Section 3. System Board

Description

This section describes the various system board components and how they interact. Hardware descriptions and programming information is provided to acquaint hardware designers and programmers with the basic operation of the system.

System Microprocessor

The 80286 microprocessor sub-system has the following:

- 10-MHz clock operation
- 24-bit address
- 16-bit data interface
- Extensive instruction set, including string I/O
- Hardware fixed-point multiply and divide
- Two operational modes
  - 8086-compatible Real Address Mode
  - Protected Virtual Address Mode
- 16M (M = 1,048,576 or 2^{20}) of physical address space
- 1G (G = 1,073,741,824 or 2^{30}) of virtual address space

Real Address Mode

In the Real Address Mode, the system microprocessor address space is a contiguous array of up to 1M. The system microprocessor addresses memory by generating 20-bit physical addresses.

The segment portion of the pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower 4 bits of the 20-bit segment address are always zero. Therefore, segment addresses begin on multiples of 16 bytes.

All segments in the Real Address Mode are 64K in size and can be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (for example, a word with its low-order byte at offset hex FFFF and its high-order byte at hex 0000). If, in the Real Address Mode, the information contained in the segment does not use the full 64K, the

unused end of the segment can be overlaid by another segment to reduce physical memory requirements.

Protected Virtual Address Mode

The Protected Virtual Address Mode (Protected Mode) offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

The Protected Mode provides a 1-gigabyte virtual address space per task mapped into a 16-megabyte physical address space. The virtual address space may be larger than the physical address space, because any use of an address that does not map to a physical memory location will cause a restartable exception.

As in the Real Address Mode, the Protected Mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector specifies an index into a memory-resident table rather than the upper 16 bits of a real memory address. The 24-bit base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address. The tables are automatically referred to by the system microprocessor whenever a segment register is loaded with a selector. All instructions that load a segment register refer to the memory-based tables without additional program support. The memory-based tables contain 8-byte values.

Performance

The 80286 microprocessor operates at 10 MHz for a clock cycle time of 100 nanoseconds.

The system inserts one wait state whenever it accesses system board RAM or ROM, which results in a 300 nanosecond, 16-bit memory cycle time. The system inserts a minimum of one wait state whenever it does a system board I/O operation, which results in a minimum I/O cycle time of 300 nanoseconds.

The refresh controller operates at 10 MHz, which results in a clock cycle time of 100 nanoseconds. The bandwidth for memory refresh is approximately 7.0%.

The bus cycle time for memory operations is 200 nanoseconds for direct memory access (DMA) operations, with the DMA controller operating at 10 MHz with no wait states inserted. I/O operations and system board memory operations have one wait state inserted.

80287 Math Coprocessor

The optional 80287 Math Coprocessor enables the system to perform high-speed arithmetic, logarithmic, and trigonometric operations. The coprocessor works in parallel with the microprocessor. The parallel operation decreases operating time by allowing the coprocessor to do mathematical calculations while the microprocessor continues to do other functions.

The coprocessor works with seven numeric data types, which are divided into the following three classes:

- Binary integers (3 types)
- Decimal integers (1 type)
- Real numbers (3 types).

Programming Interface

The coprocessor offers extended data types, registers, and instructions to the microprocessor. The coprocessor has eight 80-bit registers, which provide the equivalent capacity of forty 16-bit registers. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access and improving speed as well as bus availability. The register space can be used as a stack or as a fixed register set. When used as a stack, only the top two stack elements are operated on.

The following figure shows representations of large and small numbers in each data type.

Data Type	Bits	Significant Digits (Decimal)	Approximate Range (Decimal)
Word Integer	16	4	$-32{,}768 \leq x \leq +32{,}767$
Short Integer	32	9	$-2 \times 10^{9} \leq x \leq +2 \times 10^{9}$
Long Integer	64	19	$-9 \times 10^{18} \leq x \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-9..99 \leq x \leq +9..99$ (18 digits)
Short Real *	32	6 - 7	$8.43 \times 10^{-37} \leq x \leq 3.37 \times 10^{38}$
Long Real *	64	15 - 16	$4.19 \times 10^{-307} \leq x \leq 1.67 \times 10^{308}$
Temporary Real **	80	19	$3.4 \times 10^{-4932} \leq x \leq 1.2 \times 10^{4932}$

Figure 3-1. Data Types

* The Short Real and Long Real data types correspond to the single- and double-precision data types.

** The Temporary Real data type corresponds to the extended-precision data type.

Hardware Interface

The coprocessor uses the same clock generator as the microprocessor and operates at 10 MHz in the asynchronous mode. The coprocessor is wired so that it functions as an I/O device through I/O port addresses hex 00F8, 00FA, and 00FC. The microprocessor sends OP codes and operands through these I/O ports. The microprocessor also receives and stores results through the same I/O ports. The coprocessor 'busy' signal informs the microprocessor that it is executing; the microprocessor Wait instruction forces the microprocessor to wait until the coprocessor is finished executing.

The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor sets its error signal. This error signal generates a hardware interrupt (interrupt hex 13) and causes the 'busy' signal to the coprocessor to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write

command to address hex 00F0 with D7 through D0 equal to 0. This action also clears interrupt hex 13.

The power-on self-test code in the system ROM enables IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI interrupt vector. This maintains code compatibility across the IBM Personal Computer and Personal System/2 product lines. The NMI interrupt handler reads the coprocessor status to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control is passed to the original NMI interrupt handler.

The coprocessor has two operating modes similar to the two modes of the microprocessor. When reset by a power-on reset, system reset, or an I/O write operation to port hex 00F1, the coprocessor is in the real-address mode. This mode is compatible with the 8087 Math Coprocessor used in IBM Personal Computers. The coprocessor is placed in the protected mode by executing the SETPM ESC instruction. It is placed back in the real mode by an I/O write operation to port hex 00F1, with D7 through D0 equal to 0.

Detailed information for the internal functions of the Intel 80287 Coprocessor can be found in books listed in the Bibliography.

8087 to 80287 Compatibility

The 80287 operating in the Real Address Mode can execute 8087 software without major modifications. However, because of differences in the handling of numeric exceptions by the 80287 and the 8087, exception-handling routines *may* need to be changed.

The following summarizes the differences between the 80287 and 8087 Math Coprocessors, and provides details showing how 8087 software can be ported to the 80287 Math Coprocessor.

