1981 Data Book

Microcomputer Components Data Book

Microcomputers in Every Form

Zilog offers microcomputers in every form: from components and development systems to board-level products and complete general-purpose microcomputer systems. This edition of the *Zilog Data Book* describes Zilog components, development systems, and microcomputer boards. You'll also find a section on the in-depth training courses now offered about most Zilog products.

Zilog components, the basic building blocks for our other microcomputer products, include the 8-bit Z80® Microprocessor and its family of intelligent peripherals, the Z8™ Family of Single-Chip Microcomputers, and the 16-bit Z8000™ Microprocessor and its family of intelligent peripherals.

Zilog offers a wide variety of development environments, ranging from the inexpensive Z8 and Z8000 Development Modules to the more elaborate PDS 8000 and ZDS-1 Development Systems to the ultra-sophisticated multi-user Z-LAB 8000 Development System. In addition, the Z-SCAN 8000 provides in-circuit emulation for both the Z8001 and Z8002 Microprocessors.

Our Z80 MCB Board Family offers a complete solution for prototype and production designs in which you don't want to design a microcomputer from scratch. This well-established family includes a Z80 CPU board, several types of memory boards, and boards for all types of digital and analog I/O. A complete set of card cages, enclosures, and other accessories makes this family easy to use.

The card at the beginning of the data book allows you to register for an on-going program to keep you informed of the latest developments at Zilog. New information will be published as "stand-alone" data sheets, also in this convenient 7" × 9" size. If you are interested in receiving this information as well as a handy binder to hold it in, simply fill out the card and return it to us.

Table of Contents

Table of Contents (Continued)

Z80 Family

Zilog

Zilog Z80® Family

Zilog

Faster Z80B Peripheral Controllers

March 1981

Zilog has become an industry leader, thanks to innovation in microcomputer concepts and integrated design exemplified in the Z80 Family of microcomputer products.

At Zilog, innovation means using proven, sophisticated mainframe and minicomputer concepts and translating them into the latest LSI technologies. Integration means more than designing an ever-greater number of functions onto a single chip. Zilog integrates technologies—LSI design enhanced by advances in computer-based system architecture and system design technologies.

Zilog offers microprocessor solutions to computing problems: from components and development systems to OEM board-level products and general-purpose microcomputer systems.

This guide to the Z80 Family of state-of-the-art microprocessors and intelligent peripheral controllers demonstrates Zilog's continued support for the Z80 microprocessor and the other members of the Z80 product family—a family first introduced in 1976 that continues to enjoy growing customer support while family chips are upgraded to newer and ever-higher standards.

The design philosophy of all Z80 Family members is to help engineers design microcomputer systems with fewer components that have more functions per chip. The Z80 CPU offers many more features and functions than its competitor.

The **Z8400 Z80 CPU Central Processing Unit** has rapidly established itself as the most sophisticated, most powerful, and most versatile 8-bit microprocessor in the world. In addition to being source-code compatible with the 8080A microprocessor, the Z80 offers more instructions than the 8080A (158 vs. 78) and numerous other features that simplify hardware requirements and reduce programming effort while increasing throughput. The dual-register set of the Z80 CPU allows high-speed context switching and more efficient interrupt processing. Two index registers give additional memory-addressing flexibility and simplify the task of programming. Interfacing to dynamic memory is simplified by on-chip, programmable refresh logic. Block moves plus string- and bit-manipulation instructions reduce programming effort, program size, and execution time.

The four traditional functions of a microcomputer system (parallel I/O, serial I/O, counting/timing, and direct memory access) are easily implemented by the Z80 CPU and the following well-proven family of Z80 peripheral devices: Z80 PIO, Z80 SIO, Z80 DART, Z80 CTC, and Z80 DMA.

The easily programmed, dual-channel **Z8420 Z80 PIO Parallel Input/Output Controller** offers two 8-bit I/O ports with individual handshake and pattern recognition logic. Both I/O ports operate in either a byte or a bit mode. In addition, this device can be programmed to generate interrupts for various status conditions.

All common data communications protocols, asynchronous as well as synchronous, are remarkably well handled by the **Z8440 Z80 SIO Serial Input/Output Controller.** This dual-channel receiver/transmitter device offers on-chip parity and CRC generation/checking. FIFO buffering and flag- and frame-detection generation logic are also offered.

If asynchronous-only applications are required, the cost-effective **Z8470 Z80 DART Dual Asynchronous Receiver/Transmitter** can be used in place of the Z80 SIO. The Z80 DART offers all Z80 SIO asynchronous features in two channels.

Timing and event-counting functions are the forte of the **Z8430 Z80 CTC Counter/Timer Controller.** The CTC provides four counters, each with individually programmable prescalers. The CTC is a convenient source of programmable clock rates for the SIO.

With the **Z8410 Z80 DMA Direct Memory Access Controller,** data can be transferred directly between any two ports (typically, I/O and memory). The DMA transfers, searches, or search/transfers data in Byte-by-Byte, Burst, or Continuous modes. This device can achieve an impressive 2M bits per second data rate in the Search mode.

Z8400
Z80® CPU Central Processing Unit

Zilog

Product Specification

March 1981

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

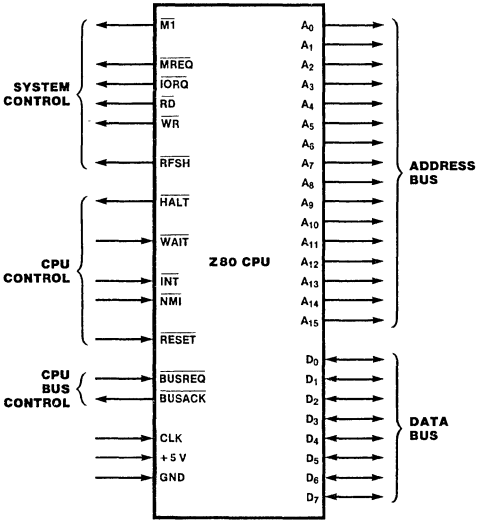


Figure 1. Pin Functions

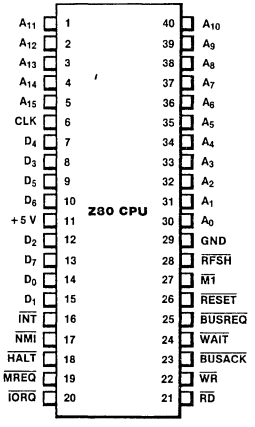


Figure 2. Pin Assignments

General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

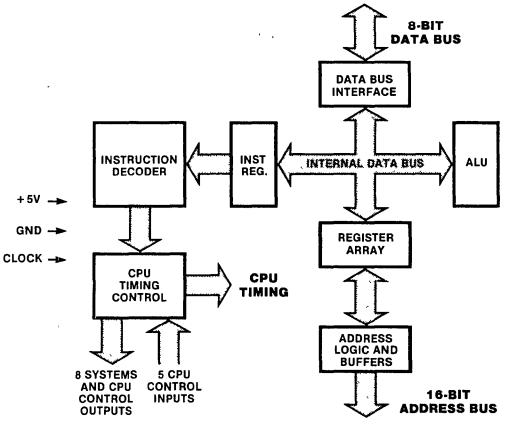


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

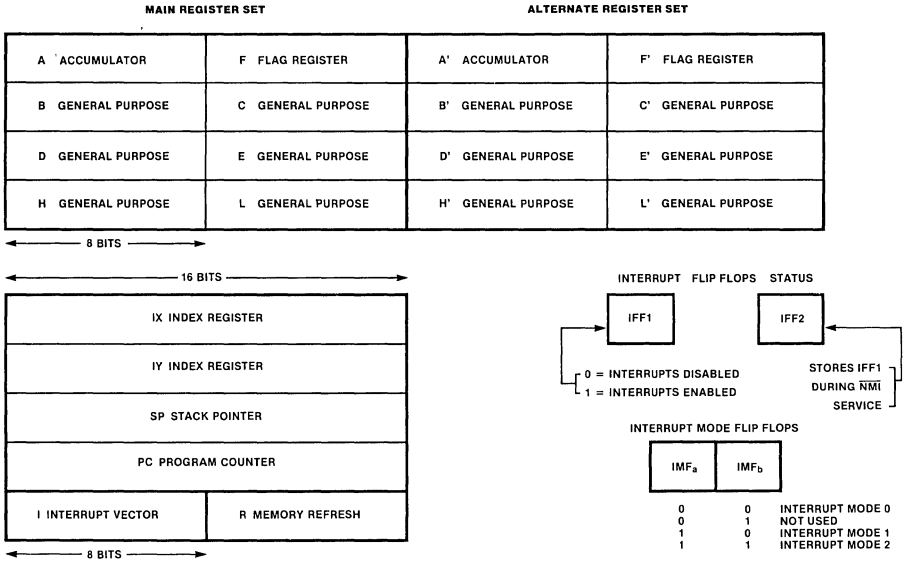


Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The $\overline{NMI}$ is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt since it requires that interrupts be enabled in software in order to operate. Either $\overline{NMI}$ or $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

Mode 0 Interrupt Operation. This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus, which is then acted on six times by the CPU. This is normally a Restart Instruction, which will initiate an unconditional jump to the selected one of eight restart locations in page zero of memory.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit address vector on the data bus during the interrupt acknowledge cycle. The high-order byte of the interrupt service routine address is supplied by the I (Interrupt) register. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwared to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	IFF_2 → Parity flag
LD A,R instruction execution	•	•	IFF_2 → Parity flag
Accept $\overline{NMI}$	0	IFF_1	IFF_1 → IFF_2 (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	IFF_2 → IFF_1 at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- ☐ 8-bit loads
- ☐ 16-bit loads
- ☐ Exchanges, block transfers, and searches
- ☐ 8-bit arithmetic and logic operations
- ☐ General-purpose arithmetic and CPU control
- ☐ 16-bit arithmetic operations
- ☐ Rotates and shifts
- ☐ Bit set, reset, and test operations
- ☐ Jumps
- ☐ Calls, returns, and restarts
- ☐ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- ☐ Immediate
- ☐ Immediate extended
- ☐ Modified page zero
- ☐ Relative
- ☐ Extended
- ☐ Indexed
- ☐ Register
- ☐ Register indirect
- ☐ Implied
- ☐ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000 B 001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 101 ← d →	DD	3	5	19	011 E 100 H 101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111 A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10	
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9	

NOTES r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop, (IFF) is copied into the P/V flag
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 001 21 ← n → ← n →	4	4	14	
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 001 21 ← n → ← n →	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 2A ← n → ← n →	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 ED 01 dd1 011 ← n → ← n →	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 010 2A ← n → ← n →	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 010 2A ← n → ← n →	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 22 ← n → ← n →	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 ED 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 010 22 ← n → ← n →	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 010 22 ← n → ← n →	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001 F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 FD 11 111 001 F9	2	2	10	
PUSH qq	(SP−2) ← qq_L (SP−1) ← qq_H SP → SP −2	•	•	X	•	X	•	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP−2) ← IX_L (SP−1) ← IX_H SP → SP −2	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 101 E5	2	4	15	
PUSH IY	(SP−2) ← IY_L (SP−1) ← IY_H SP → SP −2	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 101 E5	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP) SP → SP +2	•	•	X	•	X	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP) SP → SP +2	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 001 E1	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP) SP → SP +2	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 001 E1	2	4	14	

NOTES dd is any of the register pairs BC, DE, HL, SP.
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively,
e g , BC_L = C, AF_H = A

Exchange, Block Transfer, Block Search Groups

EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000 08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011 E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC−1	•	•	X	0	X	①	0	•	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC−1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
					①							
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X 0 X	↕	0	•	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X 0 X	0	0	•	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
			②		①							
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	↕	X ↕ X	↕	1	•	11 101 101 ED 10 100 001 A1	2	4	16	
			②		①							
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕	X ↕ X	↕	1	•	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
			②		①							
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	↕	↕	X ↕ X	↕	1	•	11 101 101 ED 10 101 001 A9	2	4	16	
			②		①							
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕	X ↕ X	↕	1	•	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r	↕	↕	X ↕ X	V	0	↕	10 [000] r	1	1	4	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X ↕ X	V	0	↕	11 [000] 110 ← n →	2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X ↕ X	V	0	↕	10 [000] 110	1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X ↕ X	V	0	↕	11 011 101 DD 10 [000] 110 ← d →	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X ↕ X	V	0	↕	11 111 101 FD 10 [000] 110 ← d →	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X ↕ X	V	0	↕	[001]				s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction The indicated bits replace the [000] in the ADD set above
SUB s	A ← A − s	↕	↕	X ↕ X	V	1	↕	[010]				
SBC A, s	A ← A − s − CY	↕	↕	X ↕ X	V	1	↕	[011]				
AND s	A ← A ∧ s	↕	↕	X 1 X	P	0	0	[100]				
OR s	A ← A ∨ s	↕	↕	X 0 X	P	0	0	[110]				
XOR s	A ← A ⊕ s	↕	↕	X 0 X	P	0	0	[101]				
CP s	A − s	↕	↕	X ↕ X	V	1	↕	[111]				
INC r	r ← r + 1	↕	↕	X ↕ X	V	0	•	00 r [100]	1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X ↕ X	V	0	•	00 110 [100]	1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X ↕ X	V	0	•	11 011 101 DD 00 110 [100] ← d →	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X ↕ X	V	0	•	11 111 101 FD 00 110 [100] ← d →	3	6	23	
DEC m	m ← m − 1	↕	↕	X ↕ X	V	1	•	[101]				m is any of r, (HL), (IX + d), (IY + d) as shown for INC DEC same format and states as INC Replace [100] with [101] in opcode

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands.	↕	↕	X	↕	X	P	•	↕	00 100 111 27	1	1	4	Decimal adjust accumulator.
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111 2F	1	1	4	Complement accumulator (one's complement).
NEG	$A \leftarrow 0 - A$	↕	↕	X	↕	X	V	1	↕	11 101 101 ED 01 000 100 44	2	2	8	Negate acc (two's complement)
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111 3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111 37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000 00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110 76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011 F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011 FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 110 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 ED 01 010 110 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 ED 01 011 110 5E	2	2	8	

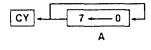
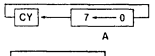
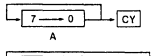
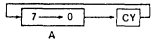
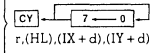
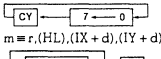
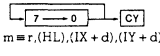
NOTES IFF indicates the interrupt enable flip flop
CY indicates the carry flip flop
★ indicates interrupts are not sampled at the end of EI or DI

16-Bit Arithmetic Group

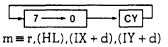
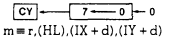
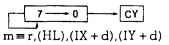
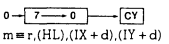
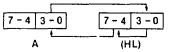
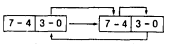
Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001	1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 ED 01 ss1 010	2	4	15	
SBC HL, ss	HL ← HL − ss − CY	↕	↕	X	X	X	V	1	↕	11 101 101 ED 01 ss0 010	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 DD 01 pp1 001	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 FD 00 rr1 001	2	4	15	rr Reg 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 011 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 011 23	2	2	10	
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 011 2B	2	2	10	
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 011 2B	2	2	10	

NOTES ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111 07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	↕	00 010 111 17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111 0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111 1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 r	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 110	2	4	15	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 DD 11 001 011 CB ← d → 00 000 110	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 FD 11 001 011 CB ← d → 00 000 110	4	6	23	
RL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	010				Instruction format and states are as shown for RLC's To form new opcode replace 000 or RLC's with shown code
RRC m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	001				

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RR m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	011					
SLA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	100					
SRA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	101					
SRL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	111					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL)
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
BIT b, r	$Z \leftarrow \overline{r_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)b	$Z \leftarrow \overline{(IX+d)_b}$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)b	$Z \leftarrow \overline{(IY+d)_b}$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 ← d → 11 b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 ← d → 11 b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	10					To form new opcode replace 11 of SET b, s with 10 Flags and time states for SET instruction

NOTES The notation m_b indicates bit b (0 to 7) of location m

Jump Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e − 2 →	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e − 2 →	38	2	2	7	If condition not met
	If C = 1, PC ← PC + e											2	3	12	If condition is met.
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e − 2 →	30	2	2	7	If condition not met
	If C = 0, PC ← PC + e											2	3	12	If condition is met
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e − 2 →	28	2	2	7	If condition not met
	If Z = 1, PC ← PC + e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e − 2 →	20	2	2	7	If condition not met
	If Z = 0, PC ← PC + e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 FD 11 101 001 E9	2	2	8	
DJNZ, e	B ← B−1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 10 ← e−2 →	2	2	8	If B = 0.
	If B ≠ 0, PC ← PC+e										2	3	13	If B ≠ 0.

