

Z8410 Z80[®] DMA Direct Memory Access Controller

Zilog

Product Specification

April 1985

FEATURES

- Transfers, searches, and search/transfers in Byte-at-a-Time, Burst, or Continuous modes. Cycle length and edge timing can be programmed to match the speed of any port.
- Dual port addresses (source and destination) generated for memory-to-I/O, memory-to-memory, or I/O-to-I/O operations. Addresses may be fixed or automatically incremented/decremented.
- Next-operation loading without disturbing current operations via buffered starting-address registers. An entire previous sequence can be repeated automatically.
- Extensive programmability of functions. CPU can read complete channel status.
- Standard Z80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic. Sophisticated, internally modifiable interrupt vectoring.
- Direct interfacing to system buses without external logic.

Z80 DMA

GENERAL DESCRIPTION

The Z80 DMA (Direct Memory Access) 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 that optimize transfer speed and control with little or no external logic in systems using an 8- or 16-bit data bus and a 16-bit address bus.

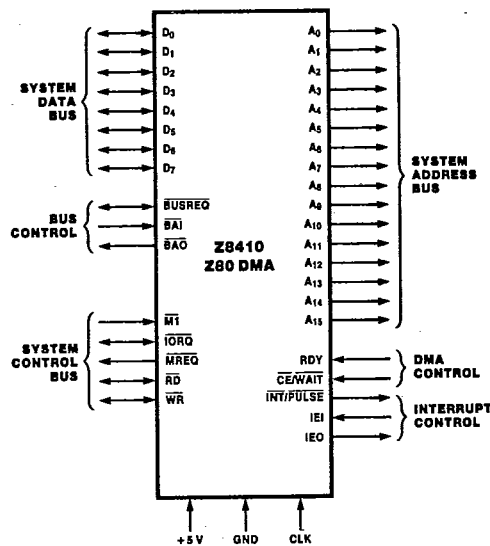


Figure 1. Pin Functions

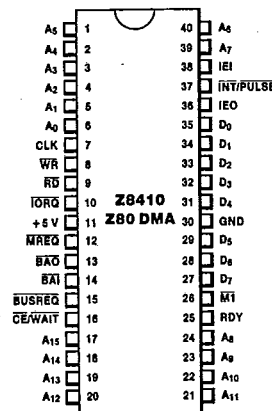


Figure 2. 40-pin Dual-In-Line Package (DIP), Pin Assignments

Transfers can be done between any two ports (source and destination), including memory-to-I/O, memory-to-memory, and I/O-to-I/O. Dual port addresses are automatically generated for each transaction and may be either fixed or incrementing/decrementing. In addition, bit-maskable byte searches can be performed either concurrently with transfers or as an operation in itself.

The Z80 DMA contains direct interfacing to, and independent control of, system buses, as well as sophisticated bus and interrupt controls. Many

programmable features, including variable cycle timing and auto-restart, minimize CPU software overhead. They are especially useful in adapting this special-purpose transfer processor to a broad variety of memory, I/O and CPU environments.

The Z80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin, plastic or ceramic DIP. It uses a single +5V power supply and the standard Z80 Family single-phase clock.

FUNCTIONAL DESCRIPTION

Classes of Operation. The Z80 DMA has three basic classes of operation:

- Transfers of data between two ports (memory or I/O peripheral)
- Searches for a particular 8-bit maskable byte at a single port in memory or an I/O peripheral
- Combined transfers with simultaneous search between two ports

Figure 4 illustrates the basic functions served by these classes of operation.

During a transfer, the DMA assumes control of the system address and data buses. Byte by byte, data is read from one addressable port and written to the other addressable port. The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data may be written from one peripheral to another, from one area of main memory to another, or from a peripheral to main memory and vice versa.

During a search-only operation, data is read from the source port and compared byte by byte with a DMA-internal register containing a programmable match byte. This match byte may optionally be masked so that only certain bits within the match byte are compared. Search rates up to 1.25M bytes per second can be obtained with the 2.5 MHz Z80 DMA or 2M bytes per second with the 4 MHz Z80A DMA.

In combined searches and transfers, data is transferred between two ports while simultaneously searching for a bit-maskable byte match.

Data transfers or searches can be programmed to stop, or interrupt, under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

Modes of Operation. The Z80 DMA can be programmed to operate in one of three transfer and/or search modes:

- *Byte-at-a-Time:* data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.

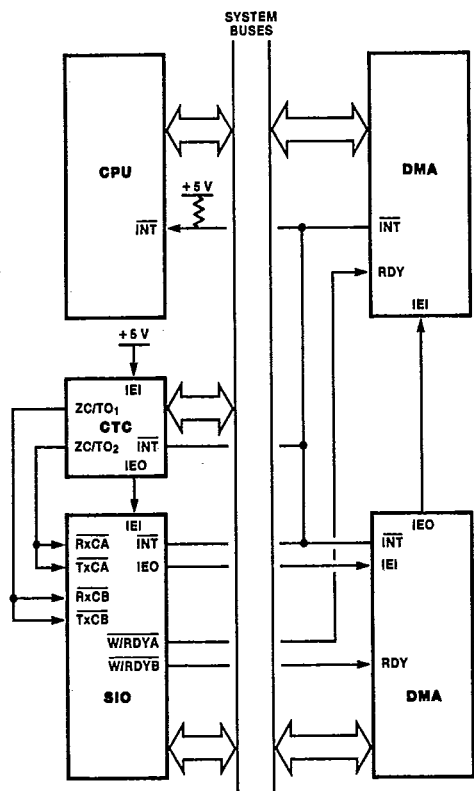


Figure 3. Typical Z80 Environment

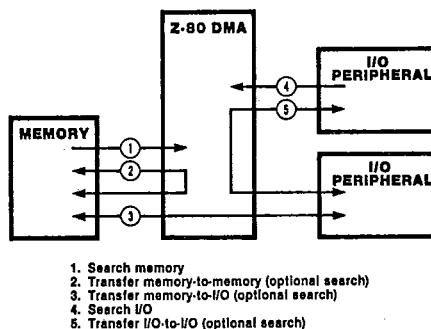


Figure 4. Basic Functions of the Z80 DMA

■ **Burst:** data operations continue until a port's Ready line to the DMA goes inactive. The DMA then stops and releases the system buses after completing its current byte operation.

■ **Continuous:** data operations continue until the end of the programmed block of data is reached before the system buses are released. If a port's Ready line goes inactive before this occurs, the DMA simply pauses until the Ready line comes active again.

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

Due to the DMA's high-speed buffered method of reading data, operations on one byte are not completed until the next byte is read in. This means that total transfer or search block lengths must be two or more bytes, and that block lengths programmed into the DMA must be one byte less than the desired block length (count is $N-1$ where N is the block length).

Commands and Status. The Z80 DMA has several writable control registers and readable status registers available to the CPU. Control bytes can be written to the DMA whenever the DMA is not controlling the system buses, but the act of writing a control byte to the DMA disables the DMA until it is again enabled by a specific command. Status bytes can also be read at any such time, but writing the Read Status Byte command or the Initiate Read Sequence command disables the DMA.

Control bytes to the DMA include those which effect immediate command actions such as enable, disable, reset, load starting-address buffers, continue, clear counters, and clear status bits. In addition, many mode-setting control bytes can be written, including mode and class of operation, port configuration, starting addresses, block length, address counting rule, match and match-mask byte, interrupt conditions, interrupt vector, status-affects-vector condition, pulse counting, auto restart, Ready-line and Wait-line rules, and read mask.