- The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80287 environment. If the 80287 encounters one of these opcodes in its instruction stream, the instruction is effectively ignored; none of the 80287 internal states are updated. While 8086 code containing these instructions may be executed on an 80287, it is unlikely that the exception-handling routines

containing these instructions will be completely portable to the 80287.

- The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
- In the Protected Mode, the format of the 80287 saved instruction and address pointers is different than the format of the 8087. The instruction opcode is not saved in the Protected Mode; exception handlers have to retrieve the opcode from memory if needed.
- Interrupt 7 occurs in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a Wait instruction also causes interrupt 7. An exception handler should be included in 80286 code to handle these exceptions.
- Interrupt 9 occurs if the second or subsequent words of a floating-point operand fall outside a segment size. Interrupt 13 occurs if the starting address of a numeric operand falls outside a segment size. An exception handler should be included in the 80286 code to report these programming errors.
- Most 80287 numeric instructions are automatically synchronized by the 80286. The 80286 automatically tests the 'busy' signal from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. Explicit Wait instructions are not required to ensure this synchronization. An 8087 used with 8086 and 8088 system microprocessors requires explicit Waits before each numeric instruction to ensure synchronization. Although 8086 software having explicit Wait instructions executes perfectly on the 80286 without reassembly, these Wait instructions are unnecessary.

  The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of the mnemonic. The WAIT forms of these instructions cause the assembler to precede the ESC instruction with a microprocessor Wait instruction.

DMA Controller

The Direct Memory Access (DMA) controller allows I/O devices to transfer data directly to and from memory. This allows a higher system microprocessor throughput by freeing the system microprocessor of I/O tasks.

The DMA controller is software programmable. The system microprocessor can address the DMA controller and read or modify the internal registers to define the various DMA modes, transfer addresses, transfer counts, channel masks, and page registers.

The functions of the DMA controller can be grouped into two categories: program condition and DMA transfer. During program condition, the DMA registers can be programmed or read.

Program condition commences when the system microprocessor refers to the DMA controller within a specific address range. These addresses are identified in Figure 3-3 on page 3-13. The DMA controller needs a minimum of 300 nanoseconds (1 wait state) to complete any program command generated by the system microprocessor.

The second function of the controller is the actual DMA transfer. This period is initiated when a DMA slave has won the arbitration bus and the DMA controller has been programmed to service the winning request in process. DMA transfers can be in the form of a single transfer, multiple transfer (burst), or read verification. During read verification, one byte is read from memory and no transfer to I/O occurs. Burst transfers require a minimum of [300 + (500)N] nanoseconds where N is the number of byte or word transfers during the burst. Burst transfers to or from system board memory require a minimum of [300 + (600)N] nanoseconds. Single transfers require a minimum of 800 nanoseconds to transfer either a byte or a word (single transfers to or from system board memory require a minimum of 900 nanoseconds). The DMA controller used in other systems in this product line may require additional time.

Deactivation of CD CHRDY by a device can extend these accesses for slower I/O or memory devices.

The DMA controller supports the following:

- Register/Program compatibility with the IBM Personal Computer AT® DMA channels (8237 Compatible Mode)
- Control status for the bus control logic
- 16M-byte (24-bit) address capability for memory and 64K-byte (16-bit) address capability for I/O
- Eight independent DMA channels capable of transferring data between memory and I/O devices
- Serial DMA operation with a separate read and write cycle for each transfer operation
- Each channel programmable to byte or word transfer
- Sharing of the system bus interface and control logic
- Extended Operations
  - Extended program control
  - Extended Mode register.

DMA Controller Operations

The DMA Controller does three types of operations. They are:

- Data transfers between memory and I/O devices
- Read verification operations
- Memory refresh cycles.

Data Transfers Between Memory and I/O Devices

The DMA controller performs serial transfers with a minimum of three 100-nanosecond clock cycles for I/O read and write operations and a minimum of two 100-nanosecond clock cycles for memory read and write operations (read and write operations to system board memory require three clock cycles). These transfers can be between memory and I/O on any channel. Data is read from a device and latched in the DMA controller before it is written back to a second device. The memory address (target address) needs to be specified only for the DMA data transfer. If the programmable 16-bit I/O address is not selected, the I/O address is forced to hex 0000 during the I/O transfer. A programmable 16-bit I/O address can be provided during the I/O portion of the transfer as a programmable option.

Personal Computer AT is a registered trademark of the International Business Machines Corporation.

Read Verification Operations

The DMA controller can do a memory read operation without a transfer. The address and the count are updated and terminal count provided.

DMA Controller Bus Cycle States

The DMA controller has three basic bus states:

- Idle (Ti)
- Status (Ts)
- Command (Tc).

Idle State (Ti)

The Idle state indicates the DMA controller has no data transfer in progress. It is only during this state that the DMA controller can be programmed. The DMA controller enters this state upon detecting a reset (hardware or software), or upon completing the last DMA transfer for the current active device.

Status State (Ts)

The beginning of the active state is signaled by -S0 or -S1 being driven low by the DMA controller upon activating a channel. The DMA controller sends the status to the bus controller for generation of a command. All bus operations are coded in the status signals as shown in the following figure.

M/-IO	-S0	-S1	DMA Operation
0	0	0	Reserved A
0	0	1	IO Write
0	1	0	IO Read
0	1	1	Reserved B
1	0	0	Reserved C
1	0	1	Memory Write
1	1	0	Memory Read
1	1	1	Reserved D

Figure 3-2. -S0, -S1 Bus States

Command State (Tc)

The second active state is Tc. Memory or I/O devices respond during this period to either a Read or a Write command. Tc states are repeated as long as the 'channel ready' signal (CD CHRDY) is driven inactive to allow time for slow devices to respond.

Byte Pointer

A byte pointer allows 8-bit ports to access consecutive bytes of registers greater than 8 bits. These registers are the Memory Address registers (3 bytes), the Transfer Count registers (2 bytes), and the I/O Address registers (2 bytes).

Note: Developers of adapters or application programs for the IBM Personal System/2 Models 50 and 60 that use DMA channels should include an instruction sequence similar to the following before activating a DMA channel:

```
MOV   DX,31A8H
MOV   AL,ARB_LVL_NO  ; ARBITRATION LEVEL NUMBER
OUT   DX,AL
JMP   $+2
MOV   DX,31A6H
IN    AL,DX
JMP   $+2
MOV   DX,31A9H
OUT   DX,AL
JMP   $+2
```

I/O instructions to addresses hex 31A6, 31A8, and 31A9 must be byte transfers.

