APPLICATION NOTE

PACE1750A
SYSTEM DESIGN

I. About This Note

This note is meant to assist system designers and programmers of PACE1750A systems with the task of designing and operating 1750A computers based on the PACE1750A chip set. The PACE1750A system, comprised of the PACE1750A CPU or PACE1750AE CPU, the PACE1753 MMU/COMBO and the PACE1754 PIC, is a very high performance CMOS implementation of the MIL-STD-1750A Instruction Set Architecture requiring a minimum amount of external logic. These products are capable of 40 Mhz operation with DAIS mix throughput in excess of 2.5 MIPS when used in carefully designed systems. This note will describe the PACE1750A/AE, 1753 and 1754 chips as well as some of the steps required to design a system utilizing the full capabilities of them. A discussion of some of the material provided in MIL-STD-1750A is included, but this note is not intended to be used in lieu of the standard. Every system designer and programmer should read the standard for a thorough description of the requirements and provisions of MIL-STD-1750A.

II. The PACE1750A System

The PACE1750A system, for the purposes of this discussion, is based on the PACE1750A Processor or PACE1750AE Processor (CPU), the PACE1753 Memory Management Unit/Block Protection Unit Combination (COMBO) and the PACE1754 Processor Interface Circuit (PIC). Very good systems can be designed using only the CPU, or the CPU with any combination of PIC and COMBO. When all three devices are used, a full implementation of the 1750A standard may be achieved with minimum power dissipation and part count. The basic interconnection of the system elements is depicted in Figure 2.1.

2.1 The PACE1750A and AE CPUs

The PACE1750A or PACE1750AE CPU is the heart of the 1750A system. It implements all of the requirements of MIL-STD-1750A, as well as many of the options. The CPU is available in 20 Mhz, 30 Mhz and 40 Mhz versions to meet the throughput requirements of a particular system. The chip is functionally pin for pin compatible with the Fairchild F9450 1750A CPU in the 15 & 20 Mhz DIP package.

As defined by MIL-STD-1750A, the CPU is a 16-bit machine capable of performing 16 and 32-bit integer arithmetic as well as 32 and 48-bit floating point arithmetic. It can address 65,536 16-bit words of memory directly or, with the optional memory management unit (MMU), can access up to 2M words of instruction and data memory. The CPU operates with a single time-multiplexed 16-bit bus used to transmit address and data information for both memory and I/O cycles. Operation of the CPU is possible at up to 40 Mhz without any system bus wait states, even with the optional MMU and/or Block Protect Unit (BPU) implemented. The capability to add wait states during either the address phase or data phase of a bus cycle is provided by the CPU to accommodate memory and I/O devices of any speed. The chip includes 24 user accessible 16-bit registers comprised of 16 general purpose registers and 8 special purpose registers.

The CPU provides the capability to receive and service up to 16 prioritized interrupts, 9 of which are accessible from outside of the chip. The internal interrupts are as defined by the standard, but the external ones may be defined for a particular system application (except for the Power Down interrupt which is also defined by the standard). Two of the internal interrupts are provided by two programmable timers, A and B, which are one of the implemented options of the standard. One of the internal interrupts is sourced by the extensive fault management system of the processor which handles a variety of internally and externally generated faults and errors.

The CPU also provides a console interface for development, test and maintenance purposes. In addition, emulation and development systems are available to assist with software development and initial system check-out.

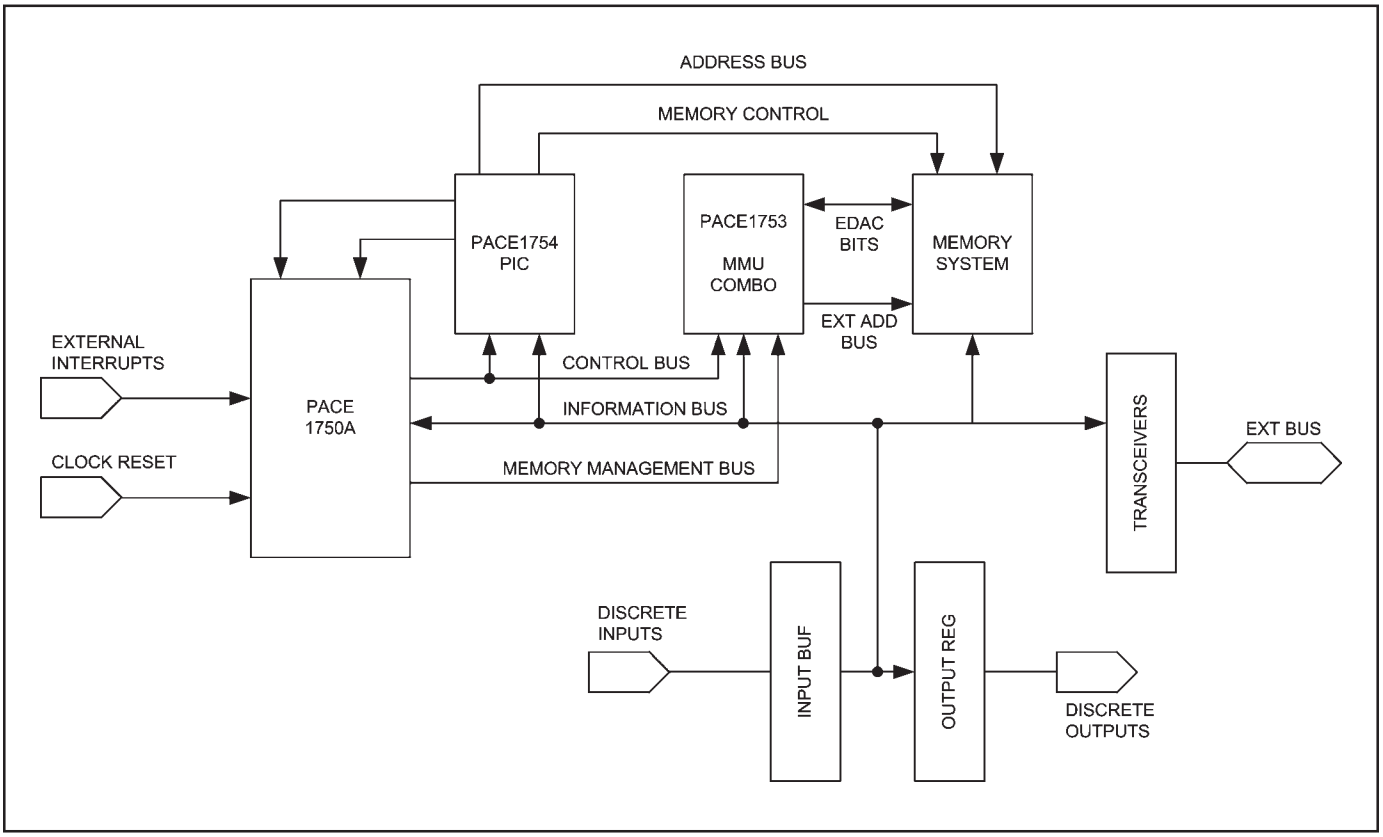


Figure 2.1 PACE1750A System

2.2 The PACE1754 PIC

The P1754 Processor Interface Circuit (PIC) is a support chip for the P1750A/AE CPUs created to assist the system designer by providing many of the functions required by most system applications in a single VLSI CMOS chip capable of the same operation speeds as the CPU. It minimizes the part count required to implement a full system. Like the CPUs, it is available in three speed grades: 20Mhz, 30Mhz and 40Mhz.

The PIC de-multiplexes the system bus, providing a high-drive 16-bit address latch to take the address off of the bus and drive it directly to the system memory. It also supports bus operations by decoding the bus control signals to provide memory and I/O read and write strobes. It can be programmed to provide the data phase wait states necessary to accommodate slow memory and I/O devices and can also provide wait states as requested by external devices on two input pins.

The PIC supports the memory system by generating and checking parity, if implemented. Both odd and even parity are supported and may be used without any additional wait states. The address of the first parity error detected is latched in an accessible register for diagnostic and fault management purposes. The PIC also supports address decoding by detecting accesses to unimplemented memory and I/O devices and providing the appropriate error signals to the processor.

Two watch-dog timers are provided to notify the system of bus time-out errors as well as other system errors which can tie up the CPU. One of the timers is a programmable trigger-go timer as specified by the standard. The other is a programmable bus watch-dog capable of detecting long or non-terminating bus cycles.

One of the most important features of the PIC is the built-in system test. With the test enabled, the PACE1750A system (CPU, PIC and COMBO) is automatically tested at power up and the result supplied to the user for self-detection of system errors. When combined with other system level testing, very high fault coverages are attainable.

2.3 The PACE1753 MMU/COMBO

The PACE1753 COMBO is a single-chip CMOS implementation of the 1750A memory management and block protection units capable of running at the full processor speeds without imposing any bus wait states. It is available, like the CPU and PIC, in three speed grades: 20Mhz, 30Mhz and 40Mhz.

The COMBO provides the extended address bits necessary to access up to 1 M word of memory, which can be expanded further to 2 M words when separate data and instruction memories are provided. It also supports the protection and access lock and key schemes defined by the standard. The block protection unit is also included in the COMBO chip to provide memory write protection in 1k word pages for both the CPU and DMA accesses as defined by the standard.

In addition to the above MIL-STD-1750A options, the COMBO provides many other useful features. Error Detection and Correction (EDAC) capability is supported over the full memory space for correction of single-bit memory errors and detection of double-bit errors and all 1's or all 0's. The COMBO generates the six required Hamming parity bits during memory writes and checks them during memory reads. If the full 22-bit memory necessary to support EDAC is not available, simple single-bit odd or even parity is also supported over the full memory address space. User accessible registers latch the logical and physical addresses of the first data error detected in either EDAC or parity mode for diagnostic or fault management capability. During EDAC mode, the position of the first corrected data bit is latched in a user accessible register also for diagnostic or fault management capability.

Like the PIC, the COMBO provides address decoding for memory and I/O accesses and flags attempts to access unimplemented addresses. The COMBO, however, unlike the PIC, is capable of detecting illegal addresses over the full 1M word address range. Also like the PIC, the COMBO can cause a wait state to occur on the system bus, but during the address phase rather than the data phase of a bus cycle. Also provided is the capability to support DMA and arbitrate the system bus among up to four masters which may be coprocessors or DMA devices.

III. Detailed Functional And Operational Descriptions

This section provides an in-depth look at the three chip PACE1750A system and its capabilities and modes of operation. Additional information may be obtained from MIL-STD-1750A.

3.1 Processor

3.1.1 Functional Description

The PACE1750A processor is organized as a highly parallel, pipelined machine comprised of six major functional blocks as shown in Figure 3.1. Three internal buses provide the necessary communication links among these blocks: the A-bus, B-bus and SHIFTER-bus. Data and address information are passed to the outside world via the 16-bit multiplexed INFORMATION BUS (IB).

EXECUTION UNIT

The execution unit performs all of the necessary data processing functions to support the instruction set. It includes the hardware necessary to perform both 16 and 32 bit integer operations and the hardware for the 32 and 48 bit floating point operations. It also includes hardware to perform fast and efficient multiplication, division, normalization and scaling. This relieves these operations from being performed in lengthy microcode routines. The 16 general purpose registers (R0-R15) and other temporary registers are also included in this block.

ADDRESS PROCESSOR

The address processor performs the necessary address manipulation for both instruction and operand accesses. Using the execution unit for the necessary calculations, the address processor provides the fetch addresses for

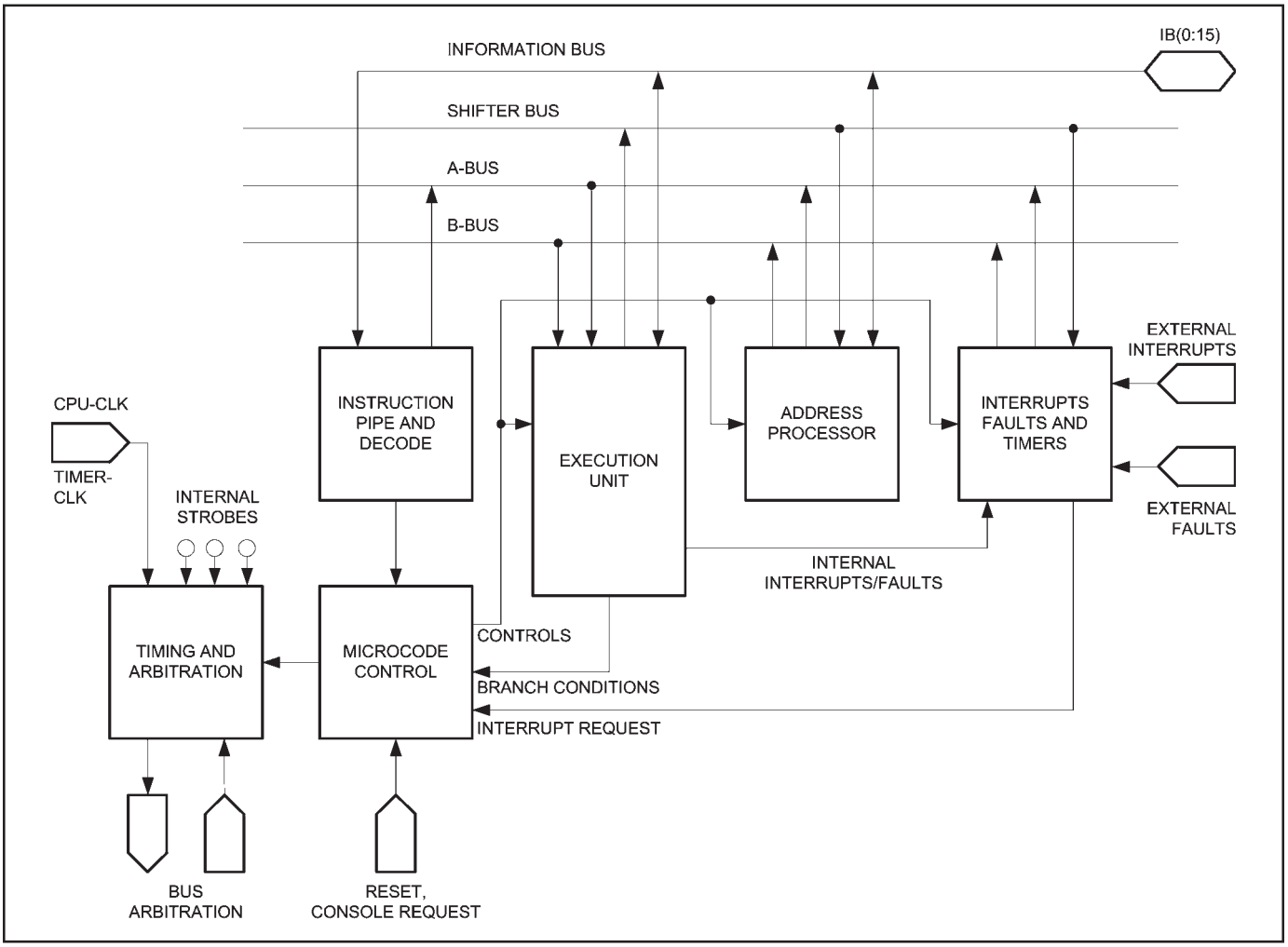


Figure 3.1 CPU Functional Block Diagram

instructions using the Instruction Counter (IC) and generates the effective addresses for operand fetching and storing. The address is placed on the IB by the address processor during the address phase of the bus cycle (when STRBA is active) and, when indirect addressing is used, the address is input to the address processor from the IB during the data phase of the bus cycle.

INTERRUPTS, FAULTS AND TIMERS

This block receives and processes the internally and externally generated faults and interrupts as well as provides the two timers (A and B) which are defined as options by the standard.

The interrupt controller accepts and processes 16 prioritized interrupts (9 external and 7 internal). The interrupts are stored in a pending interrupt register, processed through a 16-bit mask register and the highest priority unmasked interrupt passed along to the control unit if interrupts are enabled.

The two programmable timers, timer-A and timer-B, function in accordance with MIL-STD-1750A. The A-timer counts at a 100kHz rate from a 100kHz clock provided externally at the TIMER CLK input. This clock is also divided by 10 to create a 10kHz clock which drives the B-timer. Both timers generate an interrupt to the interrupt controller upon reaching their terminal counts.

The MIL-STD-1750A compliant fault register also resides in this block, latching three internally generated faults and eight externally generated faults at the end of every bus cycle as defined by the rising edge of the $\overline{\text{BUS BUSY}}$ signal. Setting a bit in the fault register also causes an interrupt to be sent to the interrupt controller. Two signals, MAJ ERR and UNRCV ER, are generated by the fault logic to allow management of faults when they occur.

INSTRUCTION PIPE AND DECODER

To increase throughput capability, an instruction pipeline is utilized allowing pre-fetch of instructions during execution of other instructions. From the instruction pipe, instructions are decoded and address pointers into the microcode store are generated. An instruction fetch is initiated whenever a word is used from the pipeline, insuring that the pipe remains full after the completion of each instruction. The pipe is flushed on the occurrence of non-sequential events such as jumps taken and context switches.

MICROCODE CONTROL

The microcode unit controls the operation of all sub-units in the PACE1750A processor. The microcode sequences are normally initiated by the starting address pointers generated by the instruction decoder; however, sequences are also initiated by the interrupt controller and external inputs such as $\overline{\text{RESET}}$ and console request ($\overline{\text{CON REQ}}$). The output of the microcode ROM is latched into the microcode register at the end of each micro (machine) cycle.

TIMING AND ARBITRATION

The timing and arbitration unit is responsible for the generation of all internal and external clocks and strobes. It also interfaces with an external bus arbiter (if implemented) via the BUS REQ, $\overline{\text{BUS GNT}}$, $\overline{\text{BUS LOCK}}$ and $\overline{\text{BUS BUSY}}$ signals to determine whether the processor has access to the bus. If not, it causes the IB, status and control lines to become high impedance. The heart of this unit is a pair of state machines, both driven by the CPU clock and one controlled by the bus arbitration signals (as described above) and the wait state control signals (RDYA and RDYD).

A typical non-bus cycle is 3 CPU clocks long and the minimum bus cycle is 4 CPU clocks long. Bus cycles may be extended indefinitely in either the address phase or the data phase by RDYA or RDYD being inactive. The state machine is designed such that bus cycle wait states do not delay simultaneously executing ALU operations unless completion of the bus cycle is required to provide data for the next ALU cycle (ie., an operand read from memory to be operated on in the next ALU cycle). In many cases, wait states may be inserted with no effect on instruction execution time since the required bus cycles are occurring in parallel with the ALU cycles.

Due to the parallelism of the P1750A architecture, in a typical machine cycle:

- A micro instruction executes in the execution unit.
- A new instruction is fetched and placed into the instruction pipe.
- The address of the next instruction fetch is prepared.
- The first instruction in the pipe is decoded.
- The microcode ROM is accessed and the output latched in the microcode register.
- A pending interrupt is processed through the mask and priority logic.
- Newly received interrupts are latched in the pending interrupt register.
- All faults present are latched into the fault register.

TIMING GENERATOR STATE DIAGRAMS

The CPU machine cycle is determined by two nearly independent state machines. The Execution Unit is governed by a three-state minimum cycle and the External Bus Unit by a four-state minimum cycle. The cycles are defined in Figures 3.2 and 3.3 respectively.

Referring to Figure 3.2 Diagram A, the paths for the Execution Unit are defined as follows:

(0) External Reset True

(1) External Reset False

(2) ALU Wait or Bus Wait

(3) ALU Branch True

(4) ALU Branch False

Referring to Figure 3.3 Diagram B, the paths for the External Bus Unit are defined as follows:

(0) External Reset False

(8) Bus Request False

(9) Bus Request True and Bus Available True

(10) Bus Request True and Bus Available False

(11) Bus Available False

(12) Bus Available True

(13) RDYA False

(14) RDYA True

(16) RDYD False

(17) RDYD True and Bus Request True and Bus Available True

(18) RDYD True and Bus Request False

(19) RDYD True and Bus Request True and Bus Available False

(20) Bus Request True and Bus Available True

where Bus Available is defined as Bus Grant true and Bus Busy false and Bus Lock false.

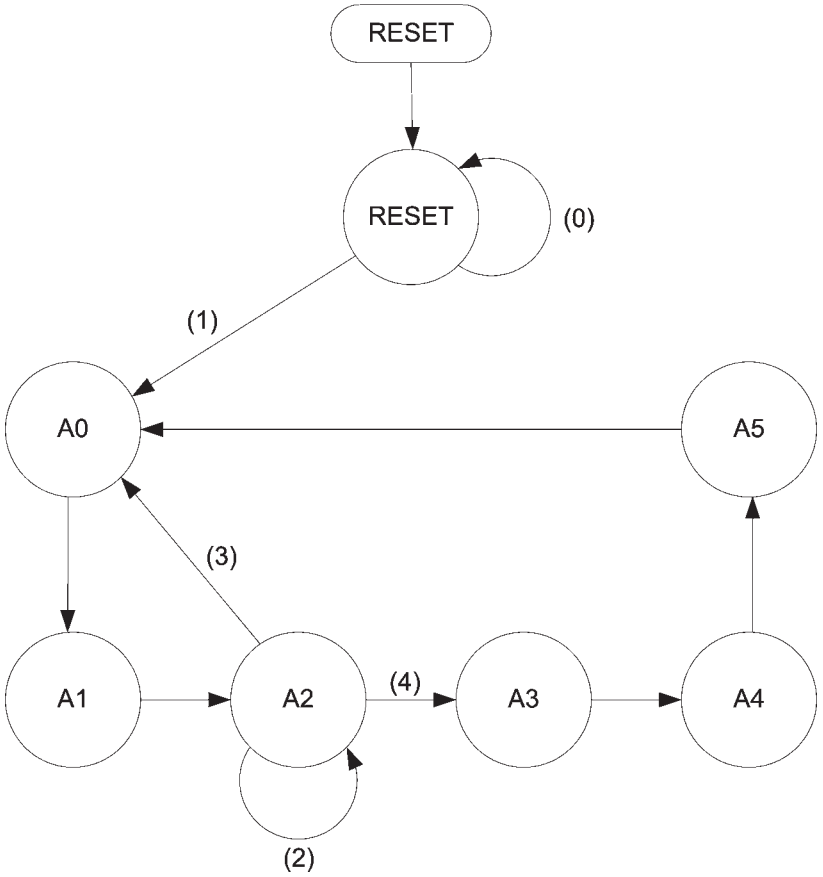


Figure 3.2 Diagram A

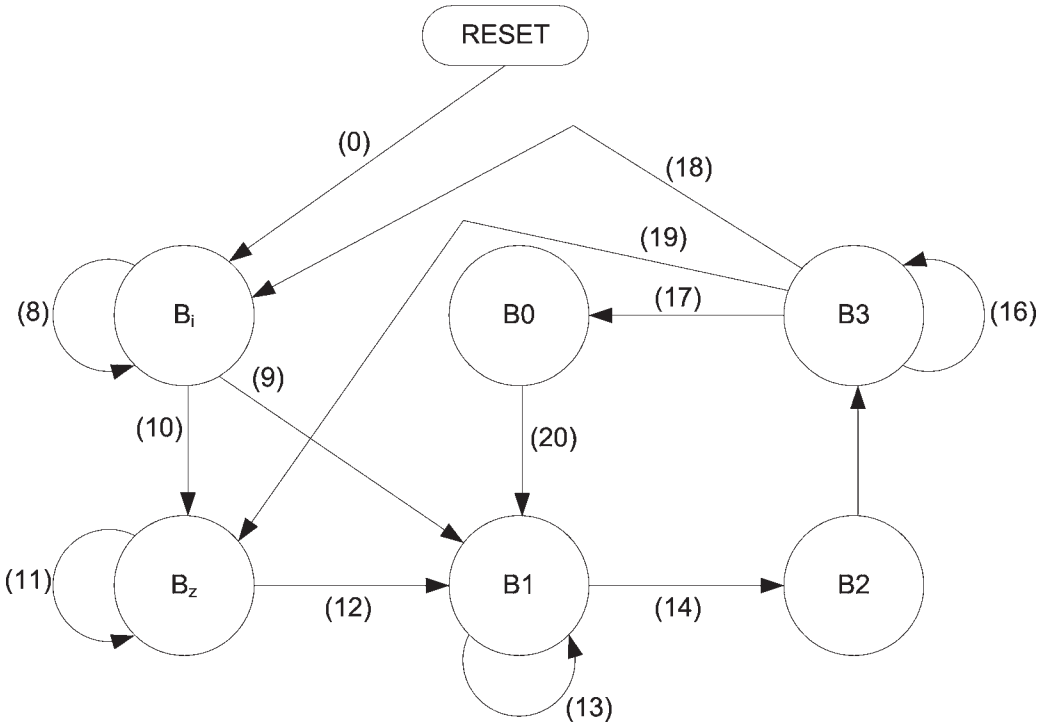


Figure 3.3 Diagram B

The Internal Execution unit includes the ALU and operates according to the Diagram A in a minimum three-state cycle. Normal ALU micro instructions are performed in that three state cycle; however, some require an extended A2 state resulting in more than one clock cycle spent in that state. Whenever an ALU branch (microprogram branch based on a current ALU result) is encountered, the machine passes from state A2 through states A3, A4 and A5 during which time the micro controller re-accommodates itself to continue.

