line in response to these requests.

The clock for the SCC is provided by a divide-by-four counter, which is driven by the OSC line

Figure 1. Multi-Function Card Block Diagram



on the MC Bus. This clock divider is implemented in a small portion of an EPB1400. The OSC line provides a precision clock input of frequency 14.31818 Megahertz. The clock divider generates a 3.58 MegaHertz output for the SCC. As a result, a local oscillator is not needed for the SCC.

Wait-state logic associated with the Z8530 is also designed into the EPB1400. In a Micro Channel default bus cycle, the width of the -CMD bus transfer strobe may be as narrow as 90 nS. The 4 MegaHertz Z8530 used in this design requires a 250 nS minimum width on -RD and -WR to the device. As a result, the bus cycle must be "stretched" at least 160nS. This is done by inserting wait-states to extend the cycle. This logic will be described in detail later.

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

The two pairs of serial data lines (RxD and TxD, Channel A & B) from the SCC are connected to 1488/1489 line driver/receivers which in turn form RS-232-compatible interfaces. In this application, the full DCD/CTS/RTS handshake has not been shown. This could be easily added by the addition of 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.

EPB2001 INTERFACE FUNCTIONS

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:

BOARD I.D.

As required for any MC Bus add-on, the EPB2001 provides two CMOS EPROM bytes at location 0100-0101H for the board I.D. These are read-only locations accessible from the MC Bus. These I.D.'s are unique to a given board design and are allocated by IBM to registered Independent Developers. The developer must contact IBM directly for such I.D.'s.

POS REGISTERS

The four required POS registers 0102-0105H are used in this application 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 latter DMA control functions are also mapped into the satellite POS register bits on the EPB2002. The bit-mapping of these functions is illustrated in Figure 2.

POS bits which are required outside the EPB2001 are brought- out via the POS I/O pins. This mapping is shown in Figure 2 as well.

BOARD CONTROL

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

The Board Enable (-BDENBL) output of the EPB2001, which reflects POS register 0102H bit 0, is used in this application to mask DMA requests and interrupts prior to enabling the card at system boot time. In addition, this bit is internally factored into the -CSx enable functions to disable chip selects when the board is disabled. This also suppresses generation of the -CDSFDBK line prior to card set-up.

CHIP SELECT LOGIC

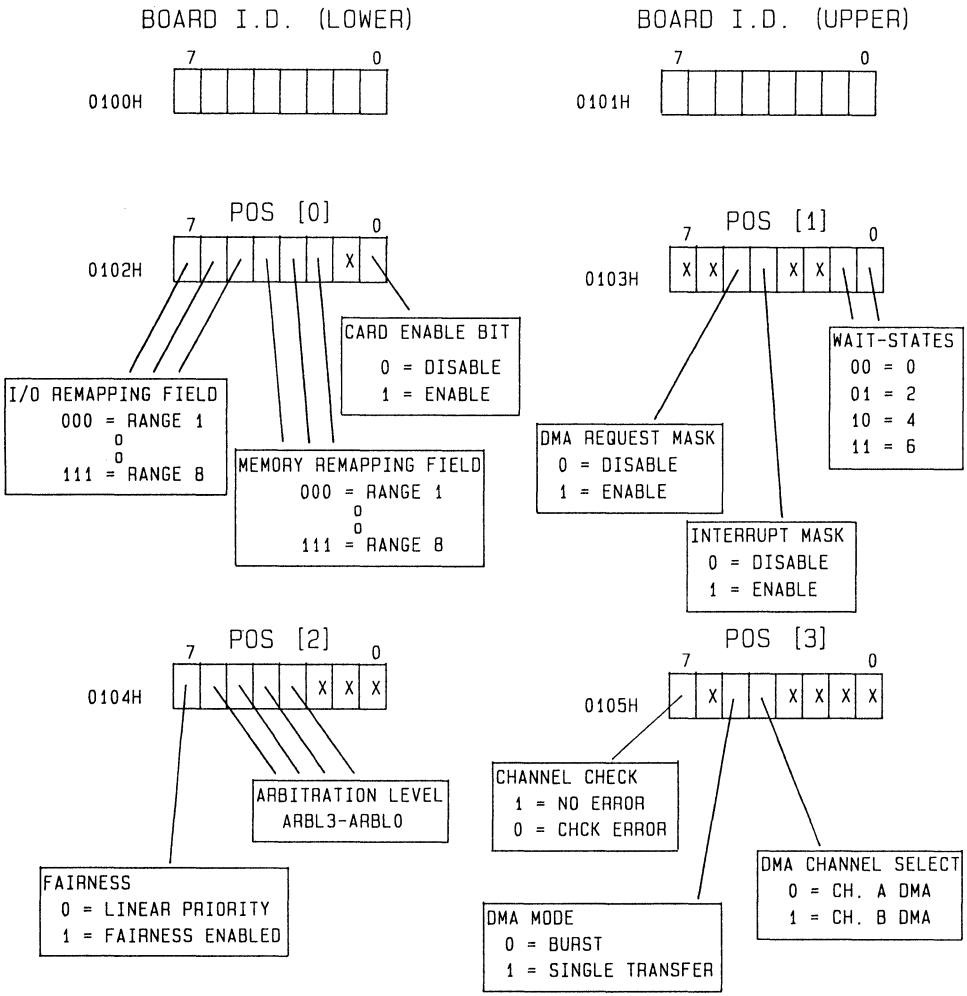
The chip select block in the EPB2001 provides chip select outputs for the CMOS SRAM, SCC, and BUSTER Output Port. In addition, one of the -CSx outputs is used to drive the -CDDS16 (card data size 16) output to the MC Bus whenever the SRAM (organized as words) is accessed.

In this case, the first three chip select outputs will be 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. In this example, all -CSx functions with the exception of -CDDS16 will be 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 could be used to drive other peripheral or memory chips, or supply design- specific board logic as described later.

Each of the chip selects has 8 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

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

select decode block. In this way the PS/2 can control address response ranges for the card and associated memory / I/O resources as it configures the system. This is used to eliminate address conflicts between cards without mechanical setting of DIP switches or jumpers, an error-prone process. The use of such mechanical arrangements in fact violates the Micro Channel specification.

Due to the structure of the chip select block as defined in the EPB2001/2002 Data Sheet, an address block of 2**N locations must 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 256K RAM chip select must have a base address which is a multiple of 256K. This is not a severe restriction in most practical applications, but should be kept in mind as alternate address locations are selected.

EPB2002 INTERFACE FUNCTIONS

The EPB2002 does not require the degree of programmability that the EPB2001 contains. The EPB2002 implements the bus arbitration protocol defined for the Micro Channel in an asynchronous state machine. POS register bits are included which are readable and writeable from the MC Bus. These bits control the four-bit Arbitration Level for the board (arbitration priority) and enable Fairness (Fairness is a mechanism specified by IBM to eliminate bus "hogging" by bursting DMA devices).

The actual arbitration process is discussed in detail in the EPB2001/2002 Data Sheet and in the IBM documentation listed at the end of this note. The EPB2002 provides direct a.c. and d.c. compatibility with the interface signals required. For more information on this process the reader should consult the referenced documents.

The bit positions and POS register location which these bits will be mapped into is up to the board designer. The MC Bus specification does not define required POS locations for them. Any of POS registers 0102-0105H may be used for this purpose. The EPB2002 allows the remapping of these bits by appropriate connections to the SELx inputs to the chip. This is described in detail in the EPB2001/2002 Data Sheet. By connecting the SELx inputs as shown in Figure 1, these bits have been mapped into POS register 0104H for this application, bit locations D3-D7 on the bus.

Further details on the electrical interface of the EPB2002 to the BUSTER and SCC components will be given in a later section.

Z8530 INTERFACE CONSIDERATIONS

SYSTEM INTERFACE

Basic signals involved in interfacing the Z8530 to an MPU bus (the MC Bus in this case) are shown in Figure 3. These 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. Here we are using the IBM MC Bus specification convention of a leading hyphen to indicate an active-low signal name. The Z8530 includes 9 Read-able and 15 Write-able registers to set device configuration, report status and access data. To reduce address input pin requirements, the Z8530 uses the concept of 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 subsequent access, the pointer register is automatically reset to the address pointer register. Every command register access is thus a two bus cycle process: write address pointer, write or read selected register.

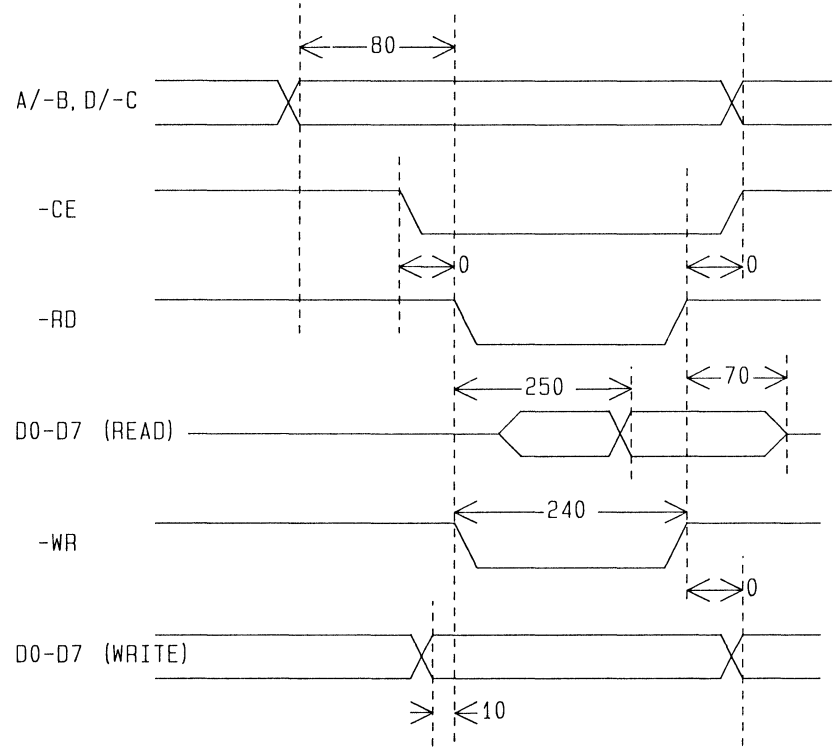
The waveforms shown are for data transfer cycles. Timing for a command cycle is similar, but involves the additional cycle mentioned above. Timing data is for the 4 MegaHertz Z8530. In general, the timing is very straightforward: the Z8530 requires that -CE, D/-C, and A/-B be set-up prior to the strobe (-RD or -WR) going active low. -CE's set-up 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 will place valid data on its D0-D7 pins within 250 nS of the -RD active edge.

These are the SCC device requirements. The subsequent 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 will be connected to the latched address inputs provided by the 74AS373 devices. The MC Bus provides a minimum set-up of 85 nS from address valid on the MC Bus to -CMD falling. Using 74AS373's, 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 30 nS. The Z8530 requires only 0 nS, so over 55 nS of timing

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



margin is available on the -CE path. The EPB2001 latches the -CE line with -ADL for the entire bus cycle.

The address ranges which the EPB2001 decodes for the -CE input to the SCC consist of blocks of 4 locations. The EPB2001 allows the designer to pre-specify up to 8 such ranges. 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 which resides on a 4-byte boundary.

The EPB2001 asserts the -DEN line for the 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 spec has a minimum of 40 nS from -ADL active to -CMD active. The -CMD signal triggers -IOWR, and -IOWR has a delay of up to 20 nS, which tracks parametrically with the -DEN delay. As a result, there will be a minimum of 20 nS between -DEN falling and -IOWR falling. The 74AS245's 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 set-up time requirements of the Z8530.

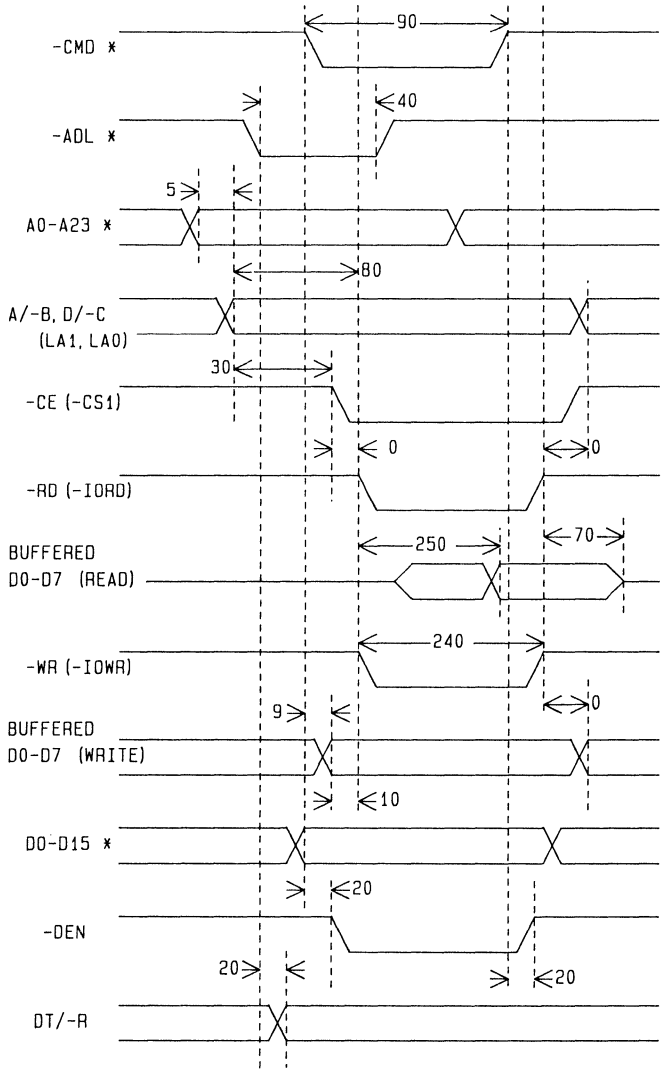
The MC Bus requires drivers to tristate within 30 nS of -CMD going inactive. The 74AS245's 9 nS tristate delay, coupled with the 20 nS -CMD to -DEN delay on the trailing edge, is a good match with the bus spec.

INTERRUPT INTERFACE

The SCC may be programmed to run in either an 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 on a variety of error conditions. The reader is referred to 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, it's drive is

Figure 4. EPB2001/Z8530 Micro Channel Timing



insufficient to handle the MC Bus Iol requirements and must be re-buffered. Several prioritized -IRQx lines are available on the MC Bus, and in this case selection of the -IRQ3 line fixes the SCC's priority at a relatively low level, fourth lowest in the hierarchy. This 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 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) which 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. This also resets POS register 0105H, bit 7 (when inactive, a logic one). 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 will handshake with the EPB2002 when DMA transfers are enabled. Due to constraints in the basic DMA structure of the Micro Channel / 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 BUSTER support device. It has the two -W/REQx outputs of the SCC as inputs, and generates either a -BRSTREQ or -BUSREQ line for the EPB2002. 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 EPB2002 -BRSTREQ or -BUSREQ input, and results in the EPB2002 asserting -PREEMPT. Timing for the assertion is not critical.

Timing for the removal of the request is important, however. 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 EPB1400. In the burst transfer mode, the EPB2002 will deassert -BURST approximately 50 nS after this signal is deasserted, and at the same time BUSGNT will be 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 the absence of DMA overrun (erroneous extra transfers), the DMA transfer cycle must be extended via wait states so this requirement is satisfied.

Since -BURST is not used in single transfer mode, this timing constraint need not be considered. However, for simplicity, in this design single transfer and burst wait-states will be the same.

In of single transfer mode, it is necessary to deassert the -BUSREQ line on the occurrence of BUSGNT active and the -S0, -S1 status lines going to the active state (indicating start of the DMA transfer). The logic in the lower left of the BUSTER schematic (Figure 6) 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 will 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 otherwise indicate to board logic the granting of the bus.

SERIAL INTERFACE

Minimal serial interfaces for Channel A & B are shown in this design. Modem control signals (-RTS, -CTS, -DCD) are not used to control data transfer over the RS-232 links. This 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 Ground lines for each RS-232 line need be connected.

BUSTER SUPPORT LOGIC

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 use of a wait-state generator, in this case designed into a BUSTER component. The timing source for the wait-state generator, the OSC signal from the MC Bus, has a period of about 70 nS. The wait-state generator will pull 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 software programmable, and can be selected by parameters in

the add-on card's Adapter Description File (ADF). The ADF, required for every add-on card, describes to the PS/2 the available address response ranges and configuration options for a given card. The system reads the board I.D. for each card in its backplane when the syetem is booted. The board I.D. is used to find the associated ADF file 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.

BUSTER's 28 MegaHertz operating frequency supports the wait- state generator and clock divider functions with considerable margin given the 14 MegaHertz clock inputs.

The number of wait-states can be varied from 0 to 6. A minimum number of clock edges to support the 250 nS width required by the 4 MegaHertz SCC is 4 x 70 nS = 280 nS. If DMA is used as described earlier, 6 x 70 nS = 420 nS 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 zero. While counting, CHRDY is held low.

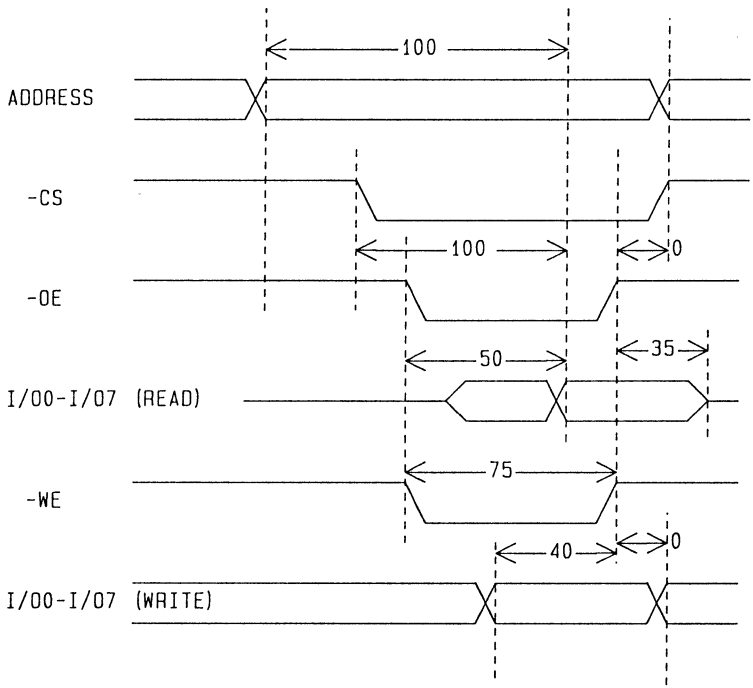
OUTPUT PORT

As an added feature, an output port is constructed in a portion of the BUSTER EPB1400 for general-purpose use. These 8 lines may be used to control whatever output functions are required. This block is shown at the top of Figure 6. It is entered using Altera's LogiCaps schematic capture package and TTL MacroFunction elements. The integral MPU data port on the EPB2001 simplifies the MC Bus interface.