Readable status registers include a general status byte reflecting Ready-line, end-of-block, byte-match, and interrupt conditions, as well as 2-byte registers for the current byte count, Port A address, and Port B address.

Variable Cycle. The Z80 DMA has the unique feature of programmable operation-cycle length. This is valuable in tailoring the DMA to the particular requirements of other system components (fast or slow) and maximizes the data-transfer rate. It also eliminates external logic for signal conditioning.

There are two aspects to the variable cycle feature. First, the entire read and write cycles (periods) associated with the source and destination ports can be independently programmed as 2, 3, or 4 T-cycles long (more if Wait cycles are used), thereby increasing or decreasing the speed with which all DMA signals change (Figure 5).

Second, the four signals in each port specifically associated with transfers of data (I/O Request, Memory Request, Read and Write) can each have its active trailing edge terminated one-half T-cycle early. This adds a further dimension of flexibility and speed, allowing such things as shorter-than-normal Read or Write signals that go inactive before data starts to change.

Address Generation. Two 16-bit addresses are generated by the Z80 DMA for every transfer operation, one address for the source port and another for the destination port. Each address can be either variable or fixed. Variable addresses can increment or decrement from the programmed starting address. The fixed-address capability eliminates the need for separate enabling wires to I/O ports.

Port addresses are multiplexed onto the system address bus, depending on whether the DMA is reading the source port or writing to the destination port. Two readable address counters (2 bytes each) keep the current address of each port.

Auto Restart. The starting addresses of either port can be reloaded automatically at the end of a block. This option is selected by the Auto Restart control bit. The byte counter is cleared when the addresses are reloaded.

The Auto Restart feature relieves the CPU of software overhead for repetitive operations such as CRT refresh and many others. Moreover, when the CPU has access to the buses during byte-at-a-time or burst transfers, different starting addresses can be written into buffer registers during transfers, causing the Auto Restart to begin at a new location.

Interrupts. The Z80 DMA can be programmed to interrupt the CPU on four conditions:

- Interrupt on Ready (before requesting bus)
- Interrupt on Match
- Interrupt on End of Block

Any of these interrupts cause an interrupt-pending status bit to be set, and each of them can optionally alter the DMA's interrupt vector. Due to the buffered constraint mentioned under "Modes of Operation," interrupts on Match at End of Block are caused by matches to the byte just prior to the last byte in the block.

The DMA shares the Z80 Family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z80 CPU environment, the DMA passes

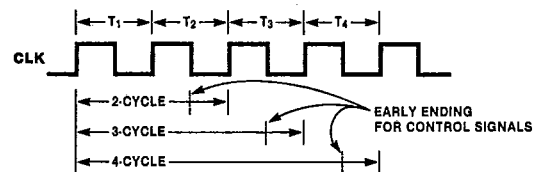


Figure 5. Variable Cycle Length

its internally modifiable 8-bit interrupt vector to the CPU, which adds an additional eight bits to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. In this process, CPU control is transferred directly to the interrupt routine, so that the next instruction executed after an interrupt acknowledge is the first instruction of the interrupt routine itself.

Pulse Generation. External devices can keep track of how many bytes have been transferred by using the DMA's pulse

output, which provides a signal at 256-byte intervals. The interval sequence may be offset at the beginning by 1 to 255 bytes.

The Interrupt line outputs the pulse signal in a manner that prevents misinterpretation by the CPU as an interrupt request, since it only appears when the Bus Request and Bus Acknowledge lines are both active.

PIN DESCRIPTION

A₀-A₁₅. *System Address Bus* (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

$\overline{\text{BAI}}$. *Bus Acknowledge In* (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the $\overline{\text{BAI}}$ pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their $\overline{\text{BAI}}$ connected to the $\overline{\text{BAO}}$ of a higher-priority DMA.

$\overline{\text{BAO}}$. *Bus Acknowledge Out* (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system buses. $\overline{\text{BAI}}$ and $\overline{\text{BAO}}$ form a daisy chain for multiple-DMA priority resolution over bus control.

$\overline{\text{BUSREQ}}$. *Bus Request* (bidirectional, active Low, open drain). As an output, it sends requests for control of the system address bus, data bus, and control bus to the CPU. As an input when multiple DMAs are strung together in a priority daisy chain via $\overline{\text{BAI}}$ and $\overline{\text{BAO}}$, it senses when another DMA has requested the buses and causes this DMA to refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a buffer between it and the CPU because it is unidirectional into the CPU. A pull-up resistor is connected to this pin.

$\overline{\text{CE/WAIT}}$. *Chip Enable and Wait* (input, active Low). Normally this functions only as a $\overline{\text{CE}}$ line, but it can also be programmed to serve a $\overline{\text{WAIT}}$ function. As a $\overline{\text{CE}}$ line from the CPU, it becomes active when $\overline{\text{WR}}$ or $\overline{\text{RD}}$ and $\overline{\text{IORQ}}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control, command bytes from the CPU to the DMA, or status bytes from the DMA to the CPU. As a $\overline{\text{WAIT}}$ line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. *System Clock* (input). Standard Z80 single-phase clock at 2.5 MHz (Z80 DMA) or 4.0 MHz (Z80A DMA). For 2.5 MHz clocks, a TTL gate with pullup resistor may be adequate to meet the timing and voltage level specification. For 4.0 MHz clocks, use a clock driver with an active pullup to meet the V_{IH} specification and rise-time requirements. In

all cases there should be a resistive pullup to the power supply of 10K ohms (max) to ensure proper power when the DMA is reset.

D₀-D₇. *System Data Bus* (bidirectional, 3-state). Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.

IEI. *Interrupt Enable In* (input, active High). This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT/PULSE}}$. *Interrupt Request* (output, active Low, open drain). While the CPU is the bus master, this output requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{\text{IORQ}}$ output Low during an $\overline{\text{M1}}$ cycle. It is typically connected to the $\overline{\text{INT}}$ pin of the CPU with a pullup resistor and tied to all other $\overline{\text{INT}}$ pins in the system. This pin can also be used to generate periodic pulses to an external device when the DMA is bus master (i.e., the CPU's $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ lines are both Low and the CPU cannot see interrupts). While the DMA is the bus master, this output can be programmed to pulse each time 256 transfers have occurred.

$\overline{\text{IORQ}}$. *Input/Output Request* (bidirectional, active Low, 3-state). As an input, this indicates that the lower half of the address bus holds a valid I/O port address for transfer of control or status bytes from, or to, the CPU, respectively; this DMA is the addressed port if its $\overline{\text{CE}}$ pin and its $\overline{\text{WR}}$ or $\overline{\text{RD}}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the lower half of the address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both active simultaneously, an interrupt acknowledge is indicated.

$\overline{\text{M1}}$. *Machine Cycle One* (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI, ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{\text{M1}}$ is active as each opcode byte is

fetched. An interrupt acknowledge is indicated when both $\overline{M1}$ and \overline{IORQ} are active.

MREQ. *Memory Request* (output, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. After the DMA has taken control of the system buses, it indicates a DMA transfer request from, or to, memory.

RD. *Read* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.

RDY. *Ready* (input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (Byte, Burst, or Continuous), the RDY line indirectly controls DMA activity by causing the \overline{BUSREQ} line to go Low or High.

