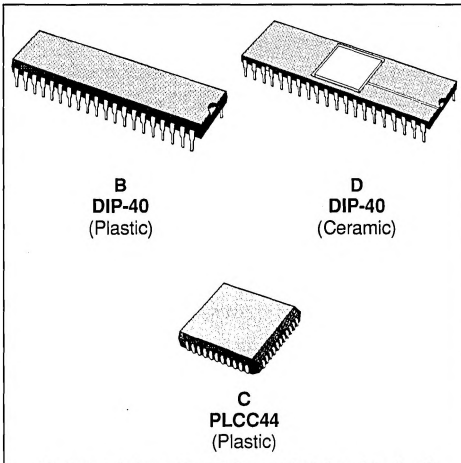
Z80 DMA CMOS DIRECT MEMORY ACCESS CONTROL

- 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 (sources 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
- SINGLE 5 V ± 10 % POWER SUPPLY
- LOW POWER CONSUMPTION :
  - 5 mA TYP. AT 4 MHz
  - 6 mA TYP. AT 6 MHz
  - LESS THAN 10 µA IN POWER DOWN MODE
- EXTENDED OPERATING TEMPERATURE
  - 40 °C TO + 85 °C



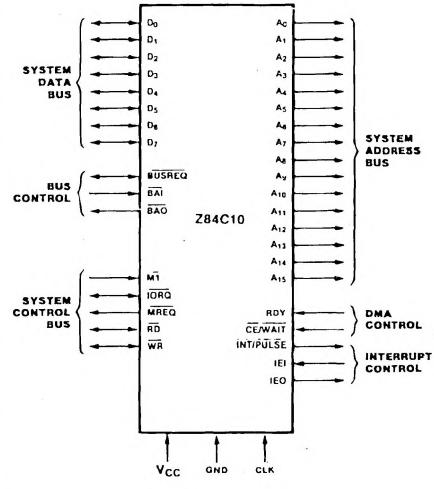
B
DIP-40
(Plastic)

D
DIP-40
(Ceramic)

C
PLCC44
(Plastic)

(Ordering Information at the end of the datasheet)

LOGIC FUNCTIONS



DESCRIPTION

The Z80C 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.

Transfers can be done between any two ports (source and destination), including memory-to I/O,

Figure 1 : Dual in Line Pin Configuration.

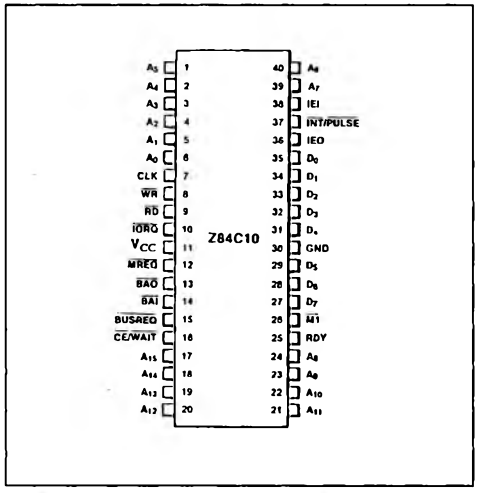
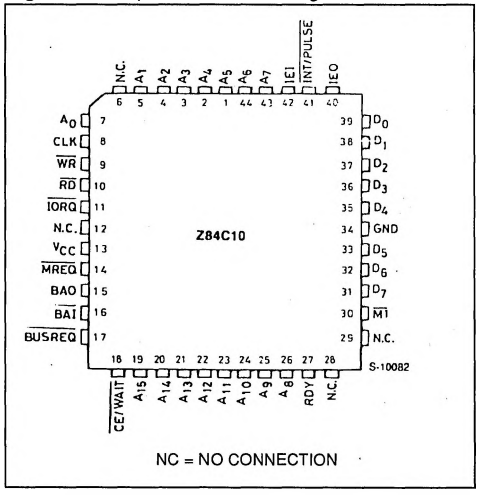


Figure 2 : Chip Carrier Pin Configuration.



NC = NO CONNECTION

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 Z80C 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 autorestart, 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 Z80C DMA is an n-channel silicon-gate depletion-load device and uses a single + 5 V power supply and the standard Z80C Family single-phase clock.

FUNCTIONAL DESCRIPTION

Classes of Operation. The Z80C 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. Data is read from one addressable port and written to the other addressable port, byte by byte. 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 2 M bytes per second can be obtained with the 4 MHz Z80C DMA or 3 M bytes per second with the 6 MHz Z80C DMA.

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

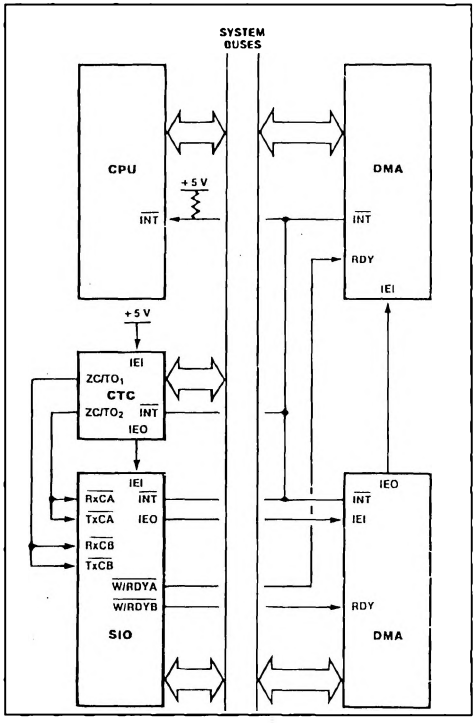
Data transfers or searches can be programmed to stop or interrupt under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

MODES OF OPERATION

The 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

Figure 3 : Typical Z80C Environment.



