ZILOG

Product Specification

Z280™ MPU Microprocessor Unit

FEATURES

- Designed in CMOS for low power operations.
- Enhanced Z80® CPU instruction set that maintains object-code compatibility with Z80 microprocessor.
- Three-stage pipelined, 16-bit CPU architecture with user and system modes.
- Direct coprocessor and multiprocessor interface support.
- On-chip paged Memory Management Unit (MMU) addresses up to 16 Mbytes.
- On-chip 256-byte instruction and data associative cache memory with burst load.
- High performance 16-bit Z-BUS® bus interface or 8-bit Z80 CPU compatible bus interface.
- Three on-chip 16-bit counter/timers.
- Four on-chip DMA channels.
- On-chip full duplex UART.
- Refresh controller for dynamic RAMs.
- **On-chip oscillator or direct input clock options.**
- 25 **MHz oscillator clock frequency.**

GENERAL DESCRIPTION

The Z280 microprocessor features a high-performance microprocessor designed to give the end-user a powerful and cost effective solution to application requirements. The Z280 microprocessor unit (MPU) incorporates advanced architectural features that allow fast and efficient throughput and increased memory addressing while maintaining Z80 object-code compatibility. The Z280 microprocessor offers a continuing growth parth for present Z80-based designs and serves as a high-performance microprocessor for new, advanced designs.

Central to the Z280 microprocessor is an enhanced version of the Z80 Central Processing Unit (CPU). To assure system integrity, the Z280 microprocessor can operate in either user or system mode, allowing protection of system resources from user tasks and programs. System mode operation is supported by the addition of the system Stack Pointer to the working register set. The IX and IY registers have been modified so that in addition to their regular function as index registers, each register can be accessed as a 16-bit general purpose register or as two byte registers. The R register, used for refresh by the Z80 CPU, is now available to the programmer as a data register in the Z280 microprocessor.

The Z80 CPU instruction set has been retained, meaning that the Z280 microprocessor is completely binary-code compatible with present Z80 code. The basic addressing modes of the Z80 microprocessor have been augmented with the addition of Indexed mode with full 16-bit displacement, Program Counter Relative with 16-bit displacement, Stack Pointer Relative with 16-bit displacement, and Base Index modes. The new addressing modes are incorporated into many of the old Z80 CPU instructions, resulting in greater flexibility and power. Some additions to the instruction set include 8-and 16-bit signed and unsigned multiply and divide, 8-and 16-bit sign extension, and a test and set instruction to support multiprocessing. The 16-bit instructions have been expanded to include 16-bit compare, memory increment, memory decrement, negate, add, and subtract, in addition to the previously mentioned multiply and divide.

A requirement of many of today's microprocessor-based system designs is to increase the memory address space beyond the 64K byte range of typical 8-bit microprocessors. The Z280 microprocessor has an on-chip Memory Management Unit (MMU) that enables addressing of up to 16M bytes of memory. In addition to enabling the address

space to be expanded, the MMU performs other memory management functions previously handled by dedicated off-chip memory management devices.

I/O address space has been expanded by the addition of an I/O Page register used to select pages of I/O addresses. The 8-bit I/O Page register can select one of 256 possible pages of I/O addresses to be active at one time, allowing a total of 64K I/O addresses to be accessed.

There are 256 bytes of on-chip memory present on the Z280 MPU. This memory can be configured as a high-speed cache or as a fixed address local memory. When configured as a cache, the memory can be programmed to be instruction only, data only, or both data and instruction. The cache memory allows programs to run significantly faster by reducing the number of external bus accesses. Operation and update of the cache is performed automatically and is completely transparent to the user. When used as a local memory, the addresses are programmable, which permits selected storage of time-critical loops in local memory.

Many features that have traditionally been handled by external peripheral devices have been incorporated in the design of the Z280 microprocessor. The "on-chip peripherals" reduce system chip count and reduce interconnection on the external bus. The Z280 MPU contains an on-chip clock oscillator and a refresh controller that provides 10-bit refresh addresses for dynamic memories. Also present are additional on-chip peripherals to provide system design flexibility. To support high-bandwidth data transmission, four Direct Memory Access (DMA) channels are incorporated on-chip. Each DMA channel operates using full 24-bit source and destination addresses with a 16-bit count. The channels can be programmed to operate in single transaction, burst, or continuous mode. System event counting and timing requirements are met with the help of the three 16-bit counter/timers. The counter/timer functions can be externally controlled with gate and trigger inputs, and can be programmed as retriggerable or nonretriggerable. A full duplex UART, capable of handling a variety of data and character formats, is present to facilitate asynchronous serial communication.

The Z280 MPU also features programmable bus timing, allowing the user to tailor timing to the individual system. Upon reset the Z280 microprocessor can be programmed for system timing that is one-fourth, one-half, or equal to the speed of the MPU's internal Central Processing Unit (CPU), with one-half being the default. In addition to clock scaling, programmable wait states can be inserted during various bus transactions. Without the use of external hardware, one to three wait states can be inserted into memory, I/O, and interrupt acknowledge transactions. Furthermore, separate memory wait states can be specified for upper and lower memory areas, facilitating the use of different speeds of ROMs and RAMs in the same system.

An additional feature of the 16-bit bus interface is the ability to support "nibble-mode" dynamic RAMs. Using this feature (known as burst mode), the bus bandwidth of memory read transactions is essentially doubled. Burst mode transactions have the further benefit of allowing the cache to operate more efficiently by guaranteeing a high probability that the contents of the accessed memory will be present in the cache.

The Z280 MPU supports Zilog's Extended Processor Architecture (EPA) in a number of ways. It is capable of trapping Extended Processor Unit (EPU) instructions in order to perform software emulation of the EPU. With its 16-bit external bus interface, the Z280 MPU directly interfaces with an EPU and operates in a manner that is completely transparent to the user and the program.

Multiprocessor system architectures are also supported by the Z280 MPU. When operating in multiprocessor mode, the Z280 MPU's Local Address register is used to distinguish between local and global memory access. Global accesses are controlled through a global request and global acknowledge protocol.

The pin functions and the pin assignments of the Z280 MPU are illustrated in Figures 1 and 2. Figure 3 shows the block diagram.

Z280 CPU

User and System Modes of Operation

The Z280 CPU can operate in either user or system mode. In user mode, some instructions cannot be executed and some registers of the CPU are inaccessible. In general, this mode of operation is intended for use by application programs. In system mode, all of the instructions can be executed and all of the CPU registers can be accessed. This mode is intended for use with programs that perform operating system functions. This separation of CPU resources promotes the integrity of the system, since programs operating in user mode cannot access those aspects of the CPU that deal with system interface events.

To further support the dual user/system mode, there are two Stack Pointers—one for the user stack and another for the system stack. These two stacks facilitate the task switching involved when interrupts or traps occur. To ensure that the user stack is free of system information, the information saved on the occurrence of interrupts or traps is always pushed onto the system stack before the new program status is loaded.

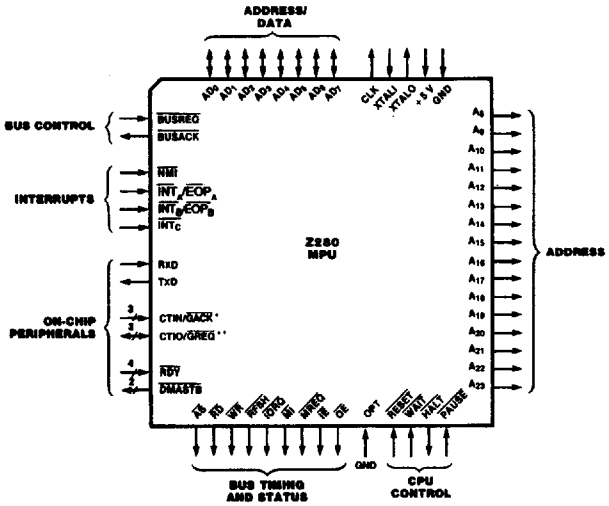


Figure 1a. Z280 Pin Functions, Z80 Bus Configuration
(Input OPT tied to GND)

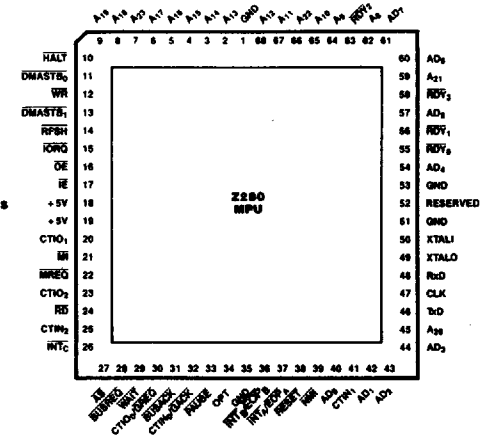


Figure 1b. Z280 Pin Assignments, Z80 Bus
(Input OPT tied to GND)

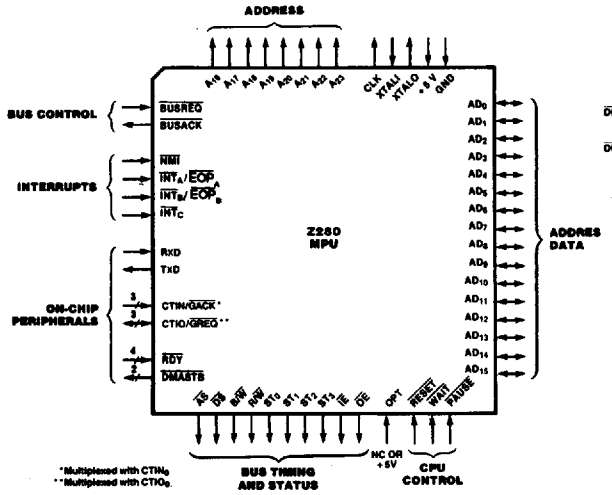


Figure 2a. Z280 Pin Functions, Z-BUS Configuration
(Input OPT tied to +5V or not connected)

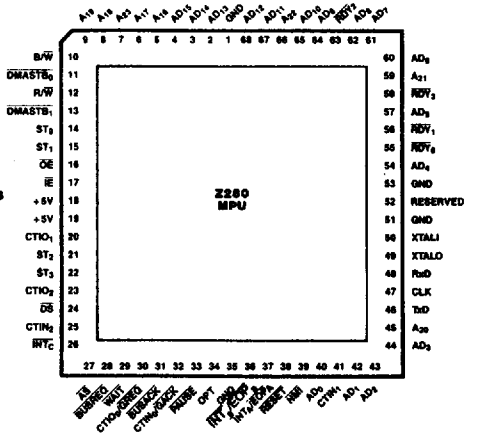


Figure 2b. Z280 Pin Assignments, Z-BUS
(Input OPT tied to +5V or not connected)

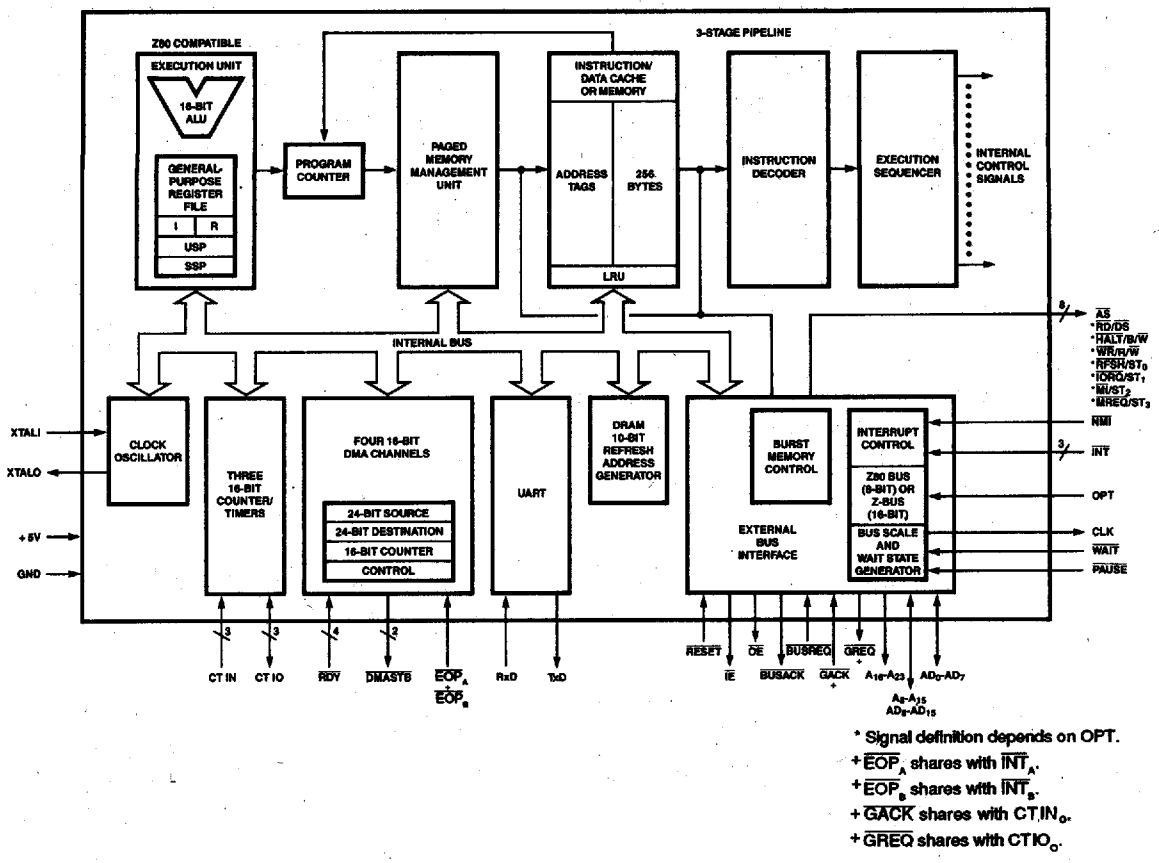


Figure 3. Z280 MPU Block Diagram.

Address Spaces

The Z280 CPU architecture supports four distinct address spaces corresponding to the different types of locations that can be accessed by the CPU. These four address spaces are:

- CPU register space
- CPU control and status register space
- Memory address space
- I/O address space

CPU Register Space. The CPU register space (Figure 4) consists of all of the registers in the CPU register file. The CPU registers are used for data and address manipulation. Access to these registers is specified in the instruction. The CPU registers are labeled A, F, B, C, D, E, H, L, A′, F′, B′, C′, D′, E′, H′, L′, IX, IY, SSP, USP, PC, I, and R.

CPU Control and Status Register Space. The CPU control register space consists of all of the control and status registers found in the CPU control register file (Figure 5). These registers govern the operation of the CPU and are accessible only by the privileged Load Control instruction. The registers in the CPU control file consist of the Bus Timing and Control register, Bus Timing and Initialization register, Local Address register, Cache Control register, Master Status register, Interrupt Status register, Interrupt/Trap Vector Table Pointer, I/O Page register, Trap Control register, and System Stack Limit register.

Memory Address Space. Two memory address spaces are supported by the Z280 CPU; one for user and one for system mode of operation. They are selected by the User/System Mode (U/S) bit in the Master Status register, which governs the selection of Page Descriptor registers during address translation.

Each address space can be viewed as a string of 64K bytes numbered consecutively in ascending order. The 8-bit byte is the basic addressable element in the memory address spaces. However, there are other addressable data elements: bits, 2-byte words, byte strings and multiple-byte EPU operands.

The address of a multiple-byte entity is the address of the byte with the lowest address. Multiple-byte entities can be stored beginning at either even or odd memory addresses.

I/O Address Space. I/O addresses are generated only by the I/O instructions (IN, OUT, and the I/O block move instructions). Logical I/O addresses are eight bits in length, augmented by the A register on lines A_8-A_{15} in Direct Address addressing mode and by the B register on lines A_8-A_{15} in Indirect Register addressing mode and for block I/O instructions. The 16-bit logical I/O address is always extended by appending the contents of the 8-bit page register to the augmented I/O address. Thus the complete address generated to address an I/O port consists of an I/O page number on A_{23}-A_{16}, the contents of the A or B register on A_8-A_{15}, and the 8-bit I/O address on A_7-A_0.

Unlike memory references, in which a 16-bit word store or fetch can generate two memory references, an I/O word store or fetch is always one I/O bus transaction, regardless of bus size or I/O port address. Note, however, that on-chip peripherals with word registers are accessed via word I/O instructions for those 16-bit registers, regardless of the external bus size (Table 1).

Data Types

The CPU can operate on bits, binary-coded decimal (BCD) digits (4 bits), bytes (8 bits), words (16 bits), byte strings, and word strings. Bits in registers or memory can be set, cleared, and tested. BCD digits, packed two to the byte, can be manipulated with the Decimal Adjust Accumulator instruction (in conjunction with binary addition and subtraction) and the Rotate Digit instructions. Bytes are operated on by 8-bit load, arithmetic, logical, and shift and rotate instructions. Words are operated on in a similar manner by the 16-bit load and 16-bit arithmetic instructions. Block move and search operations can manipulate byte strings up to 64K bytes long. Block I/O word instructions can manipulate word strings up to 32K words long. To support EPU operations, byte strings up to 16 bytes in length can be transferred by the CPU.

CPU Registers

The Z280 MPU contains 23 programmable registers (Figure 4) in the CPU register address space.

Primary and Working Register Set. The working register set is divided into the two 8-bit register files—the primary file and alternate (designated by ′) file. Each file contains an 8-bit accumulator (A), a Flag register (F), and six general-purpose registers (B, C, D, E, H, and L). Only one file can be active at any given time. Upon reset, the primary register file is active. Exchange instructions allow the programmer to exchange the active file with the inactive file.

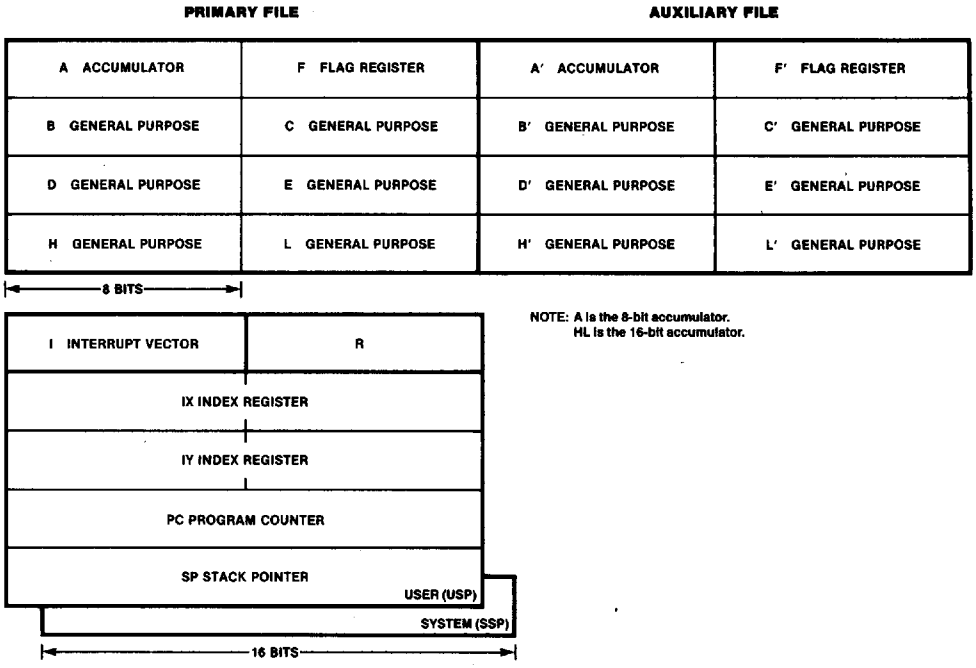
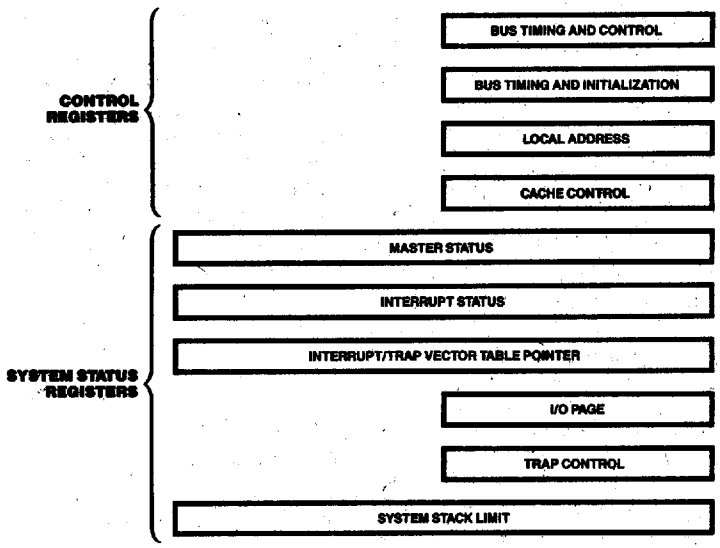


Figure 4. CPU Register Configuration



The accumulator is the destination register for 8-bit arithmetic and logical operations. The six general-purpose registers can be paired (BC, DE, and HL) to form three 16-bit general-purpose registers. The HL register pair serves as a 16-bit accumulator for 16-bit arithmetic operations.

CPU Flag Register. The Flag register contains six flags that are set or reset by various CPU operations. This register is illustrated in Figure 6.

Figure 6. CPU Flag Register

The flags in this register are:

Carry (C). This flag is set when an add instruction generates a carry or a subtract instruction generates a borrow. Certain logical and rotate and shift instructions affect the Carry flag.

Add/Subtract (N). This flag is used by the Decimal Adjust Accumulator instruction to distinguish between add and subtract operations. The flag is set for subtract operations and cleared for add operations.

Parity/Overflow (P/V). During arithmetic operations this flag is set to indicate a two's complement overflow. During logical and rotate operations, this flag is set to indicate even parity of the result or cleared to indicate odd parity.

Half Carry (H). This flag is set if an 8-bit arithmetic operation generates a carry or borrow between bits 3 and 4, or if a 16-bit operation generates a carry or borrow between bits 11 and 12. This bit is used to correct the result of a packed BCD addition or subtract operation.

Zero (Z). This flag is set if the result of an arithmetic or logical operation is a zero.

Sign (S). This flag stores the state of the most significant bit of the accumulator. The Sign flag is also used to indicate the results of a test and set instruction.

Dedicated MPU Registers

Index Registers. The two Index registers, IX and IY, each hold a 16-bit base address that is used in the Indexed addressing mode. The Index registers can also function as general-purpose registers with the upper and lower bytes capable of being accessed individually. The high and low bytes of the IX register are called IXH and IXL. The high and low bytes of the IY register are called IYH and IYL.

Interrupt Register. The Interrupt register (I) is used in interrupt mode 2 to generate a 16-bit indirect logical address to an interrupt service routine. The Interrupt register supplies the upper eight bits of the indirect address and the interrupting peripheral supplies the lower eight bits.

Program Counter. The Program Counter (PC) is used to sequence through instructions in the currently executing program and to generate relative addresses. The Program Counter contains the 16-bit logical address of the current instruction being fetched from memory.

R Register. The R register can be used as a general-purpose 8-bit read/write register. The R register is not associated with the refresh address and its contents are changed only by the user.

NOTE: To be compatible with possible future enhancements, a user should write 0's into reserved register bits. A user should not rely on values read from reserved register bits. In figures and tables, unless otherwise noted, reserved bits are labeled with "X".

Table 1. On-Chip Peripheral I/O Port Addresses

Peripheral	Address (Hexadecimal)			
Refresh Rate Register	FFxxE8			
UART				
Configuration	FExx10			
Transmitter Control/Status	FExx12			
Receiver Control/Status	FExx14			
Receiver Data	FExx16			
Transmitter Data	FExx18			
MMU				
Master Control	FFxxF0			
Page Descriptor Register Pointer	FFxxF1			
Descriptor Select Port	FFxxF5			
Block Move Port	FFxxF4			
Invalidation I/O Port	FFxxF2			
Page Descriptor Registers*				
User PDR 0	00			
User PDR 1	01			
. .	. .			
User PDR 14	0E			
User PDR 15	0F			
System PDR 0	10			
System PDR 1	11			
. .	. .			
System PDR 14	1E			
System PDR 15	1F			
DMA				
Master Control	FFxx1F			
	DMA0	**DMA1**	**DMA2**	**DMA3**
Destination Address (bits 0-11)	FFxx00	FFxx08	FFxx10	FFxx18
Destination Address (bits 12-23)	FFxx01	FFxx09	FFxx11	FFxx19
Source Address (bits 0-11)	FFxx02	FFxx0A	FFxx12	FFxx1A
Source Address (bits 12-23)	FFxx03	FFxx0B	FFxx13	FFxx1B
Count	FFxx04	FFxx0C	FFxx14	FFxx1C
Transaction Descriptor	FFxx05	FFxx0D	FFxx15	FFxx1D
Counter/Timer	**C/T0**	**C/T1**	**C/T2**	
Configuration	FExxE0	FExxE8	FExxF8	
Command/Status	FExxE1	FExxE9	FExxF9	
Time Constant	FExxE2	FExxEA	FExxFA	
Count-Time	FExxE3	FExxEB	FExxFB	

*The Page Descriptor register address must be loaded into the Page Descriptor Register Pointer in order to access that Page Descriptor register.

Stack Pointers. Two hardware Stack Pointers, the User Stack Pointer (USP) and the System Stack Pointer (SSP), support the dual mode of operation of the microprocessor. The SSP is used for saving information when an interrupt or trap occurs and for supporting subroutine calls and returns in system mode. The USP is used for supporting subroutine calls and returns in user mode.

Status and Control Registers. There are ten status and control registers available to the programmer in the Z280 MPU. Table 2 shows the addresses occupied by the registers in the status and control register addressing space.

Table 2. Status and Control Register I/O Port Addresses

Control Register Name	Address (Hexadecimal)
Bus Timing and Control	Control 02
Bus Timing and Initialization	Control FF
Cache Control[1]	Control 12
Interrupt Status	Control 16
Interrupt/Trap Vector Table	Control 06
I/O Page Register	Control 08
Local Address Register[2]	Control 14
Master Status (MSR)	Control 00
Stack Limit	Control 04
Trap Control	Control 10

NOTES:
1. See section on on-chip memory for register description.
2. See section on multiprocessing mode of operation for register description.

Bus Timing and Control Register. This 8-bit register (Figure 7) governs the timing of transactions to high memory addresses and the daisy-chain timing for interrupt requests, as well as the functionality of requests on the various Z280 MPU interrupt request lines.

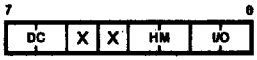


Figure 7. Bus Timing and Control Register

The fields in this register are:

I/O Wait Insertion (I/O). This 2-bit field specifies the number of additional wait states (in addition to the one automatically inserted for I/O) to be inserted by the CPU in both I/O transactions and vector response timing (00 = none, 01 = one, 10 = two, 11 = three).

High Memory Wait Insertion (HM). This 2-bit field specifies the number of automatic wait states (00 = none, 01 = one, 10 = two, 11 = three) for the CPU to insert in memory transactions when the MMU is enabled and there is a 1 in bit 15 of the selected Page Descriptor register.

Daisy Chain Timing (DC). This 2-bit field determines the number of additional automatic wait states the CPU inserts while the interrupt acknowledge daisy chain is settling (00 = none, 01 = one, 10 = two, 11 = three). A value of 01 in the DC field indicates that one additional cycle will be added to the four cycles that normally elapse between interrupt acknowledge, $\overline{AS}$ and $\overline{DS}$ (or $\overline{IORQ}$) assertions.