WR. *Write* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled write to a memory or I/O port address.

INTERNAL STRUCTURE

The internal structure of the Z80 DMA includes driver and receiver circuitry for interfacing with an 8-bit system data bus, a 16-bit system address bus, and system control lines (Figure 6). In a Z80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{CE}/\overline{WAIT}$ line.

The DMA's internal data bus interfaces with the system data bus and services all internal logic and registers. Addresses generated from this logic for Ports A and B (source and destination) of the DMA's single transfer channel are multiplexed onto the system address bus.

Specialized logic circuits in the DMA are dedicated to the various functions of external bus interfacing, internal bus control, byte matching, byte counting, periodic pulse generation, CPU interrupts, bus requests, and address generation. A set of twenty-one writable control registers and seven readable status registers provides the means by which the CPU governs and monitors the activities of these logic circuits. All registers are eight bits wide, with double-byte information stored in adjacent registers. The two address counters (two bytes each) for Ports A and B are buffered by the two starting addresses.

The 21 writable control registers are organized into seven base-register groups, most of which have multiple registers. The base registers in each writable group contain both

control/command bits and pointer bits that can be set to address other registers within the group. The seven readable status registers have no analogous second-level registers.

The registers are designated as follows, according to their base-register groups:

WR0-WR6—Write Register groups 0 through 6 (7 base registers plus 14 associated registers)

RR0-RR6—Read Registers 0 through 6

Writing to a register within a write-register group involves first writing to the base register, with the appropriate pointer bits set, then writing to one or more of the other registers within the group. All seven of the readable status registers are accessed sequentially according to a programmable mask contained in one of the writable registers. The section entitled Programming explains this in more detail.

A pipelining scheme is used for reading data in. The programmed block length is the number of bytes compared to the byte counter, which increments at the end of each cycle. In searches, data byte comparisons with the match byte are made during the read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

Z80 DMA

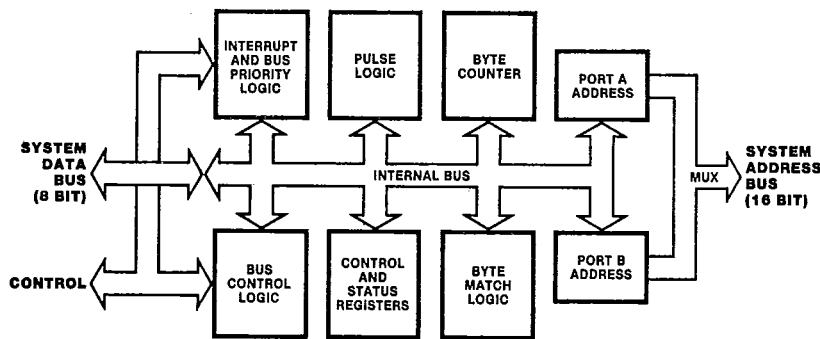


Figure 6. Block Diagram

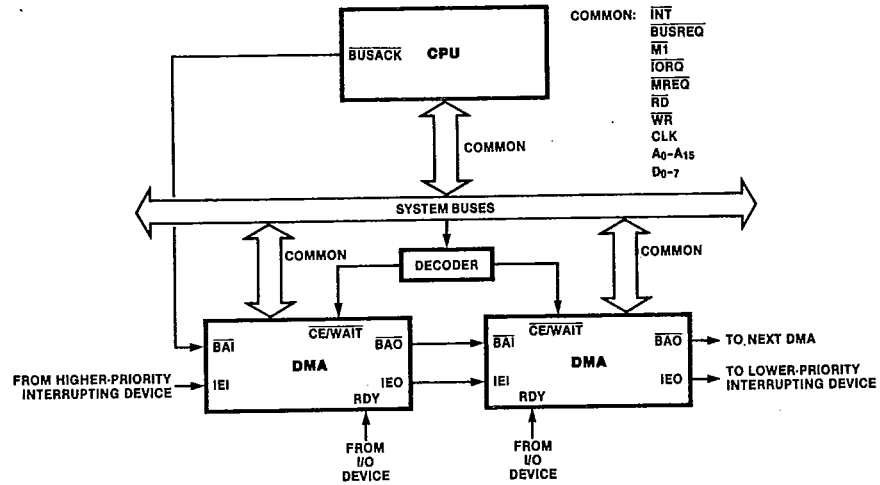


Figure 7. Multiple-DMA Interconnection to the Z80 CPU

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected (Zilog Microprocessor Applications Reference Book, Volume 1, # 00-2145-01, *The Z80 Family Program Interrupt Structure*). The system bus, however, may not be pre-empted. Any DMA that gains access to the system buses keeps them until it is finished.

Read Registers	
RR0	Status byte
RR1	Byte counter (low byte)
RR2	Byte counter (high byte)
RR3	Port A address counter (low byte)
RR4	Port A address counter (high byte)
RR5	Port B address counter (low byte)
RR6	Port B address counter (high byte)

Write Registers	
WR0	Base register byte Port A starting address (low byte) Port A starting address (high byte) Block length (low byte) Block length (high byte)
WR1	Base register byte Port A variable-timing byte
WR2	Base register byte Port B variable-timing byte
WR3	Base register byte Mask byte Match byte
WR4	Base register byte Port B starting address (low byte) Port B starting address (high byte) Interrupt control byte Pulse control byte Interrupt vector
WR5	Base register byte
WR6	Base register byte Read mask

PROGRAMMING

The Z80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests nor data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enable command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z80 CPU).

Reading. (Figure 8a) The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Writing. Control or command bytes are written into one or more of the Write Register groups (WRO-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

This is illustrated in Figure 8b. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)," then the next two bytes written to the DMA will be stored in that order in these two registers.

Fixed-Address Programming. A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address with LOAD command to WR6.
3. Declare Port A as a source in WR0.
4. Load Port A address with LOAD command to WR6.
5. Enable DMA in WR6.

Figure 9 illustrates a program to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050_H and the Port B peripheral fixed address is 05_H. Note that the data flow is 1001_H bytes—one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as the Z80 CPU's OTIR instruction.