The purpose of this code is to allow operation with the IBM Personal System/2 80286 Expanded Memory Adapter/A when it is used in conjunction with the IBM 3270 Workstation Program.

DMA Registers

All system microprocessor access to the DMA must be 8-bit I/O instructions. The following figure lists the name and size of the DMA registers:

Register	Size (bits)	Quantity of Registers	Allocation
Memory Address	24	8	1 per Channel
I/O Address	16	8	1 per Channel
Transfer Count	16	8	1 per Channel
Temporary Holding	16	1	All Channels
Mask	4	2	1 for Channels 7 - 4 1 for Channels 3 - 0
Arbus	4	2	1 for Channel 4 1 for Channel 0
Mode	8	8	1 per Channel
Status	8	2	1 for Channel 7 - 4 1 for Channel 3 - 0
Function	8	1	All Channels
Refresh	9	1	Independent of DMA

Figure 3-4. DMA Registers

Memory Address Register

Each channel has a 24-bit Memory Address register. The register is loaded by the system microprocessor. The address is incremented or decremented as specified by the Mode register. This register can be read by the system microprocessor in successive I/O byte operations. To read this register, the microprocessor can use the 8237 commands or the extended DMA commands.

I/O Address Register

Each channel has a 16-bit I/O Address register. The register is loaded by the system microprocessor. The bits in this register do not change during DMA transfers. This register can be read by the system microprocessor in successive I/O byte operations. To read this register, the system microprocessor can use only the extended DMA commands.

DMA I/O Address Map

The following figure shows the I/O addresses decoded in the DMA controller.

Address (hex)	Description	Bit Description	Byte Pointer
0000	Channel 0, Memory Address Register	00-15	Yes
0001	Channel 0, Transfer Count Register	00-15	Yes
0002	Channel 1, Memory Address Register	00-15	Yes
0003	Channel 1, Transfer Count Register	00-15	Yes
0004	Channel 2, Memory Address Register	00-15	Yes
0005	Channel 2, Transfer Count Register	00-15	Yes
0006	Channel 3, Memory Address Register	00-15	Yes
0007	Channel 3, Transfer Count Register	00-15	Yes
0008	Channel 0-3, Status Register	00-07	
000A	Channel 0-3, Mask Register (Set/Reset)	00-02	
000B	Channel 0-3, Mode Register (Write)	00-07	
000C	Clear Byte Pointer (Write)	NA	
000D	Master Clear (Write)	NA	
000E	Channel 0-3, Clear Mask Register (Write)	NA	Yes
000F	Channel 0-3, Write Mask Register	00-03	Yes
0018	Extended Function Register (Write)	00-07	
001A	Extended Function Execute	00-07	Yes *
0081	Channel 2, Page Table Address Register **	00-07	
0082	Channel 3, Page Table Address Register **	00-07	
0083	Channel 1, Page Table Address Register **	00-07	
0087	Channel 0, Page Table Address Register **	00-07	
0089	Channel 6, Page Table Address Register **	00-07	
008A	Channel 7, Page Table Address Register **	00-07	
008B	Channel 5, Page Table Address Register **	00-07	
008F	Channel 4, Page Table Address Register **	00-07	
00C0	Channel 4, Memory Address Register	00-15	Yes
00C2	Channel 4, Transfer Count Register	00-15	Yes
00C4	Channel 5, Memory Address Register	00-15	Yes
00C6	Channel 5, Transfer Count Register	00-15	Yes
00C8	Channel 6, Memory Address Register	00-15	Yes
00CA	Channel 6, Transfer Count Register	00-15	Yes
00CC	Channel 7, Memory Address Register	00-15	Yes
00CE	Channel 7, Transfer Count Register	00-15	Yes
00D0	Channel 4-7, Status Register	00-07	
00D4	Channel 4-7, Mask Register (Set/Reset)	00-02	
00D6	Channel 4-7, Mode Register (Write)	00-07	
00D8	Clear Byte Pointer (Write)	NA	
00DA	Master Clear (Write)	NA	
00DC	Channel 4-7, Clear Mask Register (Write)	NA	
00DE	Channel 4-7, Write Mask Register	00-03	

* Dependent upon the function used.
** Upper byte of Memory Address register.

Figure 3-3. DMA I/O Address Map

Transfer Count Register

Each channel has a 16-bit Transfer Count register. The register is loaded by the system microprocessor. The transfer count determines the number of transfers to be executed by the DMA channel before reaching terminal count (TC). The number of transfers is always 1 more than the count specifies. If the count is 0, the DMA does one transfer. The contents of this register does not change during DMA transfers. When the value in the register goes from hex 0000 to FFFF a terminal count pulse is generated by the DMA. This register can be read by the system microprocessor in successive I/O bytes using the 8237 commands or the extended DMA commands.

Temporary Holding Register

This 16-bit register holds the intermediate value for the serial DMA transfer taking place. A serial DMA requires the data to be held in the register before it is written back. This register is not accessible by the system microprocessor.

Mask Register

Bit	Function
7 - 3	Reserved = 0
2	0 Clear Mask Bit 1 Set Mask Bit
1, 0	00 Select Channel 0 or 4 01 Select Channel 1 or 5 10 Select Channel 2 or 6 11 Select Channel 3 or 7

Figure 3-5. Set/Clear Single Mask Bit Using 8237 Compatible Mode

Bit	Function
7 - 4	Reserved = 0
3	0 Unmask Channel 3 or 7 Mask Bit 1 Set Channel 3 or 7 Mask Bit
2	0 Unmask Channel 2 or 6 Mask Bit 1 Set Channel 2 or 6 Mask Bit
1	0 Unmask Channel 1 or 5 Mask Bit 1 Set Channel 1 or 5 Mask Bit
0	0 Unmask Channel 0 or 4 Mask Bit 1 Set Channel 0 or 4 Mask Bit

Figure 3-6. DMA Mask Register Write Using 8237 Compatible Mode

Each channel has a corresponding mask bit that, when set, disables the DMA from servicing the requesting device. Each mask bit can be set or cleared by the system microprocessor. A system reset or DMA master clear sets all mask bits to 1. A Clear Mask Register command sets all mask bits to 0.