Bus Timing and Initialization Register. This 8-bit register (Figure 8) is used to specify the duration of control signals for the external interface bus when the MMU is disabled or when the MMU is enabled and there is a 0 in bit 15 of the selected Page Descriptor register. It also controls the relationship between internal processor clock rates and bus timing. It can be programmed by external hardware upon reset.

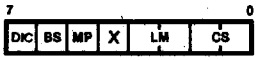


Figure 8. Bus Timing and Initialization Register

During reset this register is initialized to one of two settings, depending on the state of the $\overline{WAIT}$ input line on the rising edge of Reset: if the $\overline{WAIT}$ line is not asserted, the register is set to 00_H. If the $\overline{WAIT}$ line is asserted during reset, then this register is set to the contents of the AD lines.

The fields in this register are:

Clock Scaling (CS). This 2-bit field specifies the scaling of the CPU clock for all bus transactions (00 = one bus clock cycle is equal to two internal processor clock cycles, 01 = bus clock cycle is equal to the internal processor clock cycle, 10 = one bus clock cycle is equal to four internal processor clock cycles, 11 = reserved). This field cannot be modified by software.

Low Memory Wait Insertion (LM). This 2-bit field specifies the number of automatic wait states (00 = none, 01 = one, 10 = two, 11 = three) for the CPU to insert in memory transactions when the MMU is disabled or when the MMU is enabled and there is a 0 in bit 15 of the selected Page Descriptor register.

Multiprocessor Configuration Enable (MP). This 1-bit field enables the multiprocessor mode of operation (0 = disabled, 1 = enabled). (See the Multiprocessor Mode section.)

Bootstrap Mode Enable (BS). This 1-bit field enables the bootstrap mode of operation (0 = disabled, 1 = enabled). (See the UART section for details about bootstrap mode.)

Direct Input Clock Option (DIC). **This bit when set (0=disabled, 1=enabled) selects the direct clock source option for the XTALI input. In this mode, the crystal oscillator and divide by 2 circuits are bypassed and XTALI input is used to directly generate the MPU internal clocks. The XTALI input must have TTL levels, 50% duty cycle, and 10MHz maximum frequency. When disabled, the input frequency is divided by 2 to generate the internal processor clock. A maximum crystal or input clock frequency of 20MHz is supported in this case.**

Interrupt Status Register. This 16-bit register (Figure 9) indicates which interrupt mode is in effect and which interrupt sources have interrupt requests pending. It also contains the bits that specify whether the interrupt inputs are to be vectored. Only the interrupt vector enable bits are writeable; all other bits are read-only.

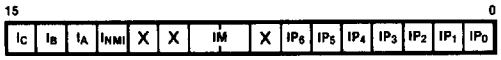


Figure 9. Interrupt Status Register

The fields in this register are:

Interrupt Request Pending (IP). When bit IP_n is set to 1, an interrupt request from sources at level n is pending. (See the Interrupt and Trap Structure section.)

Interrupt Mode (IM). A value of n in this 2-bit field indicates that interrupt mode n is in effect. This field can be changed by executing the IM instruction.

Interrupt Vector Enable (I). These four bits indicate whether each of the four interrupt inputs are to be vectored. When I_n is set to 1, interrupts on the Interrupt n line are vectored when the CPU is in interrupt mode 3; when cleared to 0, all interrupts on this line use the same entry in the Interrupt/Trap Vector Table. These bits are ignored except in interrupt mode 3.

Interrupt/Trap Vector Table Pointer. **This 16-bit register (Figure 10) contains the most significant 12 bits of the physical address at the beginning of the Interrupt/Trap Vector Table: the lower 12 bits of the physical address are assumed to be 0.**

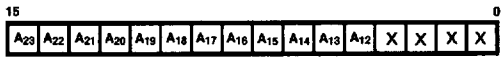


Figure 10. Interrupt/Trap Vector Table Pointer

I/O Page Register. This 8-bit register (Figure 11) indicates the bits to be appended to the 16 bits that are output during the I/O address phase of I/O transactions.

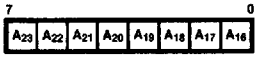


Figure 11. I/O Page Register

Master Status Register. The Master Status register (Figure 12) is a 16-bit register containing status information about the currently executing program. This register is cleared to 0 during reset.

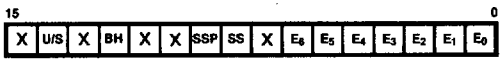


Figure 12. Master Status Register

The fields in this register are:

Interrupt Request Enable (E_n). There are seven Interrupt Enable bits, one for each type of maskable interrupt source (both external and internal). When bit E_n is set to 1, interrupt requests from sources at level n are accepted by the CPU; when this bit is cleared to 0, interrupt requests at level n are not accepted.

Single-Step (SS). While this bit is set to 1, the CPU is in single-stepping mode; while this bit is cleared to 0, automatic single-stepping is disabled. This bit is automatically cleared when a trap or interrupt is taken.

Single-Step Pending (SSP). While this bit is set to 1, the CPU generates a trap prior to executing an instruction. The SS bit is automatically copied into this field at the completion of each instruction. This bit is automatically cleared to 0 when a Single-Step, Page Fault, Privileged Instruction, Breakpoint-on-Halt or Division trap is taken so that the SSP bit in the saved Master Status register is cleared to 0.

Breakpoint-on-Halt Enable (BH). While this bit is set to 1, the CPU generates a Breakpoint trap whenever a HALT instruction is encountered; while this bit is cleared to 0, the HALT instruction is executed normally.

User/System Mode (U/S). While this bit is cleared to 0, the CPU is in the system mode of operation; while it is set to 1 the CPU is in the user mode of operation.

System Stack Limit Register. **This 16-bit register (Figure 13) indicates when a System Stack Overflow Warning trap is to be generated. If enabled, by setting a control bit in the Trap Control register, pushes onto the system stack cause the 12 most significant bits in this register to be compared to the upper 12 bits of the system Stack Pointer and a trap is generated if they match.**

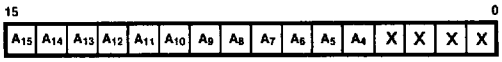


Figure 13. System Stack Limit Register

Trap Control Register. This 8-bit register (Figure 14) enables the maskable traps. Upon reset this register is initialized to all 0s.

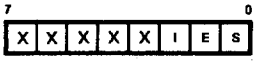


Figure 14. Trap Control Register

The bits in this register are:

System Stack Overflow Warning (S). While this bit is set to 1, the CPU generates a Stack Overflow Warning trap when the system stack enters the specified region of memory.

EPU Enable (E). While this bit is cleared to 0, the CPU generates a trap whenever an EPA instruction is encountered.

Inhibit User I/O (I). While this bit is set to 1, the CPU generates a Privileged Instruction trap when an I/O instruction is encountered in user mode.

Cache Control and Local Address Registers. See the On-Chip Memory section for information about the Cache Control register and the Multiprocessor Mode section for information about the Local Address register.

Interrupt and Trap Structure

The Z280 MPU provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions.

Interrupts. Two types of interrupt, nonmaskable and maskable, are supported by the Z280 MPU. The nonmaskable interrupt ($\overline{NMI}$) cannot be disabled (masked) by software and is generally reserved for highest priority external events that require immediate attention. Maskable interrupts, however, can be selectively disabled by software. Both nonmaskable and maskable interrupts can be programmed to be vectored or nonvectored. Interrupts are always accepted between instructions and acknowledged after execution of the prior instruction is complete. The block move, search, and I/O instructions can be safely interrupted after any iteration and restarted after the interrupt is serviced.

Interrupt Sources. The Z280 MPU accepts nonmaskable interrupts on the $\overline{NMI}$ pin only. The Z280 MPU accepts maskable interrupts on the $\overline{INT}$ pins and from the on-chip counter/timers, DMA channels, and the UART receiver and transmitter.

Interrupt Lines A, B, and C can be selectively programmed to support vectored interrupts by setting the appropriate bits in the Interrupt Status register. The external interrupts can be programmed to be vectored or nonvectored in interrupt mode 3.

Interrupt Modes of Operation. The CPU has four modes of interrupt handling. The first three modes extend the Z80 interrupt modes to accommodate additional interrupt input lines in a compatible fashion. The fourth mode provides more flexibility in handling the interrupts. On-chip peripherals use the fourth mode regardless of which mode is selected for externally generated interrupt requests. The interrupt mode is selected by using the privileged instructions IM 0, IM 1, IM 2, or IM 3. On reset, the Z280 MPU is automatically set to interrupt mode 0. The current interrupt mode in effect can be read from the Interrupt Status register.

Mode 0. This mode is identical to the 8080 interrupt response mode. With this mode, the interrupting device on any of the maskable interrupt lines can place a call or restart instruction on the data bus and the CPU will execute it. As a result, the interrupting device, instead of the memory, provides the next instruction to be executed.

Mode 1. When this mode is selected, the CPU responds to a maskable external interrupt by executing a restart to the logical address 0038_H in the system program address space.

Mode 2. This mode is a vectored interrupt response mode. With a single 8-bit byte from the interrupting device, an indirect call can be made to any memory location. With this mode the system maintains a table of 16-bit starting addresses for every interrupt service routine. This table can be located anywhere in the system mode logical data address space on a 256-byte boundary. When an interrupt is accepted, a 16-bit pointer is formed to obtain the desired interrupt service routine starting address from the table. The upper eight bits of this pointer are formed from the contents of the I register. The lower eight bits of the pointer must be supplied by the interrupting device. The 16-bit pointer so formed is treated as a logical address in the system data address space, which can be translated by the MMU to a physical address.

Mode 3. This is the intended mode of operation for systems that take advantage of the enhancements of the Z280 microprocessor (such as single-step and user/system mode) since the Master Status register is automatically saved and another loaded for the interrupts. Also, vector tables can be used for the external interrupt sources to provide more interrupt vectors for the Z8000® family, Z80 family, and Z8500 Universal Peripherals. When an interrupt request (either maskable or nonmaskable) is accepted, the Master Status register, the address of the next instruction to be executed, and a 16-bit "reason code" are pushed onto the system stack. A new Master Status register and Program Counter are then fetched from the Interrupt/Trap Vector Table. The "reason code" for externally generated interrupts is the contents of the bus during the interrupt acknowledge sequence; for 8-bit data buses, the least significant byte of the reason code is all 1's. For interrupts generated by on-chip peripherals, the reason code identifies which peripheral generated the interrupt and is identical to the vector address in the Interrupt/Trap Vector Table. The Interrupt/Trap Table Pointer is used to reference the table.

Traps. The Z280 CPU supports eight traps that are generated internally. The following traps can be disabled: the EPA trap, which allows software to emulate an EPU; the Stack Warning trap, which is taken at the end of an instruction causing the trap; the Breakpoint-on-Halt trap, which is taken when a HALT instruction is encountered; and the Single-Step trap, which is taken for each instruction. In addition, I/O instructions can be specified as privileged instructions. Traps cause the instruction to be terminated without altering CPU registers (except for the System Stack

Pointer, which is modified when the program status is pushed onto the system stack).

The saving of the program status on the system stack and the fetching of a new program status from the Interrupt/Trap Vector Table is the same in any interrupt mode of operation.

Traps can only occur if the trap generating features of the Z280 CPU (such as System Stack Overflow warning) have been explicitly enabled. Traps cannot occur on instructions of the Z80 instruction set unless explicitly enabled by the operating system using Z280 CPU extensions.

Extended Instruction. This trap occurs when the CPU encounters an extended instruction while the Extended Processing Architecture (EPA) bit in the Trap Control register is 0. Four trap vectors are used by the EPA trap—one for each type of EPA instruction. This greatly simplifies trap handlers that use I/O instructions to access an EPU or software to emulate an EPU.

Privileged Instruction. This trap occurs whenever an attempt is made to execute a privileged instruction while the CPU is in user mode (User/System Mode control bit in the Master Status register is 1).

System Call. This trap occurs whenever a System Call (SC) instruction is executed.

Access Violation. This trap occurs whenever the MMU's translation mode is enabled and an address to be translated is invalid or (for writes) is write-protected.

System Stack Overflow Warning. This trap occurs only while the Stack Overflow Warning bit in the Trap Control register is set to 1. For each system stack push operation, the most significant bits in the Stack Pointer register are compared with the contents of the Stack Limit register and a trap is signaled if they match. The Stack Overflow Warning bit is then automatically cleared in order to prevent repeated traps.

Division Exception. This trap occurs whenever the divisor is zero (divide-by-zero case) or the true quotient cannot be represented in the destination precision (overflow); the CPU flags are set to distinguish these two cases.

Single-Step. This trap occurs before executing an instruction if the Single-Step Pending control bit in the Master Status register is set to 1. Two control bits in the Master Status register are used for the Single-Step trap. The Single-Step bit (bit 8), on being set when previously clear, causes a trap to occur after the execution of the next instruction. While this bit is set to 1, if an instruction execution causes a trap, the Single-Step trap occurs after the execution of the trap-handling routine. The Single-Step Pending bit (bit 9), is used by the processor to ensure that only one Single-Step trap occurs for each instruction executed while the Single-Step bit is set to 1.

Breakpoint-on-Halt. This trap occurs whenever the Breakpoint-on-Halt control bit in the Master Status register is 1 and a HALT instruction is encountered.

Interrupt and Trap Disabling. Maskable interrupts can be enabled or disabled independently via software by setting or clearing the appropriate control bits in the Master Status register.

A 7-bit mask field in the Master Status register indicates which of the requested interrupts will be accepted. Interrupt requests are grouped as follows, with each group controlled by a separate Interrupt Enable control bit. The list is presented in order of decreasing priority, with sources within a group listed in order of descending priority.

- Maskable Interrupt A line (bit 0)
- Counter/Timer 0, DMA0 (bit 1)
- Maskable Interrupt B line (bit 2)
- Counter/Timer 1, UART receiver, DMA1 (bit 3)
- Maskable Interrupt C line (bit 4)
- UART Transmitter, DMA2 (bit 5)
- Counter/Timer 2, DMA3 (bit 6)

When a source of interrupts has been disabled, the CPU ignores any interrupt request from that source.

The System Stack Overflow Warning trap, Privileged Instruction trap (I/O instructions in user mode), or Extended Instruction trap can be enabled by setting control bits in the Trap Control register, and the Single-Step and Breakpoint-on-Halt trap can be enabled by setting control bits in the Master Status register; these are the only traps that can be disabled.

Interrupt/Trap Vector Table. The format of the Interrupt/Trap Vector Table consists of pairs of Master Status register and Program Counter words, one pair for each separate on-chip interrupt or trap source. For each external interrupt, there is a separate Master Status register word and Program Counter word (for use if the input is not vectored). If the external interrupt is vectored, a vector table consisting of one Program Counter word for each of the 128 possible vectors that can be returned for each input line is used instead of the dedicated Program Counter word; thus for vectored interrupts, there is only one Master Status register for each interrupt type.

The format of the Interrupt/Trap Vector Table is shown in Table 3.

Table 3. Interrupt/Trap Vector Table

Address (Hexadecimal)	Contents
00	Reserved
04	NMI Vector
08	Interrupt Line A Vector
0C	Interrupt Line B Vector
10	Interrupt Line C Vector
14	**Counter/Timer 0 Vector**
18	**Counter/Timer 1 Vector**
1C	**Reserved**
20	**Counter/Timer 2 Vector**
24	DMA0 Vector
28	DMA1 Vector
2C	DMA2 Vector
30	DMA3 Vector
34	UART Receiver Vector
38	UART Transmitter Vector
3C	Single-Step Trap Vector
40	Breakpoint-on-Halt Trap Vector
44	Division Exception Trap Vector
48	Stack Overflow Warning Trap Vector
4C	Page Fault Trap Vector
50	System Call Trap Vector
54	Privileged Instruction Trap Vector
58	EPU ← Memory Trap Vector
5C	Memory ← EPU Trap Vector
60	A ← EPU Trap Vector
64	EPU Internal Operation Trap Vector
68-6C	Reserved
70-16E	**128 Program Counter Values for NMI and Interrupt Line A Vectors (MSR from 04 and 08, respectively)**
170-26E	**128 Program Counter Values for Interrupt Line B Vectors(MSR from 0C)**
270-36E	**128 Program Counter Values for Interrupt Line C Vectors(MSR from 10)**

Addressing Modes

Addressing modes (Figure 15) are used by the CPU to calculate the effective address of an operand needed for execution of an instruction. Nine addressing modes are supported by the Z280 CPU. Of these nine, four are additions to the Z80 addressing modes (Indexed with 16-bit displacement, Stack Pointer Relative, Program Counter Relative, and Base Index) and the remaining five modes are either existing or extensions to the existing Z80 addressing modes.

Register. The operand is one of the 8-bit registers (A, B, C, D, E, H, L, IXH, IXL, IYH or IYL); or one of the 16-bit registers (BC, DE, HL, IX, IY, or SP), or one of the special byte registers (I or R).

Immediate. The operand is in the instruction itself and has no effective address.

Indirect Register. The contents of a register specify the effective address of an operand. The HL register is the register used for memory accesses. (For the Load To or Load From Accumulator instruction, BC and DE can also be used for indirection; for the JP instruction, IX and IY can also be used for indirection.) The C register is used for I/O and control register space accesses.

Direct Address. The effective address of the operand is the location whose address is contained in the instruction. Depending on the instruction, the specified operand is either in the I/O or data memory address space.

Indexed. The effective address of the operand is the location specified by adding the 16-bit address contained in the instruction to a two's complement "index" contained in the HL, IX, or IY register.

Short Index. The effective address of the operand is the location computed by adding the 8-bit two's complement signed displacement contained in the instruction to the contents of the IX or IY register. This addressing mode is equivalent to the Z80 CPU indexed mode.

Program Counter Relative. An 8- or 16-bit displacement contained in the instruction is added to the Program Counter to generate the effective address of the operand.

Stack Pointer Relative. The effective address of the operand is the location computed by adding a 16-bit two's complement displacement contained in the instruction to the contents of the Stack Pointer.

Base Index. The effective address of the operand is the location whose address is computed by adding the contents of HL, IX, or IY to the contents of another of these three registers.

EXTENDED PROCESSING ARCHITECTURE

Features

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z280 CPU. Features of the EPA include:

- Allows Z280 CPU instruction set to be extended by external devices.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Permits modular design of Z280 CPU-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Direct interconnection between EPUs and Z280 MPU requires no additional external supporting logic.
- EPUs can be added as the system grows and as EPUs with specialized functions are developed.

General Description

The processing power of the Zilog Z-BUS Z280 microprocessor can be boosted beyond its intrinsic capability by the Extended Processing Architecture (EPA). The EPA allows the Z280 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream.

The EPUs connect directly to the Z-BUS and continuously monitor the CPU instruction stream for an instruction intended for the EPU (template). When a template is detected, the appropriate EPU responds, obtaining or placing data or status information on the Z-BUS by using the Z280 CPU-generated control signals and performing its function as directed.

The CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes templates intended for it and executes them, using data supplied with the template and/or data within its internal registers. There are three classes of EPU instructions:

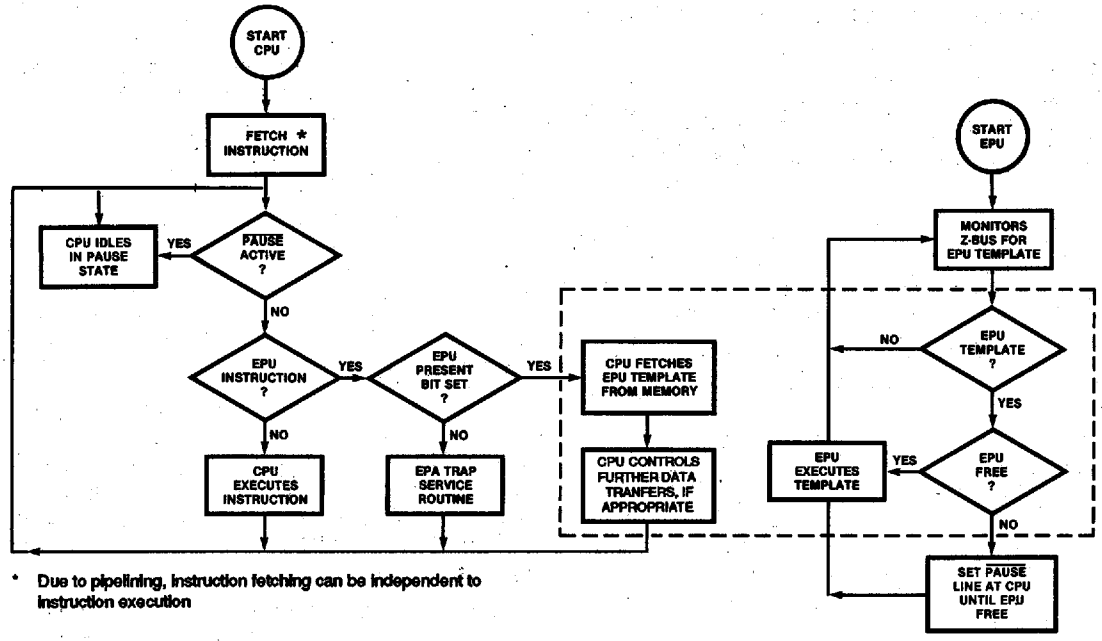
- Data transfers between main memory and EPU registers
- **Data transfers between CPU registers and EPU status registers**
- EPU internal operations

Six addressing modes can be utilized with transfers between EPU registers and the MPU and main memory:

- Indirect Register
- Direct Address
- Indexed
- Program Counter Relative
- Stack Pointer Relative
- Base Index

In addition to the hardware-implemented capabilities of the EPA, there is an extended instruction trap mechanism to permit software simulation of EPU functions. An EPU present bit in the Z280 MPU Trap Control register indicates **whether actual EPUs are present or not. If not, the CPU generates a trap when an extended instruction is detected, and a software "trap handler" can emulate the desired EPU function. Thus, the EPA software trap routine supports systems not containing an EPU.**

EPA and CPU instruction execution are shown in Figure 16. If an instruction has been fetched and decoded, the CPU determines whether or not it is an EPU instruction. If the instruction is an EPU instruction, the state of the EPU Enable bit in the Trap Control register is examined. If the EPU Enable bit is reset (E = 0), the CPU generates a trap and the EPU instruction can be simulated by an EPU instruction trap software routine. However, if the EPU Enable bit is set (E = 1), indicating that an EPU is present in the system, then the 4-byte EPU template is fetched from memory. The fetching of the EPU template is indicated by the status lines ST_0-ST_3. Each EPU continuously monitors the Z-BUS and the status lines for its own templates. After fetching the EPU template, the CPU, if necessary, transfers appropriate data between the EPU and memory or between the CPU and the EPU. These transactions are indicated by unique encodings of the status lines. If the EPU is free when the template and the data appear, the EPU template is executed. If the EPU is still processing a previous instruction, the $\overline{PAUSE}$ line can be activated to halt further execution of CPU instructions until EPU execution is complete. After the execution of the template is complete, the EPU deactivates the $\overline{PAUSE}$ line and CPU instruction execution continues.



* Due to pipelining, instruction fetching can be independent to instruction execution

Figure 16. EPA and Z280 MPU Instruction Execution.

MEMORY MANAGEMENT

Features

- On-chip dynamic address translation
- Permits addressing of 16M bytes of physical memory
- Separate translation facilities for user and system modes
- Permits instructions and data to reside in separate memory areas.
- Write protection for individual pages of memory.
- Aborts CPU on access violation to support virtual memory

General Description

The Z280 microprocessor contains an on-chip Memory Management Unit (MMU), which translates logical addresses into physical addresses. This allows access to more than 64K bytes of physical memory and provides memory protection features typical of those found on large systems. With the MMU, the CPU can access up to 16M bytes of physical memory. The MMU features a sophisticated trapping mechanism that generates page faults on error conditions. Instructions that are aborted by a page fault can be restarted in a manner compatible with virtual memory system requirements. On reset, the MMU features are not enabled, thus permitting logical addresses to pass to the physical memory untranslated.

The physical address space is expanded by dividing the 64K byte logical address space (the space manipulated by the program) into pages. The pages are then mapped (translated) into the larger physical address space of the Z280 microprocessor. The mapping process makes the user software addresses independent of the physical memory, so the user is freed from specifying where information is actually stored in physical memory. The actual size of the page depends on whether the program/data separation mode is enabled—if it is enabled, each page is 8K bytes in length, and if it is not enabled, the page length is 4K bytes. With the page mapping technique, 16-bit logical addresses can be translated into 24-bit physical addresses. Address translation can occur both in system and in user mode, with separate translation facilities available to each mode. The MMU further allows instruction references to be separated from data references, which enables programs of up to 64K bytes in length to manipulate up to 64K bytes of data without operating system intervention.

MMU Architecture

The Z280 MMU consists of two sets of sixteen Page Descriptor registers (Figure 17) that are used to translate addresses, a 16-bit control register that governs the translation facilities, a Page Descriptor Register Pointer, an I/O write-only port that can be used to invalidate sets of page descriptors, and two I/O ports for accesses to the Page Descriptor registers. One set of Page Descriptor registers is dedicated to the system mode of operation and the other set is dedicated to the user mode of operation.

While an address is being translated, attributes associated with the logical page containing that location are checked. The correct logical page is determined by the CPU mode (user or system), address space (program/data), and the four most significant bits of the logical address. Pages can be write-protected to prevent them from being modified by the executing task and can also be marked as non-cacheable to prevent information from being copied into the cache for later reference. The latter capability is useful in multiprocessor systems, to ensure that the processor always accesses the most current version of information being shared among multiple devices. The MMU also maintains a bit for each page that indicates if the page has been modified.

Each Page Descriptor register contains a Valid bit, which indicates that the descriptor contains valid information. Any attempt by the MMU to translate an address using an invalid descriptor generates a page fault. Valid bits for groups of Page Descriptor registers can be reset by writing to an MMU control port.

Figure 17. Page Descriptor Register

For each mode of CPU operation, the MMU can be configured to separate instruction fetches from data fetches, and thus separate the program address space from the data address space. When the program/data separation mode is in effect, the sixteen Page Descriptor registers for the current CPU mode of operation (user or system) are partitioned into two sets, one for instruction fetches and one for data fetches. An instruction fetch or data access using the Program Counter Relative addressing mode is translated by the MMU registers associated with the program address space; data accesses using other addressing modes and accesses to the Interrupt Vector Table in interrupt mode 2 use the MMU registers associated with the data address space. In this mode of MMU operation, the page size is 8192 bytes. There are two control bits in the MMU Master Control register that independently specify whether the user and system modes of MPU operation have separate program and data address spaces.

Each 16-bit Page Descriptor register consists of a 4-bit attribute field and a 12-bit page frame address field. The attribute field consists of the least significant bits of the descriptor and contains four control and status bits, listed below.

Modified (M). This bit is automatically set whenever a write is successfully performed to a logical address in this page; it can be cleared to 0 only by a software routine that loads the Page Descriptor register. If the Valid bit is 0, the contents of this bit are undefined.

Cacheable (C). While this bit is set to 1, information fetched from this page can be placed in the cache. While this bit is cleared to 0, the cache control mechanism is inhibited from retaining a copy of the information.

Write-Protect (WP). While this bit is set to 1, CPU writes to logical addresses in this page cause a page fault to be generated and prevent a write operation from occurring. While this bit is cleared to 0, all valid accesses are permitted.

Valid (V). While this bit is set to 1, the descriptor contains valid information. While this bit is cleared to 0, all CPU accesses to logical addresses in this page cause a page fault to be generated.