NOTES e represents the extension in the relative addressing mode.
e is a signed two's complement number in the range < −126, 129 >
e−2 in the opcode provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e

Call and Return Group

CALL nn	(SP−1) ← PC_H (SP−2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 CD ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 ← n → ← n →	3	3	10	If cc is false
											3	5	17	If cc is true.
RET	PC_L ← (SP) PC_H ← (SP+1)	•	•	X	•	X	•	•	•	11 001 001 C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000	1	1	5	If cc is false
											1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 001 101 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 101 45	2	4	14	
RST p	(SP−1) ← PC_H (SP−2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 t 111	1	3	11	

cc	Condition
000	NZ non-zero
001	Z zero
010	NC non-carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

NOTE [1]RETN loads IFF_2 → IFF_1

Input and Output Group

IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 DB ← n →	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 ED 01 r 000	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B−1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 100 010 A2	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B−1 HL ← HL + 1 Repeat until B = 0	X	1 ①	X	X	X	X	1	•	11 101 101 ED 10 110 010 B2	2	5 (If B≠0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B=0)	16	
IND	(HL) ← (C) B ← B−1 HL ← HL−1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 101 010 AA	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B−1 HL ← HL−1 Repeat until B = 0	X	1 ①	X	X	X	X	1	•	11 101 101 ED 10 111 010 BA	2	5 (If B≠0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B=0)	16	
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 D3 ← n →	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	• ①	X	•	X	•	•	•	11 101 101 ED 01 r 001	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B−1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 100 011 A3	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B−1 HL ← HL + 1 Repeat until B = 0	X	1 ①	X	X	X	X	1	•	11 101 101 ED 10 110 011 B3	2	5 (If B≠0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B=0)	16	
OUTD	(C) ← (HL) B ← B−1 HL ← HL−1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 101 011 AB	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE ① If the result of B−1 is zero the Z flag is set, otherwise it is reset

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
OTDR	(C) ← (HL) B ← B−1 HL ← HL−1 Repeat until B = 0	X	1	X	X	X	1	•	11 101 101 ED 10 111 011	2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

Summary of Flag Operation

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s, ADC A, s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s, XOR s	↕	↕	X	0	X	P	0	0	Logical operations
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement
ADD DD, ss	•	•	X	X	X	•	0	↕	16-bit add
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL m, RLC m, RR m; RRC m, SLA m, SRA m; SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI, IND, OUTI, OUTD	X	↕	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
INIR; INDR, OTIR, OTDR	X	1	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
LDI, LDD	X	X	X	0	X	↕	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
LDIR, LDDR	X	X	X	0	X	0	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	X	↕	X	X	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LD A, I, LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care"
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{\text{BUSACK}}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{\text{BUSREQ}}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{\text{NMI}}$ and is always recognized at the end of the current machine cycle. $\overline{\text{BUSREQ}}$ forces the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ to go to a high-impedance state so that other devices can control these lines. $\overline{\text{BUSREQ}}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{\text{BUSREQ}}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{\text{HALT}}$. *Halt State* (output, active Low). $\overline{\text{HALT}}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{\text{INT}}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{\text{INT}}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{\text{IORQ}}$. *Input/Output Request* (output, active Low, 3-state). $\overline{\text{IORQ}}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{\text{IORQ}}$ is also generated concurrently with $\overline{\text{M1}}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\text{M1}}$. *Machine Cycle One* (output, active Low). $\overline{\text{M1}}$, together with $\overline{\text{MREQ}}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{\text{M1}}$, together with $\overline{\text{IORQ}}$, indicates an interrupt acknowledge cycle.

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

$\overline{\text{NMI}}$. *Non-Maskable Interrupt* (input, active Low). $\overline{\text{NMI}}$ has a higher priority than $\overline{\text{INT}}$. $\overline{\text{NMI}}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{\text{RD}}$. *Memory Read* (output, active Low, 3-state). $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\text{RESET}}$. *Reset* (input, active Low). $\overline{\text{RESET}}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{\text{RESET}}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{\text{RFSH}}$. *Refresh* (output, active Low). $\overline{\text{RFSH}}$, together with $\overline{\text{MREQ}}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{\text{WAIT}}$. *Wait* (input, active Low). $\overline{\text{WAIT}}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{\text{WAIT}}$ periods can prevent the CPU from refreshing dynamic memory properly.

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

CPU Timing

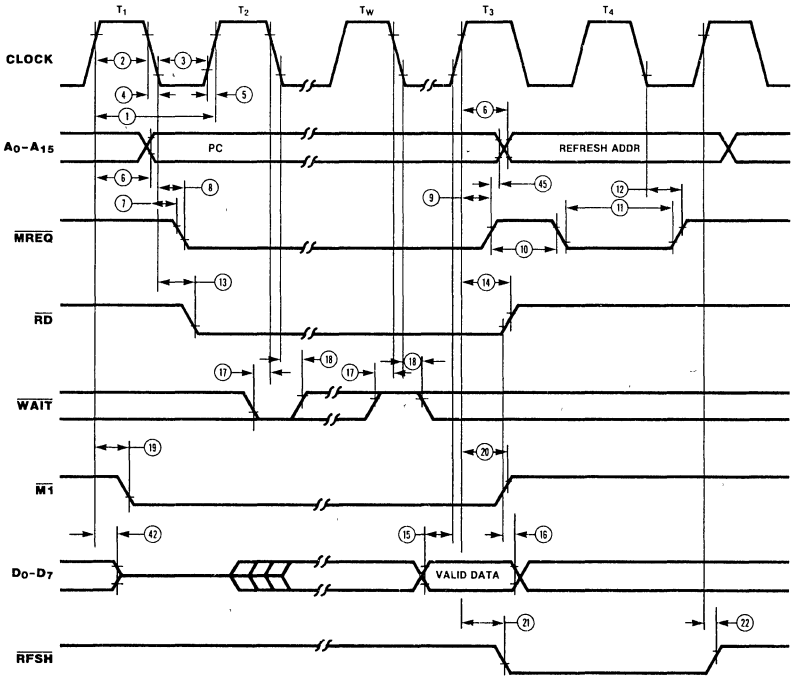
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. The falling edge of $\overline{MREQ}$ can be used directly as a Chip Enable to dynamic memories. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the rising edge of clock state T3. During clock states T3 and T4 of an M1 cycle dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



NOTE: T_W-Wait cycle added when necessary for slow ancillary devices

Figure 5. Instruction Opcode Fetch

CPU Timing (Continued)

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable, so that it can be used directly as a Chip Enable for dynamic memories. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

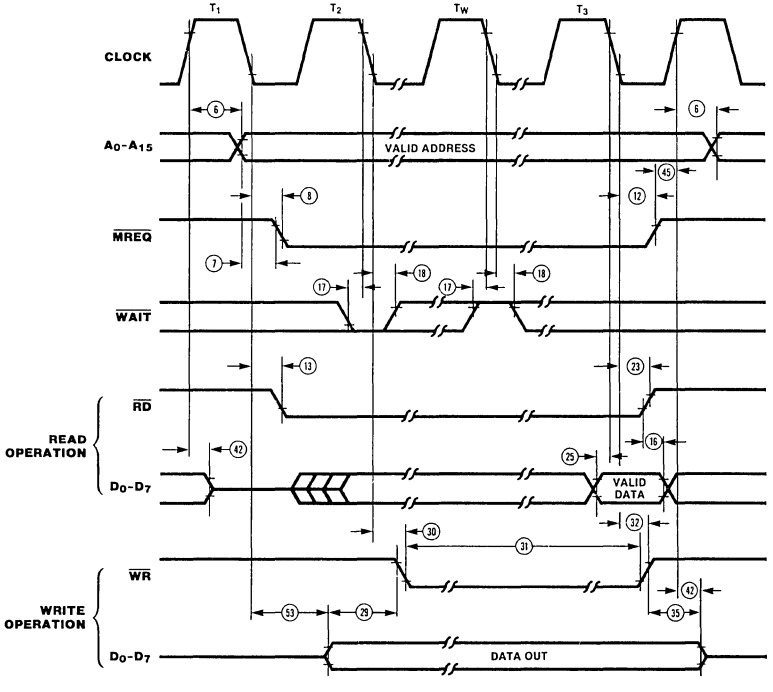
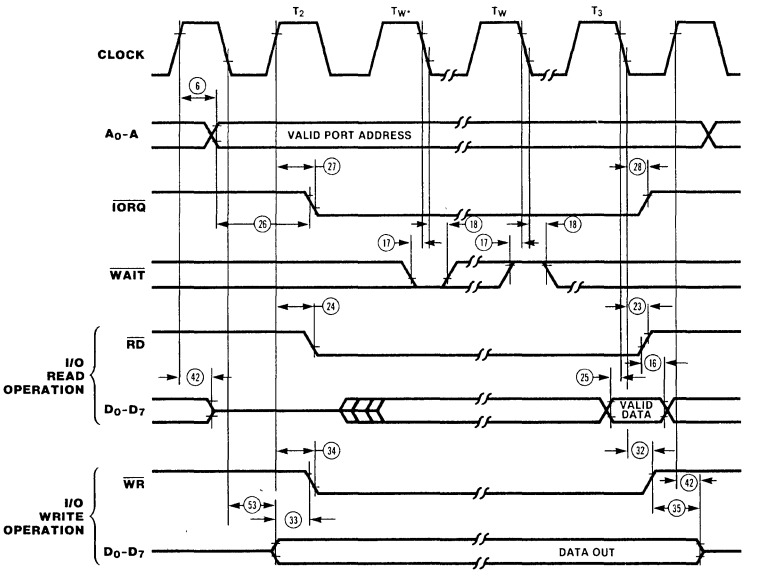


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

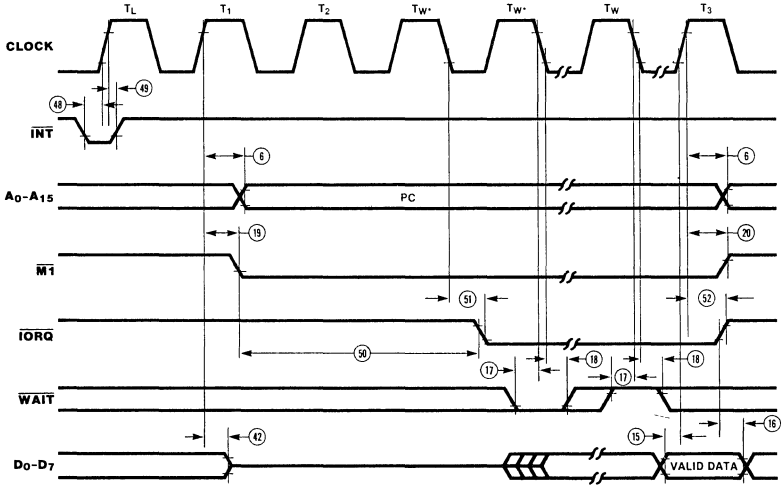
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_W). This extra Wait state allows sufficient time for an I/O port to decode the address and the port address lines.



NOTE. T_W* = One Wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.

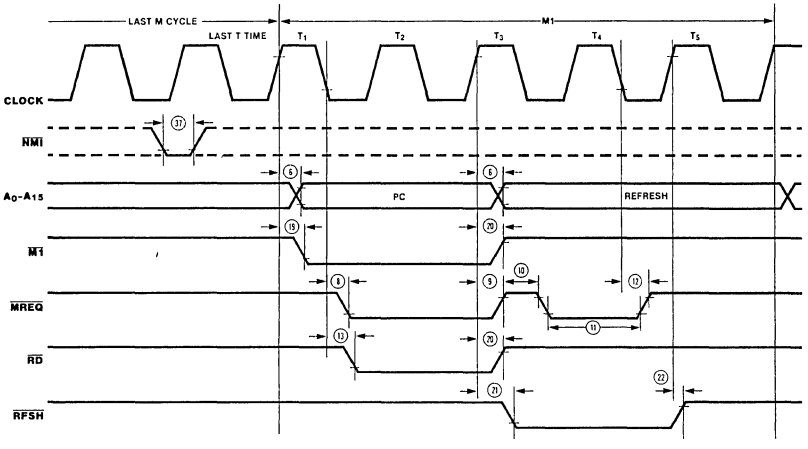


NOTE· 1) T_L = Last state of previous instruction. 2) Two Wait cycles automatically inserted by CPU(*).

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

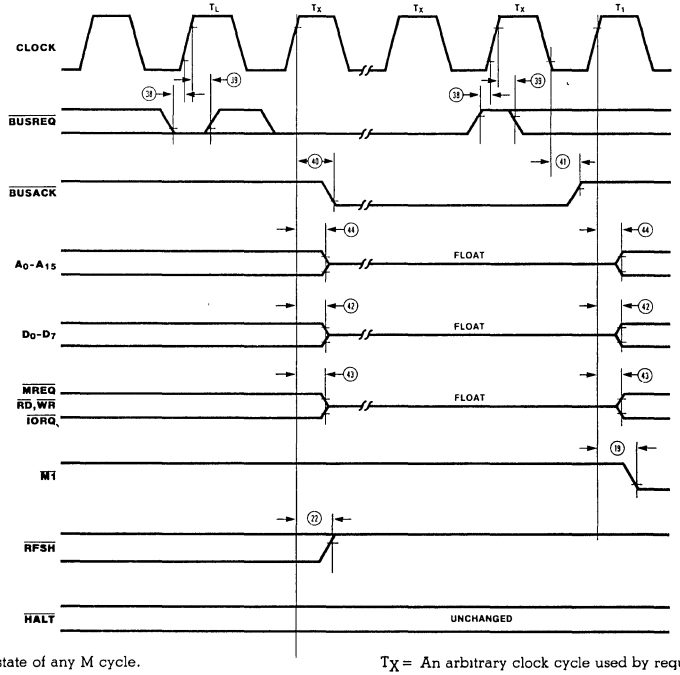
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Figure 9. Non-Maskable Interrupt Request Operation

Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.

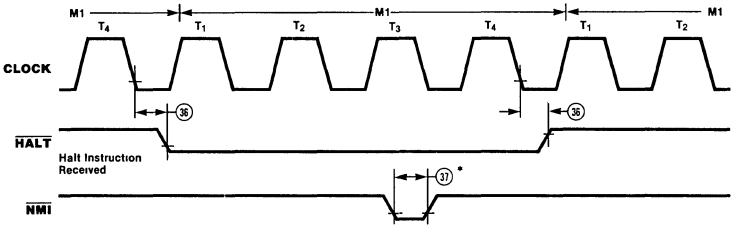


NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Bus Request/Acknowledge Cycle

CPU Timing (Continued)

Halt Acknowledge Cycle. When the CPU receives a $\overline{HALT}$ instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. When in the Halt state, the $\overline{HALT}$ output is active and remains so until an interrupt is processed (Figure 11).



NOTE: INT will also force a Halt exit.

*See note, Figure 9.

Figure 11. Halt Acknowledge Cycle

Reset Cycle. $\overline{RESET}$ must be active for at least three clock cycles for the CPU to properly accept it. As long as $\overline{RESET}$ remains active, the address and data buses float, and the control outputs are inactive. Once $\overline{RESET}$ goes inactive, two internal T cycles are consumed before the CPU resumes normal processing operation. $\overline{RESET}$ clears the PC register, so the first opcode fetch will be to location 0000 (Figure 12).

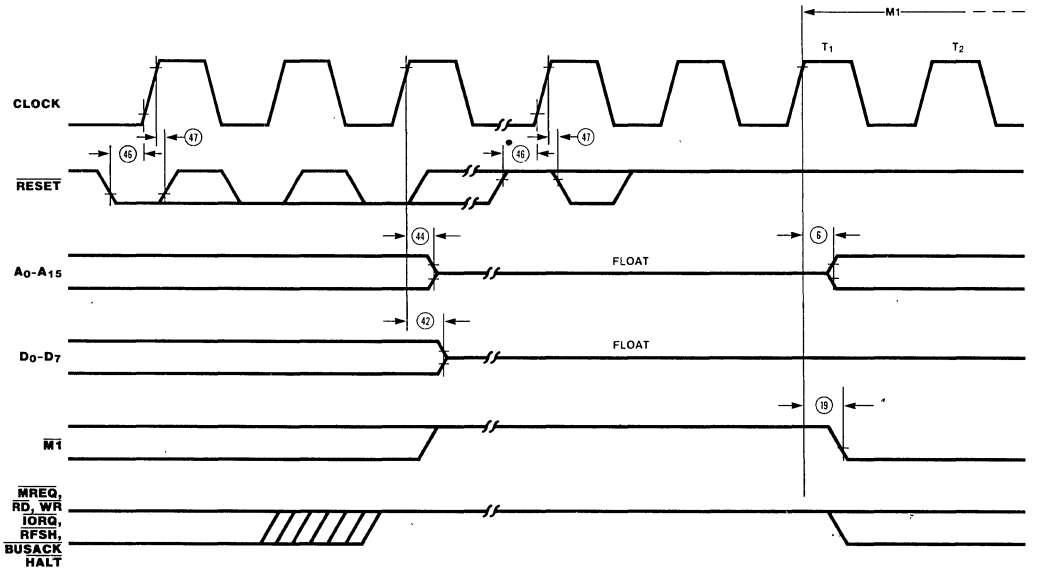


Figure 12. Reset Cycle

AC Characteristics

Number	Symbol	Parameter	Z80 CPU Min (ns)	Z80 CPU Max (ns)	Z80A CPU Min (ns)	Z80A CPU Max (ns)	Z80B CPU Min (ns)	Z80B CPU Max (ns)
1	TcC	Clock Cycle Time	400*		250*		165*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000
4	TfC	Clock Fall Time	—	30	—	30	—	20
5	TrC	Clock Rise Time		30		30		20
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	145	—	110	—	90
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*	—	65*	—	35*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	100	—	85	—	70
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*	—	220*	—	135*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	130	—	95	—	80
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	100	—	85	—	70
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70	—	70	—	60	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	130	—	100	—	80
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay		130		100		80
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	180	—	130	—	110
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	150	—	120	—	100
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	110	—	85	—	70
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	100	—	85	—	70
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*	—	180*	—	110*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	90	—	75	—	65
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	110	—	85	—	70
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*	—	80*	—	25*	—
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80		70
31	TwWR	$\overline{WR}$ Pulse Width	360*	—	220*	—	135*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	100	—	80	—	70
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	—	-10*	—	-55*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	80	—	65	—	60
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	300	—	300	—	260
37	TwNMI	$\overline{NMI}$ Pulse Width	80	—	80	—	70	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	—	50	—	50	—

*For clock periods other than the minimums shown in the table, calculate parameters using the expressions in the table on the following page.