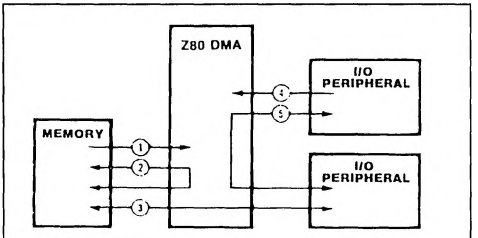
the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.

- *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 gain.

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

Figure 4 : Function of the Z80 DMA.



1. Search memory
2. Transfer memory-to-memory (optional search)
3. Transfer memory-to-I/O (optional search)
4. Search I/O
5. Transfer I/O-to-I/O (optional search)

block length (cont is N-1 where N is the block length).

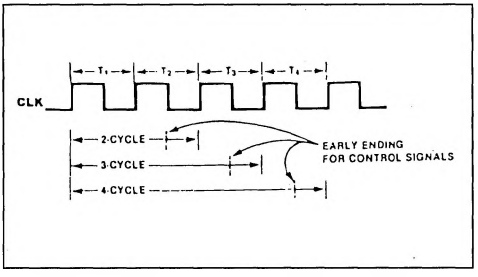
COMMANDS AND STATUS

The Z80C 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 writting the Read Status Byte command or the Initial 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, clear status bits and the like. 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

Figure 5 : Variable Cycle Lenght.



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 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 Z80C 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 Z80C DMA can be programmed to interrupt the CPU on three 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 Z80C CPU environment, the DMA passes 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 DESCRIPTIONS

A_0-A_{15} .*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.

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

$\overline{BAO}$.*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. $\overline{BAI}$ and $\overline{BAO}$ from daisy chain for multiple-DMA priority resolution over bus control.

$\overline{BUSREQ}$.*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 $\overline{BAI}$ and $\overline{BAO}$, 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

$\overline{CE}$/$\overline{WAIT}$.*Chip Enable and Wait* input, active Low). Normally this functions only as a $\overline{CE}$ line, but it can also be programmed to serve a $\overline{WAIT}$ function. As a $\overline{CE}$ line from the CPU, it becomes active when $\overline{WR}$ and $\overline{IORQ}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control or command bytes from the CPU to the DMA. As a $\overline{WAIT}$ 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.

$\overline{CLK}$.*System Clock* (input). Standard Z80 single-phase clock at 4.0 MHz (Z80CA DMA) or 6.0 MHz (Z80CB DMA). For slower system clocks, a TTL gate whith a pullup resistor may be adequate to meet the timing and voltage level specification. For higher-speed systems, use a clock driver with an active pullup to meet the V_{IH} specification and risetime requirements. In all cases there should be a resistive pullup to the power supply of 10 K ohms (max) to ensure proper power when the DMA is reset.

D_0-D_7 .*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 block lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

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

$\overline{IORQ}$.*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 $\overline{CE}$ pin and its $\overline{WR}$ or $\overline{RD}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the 8-bit or 16-bit address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{IORQ}$ and $\overline{M1}$ are both active simultaneously, an interrupt acknowledge is indicated.

$\overline{M1}$.*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, $\overline{M1}$ is active as each opcode byte is fetched. An interrupt acknowledge is indicated when both $\overline{M1}$ and $\overline{IORQ}$ are active.

$\overline{MREQ}$.*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.

$\overline{RD}$. *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 $\overline{\text{BUSREQ}}$ line to go Low or High.

$\overline{\text{WR}}$ *Write* (bidirectional, active Low, 3-state). As and 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 Z80C 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 Z80C CPU environment, the DMA can be tied directly to the analogous pins on the CPU (figure 7) with no additional buffering, except for the CE/$\overline{\text{WAIT}}$ 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 request, 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 register 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 register 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

Figure 6 : Block Diagram.

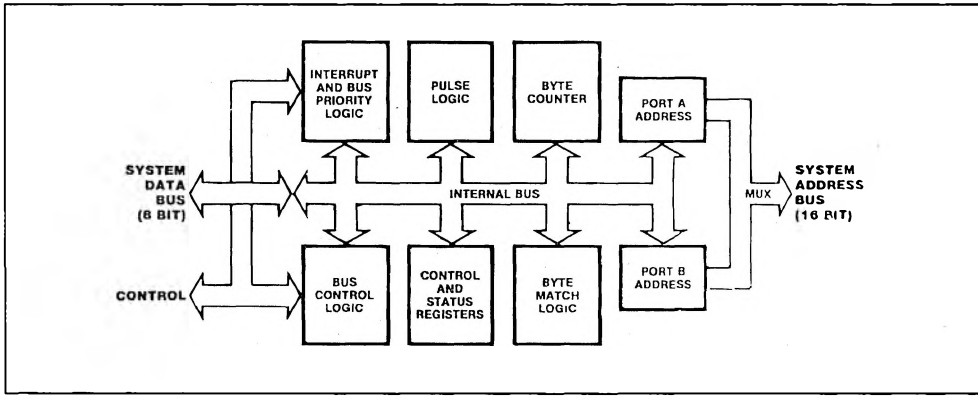
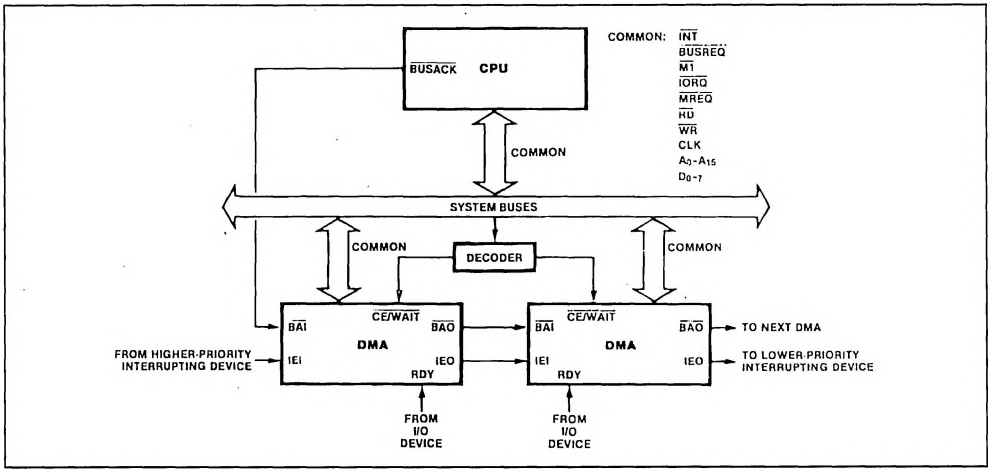


Figure 7 : Multiple-DMA Interconnection to the Z80 CPU.