MMU Control Registers and I/O Ports

MMU operation is controlled by one control register and four dedicated I/O ports. The MMU Master Control register (Figure 18) determines the program/data address space separation in effect in both user and system modes and whether logical addresses generated in user and system mode will be translated by the MMU. Page Descriptor registers are accessed indirectly through the register address contained in the Page Descriptor Register Pointer. The Descriptor Select Port is used to access the Page Descriptor register that is pointed to by the Page Descriptor Register Pointer. After this access the Page Descriptor Register Pointer is left unchanged. The Block Move I/O Port is used to move blocks of words between the Page Descriptor registers and memory; reads or writes to this I/O port access data pointed to by the Page Descriptor Register Pointer, then increment the pointer by one. The Invalidation I/O Port is used to invalidate blocks of Page Descriptor registers; writes to this port cause the Valid bits in selected blocks of Page Descriptor registers to be cleared to 0, which indicates that the descriptors no longer contain valid information.

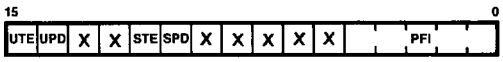


Figure 18. MMU Master Control Register

MMU Master Control Register. The MMU Master Control register (I/O address location FFxxF0) controls the operation of the MMU. This register contains four control bits; all other bits in this register must be cleared to 0. The four control bits of the MMU Master Control register are described below.

Page Fault Identifier (PFI). This 5-bit field latches information that indicates which Page Descriptor register was being accessed when the access violation was detected.

System Mode Program/Data Separation Enable (SPD). While this bit is set to 1, instruction fetches and data accesses via the PC Relative addressing mode use the system mode Page Descriptor registers 8-15, and data references that do not use the PC Relative addressing mode use the system mode Page Descriptor registers 0-7. While this bit is cleared to 0, system mode Page Descriptor registers 0-15 are used to translate instruction and data references.

System Mode Translate Enable (STE). While this bit is set to 1, logical addresses generated in the system mode of operation are translated. While this bit is cleared to 0, addresses are passed through the MMU extended with zeros in the most significant bits and no attribute checking or modified bit setting is performed.

User Mode Program/Data Space Separation Enable (UPD). While this bit is set to 1, instruction fetches and data accesses via the PC Relative addressing mode use the user mode Page Descriptor registers 8-15, and data references that do not use the PC Relative addressing mode use the user mode Page Descriptor registers 0-7. While this bit is cleared to 0, user mode Page Descriptor registers 0-15 are used to translate instruction and data references.

User Mode Translated Enable (UTE). While this bit is set to 1, logical addresses generated in the user mode of operation are translated. While this bit is cleared to 0, addresses are passed through the MMU extended with zeros in the most significant bits and no attribute checking or modified bit setting is performed.

Page Descriptor Register Pointer. Moves of data into and out of the MMU Page Descriptor registers use the Page Descriptor Register Pointer, which is at I/O address location FFxxF1. This 8-bit register contains the address of one of the Page Descriptor registers. When a word I/O instruction accesses I/O address FFxxF5 (Descriptor Select Port), this register is used to access a Page Descriptor register. When a word I/O instruction accesses I/O address FFxxF4 (Block Move I/O Port), this register is also used to access a Page Descriptor register, but after the access, this register is automatically incremented by one.

Descriptor Select Port. Moves of one word of data into and out of a Page Descriptor register are accomplished by writing and reading words to or from this dedicated I/O port at location FFxxF5. Any word I/O instruction can be used to access a Page Descriptor register via this port, provided that the Page Descriptor Register Pointer is properly initialized.

Block Move I/O Port. Block moves of data into and out of the Page Descriptor registers are accomplished by writing and reading words to or from this dedicated I/O port at location FFxxF4. Any word I/O instruction can be used to access Page Descriptor registers via this port, provided that the Page Descriptor Register Pointer is properly initialized.

Invalidation I/O Port. Valid bits can be cleared (i.e., the Page Descriptor registers invalidated) by writing to this dedicated 8-bit port (Table 4), which is at I/O address location FFxxF2. Individual Valid bits can subsequently be set by software writing to the Page Descriptor registers. Reading this I/O port returns unpredictable data.

Table 4. Invalidation Port Table

Encoding	Registers Invalid
01_H	System Page Descriptor Registers 0-7
02_H	System Page Descriptor Registers 8-15
03_H	System Page Descriptor Registers 0-15
04_H	User Page Descriptor Registers 0-7
08_H	User Page Descriptor Registers 8-15
$0C_H$	User Page Descriptor Registers 0-15

Translation Mechanism

Address Translation: Address translation is illustrated in Figure 19. While the Program/Data Space Separation bit is cleared to 0, the 16-bit logical address is divided into two fields, a 4-bit index field used to select one of 16 Page Descriptor registers and a 12-bit offset field that forms the lower 12 bits of the physical address. The physical address is composed of the 12-bit page frame address (bits 4-15) supplied by the selected Page Descriptor register and the 12-bit offset supplied by the logical address.

While the Program/Data Space Separation bit is set to 1, the logical address is divided into a 3-bit index field and a 13-bit offset field. The Page Descriptor register consists of an 11-bit Page Frame Address field (bits 5-15, with bit 4 = 0). The physical address is a result of concatenating the page frame address and the logical offset. The Page Descriptor register is chosen by a 4-bit index field, which consists of a Program/Data Address bit from the CPU and the three Index bits from the logical address.

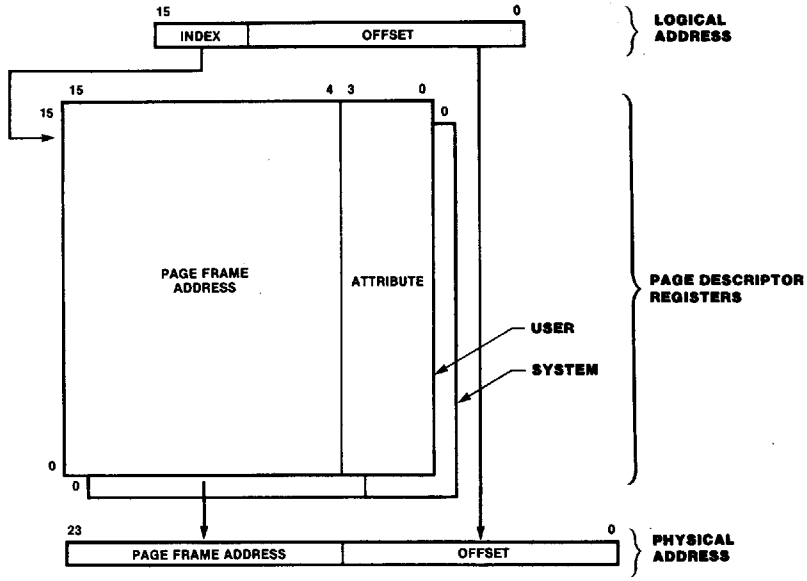


Figure 19. Address Translation

ON-CHIP MEMORY

Features

- 256-byte local memory
- Configurable as high-speed associative cache
- Programmable to cache instructions, data, or both
- Permits faster execution by minimizing external bus accesses
- Operation is transparent to user
- Configurable as local RAM with user-definable addresses

The Z280 MPU has 256 bytes of on-chip memory, which can be dedicated to memory locations programmed by the system or used as a cache for instructions or data. Its mode of use (dedicated memory or cache) is programmable; on reset it is automatically enabled for use as a cache for instructions only.

On-Chip Memory Architecture

The on-chip memory is organized as 16 lines of 16 bytes each. Each line can hold a copy of 16 consecutive bytes in physical memory locations whose 20 most significant bits of physical address are identical. Each byte in the cache has an associated Valid bit that indicates whether the cache holds a valid copy of the memory contents at the associated physical memory location. Figure 20 illustrates the cache organization.

	20 BITS	16 BITS	16 x 8 BITS
LINE 0	TAG 0	VALID BITS	CACHE DATA
LINE 1	TAG 1	VALID BITS	CACHE DATA
LINE 2	TAG 2	VALID BITS	CACHE DATA
•	•	•	•
•	•	•	•
•	•	•	•
LINE 15	TAG 15	VALID BITS	CACHE DATA

Tag n = the 20 Address bits associated with line n
Valid bits = 16 bits that indicate which bytes in the cache line contain valid data
Cache data = 16 bytes

Figure 20. Cache Organization

The on-chip memory has two modes of operation. If the Memory/Cache bit in the Cache Control register is set to 1, then the 256 bytes of on-chip memory are treated as physical memory locations. Memory accesses to addresses covered by the on-chip memory do not generate bus transactions on the external bus and hence the accesses are faster. In this mode, the Valid bits are ignored.

If the Memory/Cache bit is cleared to 0, then the 256 bytes of on-chip memory are treated as a cache memory. The lines are allocated using a least-recently used (LRU) algorithm. When a cache "miss" on a read occurs (and the MMU does not assert cache inhibit), the line in the cache that has been least recently accessed is selected to hold the newly read data. All bytes in the selected line are marked invalid except for the bytes containing the newly accessed data. On a cache miss, one or two bytes, depending on the bus size, are fetched from main memory. Except for burst mode instruction fetches, the cache does not pre-fetch beyond the currently-requested address. A cache miss on a data write does not cause a line to be allocated to the memory location accessed.

The cache can hold both instructions and data. Two control bits in the Cache Control register can be separately set to enable the cache to hold instructions and to hold data. If the cache contains data, writes to data at locations contained in the cache also cause external bus transactions to update the appropriate memory location.

Both the CPU and the on-chip DMAs access the cache. For the CPU, if the MMU is enabled, the access can be either cacheable or non-cacheable, depending on the value of the Cacheable bit in the Page Descriptor register used to translate the logical address. If the MMU is not enabled, all memory transactions are considered to be cacheable. Two bits in the Cache Control register, the Cache Instructions Disable bit and the Cache Data Disable bit, further determine the operation of the cache for various situations. These bits enable the cache for instructions and for data.

When the on-chip memory is used as fixed memory locations, neither the Cache Instruction Disable or Cache Data Disable bits are used, and no distinction is made as to whether the CPU is accessing data or instructions.

In general, when devices such as on-chip DMAs transfer data to the memory, the cache data is modified if the write is to a valid location in the cache but the LRU mechanism is unaffected. Also, for the EPU to memory transfer, if the cache contains valid locations that are updated by an EPU transaction, the on-chip cache is also updated.

Cache Control Register. The operation of the on-chip memory is controlled by an 8-bit Cache Control register (Figure 21) that is accessed using a load control instruction. This register contains five control bits.

Figure 21. Cache Control Register

The bits in this register are:

High Memory Burst Capability (HMB). This 1-bit field specifies whether a memory burst transaction occurs when the MMU is enabled and there is a 1 in bit 15 of the selected Page Descriptor register (0 = burst mode not supported, 1 = burst mode supported).

Low Memory Burst Capability (LMB). This 1-bit field specifies whether a memory burst transaction occurs when the MMU is disabled or when the MMU is enabled and there is a 0 in bit 15 of the selected Page Descriptor register (0 = burst mode not supported, 1 = burst mode supported).

Cache Data Disable (D). While this bit is cleared to 0, data fetches are copied into the cache if the M/C bit = 0 (cache mode). If M/C = 1, the state of this bit is ignored.

Cache Instructions Disable (I). While this bit is cleared to 0, instruction fetches are copied into the cache when the M/C bit = 0 (cache mode). When M/C = 1, the state of this bit is ignored.

Memory/Cache (M/C). While this bit is set to 1, the on-chip memory is to be accessed as physical memory; while it is cleared to 0, the memory is accessed associatively as a cache.

If the on-chip memory is to be used as fixed memory locations, the user can programmably select the ranges of memory addresses for which the on-chip memory responds.

CLOCK OSCILLATOR

The Z280 MPU has an on-chip clock oscillator/generator that can be connected to a fundamental, parallel-resonant crystal or any suitable clock source. The bus timing clock generated from the on-chip oscillator is output for use by the rest of the system.

REFRESH

The Z280 MPU has an internal mechanism for refreshing dynamic memory. This mechanism can be activated by setting the Refresh Enable bit in the Refresh Rate register to 1. Memory refresh is performed periodically at a rate specified by the Refresh Rate register. Refresh transactions are identical to memory transactions except that different status signals are used and no data is transferred. They can be inserted immediately after the last clock cycle of any bus transaction, including an internal operation.

The refresh transaction is generated as soon as possible after the refresh period has elapsed (generally after the last clock cycle of the current bus transaction). If the MPU receives an interrupt request, the refresh operation is performed first. When the Z280 MPU does not have control of the bus or is in the Wait state, internal circuitry records the number of refresh periods that have elapsed and refresh cycles cannot be generated. When the MPU regains control of the bus or the $\overline{WAIT}$ input signal is deactivated and the bus transaction completes, the refresh mechanism immediately issues the skipped refresh cycles. The internal circuitry can record up to 256 such skipped refresh operations.

A 10-bit refresh address is generated for each refresh operation with the refresh address being incremented by two between refreshes for 16-bit data bus and by one for 8-bit data bus.

On reset, the Refresh Rate register contains 88_H, refresh is enabled, the rate is 32 processor clock cycles, and the refresh address is not affected.

The Refresh mechanism is controlled by an 8-bit control register, described below.

Refresh Rate Register

This 8-bit register (Figure 22) enables the refresh mechanism and specifies the frequency of refresh transactions.

Figure 22. Refresh Rate Register

The fields in this register are:

Refresh (Rate). This field indicates in processor clock cycles the rate at which refresh transactions are to be generated; a value of n in this field indicates a refresh period of 4n, with Rate = 0 indicating 256 clock cycles.

Refresh Enable (E). When this 1-bit field is set to 1, the refresh mechanism is enabled.

UART

The Z280 UART transmits and receives serial data using any common asynchronous data-communication protocol.

Transmission and reception can be performed independently with five, six, seven, or eight bits per character, plus optional even or odd parity. The transmitter can supply one or two stop bits and can provide a break output at any time. Reception is protected from spikes by a "transient spike-rejection" mechanism that checks the signal one-half a bit time after a Low level is detected on the receiver data input; if the Low does not persist—as in the case of a transient—the character assembly process is not started. Framing errors and overruns are detected and buffered with the partial character on which they occur. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The UART uses the same clock frequency for both the transmitter and the receiver. The input for the UART clocking circuitry is derived from counter/timer 1, either from its external input line for an external clock or from the counter/timer output for a bit rate generated from the internal processor clock. The UART input clock is further scaled by 1, 16, 32, or 64 for clocking the transmitter and receiver.

Two of the DMA channels can be used independently to move characters between memory and the transmitter or receiver without CPU intervention. Both the transmitter and receiver can interrupt the CPU for processor assistance.

The UART uses two external pins, Transmit and Receive. Data that is to be transmitted is placed serially on the Transmit pin and data that is to be received is read in from the Receive pin.

Asynchronous Transmission

The Transmitter Data Output line is held High (marking) when the transmitter has no data to send. Under program control, the Send Break command can be issued to hold the Data Output line Low (spacing) until the command is cleared.

The UART automatically adds the start bit, the programmed parity bit (odd, even, or no parity), and the programmed number of stop bits to the data character to be transmitted. When the character is five, six, or seven bits, the unused most significant bits in the Transmitter Data register are automatically ignored by the UART.

Serial data is shifted from the transmitter at a rate equal to 1, 1/16th, 1/32nd or 1/64th of the clock rate supplied to the transmitter clock input. Serial data is shifted out on the falling edge of the clock input.

Asynchronous Reception

An asynchronous receive operation begins when the Receive Enable bit in the Receiver Control/Status register is set to 1. A Low (spacing) condition on the Receive input line indicates a start bit. If this Low persists for at least one-half of a bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on an otherwise marking line. If the × 1 clock mode is selected, bit synchronization must be accomplished externally; received data is sampled on the rising edge of the clock.

Received characters are read from the Receive Data register. If parity is enabled, the parity bit is assembled as part of the character and is not removed from the assembled character for character lengths other than 8 bits. If the resulting character is still less than 8 bits, 1s are appended in the unused high-order bit positions.

Since the receiver is buffered by one 8-bit register in addition to the receiver shift register, the CPU has adequate time to service an interrupt and to accept the data character assembled by the UART. The receiver also has a buffer that stores error flags for each data character in the receive buffer. These error flags are loaded at the same time as the data character.

After a character is received, it is checked for the following error conditions:

- Parity Error: when the parity bit of the character does not match the programmed parity.
- Framing Error: if the character is assembled without any stop bits (i.e., a Low level is detected for a stop bit).
- Receiver Overrun Error: if the CPU fails to read a data character when more than one character has been received.

Since the Parity Error and Receiver Overrun Error flags are latched, the error status that is read reflects an error in the current character in the Receiver Data register plus any Parity or Overrun Errors detected since the last write to the Receiver Control/Status register. To keep correspondence between the state of the error buffers and the contents of the receiver data buffers, the Receiver Control/Status register must be read before the data.

Polled Operation

In a polled environment, the Receive Character Available bit in the Receiver Control/Status register must be monitored so the CPU can know when to read a character. This bit is automatically cleared when the Receiver Data register is read. To prevent overwriting data in polled operations, the transmitter buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit in the Transmitter Control/Status register is set to 1 whenever the transmit buffer is empty.

UART Control and Status Registers

The UART operation is controlled by three control and status registers. The UART configuration register specifies the character size, parity, clock source, scaling, and loop-back enable. Both the transmitter and the receiver have their own control/status register.

UART Configuration Register. This 8-bit register (Figure 23) contains control information for both the transmitter and receiver.

Figure 23. UART Configuration Register

The control fields for this register are:

Loopback Enable (LB). The UART is capable of local loopback. In this mode the internal transmit data line is tied to the internal receiver line and the external receiver input is ignored. If this bit is set to 1, loop back mode is enabled.

Clock Rate (CR). These two bits specify the multiplier between the clock and data rates (00 = data rate × 1, 01 = data rate × 16, 10 = data rate × 32, 11 = data rate × 64). The same rate is used for both the receiver and transmitter. If the × 1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Select (CS). This bit specifies the clock input for the UART. If the bit is set to 1, the counter/timer 1 output pulse is used for bit-rate generation; if the bit is cleared to 0, the input line to counter/timer 1 provides the clock from an external source.

Parity Even/Odd (E/O). If parity is specified, this bit determines whether it is sent and checked as even or odd (1 = even).

Parity (P). If this bit is set to 1, an additional bit position (in addition to those specified in the bits/character control field) is added to transmitted data and is expected in received data. In the Receiver, the parity bit received is transferred to the CPU as a part of the character, unless eight bits/character is selected.

Bits/Character (B/C). Together, these two bits determine the number of bits to form a character. If these bits are changed during the time that a character is being assembled, the results are unpredictable (00 = 5 bits/character, 01 = 6 bits/character, 10 = 7 bits/character, 11 = 8 bits/character).

Transmitter Control/Status Register. This 8-bit register (Figure 24) specifies the operation of the transmitter.

Figure 24. Transmitter Control/Status Register

The control bits for this register are:

Transmitter Buffer Empty (BE). This bit is automatically set to 1 whenever the transmitter buffer becomes empty and cleared to 0 when a character is loaded into the transmit buffer. This bit is in the set condition after a reset. This bit is controlled by the UART control circuitry; it can be read by an I/O read but cannot be set to 1 or cleared to 0 by an I/O write.

Value (VAL). This bit determines the value of the bits transmitted while the FRC bit is 1 and dummy characters are loaded into the transmitter buffer. When this bit is 1, a mark character (all 1s) is sent; when this bit is 0, a break character (all 0s) is sent.

Force Character (FRC). When this bit is set to 1, any character loaded into the transmitter buffer causes the transmitter output to be held High or Low (as indicated by the VAL bit) for the length of time required to transmit a character. This allows a program to generate a marking signal or a break of multiple-character duration simply by setting this bit to 1, setting the VAL bit to 1 or 0, and loading the appropriate number of dummy characters into the transmitter buffer.

Send Break (BRK). When set to 1, this bit immediately forces the transmitter output to the spacing condition, regardless of any data being transmitted. When this bit is cleared to 0, the transmitter returns to marking.

Stop Bits (SB). This bit determines the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit. If this bit is set to 1, two stop bits are automatically appended to the character sent; if this bit is cleared to 0, only one stop bit is appended.

Transmitter Interrupt Enable (IE). When this bit is set to 1, interrupt requests are generated whenever the transmitter buffer becomes empty; when this bit is cleared to 0, no requests are made.

Transmitter Enable (EN). While this bit is cleared to 0, data is not transmitted and the transmitter output is held marking. Data characters in the process of being transmitted are completely sent if this bit is cleared to 0 after transmission has started.

Receiver Control/Status Register. This 8-bit register (Figure 25) specifies the operation of the receiver. The control bits are described below.

7							0
EN	IE	X	CA	FE	PE	OVE	ERR

Figure 25. Receiver Control/Status Register

Receiver Error (ERR). This bit is the logical OR of the PE, OVE, and FE bits.

Framing Error (FE). This bit is automatically set to 1 for the received character in which the framing error occurred. Detection of a framing error adds an additional one-half of a bit time to the character time so the framing error is not interpreted as a new start bit.

Parity Error (PE). When parity is enabled, this bit is automatically set to 1 for those characters whose parity does not match the programmed sense (even/odd). This bit is latched, so once an error occurs, it remains set until it is cleared by software.

Receiver Overrun Error (OVE). This bit is automatically set to 1 to indicate that more than two characters have been received without a read from the CPU (or DMA). Only the most recently received character is flagged with this error, but when this character is read, the error condition is latched until cleared by software.

Receiver Character Available (CA). This bit is automatically set to 1 when at least one character is available in the receive buffer; it is automatically cleared to 0 when the Receiver Data register is read. This bit is controlled by the UART control circuitry; it can be read by an I/O read but cannot be set or cleared by an I/O write.

Receiver Interrupt Enable (IE). While this bit is set to 1, interrupt requests are generated whenever the receiver detects an error or the receiver has a character available.

Receiver Enable (EN). When this bit is set to 1, receiver operations begin. This bit should be set only after the parameters in the UART Configuration register are set.

UART Bootstrapping Option

The Z280 CPU supports an automatic initialization of memory via the UART after a reset operation. This system bootstrapping capability permits ROMless system configurations: the memory can be initialized by a serial link before the Z280 CPU fetches information from memory after the reset.

On the rising edge of Reset, the AD lines are sensed if $\overline{WAIT}$ is asserted; if AD_6 is being driven High, the Z280 CPU automatically enters a Halt state. The UART is also automatically initialized to receive 8-bit character data with odd parity at a × 16 clock rate. An external clock source is assumed. A minimum of 15 processor clock cycles must elapse before the transmission can begin.

During the bootstrapping operation, DMA Channel 0 is used to transfer received characters into the memory. This channel is initialized as follows:

***Transaction Descriptor register*--IE, EPS, and TC cleared, ST-byte transfer, BRP-continuous, TYPE-flowthrough, DAD-Auto-increment memory address**

DMA Master Control register—DOR and EOP set

Count register—0100 (256 bytes to be transferred)

Destination Address register—000000 (starting memory address = 0)

Source Address register—undefined (not used when DMAO is linked to UART

Characters received are placed in memory starting at physical memory location zero. If an error occurs, the Z280 CPU drives the Transmitter Output line Low. External circuitry monitoring this line can use this signal to cause the transmitting device to begin the initialization procedure again, starting with a reset and AD_6 asserted on the rising edge of Reset.

After 256 bytes of data have been transferred, the Z280 CPU automatically begins execution by fetching the first instruction from memory location 0.

DMA CHANNELS

The Z280 MPU has four on-chip Direct Memory Access (DMA) channels to provide high bandwidth data transmission capabilities. There are two types of DMA channels; two support flyby transactions and the other two do not. The two types of DMA channels otherwise have identical capabilities, although they have different priorities with respect to interrupt requests and bus requests.

Each DMA channel is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features requiring little or no external logic in systems using an 8- or 16-bit data bus.

Transfers can be performed between any two ports (source and destination), including memory-to-I/O, I/O-to-memory, memory-to-memory, and I/O-to-I/O. Except for flyby, two port addresses are automatically generated for each transaction and can be either fixed or incrementing/decrementing.

During a transfer, a DMA channel assumes control of the system address and data bus. Data is read from one addressable port and written to the other addressable port, byte-by-byte or word-by-word. The ports can be programmed to be either system main memory or peripheral I/O devices.

For both flyby and flowthrough DMA transactions, if the destination is a memory location that corresponds to an entry in the on-chip memory (either cache or fixed memory location), the on-chip memory is updated to reflect the new contents of the memory location.

Except in flyby mode, two 24-bit addresses are generated by the DMA for every transfer operation, one address for the source port and another for the destination port. Two readable address counters (three bytes each) keep the current address of each port.

The DMA devices use the same memory and I/O timing as the CPU for bus transactions, as indicated by the appropriate bus timing register.

Modes of Transfer Operation

Each DMA can be programmed to operate in one of three transfer modes:

- *Single Transaction.* Data operations are performed one byte or word at a time.
- *Burst.* Data operations continue until a port's Ready line to the DMA goes inactive.
- *Continuous.* Data operations continue until either the end of the programmed block of data is reached or an end of process has been signaled before the system bus is released.

In all modes, once a byte or word of data is read by the DMA channel, the operation is completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Pin Descriptions

Each DMA channel has a Ready input line. In addition, two DMA channels have a flyby output line to support high speed data transfers between I/O devices and memory.

The flyby output is asserted by the DMA channel to signal a peripheral device associated with the DMA channel that it should participate in the data transmission during the current flyby bus transaction.

If Ready is active, the DMA channel requests control of the external system bus to perform the DMA transaction. When the external system bus is available for DMA transfers, the DMA channel with a request pending and the highest priority assumes bus mastership. The priority of DMA channels from highest to lowest is: DMA0, DMA1, DMA2, and DMA3. A DMA channel in burst mode relinquishes bus mastership to a higher priority DMA channel only when its Ready line is deasserted (or EOP is signaled or terminal count is reached). A DMA channel in continuous mode relinquishes bus mastership only when EOP is signaled or terminal count is reached.

Priority of On-Chip DMA Channels and External Bus Requesters

The on-chip DMA channels are arranged in a daisy chain with the external Bus Request input line being the "next lower bus requester" on this chain. The on-chip DMAs behave as if they were external bus requestors with respect to acquiring the bus, relinquishing the bus, and priority access to the bus.

End-of-Process

If the end-of-process (EOP) capability is enabled, transfers by DMA channels can be prematurely terminated by a Low on Interrupt A line or Interrupt B line during the transfer. This capability is programmed by control bits in the DMA Master Control register. EOP occurs regardless of the

setting of the Interrupt A Enable bit in the Master Status register. When an EOP is signaled, the EOP Signaled (EPS) bit in the Transaction Descriptor register of the active DMA channel is set to 1 and the Enable bit is cleared to 0. If interrupt requests are enabled (IE = 1 in the Transaction Descriptor register), an interrupt request is generated by the channel that was active when the EOP was signaled. After an EOP has been signaled, the DMA relinquishes the bus within 16 cycles of the last DMA bus transaction.

If the End-Of-Process signal on Interrupt A or B line is still asserted when the CPU is bus master, the signal is interpreted as an interrupt request; thus, both the DMA channel and the external EOP generating device can request interrupts simultaneously. Separate mask bits in the Master Status register enable the CPU to accept interrupts from these two sources.

On a flowthrough transaction, if the EOP signal is received while the information is being read into the Z280 MPU, the transfer is aborted and the data is not written out from the Z280 MPU.

DMA Linking

The DMA devices can be linked together to achieve DMA transfers to non-contiguous memory locations (linked operation). Bits in the DMA Master Control register allow DMA3 to be linked to DMA1 and DMA2 to be linked to DMA0. If the appropriate bit is set to 1 in the DMA Master Control register, the master DMA (0 or 1) signals its linked DMA each time its transfer is complete (count = 0). This acts as an internal ready input to the linked DMA that reloads the master DMA control registers.