This output port uses another of the EPB2001's -CSx outputs for selection during write operations. The -WS input to the EPB1400 is connected to the -IOWR output of the EPB2001. Since the output port is the only resource on the BUSTER device mapped into the I/O address space, no address inputs are needed.

BUSTER's write control interface timing is very simple: a minimum write strobe (-WS) low width of 20 nS will insure 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 BUSTER's -WS input falling, and the chip select 10 nS prior to -WS falling. Since MC Bus data is valid at -CMD's leading edge (prior to -IOWR leading edge), and the EPB2001's -CSx lines are valid at least 55 nS prior to

Figure 5. 62256LP-10 Timing Requirements



-CMD falling, the minimum overlaps are easily satisfied. Minimum strobe width of 90 nS is also more than adequate.

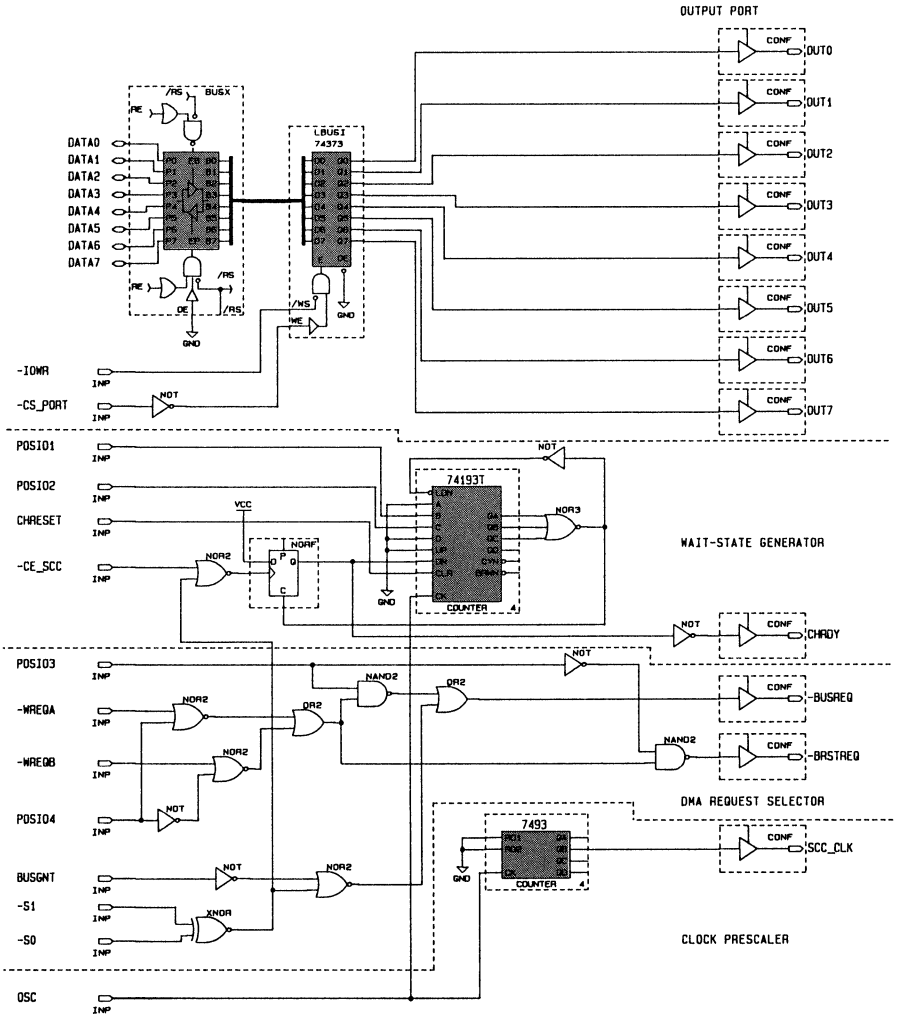
STATIC RAM INTERFACE CONSIDERATIONS

The 32K word static RAM on the Multi-Function card has been implemented using two 32K x 8 CMOS SRAM devices. These devices are 62256LP-10 RAMs, packaged in 28 lead, 600 mil DIP packages. The devices have 15 address inputs (A0-A14), eight bidirectional data lines (I/O0-I/O7), a chip select (-CS), output enable (-OE) and write enable (-WE).

For these chips, access time is 100 nS from address or chip select. Output enable delay from -OE to valid read data is 50 nS. Minimum write enable width (-WE) is 75 nS, with a 40 nS set-up time for of valid write data to -WE trailing edge.

As shown in the Multi-Function board schematic (Figure 1), addresses from the 74AS373's drive the SRAM address inputs. SRAM I/O's drive the upper and lower bytes of the board data bus. The -CE inputs

Figure 6. BUSTER Multi-Function Support Chip



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 85 nS set-up of addresses to -CMD. 5 nS delay through the 74AS373's gives greater than 80 nS of address to -MEMRD or -MEMWR set-up at the SRAM devices. Even assuming minimum strobe width of 90 nS (minimum -CMD width on the MC Bus), this allows 170 nS of address access time at the SRAMs. Chip select access is reduced by the chip select decode delay on the EPB2001 (30 nS), but is still 140 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 set-up to the leading edge of -CMD, and a minimum 90 nS -CMD width. This results in 80 nS of overlap, more than adequate even with the 9 nS buffer delay.

GENERAL EPB2001/2002 DESIGN TIPS

EPB2001 CHIP SELECT LOGIC USE IN 32-BIT SYSTEMS

The decoding inputs to the EPB2001 are labelled A0-A23. This allows 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. This is true because the granularity required in chip select ranges is often quite coarse.

A typical adapter may be memory only, 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 assure correct access of the POS registers and board I.D. 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 2K bytes over the full 32-bit address range. Since most RAM chips in use today are much greater than 2K byte density, this is usually sufficient for memory addressing and memory-only applications.

Mixed memory/I/O adapters can also use this scheme. However, it implies that I/O peripherals also be placed on 2K byte boundaries, and 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 such cases, more than enough address inputs are available on the EPB2001 for 16- or 32-bit applications.

ALTERNATIVE ADDRESS INPUT USE

If an I/O-only adapter is being designed, or one for 16-bit applications specifically, all 24 address inputs, as well as MADE24 may not be needed. In such cases, these inputs may be used to provide additional board-specific functions.

For example, it may be desireable to generate read or write strobes for specific board I.C.'s. 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 includes:

- 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 be used to access POS register bits (output only). These may be used to increase POS register output pins to 24 from the 16 POSI/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 a devices's data bus width as 32 or 16 bits, respectively. They are derived as unlatched decodes of appropriate address ranges.

As described in the EPB2001/2002 Data Sheet, 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 desireable to factor into certain chip select enabling functions 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 during error recovery (channel check) routines.

MC MAP SOFTWARE SUPPORT

The MC Map Micro Channel Interface Assembler, available from Altera, provides an easy table-driven entry mechanism for specifying EPB2001 designs. All programmable options mentioned above are easily specified using 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. Programming of the component may be accomplished using Altera's PC-based hardware: an LP4 or LP5 programming card, PLE3-12 Master Programming Unit and PLEJ2001 programming pinout adapter. The actual device is programmed in seconds and ready for use on the board.

Further information on the MC Map software and programming hardware may be obtained from Altera's Marketing group.

SUMMARY

The EPB2001 and EPB2002 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. Coupled with other EPLDs such as BUSTER for unique application features, extremely efficient interfaces for PS/2 adapters can be rapidly designed and implemented.

WHERE TO GET MORE INFORMATION

Information on IBM's Independent Developer Assistance Program may be obtained from IBM through

IBM Independent Developer Assistance Program - (800)426-7736

Registration in this program is a prerequisite for obtaining technical assistance and board I.D. assignments from IBM.

Specifications and technical reference manuals may be obtained from IBM through

IBM Technical Directory - (800)426-7282

REFERENCES

1. EPB2001/2002 Data Sheet, Rev. 1.0, February,1988

2. Z8030/Z8530 Data Sheet, Rev. E, August,1987

3. IBM Personal System/2 Model 50/60 Technical Reference (IBM# 68X-2224)

4. IBM Personal System/2 Model 80 Technical Reference (IBM# 68X-2256)

5. IBM Personal System/2 Model 50,60,80 Micro Channel Architecture (IBM# G360-2637) Rev. 5.3, May,1987

6. Altera Data Book, January, 1988

IBM PS/2 ADD-ON CARD SOFTWARE DESIGN

CONTENTS

- Programmable Option Select (POS)
- Micro Channel Adapter Installation
- Adapter Description Files (ADF)
- Installation Procedure
- Power-On Self-Test (POST)
- Utility to read back Adapter I.D. and POS Register contents
- Altera Multi-Function Adapter Description File
- Software Driver Source Code

INTRODUCTION

This Application Note provides the software design aspects for an adapter card for the IBM Micro Channel bus architecture, described in Application Note 14. Configuration and installation information for Micro Channel-based adapter cards is also discussed. Included is a utility to interrogate each adapter slot and read back the adapter I.D and POS register contents.The Altera Multi-Function adapter card developed for the IBM PS/2 Models 50/60/80 is described. The Adapter Description File(ADF) and a software driver written for this adapter illustrate the software design issues involved. This note, together with Application Note #14, provide tested hardware and software design examples to speed and simplify IBM PS/2 add-on card design using Altera's user-configurable solutions.

The card is designed utilizing Altera's Micro Channel interface and DMA arbitration chips, the EPB2001 and EPB2002. The Altera EPB2001 and EPB2002 Micro Channel Bus interface chips handle interface requirements for a PS/2 adapter, ranging from POS register implementation and address range definition for chip selects to bus arbitration for single cycle or "burst" DMA transfers. For a detailed description and timing information related to the Altera Multi-Function add-on card, the reader is referred to Altera Application Note 14 entitled "PS/2 Add-on Card Interfacing With The EPB2001 and EPB2002".

PROGRAMMABLE OPTION SELECT (POS)

The Programmable Option Select (POS) feature specified in the Micro Channel Bus Architecture for IBM PS/2 Models 50/60/80 eliminates configuration switches on add-on cards or adapters. Programmable Option Select registers are implemented through programmable registers residing at I/O addresses 0100 Hex through 0107 Hex on each add-on card. Other objectives of POS are to permit installation of multiple identical feature cards, positive identification of any card by slot and resolution of resource assignment conflicts. Each add-on card type needs to be identifiable by a unique 16-bit I.D. number. The adapter I.D. is implemented by providing two read-only registers at addresses 0100H (for the low order byte) and 0101H (for the high order byte) on every card.

IBM has published the following guidelines for adapter I.D numbers:

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 multifunction cards
8000-8FFF	Video

A card developer must register with the IBM Independent Developer Assistance Program (IDAP). Call (800) 426-3333 for information. An IBM IDAP number is assigned to the developer which must be used for further inquiries and technical assistance including obtaining adapter I.D. numbers.

The Micro Channel specification is quite complex. The reader is encouraged to review the IBM Technical Reference Manual for the applicable PS/2 Model (50/60/80). The EPB2001 and EPB2002 greatly simplify the Micro Channel interface portion of an add-on card design and insure compatibility with the Micro Channel specification.

ADAPTER INSTALLATION

ADAPTER DESCRIPTION FILES

After the add-on card hardware design is complete, based on the POS register bit assignment, an Adapter Description File (ADF) must be created on a 3 1/2" diskette. The ADF should be an ASCII file and may be created using any text editor. Each add-on card or adapter must have a separate ADF with file name @CARDID and an extension of "ADF" prior to installation. For example, an adapter that uses an I.D. of ABCD Hex (POS Reg. at address 0100H contains the byte CD and POS Reg. at address at 0101H contains the byte AB), must be accompanied by an ADF named @ABCD.ADF on a 3 1/2" diskette called an option diskette. The ADF provides information regarding POS settings and usage of resources for

Automatic Configuration. The ADF also provides input for the System Configuration Utilities (these are packaged with every system on a 3 1/2" Reference diskette), help screens and prompts. The ADF provides to the PS/2 configuration utilities information regarding the available address response ranges and configuration options for a given card. The system reads the board I.D. for each card in its backplane when the system is booted and compares it to the information stored in its CMOS RAM. When a new adapter is installed the board I.D. is used to find the associated ADF file 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 ADF files supplied for each add-on card (adapter) by matching the unique adapter I.D. with an ADF file. The configuration data is subsequently stored on battery-backed CMOS RAM along with the adapter I.D. numbers.

CREATING AN ADF

Syntax for creating an ADF is available in the relevant IBM PS/2 Model Technical Reference Manual. Syntax rules must be followed strictly since the 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] - POS register at I/O address 0102H
pos[1] - POS register at I/O address 0103H
pos[2] - POS register at I/O address 0104H
pos[3] - POS register at I/O address 0105H

CAUTION: Bit 0 in pos[0] and bit 7 in pos[3] are reserved for Card Enable and -CHCK (Channel Check) functions and must not be written to. The letter "X" must be specified in the respective bit positions in the ADF when specifying configuration data for POS registers 0 and 3 to avoid writing to those two bits.

The NamedItem structure is used to specify a field providing a choice of resource or configuration options. The text within quotes following the keyword "Prompt" is what 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 so that the user can choose a particular option.

The text within quotes following the "choice" keyword associates the bit pattern in the POS register to one of the configuration choices that the resource may have. This text is displayed in a field in the configuration screen. The user steps through this field using [Next] and [Previous] function keys while the various "choices" to which the resource may be configured are displayed. 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.

For customizing the configuration, the user invokes Change Configuration in the Set Configuration menu under the System Configuration Utilities. If a particular choice conflicts with a previously assigned configuration for the same resource while stepping through the various "choices" for a resource a "* Conflicts" flag appears at the top right hand corner of the screen and an * appears next to the field that caused the conflict. The conflict can be resolved by stepping through and choosing a different resource choice either for the new adapter or for the existing adapter. In each field the user may invoke help by pressing the F1 function key whereby the text within quotes following the Help keyword appears.

INSTALLATION PROCEDURE

Make a back-up copy of the Reference diskette (the original diskette is read-only) to allow update of the system configuration after installation of a new adapter. The ADF file for the adapter being installed may either be on a separate option diskette or it may also be on the back-up diskette.

Power-up the system with the Reference diskette (the original one) in drive A. The System Configuration Utilities Screen comes up (blue screen with the IBM logo).

Press ENTER to continue and invoke "Set Configuration". In the Set Configuration menu invoke the "Copy an Option Diskette" utility.

When prompted for the new option diskette, insert the back-up diskette (or the option diskette as the case may be) in drive A. At this point the new ADF or option file is read and the new adapter configuration information is merged with the previous information.

Use "Back-up Configuration" to write the up-to-date configuration data on the back-up diskette. This diskette should subsequently be used whenever System Configuration Utilities need to be run. Re-start

the computer without the back-up diskette in drive A. This time the system boots up with the updated configuration stored in the battery-backed CMOS RAM.

If the system boots up, the adapter has been installed successfully. In order to view the updated configuration, insert the back-up diskette in drive A. Make A: the default drive and invoke "sc". This invokes the Set Configuration utility under the System Configuration Utilities. While in the Set Configuration menu, the user may simply view the current configuration, or invoke another utility within this menu 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 I.D. number 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 need to be run again. A configuration error is flagged by the number "165" on the screen at system power-up. 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. In either case configuration error number 165 is flagged.

If an adapter is turned off (e.g. an external drive or the terminal) or is not working properly, error number "162" occurs. When this error occurs, make sure the adapters and devices attached to the adapter cards have power. Boot the system with the back-up diskette in drive A and follow the prompts in the System Configuration Utilities. If cards are moved to different slots, POST flags error number 165. In this case, follow the prompts for running "Automatic Configuration". POST automatically reconfigures the system and updates the CMOS RAM accordingly.

Figure 1 contains a listing of the source code in "C" for a utility that interrogates each slot in the system. It runs setup cycles to each adapter in order to read back the contents of I/O addresses 0100H to 0105H. This routine was utilized on a PS/2 Model 80 to read back POS register contents of adapters in all the 8 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, if applicable. The number of slots being interrogated may also vary depending on the model.

The utility confirms the POS bit settings slotwise, for the various NamedItem(s) in the ADF file.

The source file is available under the file name "rdpos.c" on the Altera Bulletin Board Service.

Figure 2 contains the output listing as obtained 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:

Slot #1-5	Empty
Slot #6	Altera MultiFunction Card
Slot #7	Empty
Slot #3	HardDisk

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

ADAPTER HARDWARE DESCRIPTION

The block diagram in Figure 3 shows the overall structure of the Multi-Function card design. The board has 32Kx16 memory, a serial communications chip (AmZ8530) and the Altera Micro Channel Bus Interface and DMA arbitration chips, EPB2001 and EPB2002. The common Micro Channel Bus lines are shown on the right. The EPB2001 provides the POS register functions mandated by the IBM specification for any add-on card interface. In addition, board control logic, board I.D. storage and address decoding are integrated onto the chip. The EPB2002 provides DMA arbitration and fully supports the IBM bus exchange protocols.