Z80 DMA

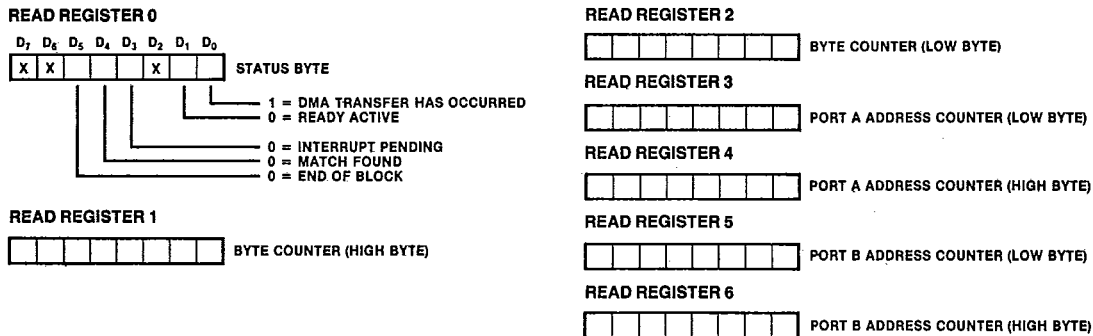


Figure 8a. Read Registers

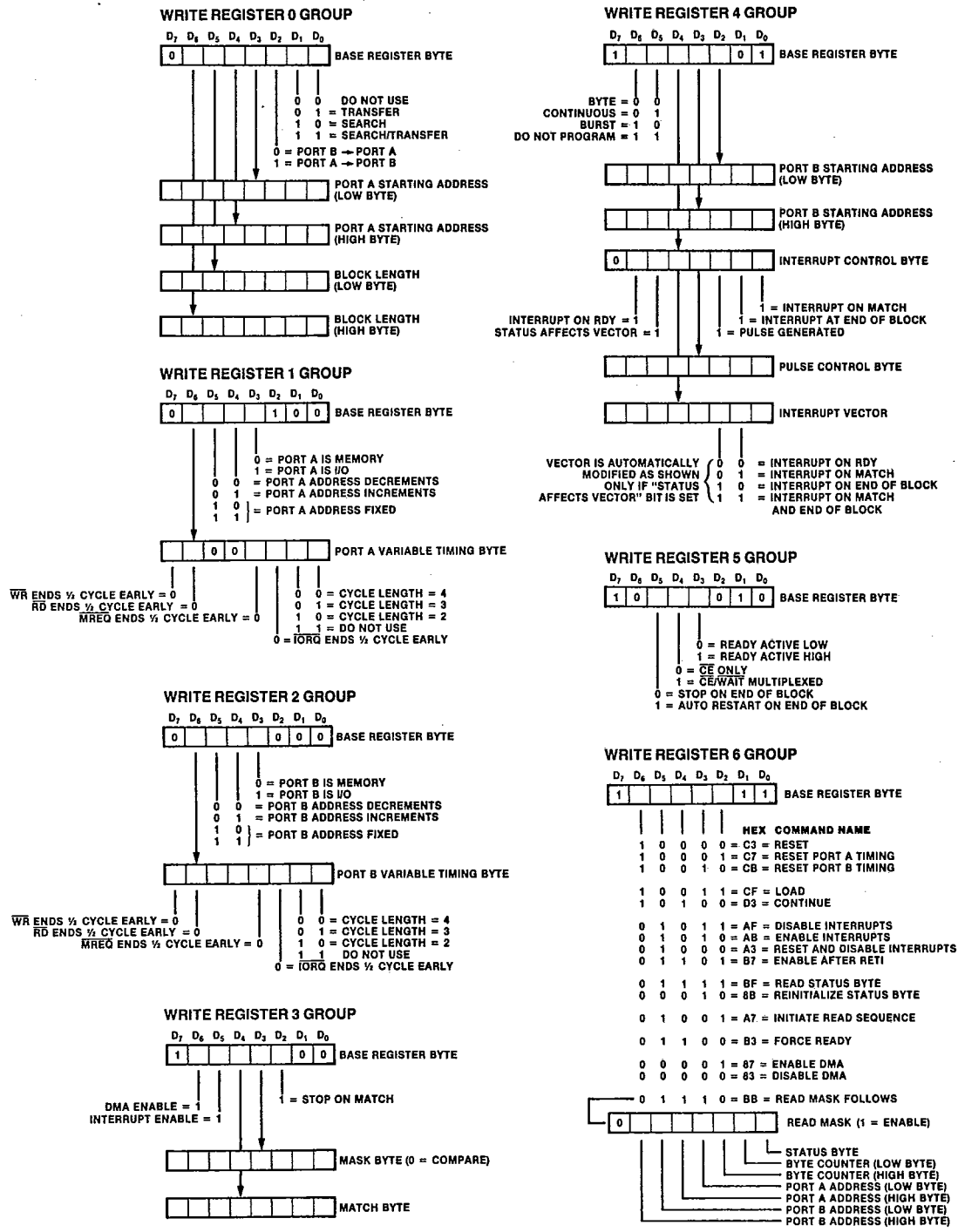


Figure 8b. Write Registers

Comments	D7	D6	D5	D4	D3	D2	D1	D0	HEX
WR0 sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0	1	Block Length Upper Follows	1	Port A Upper Address Follows	0 → A B → Temporary for Loading B Address*	0	1	79
Port A address (lower)	0	1	0	1	0	0	0	0	50
Port A address (upper)	0	0	0	1	0	0	0	0	10
Block length (lower)	0	0	0	0	0	0	0	0	00
Block length (upper)	0	0	0	1	0	0	0	0	10
WR1 defines Port A as memory with fixed incrementing address.	0	0	No Timing Follows	1	Port is Memory	1	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	0	No Timing Follows	0	1	0	0	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1	Burst Mode	0	0	1	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High.	1	0	No Auto Restart	0	1	0	1	0	8A
WR6 loads Port B address and resets block counter.*	1	1	0	0	1	1	1	1	CF
WR0 sets Port A as source.*	0	0	No Address or Block Length Bytes	0	0	1 → B A →	0	1	05
WR6 loads Port A address and resets block counter.	1	1	0	0	1	1	1	1	CF
WR6 enables DMA to start operation.	1	0	0	0	0	1	1	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.
*These entries are necessary only in the case of a fixed destination address.

Figure 9. Sample DMA Program

YMD 08Z

INACTIVE STATE TIMING

(DMA as CPU Peripheral)

In its disabled or inactive state, the DMA is addressed by the CPU as an I/O peripheral for write and read (control and status) operations. Write timing is illustrated in Figure 10.

Reading of the DMA's status byte, byte counter, or port address counters is illustrated in Figure 11. These

operations require less than three T-cycles. The \overline{CE} , \overline{IORQ} , and \overline{RD} lines are made active over two rising edges of CLK, and data appears on the bus approximately one T-cycle after they become active.

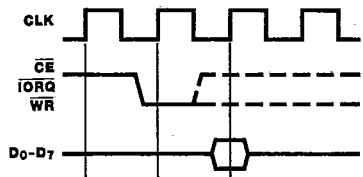


Figure 10. CPU-to-DMA Write Cycle

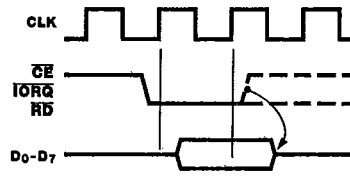


Figure 11. CPU-to-DMA Read Cycle

ACTIVE STATE TIMING

(DMA as Bus Controller)

Default Read and Write Cycles. By default, and after reset, the DMA's timing of read and write operations is exactly the same as the Z80 CPU's timing of read and write cycles for memory and I/O peripherals, with one exception: during a read cycle, data is latched on the falling edge of T_3 and held on the data bus across the boundary between read and write cycles, through the end of the following write cycle.

Memory-to-memory and I/O-to-I/O transfer timings are simply permutations of these diagrams.

The default timing uses three T-cycles for memory transactions and four T-cycles for I/O transactions, which include one automatically inserted wait cycle (T_{WA}) between T_2 and T_3 . If the $\overline{CE}/\overline{WAIT}$ line is programmed to act as a \overline{WAIT} line during the DMA's active state, it is sampled on the falling edge of T_2 for memory transactions and the falling edge of T_{WA} for I/O transactions. If $\overline{CE}/\overline{WAIT}$ is Low during this time another T-cycle is added, during which the

Figure 12 illustrates the timing for memory-to-I/O port transfers and Figure 13 illustrates I/O-to-memory transfers.