Words are loaded into the master DMA control registers in the following order: Destination Address register (two words), Source Address register (two words), Count (one word), Transfer Descriptor register (one word). After six words have been transferred, the master DMA deasserts its internal ready line and begins the transfer of the next block **of data. The master DMA can be programmed to interrupt the CPU on "count equals 0" when the last block transfer is completed by the master DMA (to notify software that the entire sequence of transfers is completed).**

When programming linked DMAs, the last word to be programmed must be the master DMA's Transaction Descriptor register. Also, the linked DMA must be programmed before the master DMA's status register is programmed.

DMA Master Control Register. This 16-bit register (Figure 26) specifies the general configuration of the four on-chip DMA channels: the linking of the DMA channels, the software ready enables, and EOP enable.

X	X	X	X	EOPCSB	EOPCSA	EOPB	SRI	SRO	EOPA	D3L	D2L	D1T	D0R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 26. DMA Master Control Register

The fields in this register are:

DMA0 to Receiver Link (D0R). When this bit is set to 1, DMA channel 0 is linked to the UART receiver.

DMA1 to Transmitter Link (D1T). When this bit is set to 1, DMA channel 1 is linked to the UART transmitter.

DMA2 Link (D2L). When this bit is set to 1, DMA channel 2 is linked to DMA channel 0.

DMA3 Link (D3L). When this bit is set to 1, DMA channel 3 is linked to DMA channel 1.

***End-of-Process (EOP_A).* When this bit is set to 1, the INT_A line is used as an end-of-process signal for the DMA channel defined by the EOPCSA field.**

***End-of-Process (EOP_B).* When this bit is set to 1, the INT_B input acts as an EOP input for the DMA channel defined by the EOPCSB field.**

Software Ready for DMA0 (SR0). When this bit is set to 1, DMA channel 0 requests service if enabled.

Software Ready for DMA1 (SR1). When this bit is set to 1, DMA channel 1 requests service if enabled.

***End-of-Process Channel Select A (EOPCSA).* This field defines the DMA channel that has $\overline{INT}_A$ as its EOP input. This field has no effect if EOP_A bit (bit 4) is cleared to zero**

00	DMA Channel 0
01	DMA Channel 1
02	DMA Channel 2
03	DMA Channel 3

***End-of-Process Channel Select B (EOPCSB).* This field defines the DMA channel that has $\overline{INT}_B$ as its EOP input. This field has no effect if EOP_B bit (bit 7) is cleared to zero.**

00	DMA Channel 0
01	DMA Channel 1
02	DMA Channel 2
03	DMA Channel 3

Note that while the EOP_A and EOP_B bits are active, $\overline{INT}_A$ and $\overline{INT}_B$ can still serve as interrupt inputs.

DMA Channel Control Registers

Transaction Descriptor Registers. These four 16-bit registers, one for each channel (Figure 27), describe the type of DMA transfer to be performed and contain control and status information.

EN	SAD	IE	ST	BRP	TYPE	TC	DAD	EPS
15								0

Figure 27. Transaction Descriptor Register

The fields in this register are:

***End-of-Process Signaled (EPS).* This bit is set to 1 automatically when the channel is active and an end-of-process is signaled for this channel as programmed on the Interrupt A or Interrupt B input lines, thus prematurely terminating the transfer.**

Destination Address Descriptor (DAD). The setting of this 3-bit field indicates the type of location (memory or I/O) and how the address is to be manipulated (incremented, decremented or left unchanged), as shown in Table 5.

Table 5. SAD and DAD Encodings

Encoding	Address Modification Operation
000	Auto-increment memory location
001	Auto-decrement memory location
010	Memory address unmodified by transaction
011	Reserved
100	Auto-increment (by 1) I/O location
101	Auto-decrement (by 1) I/O location
110	I/O address unmodified by transaction
111	Reserved

Transfer Complete (TC). This bit is set to 1 automatically when the count register has reached zero.

Transaction Type (Type). This 2-bit field specifies flyby or flowthrough type of operation (00 = flowthrough, 01 = reserved, 10 = flyby write, 11 = flyby read). In flowthrough mode of operation, two bus transactions occur for each DMA operation—a read from the source followed by a write to the destination. In a flyby operation, only one bus transaction occurs for each DMA operation. In flyby write to memory, the flyby output pin is pulsed instead of an I/O transaction being performed and the contents of the Destination Address register are output to specify the memory location. In flyby read from memory, the flyby output pin is pulsed instead of an I/O transaction being performed and the contents of the Source Address register are output to specify the memory location. Only two DMAs have flyby capability.

Bus Request Protocol (BRP). The setting of these two bits indicates the mode of DMA operation (Table 6).

Table 6. Bus Request Protocol (BRP)

Encoding	DMA
0 0	Single Transaction
0 1	Burst
1 0	Continuous
1 1	Reserved

Size of Transfer (ST). This 2-bit field specifies the size of the entity to be transferred by the DMA channel (Table 7). For word transfers to or from memory locations, the memory address must be even (least significant bit is 0). Long word (32-bit) transfers are supported only in flyby mode, with the cache disabled.

Table 7. Size of Transaction (ST)

Encoding ST1	ST0	Size of Transfer	Number to Increment/ Decrement By
0	0	Byte	1
0	1	16-bit word	2
1	0	32-bit longword	4
1	1	Reserved	

Interrupt Enable (IE). When this bit is set to 1, the DMA generates an interrupt request at end of count or end of process. When this bit is 0, no interrupt request is generated.

Source Address Descriptor (SAD). The setting of this 3-bit field indicates the type of location (memory or I/O) and how the address is to be manipulated (incremented, decremented or left unchanged), as shown in Table 5.

DMA Enable (EN). While this bit is 1, the DMA transfer is enabled.

Count Register. This 16-bit register is programmed to contain the number of DMA transfers to be performed. When the contents of the count register reach zero, further requests on the RDY input line are ignored. The DMA channel can be programmed to generate an interrupt when the count register reaches zero.

Source Address Register and Destination Address Register. These 24-bit registers contain the 24-bit physical addresses to be used during the DMA transaction. They are not translated by the MMU. In flyby mode, only one of these registers is used to supply the address for the bus transaction as indicated in the Mode field in the Transfer Descriptor register. The format for these registers is shown in Figure 28.

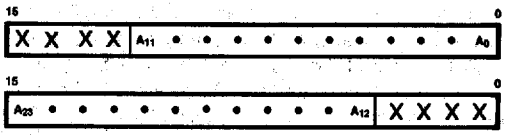


Figure 28. Source and Destination Address Registers Format

Flyby Transaction Timing

The Transaction Type field in the Transaction Descriptor register indicates whether the transaction is a read or a write. For flyby read transactions, the Source Address Descriptor indicates the transaction is a read from memory; for write flyby transactions the Destination Address Descriptor indicates the transaction is a write to memory. Additional wait states can be automatically inserted if programmed in the appropriate timing register. See Figures 29 and 30 for timing diagrams.

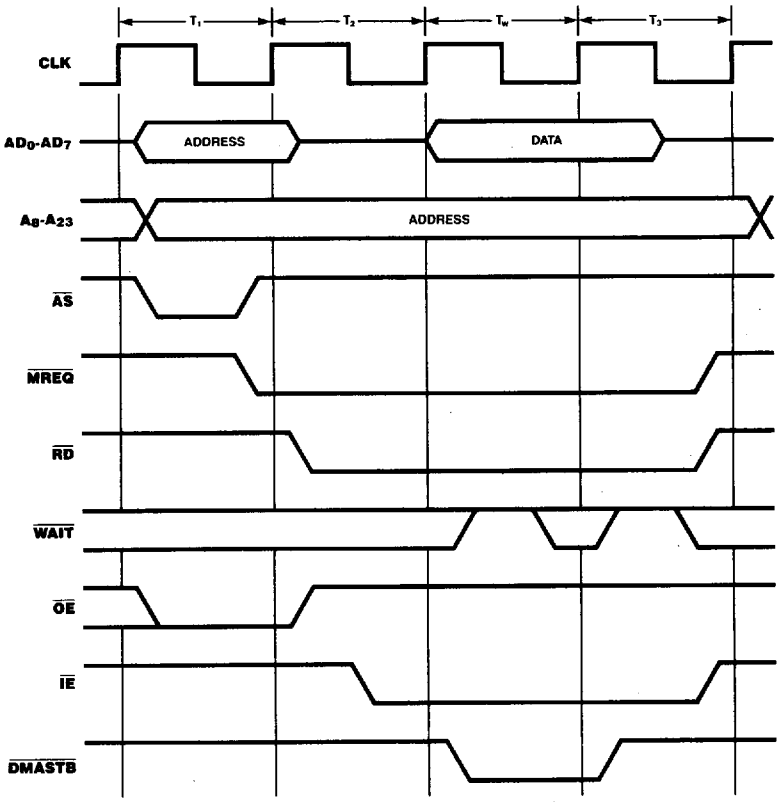


Figure 29a. On-Chip DMA Channel Flyby Memory Read Transaction, Z80 Bus

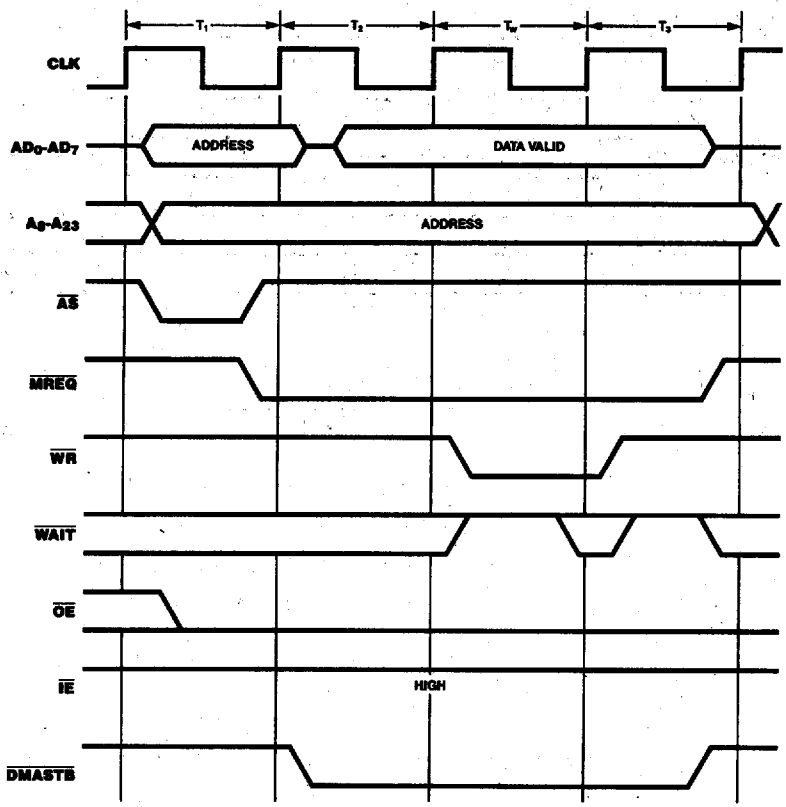


Figure 29b. On-Chip DMA Channel Flyby Memory Write Transaction, Z80 Bus

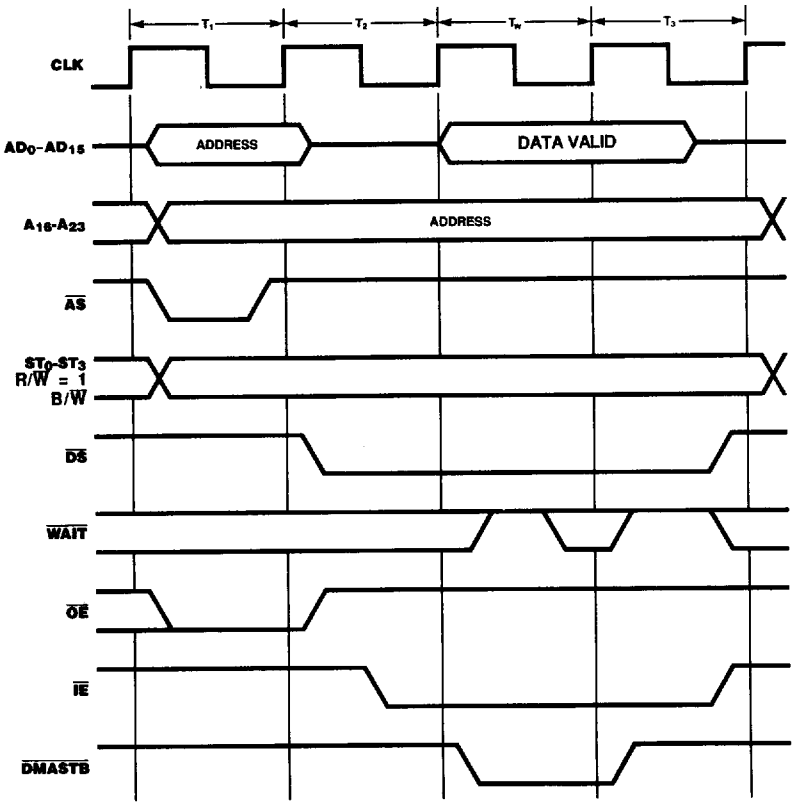


Figure 30a. On-Chip DMA Channel Flyby Memory Read Transaction, Z-BUS

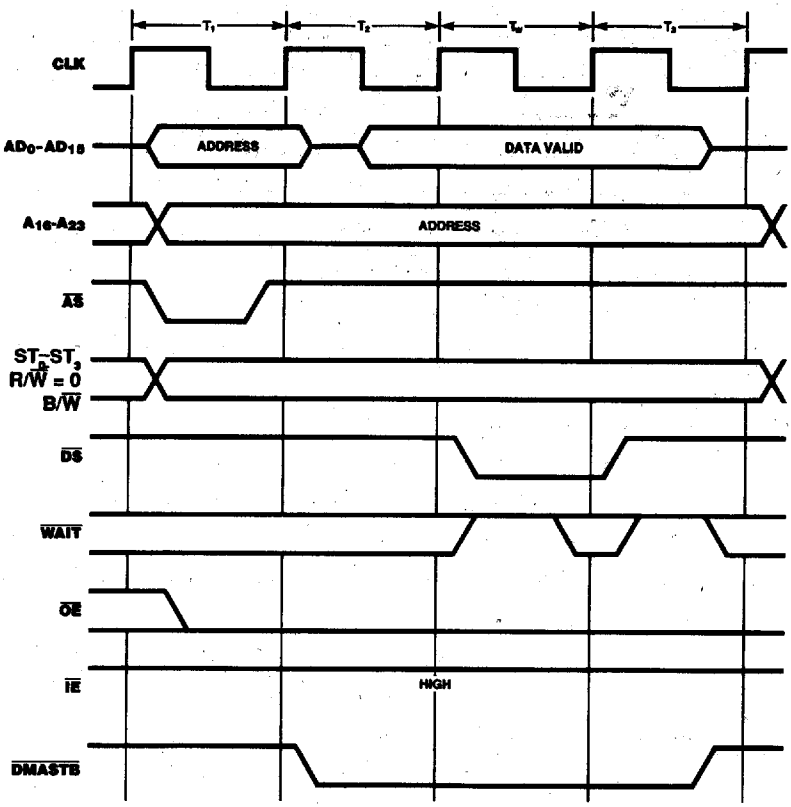


Figure 30b. On-Chip DMA Channel Flyby Memory Write Transaction, Z-BUS

COUNTER/TIMERS

The Z280 MPU's three counter/timers can be programmed by system software for a broad range of counting and timing applications. The three independently programmable channels satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock generation.

Programming the counter/timers is straightforward: each channel is programmed with four bytes. Once started, the channel counts down, and optionally reloads its time constant automatically and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because each channel uses a unique vector from the Interrupt/Trap Vector Table.

Each channel is individually programmed with three registers: a configuration byte, a control byte, and a time-constant word. The configuration byte selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. In the timing mode, the CPU processor clock is divided by four for input to the counter/timers. The time-constant word contains a value from 0 to 65,535.

During operation, the individual counter channel counts down from the present time-constant value. In counter mode operation, the counter decrements on each of the input pulses until the count/time output condition is met. Each decrement is synchronized by the scaled internal processor clock. For counts greater than 65,536, two of the counters can be programmably cascaded. When the count/time output condition is reached, the downcounter is automatically reset with the time constant value, if so programmed.

The timer mode determines time intervals without additional logic or software timing loops. Time intervals are generated by dividing the internal processor clock by four and decrementing a presettable downcounter. Thus, the time interval is an integral multiple of the processor clock period, the prescaler value four, and the time constant that is preset in the downcounter. A timer is triggered by setting the software trigger control bit in the Control/Status register or by an external input.

All three channels can generate an external output when the count/time output condition is met. The output is high when the internal presettable downcounter contains all zeros.

Each channel can be programmed to generate an Interrupt Request, which occurs only if the channel has its Interrupt Enable control bit set to 1 by software programming. When the Z280 CPU accepts the interrupt request it automatically vectors through the Interrupt Vector Table.

The three channels of the Z280 MPU are fully prioritized and fit into three different slots in the Z280 internal peripheral daisy-chain interrupt structure. Channel 0 has the highest priority and Channel 2 has the lowest. The channels have separate interrupt enables and the CPU's Master Status register has individual control bits that selectively inhibit interrupts from each channel.

Modes of Operation

The counter/timer channels have two basic modes of operation: as counters or as timers. As counters they monitor external input lines and record Low to High transitions on these lines. In the timer mode, the processor clock, scaled by four, is used instead of the external input line. The duration of this counting or timing can be either continuous from initial enabling (trigger operation) or only during intervals specified by signals on an input line (gate and gate/trigger operation). The count can be automatically restarted by programming the Retrigger Enable control bit in the counter/timer's Configuration register.

Each of the three counter/timers has a software gate and trigger facility that extends the hardware capabilities of the counter/timers.

Counting Operation. While the appropriate enabling conditions are met, the counter/timer monitors its input line for Low-to-High transitions. When such a transition occurs, the Count/Time register is decremented by 1.

Timing Operation. While the appropriate enabling conditions are met, the counter/timer monitors the internal processor clock scaled by four for Low-to-High transitions. When such a transition occurs the Count/Time register is decremented by 1.

Gate Operation. A counter/timer can be programmed to count or time only when a gating condition is met. While the counter/timer is enabled and the external gate capability is selected, an external input line is monitored; only while this line is High are the counting or timing operations performed. The software gate facility filters the state of the input line; while the software gate bit in the Command and Status register is cleared to 0, the gating condition is not met regardless of the signals on the gating line. The gate facility is illustrated in Figure 31.

Trigger Operation. A counter/timer can be programmed to count or time only after a triggering condition occurs. While the counter/timer is enabled and the external trigger capability is programmed, an external input line is monitored; only after this line makes a Low-to-High transition is a counting or timing operation performed. The software trigger facility causes the triggering condition to be met regardless of the activity of this line. The trigger operation is illustrated in Figure 32.

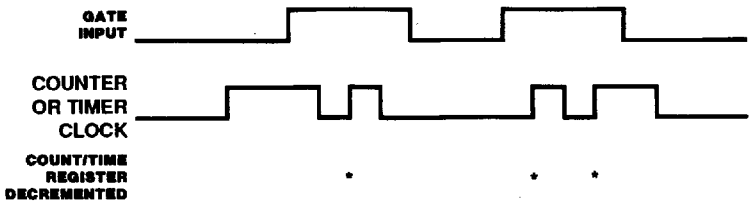


Figure 31. Gate Facility

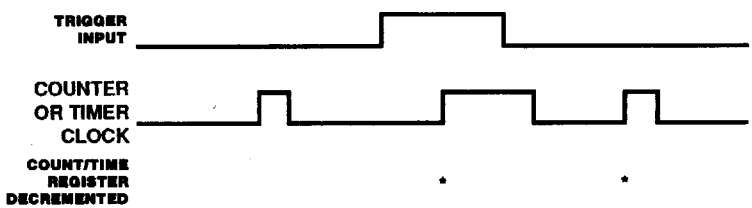


Figure 32. Trigger Operation

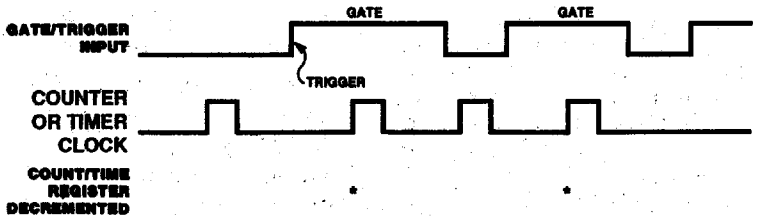


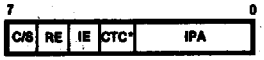
Figure 33. Gate/Trigger Operation

Gate/Trigger Operation. One input line can be used for both the gating and the triggering functions. A Low-to-High transition on this line acts as a trigger and subsequent High signals on this line function as gate signals. If non-retriggerable mode is programmed, subsequent Low-to-High transactions do not cause a trigger. Gate/Trigger Operation is shown in Figure 33.

The software gate and trigger mechanism can also be used in this mode of operation. A software gate before a trigger (hardware or software) has no effect on the counter/timer. After a hardware or software trigger, the software gate must be set to 1 for the Count/Time register to be decremented. A software trigger after a hardware or software trigger has no effect unless the Retrigger Enable control bit is set to 1.

Counter/Timer Control and Status Registers

Each counter/timer has two 8-bit control registers and two 16-bit count registers. The Configuration register and Command/Status register determine the counter/timers' operation, the Counter/Timer Command/Status register provides information about the current operation, the Time Constant register contains the initialization value for the counter/timer, and the Count/Time register contains the current value of the count in progress.



* Only the CTC bit in Counter/Timer 0 is used.

Figure 34. Counter/Timer Configuration Register

Counter/Timer Configuration Register. This 8-bit register (Figure 34) specifies the counter/timer's mode of operation: the pin configuration, whether an interrupt request is generated, and whether the countdown sequence is automatically restarted when the count reaches zero or when a trigger occurs.

The fields in this register are:

Input Pin Assignments (IPA). This 4-bit field specifies the functionality of the input lines associated with the counter/timer and whether the counter/timer monitors an external input (counting operation) or uses the scaled internal processor clock (timing operation). The four bits in this field can be associated with enabling output generation (EO), selecting the external signal or internal clock (C/T), enabling the gating facility (G), and enabling the triggering facility (T). The selected options determine the functions associated with each input line associated with the counter/timer, as illustrated in Table 8.

Table 8. Input Pin Functionality

IPA Field				Pin Functionality		
E0	C/T	G	T	Counter/Timer I/O	Counter/Timer Input	Notes
0	0	0	0	Unused	Unused	Timer
0	0	0	1	Unused	Trigger	Timer
0	0	1	0	Gate	Unused	Timer
0	0	1	1	Gate	Trigger	Timer
0	1	0	0	Unused	Input	Counter
0	1	0	1	Trigger	Input	Counter
0	1	1	0	Gate	Input	Counter
0	1	1	1	Gate/Trigger	Input	Counter
1	0	0	0	Output	Unused	Timer
1	0	0	1	Output	Trigger	Timer
1	0	1	0	Output	Gate	Timer
1	0	1	1	Output	Gate/Trigger	Timer
1	1	0	0	Output	Input	Counter
1	1	0	1	Unused	Unused	Reserved
1	1	1	0	Unused	Unused	Reserved
1	1	1	1	Unused	Unused	Reserved

Counter/Timer Cascade (CTC). When this bit is set to 1, counter/timers 0 and 1 form a 32-bit counter. When used as a 32-bit counter/timer, the fields in the Configuration register and Command/Status register for Counter/Timer 0 are ignored with the exception of the IE, CTC, EO, CIP, CC, and COR fields. The CTC bits in the Counter/Timer Configuration registers of counter/timers 1 and 2 are never used.

Interrupt Enable (IE). While this bit is set to 1, the counter/timer generates an interrupt request when the count/time output condition is met. While this bit is 0, no interrupt request is generated.

Retrigger Enable (RE). While this bit is set to 1, the time constant value is automatically loaded into the Count/Time register when a trigger input is received while the counter/timer is counting down. While this bit is 0, no reloading occurs.

Continuous/Single Cycle (C/S). While this bit is set to 1, the countdown sequence is automatically restarted when the count reaches zero by loading the time constant value into the Count/Time register. While this bit is 0, no reloading occurs.

Counter/Timer Command/Status Register. This 8-bit register (Figure 35) provides software control over the operation of the counter/timer and reflects the current status of the counter/timer's operation. Control bits in this register enable the counter/timer's operation and provide software gate and trigger capabilities. Status bits indicate whether a count is in progress, the count/time output condition has been reached, or the condition has been reached a second time.

7							0
EN	GT	TG	X	X	CIP	CC	COR

Figure 35. Counter/Timer Command/Status Register

The fields of this register are:

Count Overrun (COR). When this bit is set to 1, the count/time output condition has been reached and the CC bit is set to 1, thus indicating a count overrun condition. While this bit is cleared to 0, the count/time output condition has not been reached with the CC bit set since the time the CC bit was cleared by software. This bit can be read or written (set or cleared) by software I/O instructions.

Count/Time Output Condition has been Met (CC). When this bit is set to 1, the Count/Time register has been decremented to zero by the counter/timer control circuitry in single cycle mode or the Count/Time register has been reloaded in continuous mode. When this bit is cleared to 0, the count has not reached the count/time output condition since the bit was cleared by software. This bit can be read or written (set or cleared) by software I/O instructions.

Count In Progress (CIP). While this bit is set to 1, the counter/timer is operating and the Count/Time register is non-zero; while this bit is cleared to 0, the counter/timer is not operating. This bit is controlled by the counter/timer control circuitry; it can be read by an I/O read but cannot be set or cleared by an I/O write instruction.

Software Trigger (TG). When this bit is set to 1 (and the trigger operation of the counter/timer is enabled), if the Enable bit is also set to 1, the trigger operation is enabled on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. That is, if a hardware trigger has not already occurred, the contents of the Time Constant register are loaded into the Count/Time register and the countdown sequence begins. If a hardware trigger has already occurred, then if Retrigger Enable is set to 1, the counter/timer is retriggered; otherwise, setting this bit has no effect. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. When this bit is cleared to 0, this bit has no effect on the operation of the counter/timer.

Software Gate (GT). When this bit is set to 1 (and the gate operation of the counter/timer is enabled), if the Enable bit is also set to 1, operation begins on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. When this bit is cleared to 0, the countdown sequence is halted.

Enable (EN). While this bit is set to 1, the counter/timer is enabled; operation begins on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. Reset clears this bit. While this bit is cleared to 0, the value in the Time Constant register is constantly transferred to the Count/Time register. If the Time Constant register is all zeros, the output of the counter/timer is one. Thus, when the counter/timer is not enabled, the counter/timer output in conjunction with the Time Constant register can be used as an I/O port. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. While this bit is 0, the counter/timer performs no operation during the next (and subsequent) processor clock periods.

Time Constant Register. This 16-bit register holds the value that is automatically loaded into the Count/Time register when the counter/timer is enabled or in the continuous or retrigger mode when the count reaches zero or the trigger is asserted, respectively. This register can be read or written by I/O instructions.

Count/Time Register. This 16-bit register holds the current value of the count or timing in progress. It is automatically loaded from the Time Constant register, and can be read by software using the I/O read instructions.

Pin Descriptions

The counter/timers have two external input lines associated with them. The I/O lines transfer signals between the counter/timers and external devices. The input lines receive signals from external devices for the counter/timers. The interpretations of the signals on these lines is determined by the Input Pin Assignment field in the Configuration register.