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

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)

read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

In multiple-DMA configurations, interrupt request daisy chains are priorized by the order in which their IEI and IEO lines are connected. The system bus, however, may not be pre-empted.

Any DMA that gains access to the system bus keeps the bus until it is finished.

PROGRAMMING

The Z80C 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 Z80C CPU).

WRITING

Control or command bytes are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group.

ted registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and ponter 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 byte written to the DMA will be stored in these two registers, in that order.

READING

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 Z80C 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.

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 in WR6.
3. Declare Port A as source in WR0.
4. Load Port A address in 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 clock 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.

Figure 8a : Read Registers.

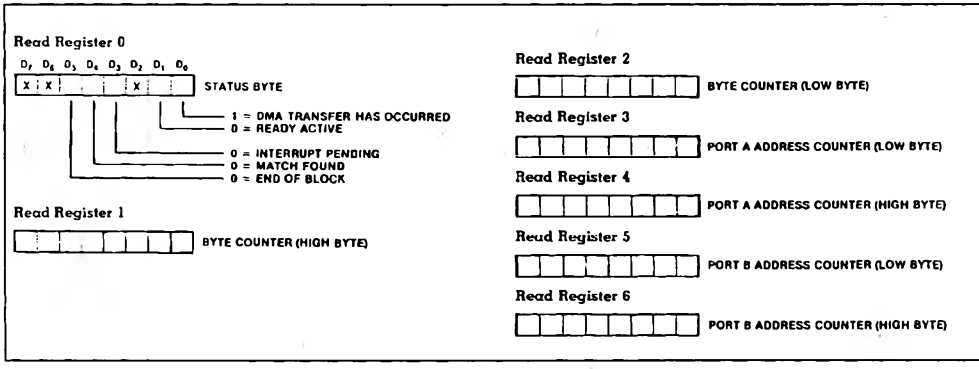


Figure 8b : Write Registers.

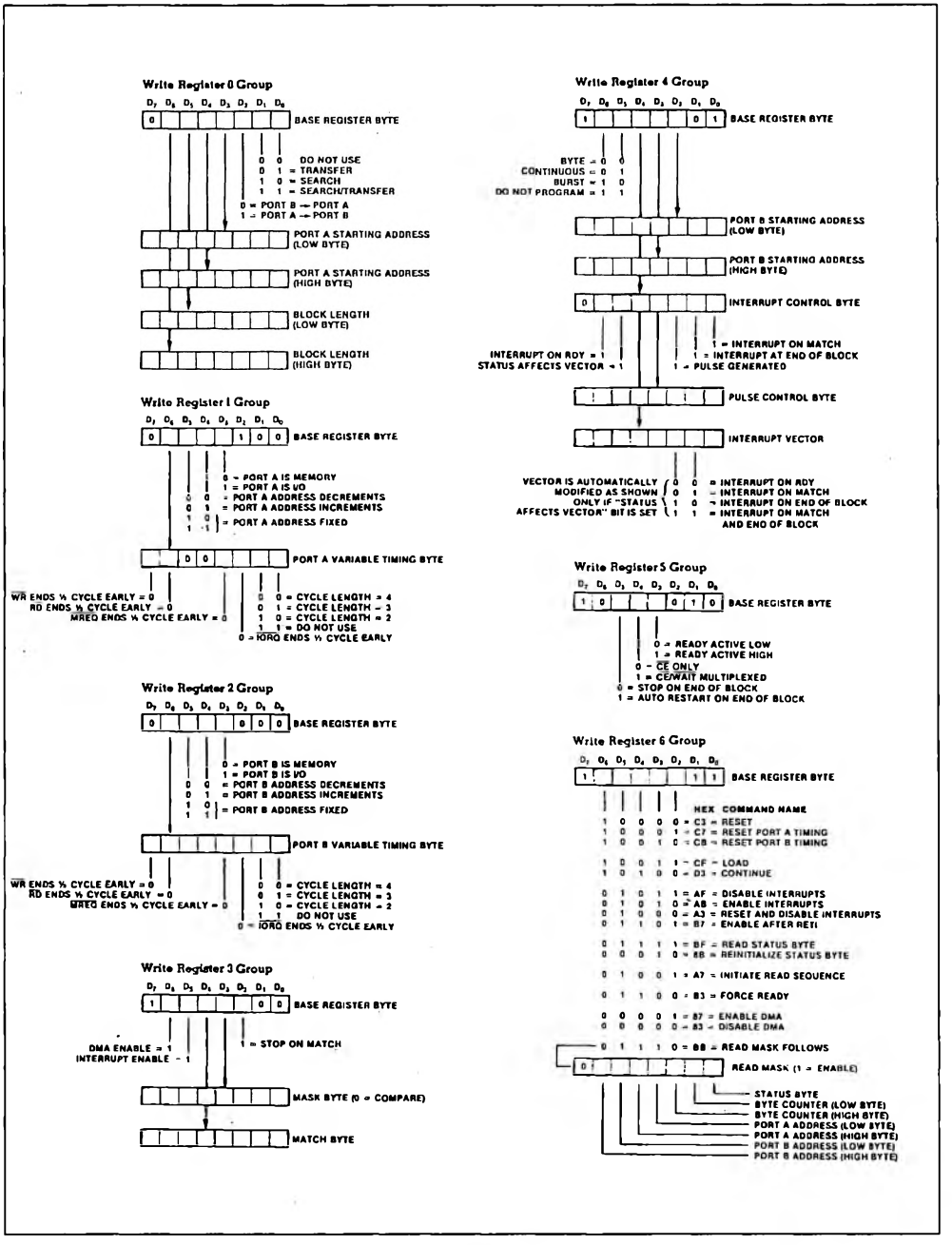


Figure 9 : Sample DMA Program.