The Altera Multi-Function adapter card is designed so that it appears to the PS/2 as either an 8-bit I/O peripheral (for accesses to the 8530 Serial Communications Controller), or as a 16-bit memory extension (32K words). The two 74AS245 transceivers buffer data during transfers between the Micro Channel and I/O or memory. 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 64K (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 using an EPB1400 EPLD is also available. The 8530 Serial Communications Controller (SCC) chip provides two 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 8530 includes handshake lines for DMA Request/Acknowledge which interface to the EPB2002. The EPB2002 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 EPB1400 EPLD (BUSTER). The OSC line provides a precise 14.31818

Figure 1. Utility to Read Adapter I.D. and POS Register Contents

```
   /* **************  (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 */

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;
```

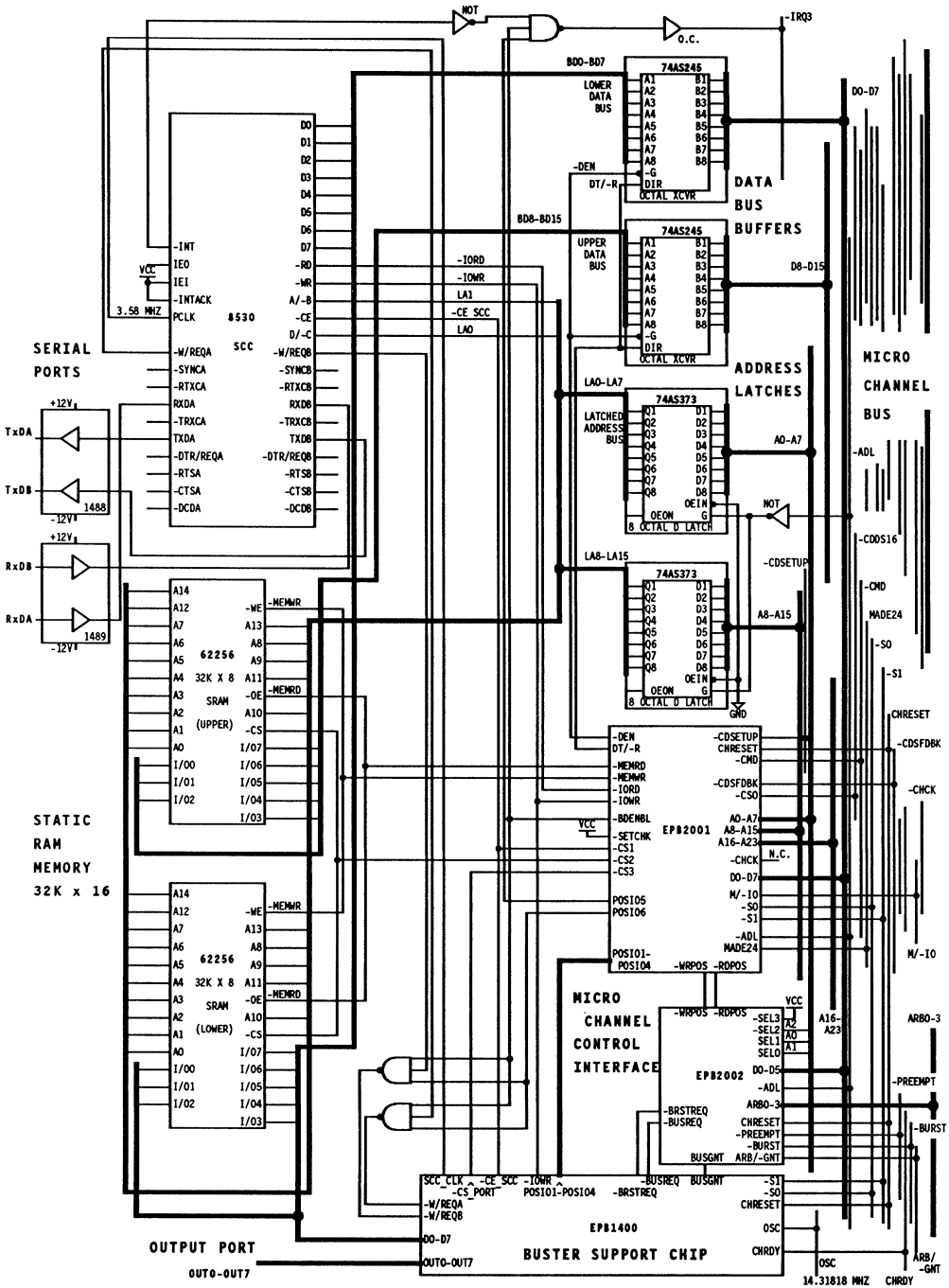
Figure 1. (Continued)

```
    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. Output Listing of Utility

```
Output Listing   (c) 1988 Altera Corp.

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
```

Figure 3. Altera Multi-Function Card Block Diagram



MHz clock. The clock divider generates a 3.58 MHz output for the SCC.

Wait-state logic associated with the 8530 is also designed into the same EPB1400 device. 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 8530 used in this design requires a 250 ns minimum width on -RD and -WR to the device. As a result, the bus cycle must be "stretched" at least 160ns. This is done by inserting wait-states to extend the cycle. 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. As such, 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 which form RS-232-compatible interfaces. In this application, the full DCD/CTS/RTS handshake has not been shown. This could be implemented by the addition of 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 provides two CMOS EPROM bytes at locations 0100-0101H for the board I.D. These are read-only locations accessible from the MC Bus. These I.D.'s are unique to a given board design and are allocated by IBM to registered Independent Developers. The developer must contact IBM directly for such I.D.'s. 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. The DMA arbitration level and the Fairness bit are also mapped into the satellite POS register bits on the EPB2002. The bit-mapping of these functions is illustrated in Figure 4. POS bits which are required outside the EPB2001 are brought-out via the POS I/O pins.

Each of the chip selects has 8 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. In this way the PS/2 can control address response ranges for the card and associated memory / I/O resources as it configures the system. This is used to eliminate address conflicts between cards without mechanical setting of DIP switches or jumpers, an error-prone process. The EPB2002 implements the bus arbitration protocol defined for the Micro Channel in an asynchronous state machine. POS register bits are included which are readable and writeable from the MC Bus. These bits control the four-bit Arbitration Level for the board (arbitration priority) and enable Fairness (Fairness is a mechanism specified by IBM to eliminate bus "hogging" by bursting DMA devices).

The bit positions and POS register location where these bits will be mapped is up to the board designer. The MC Bus specification does not define required POS locations for them. Any of POS registers 0102-0105H may be used for this purpose (except bit 0 in POS register at 0102H and bit 7 in POS register at 0105H.) The EPB2002 allows the remapping of these bits by appropriate connections to the SELx inputs to the chip. This is described in detail in the EPB2001/2002 Data Sheet. By connecting the SELx inputs as shown in Figure 3, these bits have been mapped into POS register 0104H for this application, bit locations D3-D7 on the data bus.

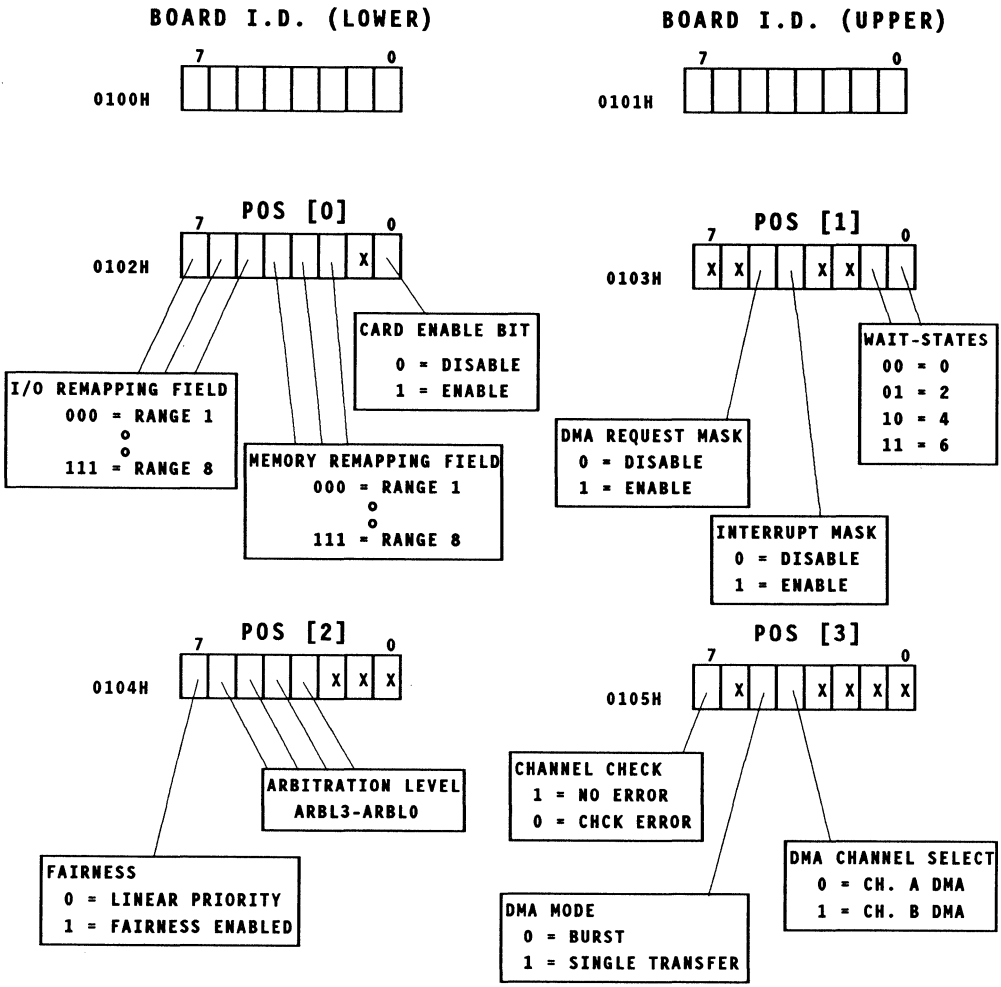
8530 INTERFACE CONSIDERATIONS

Basic signals involved in interfacing the 8530 to an MPU bus(the MC Bus in this case) 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 8530 includes 9 Read-able and 15 Write-able registers to set device configuration, report status and access data. To reduce address input pin requirements, the 8530 uses the concept of 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 subsequent access, the pointer register is automatically reset to the address pointer register. Every command register access is thus a two bus cycle process: write address pointer, 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 8530 will handshake with the EPB2002 when DMA transfers are enabled. Due to constraints in the basic DMA structure of the Micro Channel / 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 BUSTER support device. It has

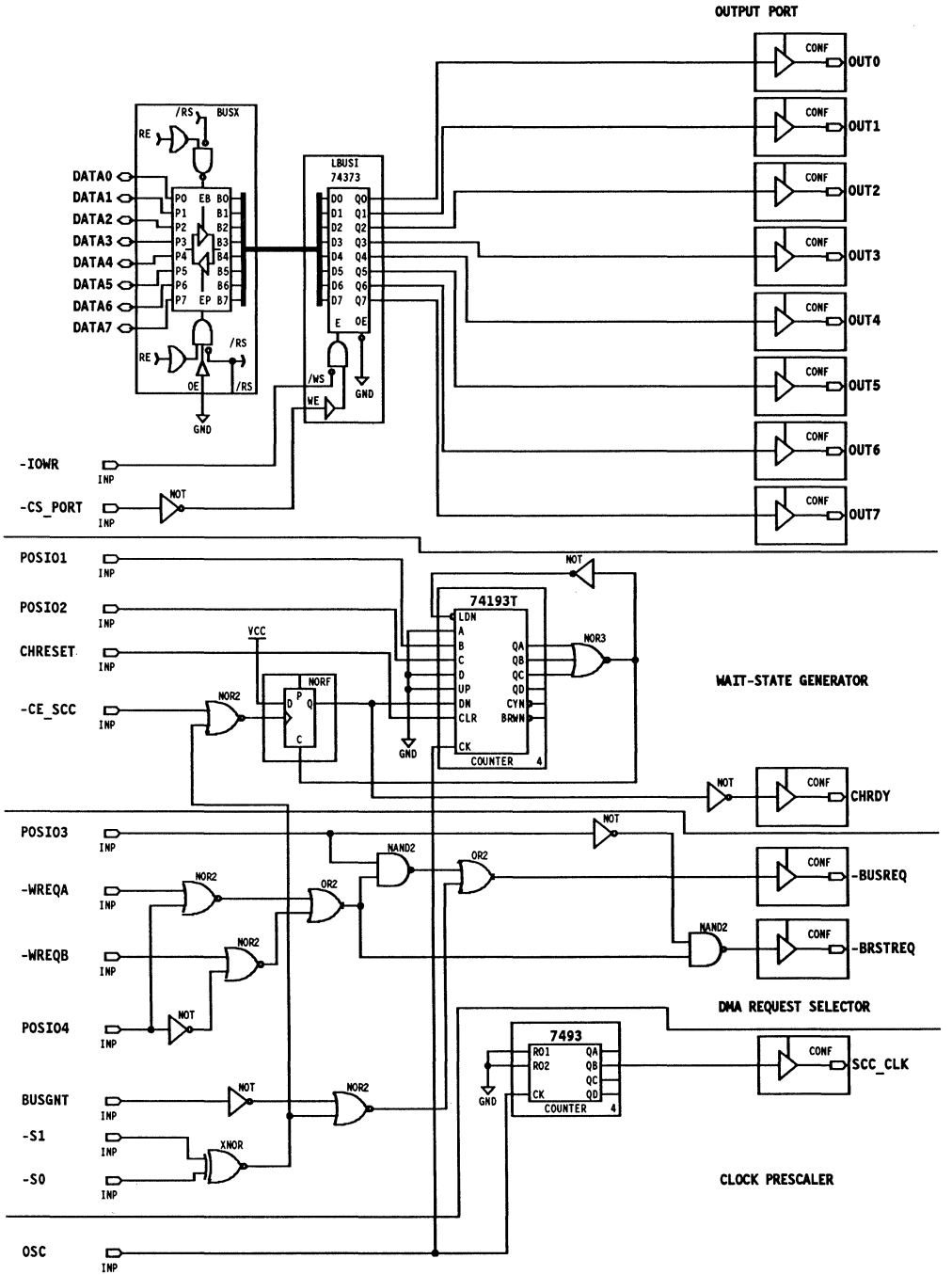
Figure 4. Multi-Function Card POS Register Bit Maps



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

Figure 5. BUSTER Multi-Function Support EPLD Design



the two -W/REQx outputs of the SCC as inputs, and generates either a -BRSTREQ or -BUSREQ line for the EPB2002. 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 8530 may be programmed to assert -W/REQx on a character receive or transmit buffer empty condition. This signal is routed to either the -BRSTREQ or -BUSREQ input on the EPB2002, and results in the EPB2002 asserting -PREEMPT. Minimal serial interfaces for Channel A & B are shown in this design. Modem control signals (-RTS, CTS, -DCD) are not used to control data transfer over the RS-232 links. This 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 Ground lines for each RS-232 line need to be connected.

OUTPUT PORT (EPB1400 EPLD)

As an added feature, an output port is constructed in a portion of the BUSTER EPB1400 for general-purpose use. These port output lines may be used to control output functions in software. This block is shown at the top of Figure 5. It is entered using Altera's LogiCaps schematic capture package and TTL MacroFunction elements. The integral MPU data port on the EPB2001 simplifies the MC Bus interface. This output port uses another of the EPB2001's -CSx outputs for selection during write operations. The -WS (write strobe) input to the EPB1400 is connected to the -IOWR output of the EPB2001. Since the output port is the only resource on the BUSTER device mapped into the I/O address space, no address inputs are needed.

ALTERA ADD-ON CARD ADF

The Adapter Description File (ADF) for the Altera adapter is presented in Figure 6. Note how 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; hence NumBytes = 4 (in the ADF).

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

Features on the multi-function card such as the number of wait-states, the SCC channel select (A or B), the mode of DMA request (Bus or Burst) and the arbitration level are software programmable, and can be selected in the System Configuration Utilities' "Change Configuration" screen. The Change Configuration screen for the ADF described herein will appear as shown in Figure 7.

NOTE: Each time the configuration is changed, the System Configuration Utilities need to be run off the back-up diskette, the changes backed-up and the computer re-booted without the configuration diskette in drive A in order to effect the changes.

A SAMPLE SOFTWARE DRIVER

The driver for the Altera Serial Port adapter has been developed using Microsoft "C" Ver. 5.0. The source code is documented to indicate how the 8530 Serial Communications Controller (SCC) and the DMA controller are configured for the application. For detailed technical information regarding the SCC, the user is referred to the 8530 Technical manual or the Data Book. Information regarding programming the DMA controller(8237) can be obtained from the IBM Technical Reference Manual for the PS/2 model under consideration.

File name: dmalpb.c (DMA transfer, SCC Local Loopback). Source code listing is included herein 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.

In the driver, a wait loop has been inserted between successsive read/write operations to the SCC control registers due to timing constraints (Refer to Section 3.2 of the AmZ8530 Technical Manual).

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.

Figure 6. Multi-Function Card Adapter Description File (ADF)

```
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."
```

Figure 6. (Continued)

```
NamedItem

        Prompt  "Memory Remapping"

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

Figure 6. (Continued)

```
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>."


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>."
```