The External Bus Unit functions within the states defined by Diagram B, passing through a minimum of four states for each cycle including one or more of B0, Bi and Bz and at least one each of B1, B2 and B3.

Following reset, the Bus Unit will proceed to Bi, a state in which no bus requests have yet been issued by the CPU. When the CPU requires the bus, it issues a Bus Request and moves to state Bz, waiting to acquire the bus, or directly to B1 if the bus is already available.

B1 can be considered the address state during which the address is placed on the bus by the CPU and STRBA is generated. If no wait states are required by the system to successfully latch the address, the CPU will spend one clock period in the B1 state. If wait states are required, as signified by RDYA being false, the Bus Unit will remain in state B1.

When RDYA becomes true, the Unit moves to state B2 during which the CPU switches from the address phase to the data phase of the cycle. If the cycle is a read, STRBD will be asserted half way through this state. If the cycle is a write, STRBD is not asserted until state B3 is reached. State B2 is always one CPU clock long.

State B3 is considered the data state during which the data is read from or written to the bus. STRBD is is asserted during this state if the cycle is a write. If no wait states are required by the system to read or write the data, the CPU will spend one clock cycle in state B3. If wait states are required, indicated by RDYD being false, the Unit will remain in state B3.

When RDYD becomes true, the Unit will proceed through one of three paths. When further bus access is not required by the CPU, it proceeds to state Bi where it releases the bus and becomes high impedance with some delay after the rising edge of the first clock following the first Bi state (during the first clock period in the Bi state, the bus will remain driven by the processor to guarantee a sufficient hold time on the bus signals). If new bus access is immediately required (Bus Request true), but the bus is not currently available because another master has gained control, the Unit will proceed to state Bz where it releases the bus, becoming high impedance with some delay after the rising edge of the first CPU clock following the first Bz state, as with state Bi described above, and waits for the bus to become available. If the CPU bus access is immediately required and the bus is available, the Unit proceeds directly to B0 and begins a new cycle without ever releasing the bus or becoming high impedance.

Internal synchronization between the A and B machines is required only at the end of A2, allowing the Units to run somewhat independently, thus operating at maximum efficiency. Moreover, in long instruction cycles, such as encountered during many of the floating point operations, the fetch bus cycle needs to be synchronized only at the end of the last A2 state, thus allowing the system to perform at very high throughputs even when relatively slow system memories are used.

3.1.2 Processor Operation

The PACE 1750A processor operates as specified in MIL-STD-1750A. This section will describe the operation of the processor in some detail; however, further information may be obtained from the standard as well as the PACE1750A data sheet.

DATA TYPES

The PACE1750A can operate on six data types:

- Bit
- Byte
- Word (16-bit)
- Double word (32-bit)
- Single Precision Floating Point (32-bit)
- Extended Precision Floating Point (48-bit)

Bit position 0 is defined as the most significant bit of all data words. The floating point numbers are represented by a normalized 24-bit two's-complement mantissa or a normalized 40-bit two's-complement mantissa for single and extended precision, respectively, and an 8-bit two's-complement exponent. Figure 3.4 depicts the floating point formats.

Figure 3.4 Floating Point Formats

REGISTER SET

The PACE1750A provides 24 user-accessible registers defined as follows:

- 16 General Purpose Registers (R0:R15)
- Pending Interrupt Register (PIR)
- Interrupt Mask Register (MK)
- Fault Register (FT)
- Status Word (SW)
- Instruction Counter (IC)
- System Configuration Register (SCR)
- Timer A and Timer B

General Purpose Registers

Registers R0:R15 are each 16-bit wide registers used for normal software operation. They may be used for any purpose by the software with the exception that R15 is usually used as the stack pointer; R0, R1, and R2 are implied operands during base addressing; and R12-R15 are address base registers during base addressing.

Pending Interrupt Register

The PIR is used to store incoming interrupts until they may be serviced by the processor. Further information is provided in the interrupt description later in this note.

Interrupt Mask Register

The MK register is used to selectively enable and disable each of the 16 interrupts received by the processor. Further information is provided in the interrupt description later in this note.

Fault Register

The FT register is used to store the 16 fault conditions monitored by the processor. Further information is provided in the faults and errors description later in this note.

Status Word

The SW is used to save the 4 processor flags, the Access Key and Address State used by the Memory Management Unit (if implemented) and the Processor State. It is a 16-bit register which may be read or modified under program control. The bit definitions are given below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	P	Z	N	Reserved				AK (PS)				AS			

Flags

C (bit 0) – Carry
P (bit 1) – Positive
Z (bit 2) – Zero
N (bit 3) – Negative

AK(PS): Access Key/Processor State

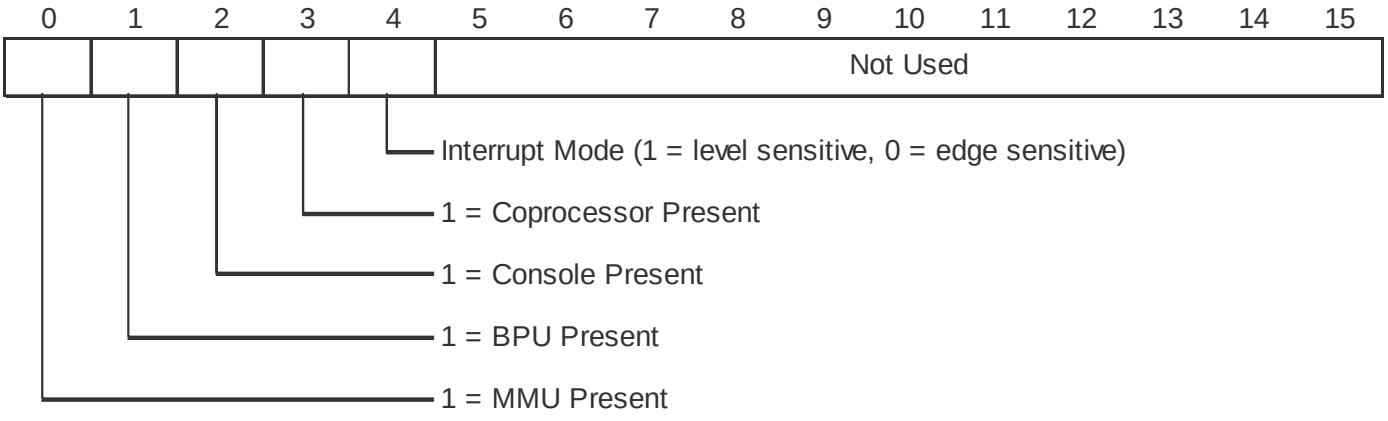
The AK bits are used to match the access lock in the MMU when implemented.
See the MMU description later in this note for further information.
The PS bits are the criteria for allowing the execution of privileged instructions. Privileged instructions are allowed to execute only when PS=0. An attempt to execute a privileged instruction when PS does not equal 0, will set bit 10 in the fault register (Privileged Instruction Fault).

AS: Address State

The AS bits determine which page register group is to be used in the MMU for memory accesses. When no MMU is implemented, any attempt to modify the AS field will set bit 11 in the fault register Address State Fault). See the MMU description later in this note for further information.

System Configuration Register (SCR)

The SCR consists of 5 bits which identify the configuration of the external system being used with the PACE1750A. The 5 bits have the meanings given below.



MMU Present

When an MMU is present, this bit should be set to a 1 to indicate to the processor that it must initialize the MMU page registers during power on initialization to the values specified by the standard. This bit also indicates to the processor whether or not it is legal to modify the address state. When this bit is a 0, it is illegal to change the address state from 0.

BPU Present

When the BPU is implemented in the system, this bit should be set to a 1 to indicate to the processor that it must initialize the BPU to all 0's during power on initialization as specified by the standard.

Console Present

When a console is present, as indicated by this bit being set to a 1, the processor will enter the console mode whenever it encounters a BPT (break point) instruction. If a console is not present, the processor will treat the BPT instruction as a NOP (no op).

Coprocessor Present

When a coprocessor is implemented, this bit should be set to a 1 to indicate to the processor that the coprocessor BIFs are implemented. The BIFs are treated as NOPs when a coprocessor is not implemented in the system.

Interrupt Mode

This bit selects the detection mode for the Power Down Interrupt and User Interrupts 0:5. A "1" causes the processor to detect these interrupts on the edge of a transition, a "0" indicates that the interrupts are to be level sensitive, sampled by the CPU clock.

The SCR is automatically loaded by an I/O read to address 8410 (hex) after reset or on the execution of a BPT instruction and cannot be modified under program control. External hardware is required to detect the I/O read to address 8410 and supply the 5 SCR bits to IB0:4 during the data phase of the read cycle. This hardware is contained in the PACE1754 PIC.

Timer A and Timer B

Timer A and Timer B are two 16-bit software programmable timers counting on 100 kHz and 10 kHz clock frequencies respectively. Each timer has a 16-bit register which may be loaded by software (via the OTA and OTB XIO instructions) with a value from which the timer begins counting up. When the timer reaches its terminal count (transitioning from FFFF to 0) an interrupt is issued. The timer then begins counting again from zero unless the software reloads them with a different value, allowing the implementation of two programmable real time interrupts. The counter values may be read on the fly by executing the ITA and ITB XIO instructions. The counters are started by execution of the TAS and TBS XIO instructions and halted by execution of the TAH and TBH XIO instructions. Both timers are halted upon entering the console mode and restarted upon resuming normal program execution.

INSTRUCTION SET

The instruction set of the PACE1750A microprocessor complies with the MIL-STD-1750A Instruction Set Architecture. Following is a summary of the instruction set features, addressing modes and input/output operations. For a detailed discussion of the instruction set, please consult the standard and the PACE1750A data sheet.

Addressing Modes

The 16-bit address field of the PACE1750A allows direct addressing of up to 65,536 words. There is no address restriction on multiple word operands in memory (the first word may reside in any memory location). The processor provides twelve addressing modes as defined below:

1. *Register Direct (R)*. The register specified by the instruction contains the required operand.

2. *Memory Direct (D)*. The instruction contains the address of the required operand in the second word of the instruction.

3. *Memory Direct-Indexed (DX)*. The memory address of the required operand is specified by the sum of the contents of an instruction-specified index register and the second word of the instruction. Registers R1 through R15 may be specified for indexing.

4. *Memory Indirect (I)*. The second word of the instruction contains the address of the memory location containing the address of the required operand.

5. *Memory Indirect with Pre-Indexing (IX)*. The sum of the second word of the instruction and the contents of an instruction-specified index register is the address of the memory location containing the address of the required operand. Registers R1 through R15 may be specified for indexing.

6. *Immediate Long (IM)*. The second word of the instruction contains the operand.

7. *Immediate Long Indexed (IMX)*. The sum of the second word of the instruction and the instruction specified index register is the required operand. Registers R1 through R15 may be specified for indexing.

8. *Immediate Short (IS)*. The required 4-bit operand is contained within the 1-word instruction. The operand may be interpreted as either positive or negative as described below:

   ISP – The operand is a positive integer between 1 and 16.

   ISN – The operand is a negative integer between -1 and -16.

9. *Instruction Counter Relative (ICR)*. This mode is used in branch instructions. The contents of the instruction counter minus one (ie. The address of the instruction currently executing) is added to the 8-bit displacement field of the instruction to obtain the address to which control may be transferred if the branch is executed. This mode allows addressing within a region of -128 to +127 words relative to the address of the current instruction.

10. *Base Relative (B)*. The contents of an instruction-specified base register is added to the 8-bit displacement field of the instruction. The displacement field is taken as a positive integer between 0 and 255. The sum points to the memory address of the required operand. This mode allows addressing within a memory region of 256 words beginning at the address pointed to by the base register. Registers R12 through R15 may be specified as base registers.

11. *Base Relative Indexed (BX)*. The sum of a specified index register and a specified base register is the address of the required operand. Registers R1 through R15 may be specified for indexing and registers R12 through R15 may be specified as base registers.

12. *Special (S)*. The special addressing mode is used where none of the other addressing modes are applicable.

Input / Output

The PACE1750A provides two modes of programmed input/output (I/O): the XIO and VIO instructions. Both instructions transfer 16-bit data to and from I/O devices addressed by a 16-bit address bus. The most significant bit of the address indicates the direction of the transfer with respect to the processor. A “1” in the most significant bit indicates a read or input command while a “0” indicates a write or output command. The remaining 15 bits of the address are used to address the device requiring data access. Thus, there are 32,768 input addresses and 32,768 output addresses each of which are accessible to the PACE1750A. The standard specifies the use of some of these addresses. The defined addresses and their uses are provided in Table 3.1. The PACE1750A system also requires the use of some of the I/O space. The addresses reserved for use by the system (CPU, PIC and COMBO) are given in Table 3.2.

The XIO command transfers a single data word to or from an I/O device while a VIO can execute a block of 16 I/O transfers between I/O devices and memory. Both of these commands are classified as *privileged*, meaning that the Processor State (PS) field in the Processor Status Word (SW) must be set to 0 prior to attempting to execute them. Failing to do this will result in a Privileged Instruction Fault (bit 10) being set in the fault register and the I/O command being aborted.

TABLE 3.1 INPUT/OUTPUT COMMANDS SET

TIMER CONTROL

Mnemonic	Function	Op. Code
TAS	Timer A Start	4008
TAH	Timer A Halt	4009
OTA	Output Timer A	400A
ITA	Input Timer A	C00A
TBS	Timer B Start	400C
TBH	Timer B Halt	400D
OTB	Output Timer B	400E
ITB	Input Timer B	C00E

INTERRUPT/DMA/FAULT CONTROL

Mnemonic	Function	Op. Code
SMK	Set Interrupt Mask	2000
CLIR	Clear Interrupt Request	2001
ENBL	Enable Interrupts	2002
DSBL	Disable Interrupts	2003
RPI	Reset Pending Interrupt	2004
SPI	Set Pending Interrupt Register	2005
RMK	Read Interrupt Mask	A000
RPIR	Read Pending Interrupt Register	A004
RCFR	Read and Clear Fault Register	A00F
DMAE	DMA Enable	4006
DMAD	DMA Disable	4007

MMU CONTROL

Mnemonic	Function	Op. Code
WIPR	Write Instruction Page Register	51XY
WOPR	Write Operand Page Register	52XY
RIPR	Read Instruction Page Register	D1XY
ROPR	Read Operand Page Register	D2XY

BPU CONTROL

Mnemonic	Function	Op. Code
LMP	Load Memory Protect RAM	50XX
RMP	Read Memory Protect RAM	D0XX
MPEN	Memory Protect Enable	4003

MISCELLANEOUS

Mnemonic	Function	Op. Code
WSW	Write Status Word	200E
RSW	Read Status Word	A00E
RNS	Rest Normal Power Up Discrete	200A
GO	Reset Trigger Go Indicator	400B
PO	Programmed Output	0YXX
PI	Programmed Input	8YXX
OD	Output Discretes	2008
CLC	Clear Console	4001
ESUR	Enable Start Up ROM	4004
DSUR	Disable Start Up ROM	4005
RIC1	Read I/O Interrupt Code, Level 1	A001
RIC2	Read I/O Interrupt Code, Level 2	A002
RDOR	Read Discrete Output Register	A008
RDI	Read Discrete Input	A009
TPIO	Test Programmed Output	A00B
RMFS	Read Memory Fault Status	A00D
RCS	Read Console Status	C001

TABLE 3.2 INPUT/OUTPUT ADDRESSES

I/O Address/Command	Input/Output	Function
8400	Input	Read console command
8401	Input	Read console data
0400	Output	Write data to console
8410	Input	Read system configuration
0800, 0900, 0A00, 0B00	Output	Write derived address to coprocessor no. 1,2,3 or 4, respectively. (used for BIF)
0801, 0901, 0A01, 0B01	Output	Write op-code into co-processor no. 1,2,3 or 4, respectively.
1000	Output	Indicates an interrupt acknowledge cycle.
1F00-1F5F 9F00-9F5F	Output Input	Reserved for Self-Test, PACE PIC and PACE COMBO

INTERRUPTS

The PACE1750A CPU provides the capability to receive and service 16 prioritized interrupts. Table 3.3 defines the interrupt priority levels. Seven of the interrupts are internal and may not be caused from outside the CPU. The remaining nine are externally sourced and provided to the CPU on input pins. These are $USER_{0-5}$, which can be used for any purpose by the system, I/O $LEVEL_{1-2}$ which also may be defined for any purpose by the system designer, and POWER DOWN which is defined by the standard as a means of notifying the CPU that power will be removed momentarily. All of the external interrupts except I/O $LEVEL_1$ and I/O $LEVEL_2$ may be programmed to be either edge or level sensitive according to the state of bit 4 in the SCR. I/O $LEVEL_1$ and I/O $LEVEL_2$ are level sensitive only.

When an interrupt is received by the CPU, it is first stored in the Pending Interrupt Register (PIR) in the appropriate bit position. The PIR is then processed through the Mask Register (MK) where it is determined if the interrupt(s) received is(are) masked. A “1” in the appropriate MK bit enables an interrupt while a “0” disables the interrupt. Then, if interrupts have been enabled previously by the ENBL I/O command, the highest priority unmasked interrupt will be serviced by the CPU after completion of the currently executing instruction. PIR/MK bit 0 is the highest priority interrupt while bit 15 is the lowest. If interrupts are not currently enabled, interrupts received will remain in the PIR until the ENBL command is executed after which the CPU will service the highest priority unmasked interrupt after completion of the instruction following the ENBL command. Two of the interrupts, POWER DOWN and EXECUTIVE CALL, may never be masked or disabled. Their respective bits in the MK may be set and reset, but they have no effect on the processing of those interrupts. The MACHINE ERROR interrupt may be masked but not disabled.

TABLE 3.3 INTERRUPT PRIORITIES

PRIORITY (PIR/MK BIT NUMBER)	INTERRUPT	LINKAGE POINTER ADDRESS	SERVICE POINTER ADDRESS	NOTES
0	Power Down	20	21	1,2
1	Machine Error	22	23	3
2	User 0	24	25	
3	Floating-Point Overflow	26	27	
4	Fixed-Point Overflow	28	29	
5	Executive Call	2A	2B	1,4
6	Floating-Point Underflow	2C	2D	
7	Timer A	2E	2F	
8	User 1	30	31	
9	Timer B	32	33	
10	User 2	34	35	
11	User 3	36	37	
12	I/O Level 1	38	39	
13	User 4	3A	3B	
14	I/O Level 2	3C	3D	
15	User 5	3E	3F	

NOTES:
1. Cannot be masked or disabled.
2. Interrupt level 0 has the highest priority.
3. Cannot be disabled.
4. Executive Call (BEX) is not a part of the interrupt priorities. It is always the highest priority.

Bits in the PIR may be selectively set and reset through execution of the RPI and SPI commands. If a PIR bit is set by the SPI command, it will be processed through the MK and serviced if interrupts are enabled just as if it had actually occurred. The PIR may be cleared in its entirety by executing the CLIR command. If the MACHINE ERROR interrupt (PIR bit 1) is cleared, the Fault Register will also be cleared.

Once an interrupt has been processed through the MK and the CPU has completed the current instruction, the CPU will begin servicing the interrupt. This is performed according to the vectoring mechanism depicted in Figure 3.5.

Each interrupt is assigned a Linkage Pointer and a Service Pointer into the memory as defined in Table 3.3. If an MMU is used, these pointer addresses will operate in the logical address space. If not, the pointers are the actual physical memory addresses. When the CPU begins servicing an interrupt, it first forces the AS to 0 if the MMU is implemented (if not, AS will already be 0). It then reads the value stored in the memory at the Service Pointer address. This value is the first address of the new three-word processor state. The CPU performs memory reads from the three consecutive locations beginning at that value. The first of these words is the new MK value, the second is the new SW value including the new AS, and the third is the new Instruction Counter (IC) value. The CPU then performs an I/O write to address 1000 with a data word equal to all 1's except for the bit position corresponding to the interrupt being serviced. This allows the external hardware to clear out its interrupt request if it was the source of the interrupt. The processor then reads the data at the Linkage Pointer address and changes to the new AS (if the MMU is implemented). The old

three-word processor state is then stored consecutively beginning at the address read from the linkage pointer address. The CPU then jumps to the new IC value and begins normal execution from there. Once the processor begins servicing an interrupt, there is a latency period of 95 CPU clocks plus 11 times the number of wait states to perform all of the above. If the interrupt was defined as level sensitive, it should be de-asserted within 35 clocks after the acknowledge cycle (as indicated by the I/O write to address 1000) to avoid being latched again into the PIR.

Interrupts are automatically disabled while being serviced and must be re-enabled by the software before further interrupts are serviced. When the software is ready to return control back to the program executing prior to the interrupt, it must execute an LST or LSTI command with the address of the linkage pointer in the address field of the instruction. To guarantee completion of the service routine before receiving another interrupt, the ENBL command hould be the last instruction prior to the LST.