MULTIPROCESSOR MODE OF OPERATION

Features

- Allows global memory areas for shared resources
- Global memory addresses are user-specified
- Separate requests for local and global buses
- Requesting mechanism is transparent to user
- Easily interfaces to external arbiters

The Z280 supports various multiprocessor configurations, wherein it is the default bus master of the local bus, and it goes through a defined protocol to access the global bus. To invoke the multiprocessor mode, the Local Address Register contents should be defined, and the MP bit of the Bus Timing and Initialization Register set.

Pin Functionality When the Z280 is in the multiprocessor mode, Counter/Timer 0's IO pin is used as the Global Request ($\overline{GREQ}$) output, and Counter/Timer 0's Input pin is used as the Global Acknowledge ($\overline{GACK}$) input.

Local Address Register. Before an external memory bus transaction is to proceed, the Z280 distinguishes whether a bus transaction uses the local or global bus by comparing the four most significant bit of the physical address (address bits 20 through 23) with a 4-bit Base field in the Local Address register (Figure 36). A mask field in this register specifies which bits are to be compared. If all corresponding address bits match the Base field bits (for those bit positions specified by the mask field), then bus transaction can proceed on the local bus without requesting the global bus; if there is a mismatch in at least one specifies bit position, then the global bus is requested and the bus transaction does not proceed until the global bus acknowledge signal is asserted.

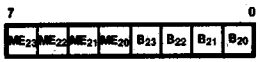


Figure 36. Local Address Register

The bits in the Local Address register are:

Base (B_n). When B_n is 1, address bit A_n must be 1 for a local bus transaction to be performed (unless Match Enable bit ME_n is 0); when bit B_n is 0, address bit A_n must be 0 for a local bus transaction to be performed.

Match Enable (ME_n). When ME_n is 1, address bit A_n is compared to base bit B_n to determine if the address requires the use of the global bus. When ME_n is 0, then any values for A_n and B_n will produce a match. If each ME_n is 0, then all bus transactions are performed on the local bus.

CPU Accesses on the Global Bus

The Z280 is the default local bus master, whether it is in the multi-processor mode or not. It relinquishes the local bus by following a protocol controlled by the $\overline{BUSREQ}$ input and $\overline{BUSACK}$ output pins. When $\overline{BUSREQ}$ is asserted, it is synchronized internally by the CPU. When the CPU is ready to relinquish the local bus, it places all its bus control outputs, including $\overline{GREQ}$, in 3-state, and then drives $\overline{BUSACK}$ active. After reset, the CPU acknowledges a request for the local bus before performing any transactions.

In multi-processor mode, the CPU determines if the next external memory transaction should access the global bus. If such is the case, and if the CPU currently is the local bus master, it puts the global address on the address outputs, and the status signals are also made valid, at the beginning of a bus clock cycle. $\overline{GREQ}$ is asserted in the second half of the same bus clock cycle. The CPU then samples $\overline{BUSREQ}$ and $\overline{GACK}$ continuously. Both inputs are synchronized internally by the CPU. The CPU will proceed with the global transaction after it samples that $\overline{GACK}$ is asserted, with the absence of $\overline{BUSREQ}$. Once the CPU controls the global bus, it can perform multiple global transactions. It relinquishes the global bus when the next transaction should not be global, when $\overline{BUSREQ}$ becomes active, or when $\overline{GACK}$ is de-asserted. A global test and set instruction is atomic (global read is followed by global write), and a global memory burst transaction completes its entire sequence of data transfers.

DMA Accesses on the Global Bus

Each on-chip DMA channel can access the global bus to perform data transfers. The address generated during each DMA-initiated memory transfer is compared with the contents of the Local Address register to determine whether the global bus should be requested. The protocol is identical to the global memory transactions initiated by the CPU.

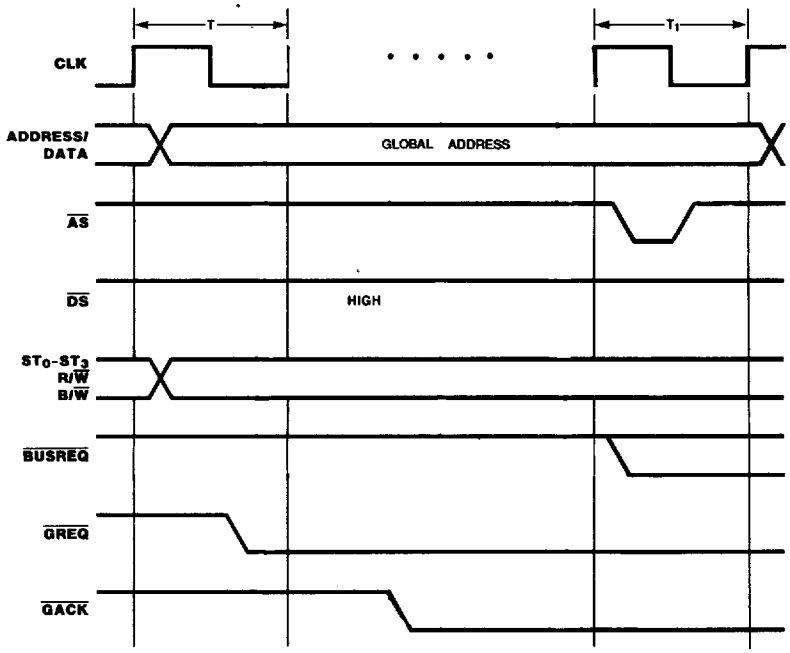


Figure 37. Multiprocessor Mode Timing, Z-Bus Example

EXTERNAL INTERFACE

The two different external interfaces for the Z280 MPU are the 8-bit Z80 Bus and the 16-bit Z-BUS.

Z80 Bus External Interface

Features

- 8-bit data bus
- Multiplexed address/data lines
- Supports Z80 Family peripherals

Pin Descriptions

A_8-A_{23}. *Address* (output, active High, 3-state). These address lines carry I/O addresses and memory addresses during bus transactions.

AD_0-AD_7. *Address/Data* (bidirectional, active High, 3-state). These eight multiplexed Data and Address lines carry I/O addresses, memory addresses, and data during bus transactions.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates the beginning of a transaction and shows that the address is valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). A Low on this line indicates that the CPU has relinquished control of the bus in response to a bus request.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). A Low on this line indicates that an external bus requester has obtained or is trying to obtain control of the bus.

CLK. *Clock Output* **(output). The frequency of the processor timing clock is derived from the oscillator input (external oscillator) or crystal frequency (internal oscillator). The processor clock is further divided by one, two, or four (as programmed) and then output on this line.**

CTIN. *Counter/Timer Input* (input, active High). These lines receive signals from external devices for the counter/timers.

CTIO. *Counter/Timer I/O* (bidirectional, active High, 3-state). These I/O lines transfer signals between the counter/timers and external devices.

$\overline{DMASTB}$. *DMA Flyby Strobe* (output, active Low). These lines select peripheral devices for flyby transfers.

$\overline{EOP}_A$, $\overline{EOP}_B$. ***End of Process*** **(input, active Low). An external source can terminate a DMA operation in progress by driving $\overline{EOP}_A$ or $\overline{EOP}_B$ Low. $\overline{EOP}$ always applies to the corresponding programmed channel; if no channel is active, $\overline{EOP}$ is ignored.**

$\overline{GACK}$. *Global Acknowledge* (input, active Low). A Low on this line indicates the CPU has been granted control of a global bus.

$\overline{\text{GREQ}}$. *Global Request* (output, active Low, 3-state). A Low on this line indicates the CPU has obtained or is trying to obtain control of a global bus.

GND. *Ground.* Ground reference.

$\overline{\text{HALT}}$. *Halt* (output, active Low, 3-state). This signal indicates that the CPU is in the Halt state and is awaiting an interrupt before operation can resume.

$\overline{\text{IE}}$. *Input Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is toward the MPU.

$\overline{\text{INT}}$. *Maskable Interrupts* (input, active Low). A Low on these lines requests an interrupt.

$\overline{\text{IORQ}}$. *Input/Output Request* (output, active Low, 3-state). This signal indicates that AD_0-AD_7 and A_{16}-A_{23} of the address bus hold a valid I/O address for an I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{\text{M1}}$ signal when an interrupt is being acknowledged, to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\text{M1}}$. *Machine Cycle One* (output, active Low, 3-state). This signal indicates that the current transaction is the opcode fetch cycle of a RETI instruction execution. $\overline{\text{M1}}$ also occurs with $\overline{\text{IORQ}}$ to indicate an interrupt acknowledge cycle.

$\overline{\text{MREQ}}$. *Memory Request* (output, active Low, 3-state). This signal indicates that the address bus holds a valid address for a memory read or write operation.

$\overline{\text{NMI}}$. *Nonmaskable Interrupt* (input, falling-edge activated). A High-to-Low transition on this line requests a nonmaskable interrupt.

$\overline{\text{OE}}$. *Output Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is away from the MPU.

OPT. *Bus Option* (input). This signal establishes the bus option during reset.

OPT	Bus Interface
0	Z80 Bus, 8-bit
1	Z-BUS, 16-bit

$\overline{\text{PAUSE}}$. *MPU Pause* (input, active Low). While this line is Low the MPU refrains from transferring data to or from an Extended Processing Unit in the system or from beginning the execution of an instruction.

$\overline{\text{RD}}$. *Read* (output, active Low, 3-state). This signal indicates that the CPU or DMA peripheral is reading data from memory or an I/O device.

$\overline{\text{RDY}}$. *DMA Ready* (input, active Low). These lines are monitored by the DMAs to determine when a peripheral device associated with a DMA port is ready for a read or write operation. When a DMA port is enabled to operate, its Ready line indirectly controls DMA activity; the manner in which DMA activity is controlled by the line varies with the operating mode (single-transaction, burst, or continuous).

$\overline{\text{RESET}}$. *Reset* (input, active Low). A Low on this line resets the CPU and on-chip peripherals.

$\overline{\text{RFSH}}$. *Refresh* (output, active Low, 3-state). This signal indicates that the lower ten bits of the Address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to perform a refresh to all dynamic memories.

RxD. *UART Receive* (input, active High). This line receives serial data at standard TTL levels.

TxD. *UART Transmit* (output, active High). This line transmits serial data at standard TTL levels.

$\overline{\text{WAIT}}$. *Wait* (input, active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

$\overline{\text{WR}}$. *Write* (output, active Low, 3-state). This signal indicates that the bus holds valid data to be stored at the addressed memory or I/O location.

XTALI. *Clock/Crystal Input* (time-base input). Connects a parallel-resonant crystal or an external single-phase clock to the on-chip oscillator.

XTALO. *Crystal Output* (time-base output). Connects a parallel-resonant crystal to the on-chip oscillator.

+5V. *Power Supply Voltage.* (+5 nominal).

Bus Operations

Two kinds of operations can occur on the system bus: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the bus and is known as the bus master. A transaction is initiated by the bus master and is responded to by some other device on the bus. Only one transaction can proceed at a time; seven kinds of transactions can occur:

DMA Flyby. This transaction is used by the DMA peripheral to transfer data between an external peripheral and memory.

Halt. This transaction is used to indicate that the CPU is entering the Halt state.

Interrupt Acknowledge. This transaction is used by the CPU to acknowledge an interrupt and to transfer additional information from the interrupting device.

I/O. This transaction is used by the CPU or DMA peripheral to transfer data to or from an external peripheral.

Memory. This transaction is used by the CPU or DMA peripheral to transfer data to or from a memory location.

Refresh. This type of transaction performed by the refresh peripheral does not transfer data; it refreshes dynamic memory.

RETI. This transaction is generated only by the CPU and is used in conjunction with the Z8400 peripheral's interrupt logic.

Only the bus master can initiate transactions. A request, however, can be initiated by a component that does not have control of the bus. Two types of these requests can occur:

Bus. This request is used by external devices to request control of the system bus to initiate transactions.

Interrupt. This request is used to request the attention of the CPU.

When an interrupt or bus request is made, it is answered by the CPU according to its type. For an interrupt request, the CPU initiates an interrupt acknowledge transaction and for bus requests, the CPU enters bus disconnect state, relinquishes the bus, and activates an Acknowledge signal.

Finally, the Z280 MPU itself may not be the system bus master. See the Multiprocessor Mode section for a discussion of this capability.

Transactions

Information transfers (both instructions and data) to and from the Z280 MPU are accomplished through the use of transactions. All transactions start when $\overline{AS}$ is driven Low and then raised High. This signal can be used to latch Z280 MPU addresses to de-multiplex the Z280 Address/Data lines required by Z80 Family peripherals. Coincident with $\overline{AS}$ assertion, the Output Enable line is also asserted.

If the transaction requires an address, it is valid on the rising edge of $\overline{AS}$. No address is required for Interrupt Acknowledge transactions.

The Read and Write lines are used to time the actual data transfer. (Refresh transactions do not transfer any data and thus do not activate $\overline{RD}$.) For write operations, a Low on $\overline{WR}$ indicates that valid data from the bus master is on the AD lines. The Output Enable line is also activated with $\overline{WR}$. For read operations, the bus master makes the AD lines 3-state before driving $\overline{RD}$ Low so that the addressed device can put its data on the bus. The bus master samples this data on the falling clock edge just before raising $\overline{RD}$ High. The Input Enable line is also activated with $\overline{RD}$.

Wait Cycle. The $\overline{WAIT}$ line is sampled on the falling clock edge when data is to be sampled (i.e., when $\overline{RD}$ or $\overline{WR}$ rises). If the $\overline{WAIT}$ line is Low, another cycle is added to the transaction before data is sampled ($\overline{RD}$ or $\overline{WR}$ rises). In this added cycle and all subsequent cycles added due to $\overline{WAIT}$ being Low, the $\overline{WAIT}$ line is sampled on the falling edge and, if it is Low, another cycle is added to the transaction. In this way, the transaction can be extended by external devices to an arbitrary length to accommodate (for example) slow memories or I/O devices that are not yet ready for data transfer.

The $\overline{WAIT}$ input is synchronous and thus must meet the specified setup and hold times in order for the Z280 MPU to function correctly. This requires asynchronously generated $\overline{WAIT}$ signals to be synchronized to the CLK output before they are input into the Z280 MPU. Automatic wait states can also be generated by programming the Bus Timing and Control register and the Bus Timing and Initialization register; these are inserted in the transaction before the external $\overline{WAIT}$ signal is sampled.

Memory Transactions. Memory transactions move instructions or data to or from memory when the Z280 MPU makes a memory access. Thus, they are generated during program execution to fetch instructions from memory and to fetch and store memory data. They are also generated to store old program status and fetch new program status during interrupt and trap handling, and are used by DMA peripherals to transfer information. A memory transaction is three bus cycles long unless extended with wait states (Figures 38 and 39).

RETI Transactions. These transactions (Figure 40) are similar to two memory read transactions except that $\overline{M1}$ is asserted throughout each read transaction, falling early in the first bus cycle, and that $\overline{MREQ}$, $\overline{M1}$, $\overline{RD}$ and $\overline{IE}$ are deasserted on the rising edge of the clock following the third cycle. Each of the read transactions is followed by a minimum of three bus cycles of inactivity. These transactions are invoked when an RETI instruction is encountered in the instruction stream; they are used during the re-fetching of the instruction from memory so that interrupt logic within Z80 peripherals that monitor the bus for this instruction will function correctly.

Note: Refresh cycles and DMA transfers may occur between RETI bus cycles.

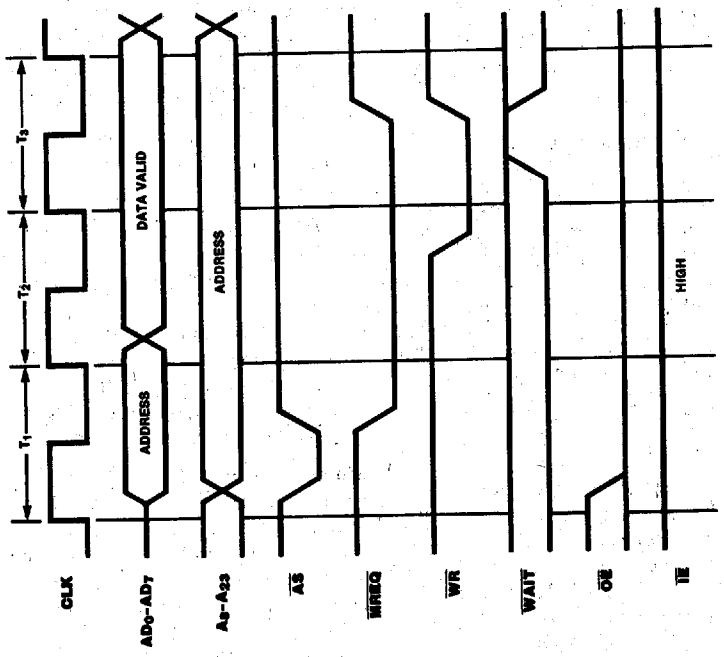


Figure 39. Memory Write Timing

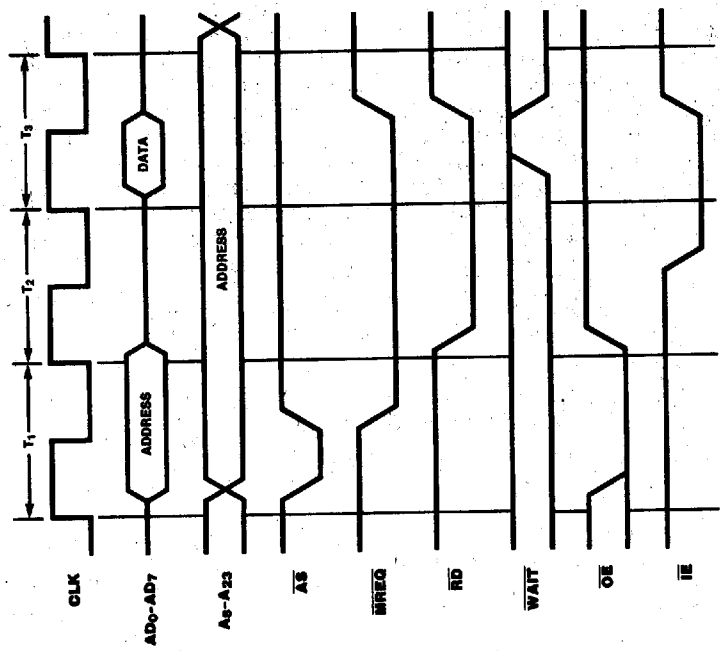


Figure 38. Memory Read Timing

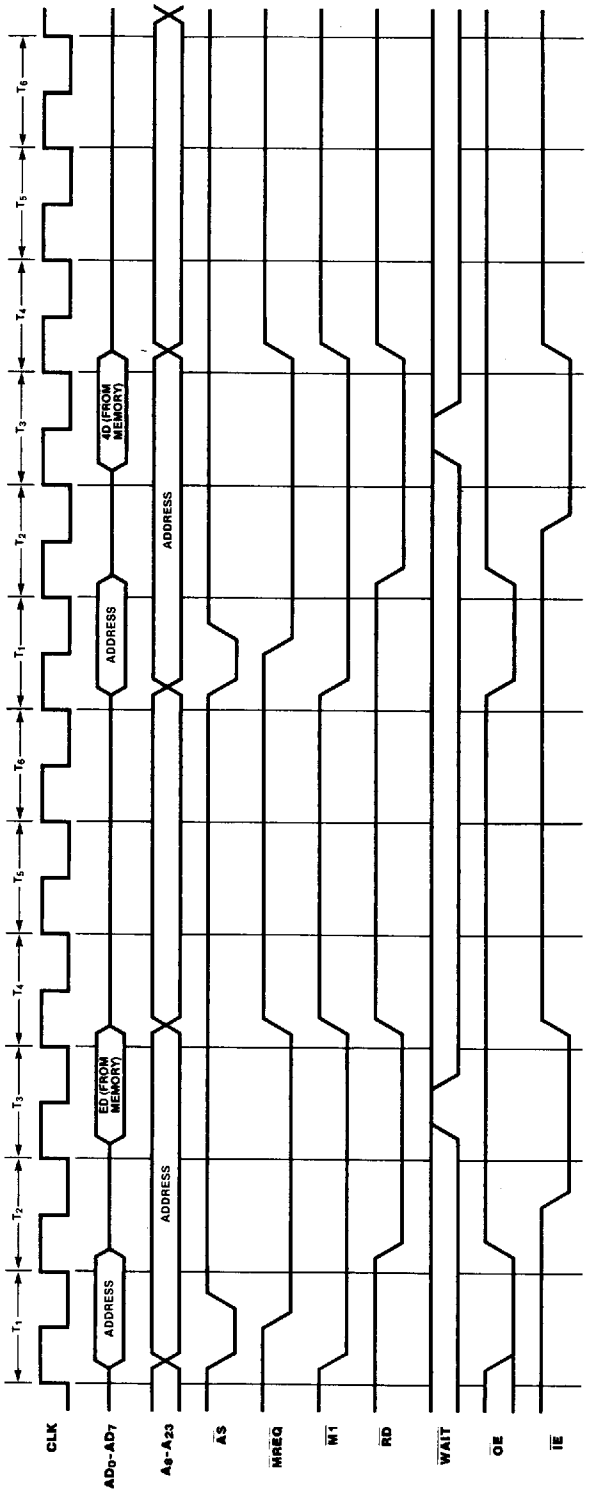


Figure 40. RETI Read Timing

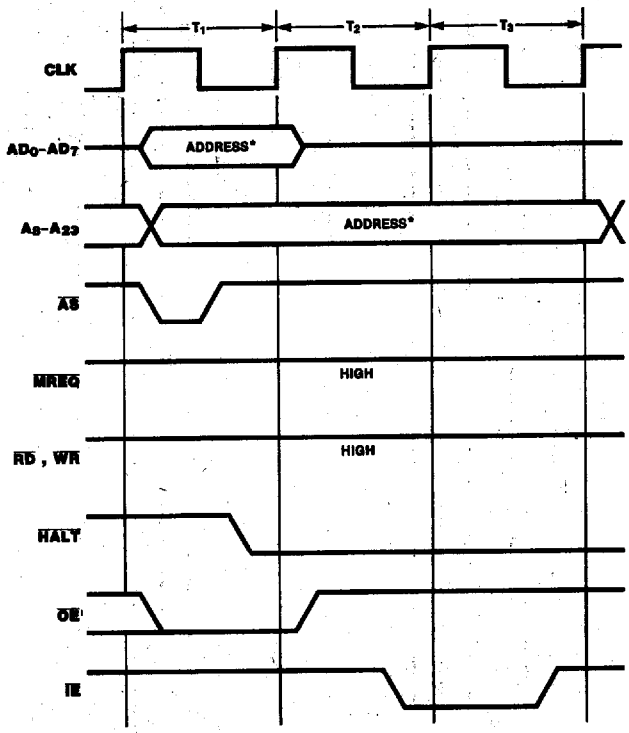
Halt Transactions. The Halt bus transaction does not transfer data (Figure 41). It looks like a memory transaction, except that $\overline{RD}$ and $\overline{WR}$ remain High and no data is transferred. The $\overline{WAIT}$ line is not sampled during the Halt transaction.

Halt transactions are identical to memory read transactions except that $\overline{HALT}$ is asserted throughout the transaction, falling during the second half of the first bus cycle, and remains asserted until an interrupt is acknowledged. This transaction is invoked when a Halt instruction is encountered in the instruction stream or a fatal sequence of traps occurs. Although the Halt transaction is three cycles, the $\overline{HALT}$ line remains asserted until an Interrupt request is acknowledged or a Reset is received. Refresh (to maintain a minimum frequency of bus transactions) or DMA transfers may occur while $\overline{HALT}$ is asserted; also, the bus can be granted. The address put out during the address phase of this cycle is the address of the Halt instruction.

I/O Transactions. I/O transactions move data to (Figure 42) or from (Figure 43) peripherals and are generated during the execution of I/O instructions.

I/O transactions are four clock cycles long at a minimum, and may be lengthened by the addition of wait cycles. The extra clock cycle allows for slower peripheral operation.

The $\overline{IORQ}$ line indicates that an I/O transaction is taking place. The I/O address is found on AD_0-AD_7 and A_8-A_{23} when $\overline{AS}$ rises.



* Address of HALT Instruction.