When a device requesting DMA cycles wins the arbitration cycle, and the mask bit is set to 1 on the corresponding channel, the DMA controller does not execute any cycles in its behalf and allows external devices to provide the transfer. If no device responds, the bus times out and causes a nonmaskable interrupt (NMI).

Note: There is an exception to this rule. When channel 2 (diskette controller channel) gets a diskette request and the mask bit is set to 1 on this channel, the DMA controller does not arbitrate on behalf of the diskette controller but ignores the diskette request.

This register can be programmed using the 8237 compatible mode commands (used by the IBM Personal Computer AT) or the extended DMA commands.

Arbus Register

This register is used for Virtual DMA Operations.

Bit	Function
7 - 4	Reserved
3 - 0	Arbitration Level

Figure 3-7. Arbus Register

Mode Register

The Mode register for each channel identifies the type of operation that takes place when that channel is activated.

Bit	Function
7, 6	Reserved = 0
5	Reserved (Must be set to 0)
4	Reserved = 0
3, 2	00 Verify Operation 01 Write Operation 10 Read Operation 11 Reserved
1, 0	00 Select Channel 0 or 4 01 Select Channel 1 or 5 10 Select Channel 2 or 6 11 Select Channel 3 or 7

Figure 3-8. 8237 Compatible Mode Register

The Mode register is programmed by the system microprocessor and its contents reformatted and stored internally in the DMA controller. In the 8237 compatible mode, this register can only be written.

In addition to the 8237 compatible mode, all channels support an extended 8-bit Mode register. This extended mode can be programmed and read by the system microprocessor.

Status Register

The Status register, which can be read by the system microprocessor, contains information about the status of the devices. This information tells which channels have reached terminal count and which channels have requested the bus since the last time the register was read.

Bit	Function
7	Channel 3 or 7 Request
6	Channel 2 or 6 Request
5	Channel 1 or 5 Request
4	Channel 0 or 4 Request
3	TC on Channel 3 or 7
2	TC on Channel 2 or 6
1	TC on Channel 1 or 5
0	TC on Channel 0 or 4

Figure 3-9. Status Register

Bits 3 through 0 in each Status register are set every time a terminal count is reached by a corresponding channel. Bits 7 through 4 are set every time a corresponding channel has controlled the bus. All bits are cleared by reset or following a system microprocessor Status Read command. This register can be read using the 8237 commands or extended DMA commands.

Function Register (F0)

This 8-bit register minimizes I/O address requirements and provides the extended program functions. The system microprocessor loads this register using the data bus.

DMA Extended Operations

The function register supports an extended set of commands for the DMA channels. The system microprocessor uses the following addresses to gain control of the internal DMA registers:

I/O Address (hex)	Command
0018	Function Register
0019	Reserved
001A	Execute Function Register
001B	Reserved

Figure 3-10. DMA Extended Address Decode

The system microprocessor uses the following steps to write to or read from any of the DMA internal registers:

1. Write to the Function register by executing an Out command to address hex 0018, with the proper data to indicate the function and the channel number. The internal Byte Pointer register is always reset to 0 when the Out to address hex 0018 is detected.

Bit	Function
7 - 4	Program Command
3	Reserved = 0
2 - 0	Channel Number

Figure 3-11. DMA Function Register

2. Execute the function by doing an In or Out to address hex 001A. The byte pointer automatically increments by 1 and points to the next byte every time the port address hex 001A is used. This step is not required for Direct commands since they are executed upon detection of the Out to address hex 0018.

Extended Mode Register

DMA supports an Extended Mode register for each channel that can be programmed and read by the system microprocessor. This register is activated whenever a DMA channel requests a DMA data transfer.

The DMA channel must be programmed to match the transfer size of the DMA slave on the channel. Bit 6 of this register is used to program the size of the DMA transfer.

Note: 16-bit DMA read transfers from 8-bit memory or 8-bit memory mapped I/O devices are not supported by the Model 50 and Model 60 systems.

Bit	Function
7	Reserved = 0
6	0 = 8-Bit transfer 1 = 16-Bit transfer
5	Reserved = 0
4	Reserved (Must be set to 0)
3	0 = Read Memory Transfer 1 = Write Memory Transfer
2	0 = Verify 1 = Transfer Data
1	Reserved = 0
0	0 = I/O Address equals 0000H 1 = Use programmed I/O Address

Figure 3-12. Extended Mode Register

Extended Commands

The following figure shows the available commands contained in the Function register.

Registers/Bits Accessed	Bits	Program Command (hex)	Byte Pointer
I/O Address Register	00-15	0	Yes
Reserved		1	
Memory Address Register Write	00-23	2	Yes
Memory Address Register Read	00-23	3	Yes
Transfer Count Register Write	00-15	4	Yes
Transfer Count Register Read	00-15	5	Yes
Status Register Read	00-07	6	
Mode Register	00-07	7	
Arbus Register	00-07	8	
Mask Register Set Single Bit *		9	
Mask Register Reset Single Bit *		A	
Reserved		B	
Reserved		C	
Master Clear *		D	
Reserved		E	
Reserved		F	

* Direct commands to the Function register

Figure 3-13. DMA Extended Commands

The following is an example showing the programming of channel 2 using the 8237 and the extended mode.

Program Step	8237 Compatible Mode	Extended Mode
	OUT to ADRS Data	OUT to ADRS Data
Set Channel Mask Bit	(000AH) x6H	(0018H) 92H
Clear Byte Pointer	(000CH) xxH	(0018H) 22H
Write Memory Address	(0004H) xxH	(001AH) xxH
Write Memory Address	(0004H) xxH	(001AH) xxH
Write Page Table Address	(0081H) xxH	(001AH) xxH
Clear Byte Pointer	(000CH) xxH	(0018H) 42H
Write Register Count	(0005H) xxH	(001AH) xxH
Write Register Count	(0005H) xxH	(001AH) xxH
Write Mode Register	(000BH) xxH	(0018H) 72H (001AH) xxH
Clear Channel 2 Mask Bit	(000AH) x2H	(0018H) A2H

Values inside of ( ) are addresses, x's represent data.