Comments	D_7	D_6	D_5	D_4	D_3
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 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows
Port A Address (lower)	0	1	0	1	0
Port A Address (upper)	0	0	0	1	0
Block Length (lower)	0	0	0	0	0
Block Length (upper)	0	0	0	1	0
WR1 defines Port A as memory with fixed incrementing address.	0	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory
WR2 defines Port B as peripheral with fixed address.	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1	0	0 No Interrupt Control Byte Follows	0 No Upper Address
		Burst Mode			
Port B Address (lower)	0	0	0	0	0
WR5 Sets Ready Active High.	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High
WR6 Loads Port B Address and Resets Block Counter. *	1	1	0	0	1
WR0 Sets Port A as Source. *	0	0	0	0	0
		No Address or Block Length Bytes			
WR6 Loads Port A Address and Resets Block Counter.	1	1	0	0	1
WR6 Enables DMA to start operation.	1	0	0	0	

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 (continued).

Comments	D_2	D_1	D_0	HEX
WR0 sets DMA to receive block length. Port A starting address and temporarily sets Port B as source.	0 B → A Temporary for Loading B Address *	0	1	79
		Transfer, No Search		
Port A Address (lower)	0	0	0	50
Port A Address (upper)	0	0	0	10
Block Length (lower)	0	0	0	00
Block Length (upper)	0	0	0	10
WR1 defines Port A as memory with fixed incrementing address.	1	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	1	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1 Port B Lower Address Follows	0	1	C5
Port B Address (lower)	1	0	1	05
WR5 Sets Ready Active High.	0	1	0	8A
WR6 Loads Port B Address and Resets Block Counter. *	1	1	1	CF
WR0 Sets Port A as Source. *	1 A → B	0	1	05
		Transfer, No Search		
WR6 Loads Port A Address and Resets Block Counter.	1	1	1	CF
WR6 Enables DMA to start operation.	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.

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 CE, IORQ and 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.

ACTIVE STATE TIME (DMA as Bus Control-

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 Z80C 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.

Figure 12 illustrates the timing for memory-to-I/O port transfers and figure 13 illustrates I/O-to-memory transfers. Memory-to-memory and I/O-to-I/O transfers 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 between T_2 and T_3. If the CE/WAIT line is programmed to act a 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_W for I/O transactions. If CE/WAIT is Low during this time an-

Figure 10 : CPU-to-DMA Wrtite Cycle.

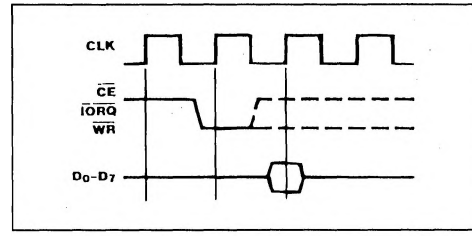


Figure 11 : CPU-to-DMA Read Cycle.

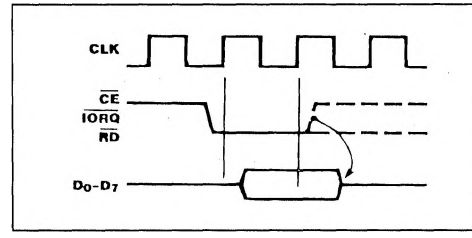
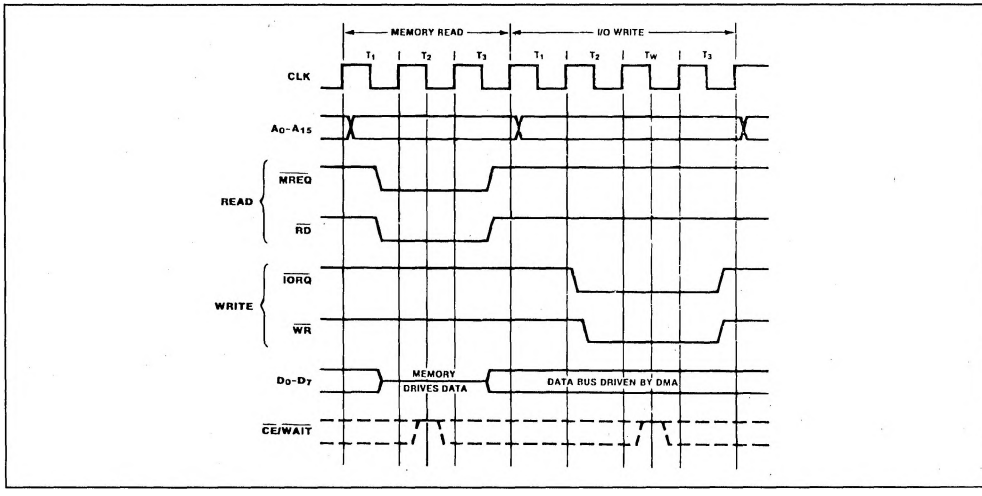


Figure 12 : Memory-to-I/O Transfer.



other T-cycle is added, during which the CE/WAIT line will again be sampled. The duration of transactions can thus be indefinitely extended.