Figure 41. Halt Timing

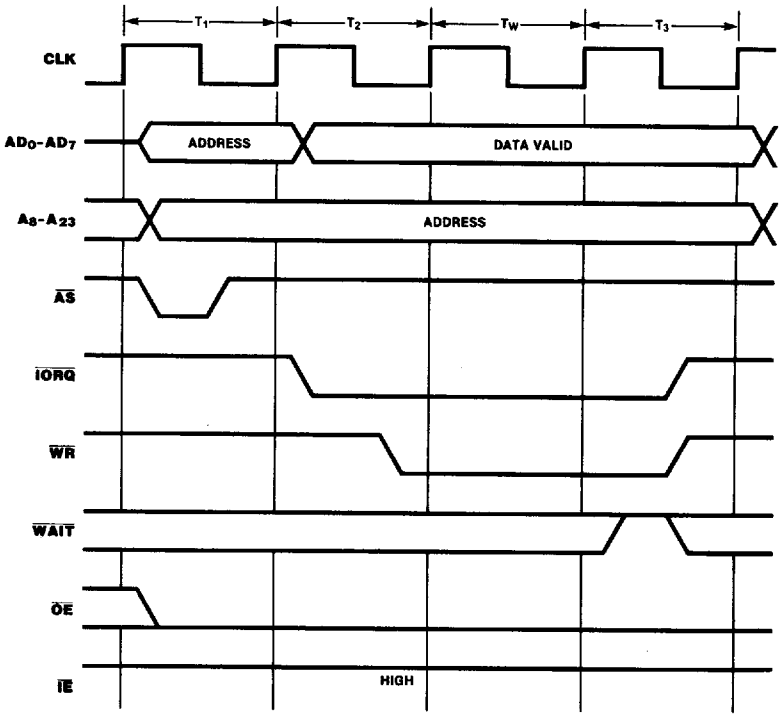


Figure 42. I/O Write Timing

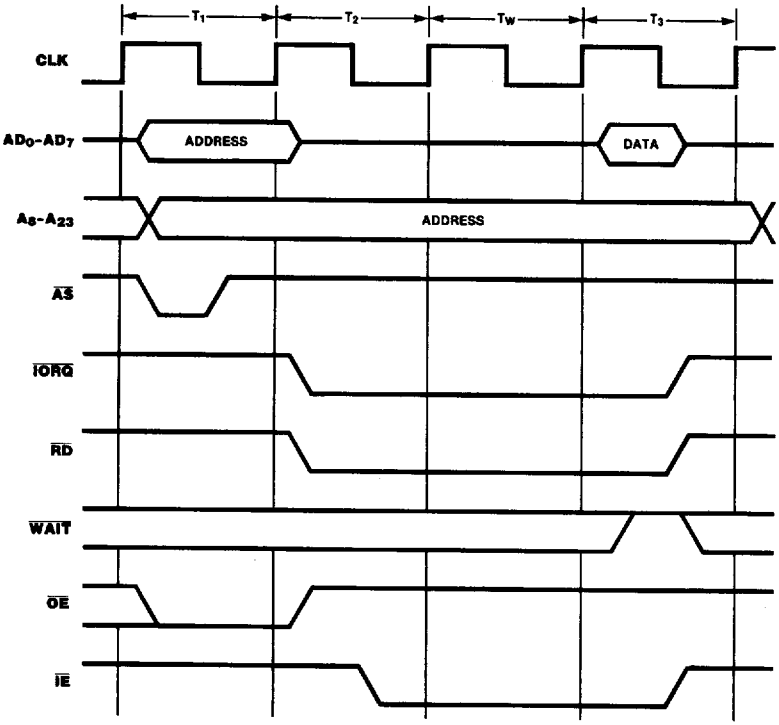


Figure 43. I/O Read Timing

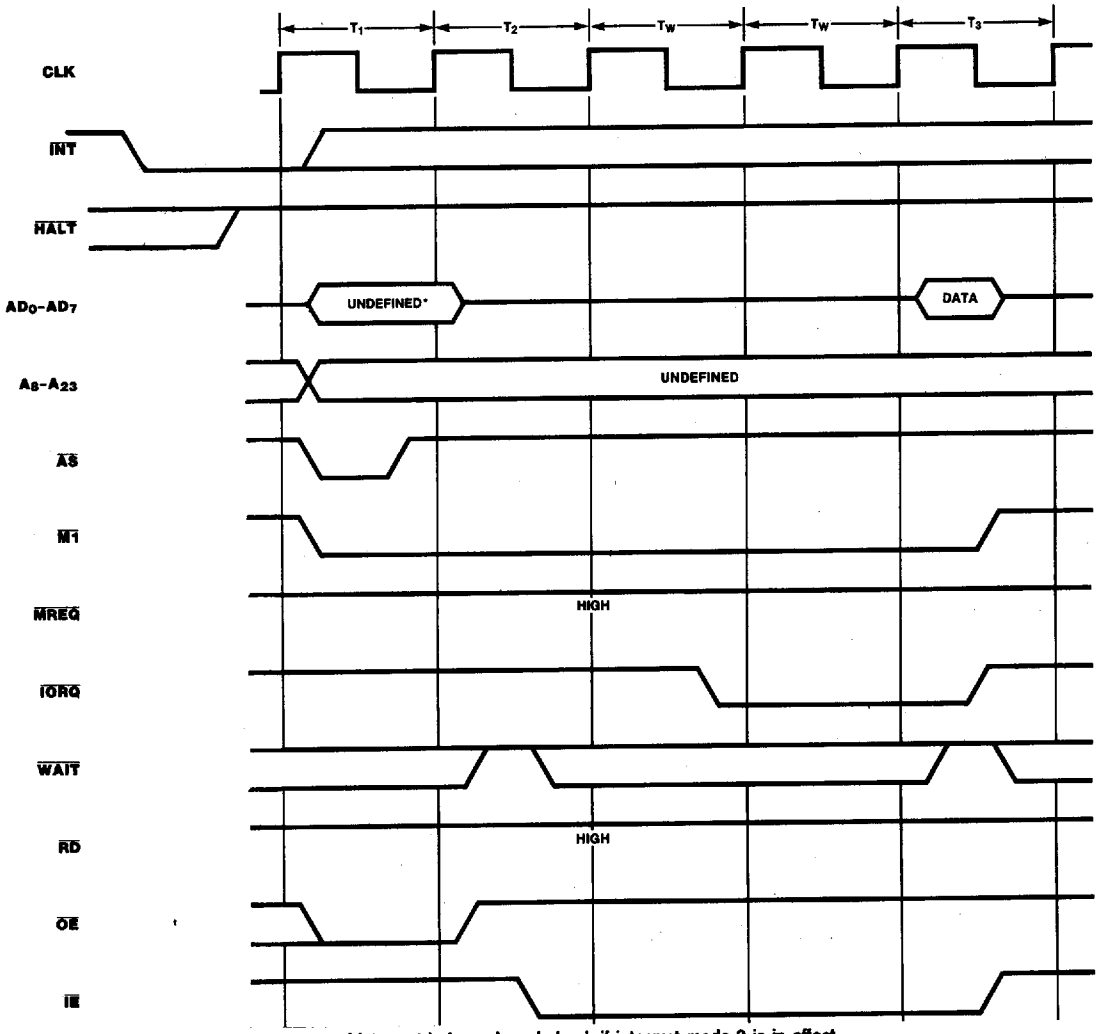
Interrupt Acknowledge Transactions. These transactions (Figure 44) acknowledge an interrupt and read information from the device that generated the interrupt. The transactions are generated automatically by the hardware when an external interrupt request is detected.

The Interrupt Acknowledge transactions are five cycles long at a minimum and have two automatic Wait cycles. The Wait cycles are used to give the interrupt priority daisy chain (or other priority resolution device) time to settle before the identifier is read. Additional automatic Wait states can be generated by programming the Bus Timing and Control register.

The Interrupt Acknowledge transaction is indicated by an $\overline{M1}$ assertion without $\overline{MREQ}$ during the first cycle. During this transaction the $\overline{IORQ}$ signal becomes active during the third cycle to indicate that the interrupting device can place an 8-bit vector on the bus. It is captured from the AD lines on the falling clock edge just before $\overline{IORQ}$ is raised High.

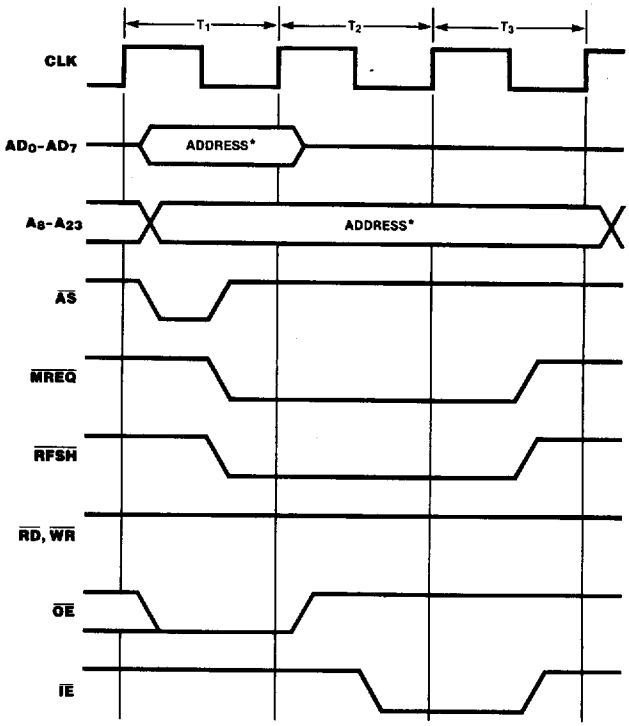
There are two places where the $\overline{WAIT}$ line is sampled and, thus, where a Wait cycle can be inserted by external circuitry. The first serves to delay the falling edge of $\overline{IORQ}$ to allow the daisy chain a longer time to settle, and the second serves to delay the point at which the vector is read.

Refresh Transactions. A memory refresh transaction (Figure 45) is generated by the Z280 refresh mechanism and can occur immediately after the final clock cycle of any other transaction. The memory refresh counter's 10-bit address is output on AD_0-AD_7 and A_8-A_9 during the normal time for addresses. The $\overline{RFSH}$ line is activated with $\overline{MREQ}$. This transaction can be used to generate refreshes for dynamic RAMs.



* AD1 and AD2 indicates type of interrupt being acknowledged, if interrupt mode 3 is in effect.

Figure 44. Maskable Interrupt Acknowledge Sequence



*10 least significant bits are Refresh address, the rest are undefined.

Figure 45. Refresh Timing

Requests

There are three kinds of request signals that the Z280 MPU supports. These are:

- Interrupt requests, which another device initiates and the CPU accepts and acknowledges.
- Bus requests, which an external potential bus master initiates and the Z280 MPU accepts and acknowledges.
- Global bus requests, which the CPU or on-chip DMA initiates to acquire a global System bus.

When a request is made, it is answered according to its type: for interrupt requests, an Interrupt Acknowledge transaction is initiated; for bus requests, an Acknowledge signal is sent; for global bus requests, an Acknowledge signal is received.

Interrupt Requests. The Z280 CPU supports two types of interrupt, maskable and nonmaskable ($\overline{NMI}$). The Interrupt Request line of a device that is capable of generating an interrupt can be tied to the $\overline{NMI}$ or maskable interrupt request lines. Several devices can be connected to one pin with the devices arranged in a priority daisy chain. However, all Z80 family peripherals should be on the same line (or no nesting of interrupts among different lines). The CPU uses different protocols for handling requests on the $\overline{NMI}$ pin than the protocol used for maskable interrupt pins. The sequence of events shown below should be followed:

Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge-detected, and the internal $\overline{NMI}$ latch is set. At the beginning of the last clock cycle in the last internal machine cycle of any instruction, the interrupt inputs are sampled along with the state of the internal $\overline{NMI}$ latch.

If a maskable interrupt is requested and the Master Status register indicates that requests on that line are to be accepted, the next possible bus transaction is the Interrupt Acknowledge transaction, which results in information from the highest-priority interrupting device being read off the AD lines. This data is used to initiate the interrupt service routine. For a nonmaskable interrupt request, the hexadecimal constant 0066 is used to initiate the interrupt service routine, except in mode 3.

Bus Requests. To generate transactions on the bus, a potential bus master (such as the DMA Controller) must gain control of the bus by making a bus request. A bus request is initiated by pulling $\overline{BUSREQ}$ Low. Several bus requesters may be wired-OR to the $\overline{BUSREQ}$ pin; priorities are resolved externally to the CPU, usually by a priority daisy chain.

The asynchronous $\overline{BUSREQ}$ signal generates an internal $\overline{BUSREQ}$, which is synchronous. If the external $\overline{BUSREQ}$ is Low at the beginning of any machine cycle, the internal $\overline{BUSREQ}$ causes the Bus Acknowledge line ($\overline{BUSACK}$) to be asserted after the current machine cycle is completed. (Exceptions are the TSET instruction where the read-modify-write cycle is atomic and DMA transfer in burst or continuous mode.) The CPU then enters Bus Disconnect state and gives up control of the bus. All MPU Output pins, except $\overline{BUSACK}$, are 3-stated.

The CPU regains control of the bus after $\overline{BUSREQ}$ rises. Any device desiring control of the bus must wait at least two bus cycles after $\overline{BUSREQ}$ has risen before pulling it down again.

The on-chip DMA channels have higher priority than external devices requesting the bus via $\overline{BUSREQ}$.

Z-BUS External Interface

Features

- 16-bit data bus
- Multiplexed address/data lines
- Supports high-speed burst mode transfers
- Provides EPA interface

Pin Descriptions

A_{16}-A_{23}. *Address* (output, active High, 3-state). These address lines carry I/O addresses and memory addresses during bus transactions.

AD_0-AD_{15}. *Address/Data* (bidirectional, active High, 3-state). These 16 multiplexed address and data lines carry I/O addresses, memory addresses, and data during bus transactions.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of Address Strobe indicates the beginning of a transaction and shows that the address, status, R/$\overline{W}$, and B/$\overline{W}$ signals are valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). A Low on this line indicates that the CPU has relinquished control of the bus in response to a bus request.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). A Low on this line indicates that an external bus requester has obtained or is trying to obtain control of the bus.

B/$\overline{W}$. *Byte/Word* (output, Low = Word, 3-state). This signal indicates whether a byte or a word of data is to be transmitted during a transaction.

CLK. *Clock Output* (output). The frequency of the processor timing clock is derived from the oscillator input (external oscillator) or crystal frequency (internal oscillator) by dividing the crystal or external oscillator input by two. The processor clock is further divided by one, two, or four (as programmed), and then output on this line.

CTIN. *Counter/Timer Input* (input, active High). These lines receive signals from external devices for the counter/timers.

CTIO. *Counter/Timer I/O* (bidirectional, active High, 3-state). These I/O lines transfer signals between the counter/timers and external devices.

$\overline{DMASTB}$. *DMA Flyby Strobe* (output, active Low). These lines select peripheral devices for DMA flyby transfers.

$\overline{DS}$. *Data Strobe* (output, active Low, 3-state). This signal provides timing for data movement to or from the bus master.

$\overline{EOP}$. *End of Process* (input, active Low). An external source can terminate a DMA operation in progress by driving $\overline{EOP}$ Low. $\overline{EOP}$ always applies to the active channel; if no channel is active, $\overline{EOP}$ is ignored.

$\overline{GACK}$. *Global Acknowledge* (input, active Low). A Low on this line indicates the CPU has been granted control of a global bus.

$\overline{GREQ}$. *Global Request* (output, active Low, 3-state). A Low on this line indicates the CPU has obtained or is trying to obtain control of a global bus.

$\overline{IE}$. *Input Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is toward the CPU.

$\overline{INT}$. *Maskable Interrupts* (input, active Low). A Low on these lines requests an interrupt.

$\overline{NMI}$. *Nonmaskable Interrupt* (input, falling-edge activated). A High-to Low transition on this line requests a nonmaskable interrupt.

$\overline{OE}$. *Output Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is away from the MPU.

OPT. *Bus Option* (input). This signal establishes the bus option during reset as follows:

OPT	Bus Interface
0	Z80-Bus, 8-bit
1	Z-BUS, 16-bit

$\overline{PAUSE}$. *CPU Pause* (input, active Low). While this line is Low the CPU refrains from transferring data to or from an Extended Processing Unit in the system or from beginning the execution of an instruction.

$\overline{RDY}$. *DMA Ready* (input, active Low). These lines are monitored by the DMA channels to determine when a peripheral device associated with a DMA channel is ready for a read or write operation. When a DMA channel is enabled to operate, its Ready line indirectly controls DMA activity; the manner in which DMA activity is controlled by the line varies with the operating mode (single-transaction, burst, or continuous).

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the CPU and on-chip peripherals.

R/$\overline{W}$. *Read/Write* (output, Low = Write, 3-state). This signal determines the direction of data transfer for memory, I/O, or EPU transfer transactions.

RxD. *UART Receive* (input, active High). This line receives serial data at standard TTL levels.

ST_0-ST_3. *Status* (output, active High, 3-state). These four lines indicate the type of transaction occurring on the bus and give additional information about the transaction.

TxD. *UART Transmit* (output, active High). This line transmits serial data at standard TTL levels.

$\overline{WAIT}$. *Wait* (input, active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

XTALI. *Clock/Crystal Input* (time-base input). Connects a parallel-resonant crystal or an external single-phase clock to the on-chip clock oscillator.

XTALO. *Crystal Output* (time-base output). Connects a parallel-resonant crystal to the on-chip clock oscillator.

+5V. *Power Supply Voltage.* (+5 nominal).

GND. *Ground.* Ground reference.

Bus Operations

Two kinds of operations can occur on the system bus: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the bus and is known as the bus master. A transaction is initiated by the bus master and is responded to by some other device on the bus. Only one transaction can proceed at a time; eight kinds of transactions can occur:

Burst Memory. These transactions are used to transfer four words of instructions from the memory to the CPU.

DMA Flyby. This transaction is used by the DMA peripheral to transfer data between an external peripheral and memory.

EPU Transfer. This transaction is used to transfer data between the CPU and an EPU.

Halt. This transaction is used to indicate that the CPU is entering the Halt state.

Interrupt Acknowledge. This transaction is used by the CPU to acknowledge an external interrupt request and to transfer additional information from the interrupting device.

I/O. This transaction is used by the bus master to transfer data to or from an external peripheral.

Memory. This transaction is used by the bus master to transfer data to or from a memory location.

Refresh. These transactions by the refresh mechanism do not transfer data; they refresh dynamic memory.

Only the bus master can initiate transactions. A request, however, can be initiated by a device that does not have control of the bus. Two types of requests can occur:

Bus. This request is used to request control of the bus to initiate transactions.

Interrupt. This request is used to request servicing by the CPU.

When an interrupt or bus request is made, it is answered according to its type: for an externally generated interrupt request, an Interrupt Acknowledge transaction is initiated by the CPU; for bus requests, the MPU enters Bus Disconnect state, relinquishes the bus, and activates an acknowledge signal.

Transactions

Data transfers to and from the Z280 MPU are accomplished through the use of transactions.

All transactions start with Address Strobe ($\overline{AS}$) being driven Low and then raised High by the Z280 MPU. On the rising edge of $\overline{AS}$, the Status lines ST_0-ST_3 are valid; these lines indicate the type of transaction being initiated (Table 9); seven types of transactions are discussed in the sections that follow. Associated with the status lines are two other lines that become valid at this time: R/$\overline{W}$, and B/$\overline{W}$.

Table 9. Status Code Table

Status Lines 3••0	Type of Transaction
0000	Reserved
0001	Refresh
0010	I/O transaction
0011	Halt
0100	Interrupt acknowledge line A
0101	$\overline{NMI}$ acknowledge
0110	Interrupt acknowledge line B
0111	Interrupt acknowledge line C
1000	Transfer between CPU and memory, cacheable
1001	Transfer between CPU and memory, non-cacheable
1010	Data transfer between EPU and memory
1011	Reserved
1100	EPU Instruction fetch, template, subsequent words
1101	EPU Instruction fetch, template, first word
1110	Data transfer between EPU and CPU
1111	Test and Set (data transfers)

If the transaction requires an address, it is valid on the rising edge of $\overline{AS}$. No address is required for EPU-CPU transfer transactions; the contents of the A and AD lines while $\overline{AS}$ is asserted are undefined. If an address is generated, the $\overline{OE}$ signal is also activated.

The Z-BUS MPUs use Data Strobe ($\overline{DS}$) to time the actual data transfer. (Note that Refresh and Halt transactions do not transfer any data and thus do not activate $\overline{DS}$.) For write operations (R/$\overline{W}$ = Low), a Low on $\overline{DS}$ indicates that valid data from the bus master is on the AD lines. The Output Enable continues to be asserted until $\overline{DS}$ is deasserted. For read operations (R/$\overline{W}$ = High), the bus master makes AD lines 3-state, deasserts $\overline{OE}$, and asserts $\overline{IE}$ after driving $\overline{DS}$ Low so that the addressed device can put its data on the bus. The bus master samples this data on the falling clock edge just before raising $\overline{DS}$ and $\overline{IE}$ High.

Wait Cycle. The $\overline{WAIT}$ line is sampled on the falling clock edge when data is sampled by the Z280 MPU (Read) or the falling clock edge before $\overline{DS}$ rises (Read or Write). If $\overline{WAIT}$ is Low, another cycle is added to the transaction before data is sampled or $\overline{DS}$ rises. In this added cycle, and all subsequent cycles added when $\overline{WAIT}$ is Low, $\overline{WAIT}$ is again sampled on the falling clock edge and, if it is Low, another cycle is added to the transaction. In this way, the transaction can be extended to an arbitrary length by external circuitry to accommodate (for example) slow memories or I/O devices that are not yet ready for data transfer. Automatic insertions of wait states by the CPU or on-chip DMA channels can be programmed by setting fields in the Bus Timing and Control register and Bus Timing and Initialization register to indicate the number to be inserted.

Memory Transactions. Memory transactions move data to or from memory when a bus master makes a memory access. Thus, they are generated during program execution to fetch instructions from memory and to fetch and store memory data. They are also generated to store old program status and fetch new program status during interrupt and trap handling and after reset.

A memory transaction is three bus cycles long unless extended when $\overline{WAIT}$ is asserted.

Bytes transferred to or from odd memory locations (address bit 0 = 1) are always transmitted on lines AD_0-AD_7 (bit 0 on AD_0). Bytes transferred to or from even memory locations (address bit 0 = 0) are always transmitted on lines AD_8-AD_{15} (bit 0 on AD_8). For byte reads (B/$\overline{W}$ High, R/$\overline{W}$ High), the CPU or on-chip DMA channel uses only the byte whose address it put out on the bus. For byte writes (B/$\overline{W}$ High, R/$\overline{W}$ Low), the memory should store only the byte whose address was output. During byte memory writes, the CPU (or on-chip DMA channel in non-Flyby transactions) places the same byte on both halves of the bus, and the proper byte must be selected by testing A_0. For word transfers (B/$\overline{W}$ = Low), all 16 bits are captured by the CPU or DMA channel (Read: R/$\overline{W}$ = High) or stored by the memory (Write: R/$\overline{W}$ = Low). For these transactions (either memory or I/O) the bytes of data appear swapped on the bus with the most significant byte on AD_7-AD_0 and the least significant byte on AD_{15}-AD_8. A word is aligned if the address is even; otherwise it is unaligned.

Memory transaction timings are shown in Figures 46-50.

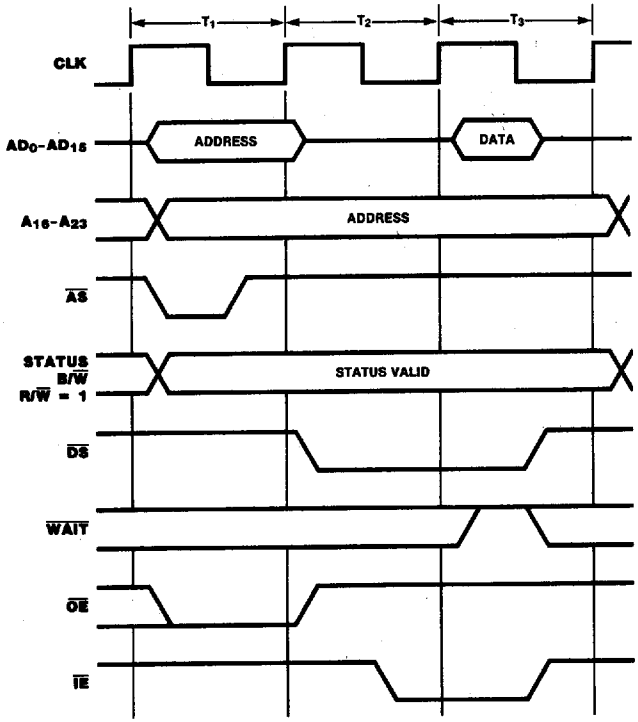


Figure 46. Memory Read Timing

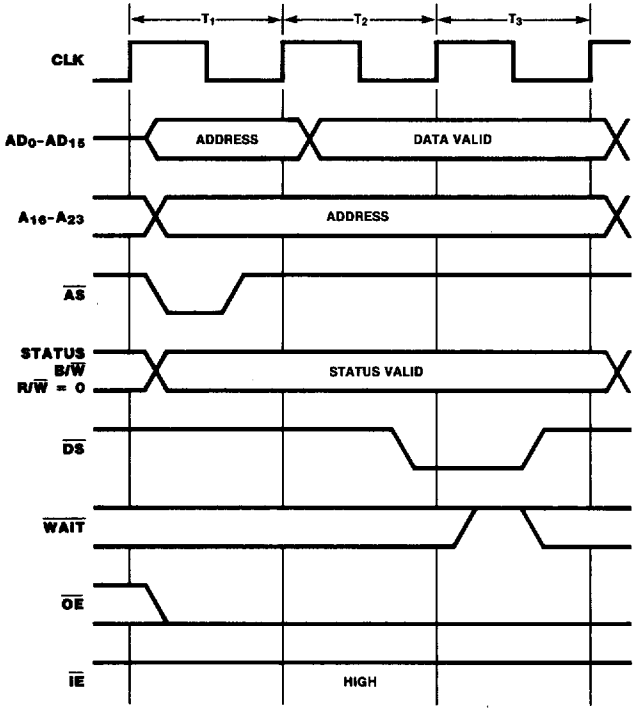


Figure 47. Memory Write Timing

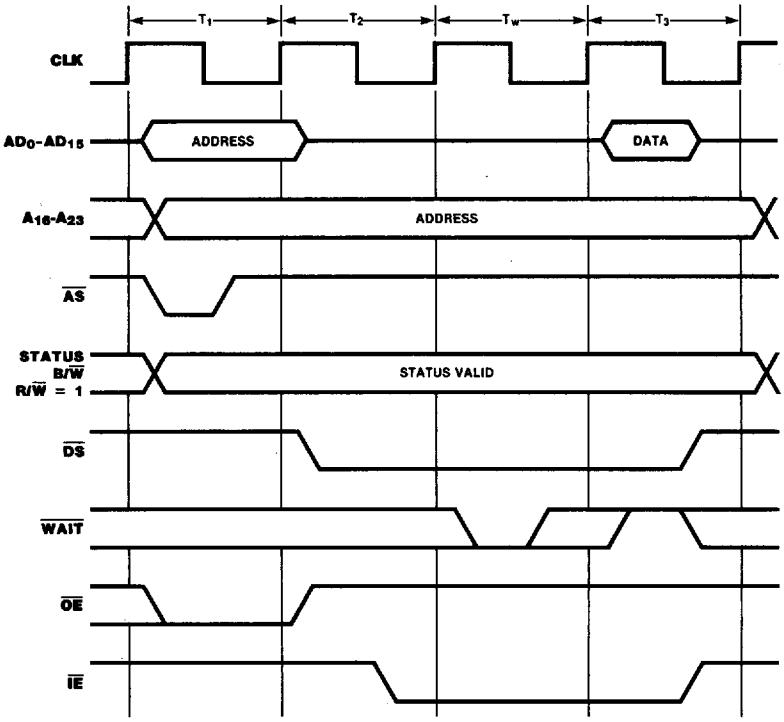


Figure 48. Memory Read Timing with External Wait Cycle

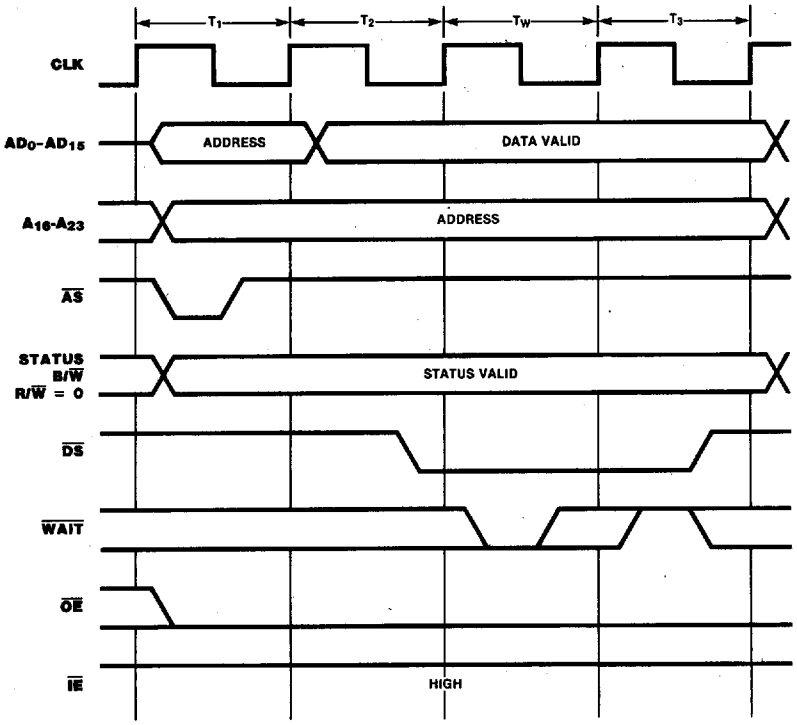


Figure 49. Memory Write Timing with External Wait Cycle

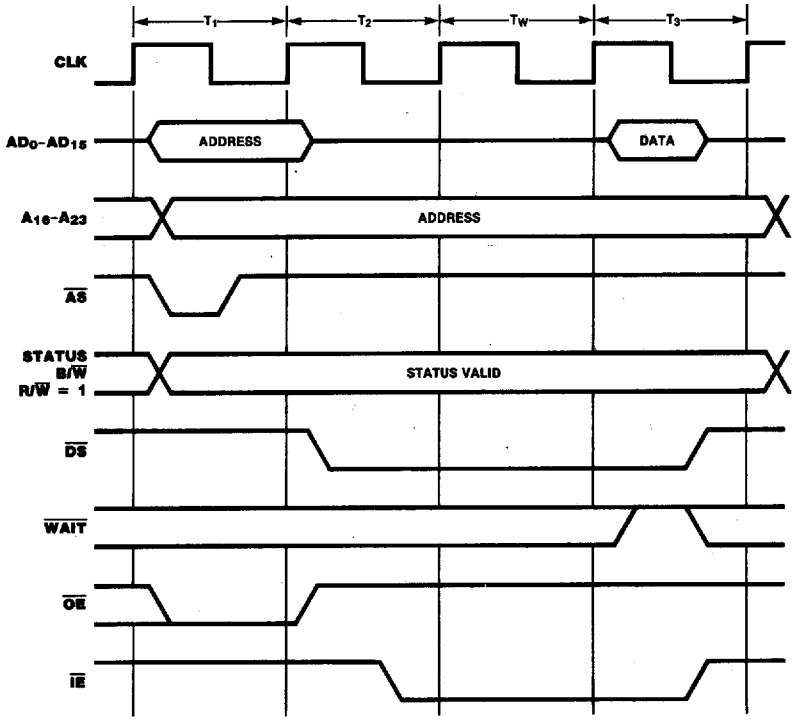


Figure 50. Memory Read Timing with Internal Wait Cycle

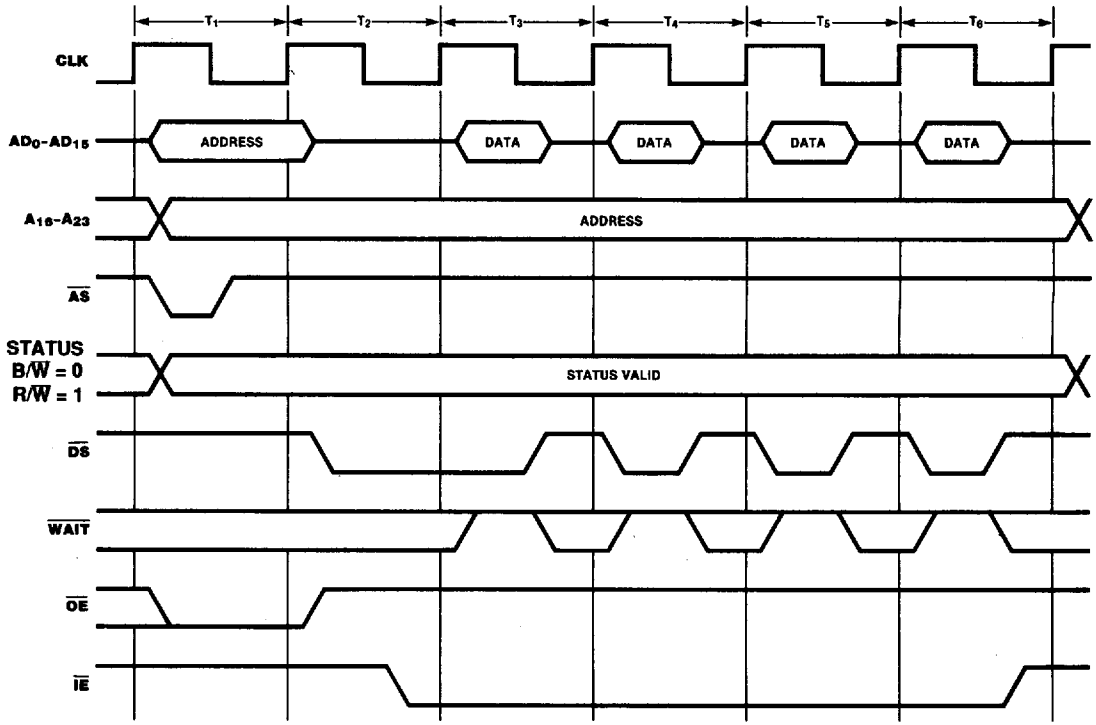


Figure 51. Burst Memory Read Timing

Burst Memory Transactions. Burst memory transactions use multiple Data Strobes associated with a single Address Strobe. The CPU uses burst transactions to read four consecutive words in four data transactions. The address of the first word read during a burst transaction has zeros in the three least significant bits. Control bits in the Cache Control register indicate whether or not portions of the memory system can support burst transactions.