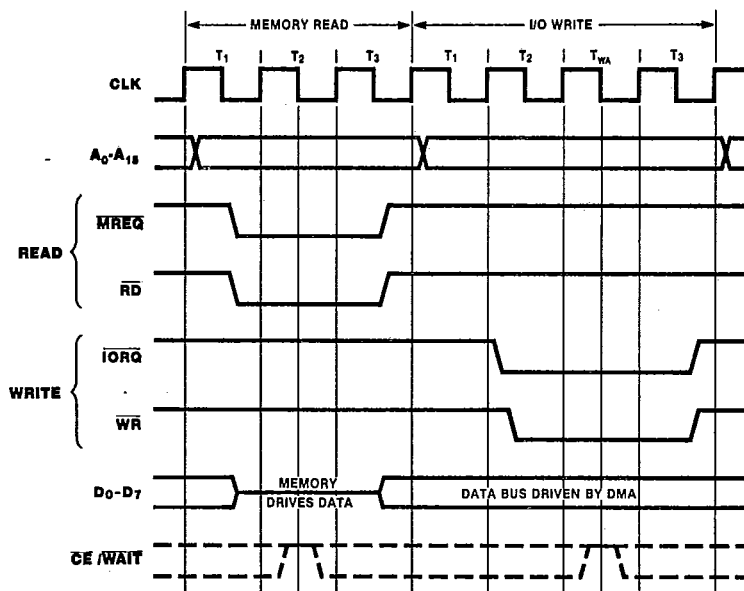


Figure 12. Memory-to-I/O Transfer

CE/WAIT line will again be sampled. The duration of transactions can thus be indefinitely extended.

Variable Cycle and Edge Timing. The Z80 DMA's default operation-cycle length for the source (read) port and destination (write) port can be independently programmed. This variable-cycle feature allows read or write cycles consisting of two, three, or four T-cycles (more if Wait cycles are inserted), thereby increasing or decreasing the speed of all signals generated by the DMA. In addition, the trailing edges of the IORQ, MREQ, RD, and WR signals can be independently terminated one-half cycle early. Figure 14 illustrates this.

In the variable-cycle mode, unlike default timing, IORQ comes active one-half cycle before MREQ, RD, and WR. CE/WAIT can be used to extend only the 3 or 4 T-cycle variable memory cycles and only the 4-cycle variable I/O cycle. The CE/WAIT line is sampled at the falling edge of T₂ for 3- or 4-cycle memory cycles, and at the falling edge of T₃ for 4-cycle I/O cycles.

During transfers, data is latched on the clock edge causing the rising edge of RD and held until the end of the write cycle.

Bus Requests. Figure 15 illustrates the bus request and acceptance timing. The RDY line, which may be programmed active High or Low, is sampled on every rising edge of CLK. If it is found to be active, and if the bus is not in use by any other device, the following rising edge of CLK drives BUSREQ Low. After receiving BUSREQ, the CPU acknowledges on the BAI input either directly or through a multiple-DMA daisy chain. When a Low is detected on BAI for two consecutive rising edges of CLK, the DMA will begin transferring data on the next rising edge of CLK.

Bus Release Byte-at-a-Time. In Byte-at-a-Time mode, BUSREQ is brought High on the rising edge of CLK prior to the end of each read cycle (search-only) or write cycle (transfer and transfer/search) as illustrated in Figure 16. This is done regardless of the state of RDY. There is no possibility of confusion when a Z80 CPU is used since the CPU cannot

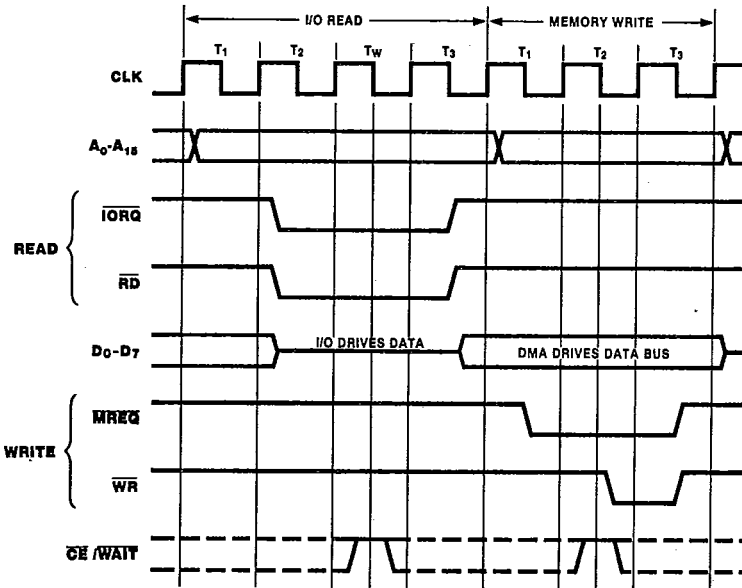


Figure 13. I/O-to-Memory Transfer

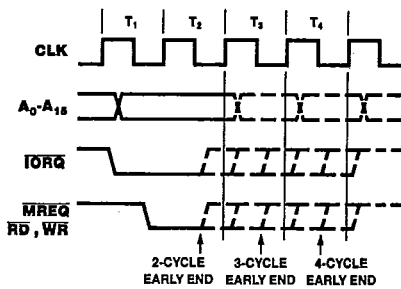


Figure 14. Variable-Cycle and Edge Timing

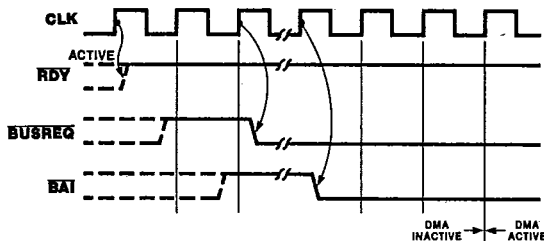


Figure 15. Bus Request and Acceptance

begin an operation until the following T-cycle. Most other CPUs are not bothered by this either, although note should be taken of it. The next bus request for the next byte will come after both $\overline{\text{BUSREQ}}$ and $\overline{\text{BAI}}$ have returned High.

Bus Release at End of Block. In Burst and Continuous modes, an end of block causes $\overline{\text{BUSREQ}}$ to go High usually on the same rising edge of CLK in which the DMA completes the transfer of the data block (Figure 17). The last byte in the block is transferred even if RDY goes inactive before completion of the last byte transfer.

Bus Release on Not Ready. In Burst mode, when RDY goes inactive it causes $\overline{\text{BUSREQ}}$ to go High on the next rising edge of CLK after the completion of its current byte operation (Figure 18). The action on $\overline{\text{BUSREQ}}$ is thus somewhat delayed from action on the RDY line. The DMA always completes its current byte operation in an orderly fashion before releasing the bus.

By contrast, $\overline{\text{BUSREQ}}$ is not released in Continuous mode when RDY goes inactive. Instead, the DMA idles after completing the current byte operation, awaiting an active RDY again.

Bus Release on Match. If the DMA is programmed to stop on match in Burst or Continuous modes, a match causes

$\overline{\text{BUSREQ}}$ to go inactive on the next DMA operation, i.e., at the end of the next read in a search or at the end of the following write in a transfer (Figure 19). Due to the pipelining scheme, matches are determined while the next DMA read or write is being performed.

The RDY line can go inactive after the matching operation begins without affecting this bus-release timing.