VARIABLE CYCLE AN EDGE TIMING

The Z80C 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 of four-T-cycle (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 an 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 T_2 for 3- or 4-cycle memory cycles, and at the falling edge of T_3 for 4-cycle I/O cycles.

During transfers, data is latched on the clock edge causing the rising edge of RD and held through 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 detect 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 Z80C CPU is used since the CPU cannot 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 BUSREQ and BAI have returned High.

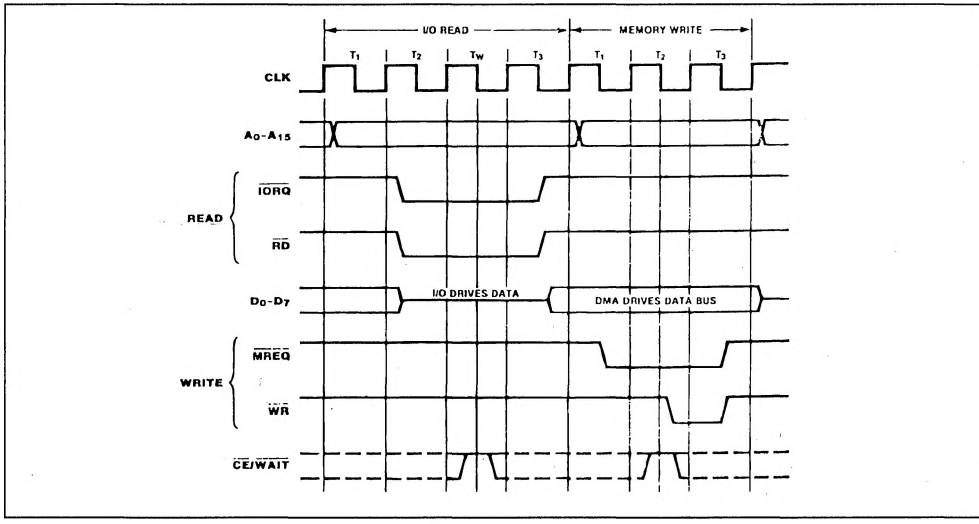
BUS RELEASE AT END OF BLOCK

In Burst and Continuos modes, an end of block causes 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 AND NOT READY

In Burst mode, when RDY goes inactive it causes BUSREQ to go High on the next rising edge of CLK after the completion of its current byte operation

Figure 13 : I/O -to-Memory Transfer.



(figure 18). The action on $\overline{BUSREQ}$ 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, $\overline{BUSREQ}$ is not released in Continuos 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 $\overline{BUSREQ}$ 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 sheme, 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 (See Applicatilonn Note "Z80 Family Interrupt Structure).

Interrupt on RDY (interrupt before requesting bus) does not directly affect the $\overline{BUSREQ}$ 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 reset the Interrupt Under Service latch in the Z80 DMA.

Figure 14 : Variable-Cycle and Edge Timing.

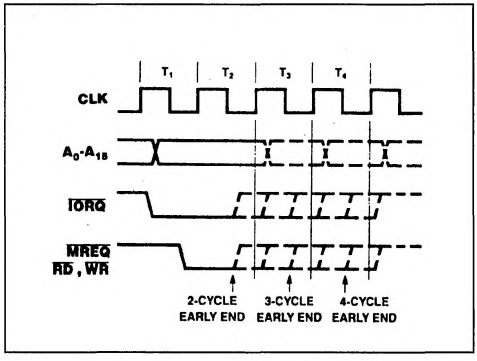


Figure 15 : Bus Request and Acceptance.

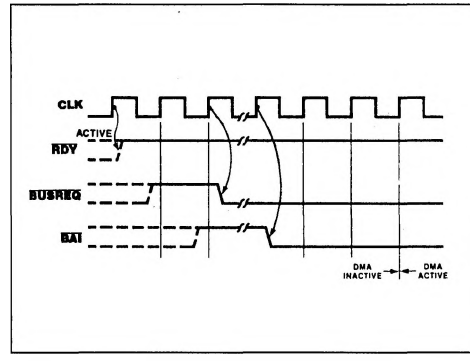


Figure 16 : Bus Release (Byte-at-a-Time-Mode).

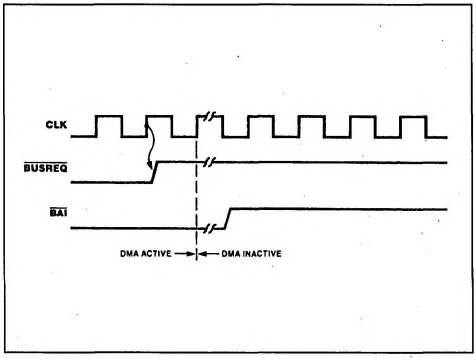


Figure 17 : Bus Release at End of Block (Burst and Continuous Modes).

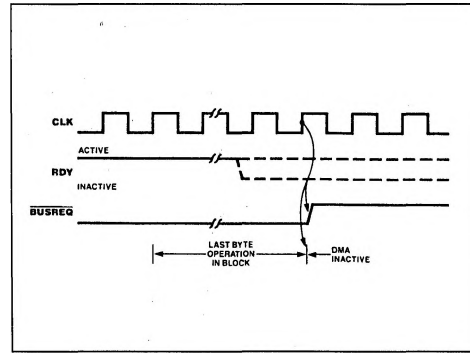


Figure 18 : Bus Release When Not Ready (Burst Mode).