The CPU uses burst mode reads only for fetching instructions. If an instruction is to be fetched from a location within a half of physical memory that supports burst transactions, the CPU reads the eight bytes that contain the first byte of the instruction. (EPA template fetches do not use the burst transaction.)

Timing for the first data transfer during a burst transaction is identical to that for a single memory read, including the automatic insertion of wait states, except there are four T_3 states. Subsequent data transfers do not include automatic wait states. On the first data transfer, if $\overline{WAIT}$ is sampled active then it is sampled again every bus clock cycle until it is inactive, at which time the data is read from the bus. Burst memory read timing is shown in Figure 51.

Note: Burst Transactions can occur only in Z-BUS mode.

Halt Transactions. Halt transactions do not transfer data. They look like a memory transaction, except that $\overline{DS}$ remains High and no data is transferred.

A Halt transaction (Figure 52) is generated when the CPU executes a HALT instruction or when a fatal sequence of traps and bus errors occurs. The address placed on the AD lines is the location of the Halt instruction or the instruction that initiated the fatal sequence of traps and errors. The Status lines indicate a Halt transaction (0011).

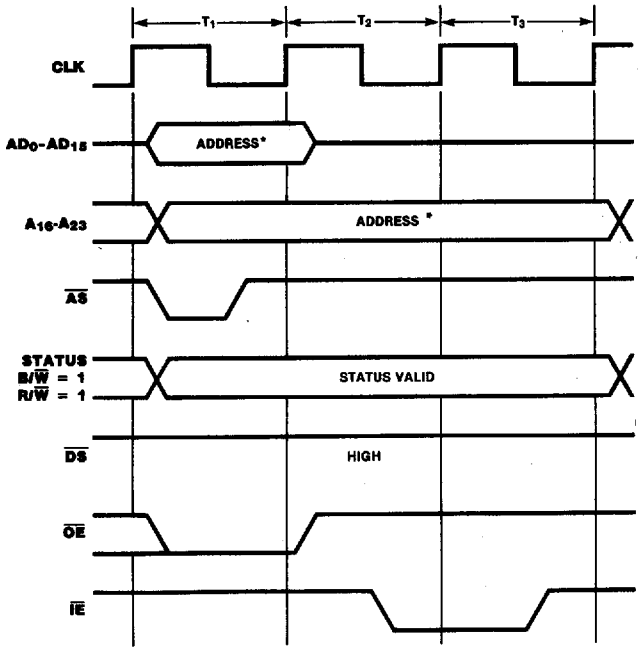
$\overline{WAIT}$ is not sampled during the Halt transaction.

I/O Transactions. I/O transactions (Figures 53 and 54) move data to or from peripherals and are generated during the execution of I/O instructions. I/O transactions to on-chip peripheral devices (I/O pages FE_H and FF_H) do not generate external bus transactions.

I/O transactions are four bus cycles long at a minimum, and they can be lengthened by the addition of wait cycles either automatically generated as indicated in the Bus Timing and Control register or generated by an external device. The extra clock cycles allow for slower peripheral operation.

The status lines indicate that the access is an I/O transaction (0010). The I/O address is found on AD_0-AD_{15} and A_{16}-A_{23}.

Byte data ($B/\overline{W}$ = High) is transmitted on AD_0-AD_7. This allows peripheral devices to attach to only eight of the AD lines. Word data ($B/\overline{W}$ = Low) is transmitted with the most significant byte on AD_0-AD_7 and the least significant byte on AD_8-AD_{15}.



*Address of Halt Instruction.

Figure 52. Halt Timing

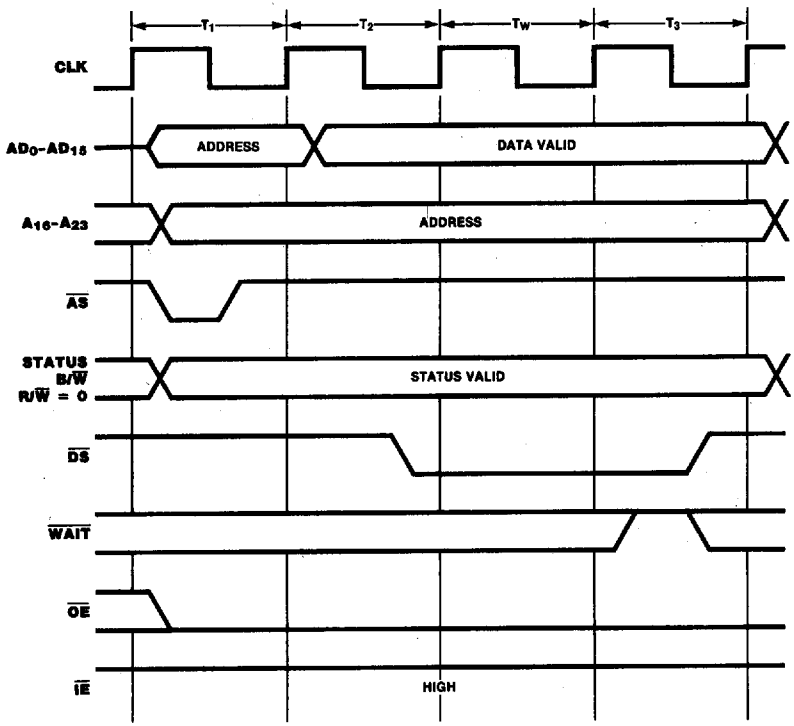


Figure 53. I/O Write Timing

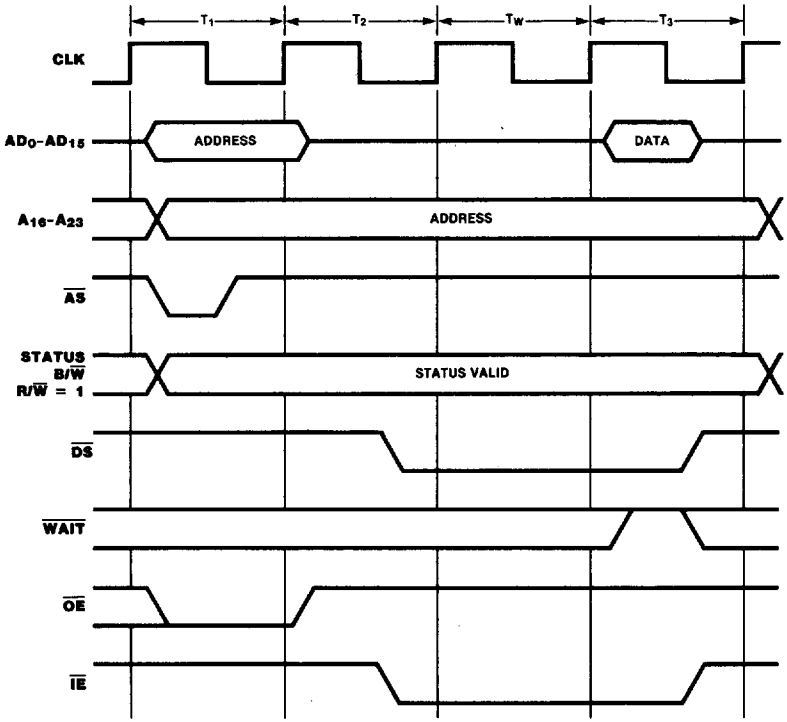


Figure 54. I/O Read Timing

Interrupt Acknowledge Transactions. These transactions (Figure 55) acknowledge an interrupt and read an identifier from the device that generated the interrupt. Interrupt Acknowledge transactions are generated automatically by the hardware when an external interrupt is detected.

These transactions are five cycles long at a minimum, with at least two automatic Wait cycles, although others can be added by programming the Bus Timing and Control register. The Wait cycles are used to give the interrupt priority daisy chain (or other priority resolution device) time to settle before the identifier is read.

The only item of data transferred is the identifier that is captured from the AD lines on the falling clock edge just before $\overline{DS}$ is raised High. The length of time that $\overline{DS}$ is asserted is identical with I/O timing programmed in the Bus Timing and Control register.

There are two places where $\overline{WAIT}$ is sampled and thus a Wait cycle can be inserted by external devices. The first place serves to delay the falling edge of $\overline{DS}$ to allow the daisy chain a longer time to settle, and the second place serves to delay the point at which data is read.

Refresh Transactions. A memory Refresh transaction (Figure 56) is generated by the refresh mechanism and can come immediately after the final clock cycle of any other transaction. The memory refresh counter's 10-bit address is output on the low order 10 bits of the bus during the first cycle of the transaction. The contents of the rest of the bus are undefined. The Status lines indicate Refresh (0001). This transaction can be used to generate refreshes for dynamic RAMs. Refreshes may occur while the CPU is in the Halt or Fatal state.

CPU-Extended Processing Unit Interaction

The Z280 CPU with a Z-BUS interface and $\overline{PAUSE}$ input line and one or more Extended Processing Units (EPUs) work together like a single CPU component, with the CPU providing address, status, and timing signals and the EPU supplying and capturing data. The EPU monitors the status and timing signals output by the CPU so that it knows when to participate in a memory transaction; for EPU to memory transfers, the CPU puts its AD lines in 3-state while $\overline{DS}$ is Low, so that the EPU can use them.

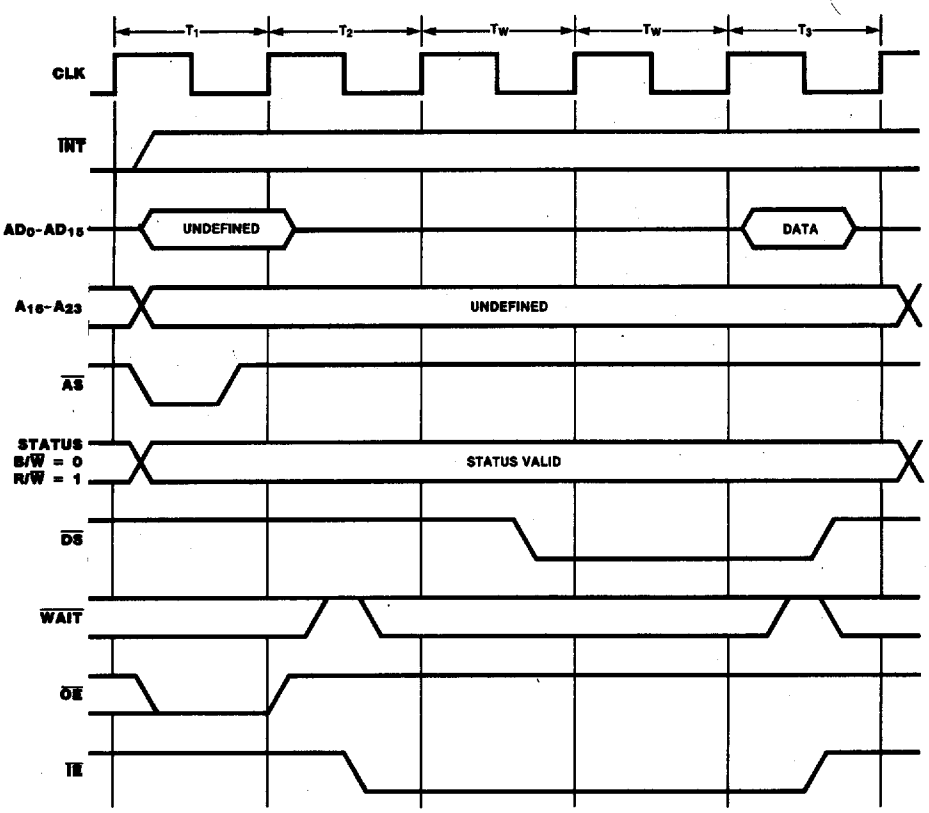
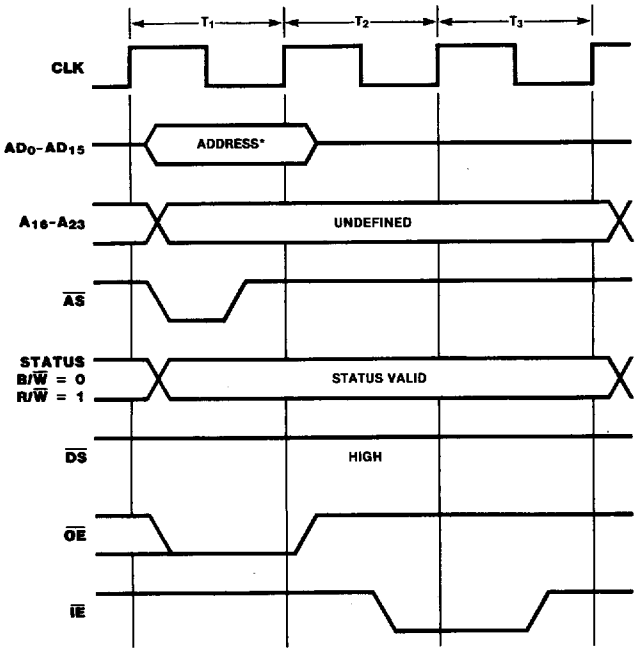


Figure 55. Interrupt Acknowledge Timing



*10 least-significant bits are Refresh address.

Figure 56. Memory Refresh Timing

In order to know which transaction it is to participate in, the EPU must track the following sequence of events:

- When the CPU fetches the first word of an EPA instruction template from memory (ST_3-ST_0 = 1101), the EPU must also capture the instruction returned by the memory. Within the template is an ID field that indicates whether or not the EPU is to execute the instruction.
- The next non-refresh transaction by the CPU is the fetch of the second word of the instruction (ST_3-ST_0 = 1100). The EPU must also capture this word. If the template is not aligned, a third fetch is made (ST_3-ST_0 = 1100).
- If the instruction involves a read or write to memory, then transfers of data between memory and the EPU (ST_3-ST_0 = 1010) are the next non-refresh transactions performed by the CPU. The EPU must supply the data (Write: R/$\overline{W}$ = Low) or capture the data (Read: R/$\overline{W}$ = High) for each transaction, just as if it were part of the CPU. In both cases, the CPU 3-states its AD lines while data is being transferred ($\overline{DS}$ Low).
- If the instruction involves a transfer from the EPU to the Z280 MPU, the next non-refresh transaction is the CPU transferring data between the EPU and CPU (ST_3-ST_0 = 1110).

In order to follow this sequence, an EPU has to monitor the status lines to verify that the transaction it is monitoring on the bus was generated by the CPU. In a multiple EPU system, there is no indication on the bus as to which EPU is cooperating with the CPU at any given time. This must be determined by the EPUs from the templates they capture.

When an EPU begins to execute an extended instruction, the CPU can continue fetching and executing instructions. If the EPU wishes to halt the CPU from executing another instruction or bus transaction, the EPU must activate the $\overline{PAUSE}$ line to stop the CPU until the EPU is ready for subsequent MPU activity. This mechanism is used to synchronize MPU-EPU activity.

EPU Transfer Transactions. These transactions (Figures 57-59) allow the CPU to transfer data to or from an EPU or to read or write an EPU's status registers. They are generated during the execution of the EPU instructions.

EPU-to-Memory transfers are five cycles unless extended by $\overline{WAIT}$. Memory-to-EPU transfers are three cycles unless extended by $\overline{WAIT}$.

EPU-CPU transfer transactions have the same form as I/O transactions and thus are four clock cycles long, unless extended by $\overline{WAIT}$. Although $\overline{AS}$ is asserted, no address is generated and the contents of the bus are undefined; only one status code is used (1110).

In a multiple EPU system, the EPU that is to participate in a transaction is selected implicitly by the ID code in the EPU template, rather than by an address. The Read/Write line ($R/\overline{W}$ = High) indicates the direction of the data transfer into the CPU.

Requests

The Z280 MPU supports three types of request signals. These are:

- Interrupt requests, which another device initiates and the CPU accepts and acknowledges.
- Bus requests, which an external potential bus master initiates and the CPU accepts and acknowledges.
- Global bus requests, which the CPU or on-chip DMA initiates to acquire a global system bus.

When a request is made, it is answered according to its type: for interrupt requests, an Interrupt Acknowledge transaction is initiated by the CPU; for bus requests, an acknowledge signal is sent; for global bus request, an acknowledge signal is received.

Interrupt Requests. The Z280 MPU supports two types of external interrupts, maskable and nonmaskable ($\overline{NMI}$). The Interrupt Request line of a device that is capable of generating an interrupt may be tied to any of the interrupt pins. Several devices can be connected to one pin, with the devices arranged in a priority daisy chain. The CPU uses the same protocol for handling requests on these pins. The sequence of events is given below:

Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge-detected, and the internal $\overline{NMI}$ latch is set. At the beginning of the last processor clock cycle of any instruction, the interrupt inputs are sampled along with the state of the internal $\overline{NMI}$ latch.

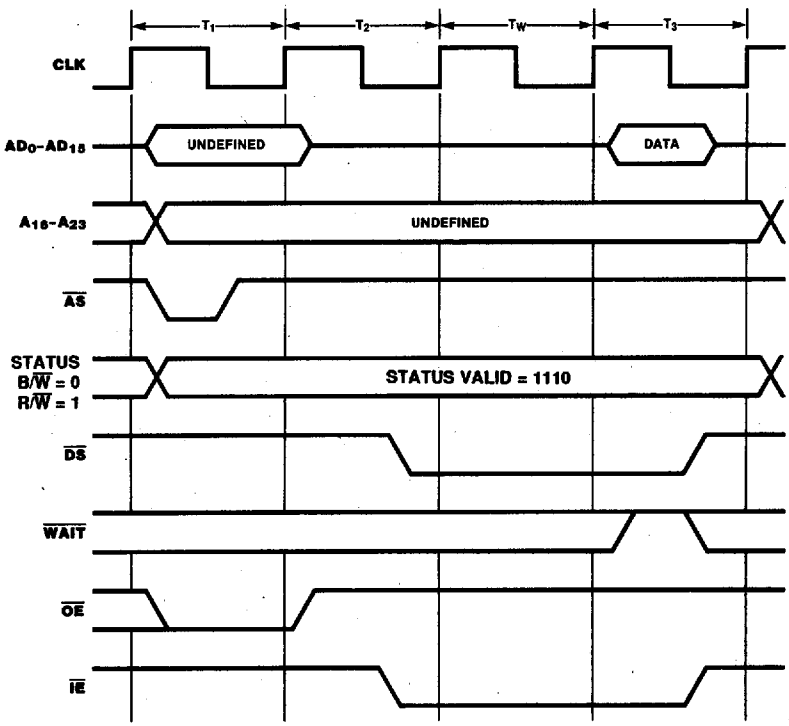


Figure 57. EPU to CPU Timing

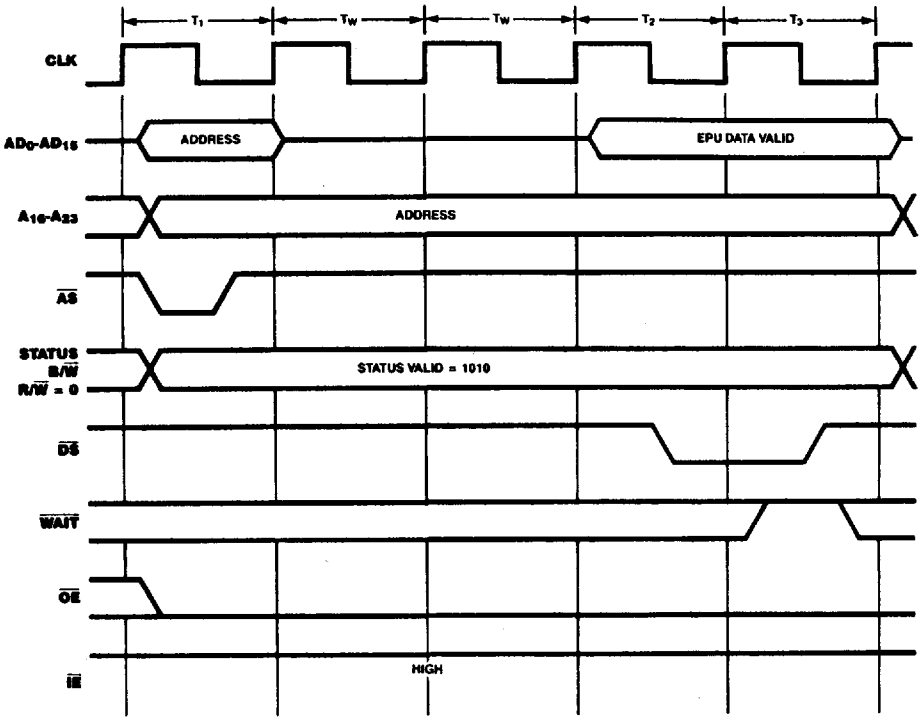


Figure 58. EPU Write to Memory

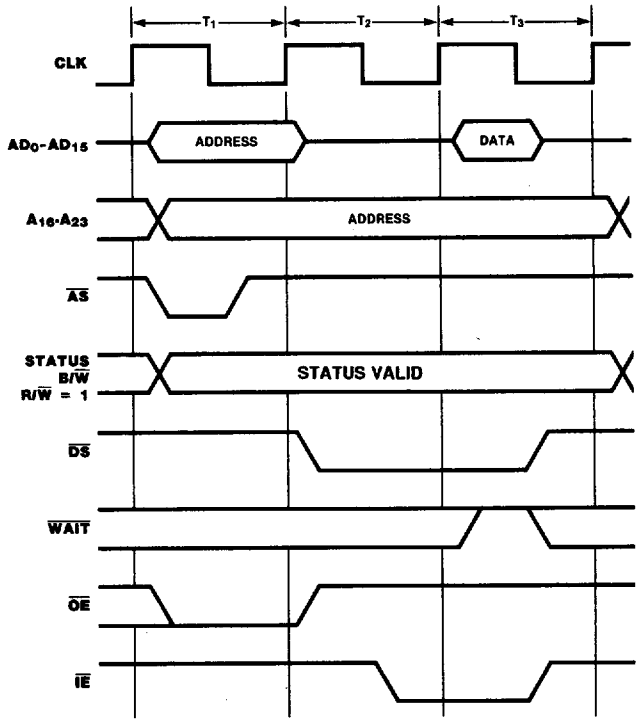


Figure 59. Memory to EPU Timing

If a maskable interrupt is requested and the Master Status register indicates that requests on that line are to be accepted, or if the $\overline{NMI}$ latch is set, the next possible bus transaction is an interrupt acknowledge transaction that results in an identifier from the highest-priority interrupting device being read off the AD lines. This data is used as specified by the current interrupt mode.

Bus Requests. To generate transactions on the bus, a potential external bus master (such as a DMA Controller) must gain control of the bus by making a bus request. A bus request is initiated by pulling $\overline{BUSREQ}$ Low. Several bus requesters can be wired-OR to the $\overline{BUSREQ}$ pin; priorities are resolved externally to the CPU, usually by a priority daisy chain.

The asynchronous $\overline{BUSREQ}$ signal generates an internal $\overline{BUSREQ}$, which is synchronous. If the external $\overline{BUSREQ}$ is Low at the beginning of any processor clock cycle, the internal $\overline{BUSREQ}$ will cause the bus acknowledge line ($\overline{BUSACK}$) to be asserted after the current bus transaction is completed or after the write transaction of a TSET instruction. The CPU then enters Bus Disconnect state and gives up control of the bus. All Z280 Output pins except $\overline{BUSACK}$ are 3-stated.

The on-chip DMA channels have higher priority than the off-chip devices requesting the external bus via $\overline{BUSREQ}$.

RESET

A hardware reset puts the Z280 MPU into a known state and optionally initializes the Bus Timing and Initialization control register of the Z280 MPU to a system specifiable value. A reset begins at the end of any processor clock cycle if the $\overline{RESET}$ line is Low. However, if a bus transaction is in progress it is allowed to be completed. A system reset overrides all other operations of the chip, including interrupts, traps and bus requests. A reset should be used to initialize a system as part of the power-up sequence.