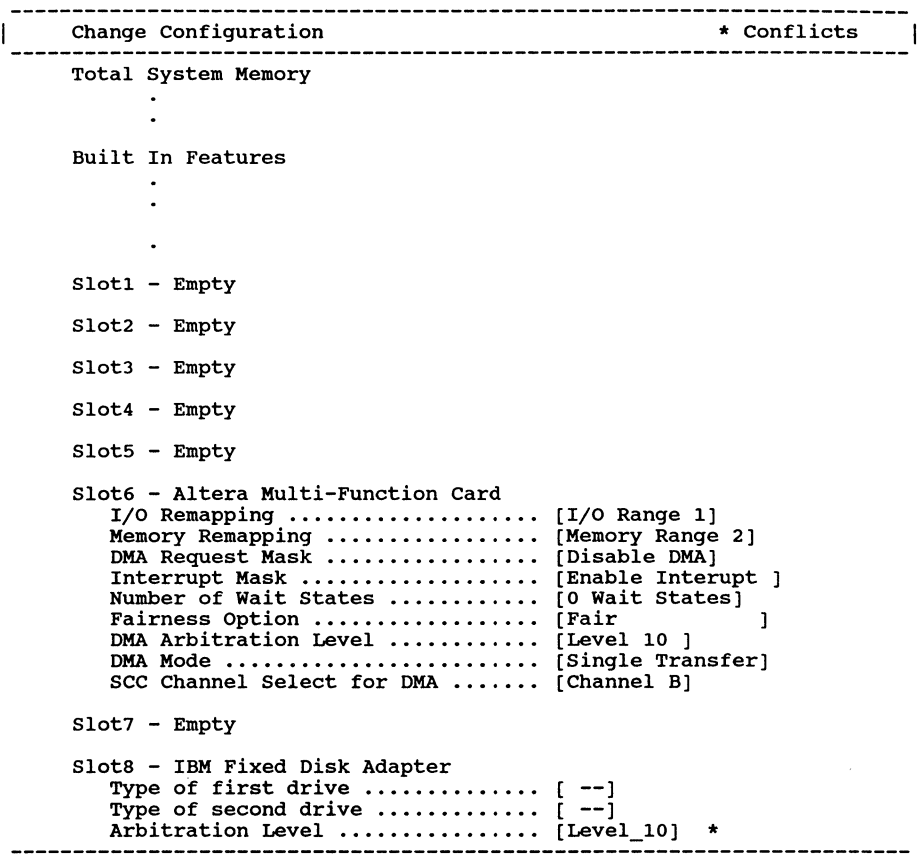
Figure 6. (Continued)

```
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. (Continued)

```
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."
```

Figure 7. A Sample Change Configuration Screen



```
/*------------(c) 1988 Altera Corporation----------------------*/
/*                                                             */
/*   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 ;
        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);
/*                                                              */
/* 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 */
/* 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 */
/*                                                    */
/* 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  */ 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  ************* */
/* ------------------------------------------------------ */
```

Figure 8. (Continued)

Figure 8. (Continued)

```
/* ------------- 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 */
/* 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);
                    /* Tx clock = Rx clock = BRG clk */

/* WRITE  TO REGISTER 12 - Low Byte of BRG Time Const*/
/*                                                         */
/* 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 - High Byte BRG Time Const. */
/*                                                         */
/* 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.*/
/*                                                         */
/* 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);
                        /* 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;
```

```
/* ------------------------------------------------------- */
/* *********** 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);

/* Initialize the rx array locations to 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 */
```

Figure 8. (Continued)

Figure 8. (Continued)

```
      outp (function_reg, 0x44);/* Write the xfer count */
      outp (exec_fcn_reg, xfer_count);    /* Low Byte */
      outp (exec_fcn_reg, 0x00);          /* High Byte */

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

      i = inp (exec_fcn_reg);
      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);

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

/* ****** Write to BUSTER output PORT, Bit 0  *****  */
      outp(port_addr, 0x00);

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

      outp(port_addr, 0x01);
                /* S/W Enable of 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);

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

Figure 8. (Continued)

```
/*     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 ");

     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*/
```

System DMA Controller (8237) - programmed using the DMA extended mode. This mode allows the DMA controller to be programmed using a Function Register and I/O addresses 0018H and 001AH so that additional read/write ports are made 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).

CONCLUSION

The EPB2001 and EPB2002 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. Coupled with other EPLDs such as BUSTER for unique application features, extremely efficient interfaces for PS/2 adapters can be rapidly designed and implemented.

WHERE TO GET MORE INFORMATION

The source code for the driver is available available under the file name "dmalpb.c" on the Altera Bulletin Board Service. The listing is also included in this Application Note under Figure 8.

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

Specifications and technical reference manuals may be obtained from IBM through

IBM Technical Directory - (800) 426-7282

REFERENCES

1. Serial Communications Controller AmZ8030/8530 Technical Manual MOS Microprocessors and Peripherals 1987-1988 AMD Data Book

2. EPB2001/2002 Data Sheet, Rev. 1.0, February,1988

3. Altera Application Note 14, "PS/2 Add-On Card Interfacing with the EPB2001 and EPB2002"

4. Z8030/Z8530 Data Sheet, Rev. E, August,1987

5. IBM Personal System/2 Model 50/60 Technical Reference (IBM# 68X-2224)

6. IBM Personal System/2 Model 80 Technical Reference (IBM# 68X-2256)

7. IBM Personal System/2 Model 50,60,80 Micro Channel Architecture (IBM# G360-2637) Rev. 5.3, May,1987

8. Altera Data Book, January, 1988

IBM PS/2 MASTER AND SLAVE ADAPTER DESIGN

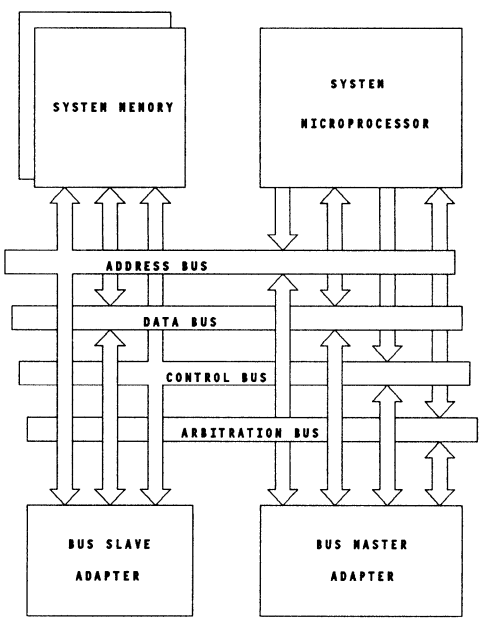
CONTENTS

- PS/2 Bus Master and DMA / Non-DMA Bus Slaves Defined
- PS/2 System DMA / Adapter Handshake
- PS/2 Bus Master Adapter Design Using the EPB2001 & EPB2002
- Using the EPB2002 with TTL/PLD or Custom Interfaces

INTRODUCTION

The concepts of bus master and bus slave are 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 EPB2002 DMA Arbitration Support device for the Micro Channel implements all the required bus arbitration protocols required for a bus master or bus slave. This Application Brief describes how the EPB2002 may be used as an essential building block in constructing a bus master peripheral adapter for the Micro Channel.

Figure 1. Bus Masters and Slaves



BUS MASTER vs. BUS SLAVE

Add-on cards or adapters on the PS/2 Micro Channel may be non-DMA bus slaves, DMA bus slaves or bus masters. The 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 to define in a general sense the terms bus master and bus slave before highlighting their specific Micro Channel differences.

A bus slave is a block of logic which can provide data to or receive 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. A bus slave is always dependent on a bus master for data transfers across the bus.

Every system has at least one bus master: the system microprocessor, or MPU. Additional bus masters are possible, depending on the bus architecture. A bus master can independently transfer data across the bus without assistance once it is granted control of the bus. These additional masters may be resident either on the main system motherboard, or on a feature add-on card, a bus master adapter. A common example of a secondary bus master found on the PS/2 motherboard would be the system DMA controller. A high-performance hard disk controller might be a bus master storage adapter.

Every bus master must have a means for generating addresses for data source and destination pointers. Many bus masters have the ability to increment memory addresses automatically as part of a block transfer to or from memory. A byte counter is also commonly found to track the required number of transfers.

Figure 1 also illustrates a typical bus master adapter interface. Notice that address and control lines, unlike a bus slave, must function as both inputs and outputs, depending on where bus ownership resides. Secondary bus masters are usually initialized by the system MPU for a given operation, and then enabled to complete the task independently. This process may involve the system MPU writing source and destination address registers, byte counter and other

Figure 2. Non-DMA Bus Slave Adapter Interface

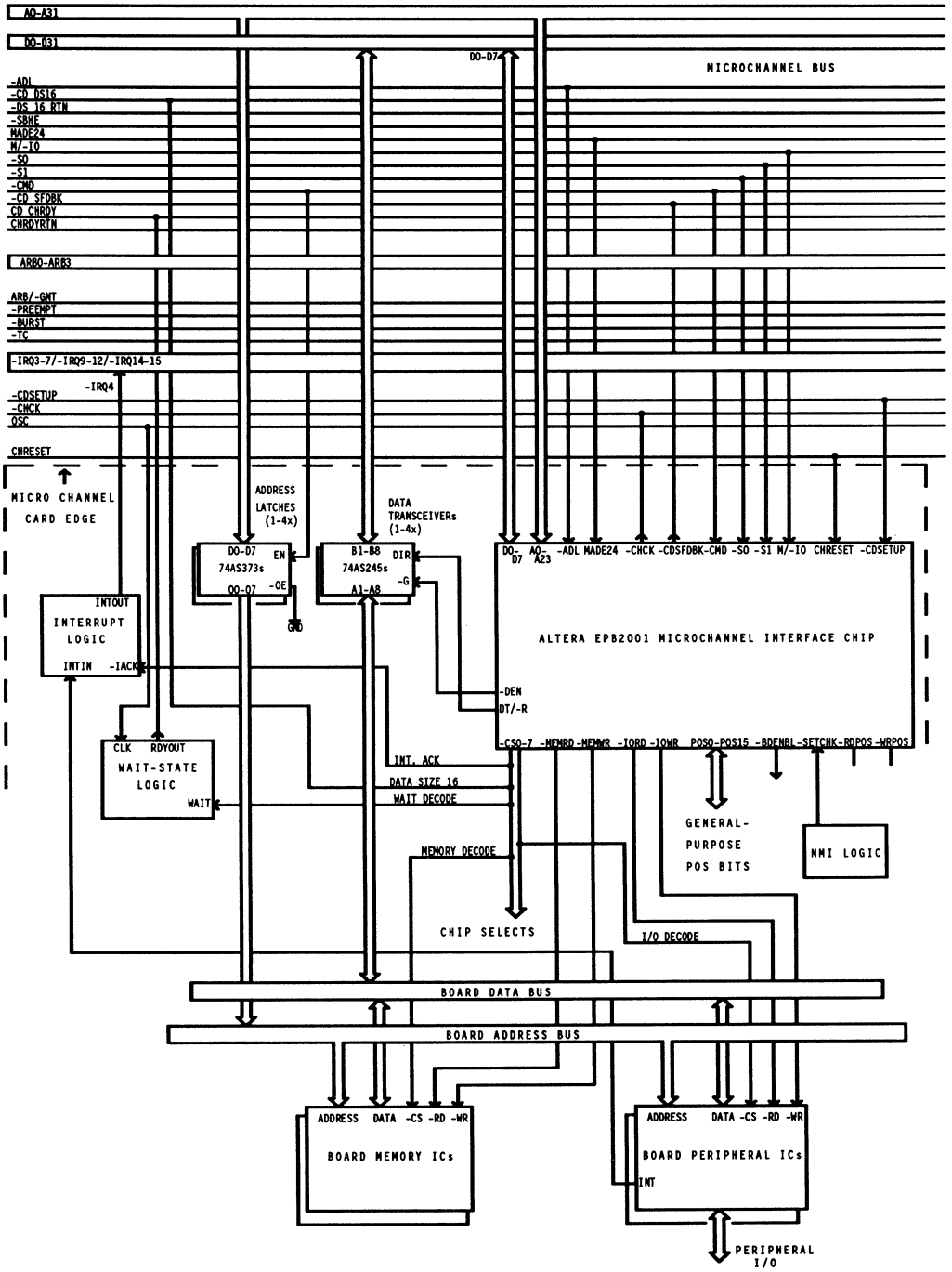


Figure 3. DMA Bus Slave Adapter Interface

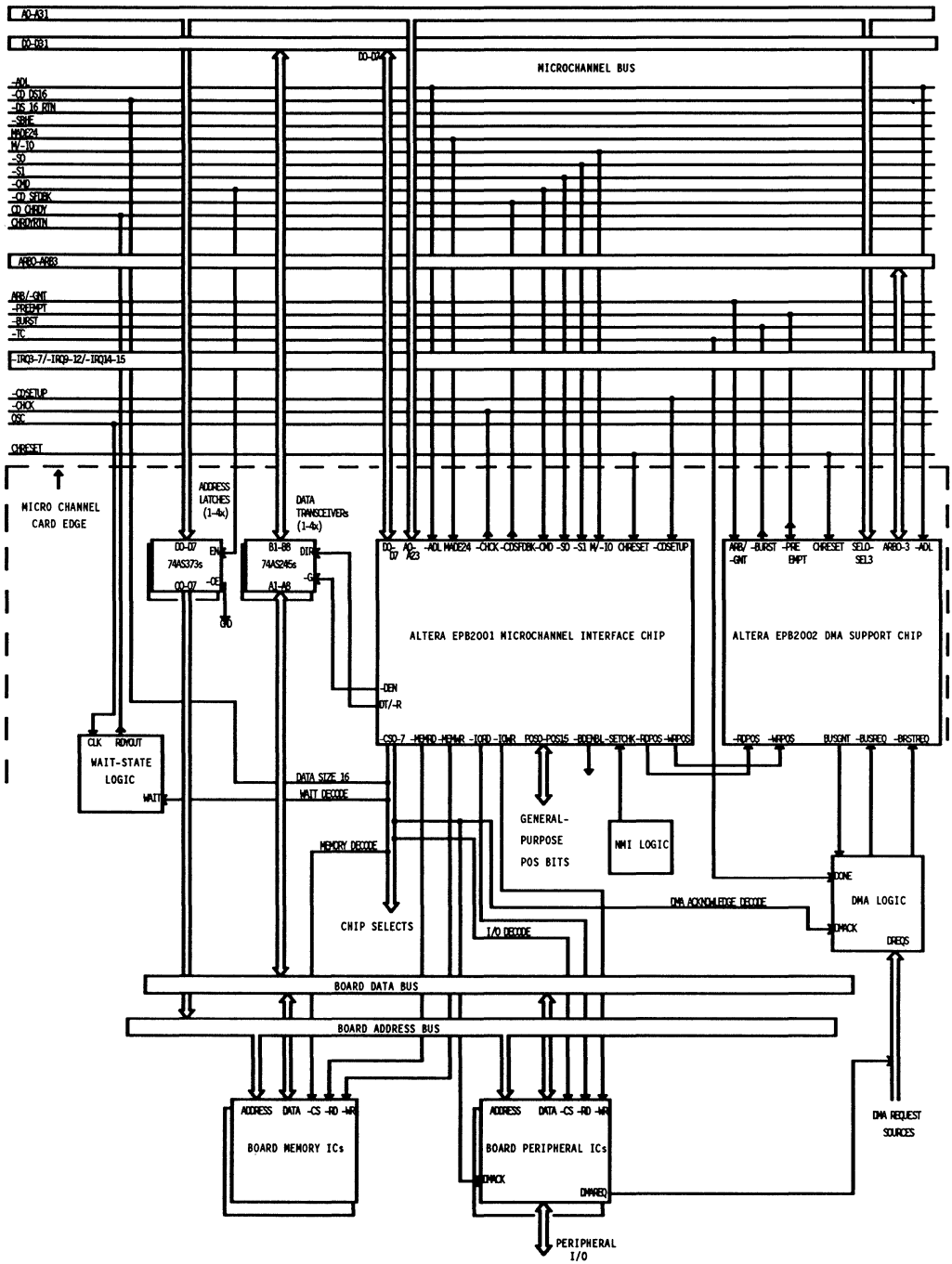
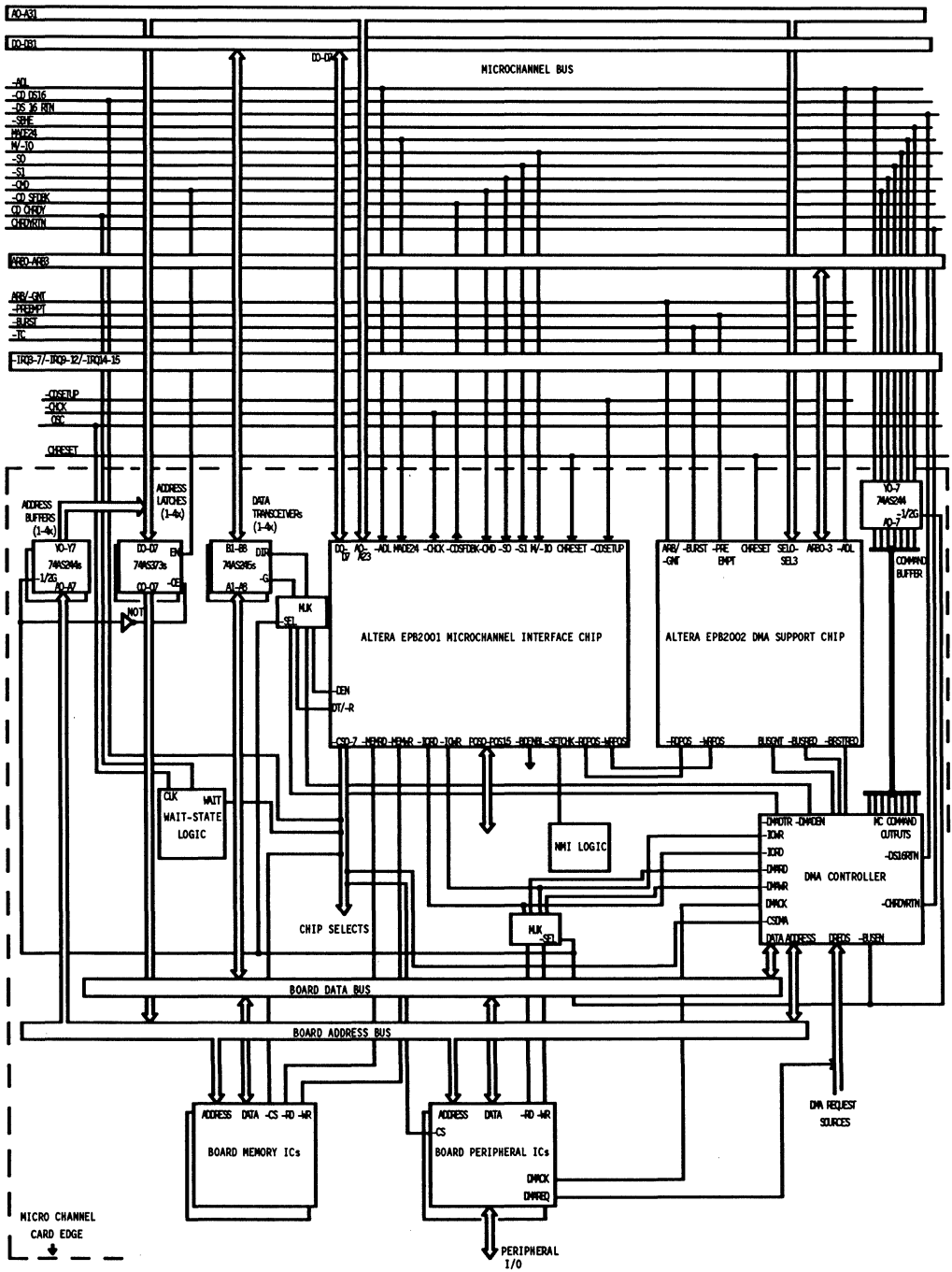


Figure 4. Bus Master Adapter Interface



transfer configuration information into the second master. While these control registers on the master are being written, the master is accessed by the MPU as a bus slave. Hence the earlier comment that a bus master's functions are a superset of a bus slave's. Rounding out the bus master block diagram, bus arbitration logic provides a means for requesting the bus, resolving priority and awarding bus control.