AC Characteristics (Continued)

Number	Symbol	Parameter	Z80 CPU Min (ns)	Z80 CPU Max (ns)	Z80A CPU Min (ns)	Z80A CPU Max (ns)	Z80B CPU Min (ns)	Z80B CPU Max (ns)
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	110	—	100	—	90
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	90	—	90	—	80
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	110	—	80	—	70
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	110	—	90	—	80
45	TdCTr(A)	Address Stable after $\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑	160*		80*		35*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	—	60	—	60	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0	—	0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	—	80	—	70	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0	—	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	110	—	85	—	70
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	—	100	—	85	—	70
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	230	—	150	—	130

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.

Footnotes to AC Characteristics

Number	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 75	TwCh + TfC − 65	TwCh + TfC − 50
10	TwMREQh	TwCh + TfC − 30	TwCh + TfC − 20	TwCh + TfC − 20
11	TwMREQl	TcC − 40	TcC − 30	TcC − 30
26	TdA(IORQf)	TcC − 80	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210	TcC − 170	TcC − 140
31	TwWR	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRr(D)	TwCl + TrC − 80	TwCl + TrC − 70	TwCl + TrC − 55
45	TdCTr(A)	TwCl + TrC − 40	TwCl + TrC − 50	TwCl + TrC − 50
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 80	2TcC + TwCh + TfC − 65	2TcC + TwCh + TfC − 50

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
$V_{IHC} = V_{CC}$ -0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Absolute Maximum Ratings

Storage Temperature . . . . . -65°C to +150°C
Temperature
under Bias . . . . . Specified operating range
Voltages on all inputs and
outputs with respect to ground . -0.3 V to +7 V
Power Dissipation . . . . . 1.5 W

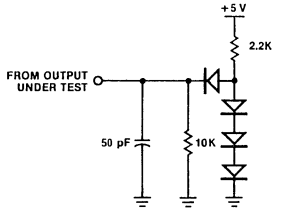
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- -40°C to +85°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- -55°C to +125°C, $+4.5\ V \leq V_{CC} \leq +5.5\ V$

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



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA
I_{CC}	Power Supply Current				
	Z80		150[1]	mA	
	Z80A		200[2]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LEAK}	3-State Output Leakage Current in Float	-10	10[3]	μA	V_{OUT} = 0.4 to V_{CC}

1 For military grade parts, I_{CC} is 200 mA
2 Typical rate for Z80A is 90 mA.
3 A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Capacitance

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8400	CE	2.5 MHz	Z80 CPU (40-pin)
Z8400	CM	2.5 MHz	Same as above
Z8400	CMB	2.5 MHz	Same as above
Z8400	CS	2.5 MHz	Same as above
Z8400	DE	2.5 MHz	Same as above
Z8400	DS	2.5 MHz	Same as above
Z8400	PE	2.5 MHz	Same as above
Z8400	PS	2.5 MHz	Same as above
Z8400A	CE	4.0 MHz	Z80A CPU (40-pin)
Z8400A	CM	4.0 MHz	Same as above
Z8400A	CMB	4.0 MHz	Same as above
Z8400A	CS	4.0 MHz	Same as above
Z8400A	DE	4.0 MHz	Z80A CPU (40-pin)
Z8400A	DS	4.0 MHz	Same as above
Z8400A	PE	4.0 MHz	Same as above
Z8400A	PS	4.0 MHz	Same as above
Z8400B	CE	6.0 MHz	Z80B CPU (40-pin)
Z8400B	CM	6.0 MHz	Same as above
Z8400B	CMB	6.0 MHz	Same as above
Z8400B	CS	6.0 MHz	Same as above
Z8400B	DE	6.0 MHz	Same as above
Z8400B	DS	6.0 MHz	Same as above
Z8400B	PE	6.0 MHz	Same as above
Z8400B	PS	6.0 MHz	Same as above

NOTES. C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C

Z8410
Z80® DMA Direct Memory Access Controller

Zilog

Product Specification

March 1981

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 Z-80 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.

General Description

The Z-80 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, 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.

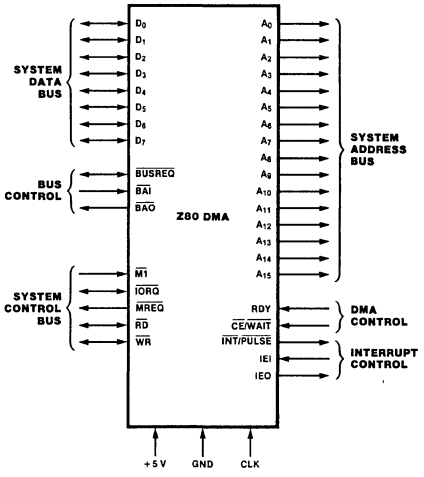


Figure 1. Pin Functions

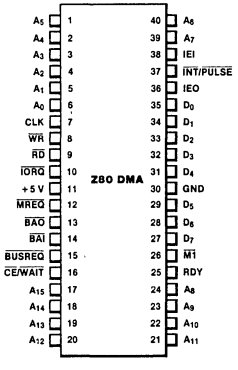


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 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 Z-80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 Family single-phase clock.

Functional Description

Classes of Operation. The Z-80 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 1.25M bytes per second can be obtained with the 2.5 MHz Z-80 DMA or 2M bytes per second with the 4 MHz Z-80A 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 Z-80 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.
- *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.

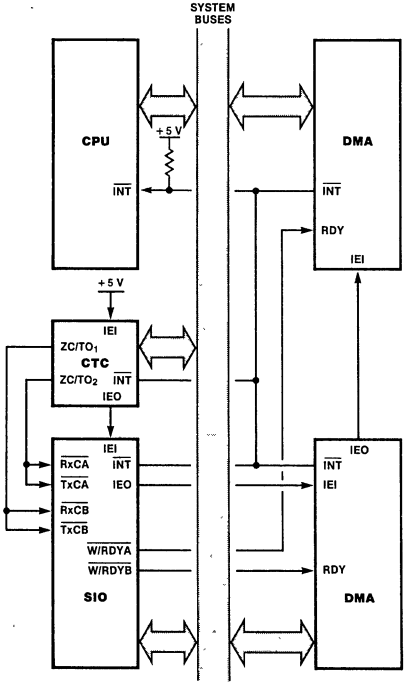


Figure 3. Typical Z-80 Environment

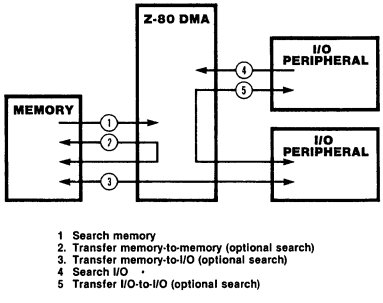


Figure 4. Basic Functions of the Z-80 DMA

Functional Description (Continued)

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 Z-80 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, 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 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 Z-80 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 Z-80 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 Z-80 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

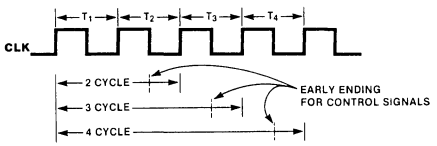


Figure 5. Variable Cycle Length

Functional Description (Continued)

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 Z-80 Family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z-80 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 Description

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 $\overline{BAO}$ 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 $\overline{BAO}$ form a 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.

CLK. *System Clock* (input). Standard Z-80 single-phase clock at 2.5 MHz (Z-80 DMA) or 4.0 MHz (Z-80A DMA). For slower system clocks, a TTL gate with 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 10K 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 blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

Pin Description (Continued)

$\overline{\text{INT}}$/PULSE. *Interrupt Request* (output, active Low, open drain). This 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. It can be used this way only 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).

$\overline{\text{IORQ}}$. *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 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 $\overline{\text{IORQ}}$ 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 $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active.

$\overline{\text{MREQ}}$. *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.

$\overline{\text{RD}}$. *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 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 Z-80 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 Z-80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{\text{CE}}$/$\overline{\text{WAIT}}$ 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 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.

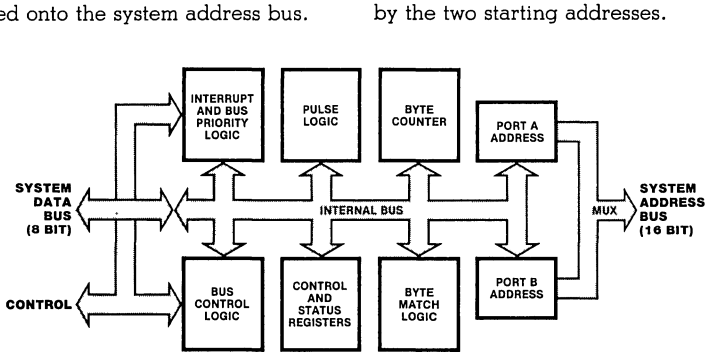


Figure 6. Block Diagram

Internal Structure (Continued)

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.

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected (Zilog Application Note 03-0041-01, *The Z-80 Family Program Interrupt Structure*). 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.

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)

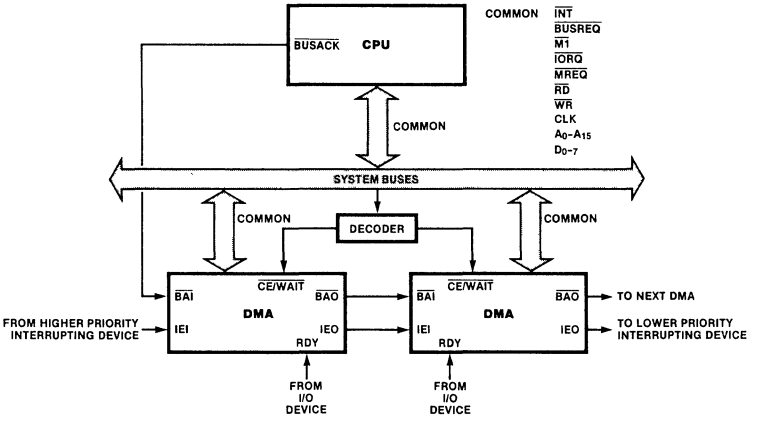


Figure 7. Multiple-DMA Interconnection to the Z-80 CPU

Programming

The Z-80 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 Z-80 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. 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 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 Z-80 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 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 Z-80 CPU's OTIR instruction.

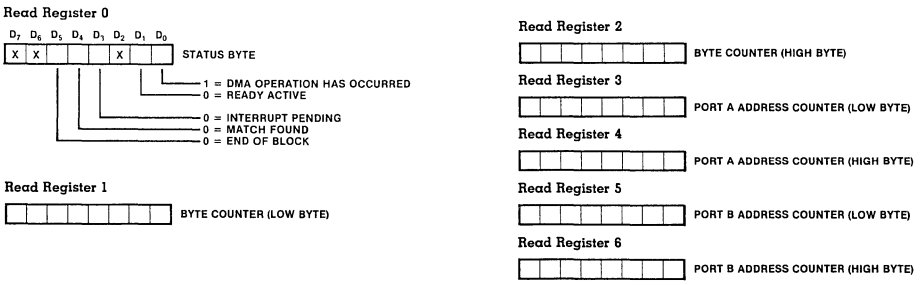


Figure 8a. Read Registers

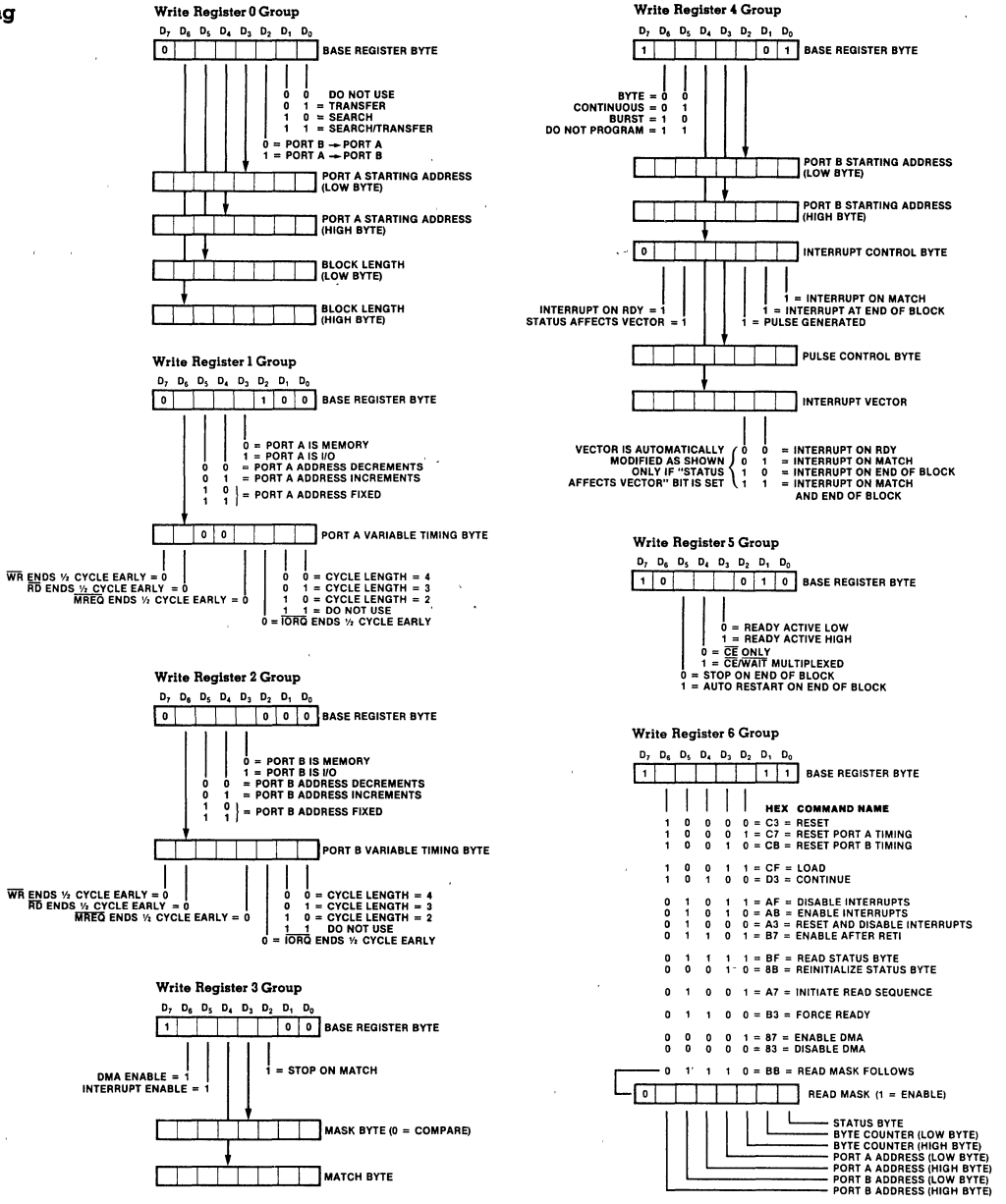


Figure 8b. Write Registers

Comments	D_7	D_6	D_5	D_4	D_3	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	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B ⟶ A Temporary for Loading B Address*	0	1 Transfer, No Search	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	0 Address Changes	1 Address Increments	0 Port is Memory	1	0	0	14
WR2 defines Port B as peripheral with fixed address	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O	0	0	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address	1	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Address Follows	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High	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	0 No Address or Block Length Bytes	0	0	1 A ⟶ B	0	1 Transfer, No Search	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

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.