Figure 3-14. DMA Channel 2 Programming Example, Extended Commands

Virtual DMA Channel Operation

This feature permits programming of the arbitration level assignment for channel 0 and channel 4 using the two 4-bit Arbus registers. These registers enable the system microprocessor to dynamically reassign the arbitration ID value by which the DMA controller responds to bus arbitration for DMA requests. This allows channels 0 and 4 to service devices at any arbitration level. The value of hex F is reserved. The extended command hex 8 programs the Arbus registers. The upper 4 bits of the Function register should be set to a value of 8, and the lower 4 bits should be set to the channel number (0 or 4).

Interrupts

The system provides 16 levels of system interrupts. Any or all of the interrupts may be masked, including the non-maskable interrupt (except for the Watchdog timer). The system board uses two Intel 8259A interrupt controllers. The interrupt controllers are initialized to level sensitive mode. Hardware external to the interrupt controllers prevents initializing edge-triggered mode.

Nonmaskable Interrupt

The nonmaskable interrupt (NMI) signals the system microprocessor that a parity error, a channel check, a system channel time-out, or a system Watchdog timer time-out has occurred. The NMI stops all arbitration on the bus until bit 6 of the Arbitration register (I/O address hex 0090) is set to 0. This can result in lost data or an overrun error on some I/O devices. The NMI masks all other interrupts and the IRET restores the interrupt flag to the state it was in prior to the interrupt. A system reset causes a reset of the NMI.

Nonmaskable interrupt requests from system board parity and channel check are subject to mask control from the NMI Mask bit at I/O address hex 0070. The Watchdog timer and system channel time-out are not masked by this bit. This address is shared with the address for the RT/CMOS RAM (See "RT/CMOS RAM I/O Operations" on page 4-183). The power-on default of the NMI mask is 1 (NMI disabled). Prior to enabling the NMI after a power-on reset (write to address hex 0070 with bit 7 equal to 0) the parity check and channel check state are initialized by the POST.

Warning: When writing port hex 0070 to enable or disable an NMI, a read to port hex 0071 must be accessed immediately. Failure to do this may cause intermittent malfunctions and unreliable operation of the MC146818A Real-Time Clock/Complementary Metal Oxide Semiconductor RAM.

Interrupt Assignments

The following figure shows the interrupt assignments, interrupt levels, and their functions. The interrupt levels are listed in the figure by order of priority. The highest priority is the NMI, and the lowest is IRQ 7.

Level	Function		
NMI	Parity, Watchdog Timer, Arbitration time-out, Channel Check		
IRQ 0	Timer		
IRQ 1	Keyboard		
IRQ 2	Cascade Interrupt Control —	IRQ 8	Real Time Clock
		IRQ 9	Redirect Cascade
		IRQ 10	Reserved
		IRQ 11	Reserved
		IRQ 12	Mouse
		IRQ 13	Math Coprocessor Exception
		IRQ 14	Fixed Disk
		IRQ 15	Reserved
IRQ 3	Serial Alternate		
IRQ 4	Serial Primary		
IRQ 5	Reserved		
IRQ 6	Diskette		
IRQ 7	Parallel Port		

IRQ 8 through 15 are cascaded through IRQ 2

Figure 3-15. Interrupt Level Assignments by Priority

Hardware interrupt IRQ9 is defined as the replacement interrupt level for the cascade level IRQ2. Program interrupt sharing should be implemented on IRQ2, interrupt hex 0A. The following processing occurs to maintain compatibility with the IRQ2 used by IBM Personal Computer products:

1. A device drives the interrupt request active on IRQ2 of the channel.
2. This interrupt request is mapped in hardware to IRQ9 input on the second interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ9 (interrupt hex 71) interrupt handler.
4. This interrupt handler performs an end of interrupt (EOI) to the second interrupt controller and passes control to IRQ2 (interrupt hex 0A) interrupt handler.

5. This IRQ2 interrupt handler, when handling the interrupt, causes the device to reset the interrupt request prior to performing an EOI to the master interrupt controller that finishes servicing the IRQ2 request.

Note: Prior to programming the interrupt controllers, interrupts should be disabled by executing a CLI instruction. This includes the Mask register, EOIs, initialization control words, and operational control words.

System Timers

The system has three programmable timer/counters. They are Channel 0, Channel 2, and Channel 3. Channel 0 and Channel 2 are similar to Channel 0 and Channel 2 of the IBM Personal Computer, IBM Personal Computer XT™, and the IBM Personal Computer AT. Channel 3 does not have a counterpart in earlier IBM Personal Computer systems. The following is a block diagram of the counters.

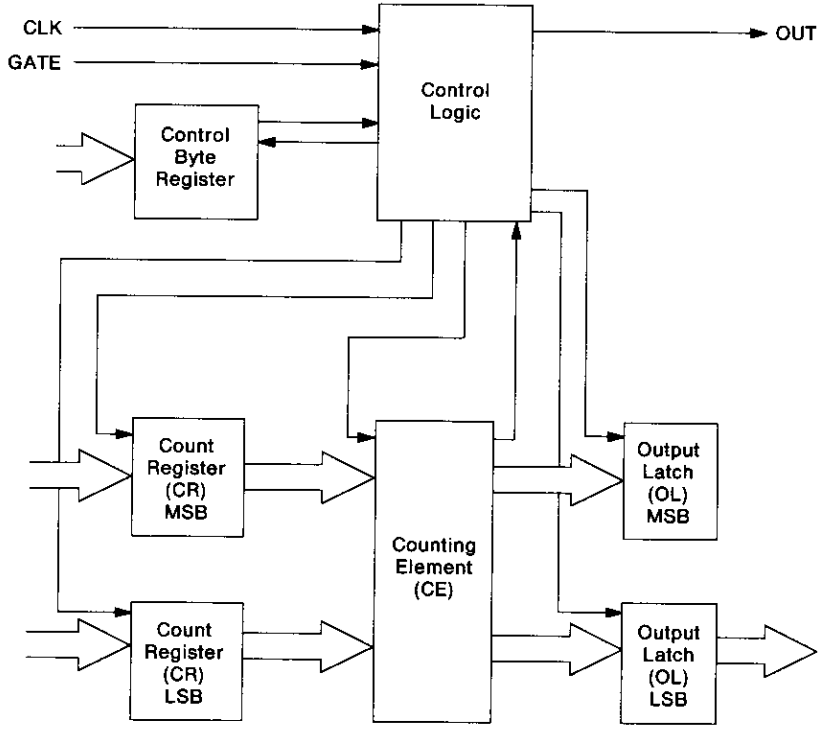

Figure 3-16. Counter Block Diagram

Personal Computer XT is a trademark of the International Business Machines Corporation.