A bus master can interact with a bus slave by a variety of schemes. In a single bus master system, the MPU can poll status registers in a non-DMA bus slave or respond to its interrupts. Data transfers occur by executing software instructions, and is termed program I/O when dealing with a slave peripheral device. DMA bus slaves issue DMA request signals to the bus master to coordinate data transfers. However, the bus master is still in ultimate control. Data transfers are efficiently interleaved with normal MPU instruction-driven bus cycles.

The benefit of a multiple bus master arrangement is increased data transfer efficiency, resulting in higher performance. Bus master cards do not have to wait for service from another master, and can initiate and complete transfers more quickly. Another benefit is that bus cycles can be "stolen" without disturbing the MPU's program flow, decoupling I/O or other operations from the MPU's primary tasks.

In order to support multiple bus masters, a bus arbitration protocol must have a means for passing control or ownership unambiguously between different bus masters. Most bus masters require use of the bus for relatively short, infrequent periods of time but require quick response when the bus is requested. For example, a disk controller bus master will typically transfer a block of data in microseconds across the main bus. This is the result of seeking, reading and buffering multiple bytes from the disk, a multimillisecondprocess.

The bus arbitration scheme must allow for prioritization of different bus master requests since multiple requests for use of the bus may occur simultaneously or stack up. Prioritization allows more essential operations to get first access to the bus.

A bus master such as a motherboard DMA controller can execute data transfers at the request of a DMA bus slave add-on card. A DMA controller can in fact service multiple slave cards by assigning them each to a DMA Channel. Each DMA Channel has independent source and destination pointers and other configuration information assigned to it, as well as an arbitration priority. To coordinate the operation, the bus slave must alert the DMA controller to indicate its readiness for a data transfer. This is typically done through a DMA request signal routed from slave to DMA controller, or by the slave's request for use of the bus, as is done in the Micro Channel via. the common -PREEMPT line. This will be described in more detail below.

MICRO CHANNEL SLAVE INTERFACES

PS/2 NON-DMA SLAVE

Figure 2 shows a typical Micro Channel non-DMA bus slave interface. The peripheral function shown on the add-on card could be a communications, graphics, storage or other controller, or a mix of functions. The EPB2001 in this instance is used without the EPB2002. The EPB2001 provides the control interface between the Micro Channel and adapter logic, including generation of control strobes and chip select signals. Board I.D. is stored in the EPB2001, along with Programmable Option Select (POS) configuration information. (See Reference #1 below). 74AS245's are used to buffer the board data bus from the Micro Channel. 74AS373's are used to latch addresses from the MC Bus. If only a few latched addresses had been needed, as is frequently encountered in an I/O-only card, two or three of the EPB2001's CS outputs could have been used for this purpose (See Reference #2 below). The latched addresses, in conjunction with the CS outputs assigned to the memory and peripheral controller(s), allow selection of specific command, status registers or memory locations. Consequently, the system MPU can configure and interrogate the controller and access memory as needed.

Also shown in the diagram is Wait-State logic. This logic provides a delayed CHRDY signal to the MC Bus for bus cycles involving 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 MPU and bus slave in this instance is via the interrupt line -IRQ4. When the bus slave requires a data transfer, it interrupts the MPU which then moves the data as part of its interrupt service routine. This is a relatively inefficient process due to interrupt latency, but for low data rate peripherals is not a problem.

The NMI (Non-Maskable Interrupt) logic interfaces to the EPB2001 and any error-generating logic blocks on the board. Typical NMI conditions might be 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 permanently tied high.

PS/2 DMA SLAVE

Figure 3 shows a Micro Channel DMA bus slave interface using the EPB2001 and EPB2002. The PS/2 motherboard DMA controller is used to perform data transfers. The only additions to the slave adapter shown in Figure 3 are the EPB2002, DMA logic and connections to the peripheral. The EPB2002 provides the DMA arbitration functions for the add-on card. Transfer coordination occurs over the arbitration bus as outlined below, and as a result the interrupt interface is not needed. The PS/2's DMA Controller on the system motherboard services the transfer requests of the peripheral function. The configuration and interaction of these two blocks requires some explanation.

On the Micro Channel, there are no dedicated DMA request lines running from adapters to the motherboard. A bus slave must signal 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 will detect the priority code on the arbitration bus (ARB0-3) associated with that bus slave. It recognizes this as a transfer request from the bus slave and can initiate the data transfer it has been configured for. When the transfers are complete, the slave releases the bus.

In order for this scheme to work, the bus slave / EPB2002 and motherboard DMA controller must be appropriately configured. The Adapter Description File (ADF) for the bus slave card must contain the Arbitration Level to be assigned to it. This information will be loaded into the EPB2002's POS register bits at system boot time. The same Arbitration Level must be loaded into the PS/2 DMA controller's command registers for the Channel assigned to the card. Along with this information, source and destination addresses, direction of transfer, byte count, and other transfer parameters must be initialized by writing the appropriate control registers. (See Reference #3 below).

When control of the bus is granted through the arbitration process, the PS/2 DMA controller transfers data in a two-cycle process. The first cycle fetches data from either memory or I/O and places it into a holding register. Then a deposit cycle is run on the Micro Channel, moving the data from the holding register to its ultimate destination. The process is much more efficient than interrupt-driven transfers, but single cycle transfers, moving directly from data source to destination are possible and offer twice the bus transfer rates. This is the benefit of a bus master adapter as discussed below.

The DMA logic shown can include selector logic for burst or non-burst transfers, request enabling logic, data transfer counters or other useful glue functions to interface the peripheral DMA functions to the EPB2002. The -TC input from the MC Bus signals the end of a block transfer. It can be used by the DMA logic to degate the request inputs to the EPB2002 to release the bus.

In general, use of the EPB2002 is not contingent on use of the EPB2001. Since the interface between the two chips consists only of -RDPOS and -WRPOS, a Programmable Logic Device (PLD)/TTL-based or ASIC Micro Channel interface could be used with the EPB2002 providing bus arbitration support. The interface need only generate -RDPOS and -WRPOS for the EPB2002. These signals can be generated as latched decodes of -CDSETUP and I/O Read or I/O Write bus cycles, respectively, strobed by -CMD. This results in the correct read and write strobes for the EPB2002 POS register bits. (For more information, see the EPB2001/2002 Data Sheet). DMA support can therefore be added to an existing design very easily by use of the EPB2002 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 the following respects:

- A master must drive the address, data transfer control and arbitration channels.
- Signals such as CHRDYRTN and DS16RTN (DS32RTN 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.
- A master arbitrates for the channel and upon being granted the bus, it controls bus operation for one or more channel cycles. This leads to significant reduction of the bandwidth and burden on the processor, higher level of functionality and overall improved performance.

The improved functionality is achieved at the cost of higher board cost, complexity, timing constraints and increased power consumption.

A bus master adapter thus offers an alternative to using the PS/2 motherboard DMA controller. Direct, single cycle transfers are then possible for higher data rates. Figure 4 illustrates such an interface using the EPB2001 and EPB2002. In addition to the logic used for the DMA bus slave, a local adapter DMA controller

is needed to generate addresses during data transfers. Drivers (74AS244's) for the Micro Channel address bus and control bus are also needed: -ADL, -S0, -S1, M/-IO, -CMD, MADE24, and -SBHE. (See Reference #3 below). 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 as does the system MPU or motherboard DMA controller.

As can be seen, the functions provided by the EPB2001 and EPB2002 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 from the EPB2001 as inputs. These, in conjunction with the -CSDMA chip select, allow the system MPU 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 muxes shown switch control from the EPB2001 to the DMA controller when the -BUSEN signal goes low. The -BUSEN signal is asserted when BUSGNT is generated as the result of 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 has been illustrated in Figure 4, upgrading to a bus master design is achieved utilizing the EPB2002, DMA control and buffer logic.

REFERENCES

1. EPB2001/2002 Data Sheet, Altera.

2. Application Note #14: PS/2 Add-On Card Interfacing with the EPB2001 & EPB2002, Altera.

3. Application Note #15: IBM PS/2 Add-On Card Software Design, Altera.

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

NOTES

USER-CONFIGURABLE ADAPTER INTERFACE CHIPS FOR PS/2 MICRO CHANNEL

EPB2001
EPB2002

FEATURES

- User-Configurable Adapter Interface Chip Set for PS/2 Micro Channel Bus Add-On Boards:

  EPB2001 Single-Chip Adapter Interface Device for Micro Channel Bus

  EPB2002 DMA Arbitration Support Device for Micro Channel Bus

- EPB2001 CMOS EPROM Device Integrates Basic Interface Functions into a Single 84-Lead Device:

  Programmable Option Select (POS) Registers 0102-0105 with 16 Programmable Adapter Interface I/O Lines.

  Two-Byte Programmable Adapter I.D. EPROM (POS Registers 0100 & 0101)

  24 mA Micro Channel Data Bus Port (Byte-Wide).

  Adapter Address Remapping / Chip Select Decode Function Provided by Eight Chip Select Blocks with Eight Programmable Address Ranges Each.

  Adapter Control Lines Provided for Control of Board Memory, I/O and Transceivers.

  Support of Channel Check and Card Enable Functions.

- EPB2002 CMOS Device Integrates All DMA Interface / Arbitration Functions into a Single 28-Lead Device:

  Supports Micro Channel Arbitration Protocol for Slave DMA Adapters.

  User-Mappable POS Register Bits for Arbitration Level and Fairness.

  Single-Transfer and Burst Cycle Modes.

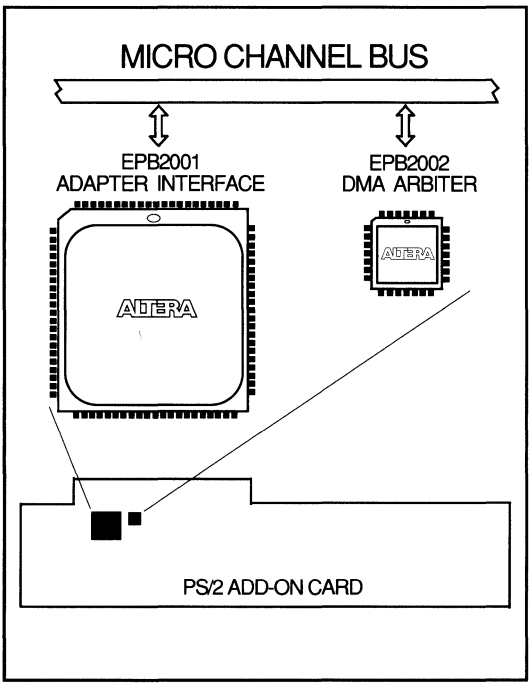
- 100% Compatible with Micro Channel A.C. Timing and D.C. Output Drive Specifications.
- EPROM Security Bit Insures Proprietary Designs.
- Quick PC-based Design Entry Using MC Map Design Software.

PRELIMINARY DATA

NOTICE: THIS IS NOT A FINAL SPECIFICATION. SOME PARAMETRIC LIMITS ARE SUBJECT TO CHANGE.

GENERAL DESCRIPTION

The Altera EPB2001 and EPB2002 are interface devices for use by manufacturers of add-on boards for IBM PS/2 computers. The EPB2001 provides in a single chip all essential interface functions required between a PS/2 add-on card, or adapter, and the Micro Channel Bus. The EPB2002 integrates optional DMA support functions needed to provide Micro Channel DMA arbitration capability. Implemented in CMOS EPROM technology, the devices provide full a.c. and d.c. interface compatibility with the Micro Channel Bus. Typically, no additional glue logic components are required. The devices are function-specific Erasable Programmable Logic Devices (EPLDs): specific board characteristics are programmable by the user for a given application. The functions integrated replace eighteen or more MSI/TTL and standard PLD devices. The 84-Lead EPB2001 and 28-Lead EPB2002 provide good p.c. board footprint efficiency (less than 2 square inches total), and may be used in tandem or independently as the design requires.



REV. 1.0

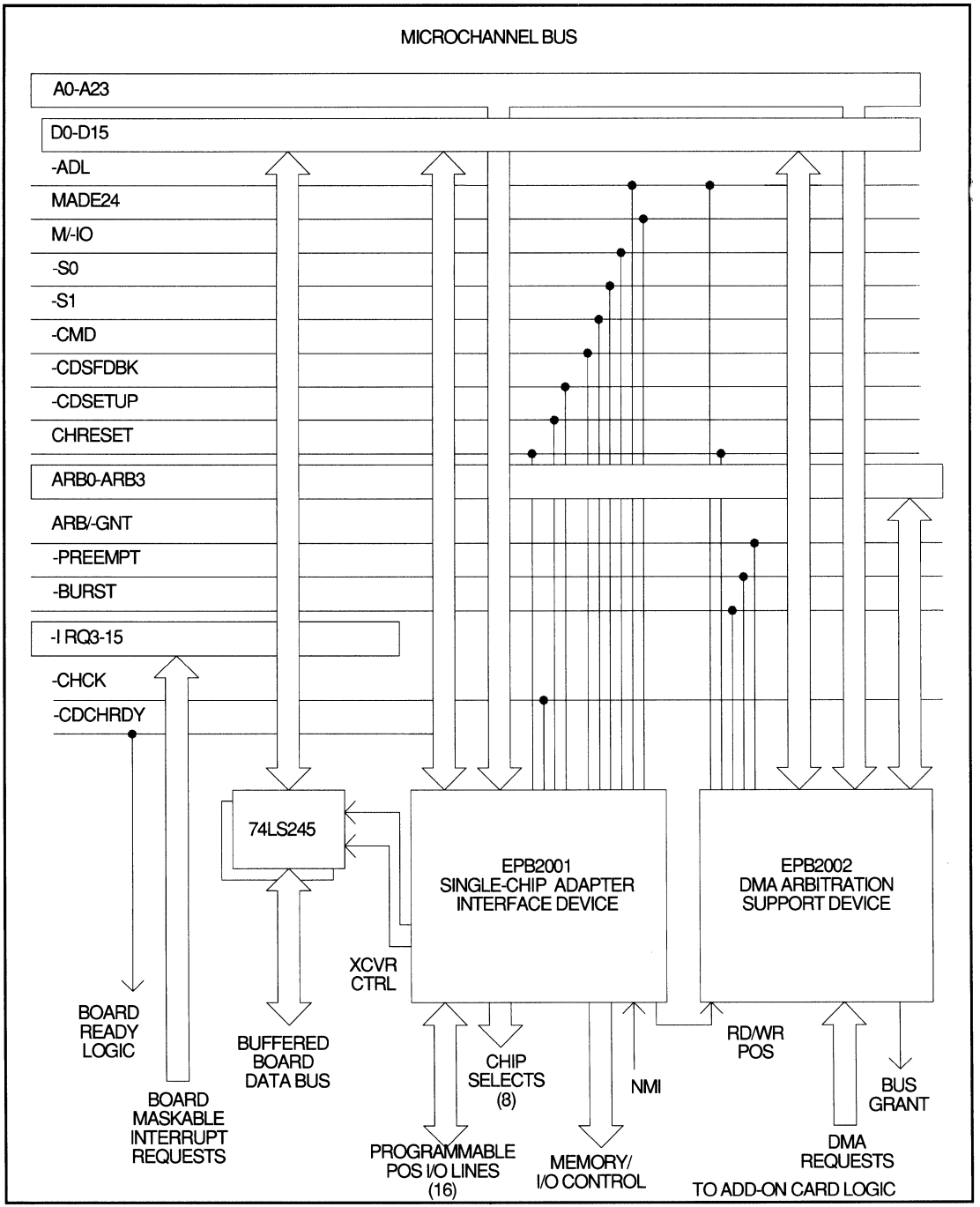


Figure 1. MICRO CHANNEL INTERFACE CONNECTION

The EPB2001 provides in a single chip all general-purpose interface functions, such as required Programmable Option Select (POS) Registers 0100-0105 (including board I.D.), access to POS Register contents on adapter-accessible I/O lines (replacing jumpers and DIP switches on the board), adapter address remapping via programmable chip select logic, and board control signals (-MEMWR,-IORD, etc.). CMOS EPROM technology is used in the device to provide non- volatile storage of board I.D., chip select ranges and POS I/O selection for reduced component count and added design security. The device is available in both One-Time-Programmable plastic and erasable/reprogrammable ceramic J-Lead chip carrier packages.

The EPB2002 provides DMA arbitration functions for those adapters requiring it. Arbitration Level POS bits, support of arbitration "fairness", burst or non-burst transfers, and full support of the arbitration protocol are all provided by the CMOS device. Pin strapping options allow mapping of the POS bits provided into any valid locations. When used in conjunction with the EPB2001, no additional components are required to provide a slave DMA adapter interface.

Programming of the EPB2001 EPROM elements is provided via PC-based MC Map Design Software. This quick, easy-to-use table- driven software leads the designer through a series of design menus. The resulting design is converted to a JEDEC file. The EPB2001 can then be programmed in seconds using Altera's PLP4 programming card, PLE3-12 programming unit and PLEJ2001 device adapter.