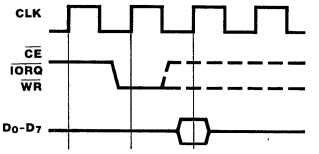


Figure 10. CPU-to-DMA Write Cycle

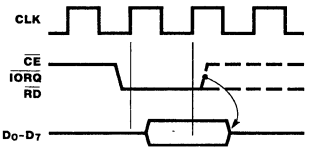


Figure 11. CPU-to-DMA Read Cycle

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 Z-80 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 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 between T_2 and T_3. If the $\overline{CE/WAIT}$ 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_W for I/O transactions. If $\overline{CE/WAIT}$ is Low during this time another T-cycle is added, during which the $\overline{CE/WAIT}$ line will again be sampled. The duration of transactions can thus be indefinitely extended.

Variable Cycle and Edge Timing. The Z-80 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,

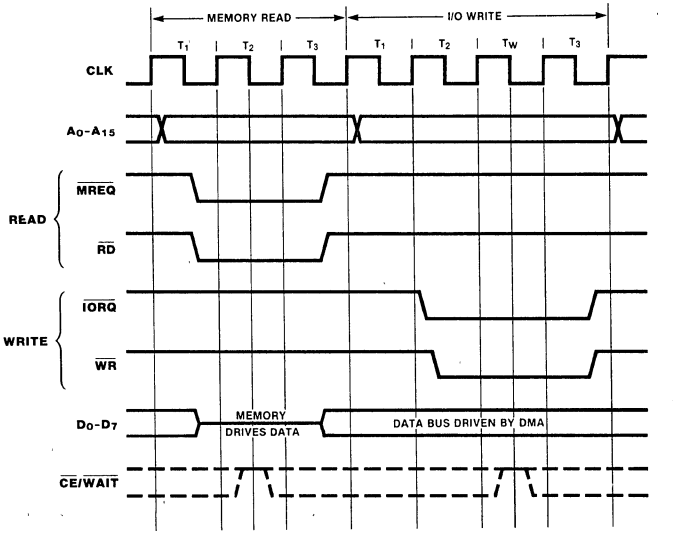


Figure 12. Memory-to-I/O Transfer

Active State Timing (DMA as Bus Controller) (Continued)

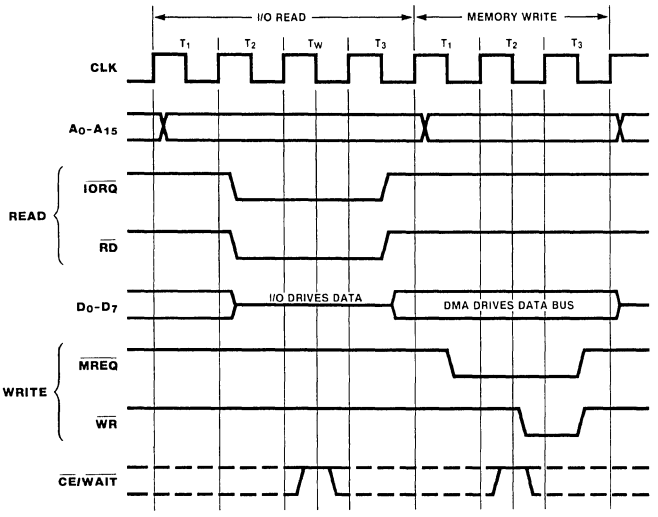


Figure 13. I/O-to-Memory Transfer

the trailing edges of the $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$ and $\overline{WR}$ signals can be independently terminated one-half cycle early. Figure 14 illustrates this.

In the variable-cycle mode, unlike default timing, $\overline{IORQ}$ comes active one-half cycle before $\overline{MREQ}$, $\overline{RD}$ and $\overline{WR}$. $\overline{CE}/\overline{WAIT}$ 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 $\overline{CE}/\overline{WAIT}$ 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 $\overline{RD}$ 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 $\overline{BUSREQ}$ low. After receiving $\overline{BUSREQ}$ the CPU acknowledges on the $\overline{BAI}$ input either directly or through a multiple-DMA daisy chain. When a Low is detected on $\overline{BAI}$ for two consecutive rising edges of CLK, the DMA will begin transferring data on the next rising edge of CLK.

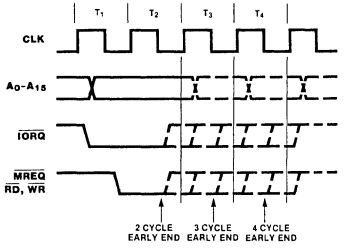


Figure 14. Variable-Cycle and Edge Timing

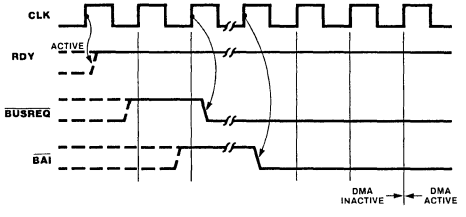


Figure 15. Bus Request and Acceptance

Active State Timing (DMA as Bus Controller) (Continued)

Bus Release Byte-at-a-Time. In Byte-at-a-Time mode, $\overline{BUSREQ}$ 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 Z-80 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 $\overline{BUSREQ}$ and $\overline{BAI}$ have returned High.

Bus Release at End of Block. In Burst and Continuous modes, an end of block causes $\overline{BUSREQ}$ 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 $\overline{BUSREQ}$ to go High on the next rising edge of CLK after the completion of its current byte operation (Figure 18). The action on $\overline{BUSREQ}$ 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 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 $\overline{BUSREQ}$ 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 Z-80 peripherals. Refer to Zilog Application Note 03-0041-01 *(The Z-80 Family Program 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 resets the Interrupt Under Service latch in the Z-80 DMA.

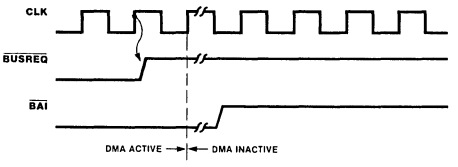


Figure 16. Bus Release (Byte-at-a-Time Mode)

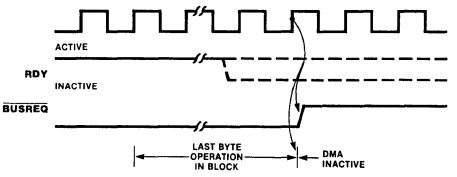


Figure 17. Bus Release at End of Block (Burst and Continuous Modes)

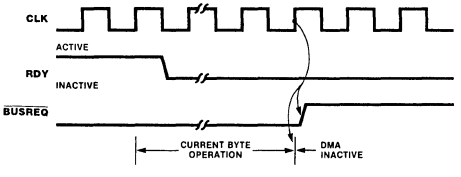


Figure 18. Bus Release When Not Ready (Burst Mode)

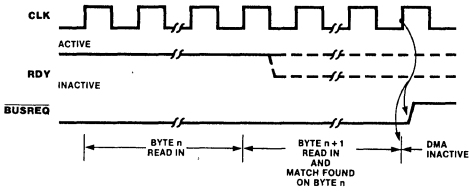


Figure 19. Bus Release on Match (Burst and Continuous Modes)

Absolute Maximum Ratings

Operating Ambient Temperature Under Bias . . . As Specified Under "Ordering Information"

Storage Temperature . . . . . -65°C to +150°C

Voltage On Any Pin with Respect to Ground . . . . . -0.3 V to +7 V

Power Dissipation . . . . . 1.5 W

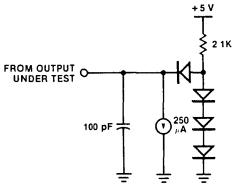
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Standard Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section. 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.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test 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 $\overline{BUSREQ}$ 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 Z-80 DMA Z-80A DMA		 150 200	 mA mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	3-State Output Leakage Current in Float		10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	3-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	µA	$0 \le V_{IN} \le V_{CC}$

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

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		35	pF	Unmeasured Pins
C_{IN}	Input Capacitance		5	pF	Returned to Ground
C_{OUT}	Output Capacitance		10	pF	

f = 1 MHz, over specified temperature range.

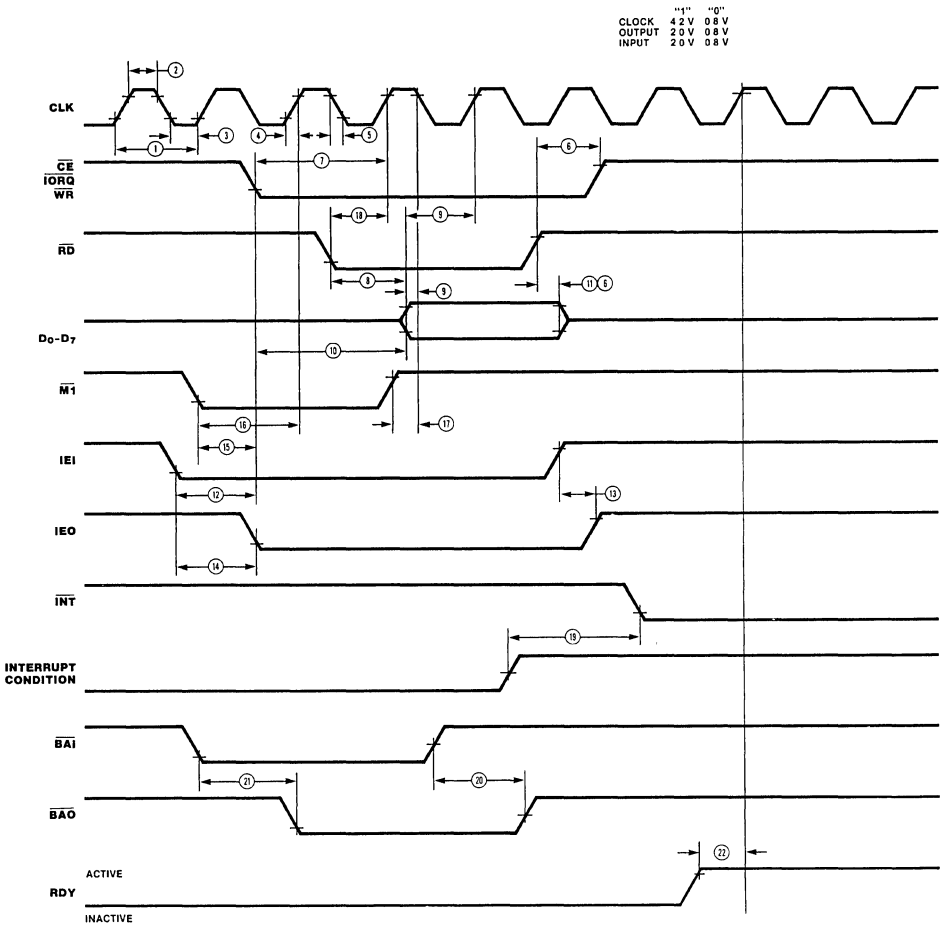
Inactive State AC Characteristics

Number	Symbol	Parameter	Z-80 DMA Min	Z-80 DMA Max	Z-80A DMA Min	Z-80A DMA Max	Unit
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	110	2000	ns
3	TwCl	Clock Width (Low)	170	2000	110	2000	ns
4	TrC	Clock Rise Time		30		30	ns
5	TfC	Clock Fall Time		30		30	ns
6	Th	Hold Time for Any Specified Setup Time	0		0		ns
7	TsC(Cr)	$\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$ ↓ to Clock ↑ Setup	280		145		ns
8	TdDO(RDf)	$\overline{RD}$↓ to Data Output Delay		500		380	ns
9	TsWM(Cr)	Data In to Clock ↑ Setup ($\overline{WR}$ or $\overline{M1}$)	50		50		ns
10	TdCf(DO)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
11	TdRD(Dz)	$\overline{RD}$ ↑ to Data Float Delay (output buffer disable)		160		110	ns
12	TsIEI(IORQ)	IEI↓ to $\overline{IORQ}$ ↓ Setup (INTA Cycle)	140		140		ns
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		210		160	ns
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		190		130	ns
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{M1}$↓)		300		190	ns
16	TsM1f(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup	210		90		ns
17	TsM1r(Cf)	$\overline{M1}$ ↑ to Clock ↓ Setup	20		-10		
18	TsRD(Cr)	$\overline{RD}$ ↓ to Clock ↑ Setup ($\overline{M1}$ Cycle)	240		115		ns
19	TdI(INT)	Interrupt Cause to $\overline{INT}$ ↓ Delay ($\overline{INT}$ generated only when DMA is inactive)		500		500	ns
20	TdBAIr(BAOr)	$\overline{BAI}$ ↑ to $\overline{BAO}$ ↑ Delay		200		150	ns
21	TdBAIf(BAOf)	$\overline{BAI}$ ↓ to $\overline{BAO}$ ↓ Delay		200		150	ns
22	TsRDY(Cr)	RDY Active to Clock ↑ Setup	150		100		ns

NOTE

1 Negative minimum setup values mean that the first-mentioned event can come after the second-mentioned event

Inactive State AC Characteristics (Continued)



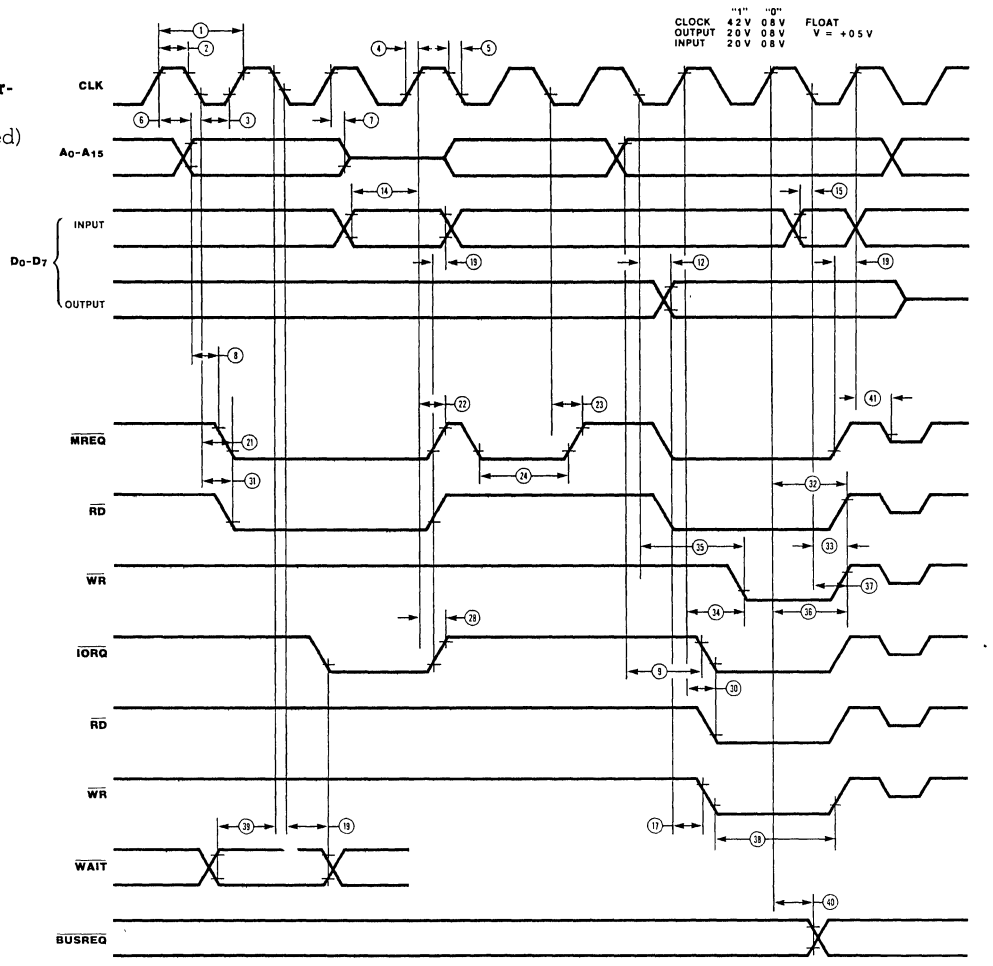
NOTE
Signals in this diagram bear no relation to one another unless specifically noted as a numbered item