Channel 0 - System Timer

- GATE 0 is always enabled.
- CLK IN 0 is driven by 1.190 MHz.
- CLK OUT 0 indirectly drives 8259A, IRQ 0.

  The 'interrupt request 0' signal (IRQ 0) is driven by a latch. This latch is set by a rising edge of the 'clock out 0' (CLK OUT 0) signal. The latch may be cleared by a system reset, an interrupt acknowledge cycle with a vector of hex 08, or an I/O write to port hex 0061 with bit 7 equal to 1.

  Signals derived from CLK OUT 0 are used to gate and clock Channel 3.

Channel 2 - Tone Generation for Speaker

- GATE 2 is controlled by bit 0 of port hex 0061.
- CLK IN 2 is driven by 1.190 MHz.
- CLK OUT 2 is connected to two places. One connection is to the input port hex 0061, bit 5. CLK OUT 2 is also logically ANDed with port hex 0061, bit 1. The output of the AND gate drives the 'audio sum node' signal.

Channel 3 - Watchdog Timer

This channel operates only in Mode 0, 8-bit binary.

- GATE 3 is tied to IRQ 0.
- CLK IN 3 is tied to CLK OUT 0 inverted.
- CLK OUT 3 when high drives the NMI active.

The Watchdog timer detects when IRQ 0 is active for more than one period of CLK OUT 0. If IRQ 0 is active when a rising edge of CLK OUT 0 occurs, the count is decremented. When the count is decremented to 0 an NMI is generated. Thus, the Watchdog timer may be used to detect when IRQ 0 is not being serviced. This is useful to detect error conditions.

BIOS interfaces are provided to enable and disable the Watchdog timer. When the Watchdog timer times out, it causes an NMI and sets port hex 0092, bit 4. This bit may be cleared by using the BIOS interface to disable the Watchdog timer.

Note: The NMI stops all arbitration on the bus until bit 6 of the Arbitration register (I/O address hex 0090) is set to 0. This can result in lost data or an overrun error on some I/O devices.

If the Watchdog timer is used to detect "tight looping" software tasks that inhibit interrupts, some I/O devices may be overrun (not serviced in time). The operating system may be required to restart these devices.

When the Watchdog timer is enabled, the 'inhibit' (INHIBIT) signal is active only when IRQ 0 is pending for longer than one period of CLK OUT 0. When INHIBIT is active any data written to Channel 0 or Channel 3 is ignored. INHIBIT is never active if the Watchdog timer is disabled.

The Watchdog timer operation is only defined when Channel 0 is programmed in Mode 2 or Mode 3. The operation of the Watchdog timer is undefined when Channel 0 is programmed in any other mode.

Counters 0, 2, and 3

Each counter is independent. Counters 0 and 2 are 16-bit down counters that can be preset. They can count in binary or binary coded decimal (BCD). Counter 3 is an 8-bit down counter that can be preset. It counts in binary only.

Programming the System Timers

The system treats the programmable interval timer as an arrangement of five external I/O ports. Three ports are treated as count registers and two are control registers for mode programming. Counters are programmed by writing a control word and then an initial count. All control words are written into the Control Word registers, which are located at I/O Address hex 0043 for Counters 0 and 2, and I/O address hex 0047 for Counter 3. Initial counts are written into the Count registers, not the Control Word registers. The format of the initial count is determined by the control word used.

The count is written to the Count register. It is then transferred to the counting element, according to the mode definition. When the count is read, the data is presented by the output latch.

Counter Write Operations

- The control word must be written before the initial count is written.
- The count must follow the count format specified in the control word.

A new initial count may be written to the counters at any time without affecting the counter's programmed mode. Counting is affected as described in the mode definitions. The new count must follow the programmed count format.

Counter Read Operations

The counters can be read using the Counter Latch command. For more information on the Counter Latch command, see "Counter Latch Command" on page 3-32.

If the counter is programmed for 2-byte counts, 2 bytes must be read. The 2 bytes need not be read consecutively; read, or write, or programming operations of other counters may be inserted between them.

Note: If the counters are programmed to read or write 2-byte counts, the program must not transfer control between writing the first and second byte to another routine that also reads or writes into the same counter. This will cause an incorrect count.

Registers

I/O Address (hex)	Register
0040	Read/Write Counter 0
0042	Read/Write Counter 2
0043	Write Control Byte for counter 0 or 2
0044	Read/Write Counter 3
0047	Write Control Byte for counter 3

Figure 3-17. System Timer/Counter Registers

Port Hex 0040 - Count Register - Channel 0

The control byte is written to port hex 0043 indicating the format of the count (least-significant byte only, most-significant byte only, or least-significant byte followed by most-significant byte). This must be done before writing the count to I/O port hex 0040.

Port Hex 0042 - Count Register - Channel 2

The control byte is written to port hex 0043 indicating the format of the count (least-significant byte only, most-significant byte only, or least-significant byte followed by most-significant byte). This must be done before writing the count to I/O port hex 0042.

Port Hex 0043 - Control Byte - Channel 0 or 2

This is a write-only register. The following gives the format for the control byte (I/O port hex 0043) for counters 0 and 2.

Bit 7 SC1

Bit 6 SC0

SC1	SC0	
0	0	Select Counter 0
0	1	Reserved
1	0	Select Counter 2
1	1	Reserved

Figure 3-18. SC - Select Counter, Port Hex 0043

Bit 5 RW1

Bit 4 RW0

RW1	RW0	
0	0	Counter Latch Command
0	1	Read/Write Counter Bits 0 - 7 only
1	0	Read/Write Counter Bits 8 - 15 only
1	1	Read/Write Counter Bits 0 - 7 first, then bits 8 - 15

Figure 3-19. RW - Read/Write Counter, Port Hex 0043

Bit 3 M2

Bit 2 M1

Bit 1 M0

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

Don't care bits (X) should be set to 0.

Figure 3-20. M - Counter Mode

Bit 0 BCD

0	Binary Counter 16 bits
1	Binary Coded Decimal Counter (4 Decades)

Figure 3-21. Binary Coded Decimal (BCD)

Port Hex 0044 - Count Register - Channel 3

The control byte is written to port hex 0047 indicating the format of the count (least-significant byte only). This must be done before writing the count to I/O port hex 0044.