EPB2001 FUNCTIONAL DESCRIPTION

BUS CONTROL SECTION

A block diagram of the EPB2001 chip is shown in Figure 2. Micro Channel Interface signals are shown on the left side of the diagram, board interface signals on the right. The upper portion of the diagram contains the bus control logic, and includes in particular 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 board side. This section generates read and write signals for the internal POS registers, as well as the board control signals noted above.

This block is "activated" by either an active -CDSETUP line in conjunction with an I/O Read or Write cycle from the processor (indicating a POS set-up or boot configuration cycle), or a valid bus cycle (I/O Read or Write, Memory Read or Write) in conjunction with -CDSFDBK active (indicating a bus cycle for **this** adapter). This assumes the board has already been enabled (see POS Register File section below). Under all other circumstances, the outputs of this block remain quiescent.

The bus control block decodes the Micro Channel Bus (MC Bus) cycles as valid combinations of the -S0, -S1 and M/-IO signals. The coding for these signals is

M/-IO	-S0	-S1	Cycle Type
0	0	0	No Op
0	0	1	I/O Write
0	1	0	I/O Read
0	1	1	No Op
1	0	0	No Op
1	0	1	Mem Write
1	1	0	Mem Read
1	1	1	No Op

The state of these lines, along with -CDSETUP and -CDSFDBK is latched by the leading (falling) edge of -ADL for the duration of the cycle. Either -CDSETUP or -CDSFDBK must be active when -ADL falls for these actions to occur.

The -CMD signal acts as a command strobe and times the generation of the appropriate control lines. Therefore, -MEMRD, -MEMWR, -IORD and -IOWR have a duration approximating that of -CMD.

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. -DEN is active during a valid Read cycle for essentially the same duration as -CMD. For a Write cycle, however, to give maximum data setup time for the board, it becomes active a short time after -ADL falls. It goes inactive after -MEMWR or -IOWR goes inactive.

The adapter setup or configuration (sometimes called POST for Power-On System Test) can occur only when -CDSETUP is active on the rising edge of -ADL (address latch input from the MC Bus) followed by an I/O read or write cycle. The rising edge of -ADL may be used to latch addresses for any type of cycle, and during setup is used to latch A0-A2 so that the correct POS register may be accessed.

The -RDPOS and -WRPOS signals are used to control optional external POS register functions. They are valid for any POS read or write operation accompanied by -CDSETUP. The timing for these signals approximates that of -IOWR and -IORD.

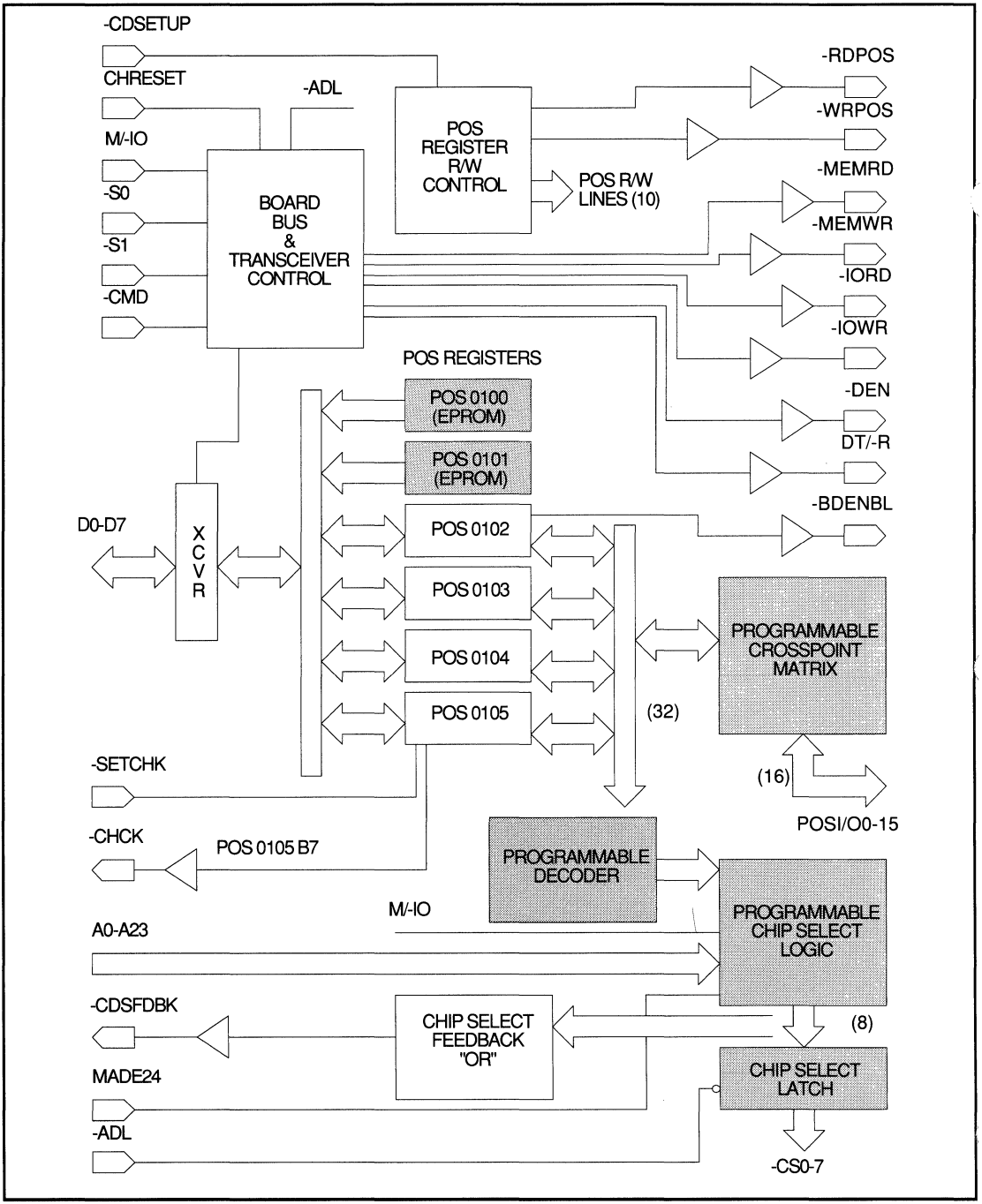


Figure 2. EPB2001 BLOCK DIAGRAM

EPB2001 Pin Description

SIGNAL	TYPE	OUTPUT DRIVE (mA) IOH	IOL	DESCRIPTION
-CDSFDBK	TP	2	6	Active LOW bus cycle acknowledge output generated by the EPB2001 for any I/O or memory cycle which activates one of the -CS outputs. Derived as an unlatched decode of Addresses, M/-IO and MADE24.
-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 (8x)	TS	4	24	Tristate bidirectional data bus lines. POS register read / write data access path. Enabled to the MC Bus only during a valid I/O read cycle
-CDSETUP	I	-	-	Active LOW set-up input. Signals a 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 earlier.) In conjunction with -CMD, used to generate board control lines (-MEMWR, -DEN, etc.).
-CMD	I	-	-	Active LOW MC 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 ouputs a short time after CHRESET rises. POS register 0102 bit 0 is also reset by this input, deactivating -BDENBL.
MADE24	I	-	-	Active HIGH input which indicates a 24 bit address is present on the MC Bus for the current cycle. When LOW, indicates an extended address (32 bits) is present.
A0-A23 (24x)	I	-	-	MC Bus address inputs. Valid while -ADL is LOW.
-ADL	I	-	-	Active LOW address latch input. Trailing 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 selected MC Bus read cycles, and LOW during selected MC Bus write cycles.

EPB2001 Pin Description (Continued)

SIGNAL	TYPE	OUTPUT DRIVE (mA) IOH	IOL	DESCRIPTION
-CS0-7 (8x)	TP	4	6	Active LOW chip select outputs. Derived as a decode of Addresses, M/-IO and MADE24. May be optionally latched on an individual basis by -ADL. Eight user-defined address ranges per output, enabled by groups of POS register bits.
POSI/O0-15 (16x)	I/OD	-	6	Bidirectional POSI/O lines. Each open drain output is driven by user-defined POS register bit. State of POSI/O pin is reflected when corresponding POS bit is read through MC Bus port. This allows board logic status reporting when POS register contents equal one (default).
-WRPOS	TP	4	6	Active LOW write POS register strobe. Active for POS set-up cycle. Used to drive EPB2002 -WRPOS inputs or control optional POS functions external to the EPB2001.
-RDPOS	TP	4	6	Active LOW read POS register strobe. Active for POS set-up cycle. Used to drive EPB2002 -RDPOS inputs or control optional POS functions external to the EPB2001.
-BDENBL	OD		24	Active LOW board enable output. Open drain output reflects the state of POS register 0102 bit 0. Active low when this register bit is set to a one. Deactivated by CHRESET.
-SETCHK	I	-	-	Active LOW set channel check input. A low pulse on this level-sensitive input resets POS register 0105 bit 7, and therefore activates the -CHCK output to the MC Bus.
VCC (2x)	-	-	-	+5 Volt Power Supply.
VSS (6x)	-	-	-	Ground.

Signal Types:
I = Input
TP = Totem-Pole (Push-Pull) Output
OD = Open-Drain Output
TS = Bidirectional Tristate Output I/O

If CHRESET is asserted at any time, any bus cycle in progress is immediately halted and the D0-D7 outputs of the EPB2001 chip tristated. Similarly, the board control lines go immediately to an inactive state.

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

The only time the EPB2001's internal transceiver (connected to D0-D7) is enabled to the MC Bus is during an I/O Read.

POS REGISTER FILE

The POS register file is accessible through the dedicated transceiver associated with pins D0-D7 on the EPB2001. Data is transferred to the selected POS register (during a write operation) while -CMD is low. The rising edge of -CMD latches the input data into the register. Data is read from the POS registers while -CMD is low, and will become valid at the D0-D7 pins within the period specified below from the -CMD leading (falling) edge.

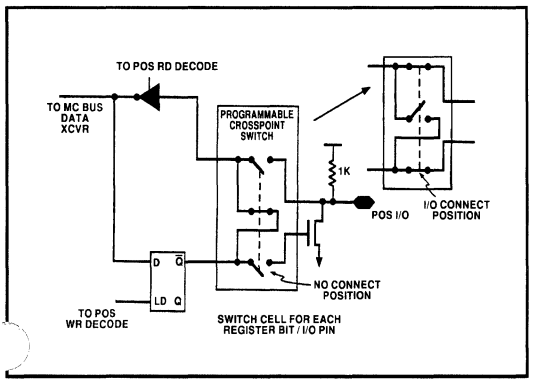
The POS register section contains required POS registers. These reside in a block at I/O addresses 0100-0105Hex for all adapters. All registers are byte-wide. Locations 0100 and 0101 are the board I.D., and are read-only non-volatile EPROM locations. POS registers 0102-0105 are user-defined, with the exception of three bit locations.

POS register 0102, bit 0 is used as a card enable bit for all adapters. This bit is reset by CHRESET, or by the

processor writing a "zero" to this bit during a -CDSETUP cycle. When zero, the EPB2001 (and adapter) will not respond to any normal bus cycles. Only setup reads and writes are allowed. When set to a "one" by the processor, the card is enabled. This bit may not be written by normal I/O Write operations to location 0102. The -BDENBL signal on the board interface reflects the state of this bit for on-board use. This pin uses a 24mA open drain output structure.

POS register 0105, bit 7 is used as a channel check flag. A card reports non-maskable interrupts (NMI) to the processor by asserting the -CHCK (channel check)

Figure 3. POS I/O CONFIGURATION



line, which is wire-ORed to all cards. On the EPB2001, a pulse on the -SETCHK input will reset this bit to a zero, which is connected to the open drain, 24 mA - CHCK output on the MC Bus. This bit is set by a CHRESET or a write to this location with a one in the bit 7 position. The bit may also be reset by a write to 0105 with zero data in the bit 7 location.

Bit 6 of register 0105 is used if channel check exception status is provided in optional POS registers 0106 and 0107. If used, these registers would typically be implemented in components such as 74LS374's. If status is available, a zero will be found in this location. If status is not provided, a one will be found there. If this bit is used for this purpose on the EPB2001 chip, one of the programmable POS I/O pins on the board interface may be used to force the appropriate value.

All remaining bits are user-defineable. These bits may be used for address remapping control or just general input or output port functions on the board (software-controlled "jumpers" or status bits). Each POS I/O pin is independently programmable as input or output, and may be assigned to any POS register bit. The remapping function will be covered in the Chip Select Logic section below.

The connection of any of the 32 POS register bits (locations 0102-0105, exclusive 0102 bit 0) with the 16 dedicated POS I/O pins on the board interface is controlled by a user-programmable cross-point switch arrangement. Each POS I/O pin has a 6mA 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 "one" is written to a given POS Register bit from the MC Bus, the associated I/O pin is not driven. This allows the I/O pin to be driven by an external signal source, and subsequently for its value to be read through the corresponding POS Register bit location. Forcing a value from the POS I/O pins does not, however, change the value in the POS Register location (see Figure 3).

CHIP SELECT LOGIC

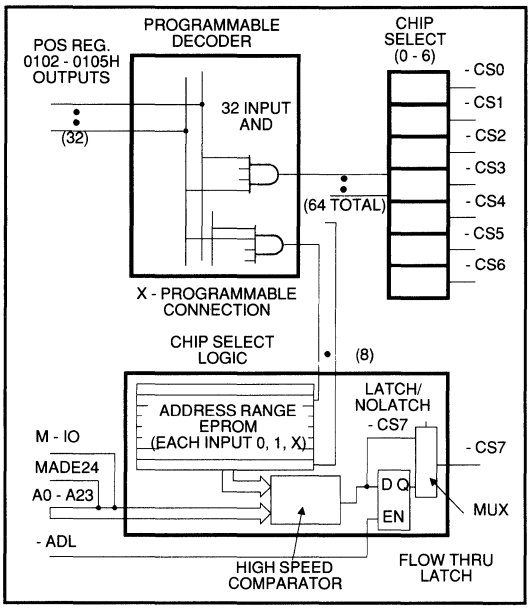
The Chip Select logic on the EPB2001 provides up to 8 user- programmable Chip Selects. Each chip select (-CS0-7) output may have up to eight pre-programmed address ranges over which it is active. The granularity of these chip selects may range from one location to the entire 24-bit (16Megabyte) available 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 logic arrays.

An additional input to the programmable chip select arrays is provided to act as an enable for the chip selects if so desired. Typically, this would be connected to the MADE24 MC Bus signal, to qualify chip selects when 32 bit addressing is involved.

Normally, chip select outputs are not latched in any fashion, and are valid only for the duration of a valid address / M/-IO combination on the bus. Optionally, the chip select outputs may be latched by user-programmable flow-through latches using -ADL. This may be done on an individual chip select basis. This results in the affected -CS output(s) going active a short time after -ADL has gone active low (the A0-A23 address lines and M/-IO input having stabilized well before -ADL -falls). The outputs are then latched on the -ADL rising edge and remain active until the next bus cycle (-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 eight distinct logic blocks (one per chip select). Each block consists of an eight word by 52 bit programmable memory (416 bits) feeding a comparator along with the address and other required inputs. Each word corresponds to a

Figure 4. CHIP SELECT LOGIC



desired chip select active range. Effectively, any input bit may be compared for zero, one or don't care in determining an address match. Thus, the need for two bits per input (26 inputs x 2 = 52 bits) to encode these possibilities.

The selection of one of the eight available chip select ranges to be used for a particular chip select output (corresponding to one of the eight words in each of the blocks) is done by user-defined combinations of POS register bits. A user may define which POS register bits, and which bit combinations, will activate a given chip select range. This information is coded into a programmable decoder on the device which generates an enable for each range. The PS/2 operating system may remap address ranges during the POST if there is a conflict, i.e., two cards which might respond to the same address ranges. This is done by changing (writing) the POS register bits controlling the chip selects, and hence enabling a new address range.

All of the chip select outputs (active low) are logically ANDed togther to form the -CDSFDBK signal presented to the MC Bus. This output signals a valid bus cycle for the adapter to the PS/2 MPU, acting as a cycle acknowledge line. -CDSFDBK is active low, and has a 6mA push-pull driver.

In the case of CHRESET active, all the chip select latches are immediately cleared to the inactive (high) state.

Each -CS output has a 6mA push-pull driver, and is active low.

EPB2002 FUNCTIONAL DESCRIPTION

The EPB2002 device performs the bus arbitration functions for the Micro Channel interface. There ar two primary sections to the chip, one the POS regis function, the other the bus arbitration state machine.

POS REGISTER

The Micro Channel Bus specification requires that each adapter interface that supports DMA functions have an MPU-programmable DMA arbitration / priority level (4 bits) and a so-called arbitration Fairness bit. These bits configure a given board's DMA function at PC boot time. These five bits may be placed anywhere in the 31 available POS Register bits, as defined by the board designer. The EPB2002 chip has five bi-directional data bus lines, D0-D4 capable of driving the Micro Channel data bus. The block diagram of the device is shown in Figure 6. Since these pins can be connected to any of the MicroChannel bus lines arbitrarily (device D0 pin could be connected to MC Bus D7), it is possible to map any of these bits into any arbitrary set of five POS register bit positions. The bit position is programmable by virtue of the connections to tl device in the actual p.c. board.

In order to program which POS byte these bits will reside in, the SEL0-3 lines come into play. These lines act as multiple chip select inputs for the EPB2002 device. 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 placed into any of the four valid POS register positions.

	MC Bus Address			EPB2002 Connection SELi			
REGISTER	A2	A1	A0	3	2	1	0
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	0	A1	A2	A0

"0" and "1" in the Connections section correspond hard strapping these pins to either ground or Vcc to obtain the corresponding mapping.

Note that all five bits must, as a result, be in the same POS Register byte.