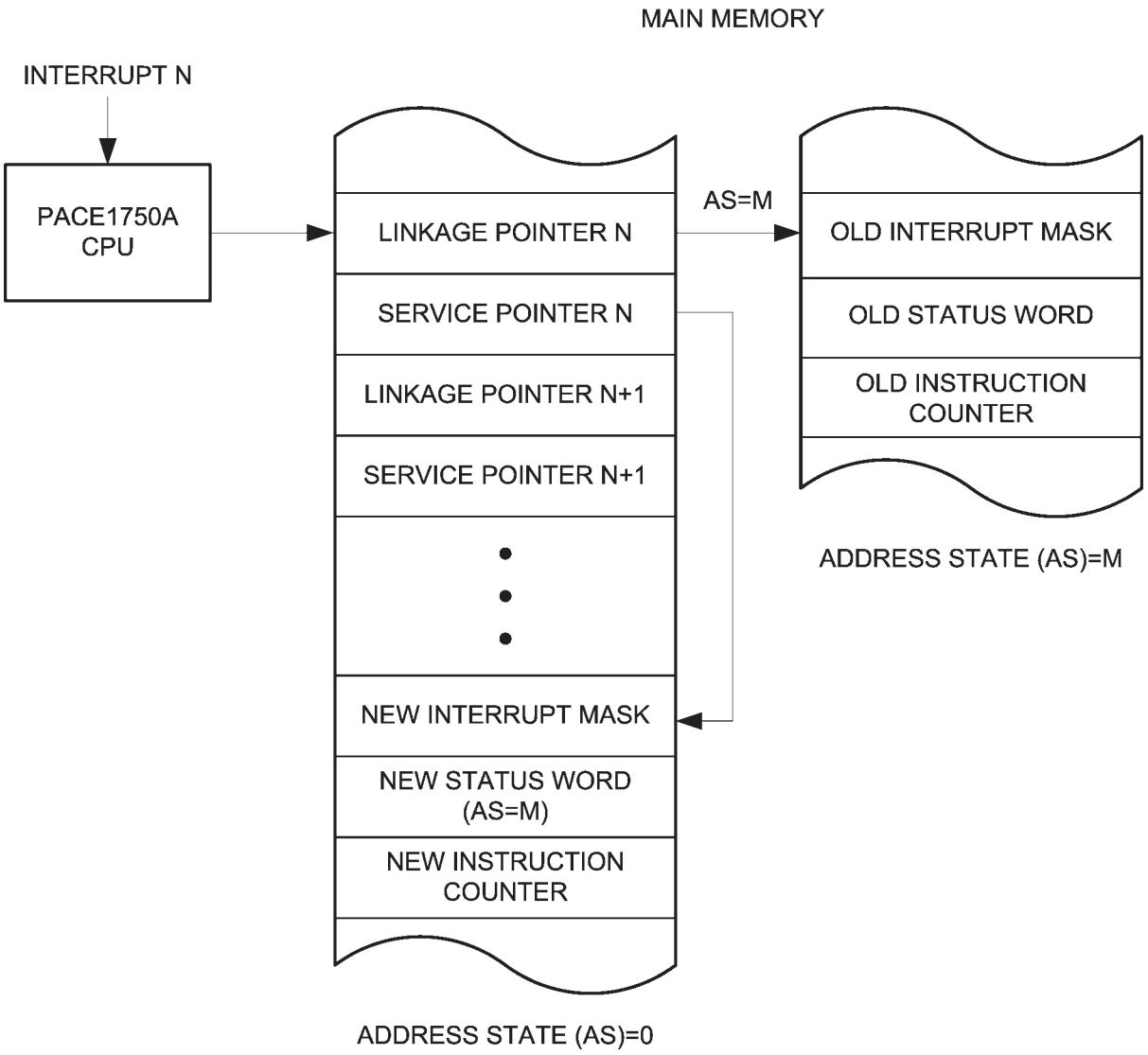


Figure 3.5 Interrupt Vectoring

INFORMATION BUS (IB)

The information bus is the means of communication into and out of the PACE1750A CPU. It is a 16-bit bi-directional time-multiplexed address and data bus controlled by seven strobe and status signals and four arbitration signals. The bus is operated according to the B machine state diagram described earlier in this note. The control and arbitration signals are described below:

STRBA – This active-high output indicates the beginning and end of the address phase of the bus cycle. The address may be latched off of the IB on the falling edge of STRBA, but transparent latches should be used to provide the address to the bus during STRBA. It becomes high-impedance during Bi and Bz states.

$\overline{\text{STRBD}}$ – This active-low output indicates the beginning and end of the data phase of the bus cycle. The data is latched off of the IB by the CPU on the rising edge of $\overline{\text{STRBD}}$ during read cycles and the data is stable on the rising edge of $\overline{\text{STRBD}}$ when supplied by the CPU during write cycles. It becomes high-impedance during Bi and Bz states.

D/$\overline{\text{I}}$ – This output indicates whether the current bus cycle is an instruction fetch or data cycle. It is LOW during instruction fetches and HIGH during data cycles. It becomes high impedance during Bi and Bz states.

R/$\overline{\text{W}}$ – This output indicates whether the current bus cycles is a read or write operation. It is HIGH during reads and LOW during writes. It becomes high-impedance during Bi and Bz states.

M/$\overline{\text{IO}}$ – This output indicates whether the current bus cycle is a memory or I/O operation. It is HIGH when the memory is accessed and LOW when an I/O device is accessed. It becomes high-impedance during Bi and Bz states.

RDYA – This active-high input indicates to the processor that the external system does not require a stretched address phase during the current cycle. A LOW on this signal will cause the CPU to insert wait states during the address phase (ie. Lengthen STRBA) until it becomes HIGH.

RDYD – This active-high input indicates to the processor that the external system does not require a stretched data phase during the current cycle. A LOW on this signal will cause the CPU to insert wait states during the data phase (ie. Lengthen $\overline{\text{STRBD}}$) until it becomes HIGH.

$\overline{\text{BUS REQ}}$ – This active-low output is generated by the CPU when a bus access is required. It can be used by an external arbiter to prioritize requests from different bus users.

$\overline{\text{BUS GNT}}$ – This active-low input indicates to the processor that it currently has the highest priority bus request. The CPU will not assume control of the bus until the $\overline{\text{BUS BUSY}}$ and $\overline{\text{BUS LOCK}}$ signals become inactive even when $\overline{\text{BUS GNT}}$ is active. When this signal is HIGH, the CPU will complete the current cycle and enter the Bz state until it becomes LOW.

$\overline{\text{BUS BUSY}}$ – This bi-directional active-low signal establishes the beginning and end of a bus cycle. When the CPU has the $\overline{\text{BUS GNT}}$ signal, and $\overline{\text{BUS BUSY}}$ and $\overline{\text{BUS LOCK}}$ are not active from another bus user, the CPU drives $\overline{\text{BUS BUSY}}$ low when it assumes control of the bus and releases it when the cycle is complete. This signal is an input when the CPU does not have $\overline{\text{BUS GNT}}$ and is not currently involved in a bus cycle.

$\overline{\text{BUS LOCK}}$ – This bi-directional active-low signal is used to lock the bus, preventing another user from gaining access to the bus for successive bus cycles such as required during read-modify-write operations. During non-locked bus cycles, this signal follows $\overline{\text{BUS BUSY}}$. It is an input when the CPU does not have $\overline{\text{BUS GNT}}$.

FAULTS AND ERRORS

The PACE1750A CPU provides extensive fault management capability. The heart of this capability is a 16-bit Fault Register (FT) that is used to store four internally generated and seven externally generated faults. Table 3.4 defines the format of the FT.

TABLE 3.4 FAULT REGISTER

Bit	Fault	Error
0	CPU Memory Protect Error	
	Data Cycle	Major
	Instruction Fetch	Unrecoverable
1	DMA Memory Protection Fault	Unclassified
2	Parity Error	
	Data Cycle	Major
	Instruction Fetch	Unrecoverable
3	Spare	
4	Spare	
5	Illegal I/O Address	Major
6	Spare	
7	System Fault 0	Unclassified
8	Illegal Memory Address	
	Data Cycle	Major
	Instruction Fetch	Unrecoverable
9	Illegal Instruction	Unrecoverable
10	Privileged Instruction	Major
11	Address State Error	Unrecoverable
12	Spare	
13	Built-In Test or System Fault 1	Unclassified
14	Spare	
15	System Fault 1	Unclassified

Faults are latched into the FT at the end of every bus cycle on the rising edge of $\overline{\text{BUS BUSY}}$ even if the bus cycle does not belong to the CPU. The bits of the FT are logically OR'ed together to form the Machine Error Interrupt to the processor. Thus, if any bit is set, the CPU will receive a level 1 interrupt. The Machine Error interrupt is guaranteed to occur prior to processing of the faulty instruction or data thereby allowing the software to handle the problem before the state of the machine is corrupted. The 5 spare bits are always set to 0.

The severity of a particular fault is classified as Unrecoverable, Major or Unclassified. If an Unrecoverable error is encountered, the instruction causing the error is aborted with the IC pointing at the faulty instruction or the one before (if the fault occurred during a pre-fetch operation). If a Major error is detected, the register file is protected and will not be modified. Major and Unrecoverable errors also cause the signals MAJ ER and UNRCV ER to be asserted for system fault management purposes. A description of each fault stored in the FT follows below.

Bit 0 – CPU Memory Protect Fault

This fault occurs when the CPU attempts to write into memory which is protected by either the MMU or BPU (if implemented), or execution of an address which is execute protected by the MMU. All cases are indicated by the $\overline{\text{MEM PRT ER}}$ input signal being LOW during the rising edge of $\overline{\text{BUS BUSY}}$. If it occurs during a data cycle, the fault is classified as major. If it occurs during an instruction fetch, it is classified as unrecoverable.

Bit 1 – DMA Memory Protect Error

This fault occurs when a non-CPU memory write is attempted to a location protected by the MMU or BPU as indicated by the signal $\overline{\text{MEM PRT ER}}$ being LOW at the rising edge of $\overline{\text{BUS BUSY}}$. This error is unclassified.

Bit 2 – Parity Error

This fault is caused by the signal $\overline{\text{MEM PAR ER}}$ being LOW during the rising edge of $\overline{\text{BUS BUSY}}$. It can occur when a parity error, or multiple error (when Error Detection and Correction is implemented), is detected during a memory read cycle. If it occurs during a data cycle, it is classified as major. If it occurs during an instruction fetch, it is classified as unrecoverable.

Bit 3 – Spare

This bit is always “0”.

Bit 4 – Spare

This bit is always “0”.

Bit 5 – Illegal I/O Address

This fault occurs when an attempt is made to access an unimplemented or reserved I/O address as indicated by the $\overline{\text{EXT ADR ER}}$ input signal being LOW and M/IO being LOW during the rising edge of $\overline{\text{BUS BUSY}}$. This error is classified as major.

Bit 6 – Spare

This bit is always “0”.

Bit 7 – System Fault 0

This fault is defined by the external system. It is caused by the rising edge of SYSFLT_0. It is unclassified.

Bit 8 – Illegal Memory Address

This fault occurs when an attempt is made to access an unimplemented memory location as indicated by the $\overline{\text{EXT ADR ER}}$ input signal being LOW and M/$\overline{\text{IO}}$ being HIGH during the rising edge of $\overline{\text{BUS BUSY}}$. If it occurs during a data cycle, it is classified as major. If it occurs during an instruction fetch, it is classified as unrecoverable.

Bit 9 – Illegal Instruction

This fault occurs when the CPU attempts to execute a data word which is not a legal MIL-STD-1750A op-code. It is classified as unrecoverable.

Bit 10 – Privileged Instruction

This fault occurs when the CPU attempts to execute an instruction defined as privileged when the PS field of the SW is not equal to 0. It is classified as major.

Bit 11 – Address State Error

This fault occurs when the CPU attempts to modify the AS field of the SW to a non-zero value when there is no MMU implemented. It is classified as unrecoverable.

Bit 12 – Spare

This bit is always “0”.

Bit 13 – Built-In Test or System Fault 1

This fault is the logical OR of two conditions. If the CPU fails its built-in test, it will be set after power up. It is also set by the rising edge of the SYSFLT_1 signal. It is unclassified.

Bit 14 – Spare
This bit is always “0”.

Bit 15 – System Fault 1
This fault is defined by the external system. It is caused by the rising edge of the $SYSFLT_1$ signal. It is unclassified.

When a LOW is detected on the external address error ($\overline{\text{EXT ADR ER}}$) input, the current bus cycle is terminated after one B3 clock cycle. RDYD is overridden in this case.

SELF-TEST AND INITIALIZATION

After every reset, including power-up, the CPU performs a self-test to insure that basic operation is possible. After the self-test has been successfully completed, the CPU must initialize itself to the reset state specified by the standard.

The self-test is a micro-coded test that checks the major data paths through the CPU and all of the major functional blocks of the hardware extensively. It begins running immediately after the $\overline{\text{RESET}}$ input becomes HIGH. If the test passes, the CPU will set the normal power up discrete (NML PWRUP) and proceed to the initialization sequence. If the test fails, bit 13 of the FT will be set and NML PWRUP will not be asserted. The test flow is depicted in Figure 3.6.

The purpose of the initialization sequence is to reset the state of the CPU to the state specified by the standard. Table 3.5 summarizes the reset state of the machine after the initialization sequence is complete. The CPU initialization is always performed, but the MMU and BPU initializations are performed only if the MMU Present and BPU Present bits are set in the Configuration Register. The test times with the various hardware configurations are given in Table 3.6. The number of clocks given in the table is cumulative (ie. The length of time given for the MMU and BPU configurations includes the time for the self-test up and SCR read).

TABLE 3.5 PACE1750A Reset State

Instruction Counter (IC)	All Zeroes
Status Word (SW)	All Zeroes
Fault Register (FT)	All Zeroes
Pending Interrupt Register (PIR)	All Zeroes
Interrupt Mask Register (MK)	All Zeroes
Interrupts	Disabled
DMA Enable	Disabled
Timer A and Timer B	All Zeroes and Started
Trigger Go Reset ($\overline{\text{TRIGO RST}}$)	Pulsed
Normal Power Up (NML PWRUP)	Set High

TABLE 3.6 Built-In Test Times

System Mode	Clock Cycles
Normal Power Up	553
Read Configuration	569
No MMU, No BPU	585
No MMU, With BPU	1877
With MMU, No BPU	5723
With MMU, With BPU	7015

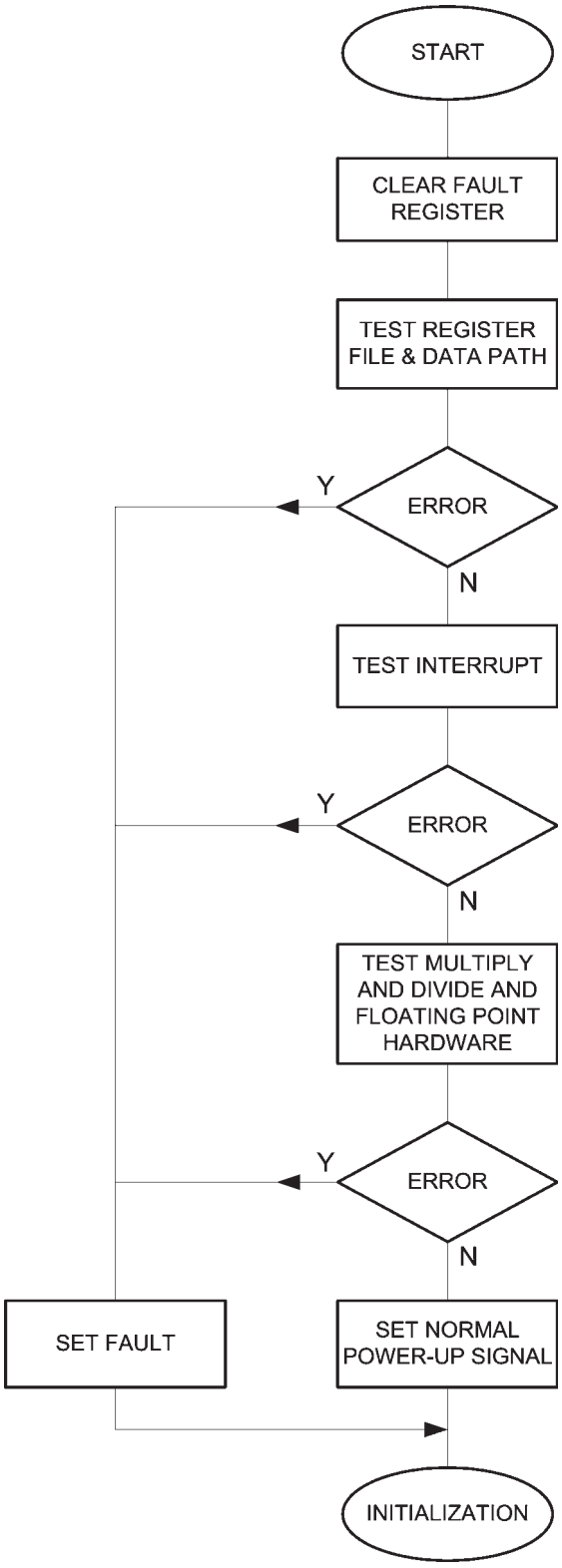


Figure 3.6 Built In Test Flow

CONSOLE MODE

The PACE1750A CPU implements a console mode to provide a port for software and hardware development and test. The console mode may be entered at any time after completion of the CPU self-test and initialization. There are two methods of entering the console mode. The first is to issue a console request by driving the $\overline{\text{CON REQ}}$ input LOW. The CPU responds by completing the current instruction and entering the console mode. The other method is by executing the breakpoint (BPT) instruction. When the CPU encounters a BPT, it first reads the SCR. If the Console Present bit is set, the CPU responds by entering the console mode. A BPT is treated as a NOP command if the Console Present bit is not set. The console request has higher priority than Interrupt requests, such that if an interrupt and console request are received simultaneously, the interrupt will remain pending until after exiting the console mode. Timer A and Timer B are both halted during the console mode and restarted after returning to the normal execution mode.

Once in the console mode, the CPU follows the flow diagram depicted in Figure 3.7. The CPU first reads the console command, which should be placed on the Information Bus by the external console controller in response to an I/O read to address 8400 (hex). The command is 16-bits wide, conforming to the definition in Figure 3.8. If the data placed on the bus is not a legal console command, it will be treated as a NOP instruction and the CPU will wait for another console request. The console controller should detect the I/O read from address 8400 and respond by removing its console request. The CPU then reads the second command word, if required, which should be placed on the bus by the console controller in response to an I/O read to address 8401 (hex). The CPU then executes the command issued by the controller and sends the result to I/O write address 0400. The console controller should read the result from the bus and issue another console request when ready. The CPU will wait for a new request indefinitely. If other console commands are to be issued, they may be executed without leaving the console mode by following the same procedure as described above. When the controller is ready to exit the console mode, the console command CONTINUE must be executed. This will return the CPU to the normal mode. While in the console mode, all bus cycles are data cycles, meaning the state of the D/$\overline{\text{I}}$ signal will always be logic “1”.

If an external address error ($\overline{\text{EXT ADR ER}}$) is detected during a bus cycle while in the console mode, the cycle will terminate just as if a normal instruction were executing at the time of the fault. When this occurs, the machine error interrupt will remain pending until after exiting the Console Mode.

The console commands are described in Table 3.8. The command set allows reading and writing of registers, memory and I/O addresses, and can be very useful as a development and test interface. Table 3.7 lists the register addresses for the console commands. Several of the registers require procedures other than the DEPOSIT and EXAMINE commands for reading and modification:

1. To read the IC, an EXAMINE REGISTER command must be performed to the IC register address. To deposit the IC, an EXAMINE MEMORY command must be executed with the desired IC value as the memory address.

2. To read the Status Word, an EXAMINE REGISTER command must be performed to the SW register address. To deposit the SW, a DEPOSIT STATUS WORD command must be executed.

3. To read the Fault Register, an EXAMINE AND CLEAR FAULT REGISTER command must be executed. To deposit the FT, a DEPOSIT REGISTER command must be executed to the FT address, but only bits 9, 10, 11 and 13 of the data word will actually be deposited into the FT.

4. It should be noted that some I/O commands (DEPOSIT XIO and EXAMINE XIO) behave differently in the Console Mode than in the normal mode. None of the registers internal to the CPU may be read or written using the EXAMINE XIO or DEPOSIT XIO commands since the CPU does not decode I/O addresses while in the Console Mode, but merely outputs the I/O address on the IB. If it is desired to access one of the internal registers, the EXAMINE or DEPOSIT REGISTER commands must be executed to the appropriate register address (e.g., to read the Pending Interrupt Register, an EXAMINE REGISTER command is executed to the PIR address listed in Table 3.8).

The DISABLE command behaves somewhat differently than the other commands described in the table. When the DISABLE command is executed, if there are any unmasked interrupts pending in the PIR and interrupts are not

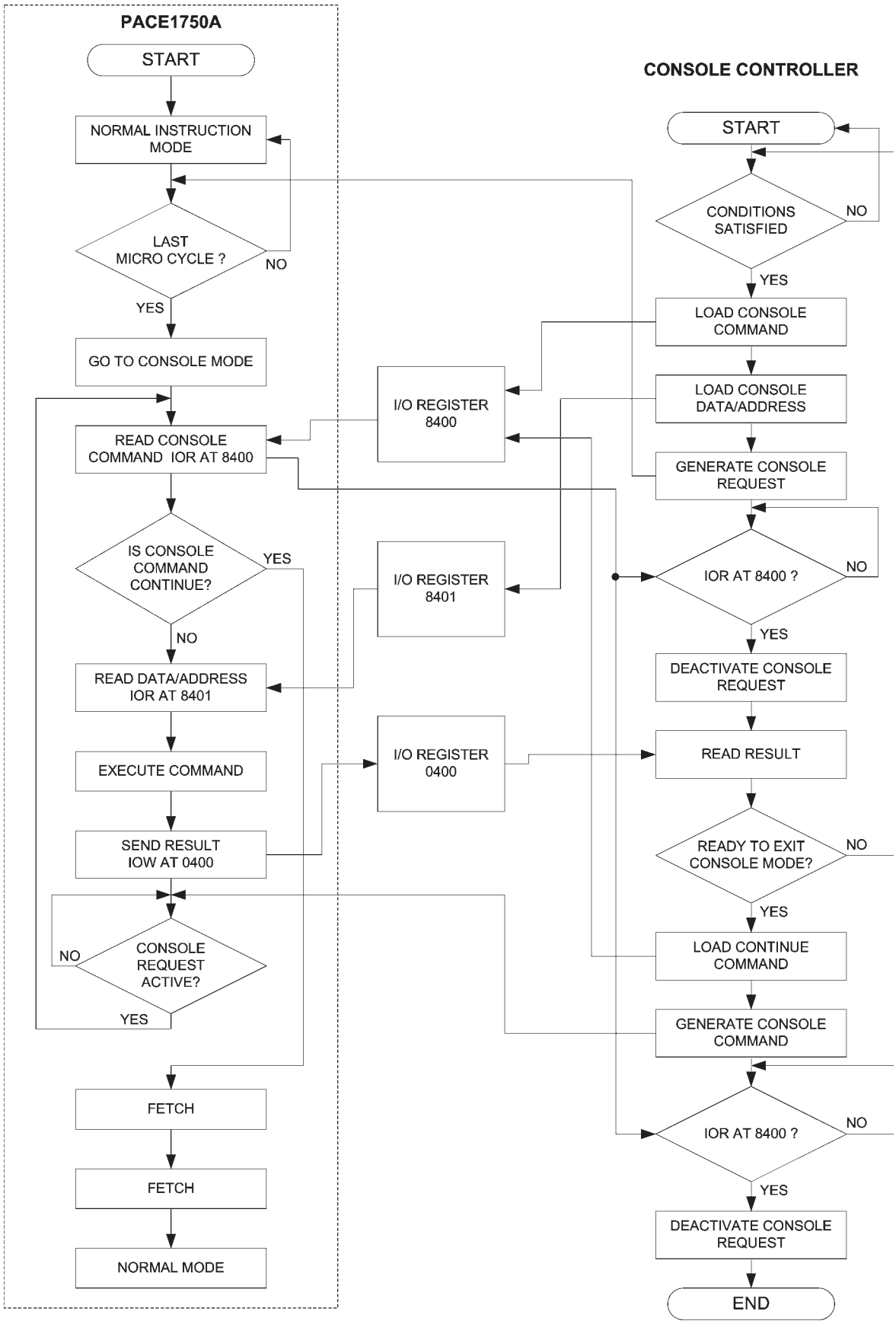
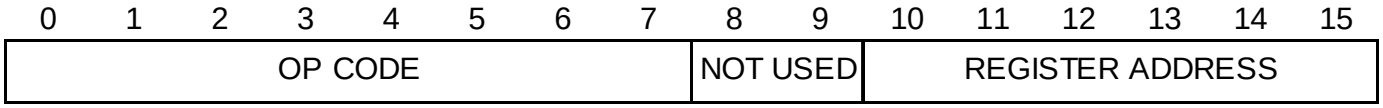


Figure 3.7 Console Flow

disabled, the CPU will service the interrupts and enter the normal mode. If there are no unmasked interrupts pending and/or interrupts are disabled, the CPU will enter the high-impedance state, alternately checking for console requests and unmasked interrupts. If a console request is detected, the CPU will enter the console mode. If an unmasked, enabled interrupt is detected, the CPU will service it and enter the normal operating mode.