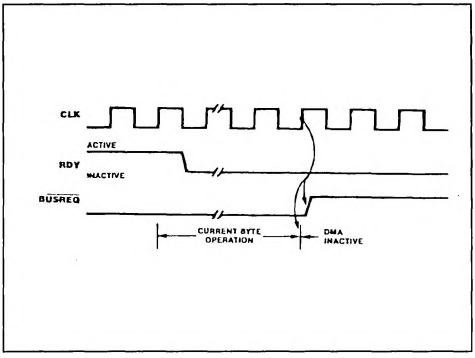
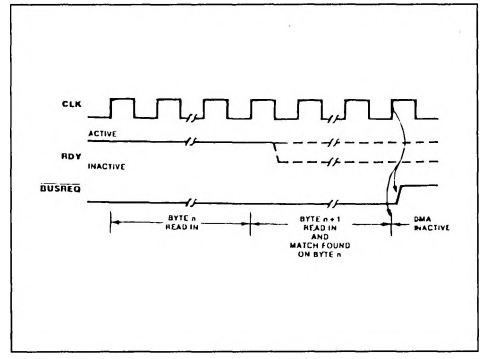


Figure 19 : Bus Release on Match (Burst and Continuous Modes).



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	V_{CC} Supply Voltage with Respect to V_{SS}	– 0.5 to 7	V
V_{IN}	Input Voltage	– 0.5 to V_{CC} + 0.5	V
P_D	Power Dissipation (T_A = 85 °C)	250	mW
T_{SOLDER}	Soldering Temperature (soldering time 10 sec)	260	°C
T_{stg}	Storage Temperature	– 65 to 150	°C
T_{opr}	Operating Temperature	– 40 to 85	°C

DC CHARACTERISTICS

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{ILC}	Clock Input Low Voltage		– 0.3		0.6	V
V_{IHC}	Clock Input High Voltage		V_{CC} – 0.6		V_{CC} + 0.3	V
V_{IL}	Input Low Voltage (except CLK)		– 0.5		0.8	V
V_{IH}	Input High Voltage (except CLK)		2.2		V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 3.2 mA for $\overline{BUSREQ}$ I_{OL} = 2.0 mA or all others			0.4	V
V_{OH1}	Output High Voltage (1)	I_{OH} = – 1.6 mA	2.4			V
V_{OH2}	Output High Voltage (2)	I_{OH} = – 250 µA	V_{CC} – 0.8			V
I_{LI}	Input Leakage Current	$V_{SS} \le V_{IN} \le V_{CC}$			± 10	µA
I_{LO}	3-State Output Leakage Current in Float	V_{SS} + 0.4 ≤ $V_{IN} \le V_{CC}$			± 10	µA
I_{CC1}	Operating Supply Current : 4 MHz 6 MHz	 V_{CC} = 5 V, fCKL = 1/T_CC(min) $V_{IH} = V_{CC}$ – 0.2 V, V_{IL} = 0.2 V		 5 6	 7 10	 mA mA
I_{CC2}	Standby Supply Current	V_{CC} = 5 V, $V_{IH} = V_{CC}$ – 0.2 V V_{IL} = 0.2 V			10	µA

TEST CONDITIONS

T_A = – 40 °C to + 85 °C

V_{CC} = 5 V ± 10 %

V_{SS} = 0 V

AC TEST CONDITIONS

Inputs except CLK (clock) are driven at 2.4 V for a logic "1" and 0.4 V for a logic "0". Clock input is driven at V_{CC} – 0.6 V for a logic "1" and 0.6 V for a logic "0".

Timing measurements are made at 2.2 V for a logic "1" and 0.8 V for a logic "0".

All AC parameters assume a load capacitance of 100 pF.

CAPACITANCE

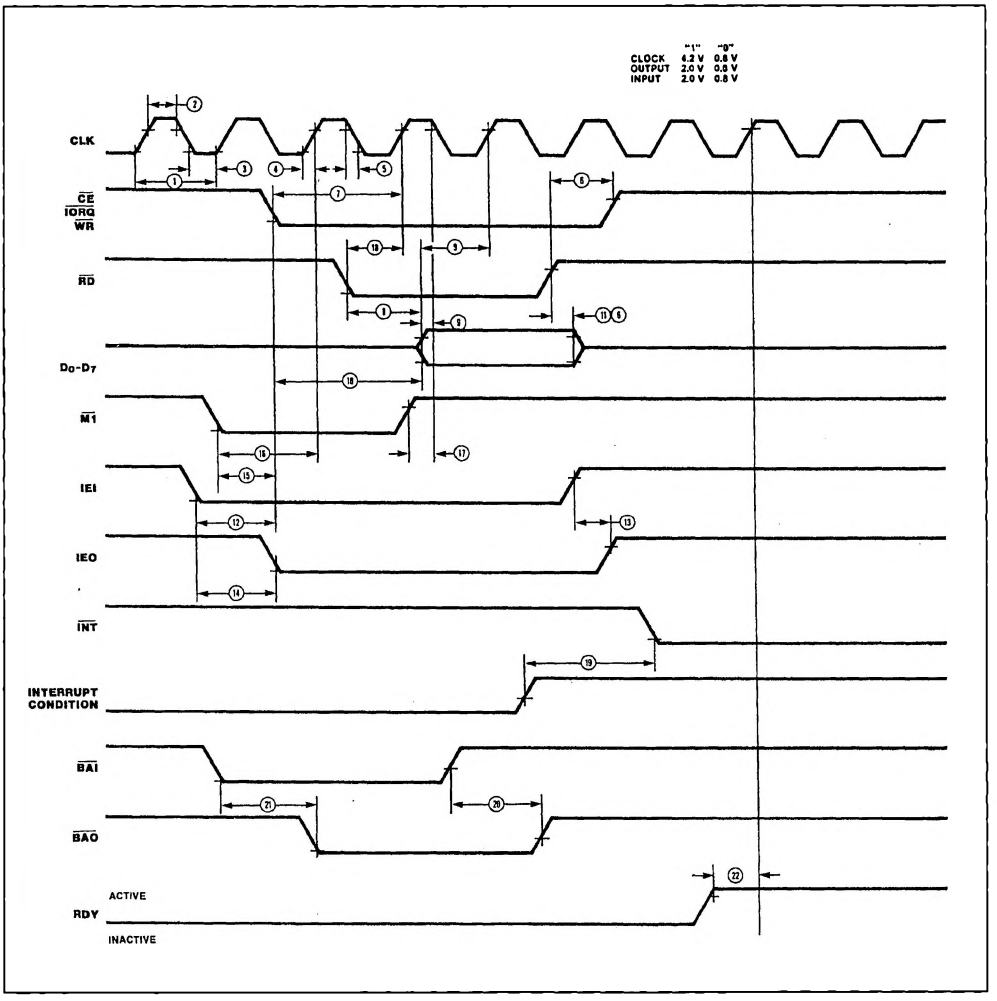
Symbol	Parameter	Test Conditions	Min.	Max.	Unit
C	Clock Capacitance	Unmeasured Pins		5	pF
C_{IN}	Input Capacitance	Returned to Ground		5	pF
C_{OUT}	Output Capacitance			5	pF