The -RDPOS and -WRPOS signals provided by the EPB2001 chip act as read and write strobes for the

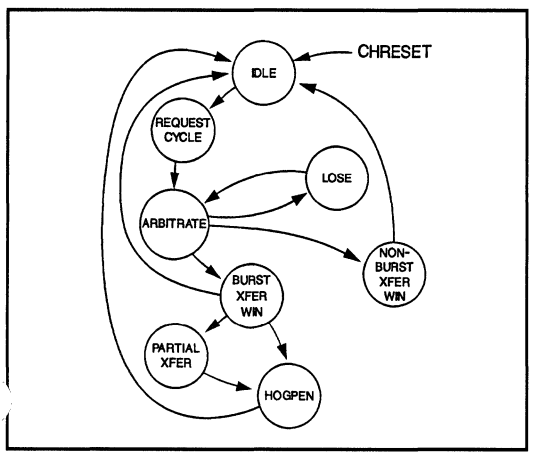
EPB2002 chip. Since there is only one five bit register on the EPB2002, any -RDPOS in conjunction with valid inputs on SEL0-3 will read the MC2 register to the D0-4 pins, and any -WRPOS signal with appropriate SEL inputs will write the EPB2002 register. -ADL latches the SEL inputs just as with addresses on the EPB2001. Timing for these operations is shown below.

ARBITRATION LOGIC

Arbitration for the MC Bus occurs during defined periods, centrally coordinated by the PC MPU. The ARB/-GNT line is the system control line which signals periods during which arbitration may occur. When high, it signals that an arbitration cycle is in progress. An adapter signals that it would like to obtain control of the bus (initiate an arbitration cycle) by driving the -PREEMPT signal low. When ARB/-GNT goes high in response, all adapters wishing bus control participate in the bus arbitration process (with one exception, see Fairness discussion below).

Arbitration levels are the means whereby the system configures the priority of a given adapter's DMA requests at system configuration. The four bits in the EPB2002's registers, as defined by the Micro Channel spec, allow sixteen different arbitration levels or priorities. Lower numbers are higher priority, so 0000 would correspond to highest priority, 1111 lowest. During arbitration, the highest priority (lowest arbitration level) adapter will win the next bus cycle.

Figure 5. EPB2002 STATE MACHINE



The basic arbitration process is straightforward. When ARB/-GNT goes high, all adapters wishing control of the bus place their arbitration levels on the ARB0-3 lines. (All adapter outputs are wire-ORed together). If a value lower that the arbitration level of a given adapter is detected on the ARB0-3 lines by that adapter, it realizes a higher priority device is requesting the bus. As a result, it releases the low-order bits of its arbitration level. This allows 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 device sees its arbitration level on the bus and has control of the bus (one cycle only in the case of non-burst mode). Devices which lose the arbitration cycle continue to assert -PREEMPT until the request is satisfied.

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

The EPB2002 device supports this operation. If the -BUSREQ line is asserted, the EPB2002 will assert -PREEMPT and arbitrate as described for the next cycle using the arbitration level programmed into its register. When the bus is granted, BUSGNT will be asserted until -BUSREQ rises signalling end of transfer.

Burst operation allows a device to hold the bus for multiple contiguous cycles. This can give greater transfer efficiency since every bus cycle does not have to be arbitrated for. The obvious danger is that a device may hog the bus, starving other adapters. The MC Bus spec specifies a fairness mechanism to avoid this. The process is as follows.

A device enters a burst transfer by arbitrating for the bus in a normal fashion and then asserting -BURST when the bus is granted. The -BURST line is asserted as long as the burst transfer lasts. The PS/2 processor will not honor any -PREEMPT bus requests while -BURST is low. As a result arbitration cycles will be suspended. The manner in which bus hogging is avoided is that the adapter which has control of the bus monitors the common -PREEMPT line. If -PREEMPT is asserted during its burst transfer, the adapter must finish-up its transfer in an orderly fashion and then release -BURST. Once -BURST is released, the PS/2 MPU is free to run a new arbitration cycle. During this arbitration cycle, the adapter releasing the bus does not participate to avoid bus hogging.

The EPB2002 supports burst operation by means of a -BRSTREQ input. If -BRSTREQ is asserted, a bus cycle is arbitrated for and when obtained, -BURST is asserted. It is assumed -BRSTREQ is asserted for as long as the burst transfer is to occur. If preemption of the burst transfer occurs, -BUSGNT will be deasserted immediately. When -BRSTREQ is deasserted in response, -BURST is deasserted. The EPB2002 arbiter will not rearbitrate until the -PREEMPT bus line has

gone high. A -BRSTREQ input after this will initiate a new arbitration cycle.

The above burst transfer discussion includes the notion of "fairness": that is, if a burst transfer is preempted, the device which is "bumped" must wait until all other arbitration requests have been satified (signalled by -PREEMPT going high) before re- entering arbitration. The device(s) waiting for this to occur are said to be in the "hog pen". This mode of operation is employed when the fairness bit in the EPB2002 control register is programmed to a one. If this bit is programmed to a zero, the burst operation reflects Linear priority. With Linear priority, a bursting adapter which is preempted

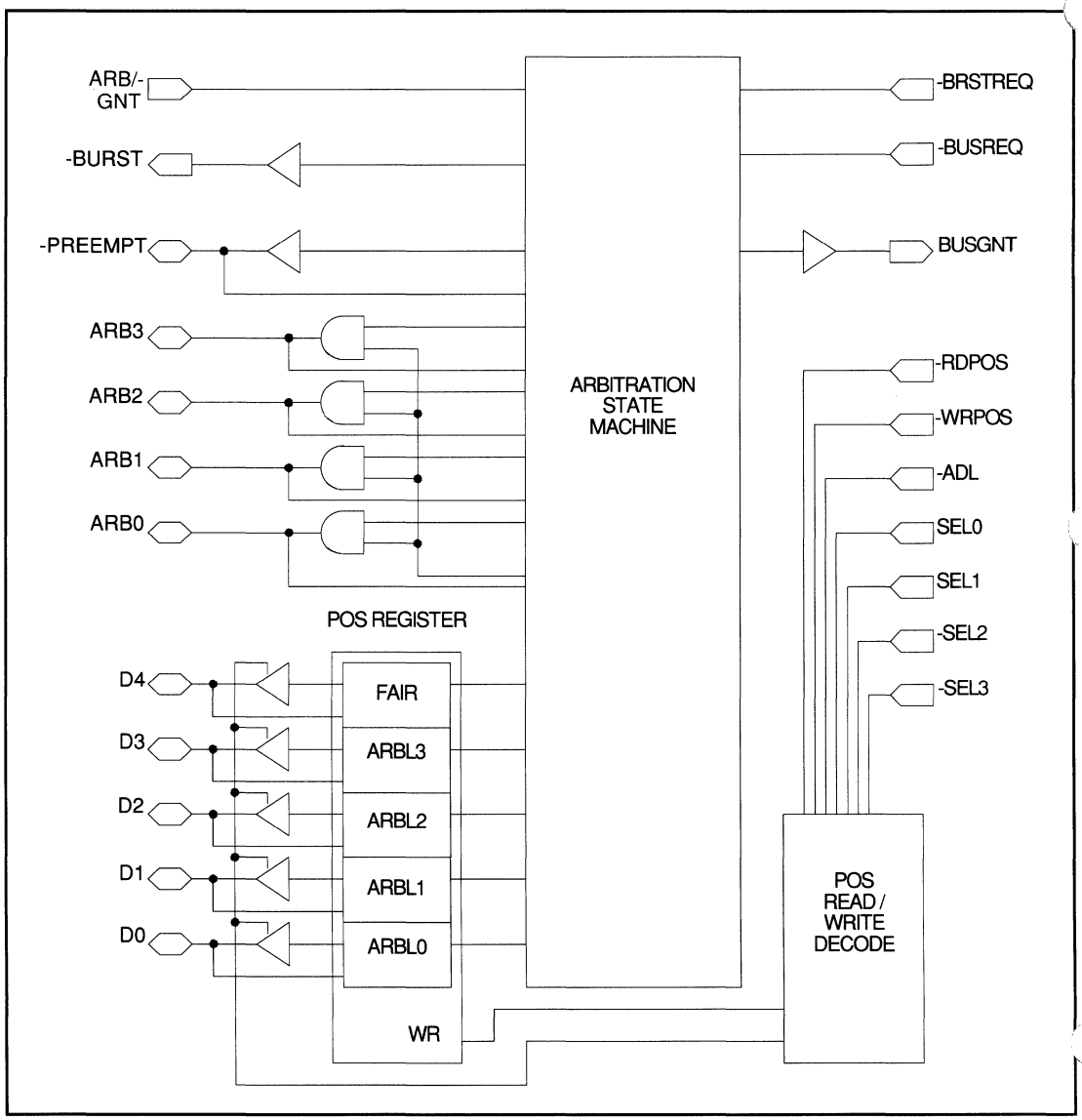
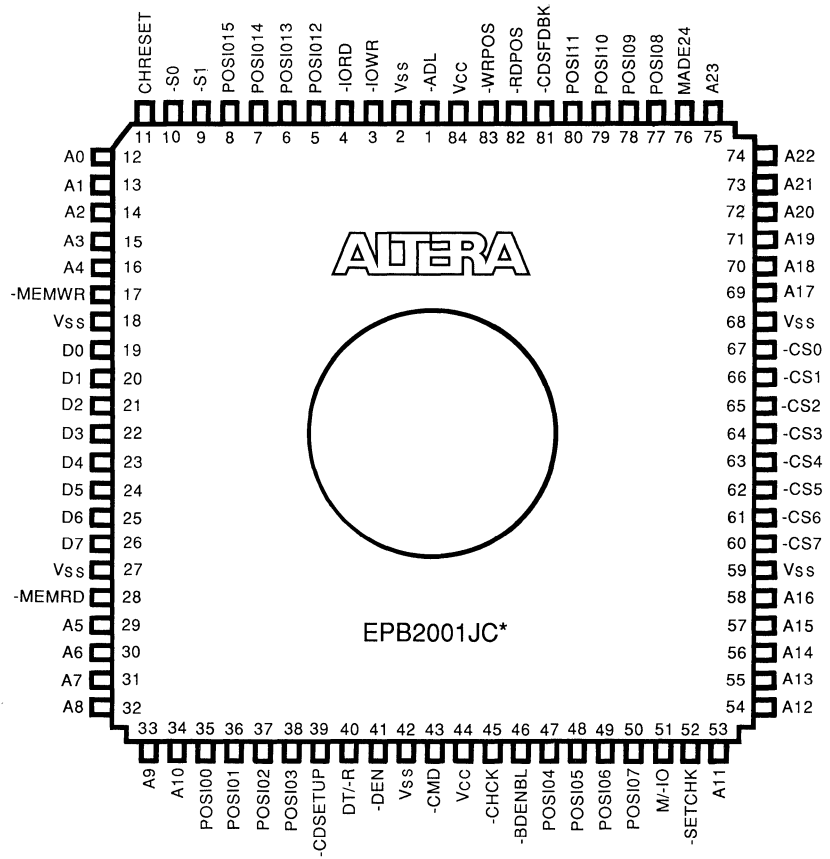
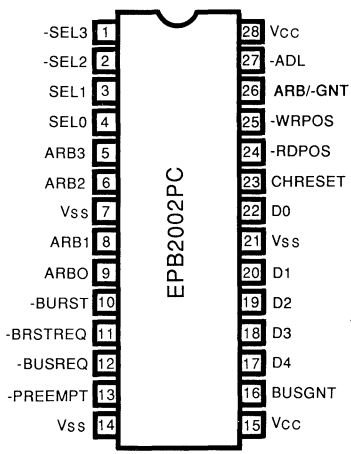


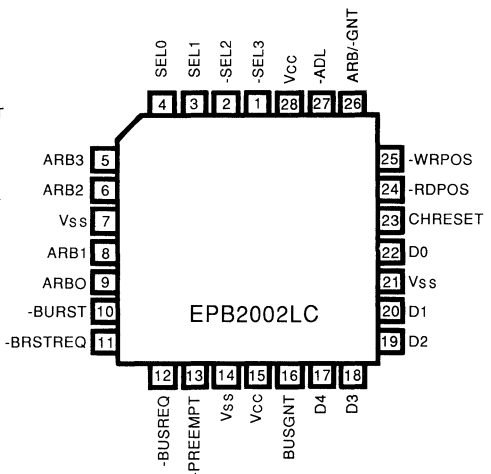
Figure 6. EPB 2002 BLOCK DIAGRAM

DEVICE PINOUTS



* EPB2001LC IDENTICAL BUT WITHOUT WINDOW





EPB2002 Pin Description

SIGNAL	TYPE	OUTPUT DRIVE (mA) IOH	IOL	DESCRIPTION
ARB/-GNT	I	-	-	Arbitration cycle strobe input from the MC Bus. When HIGH, arbitration for the MC Bus is occurring. If the EPB2002 has a pending DMA request of either type, it enables ARB0-ARB3 during this interval to compete for the bus.
-BURST	OD	-	24	Active LOW burst DMA cycle output. Asserted for the duration of a burst DMA transfer if the bus is granted as a result of -BRSTREQ active. Deasserted on CHRESET.
-PREEMPT	I/OD	-	24	Bidirectional bus preempt output and input. Driven LOW by the EPB2002 upon a -BUSREQ or -BRSTREQ input to request a bus arbitration cycle. Released when the bus is granted. Deasserted on CHRESET. If detected asserted during burst DMA transfer, indicates premature termination request from another adapter, and transfer is terminated.
D0-D4 (5x)	TS	4	24	Tristate bidirectional data bus lines. POS register read / write data access path. Enabled to the MC Bus when SEL inputs select the device and -RDPOS is asserted. D0-D3 are associated with POS register bits ARBL0-ARBL3, respectively, D4 with Fairness bit.
ARB0-ARB3 (4x)	I/OD	-	24	Bidirectional arbitration bus lines. Initially, adapter's arbitration level is output 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. Trailing 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 rising edge.
CHRESET	I	-	-	Active HIGH channel reset input. The EPB2002 deasserts all active ouputs a short time after CHRESET rises, and the arbitration state machine returns to IDLE.
-BRSTREQ	I	-	-	Active LOW burst DMA transfer request. Input is unlatched and must remain valid for the duration of the cycle request and subsequent transfer. Must be deasserted during last DMA bus cycle to terminate bus ownership.
-BUSREQ	I	-	-	Active LOW single transfer DMA request input. Input is unlatched and must remain valid for duration of the request until BUSGNT is asserted. Must be deasserted following BUSGNT to terminate bus ownership.
BUSGNT	TP	4	6	Active HIGH bus grant output. Driven active when bus ownership is won as a result of arbitration cycle. Remains active until DMA request is dropped or -PREEMPT is detected signalling early burst termination request. Goes inactive on CHRESET.
-WRPOS	I	-	-	Active LOW write POS register strobe input. Data is written from D0-D7 to the register when the chip is selected and -WRPOS is LOW.

EPB2002 Pin Description (Continued)

SIGNAL	TYPE	OUTPUT DRIVE (mA) IOH	IOL	DESCRIPTION
-RDPOS	I	-	-	Active LOW read POS register strobe. POS register data is presented on D0-D5 when the device is selected and -RDPOS is LOW.
VCC (2x)	-	-	-	+5 Volt Power Supply.
VSS (3x)	-	-	-	Ground

Signal Types:
I = Input
TP = Totem-Pole (Push-Pull) Output
OD = Open-Drain Output
TS = Bidirectional Tristate Output I/O
I/OD = Bidirectional Open-Drain Output I/O

MAY reenter arbitration on the first available cycle. Obviously, the risk of hogging the bus is now present. The EPB2002 employs the fairness algorithm by default.

DESIGN RECOMMENDATIONS

Operation of the devices described herein 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 EPB2001 and EPB2002 contain circuitry to protect device pins from high static voltages or electric fields; however, it is advised that normal precautions 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 shorted together directly at the device. Power supply decoupling capacitors of at least 0.2 microfarad 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 and EPB2002 input, output and I/O pins have been designed to resist electro-static discharge (ESD) and latch-up damage. Each of the device pins will withstand voltage energy levels exceeding those specified by MIL STD 883C. Pins will not latch-up for input voltages between -1V and V_{CC} + 1V with currents up to 100mA. During transitions the inputs may undershoot to -2.0V for periods less than 20nS. Additionally, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

PROGRAM ERASURE

Erasure of the programmed connections on the EPB2001 begins to occur on exposure to light wavelengths shorter than 4000 Angstroms. Sunlight and certain types of fluorescent lighting emit wavelengths in the range of 3000 to 4000 Angstroms and can erase a windowed EPB2001 (plastic packaged devices are obviously protected). Constant exposure to room level fluorescent lighting could erase an EPB2001 in approximately 3 years. Direct sunlighit thus could cause erasure in approximately 1 week. If the windowed EPB2001 is to be exposed to these conditions for extended periods, an opaque label should be placed over the window.

The recommended erase procedure for the EPB2001 is exposure to shortwave ultraviolet light with a wavelength of 2537 Angstroms. The integrated dose for erasure should be a minimum of 30 Wsec/cm^2. The erasure time with this dosage is approximately 45 minutes using an ultraviolet lamp with a 12000 microWatt/cm^2 power rating. The EPB2001 should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated exposure dose for an EPB2001 without damage is 7000 Wsec/cm^2. This is approximately 1 week at 12000 microWatt/cm^2. Exposure of the windowed EPB2001 to high intensity ultraviolet light for long periods of time may cause permanent damage.

The ratings on this page are applicable to both the EPB2001 and EPB2002 except where noted otherwise.