The console mode can be used to implement a single step capability for software trace and debugging. This can be accomplished by issuing a console request and placing a CONTINUE command on the bus. The console request should then be removed. When the CPU returns to the normal mode it will begin executing from the point at which it was interrupted by the console request. When execution of the first instruction begins, the start new signal (SNEW) will be asserted and this can be used to generate another console request which will again send the CPU to the console mode after completion of the instruction. It should be noted that the execution of the CONTINUE command also generates the SNEW signal, so the second occurrence of SNEW should generate the console request. Since the state of the $D/\overline{I}$ signal is always "D" (data) during Console Mode, the $D/\overline{I}$ line could be used to qualify SNEW to issue the console request only during the SNEW corresponding to the normal instruction execution ($\overline{CON\ REQ} = D/\overline{I} + \overline{SNEW}$). In this way, single-stepping can be performed indefinitely. If desired, the state of all of the registers can be examined after every instruction to provide a true trace of the machine's activity.



NOTE: For commands not requiring a register address, bits 10 through 15 are DON'T CARE.

Figure 3.8 Console Command Format

DMA SUPPORT

The CPU supports the DMA enable and disable commands defined by the standard. When the CPU detects a DMAE I/O command (4006), it drives the DMA EN output signal high and holds it until detecting the execution of a DMAD I/O command (4007) at which time it drives the DMA EN output LOW. This feature, along with the multi-master bus support provided by the CPU, makes DMA relatively easy to design into a PACE1750A system.

CLOCKS AND EXTERNAL TIMER PROVISIONS

The CPU accepts two clock inputs. One is used as the basic CPU clock and the other drives the A and B timers. The timer clock is input on the TIMER CLK pin and must be a 100kHz clock with a 40-60% duty cycle. The CPU clock is input on the CPU CLK pin and may be any frequency from 0-40MHz with a 40-60% duty cycle. The only restriction on this clock is that, if the timers are used, a frequency of at least 300kHz must be supplied on the CPU CLK input.

In addition to the A and B timers, the CPU supports a trigger go timer as defined by the standard. This timer, external to the CPU, can be designed according to specific system requirements, but the CPU decodes the GO I/O command (400B) and provides a pulse on the $\overline{TRIGO\ RST}$ when the GO command is executed.

TABLE 3.7 CONSOLE REGISTER ADDRESSES

Register	Binary	Hex	Register	Binary	Hex
R0	000000	00	R12	000110	06
R1	010000	10	R13	010110	16
R2	000001	01	R14	000111	07
R3	010001	11	R15	010111	17
R4	000010	02	A1	011000	18
R5	010010	12	PIR	001011	0B
R6	000011	03	MK	011011	1B
R7	010011	13	FT	001100	0C
R8	000100	04	SW	001101	0D
R9	010100	14	TA	001110	0E
R10	000101	05	TB	011110	1E
R11	010101	15	IC	100101	25

TABLE 3.8 CONSOLE OP-CODES

COMMAND	CODE (HEX)	1 WORD / 2 WORD	DESCRIPTION
EXAMINE REGISTER	60	1	The CPU writes the contents of the register specified by the address bits of the command word to I/O port 0400.
DEPOSIT REGISTER	61	2	The CPU writes the data from I/O port 8401 into the register specified by the address bits of the command word.
EXAMINE AND CLEAR FAULT REGISTER (FT)	62	1	The CPU writes the contents of the FT to I/O port 0400 and clears the FT.
DEPOSIT STATUS WORD	63	2	The CPU writes the data from I/O port 8401 into the status word.
EXAMINE MEMORY	66	2	The CPU reads the address from I/O port 8401 and stores it in the IC. The CPU then reads the memory location specified by the IC and writes it to I/O port 0400.
DEPOSIT MEMORY	67	2	The CPU writes the data from I/O port 8401 into the memory location specified by the IC.
EXAMINE NEXT MEMORY	6A	1	The CPU increments the IC and then reads the memory location at that address and writes it to I/O port 0400.
DEPOST NEXT MEMORY	6B	2	The CPU increments the IC and then writes the data from I/O port 0400 into the memory location specified by the IC.
EXAMINE XIO	6C	2	The CPU reads the I/O address from I/O port 8401 into A1 and then writes the data from that address to I/O port 0400.
DEPOSIT XIO	6D	2	The CPU writes the data from I/O port 8401 to the I/O address stored in register A1.
EXAMINE NEXT XIO	6E	1	The CPU increments R1 and then reads from the I/O address specified in A1 and writes the data to I/O port 0400.
DEPOSIT NEXT XIO	6F	2	The CPU increments R1 and then writes the data from I/O port 8401 to the I/O address specified by A1.
DISABLE	74	1	The DISABLE Command forces the CPU into a high-impedance state during which time it continues to check for interrupts and console requests.
CONTINUE	75	1	The CPU exits the console mode and begins normal operation.

3.2 PIC

3.2.1 Functional Description

The PACE1754 PIC is a support chip to the PACE1750A processor providing many of the functions required by most 1750A implementations. It is comprised of the seven major functional blocks shown in Figure 3.9.

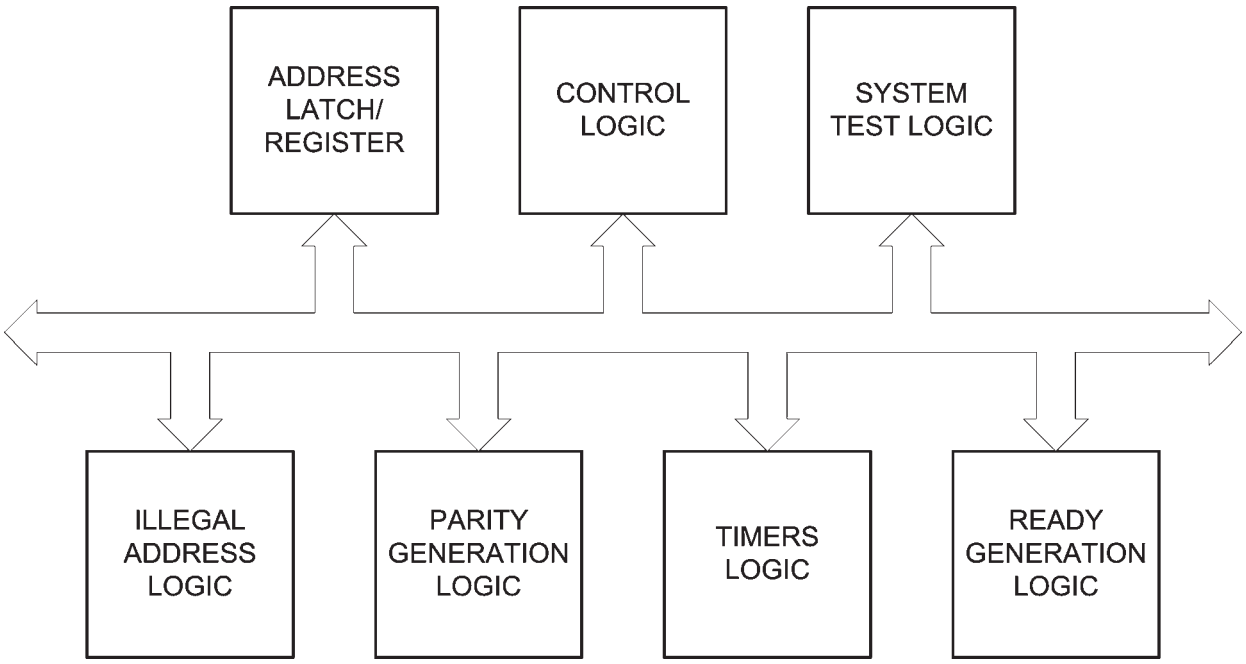


Figure 3.9 PIC Block Diagram

ADDRESS LATCH/REGISTER

The address latch/register is a 16-bit transparent latch used to pick the address off the IB during the address phase of a memory or I/O bus cycle. It passes the IB through to the A(0:15) bus during the time that STRBA is HIGH and latches it on the falling edge of STRBA. The A-bus drivers are capable of driving the address directly to the memory and I/O systems (maximum load = 130pf) and can be forced into a high-impedance state if necessary. During memory cycles when there is an MMU present, bits A(0:3) become high-impedance (A0 and A1 in fact become inputs) since only the least significant 12 bits of the address are sourced by the PIC. During all I/O cycles and during memory cycles without an MMU, all 16 bits of the address may be sourced by the PIC.

CONTROL LOGIC

The control logic provides the internal timing and control of the other six logic blocks. It contains the Control Register (described later) which provides the user-programmability of the PIC functions.

SYSTEM TEST LOGIC

This logic block, when enabled, performs a complete test of the PACE1750A system (CPU, PIC, and COMBO) at initial power-up and after every reset. It contains a 4k x 16 bit ROM which stores a 1750A program that is run by the processor after reset. It also supplies several words of RAM which are used during the test to exercise the memory read and write functions. In addition to this hardware and software, this block contains several hardware features to assist the test software in obtaining the highest possible fault coverage within the constraints of ROM size and signal access.

ILLEGAL ADDRESS LOGIC

The illegal address detection logic contains decoders for both memory and I/O addresses as well as several programmable registers which work together to detect and report attempts to access illegal and unimplemented addresses to the processor. When an illegal or unimplemented address is detected, the external address error ($\overline{EXADER}$) output is generated.

PARITY GENERATION LOGIC

The parity generation logic contains the parity generation and checking hardware to provide single-bit odd or even parity for the memory system. A control register defines whether odd or even parity is to be generated and checked and for which areas of memory parity is implemented. The PIC generates IB16 (the parity bit) to the memory on every data cycle to be written to the memory during $\overline{STRBD}$, just like the normal data on IB. When a parity error is detected during a memory read, the PIC generates the memory parity error ($\overline{MEMPAER}$) output.

TIMERS LOGIC

This block contains the two watchdog timers provided by the PIC for management of faults which can tie up the bus and/or CPU. Both timers have programmable time-out periods and generate the terminal count (TC) output upon reaching their internal counts.

READY GENERATION LOGIC

The ready generation logic contains the hardware necessary to manage the insertion of wait states during the data phase of a bus cycle. It accepts the external ready inputs (EX RDY and $\overline{EXRDY1}$) and also counts off the number of programmed wait states, using this information to generate the ready signal (RDYD) to the processor.

3.2.2 PIC Operation

This section describes the use of the PACE1754 PIC in systems utilizing the PACE1750A processor. The PIC may be used both with and without the PACE1753 COMBO. For further information, please consult the PACE1754 PIC data sheet.

CONTROL AND STATUS REGISTERS

The operation of the PIC is controlled by a 16-bit control register which is loaded and read by the CPU via XIO commands. The I/O address is 9F40 for reads and 1F40 for writes. The status of the PIC operation may be read via the 16-bit Status Register at I/O read address 9F41. The register bit definitions are depicted in the figures below. For a summary description of all PIC registers, consult Appendix A.

Control Register (1F40, 9F40)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PR1	PR2	PR3	PR4	ODD	EST	EAD	EXR	SPI	CNF	EB1	EB2	EIO	LIO	LME	0

Default = 0000

PR1 – A “1” in this bit enables parity checking for the memory address range from 0000-3FFF (hex).

PR2 – A “1” in this bit enables parity checking for the memory address range from 4000-7FFF (hex).

PR3 – A “1” in this bit enables parity checking for the memory address range from 8000-BFFF (hex).

PR4 – A “1” in this bit enables parity checking for the memory address range from C000-FFFF (hex).

ODD – A “1” in this bit sets the parity generation and checking logic for odd parity. A “0” sets even parity.

EST – A “1” in this bit enables high-impedance state control on the PIC-generated memory and I/O read and write strobes ($\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$ and $\overline{MEMW}$).

EAD – A “1” in this bit enables high-impedance state control on the PIC address lines (A_0-A_{15}).

EXR – A “1” in this bit extends the I/O ready generation over the full 64k address space. A “0” applies ready generation over just the user I/O space (0000-03FF and 8000-83FF).

SPI – A “1” in this bit enables illegal I/O address detection for I/O addresses defined by MIL-STD-1750A as spare. (1=Spare I/O legal, 0=Spare I/O illegal).

CNF – A “1” in this bit enables support of EDAC when used with PACE1753 COMBO. With EDAC support enabled, PIC outputs $\overline{MEPAER}$ and $\overline{EXADER}$ become the EDAC handshake signals RAM DIS and SING ERR respectively.

EB1 – A “1” in this bit enables the detection of accesses to block 1 of unimplemented memory as defined by the Unimplemented Memory Register.

EB2 – A “1” in this bit enables the detection of accesses to block 2 of unimplemented memory as defined by the Unimplemented Memory Register.

EIO – A “1” in this bit enables detection of illegal I/O addresses (addresses not defined as user I/O by the First Unimplemented I/O registers).

LIO – A “1” in this bit enables long I/O ready generation, 1ms to 15ms, on addresses 0000-00FF and 8000-80FF.

LME – A “1” in this bit enables long memory ready generation, 1ms to 15ms, on memory addresses 0000-3FFF.

Status Register (9F41)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPU	CMB	PIC	RESERVED		STB	ADR	TWD	TBT	RESERVED						IFL

CPU – A “1” in this bit indicates that the PACE1750A CPU passed the built-in system test.

CMB – A “1” in this bit indicates that the PACE1753 COMBO passed the built-in system test.

PIC – A “1” in this bit indicates that the PACE1754 PIC passed the built-in system test.

STB – A “1” in this bit indicates that one of the PIC strobes has been detected as stuck at 0 or 1.

ADR – A “1” in this bit indicates that one of the PIC address lines has been detected as stuck at 0 or 1.

TWD – A “1” in this bit indicates that the Watch Dog Timer has reached its terminal count.

TBT – A “1” in this bit indicates that the Bus Timer reached its terminal count.

IFL – A “1” in this bit indicates that interrupts are currently enabled by the CPU.

RESERVED – These bits are always “0”.

PROGRAMMABLE READY GENERATION

The PIC supplies the ready data (RDYD) signal to the processor to indicate the need for wait states during the data phase of a bus cycle. The PIC may be programmed to provide from 0 to 15 wait states on memory or I/O operations by withholding the RDYD signal until the programmed number of clock cycles have been counted off.

Memory Operations

The PIC provides programmable wait state generation for each quarter of the memory address space separately according to the values on A0 and A1. If an MMU is not implemented, A0 and A1 are sourced by the PIC and provide the four quarters of the 64k address space. If the MMU is implemented, A0 and A1 become inputs and should be connected to the two most significant bits of the extended address to divide the memory into four equal quarters. The number of wait states required for each memory quarter is determined by the Memory Ready Program Register (I/O read address 9F42, I/O write address 1F42). The register is 16-bits wide and is divided into four sections of 4 bits each, as shown below.

Memory Ready Program Register (1F42, 9F42)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MEM Q1				MEM Q2				MEM Q3				MEM Q4			

Default = FFFF

Each 4-bit section defines the number of wait states to be inserted during bus cycle B3 for each of the four quarters of memory (Q1 is the least significant quarter and Q4 is the most significant). The number of wait states may range from 0 to 15 for each quarter. When the bus cycle begins B3, if wait states are programmed, the PIC will de-assert RDYD and begin counting clocks until the number programmed is reached, after which RDYD is asserted and the bus cycle allowed to complete. The wait states are added in addition to the one clock normally spent in B3. At power up, the Memory Ready Program Register defaults to FFFF (hex) corresponding to 15 wait states on all of the memory.

I/O Operations

The PIC will provide the data wait states required for I/O operations much like it does for memory operations. The I/O address space is also divided into four quarters. The quarters can be over the full 64k I/O address space, or over just the user I/O space (0000-03FF, 8000-83FF) depending on the state of bit 7 (EXR) in the PIC Control Register. The number of wait states is programmed in the I/O Ready Program Register depicted below.

I/O Ready Program Register (1F43, 9F43)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IO Q1				IO Q2				IO Q3				IO Q4			

Just as in memory operations, the PIC de-asserts RDYD during the B3 phase of the I/O bus cycle to add the number of clock cycles programmed in the I/O Ready Program Register to B3. The internal I/O addresses defined by the standard, as well as those used by the PACE1750A system, are automatically given 0 wait states by the PIC when the EXR bit in the control register is “0”. When the ready generation programming is extended over the entire I/O address space (EXR = “1”), these I/O addresses are given the number of wait states programmed. The I/O Ready Program Register defaults to 0000, corresponding to 0 wait states over the entire I/O address space.

Long Wait States

The PIC provides the capability to interface to very slow memory and I/O devices through the use of the long ready capability. When the LIO bit (bit 14) of the PIC Control Register is set to “1”, the PIC can be programmed to hold RDYD inactive for 1ms – 15ms instead of 1 to 15 CPU clocks for the first quarter of the I/O address space. The length of time (number of ms) is programmed in the IO Q1 field of the I/O Ready Program Register.

Similarly, long memory waits are provided for the first quarter of memory when bit 14 (LME) in the Control Register is set to “1”. The PIC can be programmed to provide wait states from 1 to 15ms based on the MEM Q1 field of the Memory Ready Program Register.

External Ready Generation

In addition to the programmable ready generation internal to the PIC, two external ready inputs (EX RDY and $\overline{\text{EX RDY1}}$) are provided to override the ready generation programming for any memory or I/O cycle. A LOW on EX RDY or a HIGH on $\overline{\text{EX RDY1}}$ will force a LOW on RDYD regardless of the ready programming. When either of these two signals indicates not ready, the wait state counter resets and wait states are inserted by the PIC indefinitely until the external ready signals become active. Once the external ready signals become active, the PIC will again begin counting off the number of wait states programmed. Thus, the total number of wait states added to B3 will be the sum of the number added due to the external ready inputs, the number programmed in the Ready Program Register, and the number that elapsed before the external ready input was received by the PIC. The external ready inputs also override the automatic ready (zero wait state) generation for the MIL-STD-1750A I/O commands and the PACE system commands.

PARITY GENERATION AND CHECKING

The PIC provides the capability to generate and check even or odd parity on memory writes and reads within a 64k word address space. The 17th data bit (IB16) is always generated on memory writes according to bit 4 (ODD) in the Control Register. A "1" in this bit causes odd parity to be generated while a "0" causes even parity to be generated. Parity checking can be selectively enabled on each of the four memory quarters as required by the memory system. A "1" in any of bits 0-3 of the Control Register (PR1-PR4) enables parity checking for one quarter of the memory space. If a parity error is detected, the PIC asserts the memory parity error output ($\overline{\text{MEM PAR ER}}$) which should be connected to the $\overline{\text{MEM PAR ER}}$ input on the CPU.

TIMERS

The PIC provides two timers for monitoring the health of the system. For either of the timers to function, the system clock frequency must be programmed in the Program Register (shown below). The first timer is a Watch Dog for insuring that the software running on the system continues executing as intended. Once enabled and programmed, the timer will increment until reaching its terminal count, after which the TC output from the PIC will become active. The timer is reset by software through execution of the GO XIO command (I/O address 400B). Thus, the software must periodically reset the timer to prevent the TC output from occurring. If the TC output were connected to a CPU interrupt input, the CPU could be notified when the software takes too long to perform the Watch Dog reset. The time-out period is set by the software in the Watch Dog Timer Register depicted below. The timer will increment from the value in this register until reaching FFFF (hex). Upon incrementing from FFFF to 0000, the TC output will occur.

Watch Dog Timer Register (1F45, 9F45)

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

Default = 0000

The Watch Dog timer may be enabled and disabled by the EWD bit (bit 8) in the Program Register. The timer is capable of running at two clock rates. Bit 9 in the Program Register is the SWD bit which is used to select a 1 KHz or a 1 MHz clock to increment the timer. This makes possible time out ranges of 1-65,536 ms and 1-65,536 µs, respectively. Once the TC output has been set by a Watch Dog time out, only executing the GO I/O command, disabling the timer (EWD=0) or reading the Status Register will reset it.

The second timer is a bus timer used to help prevent non-terminating bus cycles. The PIC will set RDYD HIGH regardless of the state of the external ready inputs and assert the TC output whenever a bus time-out occurs. The timer may be set to expire after 64 or 256 clock cycles while the CPU is waiting for RDYD according to bit 7 (SBT) of the Program Register. The Bus Timer is enabled by bit 6 (EBT) of the Program Register. Once the Bus Timer has set the TC output, only reading the Status Register or disabling the timer (EBT=0) will reset it. Note that if the long ready generation is used, it is possible that the Bus Timer may time-out even on normal bus cycles since it is possible to program the long ready to hold RDYD longer than the Bus Timer time-out period.

Since the TC output may be caused by either the Bus Timer or the Watch Dog Timer, the PIC provides the means to detect which timer has expired. Bit 7 in the Status Register (TWD) is set when the Watch Dog Timer has reached its terminal count and Bit 8 (TBT) is set when the Bus Timer has reached its terminal count. Reading the Status Register will clear the TC output for both timers.

Program Register (1F44, 9F44)

0 1 2 3 4 5	6	7	8	9	10 11 12 13 14 15
CLOCK FREQUENCY (MHZ)	EBT	SBT	EWD	SWD	RESERVED

Default = 0000

CLOCK FREQUENCY – When the frequency of the clock is a whole number of MHz (eg. 1,2,3…) it may be binary encoded in these 6 bits to provide the frequencies required for the internal PIC timers and the 100kHz TIMER CLK output. When programmed to all 0's (the default value), the TIMER CLK output does not function.