INACTIVE STATE AC CHARACTERISTICS

N°	Symbol	Parameter	Z84C10A		Z84C10B	
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)
1	TcC	Clock Cycle Time	250	DC	165	DC
2	TwCh	Clock Width (high)	110	DC	65	DC
3	TwCl	Clock Width (low)	110	DC	65	DC
4	TrC	Clock Rise Time		30		20
5	TfC	Clock Fall Time		30		20
6	Th	Hold Time for any Specified Setup Time	0		0	
7	TsC(Cr)	$\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$ ↓ to Clock ↑ Setup	145		60	
8	TdDO(RDf)	$\overline{RD}$ ↓ to Data Output Delay		380		300
9	TsWM(Cr)	Data in to Clock ↑ Setup ($\overline{WR}$ or $\overline{MI}$)	50		30	
10	TdCf(DO)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA cycle)		160		100
11	TdRD(Dz)	$\overline{RD}$ ↑ to Data Float Delay (output buffer disable)		110		70
12	TsIEI(IORQ)	IEI ↓ to $\overline{IORQ}$ ↓ Setup (INTA cycle)	140		100	
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		160		70
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		130		70
15	TdMI(IEO)	$\overline{MI}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{MI}$ ↓)		190		100
16	TsMIf(Cr)	$\overline{MI}$ ↓ to Clock ↑ Setup	90		70	
17	TsMIr(Cf)	$\overline{MI}$ ↑ to Clock ↓ Setup	– 10		– 10	
18	TsRD(Cr)	$\overline{RD}$ ↓ to Clock ↑ Setup ($\overline{MI}$ cycle)	115		60	
19	TdI(INT)	Interrupt Cause to $\overline{INT}$ ↓ Delay ($\overline{INT}$ generated only when DMA is inactive)		500		450
20	TdBAIr(BAOr)	$\overline{BAI}$ ↑ to $\overline{BAO}$ ↑ Delay		150		100
21	TdBAIf(BAOf)	$\overline{BAI}$ ↓ to $\overline{BAO}$ ↓ Delay		150		100
22	TsRDY(Cr)	RDY Active to Clock ↑ Setup	100		50	

Note : 1. Negative minimum setup values mean that the first mentioned event can come after the second mentioned event.

INACTIVE STATE AC CHARACTERISTICS (continued).



ACTIVE STATE AC CHARACTERISTICS

N°	Symbol	Parameter	Z84C10A		Z84C10B	
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)
1	TcC	Clock Cycle Time	250	DC	165	DC
2	TwCh	Clock Width (high)	110	DC	65	DC
3	TwCl	Clock Width (low)	110	DC	65	DC
4	TrC	Clock Rise Time		30		20
5	TfC	Clock Fall Time		110		90
6	TdA	Address Output Delay		110		90
7	TdC(Az)	Clock ↑ to Address Float Delay		90		80
8	TsA(MREQ)	Address to $\overline{MREQ}$ ↓ Setup (memory cycle)	65		35	
9	TsA(IRW)	Address Stable to $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ ↓ Setup (I/O cycle)	180		110	
*10	TdRW(A)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Stable Delay	90		35	
*11	TdRW(Az)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Float	95		65	
12	TdCf(DO)	Clock ↓ to Data Out Delay		150		130
*13	TdCr(Dz)	Clock ↑ to Data Float Delay (write cycle)		90		70
14	TsDI(Cr)	Data in to Clock ↑ Setup (read cycle when rising edge ends read)	35		30	
15	TsDI(Cf)	Data in to Clock ↓ Setup (read cycle when falling edge ends read)	50		40	
*16	TsDO(WfM)	Data out to $\overline{WR}$ ↓ Setup (memory cycle)	80		25	
17	TsDO(WfI)	Data Out to $\overline{WR}$ ↓ Setup (I/O cycle)	100		55	
*18	TdWr(DO)	$\overline{WR}$ ↑ to Data Out Delay	70		30	
19	Th	Hold Time for Any Specified Setup Time	0		0	
20	TdCr(Mf)	Clock ↑ to $\overline{MREQ}$ ↓ Delay		85		70
21	TdCf(Mf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		85		70
22	TdCr(Mr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay		85		70
23	TdCf(Mr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		85		70
24	TwMl	$\overline{MREQ}$ Low Pulse Width	220		135	
*25	TwMh	$\overline{MREQ}$ High Pulse Width	120		65	
26	TdCf(If)	Clock ↓ to $\overline{IORQ}$ ↓ Delay		85		70
27	TdCr(If)	Clock ↑ to $\overline{IORQ}$ ↓ Delay		75		65
28	TdCr(Ir)	Clock ↑ to $\overline{IORQ}$ ↑ Delay		85		70
*29	TdCf(Ir)	Clock ↓ to $\overline{IORQ}$ ↑ Delay		85		70