Port Hex 0047 - Control Byte - Channel 3

This is a write-only register. The following gives the format for the control byte (I/O port hex 0047) for counter 3.

Bit 7 SC1

Bit 6 SC0

SC1	SC0	
0	0	Select Counter 3
0	1	Reserved
1	0	Reserved
1	1	Reserved

Figure 3-22. SC - Select Counter, Port Hex 0047

Bit 5 RW1

Bit 4 RW0

RW1	RW0	
0	0	Counter Latch Command Select Counter 0
0	1	R/W Counter Bits 0 - 7 only
1	0	Reserved
1	1	Reserved

Figure 3-23. RW - Read/Write Counter, Port Hex 0047

Bits 3 - 0 0

Counter Latch Command

The Counter Latch command is written to the Control Byte register. The SC0 and SC1 bits select the counter and bits D5 and D4 distinguish this command from a control byte. The following figure shows the format of the Counter Latch command.

Bit	Function
7	SC1 - Specifies the Counter to be latched
6	SC0 - Specifies the Counter to be latched
5	0 - Specifies the Counter Latch command
4	0 - Specifies the Counter Latch command
3	0 - Reserved = 0
2	0 - Reserved = 0
1	0 - Reserved = 0
0	0 - Reserved = 0

Figure 3-24. Counter Latch Command

The count is latched into the selected counter's output latch at the time the Counter Latch command is received. This count is held in the latch until it is read by the system microprocessor (or until the counter is reprogrammed). After the count is read by the system microprocessor it is automatically unlatched and the output latch (OL) returns to following the counting element (CE). Counter Latch commands do not affect the programmed mode of the counter in any way. All subsequent latch commands to a given counter, issued before the count is read, are ignored. A read cycle to the counter latch returns the value latched by the first Counter Latch command.

System Timer Modes

The following definitions are used when describing the timer modes.

CLK pulse A rising edge, then a falling edge on the counter CLK input.

Trigger A rising edge on a counter's input GATE.

Counter Load The transfer of a count from the Counter register to the counting element.

Mode 0 - Interrupt on Terminal Count

Event counting can be done using Mode 0. Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0. If GATE is equal to 1 when the control byte and initial count are written to the counter, the sequence is as follows:

1. The control byte is written to the counter, and OUT goes low.
2. The initial count is written.
3. Initial count is loaded on the next CLK pulse. The count is not decremented for this CLK pulse.

   The count is decremented until the counter reaches 0. For an initial count of N, the counter reaches 0 after N + 1 CLK pulses.
4. OUT goes high.

OUT remains high until a new count or new Mode 0 control byte is written into the counter.

If GATE equals 0 when an initial count is written to the counter, it is loaded on the next CLK pulse even though counting is not enabled. After GATE enables counting, OUT will go high N CLK pulses later.

If a new count is written to a counter while counting, it is loaded on the next CLK pulse. Counting then continues from the new count. If a 2-byte count is written to the counter the following occurs:

1. The first byte written to the counter disables the counting. OUT goes low immediately, and there is no delay for the CLK pulse.
2. When the second byte is written to the counter, the new count is loaded on the next CLK pulse. Again, OUT goes high when the counter reaches 0.

Mode 1 - Hardware Retriggerable One-Shot

The sequence for Mode 1 is as follows:

1. OUT is high.
2. On the CLK pulse following a trigger, OUT goes low, and begins the one-shot pulse.
3. When the counter reaches zero, OUT goes high.

OUT remains high until the CLK pulse after the next trigger.

The counter is armed by writing the control word and initial count to the counter. When a trigger occurs the counter is loaded. OUT goes low on the next CLK pulse, starting the one-shot pulse. For an initial count of N, a one-shot pulse is N CLK pulses long. The one-shot pulse repeats the same count of N for the next triggers. OUT remains low N CLK pulses following any trigger. GATE does not affect OUT. The current one-shot pulse is not affected by a new count written to the counter, unless the counter is retriggered. If the counter is retriggered, the new count is loaded and the one-shot pulse continues.

Note: Mode 1 is only valid on Counter 2.

Mode 2 - Rate Generator

This mode causes the counter to perform a divide-by-N function. Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0.

The sequence for Mode 2 is as follows:

1. OUT is high.
2. The initial count decrements to 1.
3. OUT goes low for one CLK pulse.
4. OUT goes high.
5. The counter reloads the initial count.
6. The process is repeated.

If GATE goes low during the OUT pulse, OUT goes high. On the next CLK pulse a trigger reloads the counter with the initial count. OUT goes low N CLK pulses after the trigger. This allows the GATE input to be used to synchronize the counter.

The counter is loaded on the next CLK pulse after a control byte and initial count are written to the counter. OUT goes low N CLK pulses after writing the initial count. This allows software synchronization of the counter.

The current counting sequence is not effected by a new count being written to the counter. If the counter receives a trigger after a new count is written and before the end of the current count, the new count is loaded on the next CLK pulse and counting continues from the new count. If the trigger is not received by the counter the new count is loaded following the current counting cycle.

Mode 3 - Square Wave

Mode 3 is similar to Mode 2 except for the duty cycle of OUT. Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0. An initial count of N results in a square wave on the 'out' signal. The period of the square wave is N CLK pulses. If OUT is low and GATE goes low, OUT goes high. On the next CLK pulse, a trigger reloads the counter with the initial count.

After writing a control byte and initial count, the counter is loaded on the next CLK pulse. This allows software synchronization of the counter.

The current counting sequence is not effected by a new count being written to the counter. If the counter receives a trigger after a new count is written, and before the end of the current count half-cycle of the square wave, the new count is loaded on the next CLK pulse, and counting continues from the new count. If the trigger is not received by the counter the new count is loaded following the current half-cycle.

The implementation of Mode 3 differs, depending on whether the count written is an odd or even number. If the count is even, OUT begins high and the following applies:

1. The initial count is loaded on the first CLK pulse.

2. The count is decremented by 2 on succeeding CLK pulses.
3. The count decrements to 0.
4. OUT changes state.
5. The counter is reloaded with the initial count.
6. The process repeats indefinitely.