Active State AC Characteristics

Number	Symbol	Parameter	Z-80 DMA Min(ns)	Z-80 DMA Max(ns)	Z-80A DMA Min(ns)	Z-80A DMA Max(ns)
1	TcC	Clock Cycle Time	400		250	
2	TwCh	Clock Width (High)	180	2000	110	2000
3	TwCl	Clock Width (Low)	180	2000	110	2000
4	TrC	Clock Rise Time		30		30
5	TfC	Clock Fall Time		30		30
6	TdA	Address Output Delay		145		110
7	TdC(Az)	Clock ↑ to Address Float Delay		110		90
8	TsA(MREQ)	Address to $\overline{MREQ}$ ↓ Setup (Memory Cycle)	(2)+(5)-75		(2)+(5)-75	
9	TsA(IRW)	Address Stable to $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ ↓ Setup (I/O Cycle)	(1)-80		(1)-70	
*10	TdRW(A)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Stable Delay	(3)+(4)-40		(3)+(4)-50	
*11	TdRW(Az)	$\overline{RD}$, $\overline{WR}$ ↑ 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 $\overline{WR}$ ↓ Setup (Memory Cycle)	(1)-210		(1)-170	
17	TsDO(WfI)	Data Out to $\overline{WR}$ ↓ Setup (I/O cycle)	100		100	
*18	TdWr(DO)	$\overline{WR}$ ↑ to Data Out Delay	(3)+(4)-80		(3)+(4)-70	
19	Th	Hold Time for Any Specified Setup Time	0		0	
20	TdCf(Mf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		100		85
21	TdCr(Mr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay		100		85
22	TdCf(Mr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		100		85
23	TwMl	$\overline{MREQ}$ Low Pulse Width	(1)-40		(1)-30	
*24	TwMh	$\overline{MREQ}$ High Pulse Width	(2)+(5)-30		(2)+(5)-20	
25	TdCr(If)	Clock ↑ to $\overline{IORQ}$ ↓ Delay		90		75
26	TdCr(Ir)	Clock ↑ to $\overline{IORQ}$ ↑ Delay		100		85
*27	TdCf(Ir)	Clock ↓ to $\overline{IORQ}$ ↑ Delay		110		85
28	TdCr(Rf)	Clock ↑ to $\overline{RD}$ ↓ Delay		100		85
29	TdCf(Rf)	Clock ↓ to $\overline{RD}$ ↓ Delay		130		95
30	TdCr(Rr)	Clock ↑ to $\overline{RD}$ ↑ Delay		100		85
31	TdCf(Rr)	Clock ↓ to $\overline{RD}$ ↑ Delay		110		85
32	TdCr(Wf)	Clock ↑ to $\overline{WR}$ ↓ Delay		80		65
33	TdCf(Wf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80
34	TdCr(Wr)	Clock ↑ to $\overline{WR}$ ↑ Delay		100		80
35	TdCf(Wr)	Clock ↓ to $\overline{WR}$ ↑ Delay		100		80
36	TwWl	$\overline{WR}$ Low Pulse Width	(1)-40		(1)-30	
37	TsWA(Cf)	$\overline{WAIT}$ to Clock ↓ Setup	70		70	
38	TdCr(B)	Clock ↑ to $\overline{BUSREQ}$ Delay		150		100
39	TdCr(Iz)	Clock ↑ to $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$ Float Delay		100		80

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 (standard) 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)



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

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8410	CE	2.5 MHz	Z80 DMA (40-pin)	Z8410A	CE	4.0 MHz	Z80A DMA (40-pin)
Z8410	CM	2.5 MHz	Same as above	Z8410A	CM	4.0 MHz	Same as above
Z8410	CMB	2.5 MHz	Same as above	Z8410A	CMB	4.0 MHz	Same as above
Z8410	CS	2.5 MHz	Same as above	Z8410A	CS	4.0 MHz	Same as above
Z8410	DE	2.5 MHz	Same as above	Z8410A	DE	4.0 MHz	Same as above
Z8410	DS	2.5 MHz	Same as above	Z8410A	DS	4.0 MHz	Same as above
Z8410	PE	2.5 MHz	Same as above	Z8410A	PE	4.0 MHz	Same as above
Z8410	PS	2.5 MHz	Same as above	Z8410A	PS	4.0 MHz	Same as above

NOTES. C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8420
Z80® PIO Parallel Input/Output Controller

Zilog

Product Specification

March 1981

Features

- Provides a direct interface between Z-80 microcomputer systems and peripheral devices.
- Both ports have interrupt-driven handshake for fast response.
- Four programmable operating modes: byte input, byte output, byte input/output (Port A only), and bit input/output.
- Programmable interrupts on peripheral status conditions.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5 V).

General Description

The Z-80 PIO Parallel I/O Circuit is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z-80 CPU. The CPU configures the Z-80 PIO to interface with a wide range of peripheral devices with no other external logic. Typical peripheral devices that are compatible with the Z-80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc.

One characteristic of the Z-80 peripheral controllers that separates them from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under interrupt control. Thus, the interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z-80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO.

Another feature of the PIO is the ability to interrupt the CPU upon occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

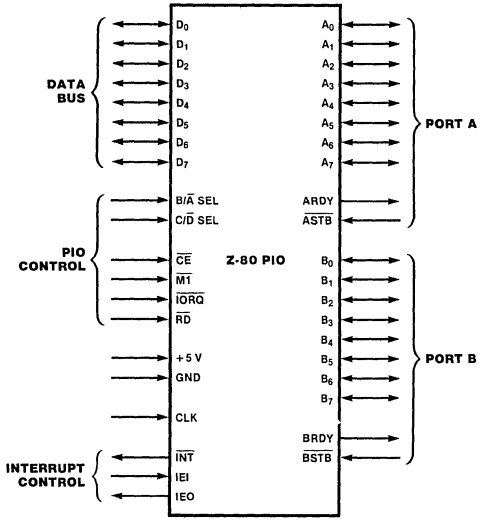


Figure 1. Pin Functions

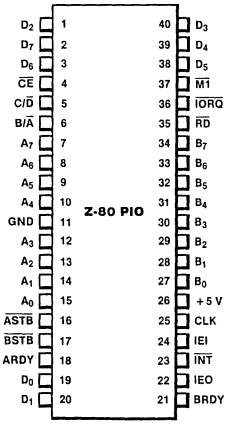


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates when a data transfer has occurred.

Operating Modes. The Z-80 PIO ports can be programmed to operate in four modes: byte output (Mode 0), byte input (Mode 1), byte input/output (Mode 2) and bit input/output (Mode 3).

In Mode 0, either Port A or Port B can be programmed to output data. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

In Mode 1, either Port A or Port B can be configured in the input mode. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is detected by the external device. The external device then places data on the I/O lines and strobes the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3 and masked off. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except that data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. For input, operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt (if enabled).

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR) or all inputs active (AND). For example, if the port is programmed for active Low inputs and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

Data outputs are controlled by the CPU and can be written or changed at any time.

- Individual bits can be masked off.
- The handshake signals are not used in Mode 3; Ready is held Low, and Strobe is disabled.
- When using the Z-80 PIO interrupts, the Z-80 CPU interrupt mode must be set to Mode 2.

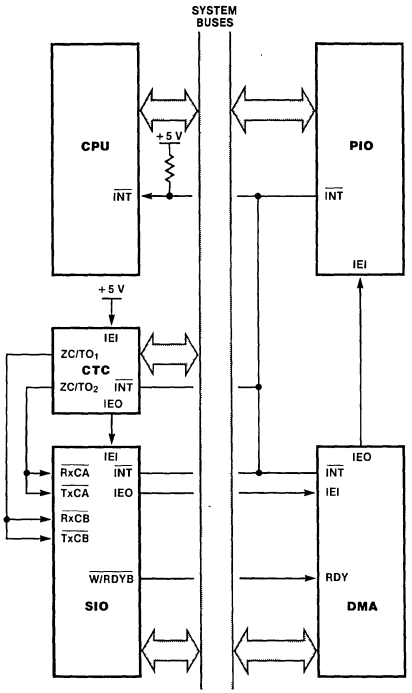


Figure 3. PIO in a Typical Z80 Family Environment

Internal Structure

The internal structure of the Z-80 PIO consists of a Z-80 CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic (Figure 4). The CPU bus interface logic allows the Z-80 PIO to interface directly to the Z-80 CPU with no other external logic. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

Port Logic. Each port contains separate input and output registers, handshake control logic, and the control registers shown in Figure 5. All data transfers between the peripheral unit and the CPU use the data input and output registers. The handshake logic associated with each port controls the data transfers through the input and the output registers. The mode control register (two bits) selects one of the four programmable operating modes.

The control mode (Mode 3) uses the remaining registers. The input/output control register specifies which of the eight data bits in the port are to be outputs and enables these bits; the remaining bits are inputs. The mask register and the mask control register control Mode 3 interrupt conditions. The mask register specifies which of the bits in the port are active and which are masked or inactive.

The mask control register specifies two conditions: first, whether the active state of the input bits is High or Low, and second, whether an interrupt is generated when any *one* unmasked input bit is active (OR condition) or if the interrupt is generated when *all* unmasked input bits are active (AND condition).

Interrupt Control Logic. The interrupt control logic section handles all CPU interrupt protocol for nested-priority interrupt structures. Any device's physical location in a daisy-chain configuration determines its priority. Two lines (IEI and IEO) are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output, or bidirectional modes, an interrupt can be generated whenever the peripheral requests a new byte transfer. In the bit control mode, an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routines completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

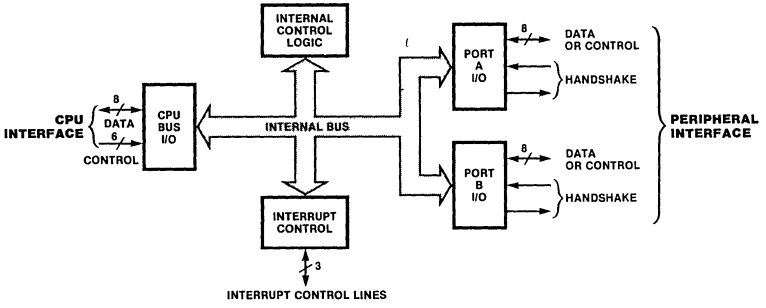


Figure 4. Block Diagram

Internal Structure (Continued)

If the CPU (in interrupt Mode 2) accepts an interrupt, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector forms a pointer to a location in memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant eight bits of the indirect pointer while the I Register in the CPU provides the most significant eight bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to 0 within the PIO because the pointer must point to two adjacent memory locations for a complete 16-bit address.

Unlike the other Z-80 peripherals, the PIO does not enable interrupts immediately after programming. It waits until $\overline{M1}$ goes Low (e.g., during an opcode fetch). This condition is unimportant in the Z-80 environment but might not be if another type of CPU is used.

The PIO decodes the RETI (Return From Interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine. No other communication with the CPU is required.

CPU Bus I/O Logic. The CPU bus interface logic interfaces the Z-80 PIO directly to the Z-80 CPU, so no external logic is necessary. For large systems, however, address decoders and/or buffers may be necessary.

Internal Control Logic. This logic receives the control words for each port during programming and, in turn, controls the operating functions of the Z-80 PIO. The control logic synchronizes the port operations, controls the port mode, port addressing, selects the read/write function, and issues appropriate commands to the ports and the interrupt logic. The Z-80 PIO does not receive a write input from the CPU; instead, the $\overline{RD}$, $\overline{CE}$, $C/\overline{D}$ and $\overline{IORQ}$ signals generate the write input internally.

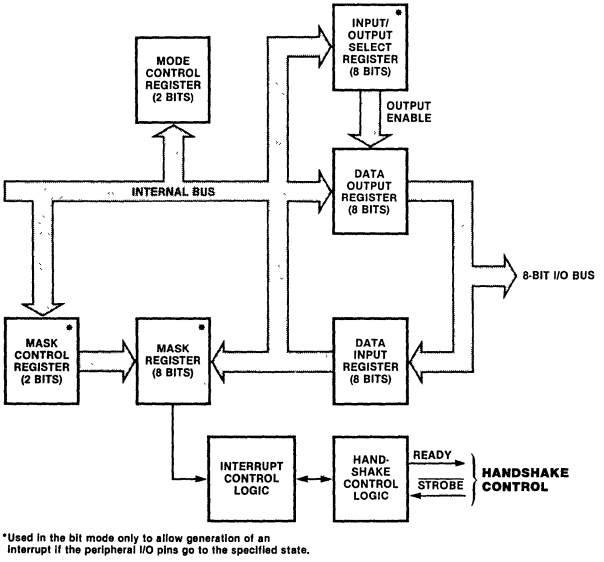


Figure 5. Typical Port I/O Block Diagram

Programming

Mode 0, 1, or 2. *(Byte Input, Output, or Bidirectional).* Programming a port for Mode 0, 1, or 2 requires two words per port. These words are:

A Mode Control Word. Selects the port operating mode (Figure 6). This word may be written any time.

An Interrupt Vector. The Z-80 PIO is designed for use with the Z-80 CPU in interrupt Mode 2 (Figure 7). When interrupts are enabled, the PIO must provide an interrupt vector.

Mode 3. *(Bit Input/Output).* Programming a port for Mode 3 operation requires a control word, a vector (if interrupts are enabled), and three additional words, described as follows:

I/O Register Control. When Mode 3 is selected, the mode control word must be followed by another control word that sets the I/O control register, which in turn defines which port lines are inputs and which are outputs (Figure 8).

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_6 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable. There is one other control word which can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

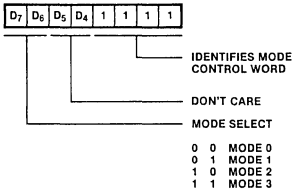


Figure 6. Mode Control Word

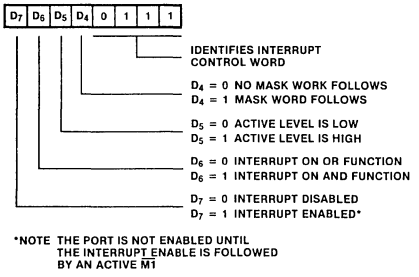


Figure 9. Interrupt Control Word

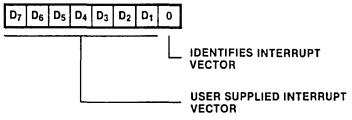


Figure 7. Interrupt Vector Word

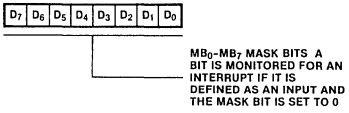


Figure 10. Mask Control Word

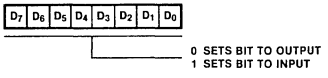


Figure 8. I/O Register Control Word

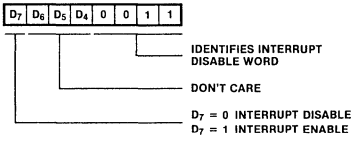


Figure 11. Interrupt Disable Word

Pin Description

A_0-A_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{ASTB}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{ASTB}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

B_0-B_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5 V to drive Darlington transistors. B_0 is the least significant bit of the bus.

B/$\overline{A}$. *Port B Or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{BSTB}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{ASTB}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{D}$. *Control Or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z-80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{A}$ Select line. A Low on this pin means that the Z-80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z-80 PIO uses the standard single-phase Z-80 system clock.

D_0-D_7. *Z-80 CPU Data Bus* (bidirectional, 3-state). This bus is used to transfer all data and commands between the Z-80 CPU and the Z-80 PIO. D_0 is the least significant bit.

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

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When $\overline{INT}$ is active the Z-80 PIO is requesting an interrupt from the Z-80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from Z-80 CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the Z-80 CPU and the Z-80 PIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are active, the port addressed by B/$\overline{A}$ transfers data to the CPU (a read operation). Conversely, when $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is not, the port addressed by B/$\overline{A}$ is written into from the CPU with either data or control information, as specified by C/$\overline{D}$. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

Pin Description (Continued)

$\overline{M1}$. *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{M1}$ and $\overline{RD}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{M1}$ and $\overline{IORQ}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{M1}$ has two other functions within the Z-80 PIO: it synchronizes the PIO interrupt logic; when $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal, the PIO is reset.

$\overline{RD}$. *Read Cycle Status* (input from Z-80 CPU, active Low). If $\overline{RD}$ is active, or an I/O operation is in progress, $\overline{RD}$ is used with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the Z-80 PIO to the Z-80 CPU.

Timing

The following timing diagrams show typical timing in a Z-80 CPU environment. For more precise specifications refer to the composite ac timing diagram.

Write Cycle. Figure 12 illustrates the timing for programming the Z-80 PIO or for writing data to one of its ports. No Wait states are allowed for writing to the PIO other than the automatically inserted T_{WA}. The PIO does not receive a specific write signal; it internally generates its own from the lack of an active $\overline{RD}$ signal.

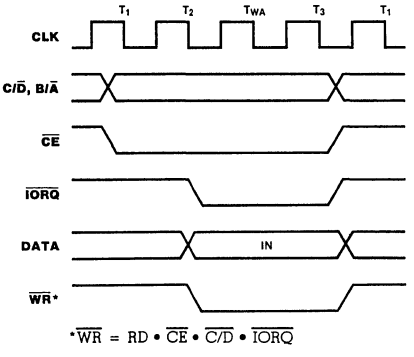


Figure 12. Write Cycle Timing

Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z-80 PIO ports. No Wait states are allowed for reading the PIO other than the automatically inserted T_{WA}.