ABSOLUTE MAXIMUM RATINGS — EPB2001/2002

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V_{CC}	Supply Voltage	With respect	-2.0	7.0	V
V_{PP}	Programming Supply Voltage	to GND (Note 2)	-2.0	13.5	V
V_I	DC Input Voltage	(Note 2)	-2.0	V_{CC}+1.0	V
I_{MAX}	DC V_{CC} or GND Current		-500	+500	mA
I_{OUT}	DC Output Current per Output Pin		-25	+25	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 — EPB2001/2002

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V_{CC}	Supply Voltage		4.75	5.25	V
V_I	Input Voltage		0	V_{CC}	V
V_{OUT}	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}= 5V±5%, T_A = 0-70 °C) EPB2001/2002

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 Output Voltage	See Table 1 & 2	2.4			V
V_{OL}	LOW Level Output Voltage	See Table 1 & 2			0.50	V
I_I	Input Leakage Current	V_I = GND-V_{CC}	-10		+10	uA
I_{OZ}	Output Hi-Z Leakage Current	V_O = GND-V_{CC}	-10		+10	uA
I_{CC}	V_{CC} Supply Current	V_I = GND-V_{CC}				
EPB2001		No Load				mA
EPB2002						mA

AC CHARACTERISTICS (V_{CC}= 5V ±5%, T_A= 0-70°C) EPB2001

SYMBOL	DESCRIPTION	CONDITIONS	MIN	TYP	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				20	nS
T10	-CMD HIGH to -MEMRD, -MEMWR, -IORD, -IOWR, -RDPOS, -WRPOS HIGH				20	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 (READ Cycle)				20	nS
T14	-CMD HIGH to -DEN HIGH (READ Cycle)				20	nS
T15	DT/-R LOW to -DEN LOW (WRITE Cycle)				20	nS
T16	-MEMWR, -IOWR HIGH to -DEN HIGH				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			
T19	MADE24, M/-IO, A0-A23 Valid to -CS0-7 LOW				30	nS
T20	MADE24, M/-IO, A0-A23 Valid to -CDSFDBK LOW				60	nS
T21	MADE24, M/-IO, A0-A23 Invalid to -CDSFDBK HIGH				60	nS
T22	-CMD LOW to D0-D7 Valid (READ Cycle)				60	nS
T23	-CMD HIGH to D0-D7 Tristate				40	nS
T24	D0-D7 Valid to -CMD LOW (WRITE Cycle)		0			nS
T25	D0-D7 Hold from -CMD HIGH (Write Cycle)		30			nS
T26	POSI/O Input Valid to POS Data Valid				175	nS
T27	-CMD LOW to POS Register Data Valid				30	nS
T28	POS Register Data Valid to POSI/O Valid				500	nS
T29	CHRESET Width		100			nS
T30	-SETCHK LOW to -CHCK LOW				30	nS
T31	CHRESET HIGH to -CHCK, -BDENBL, -DEN -MEMWR, -MEMRD, -IOWR, -IORD, HIGH				30	nS
T32	CHRESET HIGH to D0-D7 Tristate				30	

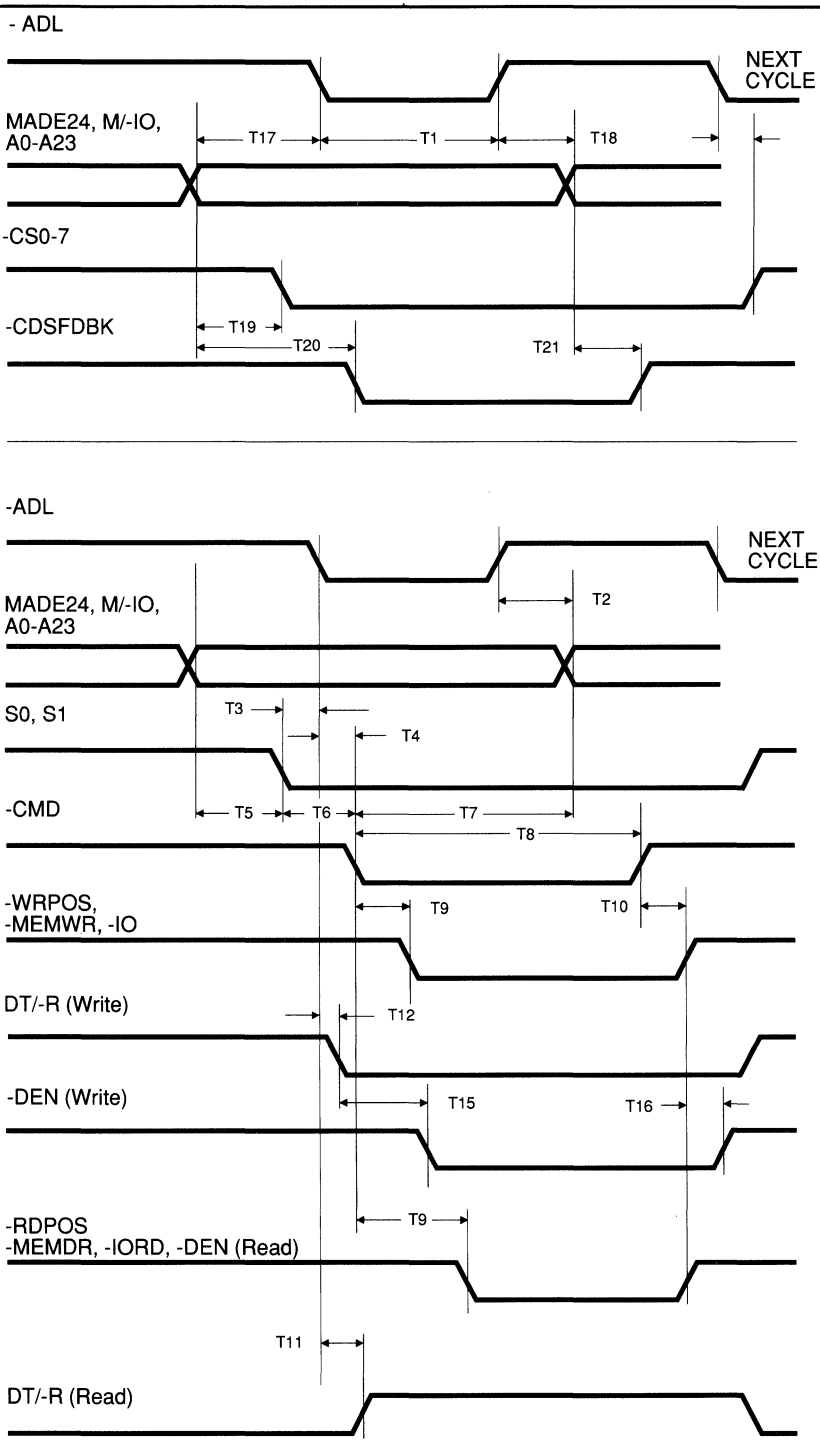
Notes:

1. Typical Values are at T_A = 25 °C, V_{CC} = 5V.
2. Minimum DC input is -0.3V. During transitions, inputs may undershoot to -2.0V for periods less than 20nS.
3. Capacitances measured at 25 °C. Sample tested only.

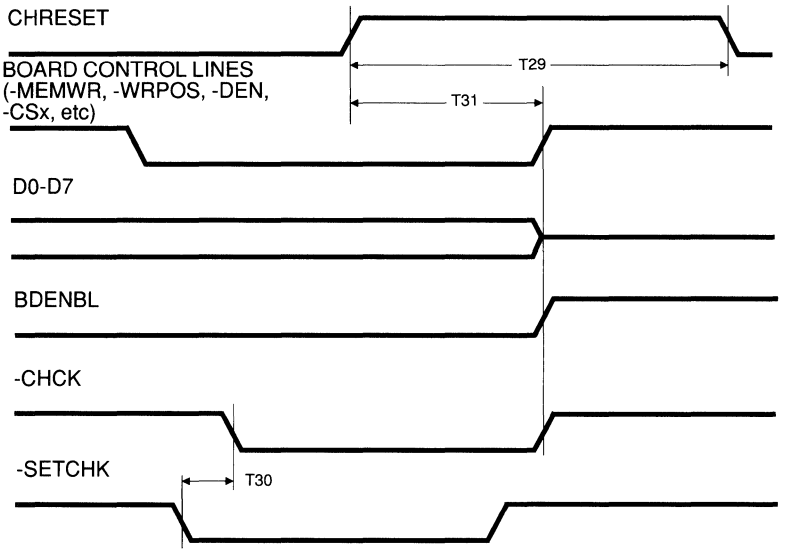
AC TIMING OUTPUT CAPACITANCE LOADINGS EPB2001

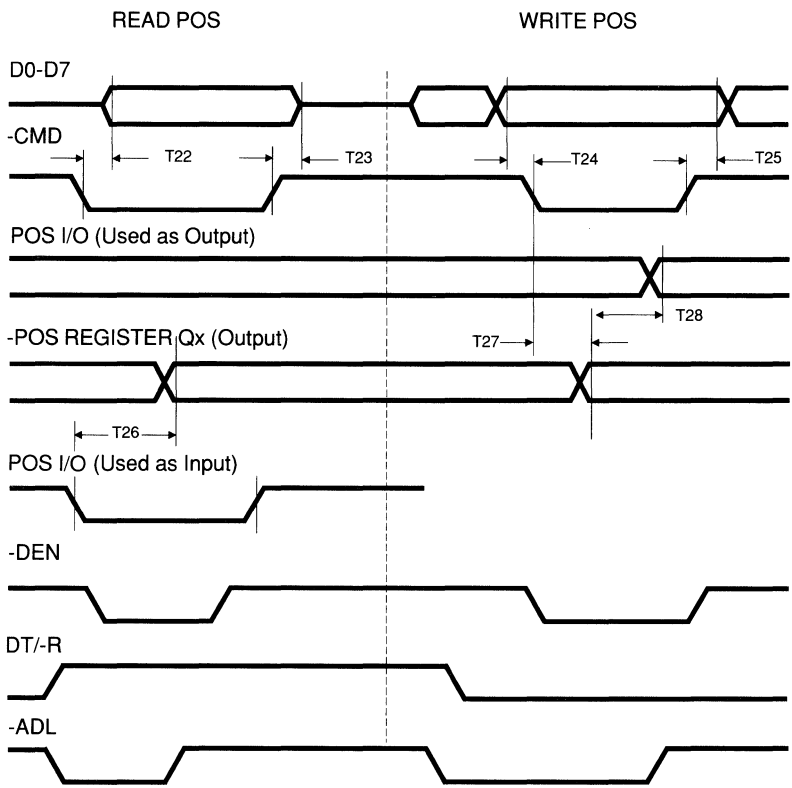
OUTPUT PINS	LOAD CAPACITANCE	NOTES
-CS0-7, -DEN, DT/-R, POSI/O0-15, -WRPOS, -RDPOS	50 pF	
-BDENBL, -MEMRD, -MEMWR, -IORD, -IOWR	200 pF	
-CDSFDBK, -CHCK, D0-D7	240 pF	

EPB2001 WAVEFORMS



EPB2001 WAVEFORMS (CONTINUED)





AC CHARACTERISTICS (V_{CC}= 5V ±5%, TA= 0-70°C) EPB2002

SYMBOL	DESCRIPTION	CONDITIONS	MIN	TYP	MAX	UNIT
T40	-ADL Width		40			nS
T41	-RDPOS LOW to D0-D4 Valid				50	nS
T42	D0-D4 Hold from -RDPOS HIGH		30			nS
T43	D0-D4 Set-up to -WRPOS LOW		0			nS
T44	D0-D4 Hold from -WRPOS HIGH		20			nS
T45	D0-D4 Valid to POS Register Output Valid				20	nS
T46	SEL0-1, -SEL2-3 Set-up to -ADL LOW		10			nS
T47	SEL0-1, -SEL2-3 Hold from -ADL HIGH		25			nS
T48	-BUSREQ, -BRSTREQ to -PREEMPT LOW				70	nS
T49	ARB/-GNT LOW to -PREEMPT HIGH				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 Valid				50	nS
T56	ARB/-GNT LOW to ARB0-ARB3 Hi-Z				50	nS
T57	ARB0-ARB3 Set-up to ARB/-GNT LOW		50			nS
T58	CHRESET HIGH to BUSGNT, -BURST, -PREEMPT, ARB0-ARB3 Inactive				50	nS

NOTES:

1. Typical Values are at T_A = 25 °C, V_{CC} = 5V.
2. Minimum DC input is -0.3V. During transitions, inputs may undershoot to -2.0V for periods less than 20nS.
3. Capacitances measured at 25 °C. Sample tested only.

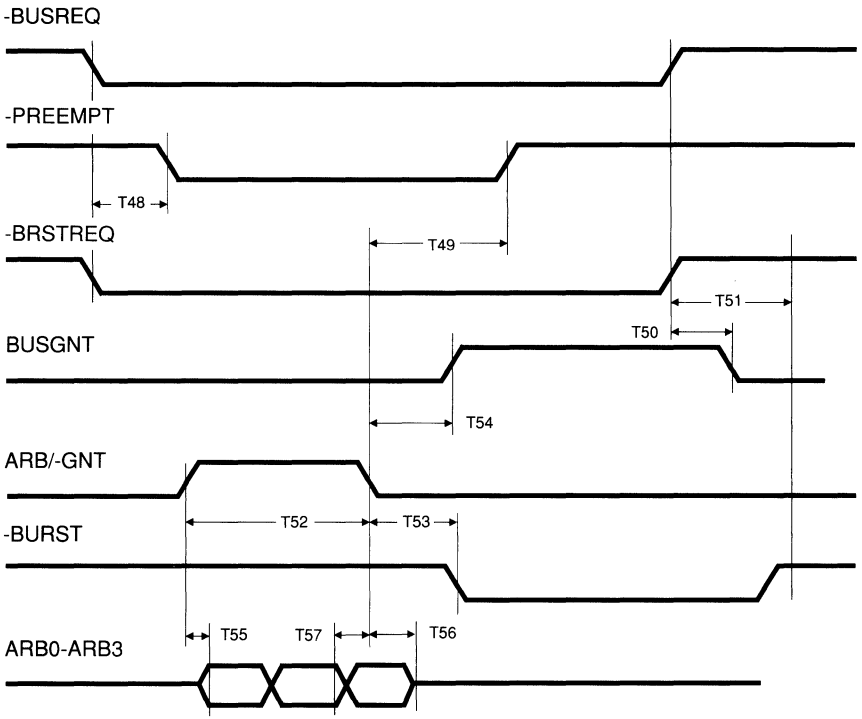
AC TIMING OUTPUT CAPACITANCE LOADINGS EPB2002

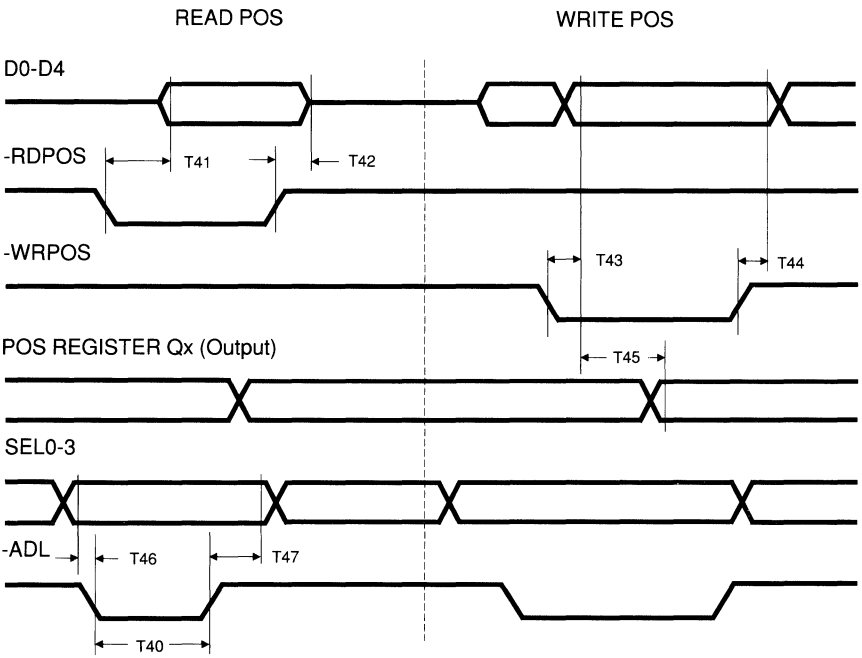
OUTPUT PINS	LOAD CAPACITANCE	NOTES
-BURST, -PREEMPT, ARB0-ARB3	200 pF	
D0-D4	240 pF	
BUSGNT	50 pF	

PIN CAPACITANCE EPB2001/2002

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
C_{IN}	INPUT PIN CAPACITANCE	V_{OUT} = 0V			pF
$C_{I/O}$	I/O PIN CAPACITANCE	f = 1.0 MHz			pF
C_{OD}	OUTPUT PIN CAPACITANCE	(Note 3)			pF

EPB2002 WAVEFORMS





The EPB2001 may be erased and reprogrammed as many times as needed within the limits described and using the recommended procedure.

DESIGN SECURITY

The EPB2001 contains a programmable design security feature that controls access to information programmed into the device. If this programmable feature is used, the custom pattern in the device is secured from external interrogation and possible reverse engineering. On an erasable EPB2001 the bit that controls this function, along with other design data stored in EPROM, may be erased using ultraviolet light as described above. This feature may be invoked by the user as part of the design entry process while using MCMap.

MC Map DEVELOPMENT SYSTEM

Altera provides a PC-based design development system called MC Map to support efficient design and use of the EPB2001. This software package features an interactive, table-driven input scheme. The designer is prompted for information concerning the programmable portions of his design: board i.d., 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, a JEDEC programming file for the EPB2001 is compiled in seconds.

Programming of the EPB2001 also occurs on the PC, using Altera's PLP4 programming card and PLE3-12 Master Programming Unit. The PLEJ2001 Programming Adapter provides an interface between this general-purpose hardware and the 84 lead EPB2001 chip carrier package. For more information concerning Development Systems, please contact Altera Corporation.

Figure 7. TYPICAL MC MAP ENTRY TABLE

Chip Select Output CS0: I/O Select : 111xxxxx

Block Size	Base Address	Register	Enable Bits
0004	0280	0102	000xxxxx
0004	0284	0102	001xxxxx
0004	0380	0102	010xxxxx
0004	0384	0102	011xxxxx
0004	0480	0102	100xxxxx
0004	0484	0102	101xxxxx
0004	0580	0102	110xxxxx
0004	0584	0102	111xxxxx

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

- IBM XT, AT or Compatible PC
- EGA (extended memory), CGA, or Hercules Graphics Adapter
- 640 KBytes RAM
- 10MByte Hard Disk and 5.25 inch Floppy Drive
- DOS Version 3.3 or later

ALTERA CORPORATION
3525 MONROE STREET
SANTA CLARA, CALIF. 95051
(408) 984-2800

ALTERA CORPORATION
3525 MONROE STREET
SANTA CLARA, CA 95051
(408) 984-2800