If the count is odd, the following applies:

1. OUT is high.
2. The initial count minus 1 is loaded on the first CLK pulse.
3. The count is decremented by 2 on succeeding CLK pulses.
4. The count decrements to 0.
5. One CLK pulse after the count reaches 0, OUT goes low.
6. The counter is reloaded with the initial count minus 1.
7. Succeeding CLK pulses decrement the count by 2.
8. The count decrements to 0.
9. OUT goes high.
10. The counter is reloaded with the initial count minus 1.
11. The process repeats indefinitely.

Mode 3, using an odd count, causes OUT to go high for a count of (N + 1)/2 and low for a count of (N-1)/2.

Mode 3 may operate such that OUT is initially set low, when the control byte is written. For this condition Mode 3 operates as follows:

1. OUT is low.
2. The count decrements to half of the initial count.
3. OUT goes high.
4. The count decrements to 0.
5. OUT goes low.
6. The process repeats indefinitely.

This process results in a square wave with a period of N CLK pulses.

Note: If it is required that OUT be high after the control byte is written, the control byte must be written twice. This applies only to Mode 3.

Mode 4 - Software Retriggerable Strobe

Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0. Counting is triggered when an initial count is written.

The sequence for Mode 4 is as follows:

1. OUT is high.
2. The control byte and initial count are written to the counter.
3. The initial count is loaded on the next CLK pulse. The count is not decremented for this clock pulse.
4. The count is decremented to 0. For an initial count of N, the counter reaches 0 after N + 1 CLK pulses.
5. OUT goes low for one CLK pulse.
6. OUT goes high.

GATE should not go low one half CLK pulse before or after OUT goes low. If this occurs, OUT remains low until GATE transitions high.

If a new count is written to a counter while counting, it is loaded on the next CLK pulse. Counting then continues from the new count. If a 2-byte count is written, the following occurs:

1. Writing the first byte does not affect counting.
2. The new count is loaded on the next CLK pulse after writing the second byte.

The Mode 4 sequence can be retriggered by software. The period from when the new count of N is written to when OUT strobes low is (N + 1) pulses.

Mode 5 - Hardware Retriggerable Strobe

The sequence for Mode 5 is as follows:

1. OUT is high.
2. The control byte and initial count are written to the counter.
3. Counting is triggered by a rising edge of GATE.
4. The counter is loaded on the next CLK pulse after the trigger. This CLK pulse does not decrement the count.
5. The count is decremented to 0.
6. OUT goes low for one CLK pulse. This occurs (N + 1) CLK pulses after the trigger.
7. OUT goes high.

The counting sequence can be retriggered. OUT strobes low (N + 1) pulses after the trigger. GATE does not effect OUT.

The current counting sequence is not effected by a new count being written to the counter. If the counter receives a trigger after a new count is written and before the end of the current count, the new count is loaded on the next CLK pulse and counting continues from the new count.

Note: Mode 5 is valid only on Counter 2.

Operations Common to All Modes

Control bytes written to a counter cause all control logic to reset. OUT goes to a known state. This does not take a CLK pulse.

The falling edge of the CLK pulse is when new counts are loaded and counters are decremented.

Counters do not stop when they reach zero. In Modes 0, 1, 4, and 5 the counter wraps around to the highest count, and continues counting. Modes 2 and 3 are periodic; the counter reloads itself with the initial count and continues from there.

The GATE is sampled on the rising edge of the CLK pulse.

The following shows the minimum and maximum initial counts for the counters.

Mode	Min Count	Max Count
0	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
1	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
2	2	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
3	2	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
4	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
5	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)

Figure 3-25. Minimum and Maximum Initial Counts, Counters 0, 2

Counter 3 can use only Mode 0 - Interrupt on Terminal Count. The minimum initial count is 1 and the maximum is hex FF.

Power-On Password

RT/CMOS RAM has 8 bytes reserved for the password and its check character. The 8 bytes are initialized to hex 00. The microprocessor can only access these bytes during power-on self-test (POST). After POST is completed, if a password is installed, the password bytes are locked and cannot be accessed by a program. A password can be from 1 to 7 characters.

During password installation, the password (1 to 7 keyboard scan codes), is stored in the security space.

Password installation is a function of a program contained on the Reference diskette. Once the password utility has been installed, the password can be changed only during the POST. When the new password is installed, changed, or removed the password is not visible on the display.

The system unit cover can be physically locked to prevent unauthorized access to the battery. This helps prevent unauthorized battery removal and loss of password and configuration information.

Audio Subsystem

The audio subsystem is a 66 mm (2.6 in.) speaker driven by a linear amplifier. The linear amplifier input node can be driven from the following sources:

- System-timer Channel 2 can be enabled to drive the speaker using bit 1 of I/O port hex 0061 set to 1. For information about system timer Channel 2 see "System Timers" on page 3-25.
- The channel using the 'audio sum node' signal.

The following block diagram shows the audio subsystem.

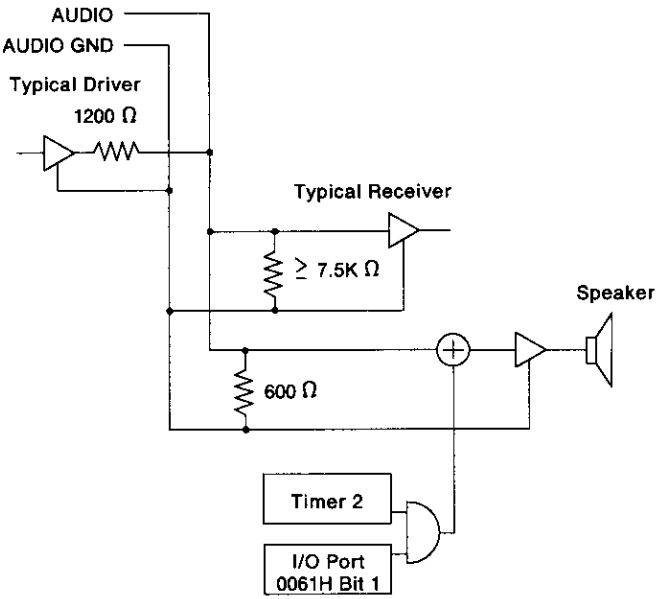

Figure 3-26. Audio Subsystem Block Diagram

Each audio driver must have a 1200 ohm source impedance, and a 7.5 kilohm or greater impedance is required for each audio receiver. Volume control is provided by the driver. Output level is a function of the number of drivers and receivers that share the AUDIO line.

Logic ground is connected to AUDIO GND at the amplifier.