Output Mode (Mode 0). An output cycle (Figure 14) is always started by the execution of an output instruction by the CPU. The $\overline{WR}$* pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The $\overline{WR}$* pulse sets the Ready flag after a Low-going edge of CLK, indicating data is available. Ready stays active until the positive edge of the strobe line is received, indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{INT}$ if the interrupt enable flip-flop has been set and if this device has the highest priority.

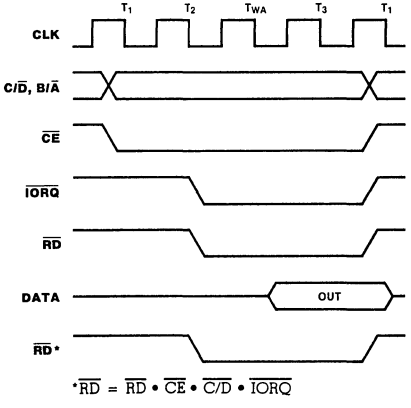


Figure 13. Read Cycle Timing

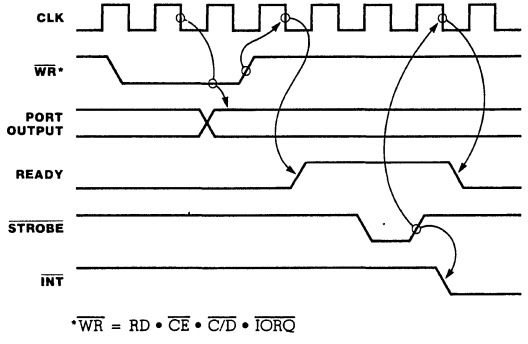


Figure 14. Mode 0 Output Timing

Timing
(Continued)

Input Mode (Mode 1). When $\overline{\text{STROBE}}$ goes Low, data is loaded into the selected port input register (Figure 15). The next rising edge of strobe activates $\overline{\text{INT}}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets Ready to an inactive state, indicating that the input register is full and cannot accept any more data until the CPU completes a read. When a read is complete, the positive edge of $\overline{\text{RD}}$ sets Ready at the next Low-going transition of CLK. At this time new data can be loaded into the PIO.

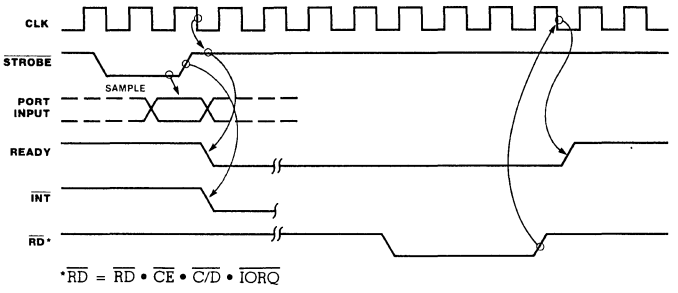


Figure 15. Mode 1 Input Timing

Bidirectional Mode (Mode 2). This is a combination of Modes 0 and 1 using all four handshake lines and the eight Port A I/O lines (Figure 16). Port B must be set to the bit mode and its inputs must be masked. The Port A handshake lines are used for output control and the Port B lines are used for input control. If interrupts occur, Port A's vector will be used during port output and Port B's will be used during port input. Data is allowed out onto the Port A bus only when $\overline{\text{ASTB}}$ is Low. The rising edge of this strobe can be used to latch the data into the peripheral.

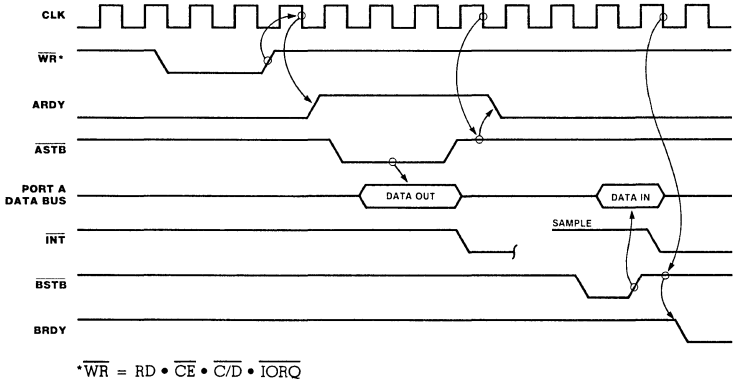


Figure 16. Mode 2 Bidirectional Timing

Timing (Continued)

Bit Mode (Mode 3). The bit mode does not utilize the handshake signals, and a normal port write or port read can be executed at any time. When writing, the data is latched into the output registers with the same timing as the output mode (Figure 17).

When reading the PIO, the data returned to the CPU is composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register contains data that was present immediately prior to the falling edge of $\overline{RD}$. An interrupt is generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers. However, if Port A is programmed in bidirectional mode, Port B does not issue an interrupt in bit mode and must therefore be polled.

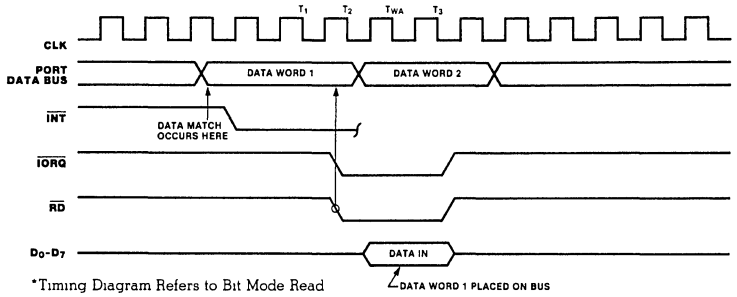


Figure 17. Mode 3 Bit Mode Timing

Interrupt Acknowledge Timing. During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the Interrupt Enable signal to ripple through the daisy chain. The peripheral with IEI High and IEO Low during $\overline{INTACK}$ places a preprogrammed 8-bit interrupt vector on the data bus at this time (Figure 18). IEO is held Low until a Return From Interrupt (RETI) instruction is executed by the CPU while IEI is High. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.

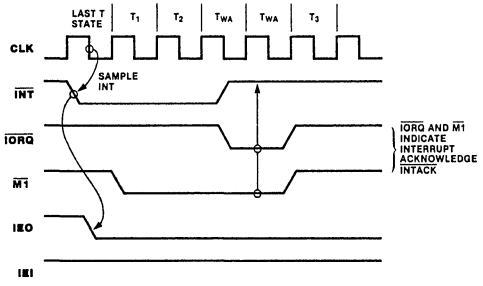


Figure 18. Interrupt Acknowledge Timing

Return From Interrupt Cycle. If a Z-80 peripheral has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always Low, inhibiting lower priority devices from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO is Low unless an "ED" is decoded as the first byte of a 2-byte opcode (Figure 19). In this case, IEO goes High until the next opcode byte is decoded, whereupon it goes Low again. If the second byte of the opcode was a "4D," then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service has its IEI High and its IEO Low. This device is the highest-priority device in the daisy chain that has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D," this peripheral device resets its "interrupt under service" condition.

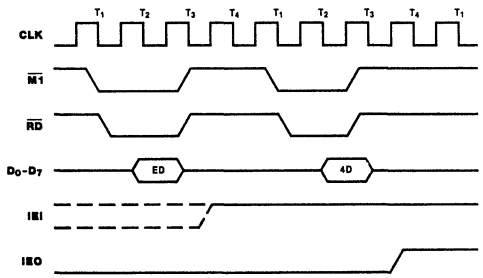
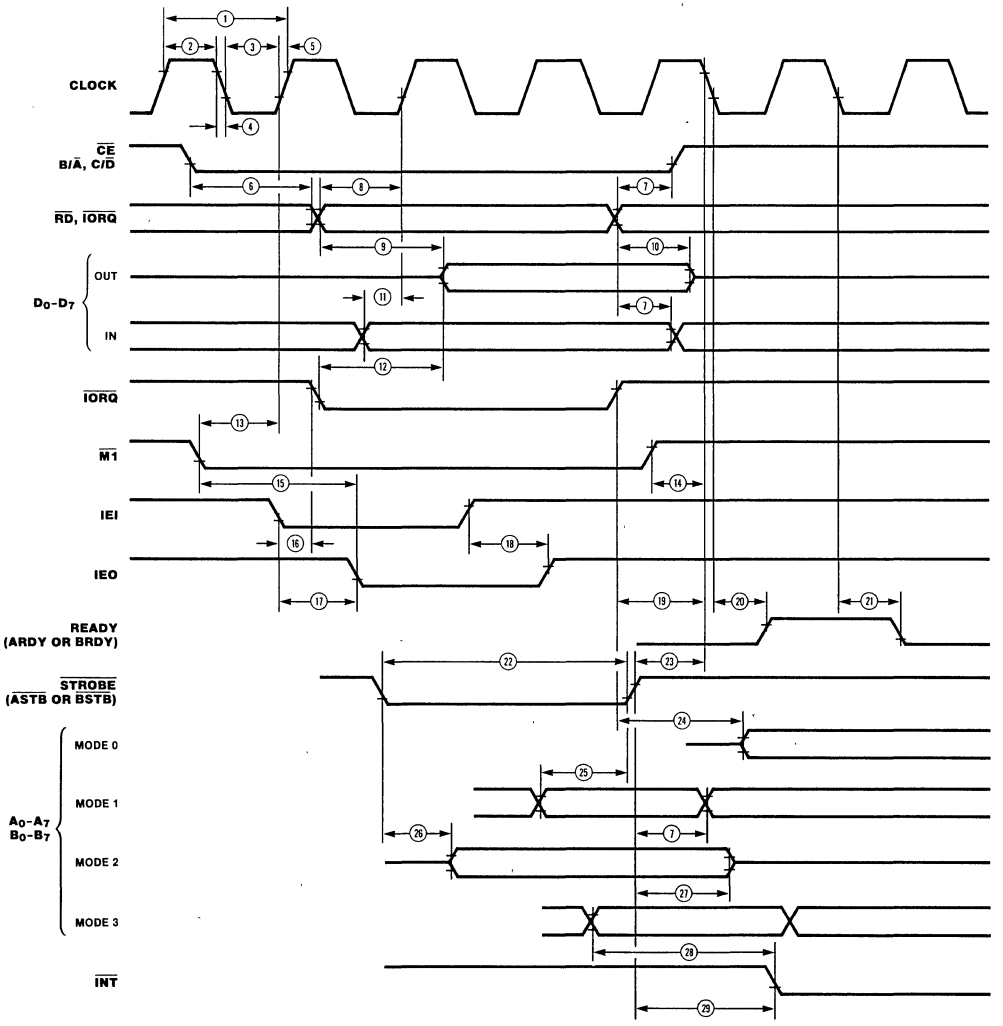


Figure 19. Return From Interrupt

AC Characteristics



Number	Symbol	Parameter	Z-80 PIO Min (ns)	Z-80 PIO Max (ns)	Z-80A PIO Min (ns)	Z-80A PIO Max (ns)	Z-80B PIO[9] Min (ns)	Z-80B PIO[9] Max (ns)	Comment
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCh	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	TsCS(RI)	$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$ to $\overline{RD}$, $\overline{IORQ}$ ↓ Setup Time	50		50		50		[6]
7	Th	Any Hold Times for Specified Setup Time	0		0		0	0	
8	TsRI(C)	$\overline{RD}$, $\overline{IORQ}$ to Clock ↑ Setup Time	115		115		70		
9	TdRI(DO)	$\overline{RD}$, $\overline{IORQ}$ ↓to Data Out Delay		430		380		300	[2]
10	TdRI(DOs)	$\overline{RD}$, $\overline{IORQ}$ ↑ to Data Out Float Delay		160		110		70	
11	TsDI(C)	Data In to Clock ↑ Setup Time	50		50		40		CL = 50 pF
12	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)	340		160		120		[3]
13	TsM1(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup Time	210		90		70		
14	TsM1(Cf)	$\overline{M1}$ ↑ to Clock ↓ Setup Time ($\overline{M1}$ Cycle)	0		0		0		[8]
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt Immediately Preceding $\overline{M1}$ ↓)		300		190		100	[5, 7]
16	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	140		140		100		[7]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		120	[5] CL = 50 pF
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED Decode)		210		160		160	[5]
19	TcIO(C)	$\overline{IORQ}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		200		170		
20	TdC(RDYr)	Clock ↓ to READY ↑ Delay	200		190		170		[5] CL = 50 pF
21	TdC(RDYf)	Clock ↓ to READY ↑ Delay	150		140		120		[5]
22	TwSTB	$\overline{STROBE}$ Pulse Width	150		150		120		[4]
23	TsSTB(C)	$\overline{STROBE}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		220		150		[5]
24	TdIO(PD)	$\overline{IORQ}$ ↑ to PORT DATA Stable Delay (Mode 0)		200		180		160	[5]
25	TsPD(STB)	PORT DATA to $\overline{STROBE}$ ↑ Setup Time (Mode 1)	260		230		190		
26	TdSTB(PD)	$\overline{STROBE}$ ↓ to PORT DATA Stable (Mode 2)		230		210		180	[5]
27	TdSTB(PDr)	$\overline{STROBE}$ ↑ to PORT DATA Float Delay (Mode 2)		200		180		160	CL = 50 pF
28	TdPD(INT)	PORT DATA Match to $\overline{INT}$ ↓ Delay (Mode 3)		540		490		430	
29	TdSTB(INT)	$\overline{STROBE}$ ↑ to $\overline{INT}$ ↓ Delay		490		440		350	

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase TdRI(DO) by 10 ns for each 50 pF increase in load up to 200 pF max.
[3] Increase TdIO(DOI) by 10 ns for each 50 pF, increase in loading up to 200 pF max.
[4] For Mode 2· TwSTB > TsPD(STB)
[5] Increase these values by 2 ns for each 10 pF increase in loading up to 100 pF max.
[6] TsCS(RI) may be reduced. However, the time subtracted from TsCS(RI) will be added to TdRI(DO)
[7] 2.5 TcC > (N-2)TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL Buffer Delay, if any.
[8] $\overline{M1}$ must be active for a minimum of two clock cycles to reset the PIO.
[9] Z80B PIO numbers are preliminary and subject to change.

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in 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.

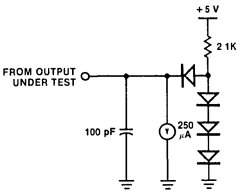
Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0° to +70°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- -40°C to +85°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- -55° to +125°C, $+4.75\ V \leq V_{CC} \leq +5.5\ V$

The product number for each operating temperature range may be found in the Ordering Information section.

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.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+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} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current	-10.0	+10.0	µA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10.0	+10.0	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100.0	mA	V_{OH} = 1.5V
I_{OHD}	Darlington Drive Current	-1.5	3.8	mA	R_{EXT} = 390 Ω

Over specified temperature and voltage range

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		10	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range, f = $1MH_z$

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8420	CE	2.5 MHz	Z80 PIO (40-pin)	Z8420A	DE	4.0 MHz	Z80A PIO (40-pin)
Z8420	CM	2.5 MHz	Same as above	Z8420A	DS	4.0 MHz	Same as above
Z8420	CMB	2.5 MHz	Same as above	Z8420A	PE	4.0 MHz	Same as above
Z8420	CS	2.5 MHz	Same as above	Z8420A	PS	4.0 MHz	Same as above
Z8420	DE	2.5 MHz	Same as above	Z8420B	CE	6.0 MHz	Z80B PIO (40-pin)
Z8420	DS	2.5 MHz	Same as above	Z8420B	CM	6.0 MHz	Same as above
Z8420	PE	4.0 MHz	Same as above	Z8420B	CMB	6.0 MHz	Same as above
Z8420	PS	4.0 MHz	Same as above	Z8420B	CS	6.0 MHz	Same as above
Z8420A	CE	4.0 MHz	Z80A PIO (40-pin)	Z8420B	DE	6.0 MHz	Same as above
Z8420A	CM	4.0 MHz	Same as above	Z8420B	DS	6.0 MHz	Same as above
Z8420A	CMB	4.0 MHz	Same as above	Z8420B	PE	6.0 MHz	Same as above
Z8420A	CS	4.0 MHz	Same as above	Z8420B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = 55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8430
Z80® CTC Counter/ Timer Circuit

Zilog

Product Specification

March 1981

Features

- Four independently programmable counter/timer channels, each with a readable downcounter and a selectable 16 or 256 prescaler. Downcounters are reloaded automatically at zero count.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Selectable positive or negative trigger initiates timer operation.
- Standard Z-80 Family daisy-chain interrupt structure provides fully vectored, prioritized interrupts without external logic. The CTC may also be used as an interrupt controller.
- Interfaces directly to the Z-80 CPU or—for baud rate generation—to the Z-80 SIO.

General Description

The Z-80 CTC four-channel counter/timer can be programmed by system software for a broad range of counting and timing applications. The four independently programmable channels of the Z-80 CTC satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock rate generation.