Interrupts. Timings for interrupt acknowledge and return from interrupt are the same as timings for these in other Z80 peripherals. Refer to the Zilog Microprocessor Applications Reference Book, Volume 1, #00-2145-01, (*The Z80 Family Program Interrupt Structure*).

Interrupt on RDY (interrupt before requesting bus) does not directly affect the $\overline{\text{BUSREQ}}$ line. Instead, the interrupt service routine must handle this by issuing the following commands to WR6:

1. Enable after Return From Interrupt (RETI) Command—Hex B7
2. Enable DMA—Hex 87
3. An RETI instruction that resets the Interrupt Under Service latch in the Z80 DMA.

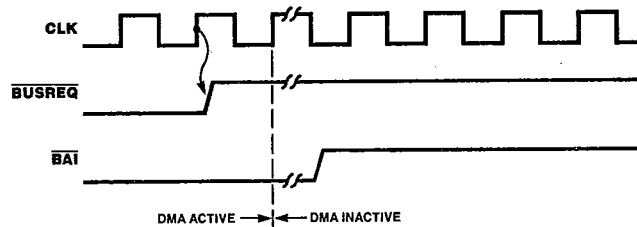


Figure 16. Bus Release (Byte-at-a-Time Mode)

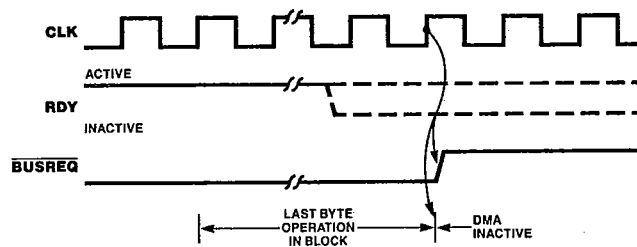


Figure 17. Bus Release at End of Block (Burst and Continuous Modes)

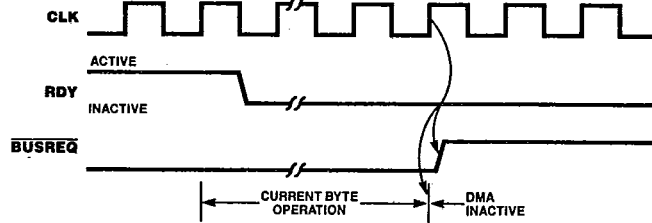


Figure 18. Bus Release When Not Ready (Burst Mode)

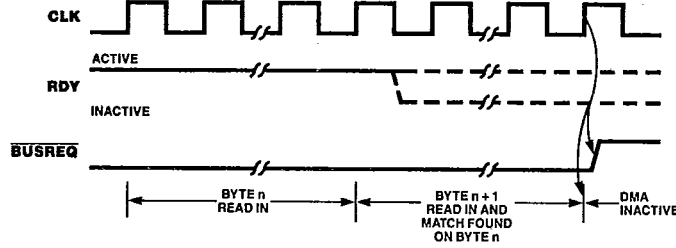
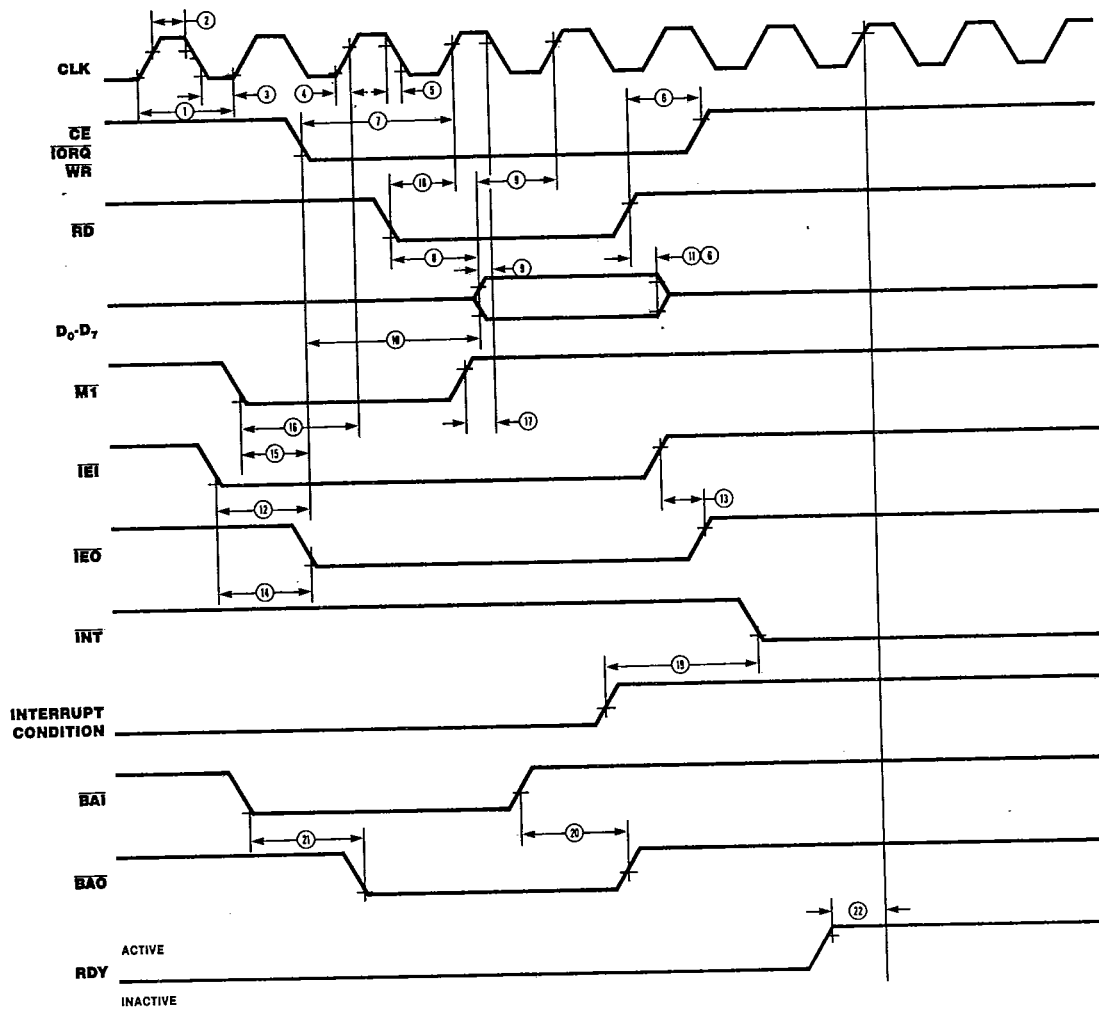


Figure 19. Bus Release on Match (Burst and Continuous Modes)

Z80 DMA

AC CHARACTERISTICS
(Inactive State)



NOTE: Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

AC CHARACTERISTICS

(Inactive State)