The $\overline{RESET}$ input must be asserted for a minimum of 128 processor clock cycles. Within this time the Z280 lines assume their reset values. For either bus, the AD lines are 3-stated, and all control outputs are forced High. While $\overline{RESET}$ is asserted, the CLK output is the processor clock frequency scaled by four.

The $\overline{RESET}$ line is sampled on the rising edge of an internal clock (derivative of XTALi). When the $\overline{RESET}$ line is sampled High (de-asserted), the state of the $\overline{WAIT}$ line is also noted: if $\overline{WAIT}$ is asserted, then the contents of the AD lines are used to program the Bus Timing and Initialization register, otherwise the constant 00 hexadecimal is used. If the hardware programming initialization option is used, AD4 must be 0 when the bus is sampled and the AD6 line determines whether the UART bootstrap option is selected.

After reset, the Z280 MPU is initialized as shown in Tables 10 and 11.

The following registers are unaffected:

- CPU register file, including user Stack Pointer
- Page Descriptor registers
- Interrupt/Trap Vector Table Pointer register

On the rising edge of $\overline{RESET}$, if Bus Request is asserted the Z280 MPU will grant the bus before fetching the first instruction from location 0.

After $\overline{RESET}$ has returned to High, the CPU begins to operate unless the Bootstrap UART feature is utilized.

Table 10. Effect of a Reset on Z280 CPU and MMU Registers

Register	Value Loaded on Reset (Hexadecimal)	Comments
Program Counter	0000	
System Stack Pointer	0000	
I	00	
R	00	
Master Status	0000	System mode, Single-Step disabled, Breakpoint-on-Halt disabled All maskable interrupts disabled
Bus Timing and Control	00**	No automatic wait states for I/O, upper 8M bytes of memory, or interrupt acknowledges
Bus Timing and Initialization	00	CLK output 2x processor clock period, no automatic wait states for lower 8 Mbytes of memory, bootstrap mode disabled, direct clock option disabled, multiprocessor configuration disabled
I/O Page	00	I/O Page 0 in use
Cache Control	20	Cache enabled for instructions All valid bits cleared to 0 Burst mode disabled
Trap Control	00	EPA trap enabled, I/O not privileged, System Stack Overflow Warning trap disabled
System Stack Limit	0000**	
Local Address	00	All memory transactions are made to local bus
Interrupt Status	00xx	Interrupt mode 0, nonvectored interrupts, current state of interrupt requests (indicated by xx)
Interrupt/Trap Vector Table Pointer		Unaffected
CPU Registers AF, BC, DE, HL, IX, IY, AF′, BC′, DE′, DE′, HL		Unaffected
User Stack Pointer		Unaffected
MMU Master Control	0000**	MMU disabled
MMU Page Descriptor Register, Page Descriptor Register Pointer		Unaffected

Table 11. Effect of a Reset on Z280 On-Chip Peripheral Registers

Register	Value Loaded on Reset (Hexadecimal)	Comments
Refresh	88	Refresh enabled, rate = 32
Counter/Timers:		
Configuration	00	Timer mode, single-cycle, non-retrigger
Command/Status	00	Timer disabled
DMA Channels:		
Master Control	0000***	No DMA linking, EOP disabled, Software Ready disabled
DMA0 Transaction Descriptor	0100*	DMA0 disabled, continuous mode
DMA1/2/3 Transaction Descriptor	—	EN, IE, TC, and EPS fields cleared, other fields unaffected
DMA0 Destination Address	000000	
DMA0 Count	0100	
UART:		
Configuration	00*	5 bits/character, parity disabled, external clock, x1 clock rate, loop back disabled
Transmitter Control/Status	01	Transmitter disabled, transmit buffer empty
Receiver Control/Status	00*	Receiver disabled

*Unless bootstrap mode is selected. **Reserved bits are undefined on reads.

ABSOLUTE MAXIMUM RATINGS

Voltage on Vcc with respect to Vss ..... -0.3V to + 7V
Voltages on all pins with respect to Vss ..... -0.3V to (Vcc + 0.3V)
Operating Ambient
Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . – 65°C to + 150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

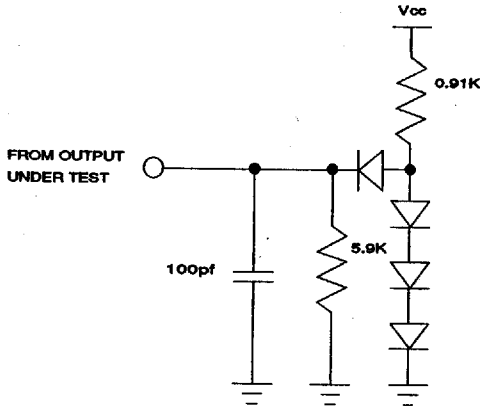
STANDARD TEST CONDITIONS

The DC Characteristics and Capacitance sections below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

Available operating temperature ranges are:

- S = 0°C to + 70°C

All ac parameters assume a load capacitance of 100pf. Add 10 ns delay for each 50 pf increase in load up to a maximum of 200 pf for the data bus.



DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Votage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4.0 ma
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 µa
V_{CC}	Operating Power Supply Voltage	4.5	5.5	V	
I_{CC}	Power Supply Current		200	ma	V_{CC} = 5.5 V XTALI = 20 MHz V_{IH} = 2.0 V V_{IL} = 0.8 V Outputs Unloaded

Z280 AC CHARACTERISTICS

Z-Bus Timing (Refer to Figures 60 and 61)

No	Symbol	Parameter	10 MHz Min	10 MHz Max	12.5 MHz Min	12.5 MHz Max	Unit	Notes
1	TdCr(ST)	Clock rise to Status Delay		20		15	nS	
2	TdCr(A)	Clock rise to Address Valid Delay		20		15	nS	
3	TdCr(ASf)	Clock rise to /AS fall Delay		20		15	nS	
4	TdCf(ASr)	Clock fall to /AS rise Delay		20		15	nS	
5	TwAS	/AS Low Width	nTcXT-20		nTcXT-20		nS	1
6	TdCr(AZ)	Clock rise to Address Float Delay		25		25	nS	
7	TdCr(DSf)	Clock rise to /DS fall Delay		20		15	nS	
8	TdCf(DSr)	Clock Fall to /DS rise Delay		35		25	nS	
9	TsD(Cf)	Data to Clock fall Setup	30		30		nS	
10	ThD(Cf)	Data from Clock fall Hold	10		10		nS	
11	TdCf(DSf)	Clock fall to /DS fall Delay		20		15	nS	
12	TdCr(D)	Clock rise to Data Valid Delay		20		15	nS	
13	TdDSr(Dx)	/DS rise Data not Valid Delay	nTcXT-40		nTcXT-30		nS	1
14	TsW(Cf)	/Wait to Clock fall Setup	50		40		nS	
15	ThW(Cf)	/Wait from Clock fall Hold	0		0		nS	
16	TdCr(OEf)	Clock rise to /OE fall Delay		20		15	nS	
17	TdCr(OEr)	Clock rise to /OE rise Delay		20		15	nS	
18	TdCf(IEf)	Clock fall /IE fall Delay		20		15	nS	
19	TdCf(IEr)	Clock fall to /IE rise Delay		35		25	nS	
20	TdA(ASr)	Address Valid to /AS rise Delay	nTcXT-25		nTcXT-20		nS	1
21	TdDSr(ASf)	/DS rise to /AS fall Delay	nTcXT-40		nTcXT-25		nS	1
22	TdASr(Ax)	/AS rise to Address not Valid Delay	nTcXT-30		nTcXT-25		nS	1
24	TdDSr(A)	/DS rise to Address Active Delay	nTcXT-40		nTcXT-30		nS	1
25	TdAz(DSf)	Address Float to /DS fall Delay	0		0			
26	TdD(DSf)	Data Valid to /DS fall Delay	nTcXT-20		nTcXT-20		nS	1
27	TwDSBh	/DS High Width(Burst Mode)	nTcXT-40		nTcXT-30		nS	1
28	TwDSbl	/DS Low Width(Burst Mode)	ntcXT-30		nTcXT-20		nS	1

Notes:

1. TcXT = XTALi Cycle Time
   Clk = 1x(1x bus clock): n=1
   2x(2x bus clock): n=2
   4x(4x bus clock): n=4

† Units in nanoseconds unless otherwise specified.
V_{IH} = 2.0V, V_{IL} = 0.8V, V_{OH} = 2.0V, V_{OL} = 0.8V

Z280 AC CHARACTERISTICS

Z80-Bus Timing (Refer to Figures 62 and 63)

No	Symbol	Parameter	10 MHz Min	10 MHz Max	12.5 MHz Min	12.5 MHz Max	Unit	Notes
1	TdCr(OEf)	Clock rise to /OE fall delay		20		15	nS	
2	TdCr(A)	Clock rise to Address Valid Delay		20		15	nS	
3	TdCr(ASf)	Clock rise to /AS fall Delay		20		15	nS	
4	TdCf(ASr)	Clock fall to /AS rise Delay		20		15	nS	
5	TwAS	/AS Low Width	nTcXT-20		nTcXT-20		nS	1
6	TdCr(AZ)	Clock rise to Address Float Delay		25		25	nS	
7	TsW(Cf)	/Wait to Clock fall setup	50		40		nS	
8	ThW(Cf)	/Wait from Clock fall hold	0		0			
9	TdA(ASr)	Address Valid to /AS rise delay	nTcXT-25		nTcXT-20		nS	1
10	TdASr(Ax)	/AS rise to Address not Valid Delay	nTcXT-30		nTcXT-25		nS	1
11	TdCr(RDf)	Clock rise to /RD fall delay		20		15	nS	
12	TdCf(RDr)	Clock fall to /RD rise Delay		35		25	nS	
14	TsD(Cf)	Data to Clock fall Setup	30		30		nS	
15	ThD(Cf)	Data from Clock fall Hold	10		10		nS	
16	TdAz(RDf)	Address Float to /RD fall Delay	0		0		nS	
19	TdCr(OEr)	Clock rise to /OE rise Delay		20		15	nS	
20	TdCf(IEf)	Clock fall to /IE fall Delay		20		15	nS	
21	TdCf(IEr)	Clock fall to /IE rise Delay		35		25	nS	
22	TdCr(IEr)	Clock rise to /IE rise Delay		20		15	nS	2
23	TdCr(RDr)	Clock rise to /RD rise Delay		20		15	nS	2
24	TdCf(WRf)	Clock fall to /WR fall Delay		20		15	nS	
25	TdCf(WRr)	Clock fall to /WR rise Delay		35		25	nS	
26	TdWRr(ASf)	/WR rise to /AS fall Delay	nTcXT-40		nTcXT-30		nS	1
27	TdWRr(A)	/WR rise to Address active Delay	nTcXT-40		nTcXT-30		nS	1
28	TdCr(D)	Clock rise to Data Valid Delay		20		15	nS	
29	TdWRr(Dx)	/WR rise to Data not Valid Delay	nTcXT-40		nTcXT-30		nS	1
30	TdD(WRf)	Data Valid to /WR fall Delay	nTcXT-20		nTcXT-20		nS	1
31	TdCf(MREQf)	Clock fall to /MREQ fall Delay		20		15	nS	
32	TdCf(MREQr)	Clock fall to /MREQ rise Delay		35		25	nS	
33	TdCr(MREQr)	Clock rise to /MREQ rise Delay		20		15	nS	2
34	TdCr(IORQf)	Clock rise to /IORQ fall Delay		20		15	nS	
35	TdCf(IORQr)	Clock fall to /IORQ rise Delay		35		25	nS	
36	TdCf(IORQf)	Clock fall to /IORQ fall Delay		20		15	nS	3
37	TdCf(M1r)	Clock fall to /M1 rise Delay		35		25	nS	3
38	TdCr(M1r)	Clock rise to /M1 rise Delay		20		15	nS	
39	TdCr(M1f)	Clock rise to /M1 fall Delay		20		15	nS	2
40	TdCf(RFSHr)	Clock fall to /RFSH rise Delay		35		25	nS	2,3
41	TdCf(RFSHf)	Clock fall to /RFSH fall Delay		20		15	nS	
42	TdCf(HALTf)	Clock fall to /HALT fall Delay		20		15	nS	

Notes:

1. TcXT = XTALi Cycle Time
   Clk = 1x(1x bus clock) : n=1
   2x(2x bus clock) : n=2
   4x(4x bus clock) : n=4
2. This parameter is used for RETI (Return From Interrupt).
3. This parameter is used for Interrupt Acknowledge.

† Units in nanoseconds unless otherwise specified.
V_{IH}=2.0V, V_{IL}=0.8V, V_{OH}=2.0V, V_{OL}=0.8V

Z280 AC CHARACTERISTICS

Z-Bus, Z80 Bus Common Signals and Peripherals Timing
(Refer to Figures 64 through 71)

No	Symbol	Parameter	10 MHz Min	10 MHz Max	12.5 MHz Min	12.5 MHz Max	Unit	Notes
1	TcXT	XTALi Cycle time	50	tbd	40	tbd	nS	
2	TwXTh	XTALi High Width	15		15		nS	
3	TwXTl	XTALi Low Width	15		15		nS	
4	TrXT	XTALi Rise Time		10		10	nS	
5	TfXT	XTALi Fall Time		10		10	nS	
6	TdXTf(C)	XTAL fall to Clock Delay		40		40	nS	
7	TrC	Clock rise time		12		10	nS	
8	TfC	Clock fall time		12		10	nS	
9	TdCr(CSf)	Clock rise to /DS, /RD, or /WR fall Delay		20		15	nS	
10	TdCr(CSr)	Clock rise to /DS, or /WR rise Delay		20		15	nS	
11	TdCr(STBf)	Clock rise to /DMASTB fall delay		20		15	nS	
12	TdCf(STBr)	Clock fall to /DMASTB rise Delay		35		25	nS	
13	TdCr(STBr)	Clock rise to /DMASTB rise Delay		20		15	nS	
14	TdCf(CSr)	Clock fall to /DS or/RD Rise Delay		35		25	nS	
15	TdCf(GREQf)	Clock fall to /GREQ Fall Delay		35		25	nS	
16	TdCf(GREQr)	Clock fall to /GREQ rise Delay		35		25	nS	
17	TdCr(BUSACKf)	Clock rise to /BUSACK fall Delay		20		15	nS	
18	TdCr(BUSACKr)	Clock rise to /BUSACK rise Delay		20		15	nS	
19	TcCTIN	CTIN Cycle Time	10TcXT		10TcXT		nS	
20	TwCTINh	CTIN High Width	4TcXT		4TcXT		nS	
21	TwCTINl	CTIN Low Width	4TcXT		4TcXT		nS	
22	TwCTIOh	CTIO High Width	4TcXT		4TcXT		nS	1
23	TwCTIOl	CTIO Low Width	4TcXT		4TcXT		nS	1
24	TdCTIN(CTIO)	CTIN to CTIO Delay	20TcXT	28TcXT	20TcXT	28TcXT	nS	2
25	TdCf(TD)	Baud Clock fall to Transmit Data Delay		70		70	nS	3
26	TsRD(Cr)	Receive Data to Baud Clock rise Setup	10		10		nS	3
27	ThRD(Cr)	Receive Data from Baud Clock rise Hold	50		50		nS	3
28	TrRESET	/Reset Rise Time		10		10	nS	
29	TfRESET	/Reset Fall Time		10		10	nS	
30	TsWAITf(RESETr)	/WAIT fall to /RESET rise Setup	4TcXT		4TcXT		nS	4
31	ThWAITr(RESETr)	/WAIT rise to /RESET rise Hold	6TcXT		6TcXT		nS	4
32	TsD(RESETr)	Data to /RESET rise Setup	0		0		nS	4
33	ThD(RESETr)	Data from /RESET rise Hold	6TcXT		6TcXT		nS	4
34	TrIN	Input Rise Time		20		20	nS	5
35	TfIN	Input Fall Time		20		20	nS	5
36	TwNMI	/NMI Low Width	4TcXT		4TcXT		nS	

Notes:

1. CTIO as Gate or Trigger Input.
2. CTIO as Output,when CTIN causes terminal count.
3. CTIN1 as X1 Baud Clock Input. Refer to specs 20 and 21 for pulse widths.
4. To program Bus Timing and Initialization Register at reset.
5. Inputs AD, /BUSREQ, CTIN, CTIO, /INT, /NMI, /RDY, RxD, /PAUSE and /WAIT.

† Units in nanoseconds unless otherwise specified.
V_{IH}=2.0V, V_{IL}=0.8V, V_{OH}=2.0V, V_{OL}=0.8V

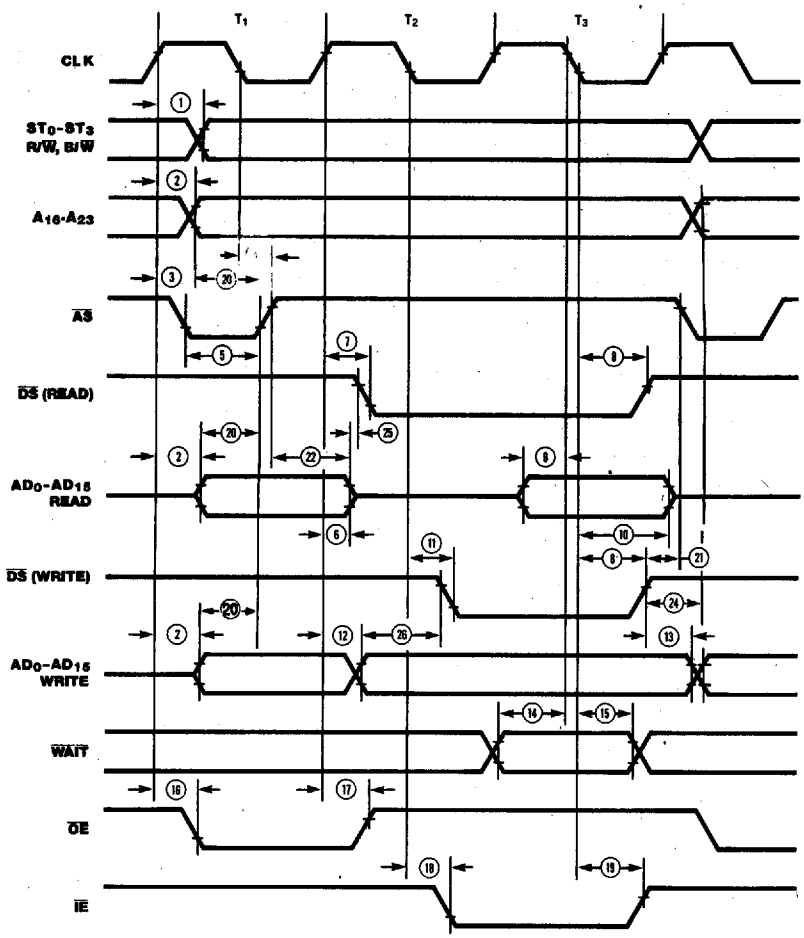


Figure 60. Z-Bus All Transactions

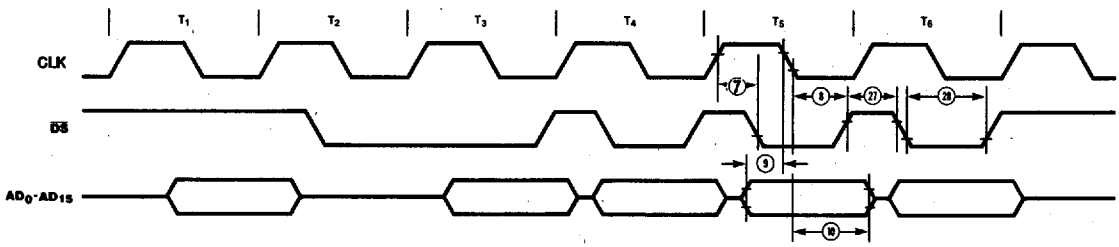


Figure 61. Z-Bus Burst Mode Timing

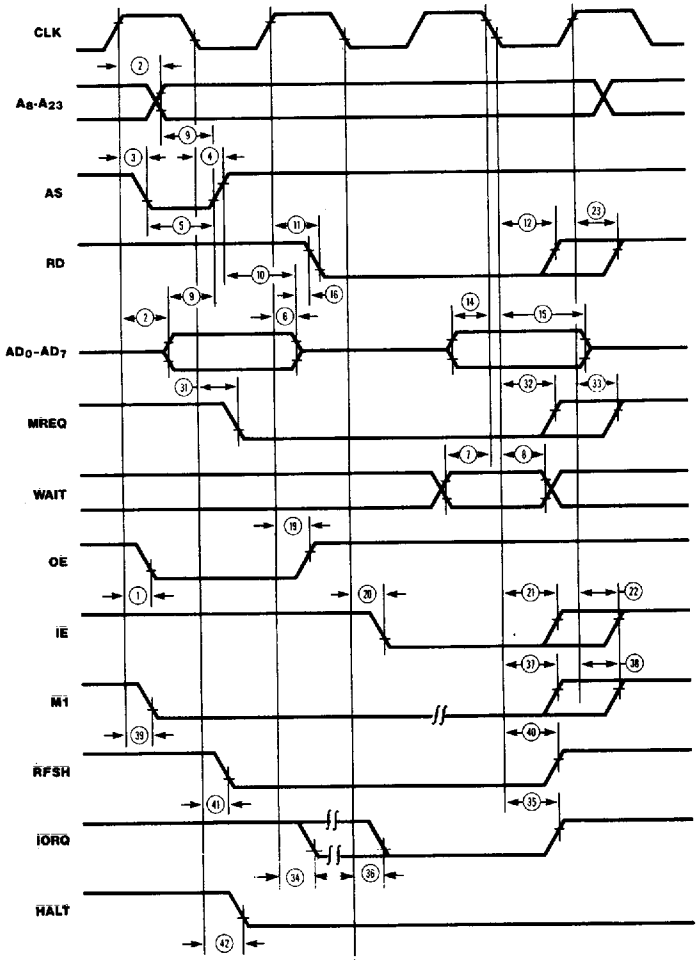


Figure 62. Z80 Bus Read Type Transactions

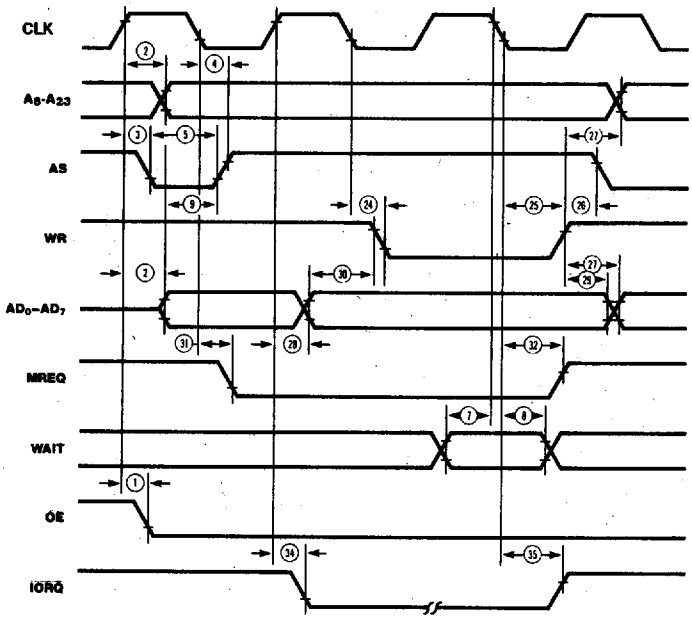


Figure 63. Z80 Bus Write Transactions

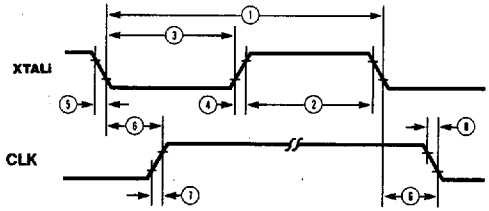


Figure 64. Z280 Clock Circuit

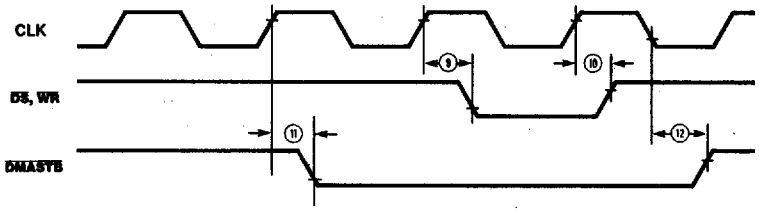


Figure 65. Flyby DMA Write to Memory
(Z-Bus: $\overline{DS}$; Z80 Bus: $\overline{WR}$)

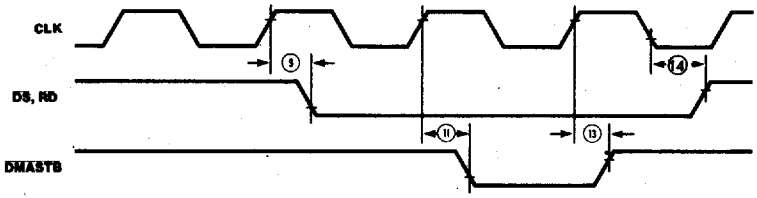


Figure 66. Flyby DMA Read from Memory
(Z-Bus: $\overline{DS}$; Z80 Bus: $\overline{RD}$)

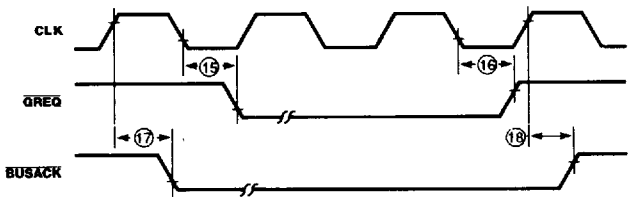


Figure 67. $\overline{GREQ}$ and $\overline{BUSACK}$ Timing

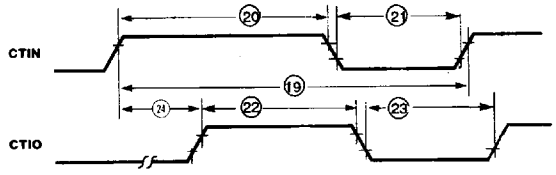


Figure 68. Counter/Timer Timing

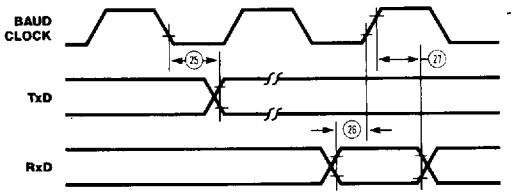


Figure 69. UART Timing

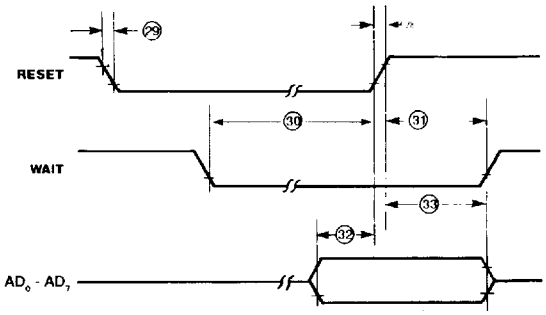


Figure 70. Reset Timing

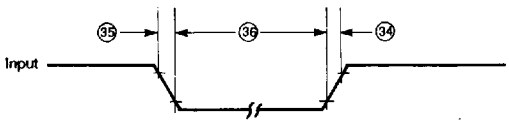


Figure 71. Inputs Timing