EBT – A "1" in this bit enables the Bus Timer function.

SBT – This bit selects the time out period of the Bus Timer. 1=256 cycles, 0=64 cycles.

EWD – A "1" in this bit enables the Watch Dog Timer function.

SWD – This bit selects the Watch Dog clock frequency. 1=1kHz, 0=1MHz.

RESERVED – Always reads as 0's.

ILLEGAL ADDRESS DETECTION

The PIC can be programmed to detect and notify the processor when accesses are attempted to unimplemented memory or I/O addresses.

Memory Operations

The PIC detects accesses to up to two blocks of unimplemented memory within the 64k address space. These unimplemented blocks may be of any size and may exist anywhere within the address space, but must fall on boundaries of 4k. To program a memory hole, the Unimplemented Memory Register (depicted below) must be programmed with the first unimplemented 4k block of memory and the last unimplemented 4k block of memory for the hole. For example, if the memory address range from 1000-6000 (hex) is to be unimplemented, the low boundary field of the Unimplemented Memory Register should be programmed to 1 (hex) and the high boundary field should be programmed to 5 (hex). If the address range from 1000-2000 (hex) is unimplemented, the low boundary = the high boundary = 1 (hex).

Unimplemented Memory Register (1F46, 9F46)

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

The Unimplemented Memory Register provides the capability to program up to two blocks of unimplemented memory. Each block may be selectively enabled and disabled by programming bits 10 and 11 (EB1 and EB2) of the Control Register. If an access to one of these holes is detected, the PIC asserts the external address error ($\overline{\text{EXT ADR ER}}$) signal which may be directly connected to the $\overline{\text{EXT ADR ER}}$ input on the CPU.

I/O Operations

The PIC also detects accesses to unimplemented and reserved I/O addresses. MIL-STD-1750A specifies four reserved areas within the 64k I/O address space. These areas are 2100-2FFF, 4100-4FFF, A100-AFFF and C100-CFFF (all hex). The PIC will detect any attempt to access these I/O address ranges and assert $\overline{\text{EXADER}}$. In addition to these areas, any attempt to execute an unimplemented command within the legal I/O space (2000-20FF, 4000-40FF, A000-A0FF and C000-C0FF) will also cause the PIC to assert $\overline{\text{EXADER}}$. The PIC also provides the capability to specify a legal range within the user I/O space defined by the standard (0000-03FF and 8000-83FF). Assuming that user I/O commands will be defined sequentially from output address 0000 and input address 8000 (hex), the first unimplemented output command may be specified in the First Unimplemented Output Command Register and the first unimplemented input command may be specified in the First Unimplemented Input Command Register (both registers are shown below). Any user I/O accesses to addresses outside of the ranges specified by these registers will also cause the $\overline{\text{EXADER}}$ output to be asserted. The detection of I/O accesses outside of the user ranges described above may be enabled and disabled by programming bit 12 (EIO) in the Control Register.

First Unimplemented Output Command (1F47, 9F47)

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

First Unimplemented Input Command (1F48, 9F48)

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

In addition to the illegal I/O address detection capability described above, the I/O addresses defined as SPARE by the standard (except those addresses used by the PACE system which are always legal) may be programmed to be legal or illegal. Bit 8 (SPI) of the Control Register, when set to “1”, causes the PIC not to flag accesses to the spare I/O addresses as illegal. When cleared to “0”, the default state, this bit enables the detection and reporting of accesses to the spare addresses as illegal, asserting the $\overline{\text{EXADER}}$ signal.

ADDRESS AND STROBE OUTPUTS

The PIC is capable of generating the address and strobes necessary to drive a memory system directly without external logic. The PIC contains the high-drive transparent address latches necessary to pass the 16-bit address from the IB onto the A bus (A0-A15) and latch it at the falling edge of STRBA. The latched address is output to the system on the A bus which is capable of driving a load of 130 pf. When the optional MMU is implemented, only the least significant 12 address bits are sourced by the PIC on memory cycles, while the remaining 8 extended address bits are sourced by the MMU. The A0 and A1 lines become inputs in this case and should be connected to the two most significant bits of the extended address to divide the memory into four quarters for the wait state generation logic. During I/O cycles, all 16 bits of the address appear on the A bus.

The A bus may be placed into a high-impedance state by the system to accommodate other masters if required. To enable three-state control on the A bus, bit 6 (EAD) in the Control register must be set. This allows the bus to be forced into the high-impedance state when the $\overline{\text{STRBEN}}$ input is LOW.

The PIC also creates the necessary memory and I/O read and write strobes ($\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$) which can also be connected directly to the memory and I/O systems. These lines are internally decoded from the $\overline{\text{STRBD}}$, R/$\overline{\text{W}}$ and M/$\overline{\text{IO}}$ lines (although the read strobes are not actually derived from $\overline{\text{STRBD}}$, but are instead created from the system clock on the same timing as $\overline{\text{STRBD}}$ to avoid the delay of waiting for $\overline{\text{STRBD}}$ to be detected and decoded by

PIC). The strobes do contain the effects of the RAM disable signal used for error correction (described later). MEMR will be inactive during a data correction cycle, as indicated by RAM DIS being active.
The strobes may be forced into a high-impedance state by STRBEN being LOW when three-state control is enabled by bit 5 (EST) in the Control Register.

ERROR DETECTION AND CORRECTION SUPPORT

The PIC supports the EDAC function on the COMBO by providing the signal necessary to disable the system memory during data corrections and by adding the necessary wait states for the correction. Bit 9 in the Control Register (CNF) controls the EDAC support mode of the PIC. When CNF=0, the EDAC mode is disabled (which is the default mode) and the PIC generates the memory parity error (MEPAER) and external address error (EXADER) signals as described earlier. When CNF=1, these two signals change meanings, becoming RAM disable (RAM DIS) and single error (SING ERR) respectively. When the COMBO detects a single-bit error on a memory read data word with EDAC enabled, it generates the SING ERR signal to the PIC. The PIC responds by pulling RDYD LOW to add data wait states and asserts RAM DIS HIGH to notify the COMBO that the system is ready for a data correction. The MEMR strobe includes the effects of the RAM DIS signal. If MEMR is not being used by the system, the RAM DIS signal should be used to disable the system memory so that the COMBO can drive the IB with the corrected data. This application of RAM DIS must be very carefully implemented and is described in detail in Section IV of this note. When the SING ERR line returns low, indicating that the correction cycle is complete, the PIC asserts RDYD to complete the bus cycle and de-asserts RAM DIS to enable the system memory.

FIRST FAILING MEMORY ADDRESS REGISTER

The PIC contains a latch, similar to the address latch described earlier, that continues to latch the address off of the IB on the falling edge of STRBA until a parity error or an external address error is detected. Once either of these errors occurs, the latch is no longer updated until the register is read by the processor. Thus, the first address causing one of these errors is preserved in the register for diagnostic purposes. The register is depicted below.

First Failing Address Register (9F49)

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

SYSTEM CONFIGURATION SUPPORT

The PIC provides the necessary I/O decoding support for the SCR to be read by the processor. In 64-pin packages, the PIC decodes I/O reads to the SCR (8410 hex) and supplies the output SCREN which can be used to enable external logic to drive the 5 system configuration bits onto the IB. In 68-pin packages, the PIC provides 5 system configuration inputs (SC0-SC4) which are driven directly onto the IB during reads from the SCR. This eliminates the need for any external logic.

INTERRUPT ACKNOWLEDGE SUPPORT

The PIC supports the processor's interrupt acknowledge sequence by providing an interrupt acknowledge strobe (INTA) during the interrupt service routine. The PIC decodes the I/O write to address 1000 (hex) which is part of the CPU's interrupt response procedure and provides a pulse on INTA corresponding to the same timing as the IOW pulse. This pulse may be gated with the appropriate bit on the IB to provide an interrupt reset for the requesting device.

START UP ROM SUPPORT

The PIC provides an output signal (STRTROM) which may be used to enable and disable a start up ROM. STRTROM becomes active upon execution of the ESUR I/O command and remains active until execution of the DSUR I/O command. The default at power up is STRTROM active.

BUILT-IN SYSTEM TEST

One of the most important features of the PACE1754 PIC is the built-in system test. The test is capable of checking out the CPU and COMBO chips as well as the PIC itself. It was designed to be run immediately after reset, following the CPU self test. The test is enabled by the $\overline{\text{TEST ON}}$ input to the PIC. If $\overline{\text{TEST ON}}$ is LOW following reset, the test will be run. There is no need to de-assert $\overline{\text{TEST ON}}$ following completion of the test as the test will be run only once for each reset regardless of the state of $\overline{\text{TEST ON}}$. The TEST END output signals the completion of the test and indicates to the system that normal operation is ready to begin. The results of the test may be read in the Status Register after test completion. The most significant three bits are the test results of the CPU, COMBO and PIC respectively with a "1" indicating a pass. For a detailed description of the PIC system test, please consult the PACE1754 PIC System Test Application Note.

3.3 COMBO

3.3.1 Functional Description

The PACE1753 COMBO is a support chip for the 1750A CPU providing a full implementation of the MMU and BPU functions as defined by the standard in addition to many useful system features. It is comprised of the ten major unctional blocks in Figure 3.10 below.

MMU LOGIC

This block performs the memory management function in accordance with MIL-STD-1750A for both CPU and non-CPU memory accesses. It contains the mapping memories and logic necessary to convert logical to physical addresses for up to 1M word of instructions and 1M word of data in 4k word pages. The COMBO is able to perform this address

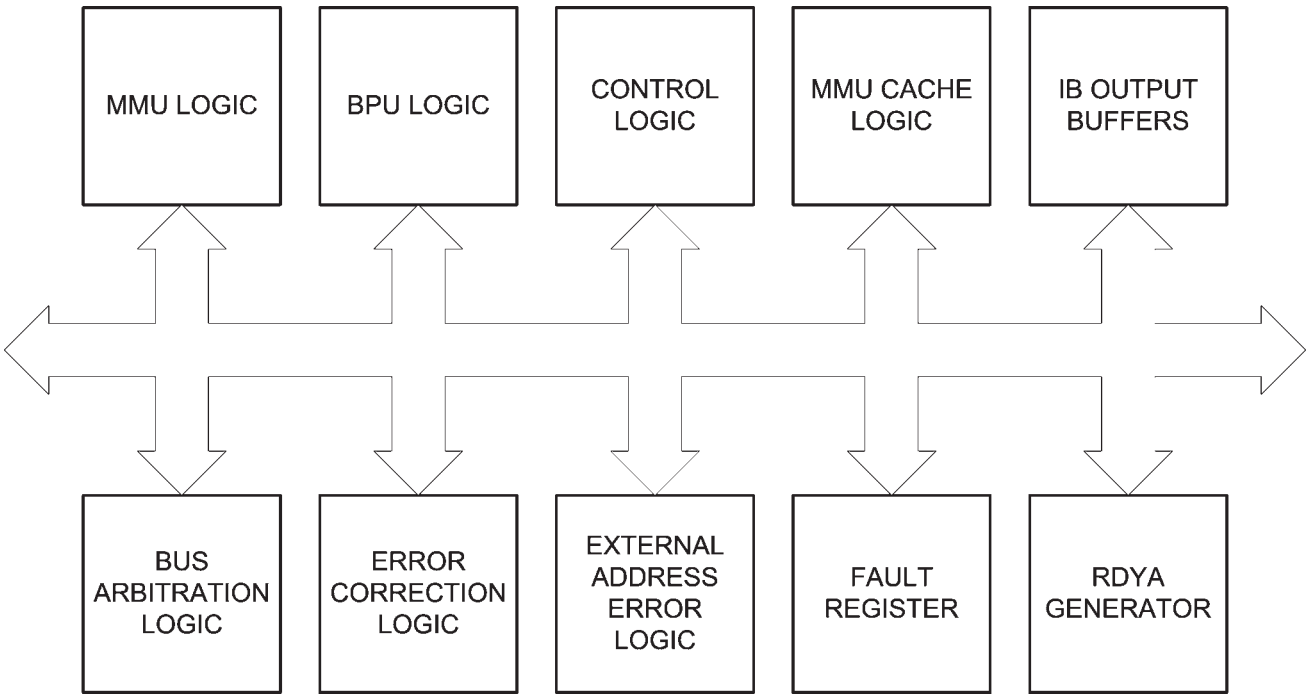


Figure 3.10 COMBO Block Diagram

translation without adding any wait states, even at 40MHz operation, due to a cache mechanism operating within this block. There is a one-word cache containing all of the necessary MMU information for a 4k word page for the instruction memory, another one-word cache for the data memory and another for DMA. This allows zero wait state operation as long as the current memory access is on the same 4k memory page as the previous access (cache hit).

BPU LOGIC

This block performs the block protect function in accordance with the standard. It contains the memories necessary to store the write protection state of each 1k word page of memory for both CPU and DMA accesses. It employs the same cache mechanism as the MMU logic to provide the same zero wait state operation.

CONTROL LOGIC

This block contains all of the necessary logic to perform the control and coordination of the other blocks.

MMU CACHE LOGIC

This block contains the three MMU cache registers and the logic necessary to control them. When a cache miss is detected, the EXT RDY output is de-asserted by this logic to force a wait state during the data phase of the bus cycle.

IB OUTPUT BUFFERS

This block contains the buffers necessary to drive the IB during reads of the COMBO registers and to drive the corrected data onto the IB during error correction read cycles. It has the capability to drive the IB without the use of external buffers.

BUS ARBITRATION LOGIC

This block contains the logic necessary to perform arbitration on the IB for up to four masters. It accepts 4 bus requests and provides 4 grants in a fixed priority scheme.

ERROR CORRECTION LOGIC

This block contains the logic to perform the Error Detection and Correction function. It generates the 6-bit Hamming codes required during writes and checks them during reads for correction of all single-bit errors, detection of all double-bit errors and detection of some multiple-bit errors.

EXTERNAL ADDRESS ERROR LOGIC

This block contains the logic and decoders necessary to detect attempted accesses to unimplemented memory and I/O addresses. It provides the external address error ($\overline{\text{EX AD ER}}$) signal to the processor.

FAULT REGISTER

This block contains the Memory Fault Status Register (MFSR) which stores the current page register when a memory fault occurs, along with other relevant information, in accordance with MIL-STD-1750A. The EDAC fault registers also reside in this block, providing diagnostic capability for EDAC errors.

RDYA GENERATOR

This block provides the RDYA signal to the processor to signal the need for address wait states during bus cycles. Wait states can be requested as programmed by the user, or by the MMU when required by a cache miss.

3.3.2 COMBO Operation

This section describes the use of the PACE1753 COMBO in systems with the PACE1750A CPU. The COMBO may be used in systems either with or without the PACE1754 PIC. For additional information, consult the PACE1753 COMBO data sheet.

MMU

One of the major functions of the COMBO is to provide the memory management unit necessary to expand the memory addressing capability of the CPU from 64k words to 1M word. The MMU works in accordance with MIL-STD-1750A and is described below.

The MMU accepts the logical address from the CPU on the 16-bit IB and the 4-bit Address State (AS) buses and converts it into the 20-bit physical address which is used to address the memory. The least-significant 12 bits of the IB directly become the least-significant 12 bits of the physical address and point to a particular word on a 4k page of memory. If a PIC is being used, these 12 bits may be taken from the A bus on lines A_4-A_{15}. The most-significant 8 bits of the physical address are generated by the MMU on the extended address bus ($\overline{EXT\,ADR_0}$-$\overline{EXT\,ADR_7}$). The extended physical address for each logical address is stored in a page register in the MMU. There are two sets of page registers in the MMU: one for instruction accesses and one for data accesses as determined by the D/$\overline{I}$ output from the CPU. The page register is addressed by the 8 most-significant bits of the logical address (AS_0-AS_3, IB_0-IB_3) giving a total of 256 page registers in each register set. The page registers can be considered as residing in two sets of 16 groups of 16 registers each. The D/$\overline{I}$ input selects the set, the AS bits select the appropriate group and the 4 most significant bits of the IB (called the Logical Page Address – LPA) select the appropriate page within the group. The extended address, also called the Physical Page Address (PPA), is contained within the page register along with the Access Lock (AL) and Execute/Write Protect bits. This memory management scheme allows addressing of the memory in pages of 4k words, each having a different Access Lock and protections state. The MMU function is described pictorially in Figure 3.11.

The Access Lock is a 4-bit field within the page register which allows the 4k memory page addressed by the PPA field to be access protected. The Access Key field of the processor Status Word (SW) is provided by the CPU to the MMU on $\overline{AK_0}$-$\overline{AK_3}$ and compared by the MMU with the AL field in the page register. When there is a mismatch, an access fault ($\overline{MEM\,PRT\,ERR}$) is generated by the MMU to be latched into the Fault Register (FT) in bit 0 for CPU memory accesses and in bit 1 for DMA memory accesses. An AL field of F (hex) is the unlocked state and allows access regardless of the AK value. Similarly, an AK value of 0 is a “master key” which will allow access regardless of the AL value. For all other values, the AK and AL fields must match.

The Execute/Write protect bit of the page register is one additional layer of protection capability provided by the MMU. If the page register is in the instruction mapping register set, the bit is an Execute Protection flag (E). When set to “1”, this bit will prevent any instruction fetching from the memory page by causing the memory access error ($\overline{MEM\,PRT\,ER}$) to be generated resulting in bit 0 of the FT being set. If the page register is in the data register set, the bit is a write protection flag (W). When set to a “1”, the bit prevents writes to any location on the page by generating $\overline{MEM\,PRT\,ER}$ to the processor resulting in bit 0 of the FT being set for CPU memory accesses and bit 1 being set for DMA memory accesses.

The page registers are written and read by the CPU via XIO commands. The instruction page register set is written at I/O addresses 5100-51FF (hex) and read from D100-D1FF (hex). The operand page set is written at I/O addresses 5200-52FF (hex) and read from D200-D2FF (hex). The register format is depicted in Figure 3.12. At power up, all page registers are initialized to map logical addresses directly into physical addresses and the access lock and E/W fileds are all 0’s.

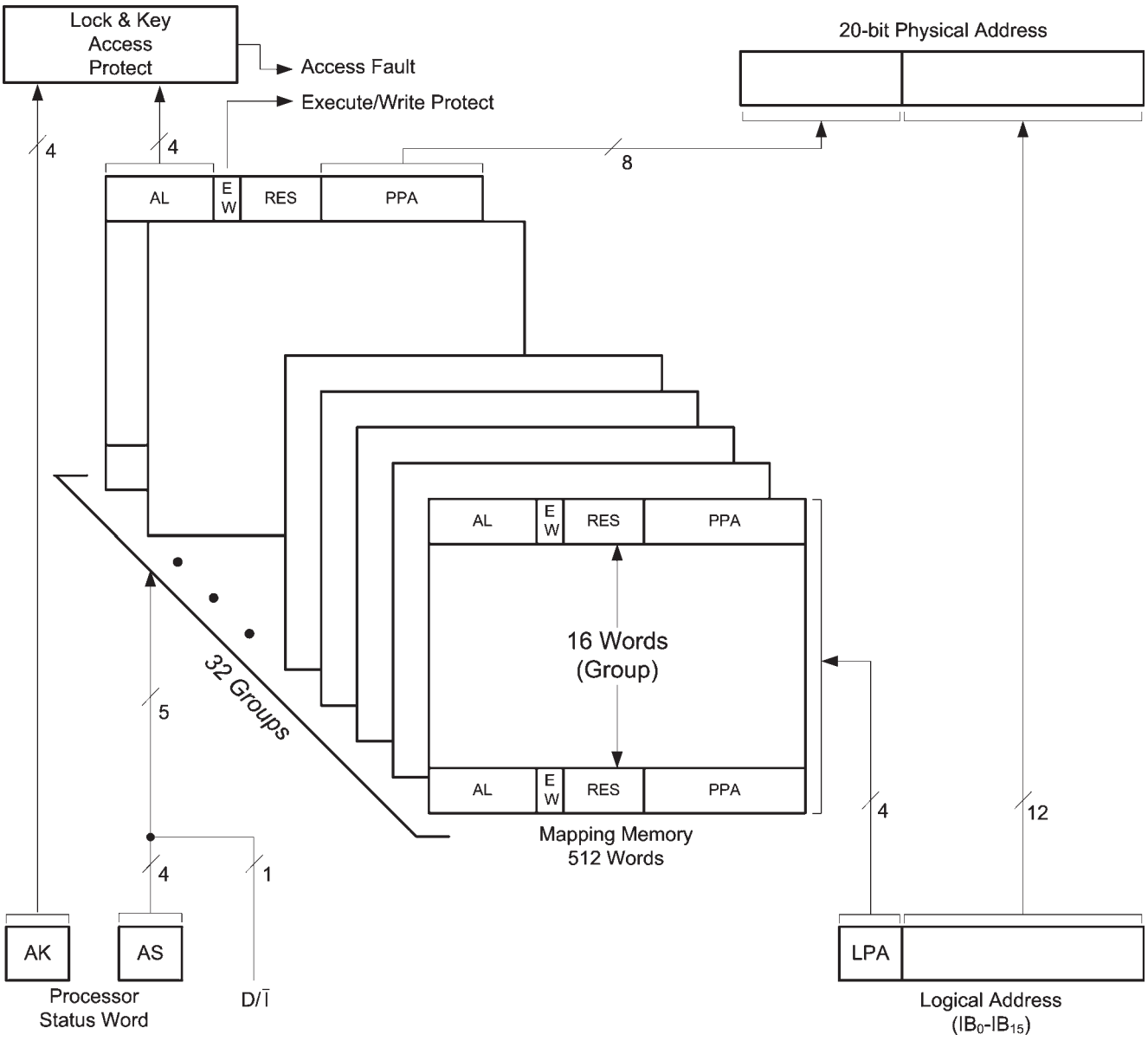


Figure 3.11 MMU Functional Diagram

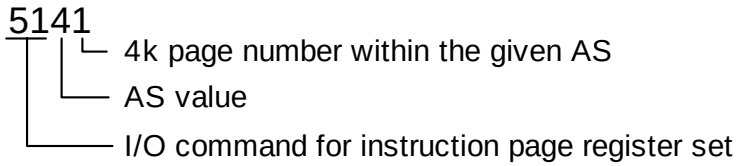
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ACCESS LOCK				E/W	RESERVED			PHYSICAL PAGE ADDRESS (EXT AD)							

Figure 3.12 Page Register Format

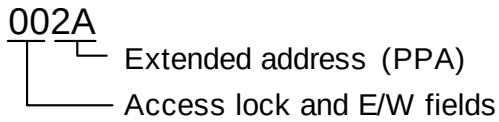
MMU example:

If it is desired to map instruction fetches from logical addresses 1000-1FFF (hex) in address state 4 (hex) into the physical address page 2A000-2AFFF (hex), an XIO instruction to I/O address 5141 (hex) with the data value 002A (hex) must be performed (assuming that the Access Lock is to be 0 and that execute protection is to be off). This is determined as follows:

I/O Address:



Data Word:



A cache is employed on the MMU memory containing the most recently used page register from the operand set for CPU access, the most recently used page register from the operand set for DMA access and the most recently used from the instruction set. This allows the extended address (PPA field of the page register) to become valid as soon as the D/$\overline{I}$ input becomes valid from the CPU (or DMA ACK from the DMA device), which is during the B1 phase of the bus cycle, provided that the current access is on the same 4k memory page as the previous one. This condition is called a cache hit. If there is a cache miss, meaning the current access is not on the same 4k page as the last, the MMU must consult the appropriate register set in the MMU memory to obtain the correct page register and load it into the cache. This cannot be performed without adding a wait state in the bus cycle unless a low clock speed is used, in which case bit 8 in Control Register 0 (SPD) may be programmed to eliminate the cache miss wait state. Before this is done, a careful analysis should be performed to determine whether the extended address will appear early enough in the cycle to permit the full address to be stable during the data phase. Operation without the cache miss wait state is not recommended.