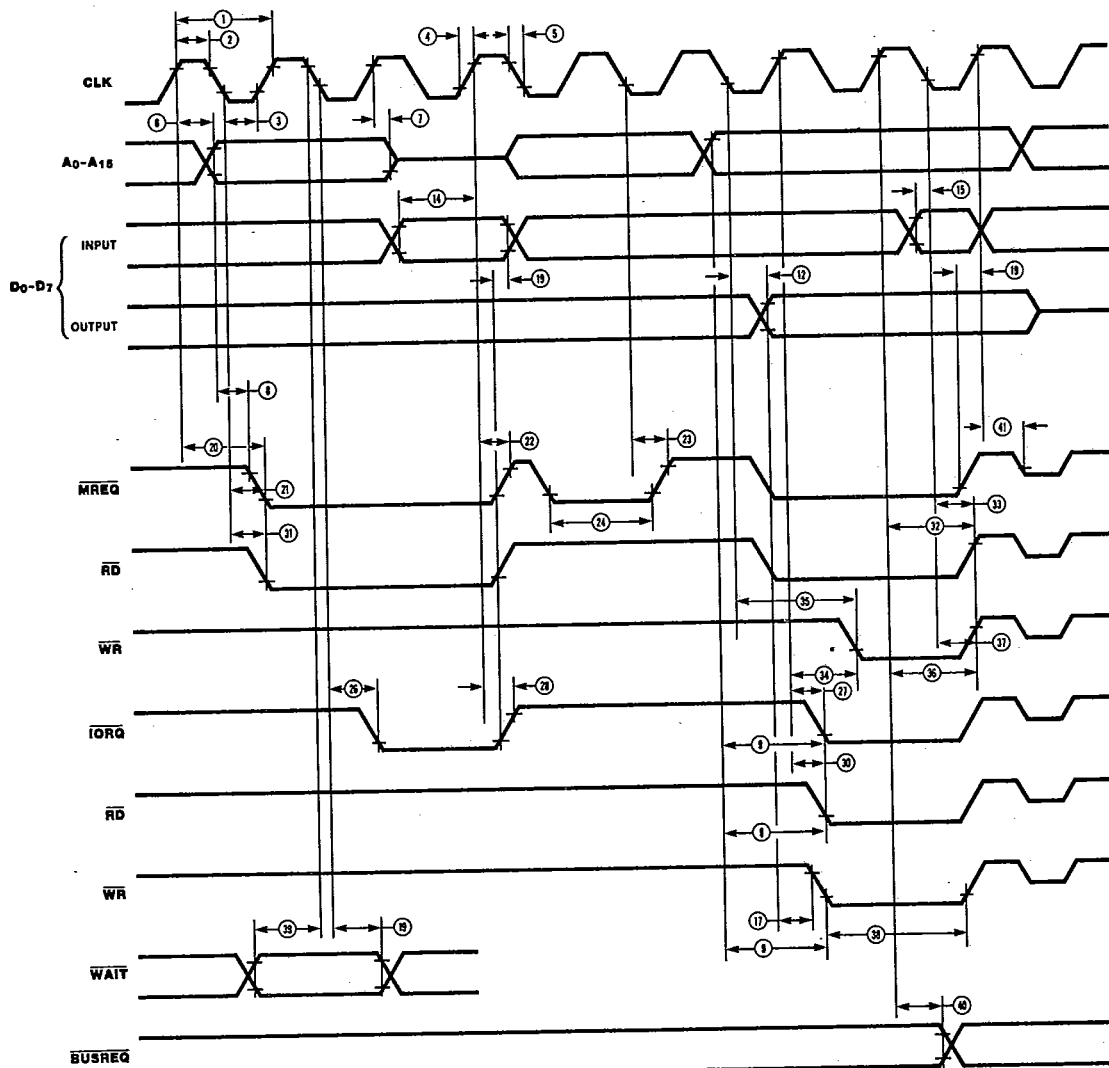
Number	Symbol	Parameter	Z80 DMA		Z80A DMA		Unit
			Min	Max	Min	Max	
1	T _c C	Clock Cycle Time	400	4000	250	4000	ns
2	T _w Ch	Clock Width (High)	170	2000	110	2000	ns
3	T _w Cl	Clock Width (Low)	170	2000	110	2000	ns
4	T _r C	Clock Rise Time		30		30	ns
5	T _f C	Clock Fall Time		30		30	ns
6	T _h	Hold Time for Any Specified Setup Time	0		0		ns
7	T _s C(Cr)	$\overline{\text{IORQ}}$, $\overline{\text{WR}}$, $\overline{\text{CE}}$ ↓ to Clock ↑ Setup	280		145		ns
8	T _d DO(RDf)	$\overline{\text{RD}}$ ↓ to Data Output Delay		500		380	ns
9	T _s Dl(Cr)	Data In to Clock ↑ Setup ($\overline{\text{WR}}$ or $\overline{\text{M1}}$)	50		50		ns
10	T _d DO(IOf)	$\overline{\text{IORQ}}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
11	T _d RD _r (Dz)	$\overline{\text{RD}}$ ↑ to Data Float Delay (output buffer disable)		160		110	ns
12	T _s IEl(IORQf)	IEl to $\overline{\text{IORQ}}$ ↓ Setup (INTA Cycle)	140		140		ns
13	T _d IEOr(IEIr)	IEl ↑ to IEO ↑ Delay		210		160	ns
14	T _d IEOf(IEIf)	IEl ↓ to IEO ↓ Delay		190		130	ns
15	T _d M1f(IEOf)	$\overline{\text{M1}}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{\text{M1}}$ ↓)		300		190	ns
16	T _s M1f(Cr)	$\overline{\text{M1}}$ ↓ to Clock ↑ Setup	210		90		ns
17	T _s M1r(Cf)	$\overline{\text{M1}}$ ↑ to Clock Setup	20		-10		ns
18	T _s RDl(Cr)	$\overline{\text{RD}}$ ↓ to Clock ↑ Setup ($\overline{\text{M1}}$ Cycle)	240		115		ns
19	T _d l(INTf)	Interrupt Cause to $\overline{\text{INT}}$ ↓ Delay ($\overline{\text{INT}}$ generated only when DMA is inactive)		500		500	ns
20	T _d BAIr(BAO _r)	$\overline{\text{BAI}}$ ↑ to $\overline{\text{BAO}}$ ↑ Delay		200		150	ns
21	T _d BAIf(BAO _f)	$\overline{\text{BAI}}$ ↓ to $\overline{\text{BAO}}$ ↓ Delay		200		150	ns
22	T _s RDY(Cr)	RDY Active to Clock ↑ Setup	150		100		ns

NOTE: Negative minimum setup values mean that the first-mentioned event can come after the second-mentioned event.

Z80 DMA

AC CHARACTERISTICS

(Active State)



NOTE: Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

Number	Symbol	Parameter	Z80 DMA ^{°†}		Z80A DMA ^{°††}	
			Min(ns)	Max(ns)	Min(ns)	Max(ns)
1	T _c C	Clock Cycle Time	400		250	
2	T _w Ch	Clock Width (High)	180	2000	110	2000
3	T _w Cl	Clock Width (Low)	180	2000	110	2000
4	T _r C	Clock Rise Time		30		30
5	T _f C	Clock Fall Time		30		30

NOTES:

- ° Numbers in parentheses are other parameter-numbers in this table; their values should be substituted in equations.
- ‡ All equations imply DMA default (standard) timing.
- † Data must be enabled onto data bus when RD is active.
- * Parameter is not illustrated in the AC Timing Diagrams.