System design is simplified because the CTC connects directly to both the Z-80 CPU and the Z-80 SIO with no additional logic. In larger systems, address decoders and buffers may be required.

Programming the CTC is straightforward: each channel is programmed with two bytes; a third is necessary when interrupts are enabled. Once started, the CTC counts down, reloads its time constant automatically, and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because only one vector need be specified; the CTC internally generates a unique vector for each channel.

The Z-80 CTC requires a single +5 V power supply and the standard Z-80 single-phase system clock. It is fabricated with n-channel silicon-gate depletion-load technology, and packaged in a 28-pin plastic or ceramic DIP.

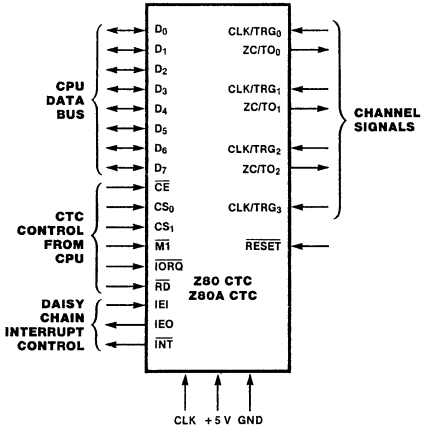


Figure 1. Pin Functions

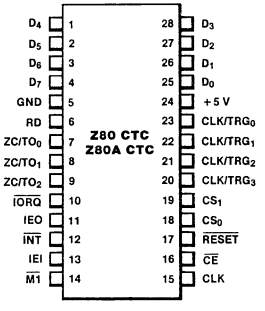


Figure 2. Pin Assignments

Functional Description

The Z-80 CTC has four independent counter/timer channels. Each channel is individually programmed with two words: a control word and a time-constant word. The control word selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. If the timing mode is selected, the control word also sets a prescaler, which divides the system clock by either 16 or 256. The time-constant word is a value from 1 to 256.

During operation, the individual counter channel counts down from the preset time constant value. In counter mode operation the counter decrements on each of the CLK/TRG input pulses until zero count is reached. Each decrement is synchronized by the system clock. For counts greater than 256, more than one counter can be cascaded. At zero count, the down-counter is automatically reset with the time constant value.

The timer mode determines time intervals as small as 4 μs (Z-80A) or 6.4 μs (Z-80) without additional logic or software timing loops. Time intervals are generated by dividing the system clock with a prescaler that decrements a preset down-counter.

Thus, the time interval is an integral multiple of the clock period, the prescaler value (16 or 256) and the time constant that is preset in the down-counter. A timer is triggered automatically when its time constant value is programmed, or by an external CLK/TRG input.

Three channels have two outputs that occur at zero count. The first output is a zero-count/timeout pulse at the ZC/TO output. The fourth channel (Channel 3) does not have a ZC/TO output; interrupt request is the only output available from Channel 3.

The second output is Interrupt Request ($\overline{INT}$), which occurs if the channel has its interrupt enabled during programming. When the Z-80 CPU acknowledges Interrupt Request, the Z-80 CTC places an interrupt vector on the data bus.

The four channels of the Z-80 CTC are fully prioritized and fit into four contiguous slots in a standard Z-80 daisy-chain interrupt structure. Channel 0 is the highest priority and Channel 3 the lowest. Interrupts can be individually enabled (or disabled) for each of the four channels.

Architecture

The CTC has four major elements, as shown in Figure 3.

- CPU bus I/O
- Channel control logic
- Interrupt logic
- Counter/timer circuits

CPU Bus I/O. The CPU bus I/O circuit decodes the address inputs, and interfaces the CPU data and control signals to the CTC for distribution on the internal bus.

Internal Control Logic. The CTC internal control logic controls overall chip operating functions such as the chip enable, reset, and read/write logic.

Interrupt Logic. The interrupt control logic ensures that the CTC interrupts interface properly with the Z-80 CPU interrupt system. The logic controls the interrupt priority of the CTC as a function of the IEI signal. If IEI is High, the CTC has priority. During interrupt

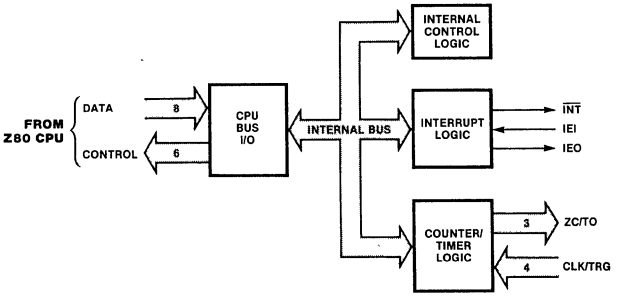


Figure 3. Functional Block Diagram

Architecture (Continued)

processing, the interrupt logic holds IEO Low, which inhibits the interrupt operation on lower priority devices. If the IEI input goes Low, priority is relinquished and the interrupt logic drives IEO Low.

If a channel is programmed to request an interrupt, the interrupt logic drives IEO Low at the zero count, and generates an $\overline{INT}$ signal to the Z-80 CPU. When the Z-80 CPU responds with interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$), then the interrupt logic arbitrates the CTC internal priorities, and the interrupt control logic places a unique interrupt vector on the data bus.

If an interrupt is pending, the interrupt logic holds IEO Low. When the Z-80 CPU issues a Return From Interrupt (RETI) instruction, each peripheral device decodes the first byte (ED_{16}). If the device has a pending interrupt, it raises IEO (High) for one $\overline{M1}$ cycle. This ensures that all lower priority devices can decode the entire RETI instruction and reset properly.

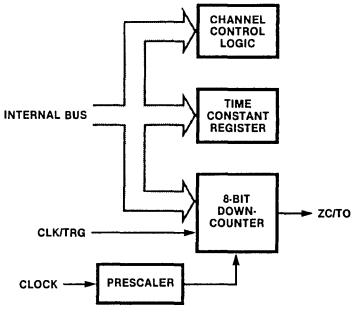


Figure 4. Counter/Timer Block Diagram

Counter/Timer Circuits. The CTC has four independent counter/timer circuits, each containing the logic shown in Figure 4.

Channel Control Logic. The channel control logic receives the 8-bit channel control word when the counter/timer channel is programmed. The channel control logic decodes the control word and sets the following operating conditions:

- Interrupt enable (or disable)
- Operating mode (timer or counter)
- Timer mode prescaler factor (16 or 256)
- Active slope for CLK/TRG input
- Timer mode trigger (automatic or CLK/TRG input)
- Time constant data word to follow
- Software reset

Time Constant Register. When the counter/timer channel is programmed, the time constant register receives and stores an 8-bit time constant value, which can be anywhere from 1 to 256 (0 = 256). This constant is automatically loaded into the down-counter when the counter/timer channel is initialized, and subsequently after each zero count.

Prescaler. The prescaler, which is used only in timer mode, divides the system clock frequency by a factor of either 16 or 256. The prescaler output clocks the down-counter during timer operation. The effect of the prescaler on the down-counter is a multiplication of the system clock period by 16 or 256. The prescaler factor is programmed by bit 5 of the channel control word.

Down-Counter. Prior to each count cycle, the down-counter is loaded with the time constant register contents. The counter is then decremented one of two ways, depending on operating mode:

- By the prescaler output (timer mode)
- By the trigger pulses into the CLK/TRG input (counter mode)

Without disturbing the down-count, the Z-80 CPU can read the count remaining at any time by performing an I/O read operation at the port address assigned to the CTC channel. When the down-counter reaches the zero count, the ZC/TO output generates a positive-going pulse. When the interrupt is enabled, zero count also triggers an interrupt request signal ($\overline{INT}$) from the interrupt logic.

Programming

Each Z-80 CTC channel must be programmed prior to operation. Programming consists of writing two words to the I/O port that corresponds to the desired channel. The first word is a control word that selects the operating mode and other parameters; the second word is a time constant, which is a binary data word with a value from 1 to 256. A time constant word must be preceded by a channel control word.

After initialization, channels may be reprogrammed at any time. If updated control and time constant words are written to a channel during the count operation, the count continues to zero before the new time constant is loaded into the counter.

If the interrupt on any Z-80 CTC channel is enabled, the programming procedure should also include an interrupt vector. Only one vector is required for all four channels, because the interrupt logic automatically modifies the vector for the channel requesting service.

A control word is identified by a 1 in bit 0. A 0 in bit 2 indicates a time constant word is to follow. Interrupt vectors are always addressed to Channel 0, and identified by a 0 in bit 0.

Addressing. During programming, channels are addressed with the channel select pins CS_1 and CS_2. A 2-bit binary code selects the appropriate channel as shown in the following table.

Channel	CS_1	CS_0
0	0	0
1	0	1
2	1	0
3	1	1

Reset. The CTC has both hardware and software resets. The hardware reset terminates all down-counts and disables all CTC interrupts by resetting the interrupt bits in the control registers. In addition, the ZC/TO and Interrupt outputs go inactive, IEO reflects IEI, and D_0-D_7 go to the high-impedance state. All channels must be completely reprogrammed after a hardware reset.

The software reset is controlled by bit 1 in the channel control word. When a channel receives a software reset, it stops counting. When a software reset is used, the other bits in the control word also change the contents of the channel control register. After a software reset a new time constant word must be written to the same channel.

If the channel control word has both bits D_1 and D_2 set to 1, the addressed channel stops operating, pending a new time constant word. The channel is ready to resume after the new constant is programmed. In timer mode, if D_3 = 0, operation is triggered automatically when the time constant word is loaded.

Channel Control Word Programming. The channel control word is shown in Figure 5. It sets the modes and parameters described below.

Interrupt Enable. D_7 enables the interrupt, so that an interrupt output ($\overline{INT}$) is generated at zero count. Interrupts may be programmed in either mode and may be enabled or disabled at any time.

Operating Mode. D_6 selects either timer or counter mode.

Prescaler Factor. (Timer Mode Only). D_5 selects factor—either 16 or 256.

Trigger Slope. D_4 selects the active edge or slope of the CLK/TRG input pulses. Note that reprogramming the CLK/TRG slope during operation is equivalent to issuing an active edge. If the trigger slope is changed by a control word update while a channel is pending operation in timer mode, the result is the same as a CLK/TRG pulse and the timer starts. Similarly, if the channel is in counter mode, the counter decrements.

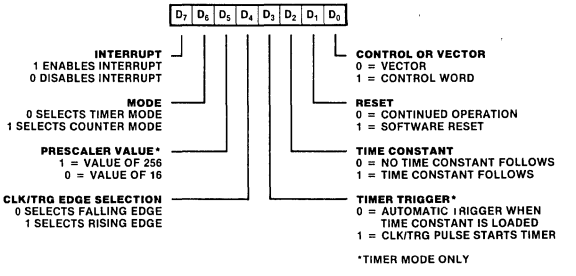


Figure 5. Channel Control Word

Programming (Continued)

Trigger Mode (Timer Mode Only). D_3 selects the trigger mode for timer operation. When D_3 is reset to 0, the timer is triggered automatically. The time constant word is programmed during an I/O write operation, which takes one machine cycle. At the end of the write operation there is a setup delay of one clock period. The timer starts automatically (decrements) on the rising edge of the second clock pulse (T_2) of the machine cycle following the write operation. Once started, the timer runs continuously. At zero count the timer reloads automatically and continues counting without interruption or delay, until stopped by a reset.

When D_3 is set to 1, the timer is triggered externally through the CLK/TRG input. The time constant word is programmed during an I/O write operation, which takes one machine cycle. The timer is ready for operation on the rising edge of the second clock pulse (T_2) of the following machine cycle. Note that the first timer decrement follows the active edge of the CLK/TRG pulse by a delay time of one clock cycle if a minimum setup time to the rising edge of clock is met. If this minimum is not met, the delay is extended by another clock period. Consequently, for immediate triggering, the CLK/TRG input must precede T_2 by one clock cycle plus its minimum setup time. If the minimum time is not met, the timer will start on the third clock cycle (T_3).

Once started the timer operates continuously, without interruption or delay, until stopped by a reset.

Time Constant to Follow. A 1 in D_2 indicates that the next word addressed to the selected channel is a time constant data word for the time constant register. The time constant word may be written at any time.

A 0 in D_2 indicates no time constant word is to follow. This is ordinarily used when the channel is already in operation and the new channel control word is an update. A channel will not operate without a time constant value. The only way to write a time constant value is to write a control word with D_2 set.

Software Reset. Setting D_1 to 1 causes a software reset, which is described in the Reset section.

Control Word. Setting D_0 to 1 identifies the word as a control word.

Time Constant Programming. Before a channel can start counting it must receive a time constant word from the CPU. During programming or reprogramming, a channel control word in which bit 2 is set must precede the time constant word to indicate that the next word is a time constant. The time constant word can be any value from 1 to 256 (Figure 6). Note that 00_{16} is interpreted as 256.

In timer mode, the time interval is controlled by three factors:

- The system clock period (ϕ)
- The prescaler factor (P), which multiplies the interval by either 16 or 256
- The time constant (T), which is programmed into the time constant register

Consequently, the time interval is the product of $\phi \times P \times T$. The minimum timer resolution is $16 \times \phi$ (4 μs with a 4 MHz clock). The maximum timer interval is $256 \times \phi \times 256$ (16.4 ms with a 4 MHz clock). For longer intervals timers may be cascaded.

Interrupt Vector Programming. If the Z-80 CTC has one or more interrupts enabled, it can supply interrupt vectors to the Z-80 CPU. To do so, the Z-80 CTC must be pre-programmed with the most-significant five bits of the interrupt vector. Programming consists of writing a vector word to the I/O port corresponding to the Z-80 CTC Channel 0. Note that D_0 of the vector word is always zero, to distinguish the vector from a channel control word. D_1 and D_2 are not used in programming the vector word. These bits are supplied by the interrupt logic to identify the channel requesting interrupt service with a unique interrupt vector (Figure 7). Channel 0 has the highest priority.

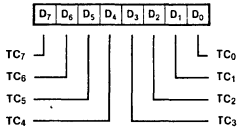


Figure 6. Time Constant Word

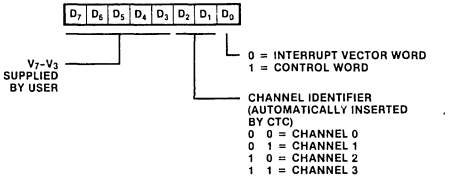


Figure 7. Interrupt Vector Word

Pin Description

$\overline{CE}$. *Chip Enable* (input, active Low). When enabled the CTC accepts control words, interrupt vectors, or time constant data words from the data bus during an I/O write cycle; or transmits the contents of the down-counter to the CPU during an I/O read cycle. In most applications this signal is decoded from the eight least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four counter-timer channels.

CLK. *System Clock* (input). Standard single-phase Z-80 system clock.

CLK/TRG$_0$-CLK/TRG$_3$. *External Clock/Timer Trigger* (input, user-selectable active High or Low). Four pins corresponding to the four Z-80 CTC channels. In counter mode, every active edge on this pin decrements the down-counter. In timer mode, an active edge starts the timer.

CS$_0$-CS$_1$. *Channel Select* (inputs active High). Two-bit binary address code selects one of the four CTC channels for an I/O write or read (usually connected to A_0 and A_1).

D$_0$-D$_7$. *System Data Bus* (bidirectional, 3-state). Transfers all data and commands between the Z-80 CPU and the Z-80 CTC.

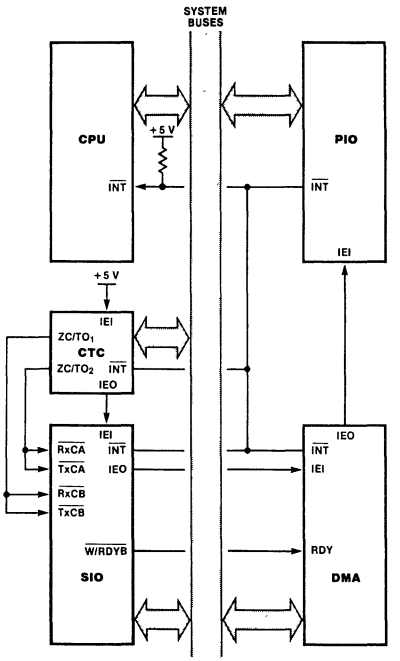


Figure 8. A Typical Z-80 Environment

IEI. *Interrupt Enable In* (input, active High). A High indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z-80 CPU.