When a cache miss occurs, assuming that SPD is programmed for a wait state, there are two possible paths that the machine can take to complete the cycle. When the COMBO is programmed to provide 0 wait states on RDYA, the MMU does not have enough time to determine that there has been a cache miss and remove RDYA soon enough for the CPU to remain in the address phase of the bus cycle. Thus, the bus cycle moves into B2 and B3 to begin the data phase while the extended address from the COMBO is still invalid. Therefore, the COMBO must add the necessary wait state during the data phase of the bus cycle. It does this by causing its external ready (EX RDY) output to become invalid. If a PIC is used in the system, the EX RDY output from the COMBO should be connected to the EX RDY input of the PIC, which will cause the PIC to remove RDYD from the processor during cache miss cycles, adding the necessary wait states. If no PIC is used, the RDYD wait state must be generated through other means. The extended address becomes valid during the wait state, after which the normal B3 phase of the cycle can begin and the bus cycle can complete. It should be noted that $\overline{\text{STRBD}}$ will become active before the extended address is valid from the COMBO in a cache miss situation. Thus, all strobes which are based solely on $\overline{\text{STRBD}}$ (such as $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ from the P1754 PIC) will also become active before the address is valid. This could cause reading or writing of an incorrect memory location unless design consideration is given to this condition. This will not present a problem during writes provided the $\overline{\text{WR PROT}}$ strobe from the COMBO is used as the write strobe. For further discussion, please see Section IV of this note.

If there is at least one address wait state programmed in the COMBO, the process is somewhat different. The MMU now has enough time to determine that there has been a cache miss and add the additional wait state required to look up the new page register during the address phase of the cycle. The extra wait state is added first, during which the extended address will become valid, and then the number of wait states programmed is counted off.

BPU

The other major function of the COMBO is to provide the Block Protect Unit in accordance with the standard. The BPU is another layer of memory protection in addition to the access protection provided by the MMU. The BPU provides memory write protection for each 1k memory block over the entire physical memory address space. To do this, the BPU contains two memories, each with 64 16-bit locations. One of these memories contains the protection information for CPU memory accesses and the other for DMA accesses. Each bit of the BPU memories contains the write protection value for a single 1k memory block. A "1" in a given bit means that the memory block that it represents is write protected. An attempt to write to a block that is protected will cause the BPU to generate $\overline{\text{MEM PRT ER}}$ to the processor, setting bit 0 in the FT for CPU accesses and bit 1 for DMA accesses. The BPU memory is written and read by the CPU via XIO commands to I/O addresses 5/D000-5/D03F (hex) for the CPU protection memory and 5/D040-5/D07F (hex) for the DMA protection memory. The most significant bit of the data word contained in the first I/O address corresponds to the first 1k block of memory (00000-0003FF hex) in the physical address space. At power up, the BPU is initialized to all 0's, meaning that every memory address is unprotected; however, the BPU is disabled and the memory globally protected until the MPEN I/O command is executed, after which the protection is as defined by the BPU.

BPU example:

When it is desired that the 1k memory block in the physical address range 21C00-21FFF (hex) be protected against DMA writes, an XIO to I/O address 5048 with a data value of 0100 (hex) must be performed. This is determined as follows:

The binary representation of the block address is:

[001000] [0111] XXXXXXXXXX

The six most significant bits, when taken as a hex number, determine the BPU memory location – 08. This is added to 5040 (hex) to determine the I/O address for the DMA protection memory. The next 4 bits encode the bit number of the BPU memory location (data word) that should be set to protect the correct block – 7.

COMBO CONTROL

The COMBO is controlled by two 16-bit registers: Control Register 1 and Control Register 0. These registers control the various operating modes and option provided by the chip. They are depicted in Figures 3.13 and 3.14 respectively. More detailed discussions of the control bits are presented in the sections following later in this note.

Control Register 1 (1F51, 9F51)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
WA0	WA1	SPI	PIB	PEG	IDL	RESERVED									

Default = C3FF

WA0/WA1 – These two bits encode the number of address wait states to be added by the COMBO during bus cycles. 0-3 wait states may be programmed.

SPI – A "1" in this bit disables illegal I/O detection of I/O addresses defined by MIL-STD-1750A as spare. (1=spare I/O legal, 0=spare I/O illegal).

PIB – Not used at this time.

PEG – A "1" in this bit enables generation of parity during memory write cycles when neither EDAC nor parity checking are enabled. (1=parity generation, 0=EDAC generation).

IDL – A "1" in this bit causes the bus arbiter to insert one idle (Bi) state between all non-locked bus cycles.

RESERVED – Always read as 1's.

Figure 3.13 COMBO Control Register 1

Control Register 0 (1F50, 9F50)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QR1	QR2	QR3	QR4	ODD	EEI	EED	EPR	SPD	WPT	EB1	EB2	EIO	GPT	DMX	DLP

Default = 00C6

QR1 – A “1” in this bit enables parity or EDAC checking on physical memory addresses 00000-3FFFF (hex).

QR2 – A “1” in this bit enables parity or EDAC checking on physical memory address 40000-7FFFF (hex).

QR3 – A “1” in this bit enables parity or EDAC checking on physical memory addresses 80000-BFFFF (hex).

QR4 – A “1” in this bit enables parity or EDAC checking on physical memory address C0000-FFFFF (hex).

ODD – A “1” in this bit causes parity generation and checking to be odd. A “0” causes even parity.

EEI – A “1” in this bit enables EDAC generation and checking on instruction fetches.

EED – A “1” in this bit enable EDAC generation and checking on data reads and writes.

EPR – A “1” in this bit enables parity generation and checking.

SPD – A “1” in this bit enables the insertion of one address wait state on MMU cache miss cycles. It is recommended that this bit remain a “1”.

WPT – A “1” in this bit causes the $\overline{\text{WR PROT}}$ output to be generated as a write protect strobe. A “0” causes this output to be generated as a write protect flag.

EB1 – A “1” in this bit enables illegal address detection of block 1 of unimplemented memory as defined in the Unimplemented Memory Register.

EB2 – A “1” in this bit enables illegal address detection of block 2 of unimplemented memory as defined in the Unimplemented Memory Register.

EIO – A “1” in this bit enables illegal address detection for reserved I/O addresses and user I/O addresses defined in the First Unimplemented Input and Output Registers.

GPT – A “1” in this bit re-enables global memory protect after the MPEN I/O command has been executed during PIC system test. Valid only during system test.

DMX – A “1” in this bit enables a de-multiplexed address/data bus during DMA cycles.

DLP – A “1” in this bit enables logical DMA mode. A “0” enables physical DMA mode.

Figure 3.14 COMBO Control Register 0

MEMORY WRITE PROTECTION STROBE/FLAG

The COMBO provides an output which is generated by the MMU and BPU to indicate to the system when writing to the memory is permitted by the protection programming and when it is not. This signal, $\overline{\text{WR PROT}}$/PROT FLAG, may be used either as an active-low memory write strobe ($\overline{\text{WR PROT}}$) or as an active-high memory protection flag (PROT FLAG). Bit 9 in Control Register 0 (WPT) determines whether the output is a strobe or a flag.

When used as a strobe, the signal approximates the timing of $\overline{\text{STR BD}}$ for legal (unprotected writes), but remains HIGH during reads and during writes to protected memory addresses. It is intended to be a write enable signal to the system memory. The $\overline{\text{WR PROT}}$ strobe also compensates for the unusual timing for writes during an MMU cache miss. Since the wait state necessary for the MMU to look up the new page register and BPU protection programming on cache miss cycles does not occur until the data phase of the bus cycle (during zero address wait state operation only), $\overline{\text{STR BD}}$ becomes active before the extended address and protection state are valid. Thus, if $\overline{\text{STR BD}}$ were used as a write enable signal for the memory, a write to an incorrect or protected address could occur. The $\overline{\text{WR PROT}}$ signal timing is such that both the address and protection are valid before $\overline{\text{WR PROT}}$ becomes active, guaranteeing proper operation.

When used as a flag, PROT FLAG becomes active during attempts to write to protected memory. It can be combined with $\overline{\text{STR BD}}$ to form a protected write strobe for the memory if care is taken for the cache miss situation described above. If the COMBO is programmed to insert one or more wait state in the address phase of a bus cycle, there will be no problem with this approach. If zero wait state operation is the design goal; however, the cache miss timing must be considered very carefully. External logic could be implemented by using the EX RDY output from the COMBO to indicate a cache miss, and creating a write protect strobe similar to $\overline{\text{WR PROT}}$ by combining PROT FLAG, $\overline{\text{STR BD}}$ and the clock during the cache miss situation.

READY GENERATION

The COMBO provides the capability to add wait states during the address phase of a bus cycle much like the PIC does for the data phase. By programming the WA0 and WA1 bits in Control Register 1 to a binary value between 0 and 3, 0 to 3 wait states will be added to the address phase, B1, of the bus cycle. The COMBO generates the RDYA signal to the CPU to perform this function. Since it is presumed that the same address decode logic is responsible for the entire memory space and that this is the logic requiring the extra time during the address phase, the COMBO cannot be programmed to provide a different number of wait states for different areas of memory like the PIC can for the data phase. The number of wait states programming in Control Register 1 are applied to the entire memory and I/O address space.

PARITY AND EDAC SUPPORT

The COMBO has the capability to provide both parity and EDAC for memory error control. These functions are controlled by bits in both Control Register 0 and Control Register 1. Parity or EDAC is always being generated, regardless of whether checking of either is enabled. Bit 4 in Control Register 1 (PEG) determines which is being generated if neither is enabled. This is to allow copying of data from a memory without parity or EDAC into a memory with parity or EDAC. The 6 EDAC Hamming code bits are output and input on signals EDC_0-EDC_5. When in the parity mode, EDC_0 becomes the parity bit while EDC_1-EDC_5 continue to output the Hamming codes.

Parity Mode

The parity mode is enabled by bit 7 of Control Register 0 (EPR). When this bit is set, parity is automatically generated, regardless of the state of the PEG bit, and checking is enabled according to the state of bits 0-3 in Control Register 0 (QR1-QR4). These four bits allow division of the memory into four quarters, each of which may or may not have parity implemented. Both odd and even parity are supported by the COMBO. Bit 4 in Control Register 0 (ODD) determines whether odd or even parity is to be used. When a parity error is detected, the COMBO generates the memory parity error output signal ($\overline{\text{MEM PAR ER}}$) to the CPU, where it is latched into bit 2 of the FT.

EDAC Mode

The EDAC mode is enabled by bits 5 and 6 in Control Register 0 (EEI and EED). These bits, when either is set, automatically enable EDAC generation, regardless of the state of the PEG bit. EEI enables EDAC checking for instructions fetches on the quarter(s) of memory enabled by the QR1-QR4 bits. EED enables checking for data reads, also on the enabled memory quarter(s). Either of these bits or both may be set. If EPR is set in addition to either EEI or EED, the EDAC mode will take precedence.

The EDAC logic is capable of detecting all single and double-bit errors, as well as some multiple-bit errors. Single-bit errors are corrected as well as detected. When a single-bit error is detected, the COMBO generates the SING ERR signal which is intended to be connected to the PIC. The PIC responds with the RAM DIS signal, which is meant to

disable the system memory and notify the COMBO to proceed with the correction. It should be noted that the RAM DIS signal from the PIC initially has the meaning of the memory parity error signal ($\overline{\text{ME PAR ER}}$), which is active LOW, until the PIC is programmed to support EDAC. This means that the RAM DIS output cannot be used directly to disable the memory, as the memory would be disabled at power up by the normally high level on the $\overline{\text{ME PAR ER}}$ signal. Thus, some logic must be implemented to hold off the RAM DIS signal from the memory until after the PIC has been programmed to support EDAC. Two wait states are inserted by the PIC after SING ERR is received to allow time for the memory to become high-impedance and for the COMBO to make the correction. The COMBO drives the corrected data onto the IB for the CPU to read, and the cycle completes.

The physical and logical addresses of the memory location requiring the correction, as well as the bit position of the failing data bit are saved in COMBO registers for fault management and diagnostic purposes. The address is saved in two registers: the First Failing Physical Address Register (9F59) and the EDAC Memory Fault Status Register (9F5B). The First Failing Physical Address Register contains the least-significant 16 bits of the physical address (EXT AD_4-EXT AD_7, IB_4-IB_{15}). The most-significant 4 bits (EXT AD_0-EXT AD_3) are stored in the EDAC Memory Fault Status Register along with other status information including all of the information found in the MIL-STD-1750A MFSR. The failing data bit is stored in the First Failing Data Register (9F5A), in which the bit position that required correction is set to 1. If the failure is in one of the Hamming Code bits (EDC_0-EDC_6), the bit position of the failing bit is encoded in a three bit field of the EDAC MFSR. This situation can be detected by reading all zeroes in the First Failing Data Register after a single-bit error has occurred. A 5 in this field indicates an error on EDC_5, etc. All of these registers will remain frozen after a failure until they are read by the CPU, after which they are again ready to store failure information. These registers are depicted below.

EDAC Memory Fault Status Register (9F5B)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LPA				EXT AD (0:3)				FAILING EDC			ID	AS			

First Failing Physical Address Register (9F59)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIRST FAILING PHYSICAL ADDRESS - EXT AD(4:7), A(4:15)															

First Failing Data Register (9F5A)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BIT POSITION OF FIRST CORRECTED BIT															

ILLEGAL ADDRESS DETECTION

The COMBO provides the capability to detect and report attempted accesses to unimplemented or illegal memory and I/O addresses. Any access to a memory or I/O address determined to be unimplemented or illegal will cause the COMBO to assert its external address error ($\overline{\text{EXT ADR ER}}$) output to the CPU where it is latched into bit 5 or bit 8 of the FT, for I/O and memory accesses, respectively.

I/O Accesses

Bit 12 in Control Register 0 (EIO) enables or disables the detection of I/O accesses to addresses within the areas defined as reserved by the standard, as well as attempts to access addresses outside of the user ranges specified by the First Unimplemented Input and Output Registers. When EIO is a “1”, all attempts to read or write to a reserved address will cause the external address error. This also allows the user to program a range of legal (implemented) I/O addresses within the space designated by the standard as user I/O space (0000-03FF for writes, 8000-83FF for

reads). This is accomplished by defining sequentially from address 0000 the range of legal output addresses and programming the First Unimplemented Output Command register (1F57) with the first unimplemented address, and then repeating the process for inputs by programming the First Unimplemented Input Command Register (1F58) with the first unimplemented input address when defined sequentially from address 8000 (hex).
When EIO is a “1”, I/O accesses to addresses outside of the programmed ranges will cause the external address error. These registers are depicted below.

First Unimplemented Output Command Register (1F57, 9F57)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NOT USED						FIRST UNIMPLEMENTED OUTPUT COMMAND									

First Unimplemented Input Command Register (1F58, 9F58)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NOT USED						FIRST UNIMPLEMENTED INPUT COMMAND									

The I/O addresses defined by the standard as spare may also be considered legal or illegal. Bit 2 of Control Register 1 (SPI) determines whether spare I/O addresses are legal. When SPI = 0 (the default value), accesses to spare I/O addresses (except for those addresses dedicated to the PACE 1750A system) cause the COMBO to generate the external address error. When SPI =1, spare I/O addresses are considered legal. Table 3.9 lists the address ranges defined as spare and reserved by the standard.

Table 3.9 Spare/Reserved I/O Addresses

OUTPUT RANGE	INPUT RANGE	USAGE
0400-1FFF	8400-9FFF	SPARE
2100-2FFF	A100-AFFF	RESERVED
3000-3FFF	B000-BFFF	SPARE
4100-4FFF	C100-CFFF	RESERVED
5300-7FFF	D300-FFFF	SPARE

Memory Accesses

The COMBO can be programmed to insert two “holes” within the full memory address space, within which reads and writes to the memory will cause the external address error to occur. The addresses of the holes are programming in Unimplemented Memory Registers 1 and 2 and the holes are enabled by bits 10 and 11 (EB1 and EB2) in Control Register 0. The Unimplemented Memory Registers are depicted below. To program a hole, the memory must be considered as constructed of blocks of 4k words each. The first unimplemented 4k block number is programmed in the BL LO field and the last unimplemented 4k block of the hole is programmed in the BL HI field of the appropriate Unimplemented Memory Register. Once a hole is programmed, the corresponding EB1 or EB2 bit must be set for the hole to be detected. In this manner, either hole #1, hole #2, both or neither may be implemented at any time.

Unimplemented Memory Register 1 (1F55, 9F55)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL1 LO								BL1 HI							

Unimplemented Memory Register 2 (1F56, 9F56)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL2 LO								BL2 HI							

BUS ARBITER

The COMBO has the capability to arbitrate the IB among up to four users on a fixed priority scheme. It performs this function through four bus request / bus grant pairs ($\overline{\text{BUS REQ}}_{0:3}$ and $\overline{\text{BUS GNT}}_{0:3}$) and the bus lock signal ($\overline{\text{BUS LOCK}}$). When a bus user requires access to the bus, it activates its bus request ($\overline{\text{BUS REQ}}_n$) output to the COMBO. If the request is currently the highest priority (0 is the highest priority, 3 is the lowest) and the bus is not locked by another bus user ($\overline{\text{BUS LOCK}}$ is HIGH), the COMBO will issue the corresponding bus grant output signal ($\overline{\text{BUS GNT}}_n$) to the requesting device. That device can then take control of the bus.

If the requesting device is of higher priority than the device currently in control of the bus, the arbiter de-asserts the current $\overline{\text{BUS GNT}}$ on the rising edge of the clock following the new $\overline{\text{BUS REQ}}$ and asserts the new $\overline{\text{BUS GNT}}$ on the rising edge of the clock following the rising edge of $\overline{\text{BUS LOCK}}$, signifying the end of the current cycle. This results in one idle clock cycle occurring between the two bus cycles, ensuring that the bus has been completely releases by the first user before the second begins to drive it.

If the requesting device is of lower priority than the device currently in control of the bus, the current $\overline{\text{BUS GNT}}$ is not de-asserted until the current user no longer requires the bus, as signified by its $\overline{\text{BUS REQ}}$ being inactive. When this is true, the current $\overline{\text{BUS GNT}}$ is removed on the rising edge of the clock following the completion of the cycle ($\overline{\text{BUS LOCK}}$ going HIGH). This is to ensure that the higher priority device retains the bus until it has finished with it. One clock later, the new $\overline{\text{BUS GNT}}$ is issued by the COMBO allowing the new user to take control. This results in two idle clock cycles occurring between the bus cycles.

In either case, the device taking control of the bus should issue the $\overline{\text{BUS LOCK}}$ signal within two rising clock edges of receiving the $\overline{\text{BUS GNT}}$ signal to avoid losing the bus to a higher priority request. If there is no new bus request after the current master has released the bus, the COMBO will maintain the current bus grant active until there is a new request, even if the current master is not requesting it. Thus, the bus grant is like a token which is passed among the bus users and is never taken away unless another user requests it.

DMA SUPPORT

The COMBO can support DMA in any of three modes to match the requirements of a particular system. The COMBO will enter the DMA mode anytime the DMA acknowledge input signal (DMA ACK) is active. This signal causes the BPU to select the DMA protection table for the write protect function and places the COMBO into the DMA mode programmed in Control Register 0.

Logical DMA

In this mode, the COMBO receives the 20-bit logical address of the transfer on AS(0:3) and IB(0:15) and translates it through the MMU into the 20-bit physical address, just as though the DMA device or controller were a CPU. The BPU functions just as it does with a CPU with the exception that it consults the DMA memory rather than the CPU memory for the protection map. The physical address is provided to the BPU internally. The IB functions as a multiplexed

address and data bus, thus requiring the DMA device to supply the full 16-bit IB, the 4-bit AS, STRBA, $\overline{\text{STRBD}}$, R/$\overline{\text{W}}$, D/$\overline{\text{I}}$ and M/$\overline{\text{IO}}$ to the COMBO. The bus timing in this mode, in relation to CPU clock edges, must match, as closely as possible, the timing of a CPU bus cycle. To enter this mode, bit 15 of Control Register 0 (DLP) must be set to "1".

Physical DMA – Multiplexed Address/Data Bus

In this mode, the MMU is by-passed and the full physical address is supplied by the DMA device on AS(0:3) and IB(0:15). The COMBO passes the 8 most significant bits, AS(0:3) and IB(0:3), through to the extended address outputs (EXT AD(0:7)) and uses tem, in addition to IB(4:5), to address the BPU memory. Since the IB is still multiplexed with address and data, the DMA device must still supply IB, AS, STRBA, $\overline{\text{STRBD}}$, R/$\overline{\text{W}}$ and M/$\overline{\text{IO}}$ to the COMBO. The bus cycle timing, in relation to CPU clock edges should again match the CPU timing as closely as possible. To enter this mode, bit 15 of Control Register 0 (DLP) must be cleared to "0" to designate physical addressing, and bit 14 of Control Register 1 must be cleared to "0" to designate a multiplexed bus.

Physical DMA – De-multiplexed Address/Data Bus

In this mode, the MMU is by-passed and the COMBO has nothing to do with address generation whatsoever. The 20-bit physical address is generated by the DMA device on EXT ADR(0:7) and A(4:15). The extended address outputs on the COMBO become inputs, which the COMBO uses to address the BPU. In order to provide the most significant 10 bits of physical address to the COMBO for the BPU function, A(4:5) must be input in addition to the extended address. This is accomplished by supplying A(4:5) to the COMBO on AS(0:1) through three-state buffers which are enabled only during DMA transfers, such as when the DMA device has received $\overline{\text{BUS GNT}}$ and DMA ACK is active. The data portion of the bus cycle is driven on the IB just as before. The DMA device must now supply only EXT ADR, AS(0:1), STRBA, R/$\overline{\text{W}}$ and M/$\overline{\text{IO}}$ to the COMBO. These signals should be supplied with the timing approximating that of a CPU as closely as possible. This mode is further simplified when the BPU is bypassed in addition to the MMU. This takes the COMBO completely out of the transfer, allowing completely asynchronous DMA to occur. To do this, the memory protect error signal ($\overline{\text{MEM PRT ER}}$) must be prevented from reaching the CPU during DMA. To enter the physical/de-multiplexed DMA mode, bit 15 of Control Register 0 (DLP) must be cleared to "0" to denote physical addressing and bit 14 of Control Register 0 must be set to "1" to denote a de-multiplexed bus.

IV. System Design Considerations

This section describes the design of complete systems based on the PACE1750A and its support chips: the PACE1753 COMBO and the PACE1754 PIC. Much of the information presented can also be applied to systems designed without the PIC and/or COMBO. Further information may be obtained from the PACE1750A, 1753 and 1754 data sheets, including detailed timing diagrams and AC parameters.