AC CHARACTERISTICS (Continued)
(Active State)

Number	Symbol	Parameter	Z80 DMA ^{°††}		Z80A DMA ^{°††}	
			Min(ns)	Max(ns)	Min(ns)	Max(ns)
6	TdA	Address Output Delay		145		110
7	TdC(Az)	Clock ↑ to Address Float Delay		110		90
8	TsA(MREQ)	Address to $\overline{\text{MREQ}}$ ↓ Setup (Memory Cycle)	(2) + (5) - 75		(2) + (5) - 75	
9	TsA(IRW)	Address Stable to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Setup (I/O Cycle)	(1) - 80		(1) - 70	
*10	TdRW(A)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Stable Delay	(3) + (4) + 40		(3) + (4) - 50	
*11	TdRW(Az)	$\overline{\text{RD}}$, $\overline{\text{WR}}$ ↑ to Addr. Float	(3) + (4) - 60		(3) + (4) - 45	
12	TdCf(DO)	Clock ↓ to Data Out Delay		230		150
*13	TdCr(Dz)	Clock ↑ to Data Float Delay (Write Cycle)		90		90
14	TsDI(Cr)	Data In to Clock ↑ Setup (Read cycle when rising edge ends read)	50		35	
15	TsDI(Cf)	Data In to Clock ↓ Setup (Read cycle when falling edge ends read)	60		50	
*16	TsDO(WfM)	Data Out to $\overline{\text{WR}}$ ↓ Setup (Memory Cycle)	(1) - 210		(1) - 170	
17	TsDO(WfI)	Data Out to $\overline{\text{WR}}$ ↓ Setup (I/O cycle)	100		100	
*18	TdWr(DO)	$\overline{\text{WR}}$ ↑ to Data Out Delay	(3) + (4) - 80		(3) + (4) - 70	
19	Th	Hold Time for Any Specified Setup Time	0		0	
20	TdCr(Mf)	Clock ↑ to $\overline{\text{MREQ}}$ ↓ Delay		100		85
21	TdCf(Mf)	Clock ↓ to $\overline{\text{MREQ}}$ ↓ Delay		100		85
22	TdCr(Mr)	Clock ↑ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
23	TdCf(Mr)	Clock ↓ to $\overline{\text{MREQ}}$ ↑ Delay		100		85
24	TwM1	$\overline{\text{MREQ}}$ Low Pulse Width	(1) - 40		(1) - 30	
*25	TwMh	$\overline{\text{MREQ}}$ High Pulse Width	(2) + (5) - 30		(2) + (5) - 20	
26	TdCf(Irf)	Clock ↓ to $\overline{\text{IORQ}}$ ↓ Delay		110		85
27	TdCr(Irf)	Clock ↑ to $\overline{\text{IORQ}}$ ↓ Delay		90		75
28	TdCr(Ir)	Clock ↑ to $\overline{\text{IORQ}}$ ↑ Delay		100		85
*29	TdCf(Ir)	Clock ↓ to $\overline{\text{IORQ}}$ ↑ Delay		110		85
30	TdCr(Rf)	Clock ↑ to $\overline{\text{RD}}$ ↓ Delay		100		85
31	TdCf(Rf)	Clock ↓ to $\overline{\text{RD}}$ ↓ Delay		130		95
32	TdCr(Rr)	Clock ↑ to $\overline{\text{RD}}$ ↑ Delay		100		85
33	TdCf(Rr)	Clock ↓ to $\overline{\text{RD}}$ ↑ Delay		110		85
34	TdCr(Wf)	Clock ↑ to $\overline{\text{WR}}$ ↓ Delay		80		65
35	TdCf(Wf)	Clock ↓ to $\overline{\text{WR}}$ ↓ Delay		90		80
36	TdCr(Wr)	Clock ↑ to $\overline{\text{WR}}$ ↑ Delay		100		80
37	TdCf(Wr)	Clock ↓ to $\overline{\text{WR}}$ ↑ Delay		100		80
38	TwWI	$\overline{\text{WR}}$ Low Pulse Width	(1) - 40		(1) - 30	
39	TsWA(Cf)	$\overline{\text{WAIT}}$ to Clock ↓ Setup	70		70	
40	TdCr(B)	Clock ↑ to $\overline{\text{BUSREQ}}$ Delay		150		100
41	TdCr(Iz)	Clock ↑ to $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ Float Delay		100		80

Z80 DMA

NOTES: ° Numbers in parentheses are other parameter-numbers in this table; their values should be substituted in equations.
 † All equations imply DMA default (standard) timing.
 ‡ Data must be enabled onto data bus when RD is active.
 * Parameter is not illustrated in the AC Timing Diagrams.

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to ground -0.3V to +7V
 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 listed below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0V). Positive current flows into the referenced pin.

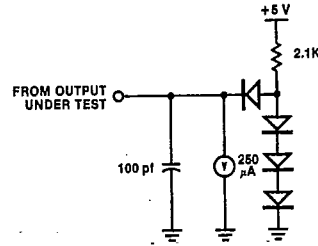
ranges and product numbers. Package drawings are in the Package Information section. Refer to the Literature List for additional documentation.

Available operating temperature ranges are:

- S = 0°C to +70°C, +4.75V ≤ V_{CC} ≤ +5.25V

All ac parameters assume a load capacitance of 100 pf max. Timing references between two output signals assume a load difference of 50 pf max.

The Ordering Information section lists package temperature



DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V _{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V _{IHC}	Clock Input High Voltage	V _{CC} - .6	5.5	V	
V _{IL}	Input Low Voltage	-0.3	0.8	V	
V _{IH}	Input High Voltage	2.0	5.5	V	
V _{OL}	Output Low Voltage		0.4	V	I _{OL} = 3.2mA for <u>BUSREQ</u> I _{OL} = 2.0 mA for all others
V _{OH}	Output High Voltage	2.4		V	I _{OH} = 250 μA
I _{CC}	Power Supply Current				
	Z80 DMA		150	mA	
	Z80A DMA		200	mA	
I _{LI}	Input Leakage Current		10	μA	V _{IN} = 0 to V _{CC}
I _{LO}	3-State Output Leakage Current in Float		± 10	μA	V _{OUT} = 0.4V to V _{CC}
I _{LD}	Data Bus Leakage Current in Input Mode		± 10	μA	0 ≤ V _{IN} ≤ V _{CC}

V_{CC} = 5V ± 5% unless otherwise specified, over specified temperature range.

CAPACITANCE

Symbol	Parameter	Min	Max	Unit
C	Clock Capacitance		35	pf
C _{IN}	Input Capacitance		5	pf
C _{OUT}	Output Capacitance		15	pf

NOTES: Over specified temperature range; f = MHz.
 Unmeasured pins returned to ground.

ORDERING INFORMATION

Z80 DMA, 2.5 MHz
40-pin DIP
 Z8410 PS
 Z8410 CS

Z80A DMA, 4.0 MHz
40-pin DIP
 Z8410A PS
 Z8410A CS

Codes

First letter is for package; second letter is for temperature.

C = Ceramic DIP
 P = Plastic DIP
 L = Ceramic LCC
 V = Plastic PCC

R = Protopack
 T = Low Profile Protopack
 DIP = Dual-In-Line Package
 LCC = Leadless Chip Carrier
 PCC = Plastic Chip Carrier (Leaded)

TEMPERATURE
 S = 0°C to +70°C
 E = -40°C to +85°C
 M* = -55°C to +125°C

FLOW
 B = 883 Class B

Example: PS is a plastic DIP, 0°C to +70°C.

† Available soon.

* For Military Orders, contact your local Zilog Sales Office for Military Electrical Specifications.

Z80 DMA