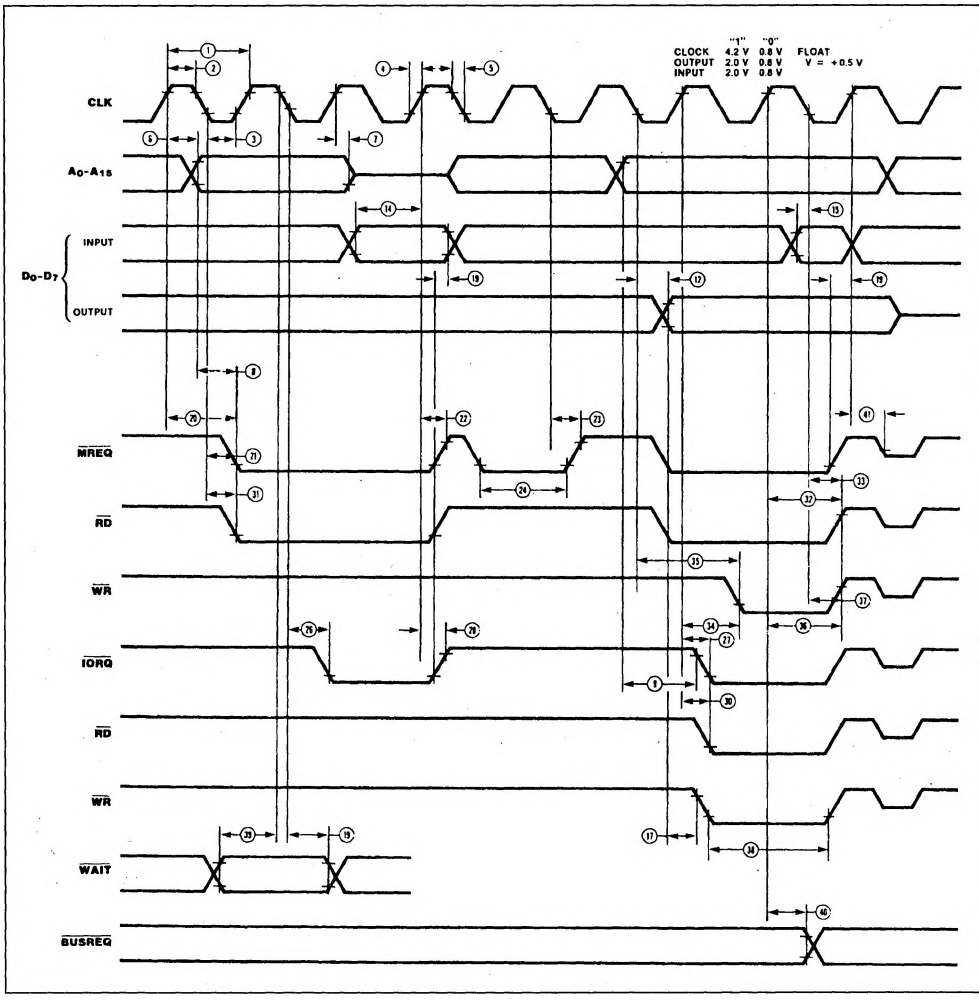
Notes : 1. Numbers in parentheses are other parameter-numbers in this table ; their values should be substituted in equations.
2. All equations imply DMA default (standart) timing.
3. Data must be enabled onto data bus when $\overline{RD}$ is active.
4. Asterisk (*) before parameter number means the parameter is not illustrated in the AC Timing Diagrams.

ACTIVE STATE AC CHARACTERISTICS (continued)

N°	Symbol	Parameter	Z84C10A		Z84C10B	
			Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)
30	TdCr(Rf)	Clock ↑ to $\overline{RD}$ ↓ Delay		85		70
31	TdCf(Rf)	Clock ↓ to $\overline{RD}$ ↓ Delay		95		80
32	TdCr(Rr)	Clock ↑ to $\overline{RD}$ ↓ Delay		85		70
33	TdCf(Rr)	Clock ↓ to $\overline{RD}$ ↑ Delay		85		70
34	TdCr(Wf)	Clock ↑ to $\overline{WR}$ ↓ Delay		65		60
35	TdCf(Wf)	Clock ↓ to WR ↓ Delay		80		60
36	TdCr(Wr)	Clock ↑ to $\overline{WR}$ ↑ Delay		80		70
37	TdCf(Wr)	Clock ↓ to $\overline{WR}$ ↑ Delay		80		70
38	TwWl	$\overline{WR}$ Low Pulse Width	220		135	
39	TsWA(Cf)	$\overline{WAIT}$ to Clock ↓ Setup	70		60	
40	TdCr(B)	Clock ↑ to $\overline{BUSREQ}$ Delay		100		90
41	TdCr(Iz)	Clock ↑ to $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$ Float Delay		80		70

Notes : 1. Numbers in parentheses are other parameter-numbers in this table ; their values should be substituted in equations.
2. All equations imply DMA default (standart) timing.
3. Data must be enabled onto data bus when $\overline{RD}$ is active.
4. Asterisk (*) before parameter number means the parameter is not illustrated in the AC tTiming Diagrams.

ACTIVE STATE AC CHARACTERISTICS (continued).



ORDERING INFORMATION

Type	Package	Temp.	Clock	Description
Z84C10AB6	DIP–40 (plastic)	– 40/ + 85 °C	4 MHz	Z80C Direct Memory Access Unit
Z84C10AC6	PLCC44 (plastic chip-carrier)	– 40/ + 85 °C		
Z84C10BB6	DIP–40 (plastic)	– 40/ + 85 °C	6 MHz	
Z84C10BC6	PLCC44 (plastic chip-carrier)	– 40/ + 85 °C		