4.1 Memory Speed

One of the first decisions to be made when designing a PACE1750A system is what the memory size and speed should be to meet throughput and cost requirements. Though the parallel nature of the CPU minimizes the effect of slow memory on system performance, the best performance is always obtained when memory fast enough to operate without wait states is used. Table 4.1 lists the DAIS mix throughput performance of the PACE1750A at various clock frequencies and wait state settings.

The memory access time required to operate without wait states is highly dependant on the particular address decoding scheme chosen by the designer. To assist the system designer with the task, a memory selection example is given in this note. Figure 4.1 depicts a simple PACE1750A system and an accompanying block of memory. In this implementation, the decoding task is performed by the logic depicted in Figure 4.2 and the Address Bus is buffered for each bank of memory by "244" octal buffers/line drivers. The decoder supports 1 M word of memory based on a 256k x 1 bit static CMOS RAM devices. The RAM is organized in 4 banks of 256k words each.

Table 4.1 DAIS Mix Throughput

CLOCK SPEED	0 WAIT STATES	1 WAIT STATE	2 WAIT STATES
20 Mhz	1.26 MIPS	1.12 MIPS	1.01 MIPS
30 Mhz	1.89 MIPS	1.68 MIPS	1.51 MIPS
40 Mhz	2.52 MIPS	2.24 MIPS	2.02 MIPS

Fast data buffers are placed between the memory and the PACE1750A system to allow the memory devices to be enabled early in the cycle without contention on the IB. Separate read and write data buses are implemented to take advantage of the separate read and write data pins available on the memory devices. The buffers are depicted in Figure 4.4.

The decoder outputs 6 signals: $\overline{\text{RAM CE}}$(0:3), $\overline{\text{RD OE}}$ and $\overline{\text{WD OE}}$. $\overline{\text{RAM CE}}$(0:3) is a group of signals used to enable the correct bank of RAM as appropriate. It is controlled by equation (1) below. $\overline{\text{RD OE}}$ is used as the output enable for the read data buffer and is controlled by equation (2) below. $\overline{\text{WD OE}}$ is used as the output enable for the write data buffer and is controlled by equation (3) below.

(1) $\overline{\text{RAM CE}}_x = \overline{\text{BUS BUSY}} + \text{M}/\overline{\text{IO}} + \overline{\text{RAM ADDRESS}}_x$
(2) $\overline{\text{RDATA OE}} = (\overline{\text{M}/\text{IO}} + \text{R}/\overline{\text{W}}) + \overline{\text{STRBD}}$
(3) $\overline{\text{WDATA OE}} = (\overline{\text{BUS BUSY}} * \text{M}/\text{IO} * \text{R}/\overline{\text{W}})$

A particular $\overline{\text{RAM CE}}$ will become active anytime there is a memory bus cycle to the appropriate address, as indicated by $\overline{\text{BUS BUSY}}$ being active and M/$\overline{\text{IO}}$ in the memory state with the appropriate address determined by the 2 most significant bits of the 20-bit address bus. $\overline{\text{RD OE}}$ is asserted anytime $\overline{\text{STRBD}}$ is active during a memory read cycle (as indicated by M/$\overline{\text{IO}}$ in the memory state and R/$\overline{\text{W}}$ in the read state). With this design, the proper RAM devices are enabled shortly after the address is valid, allowing them to begin the access as early as possible. The data buffers are enabled shortly after $\overline{\text{STRBD}}$ becomes active, when the IB is no longer driven with the address. During write cycles, $\overline{\text{WD OE}}$ becomes active shortly after $\overline{\text{BUS BUSY}}$ becomes active and remains active until shortly after $\overline{\text{BUS BUSY}}$ becomes inactive, guaranteeing sufficient data set up and hold times for the memory.

The worst case timing occurs during a memory read. The timing for this case is presented in Figure 4.3. To determine the access time required for the memory devices, the following parameters are required:

(1) $TC(ST)_V$ Time from B1 to D/$\overline{\text{I}}$ valid
(2) $TD/I(EXT\ ADR)_V$ Time from D/$\overline{\text{I}}$ valid to extended address valid
(3) TP(F138) F138 propagation delay
(4) TP(244A) 244A propagation delay
(5) $TC(IBD)_V$ IB data set-up time

The memory access time (TA) may be determined from the following equation:

$$TC(ST)_V + TD/I(EXT\ ADR)_V + TP(F138) + TP(244A) + TC(IBD)_V + TA = 3T$$

where T is the clock period. Table 4.2 below lists the various parameters for the above equation and the derived TA values for zero wait state operation at 20, 30, and 40Mhz clock frequencies. The parameters used are worst case for the military grade of each part type. If wait states are added, TA may be increased by one T for each wait state.

For zero wait state operation at a 40 MHz clock rate, a memory with an access time of 25.9 ns or less must be used.

The memory write cycle timing is a little less stringent than the read cycle timing. For completeness, the memory write cycle timing is shown in Figure 4.5. The worst case write timing requires a memory access time greater than that required for the read. Thus, if the selected memory meets the read timing requirements, it also meets the write cycle requirements.

The most significant requirement for the memory devices during the write cycle is the pulse width of the write enable signal. If the $\overline{\text{WR PROT}}$ signal is used as the write enable input to the memory as shown in this example, it should be noted that the minimum pulse width that Pyramid Semiconductor is able to guarantee for that signal is 16 ns at 40 Mhz. Thus, only memory devices capable of operation with write enable pulses of 16 ns or less may be used under worst case conditions.

This memory system loads the IB with only one CMOS Input and the Address Bus with only four CMOS inputs. This is well within the drive capability of the CPU, PIC and COMBO. The address buffers are each loaded with 16 CMOS inputs, which is within their drive capability; however, it is assumed that the buffers and memory devices do reside on the same board such that there is no capacitance added by a backplane or motherboard. The data buffers are each loaded with four CMOS inputs, which is well within their drive capability.

Table 4.2 Memory Access Times

PARAMETER	20 MHZ	30 MHZ	40 MHZ
$TC(ST)_V$	30 ns	20 ns	15 ns
$TD/I(EXT\ ADR)_V$	25 ns	20 ns	15 ns
TP(F138)	9.5 ns	9.5 ns	9.5 ns
TP(244A)	4.6 ns	4.6 ns	4.6 ns
$TC(IBD)_V$	5 ns	5 ns	5 ns
TA	75.9 ns	40.9 ns	25.9 ns

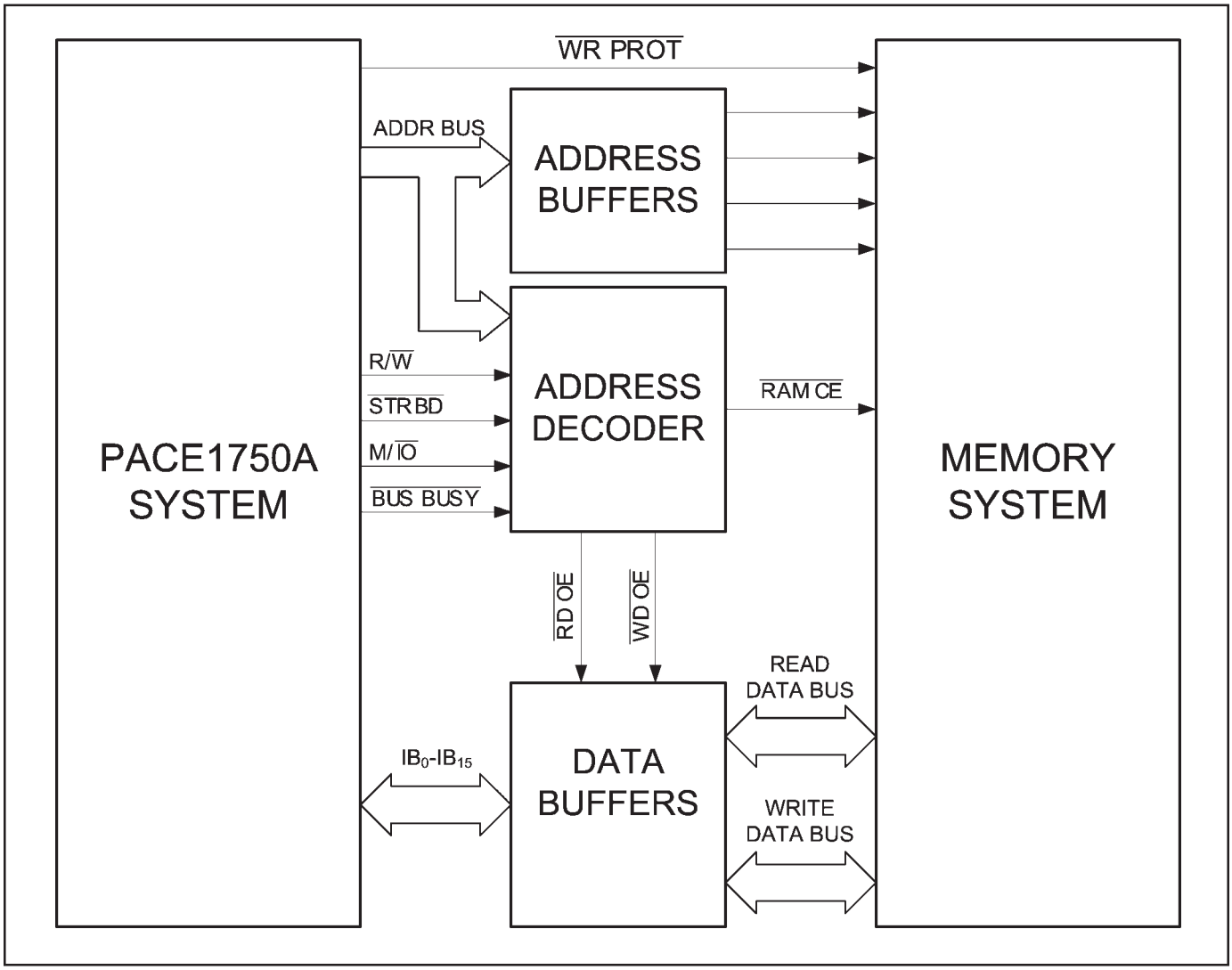


Figure 4.1 Sample Memory Interface

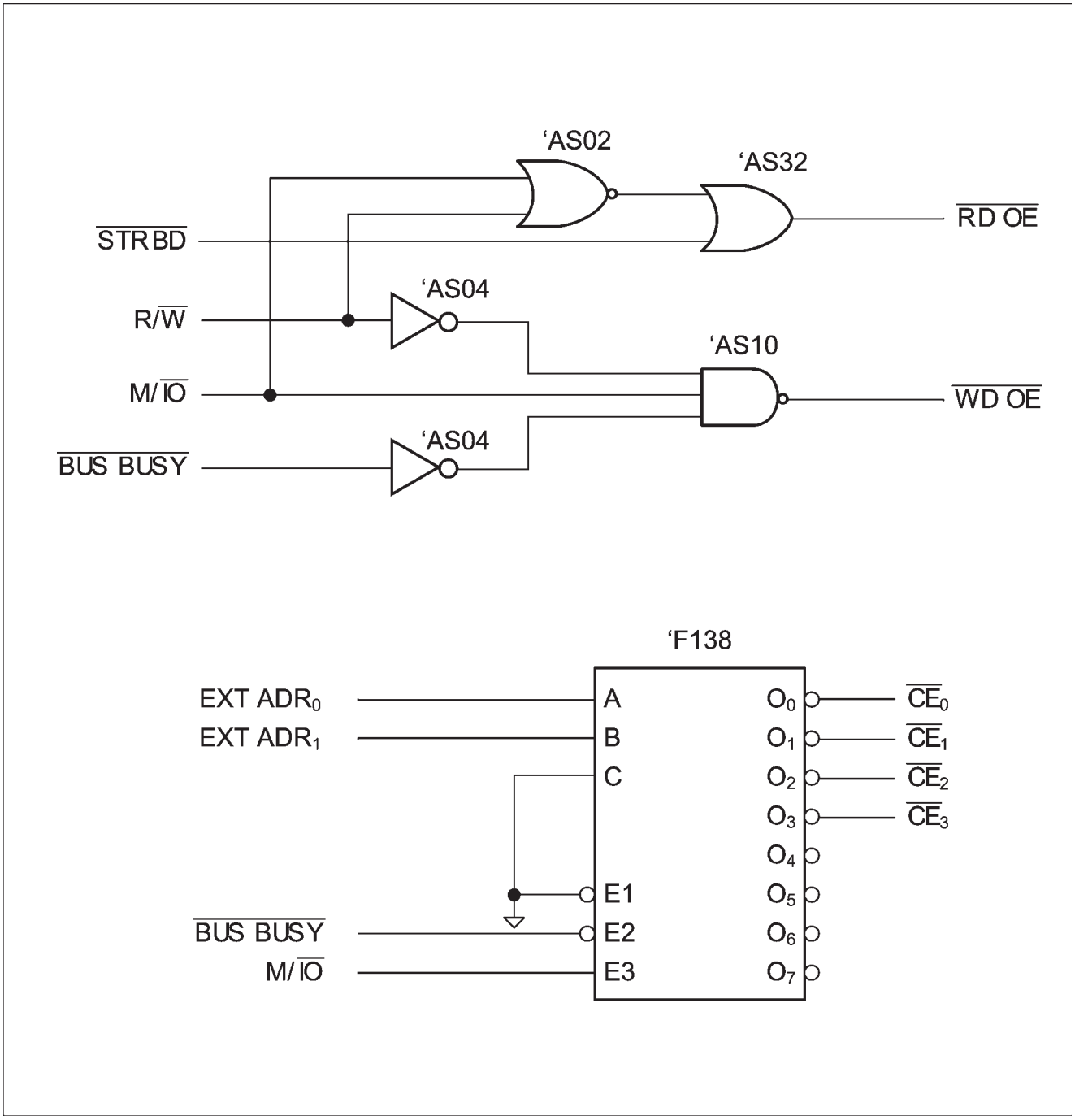


Figure 4.2 Address Decoder Logic

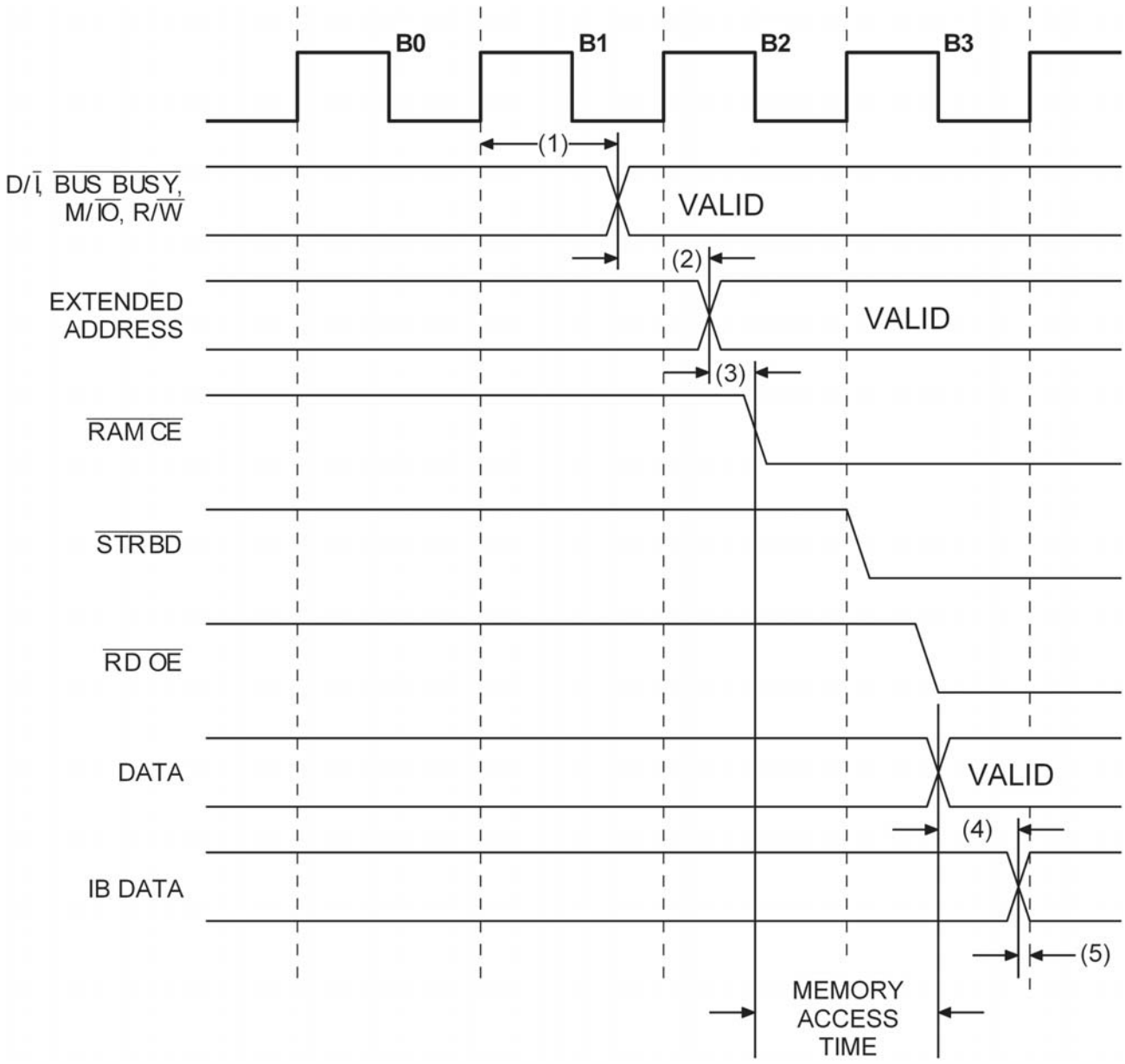


Figure 4.3 Memory Read Timing

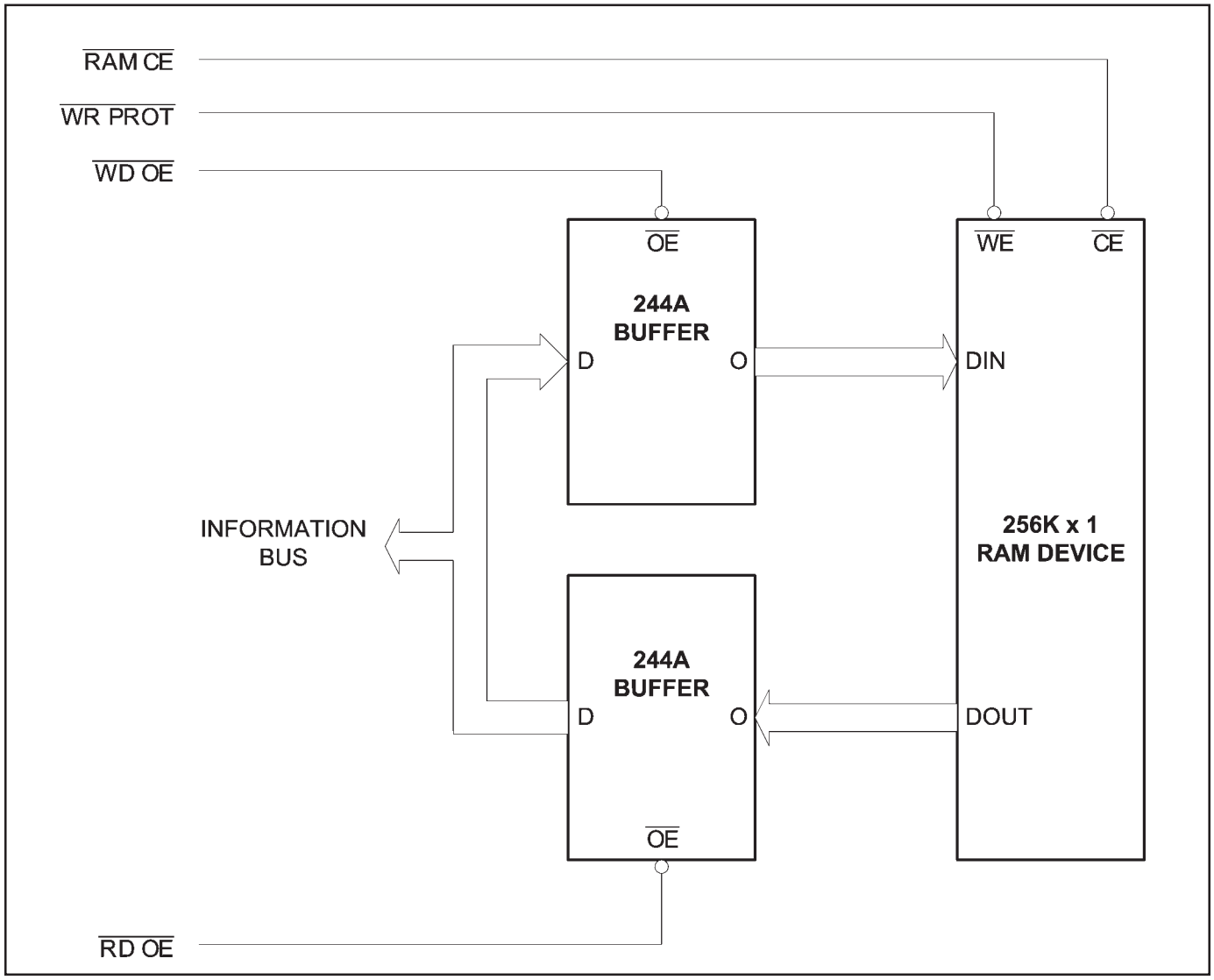


Figure 4.4 Data Buffers

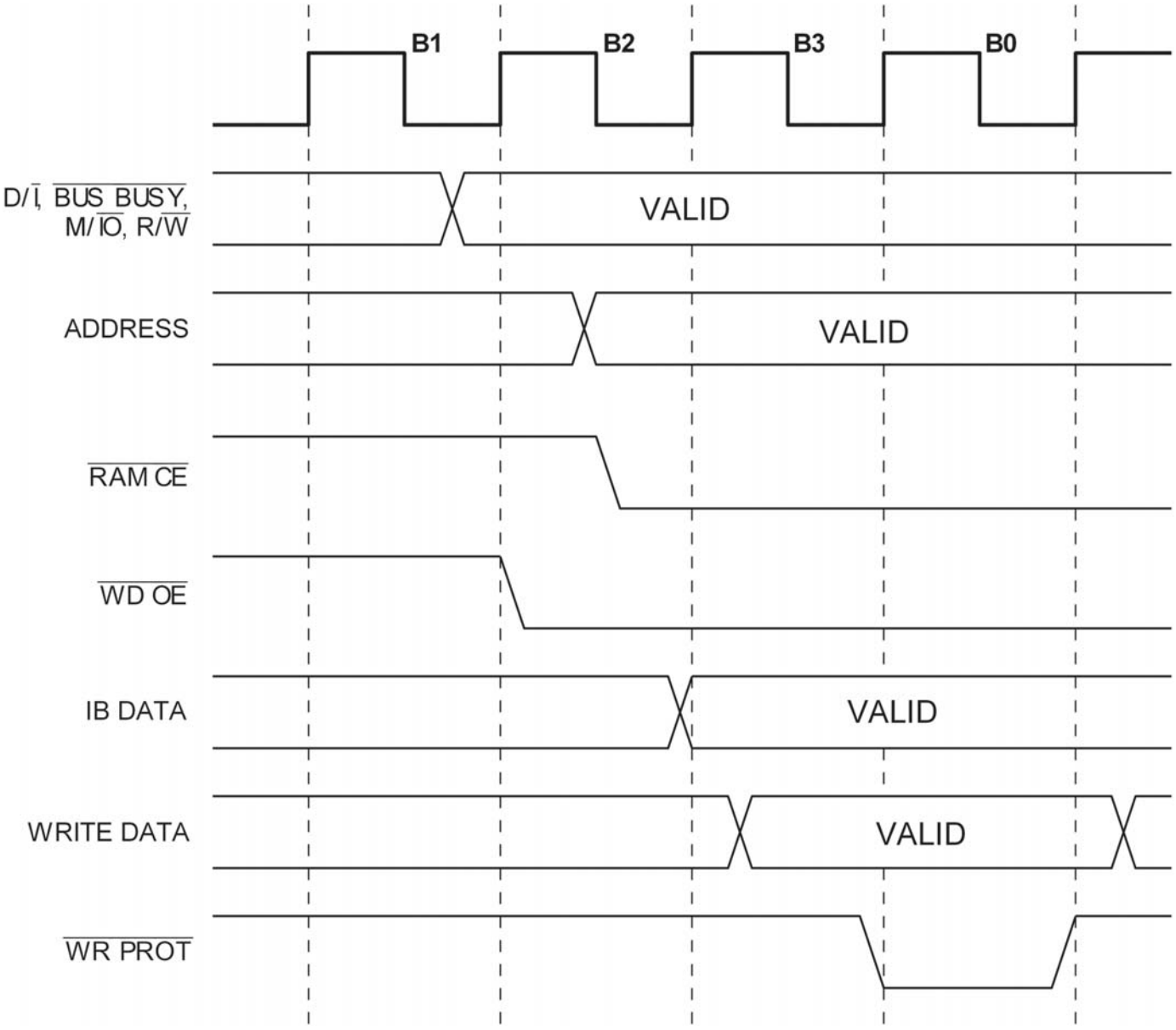


Figure 4.5 Memory Write Timing

4.2 Signal Interconnection

PULL-UP AND PULL DOWN RESISTORS

Due to the high-impedance state of the bus signal outputs from the CPU during non-bus cycles, it is important to force some of the CPU outputs to their inactive states with pull-up or pull-down resistors, as appropriate, to avoid system bus errors. These signals are listed in Table 4.3.

STRBA and $\overline{\text{STRBD}}$ should be pulled to their inactive states to prevent them from being falsely recognized during non-bus cycles. The PIC chip already contains 40 k resistors for this purpose which should be sufficient when CMOS devices are connected to the signals. When TTL devices are used, external resistors should be added. D/$\overline{\text{I}}$ should be pulled to some state other than high-impedance to prevent the COMBO chip from oscillating between the data and instruction MMU memories during non-bus cycles. $\overline{\text{BUS BUSY}}$ and $\overline{\text{BUS LOCK}}$ should be pulled inactive because they are inputs to the CPU during non-bus cycles that are used to inform the processor whether the bus is locked by another

Table 4.3 Signals Requiring External Resistors

SIGNAL	RESISTOR TYPE	VALUE	NOTES
STRBA	PULL-DOWN	10 kΩ	40 kΩ provided in P1754 PIC
$\overline{\text{STRBD}}$	PULL-UP	10 kΩ	40 kΩ provided in P1754 PIC
D/$\overline{\text{I}}$	PULL-UP OR DOWN	10 kΩ	
$\overline{\text{BUS BUSY}}$	PULL-UP	10 kΩ	
$\overline{\text{BUS LOCK}}$	PULL-UP	10 kΩ	

bus master. The CPU will not take control of the bus when $\overline{\text{BUS LOCK}}$ and $\overline{\text{BUS BUSY}}$ are recognized as active even if its $\overline{\text{BUS GNT}}$ input is active. In addition to the above recommendations, all unused interrupt and fault inputs should be pulled inactive.

4.3 Designing With PIC and COMBO

The PIC and COMBO devices have some features that overlap, so the designer must choose which chip is to be used for which feature, In particular, the following features are provided, with some differences, by both the PIC and COMBO:

1. Illegal I/O detection
2. Unimplemented memory detection
3. Parity

Following is a description of the differences between the two devices and recommendations for how they should be used together.

ILLEGAL I/O DETECTION

The PIC and COMBO devices function identically for detecting illegal and unimplemented I/O commands. The only restriction is that only one of the devices may have its external address error output ($\overline{\text{EXT ADR ER}}$) connected to the CPU at a time. Thus, the chip that ends up being connected to the CPU during the system design is the only chip capable of performing the illegal I/O detection and reporting function.

UNIMPLEMENTED MEMORY DETECTION

The PIC and COMBO function similarly when detecting accesses to unimplemented memory with one major difference: the PIC is only capable of functioning within a 64k word address space while the COMBO's memory detection logic functions over the full 1M word address space. Therefore, in designs requiring 64k of memory or less, the PIC may be used for unimplemented memory detection, but in designs requiring greater than 64k words of memory, the COMBO must be used to implement this feature. If the PIC is used, the external address error ($\overline{\text{EXT ADR ER}}$) must be sourced by the PIC. If the COMBO is used, $\overline{\text{EXT ADR ER}}$ must be sourced by the COMBO. Since $\overline{\text{EXT ADR ER}}$ is used by the processor to signal both illegal I/O and memory accesses, whichever chip is used to implement the memory detection must also be used to implement the I/O detection.

PARITY

The PIC and COMBO both provide programmable memory parity generation and detection. Like the unimplemented memory detections, the PIC is limited to 64k words of memory while the COMBO can function over the full 1M word of address space. If the PIC is used, the memory parity error signal to the CPU ($\overline{\text{MEM PAR ER}}$) must be connected to the PIC. If the COMBO is used for parity, $\overline{\text{MEM PAR ER}}$ must be sourced by the COMBO.

4.4 EDAC Implementation

The use of the EDAC feature on the COMBO is most easily accomplished when a PIC is used also. If a PIC is not used, hardware will have to be designed to perform the SING ERR / RAM DIS handshake and correction wait state generation with the timing matching that of the PIC as closely as possible.

When a PIC is used, a decision must be made regarding the use of the memory read strobe ($\overline{MEMR}$) from the PIC. If this signal is used to enable the memory data IB drivers, the PIC will ensure that the memory drivers are inhibited during the single error correction cycle when the data must be driven by the COMBO. If $\overline{MEMR}$ is not used as the memory data enable, the RAM disable (RAM DIS) output from the PIC must be used to place the memory data drivers into a high-impedance state during the correction cycle. It is recommended that the $\overline{MEMR}$ strobe be used whenever possible. If system requirements dictate that $\overline{MEMR}$ not be used, there must be some external logic implemented to allow the system to function at initial power up.

The initial default state of the PIC is without EDAC support. Thus, the SING ERR and RAM DIS signals will default to $\overline{EXT\ ADR\ ER}$ and $\overline{MEM\ PAR\ ER}$, respectively, after initial power up. The memory parity error signal ($\overline{MEM\ PAR\ ER}$) is active LOW, and thus will be HIGH except when a parity error is detected. RAM DIS is active HIGH, indicating that the memory drivers should be disabled when the signal is HIGH. Therefore, if RAM DIS is gated with $\overline{STR}$ $\overline{BD}$, R/$\overline{W}$ and M/$\overline{IO}$ to provide a memory read strobe, the system memory will be disabled initially after power up. Since the only way to change the meaning of $\overline{MEM\ PAR\ ER}$ to RAM DIS is by programming the PIC to support EDAC, the system will be hung unless a start-up ROM is used or a circuit is implemented to prevent the memory from being disabled until after the PIC has been set for EDAC. Such a circuit is depicted in Figure 4.6.

The circuit functions by nullifying the effect of RAM DIS on the system memory until a write to the PIC Control Register (1F40) is detected with bit 9 (CNF) set to “1”. Once the CNF bit is set, the RAM DIS signal is enabled and will inhibit the system memory. Two 8-bit comparators are used to decode the I/O write operation to address 1F40. A 54F74 D flip-flop is used to latch the CNF bit of the Control Register. The CNF bit is gated with RAM DIS and provides an inhibit signal to the $\overline{MEMR}$ generation circuit (a 54F138 decoder). If either RAM DIS or CNF is LOW, $\overline{MEMR}$ will not be inhibited.

Another important consideration is the SING ERR Input to the PIC. Like RAM DIS, the SING ERR signal is initially active LOW (EXT ADR ER). When the PIC is programmed to support EDAC, the signal switches to the active-HIGH SING ERR input. Since, under normal conditions, the $\overline{EXT\ ADR\ ER}$ signal is initially HIGH (inactive), when the signal switches to the SING ERR input, the capacitance of the line can hold it HIGH long enough to be detected by the PIC as a single error, erroneously placing the PIC in the correction mode. This can be corrected by connecting a pull-down resistor of about 1k on the SING ERR line. This will force the signal LOW before an error can be detected.

It should be noted that enabling EDAC requires that some extra time be provided during the data phase of the memory read cycle to allow the COMBO to determine whether the data read from the memory requires a correction and notify the PIC if a correction cycle must be initiated. The worst case timing path is described below:

$TC(ST)_V$	Time from B1 to D/$\overline{I}$ valid
$TD/I(EXT\ ADR)_V$	Time from D/$\overline{I}$ to EXT ADR valid
T_{acc}	Time from address valid to memory data valid
$TIBD_V(SING\ ERR)_H$	Time from memory data valid to SING ERR active
$TEX\ RDY(\overline{RDYD})_V$	Time from SING ERR active to $\overline{RDYD}$ inactive
$TRD_V(C)$	$\overline{RDYD}$ to B0 setup time

The above timing path must occur within three CPU clock periods (T) or $\overline{RDYD}$ will not arrive at the processor soon enough to hold the cycle in the data phase for a correction to occur if a single bit error was detected by the COMBO. The equation below is used to derive T_{acc}:

$$T_{acc} = 3T - TC(ST)_V - TD/I(EXT\ ADR)_V - TIBD_V(SING\ ERR)_H - TEX\ RDY(\overline{RDYD}) - TRD_V(C)$$

Using the published values for the above parameters, Table 4.4 below lists the required T_{acc} for the three system speed grades.

Table 4.4 EDAC T_{acc}

PARAMETER	20 MHz	30 MHz	40 MHz
$TC(D/I)_V$	30 ns	20 ns	15 ns
$TD/I(EXT\ ADR)_V$	25 ns	20 ns	15 ns
$TIBD_V(SING\ ERR)_H$	35 ns	30 ns	25 ns
TEX RDY(RDYD)	14 ns	11 ns	8 ns
TRD(C)	5 ns	5 ns	5 ns
T_{acc}	41 ns	14 ns	7 ns

Note that T_{acc} is the total time required to access the memory system and includes the address decode delay and IB buffer delay (if buffers are used). Thus, operation at 30 and 40 MHz is difficult without the addition of a wait state. This is an unfortunate but necessary penalty to pay for the extra protection that EDAC provides to a system. If 30 MHz operation without wait states is required to achieve higher throughput, using 40 MHz devices with a 30 MHz clock provides a T_{acc} of 32 ns, which should be sufficient to operate the EDAC without wait states. During the correction cycle, two wait states are automatically added by the PIC to allow the COMBO to make the data correction. These wait states are transparent to the user and are in addition to any wait states added as described above.

4.5 The $\overline{WR\ PROT}$/PROT FLAG Signal

The COMBO provides an output pin to be used for allowing protected writes into memory. The signal may be programmed to function in one of two ways, according to the state of bit 9 in Control Register 0 (WPT). When set to “1”, the default state, the signal becomes $\overline{WR\ PROT}$, a protected write strobe. This signal is meant to be connected to the memory write enable input as a write strobe. The timing of the signal usually follows that of $\overline{STRBD}$, except during a cache miss cycle with 0 address wait states programmed in the COMBO. $\overline{WR\ PROT}$ includes the effects of both the MMU and BPU protection features, becoming active only during writes to unprotected memory addresses.

When WPT is set to “0”, the signal becomes PROT FLAG. In this mode, the output is HIGH when the memory is unprotected and writes should be allowed, and LOW when the write should not be allowed. PROT FLAG should be gated with $\overline{STRBD}$, M/$\overline{IO}$ and R/$\overline{W}$ to form a memory write strobe. This gating function is simple if the COMBO is programmed for one or more wait states during the address phase. If the COMBO is programmed for zero address wait states, the task becomes more difficult. For this reason, it is recommended that the signal be used as the $\overline{WR\ PROT}$ strobe if possible.

If it is necessary to use the signal in the PROT FLAG mode due to system constraints, and zero address wait state operation is required, a special circuit must be designed to handle the delicate timing during the MMU cache miss cycle. The problem arises due to the fact that $\overline{STRBD}$ becomes active before the MMU and BPU have been able to look up the correct protection values and extended address for the memory access when a cache miss has occurred. Thus, if $\overline{STRBD}$ is used as the write strobe, a write to an incorrect or protected address may be erroneously allowed. Thus, the circuit must be able to detect the cache miss situation and hold off the write strobe until after the protection state and extended address are stable and correct. Such a circuit is illustrated in Figure 4.7.

The cache miss situation is detected by watching the EXT RDY output from the COMBO. This signal is driven LOW by the COMBO only when a cache miss has occurred. When the EXT RDY output goes LOW, an inhibit signal to the write strobe is immediately created and will remain active for one clock cycle, until the D flip-flop is able to clock in $\overline{STRBD}$. During this time, the BPU is looking up the protection state for the memory location addressed. If, after the correct protection values have been obtained, the PROT FLAG indicates that the memory location should be protected (PROT FLAG LOW), the $\overline{WR\ PROT}$ strobe will remain inhibited, preventing the write. If the write should b allowed, the $\overline{WR\ PROT}$ signal will become active with some delay after the rising edge of the first clock following the occurrence of $\overline{STRBD}$ and will become inactive with some delay after the rising edge of $\overline{STRBD}$. The write strobe will also be inhibited if the bus cycle is a read or I/O operation.

The timing of the $\overline{\text{WR PROT}}$ signal, when generated in this way, is similar to the timing of the $\overline{\text{WR PROT}}$ signal illustrated in Figure 3.5 in the COMBO data sheet for the cache miss situation and to Figure 3.1 in the COMBO data sheet for a cache hit.

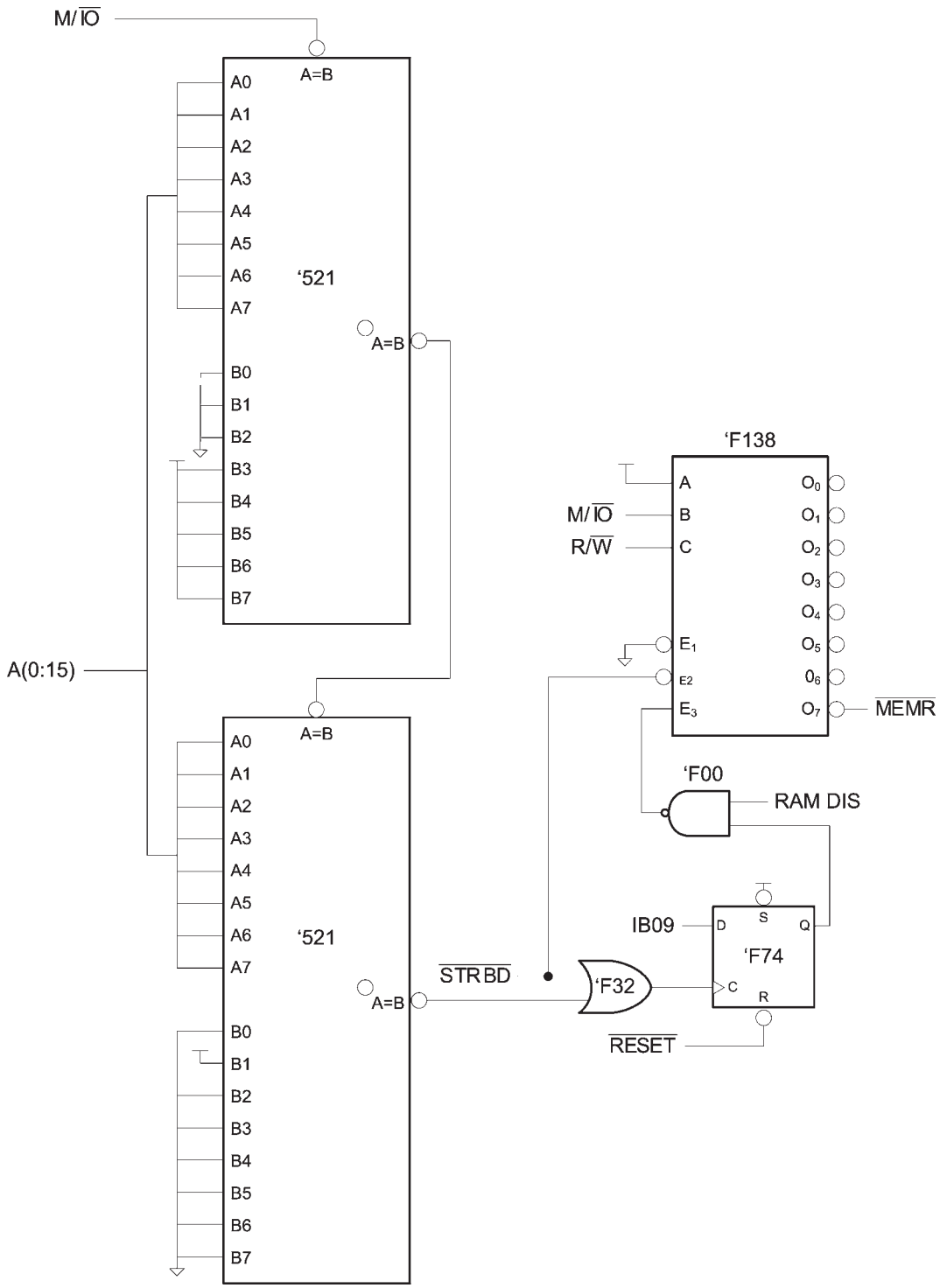


Figure 4.6 RAM Disable Circuit

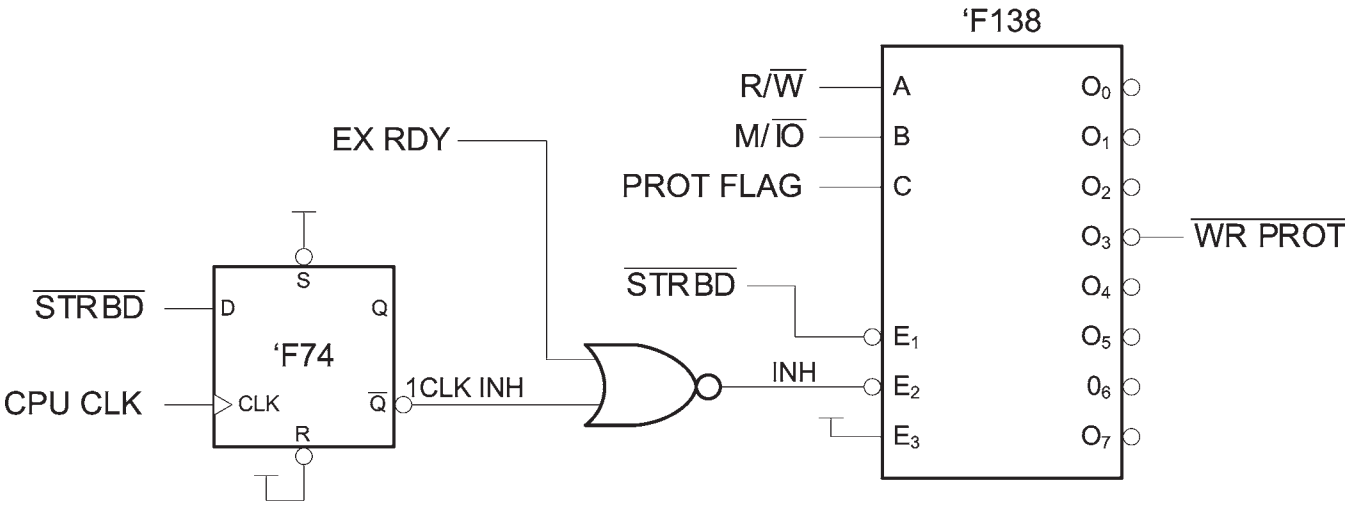


Figure 4.7 Write Protect Generation

Appendix A
PIC Register Map

Control Register (1F40, 9F40)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PR1	PR2	PR3	PR4	ODD	EST	EAD	EXR	SPI	CNF	EB1	EB2	EIO	LIO	LME	0

Default = 0000

Status Register (1F41, 9F41)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPU	CMB	PIC	RESERVED		STB	ADR	TWD	TBT	RESERVED						IFL

Default = 0000

Memory Ready Program Register (1F42, 9F42)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MEM Q1				MEM Q2				MEM Q3				MEM Q4			

Default = FFFF

I/O Ready Program Register (1F43, 9F43)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IO Q1				IO Q2				IO Q3				IO Q4			

Program Register (1F44, 9F44)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLOCK FREQUENCY (MHZ)						EBT	SBT	EWD	SWD	RESERVED					

Default = 0000

Watch Dog Timer Register (1F45, 9F45)

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

Default = 0000

Unimplemented Memory Register (1F46, 9F46)

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

First Unimplemented Output Command (1F47, 9F47)

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

First Unimplemented Input Command (1F48, 9F48)

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

First Failing Address Register (9F49)

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

Appendix B
COMBO Register Map

Control Register 0 (1F50, 9F50)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QR1	QR2	QR3	QR4	ODD	EEI	EED	EPR	SPD	WPT	EB1	EB2	EIO	GPT	DMX	DLP

Default = 00C6

Control Register 1 (1F51, 9F51)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
WA0	WA1	SPI	PIB	PEG	IDL	RESERVED									

Default = C3FF

Unimplemented Memory Register 1 (1F55, 9F55)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL1 LO								BL1 HI							

Unimplemented Memory Register 2 (1F56, 9F56)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL2 LO								BL2 HI							

First Unimplemented Output Command Register (1F57, 9F57)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NOT USED						FIRST UNIMPLEMENTED OUTPUT COMMAND									

First Unimplemented Input Command Register (1F58, 9F58)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NOT USED						FIRST UNIMPLEMENTED INPUT COMMAND									

First Failing Address Register (9F59)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FIRST FAILING PHYSICAL ADDRESS - EXT AD(4:7), A(4:15)															

First Failing Data Register (9F5A)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BIT POSITION OF FIRST CORRECTED BIT															

EDAC Memory Fault Status Register (9F5B)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LPA				EXT AD (0:3)				FAILING EDC			ID	AS			

Memory Fault Status Register (A00D)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LPA				RESERVED							ID	AS			

REVISIONS

DOCUMENT NUMBER: ANP16-002 **DOCUMENT TITLE**: APPLICATION NOTE - P1750A SYSTEM DESIGN			
REV.	ISSUE DATE	ORIG. OF CHANGE	DESCRIPTION OF CHANGE
ORIG	May-89	RKK	New Data Sheet
A	Oct-05	JDB	Added Pyramid logo