IEO. *Interrupt Enable Out* (output, active High). High only if IEI is High and the Z-80 CPU is not servicing an interrupt from any Z-80 CTC channel. IEO blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). Low when any Z-80 CTC channel that has been programmed to enable interrupts has a zero-count condition in its down-counter.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). Used with $\overline{CE}$ and $\overline{RD}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC. During a write cycle, $\overline{IORQ}$ and $\overline{CE}$ are active and $\overline{RD}$ inactive. The Z-80 CTC does not receive a specific write signal; rather, it internally generates its own from the inverse of an active $\overline{RD}$ signal. In a read cycle, $\overline{IORQ}$, $\overline{CE}$ and $\overline{RD}$ are active; the contents of the down-counter are read by the Z-80 CPU. If $\overline{IORQ}$ and $\overline{M1}$ are both true, the CPU is acknowledging an interrupt request, and the highest priority interrupting channel places its interrupt vector on the Z-80 data bus.

$\overline{M1}$. *Machine Cycle One* (input from CPU, active Low). When $\overline{M1}$ and $\overline{IORQ}$ are active, the Z-80 CPU is acknowledging an interrupt. The Z-80 CTC then places an interrupt vector on the data bus if it has highest priority, and if a channel has requested an interrupt ($\overline{INT}$).

$\overline{RD}$. *Read Cycle Status* (input, active Low). Used in conjunction with $\overline{IORQ}$ and $\overline{CE}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC.

$\overline{RESET}$. *Reset* (input active Low). Terminates all down-counts and disables all interrupts by resetting the interrupt bits in all control registers; the ZC/TO and the Interrupt outputs go inactive; IEO reflects IEI; D$_0$–D$_7$ go to the high-impedance state.

ZC/TO$_0$-ZC/TO$_2$. *Zero Count/Timeout* (output, active High). Three ZC/TO pins corresponding to Z-80 CTC channels 2 through 0 (Channel 3 has no ZC/TO pin). In both counter and timer modes the output is an active High pulse when the down-counter decrements to zero.

Timing

Read Cycle Timing. Figure 9 shows read cycle timing. This cycle reads the contents of a down-counter without disturbing the count. During clock cycle T_2, the Z-80 CPU initiates a read cycle by driving the following inputs Low: $\overline{RD}$, $\overline{IORQ}$, and $\overline{CE}$. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be read. $\overline{M1}$ must be High to distinguish this cycle from an interrupt acknowledge. No additional wait states are allowed.

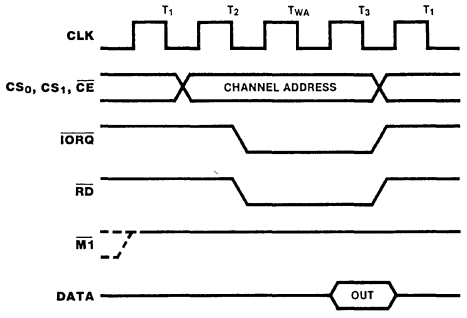


Figure 9. Read Cycle Timing

Write Cycle Timing. Figure 10 shows write cycle timing for loading control, time constant or vector words.

The CTC does not have a write signal input, so it generates one internally when the read ($\overline{RD}$) input is High during T_1. During T_2 $\overline{IORQ}$ and $\overline{CE}$ inputs are Low. $\overline{M1}$ must be High to distinguish a write cycle from an interrupt acknowledge. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be addressed, and the word being written is placed on the Z-80 data bus. The data word is latched into the appropriate register with the rising edge of clock cycle T_{WA}. No additional wait states are allowed.

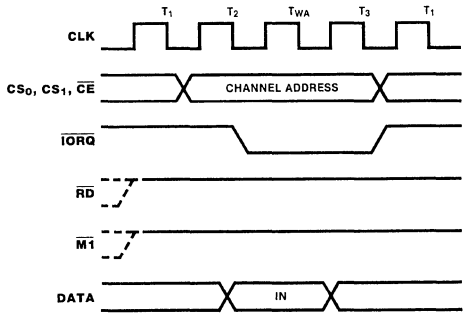


Figure 10. Write Cycle Timing

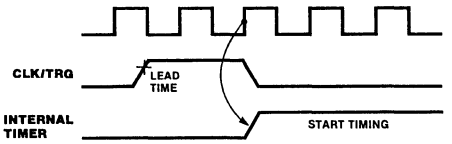


Figure 11. Timer Mode Timing

Timer Operation. In the timer mode, a CLK/TRG pulse input starts the timer (Figure 11) on the second succeeding rising edge of CLK. The trigger pulse is asynchronous, and it must have a minimum width. A minimum lead time (210 ns) is required between the active edge of the CLK/TRG and the next rising edge of CLK to enable the prescaler on the following clock edge. If the CLK/TRG edge occurs closer than this, the initiation of the timer function is delayed one clock cycle. This corresponds to the startup timing discussed in the programming section. The timer can also be started automatically if so programmed by the channel control word.

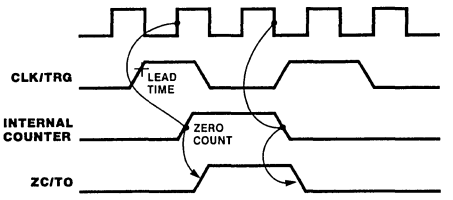


Figure 12. Counter Mode Timing

Counter Operation. In the counter mode, the CLK/TRG pulse input decrements the down-counter. The trigger is asynchronous, but the count is synchronized with CLK. For the decrement to occur on the next rising edge of CLK, the trigger edge must precede CLK by a minimum lead time as shown in Figure 12. If the lead time is less than specified, the count is delayed by one clock cycle. The trigger pulse must have a minimum width, and the trigger period must be at least twice the clock period.

The ZC/TO output occurs immediately after zero count, and follows the rising CLK edge.

Interrupt Operation

The Z-80 CTC follows the Z-80 system interrupt protocol for nested priority interrupts and return from interrupt, wherein the interrupt priority of a peripheral is determined by its location in a daisy chain. Two lines—IEI and IEO—in the CTC connect it to the system daisy chain. The device closest to the +5 V supply has the highest priority (Figure 13). For additional information on the Z-80 interrupt structure, refer to the *Z-80 CPU Product Specification* and the *Z-80 CPU Technical Manual.*

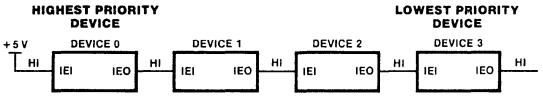


Figure 13. Daisy-Chain Interrupt Priorities

Within the Z-80 CTC, interrupt priority is predetermined by channel number: Channel 0 has the highest priority, and Channel 3 the lowest. If a device or channel is being serviced with an interrupt routine, it cannot be interrupted by a device or channel with lower priority until service is complete. Higher priority devices or channels may interrupt the servicing of lower priority devices or channels.

A Z-80 CTC channel may be programmed to request an interrupt every time its down-counter reaches zero. Note that the CPU must be programmed for interrupt mode 2. Some time after the interrupt request, the CPU sends an interrupt acknowledge. The CTC interrupt control logic determines the highest priority channel that is requesting an interrupt. Then, if the CTC IEI input is High (indicating that it has priority within the system daisy chain) it places an 8-bit interrupt vector on the system data bus. The high-order five bits of this vector were written to the CTC during the programming process; the next two bits are provided by the CTC interrupt control logic as a binary code that identifies the highest priority channel requesting an interrupt; the low-order bit is always zero.

Interrupt Acknowledge Timing. Figure 14 shows interrupt acknowledge timing. After an interrupt request, the Z-80 CPU sends an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). All channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active—about two clock cycles earlier than $\overline{IORQ}$. $\overline{RD}$ is High to distinguish this cycle from an instruction fetch.

The CTC interrupt logic determines the highest priority channel requesting an interrupt. If the CTC interrupt enable input (IEI) is High, the highest priority interrupting channel within the CTC places its interrupt vector on the data bus when $\overline{IORQ}$ goes Low. Two wait states (T_{WA}) are automatically inserted at this time to allow the daisy chain to stabilize. Additional wait states may be added.

Return from Interrupt Timing. At the end of an interrupt service routine the RETI (Return From Interrupt) instruction initializes the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the 2-byte RETI code internally and determines whether it is intended for a channel being serviced. Figure 15 shows RETI timing.

If several Z-80 peripherals are in the daisy chain, IEI settles active (High) on the chip currently being serviced when the opcode ED_{16} is decoded. If the following opcode is $4D_{16}$, the peripheral being serviced is released and its IEO becomes active. Additional wait states are allowed.

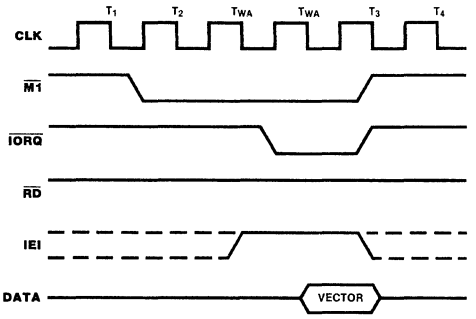


Figure 14. Interrupt Acknowledge Timing

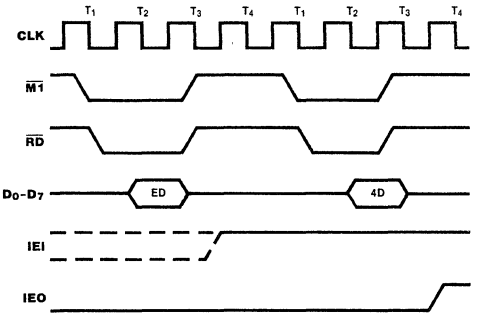


Figure 15. Return From Interrupt Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND.....-0.3 V to +7.0 V
Operating Ambient Temperature .....As Specified in Ordering Information
Storage Temperature.....-65°C to +150°C

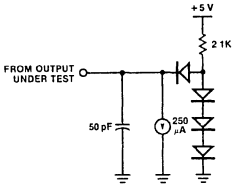
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- −40°C to +85°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- −55°C to +125°C, $+4.5\ V \leq V_{CC} \leq +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section.



DC Characteristics

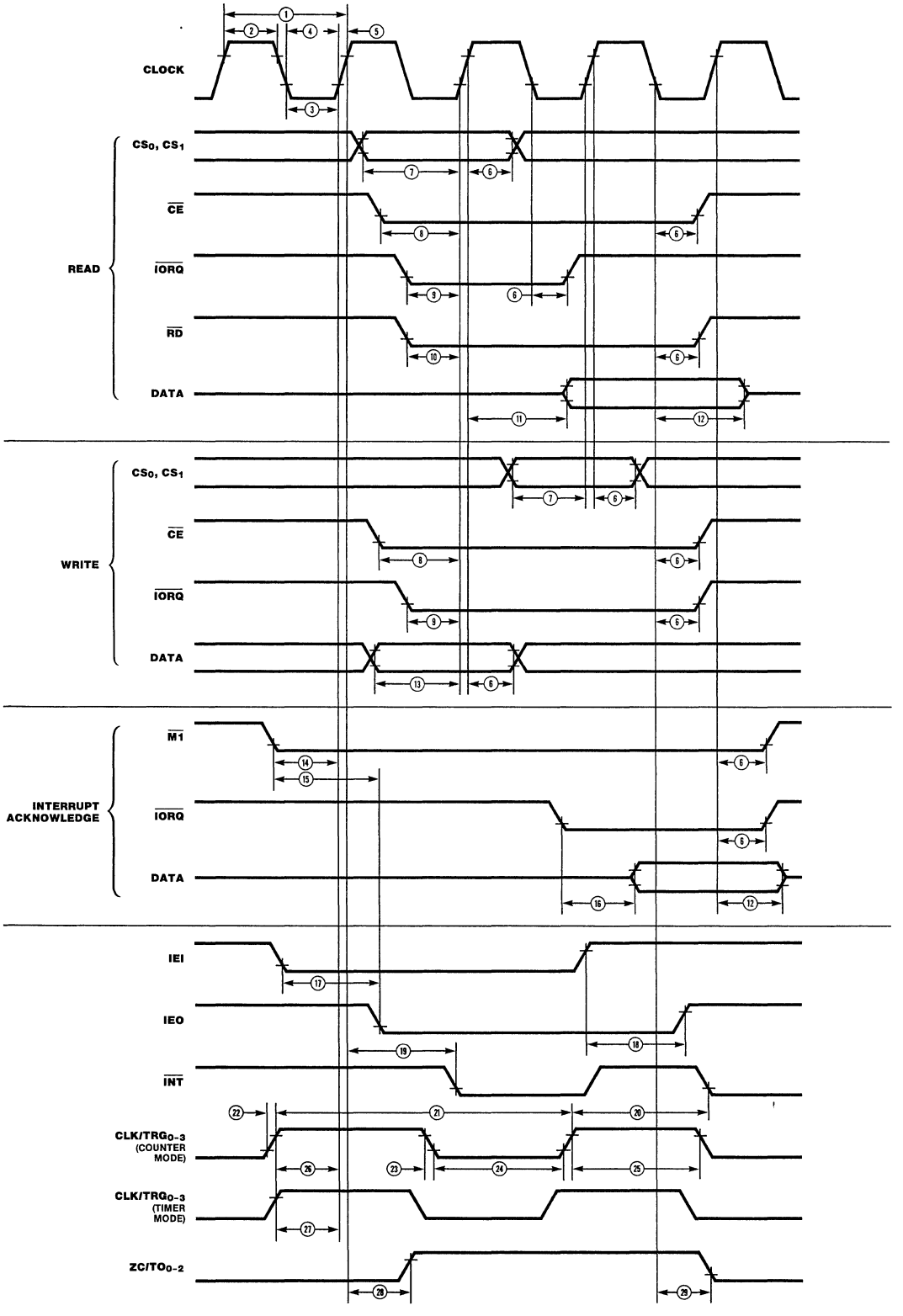
Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = 250 µA
I_{CC}	Power Supply Current		+120	mA	
I_{LI}	Input Leakage Current		+10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	3-State Output Leakage Current in Float		+10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	3-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4 V
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5 V, R_{EXT} = 390Ω

Capacitance

Symbol	Parameter	Max	Unit	Condition
CLK	Clock Capacitance	20	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

T_A = 25°C, f = 1 MHz

AC Characteristics



Number	Symbol	Parameter	Z-80 CTC Min (ns)	Z-80 CTC Max (ns)	Z-80A CTC Min (ns)	Z-80A CTC Max (ns)	Z-80B CTC Min (ns)	Z-80B CTC Max (ns)	Notes
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCH	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	Th	All Hold Times	0		0		0		
7	TsCS(C)	CS to Clock ↑ Setup Time	250		160		100		
8	TsCE(C)	$\overline{CE}$ to Clock ↑ Setup Time	200		150		100		
9	TsIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time	250		115		70		
10	TsRD(C)	$\overline{RD}$ ↓ to Clock ↑ Setup Time	240		115		70		
11	TdC(DO)	Clock ↓ to Data Out Delay		240		200		130	[2]
12	TdC(DOz)	Clock ↑ to Data Out Float Delay		230		110		90	
13	TsDI(C)	Data In to Clock ↑ Setup Time	60		50		40		
14	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		70		
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt immediately preceding $\overline{M1}$)		300		190		130	[3]
16	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160		110	[2]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		100	[3]
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (After ED Decode)		220		160		110	[3]
19	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay	(TcC + 200)		(TcC + 140)		TcC + 120		[4]
20	TdCLK(INT)	CLK/TRG ↑ to $\overline{INT}$↓							
		tsCTR(C) satisfied	(TcC + 230)		(TcC + 160)		TcC + 130		[5]
		tsCTR(C) not satisfied	(2TcC + 530)		(2TcC + 370)		2TcC + 280		[5]
21	TcCTR	CLK/TRG Cycle Time	(2TcC)		(2TcC)		2TcC		[5]
22	TrCTR	CLK/TRG Rise Time		50		50		40	
23	TfCTR	CLK/TRG Fall Time		50		50		40	
24	TwCTRl	CLK/TRG Width (Low)	200		200		120		
25	TwCTRh	CLK/TRG Width (High)	200		200		120		
26	TsCTR(Cs)	CLK/TRG ↑ to Clock ↑ Setup Time for Immediate Count	300		210		150		[5]
27	TsCTR(Ct)	CLK/TRG ↑ to Clock ↑ Setup Time for enabling of Prescaler on following clock↑	210		210		150		[4]
28	TdC(ZC/TOr)	Clock ↑ to ZC/TO ↑ Delay		260		190		140	
29	TdC(ZC/TOf)	Clock ↓ to ZC/TO ↓ Delay		190		190		140	

[A] 2.5 TcC > (n-2) TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL buffer delay, if any.
[B] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

NOTES.
[1] TcC = TwCh + TwC1 + TrC + TfC.
[2] Increase delay by 10 ns for each 50 pF increase in loading, 200 pF maximum for data lines, and 100 pF for control lines.
[3] Increase delay by 2 ns for each 10 pF increase in loading, 100 pF maximum.
[4] Timer mode.
[5] Counter mode.
[6] $\overline{RESET}$ must be active for a minimum of 3 clock cycles

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8430	CE	2.5 MHz	Z80 CTC (28-pin)	Z8430A	DE	4.0 MHz	Z80A CTC (28-pin)
Z8430	CM	2.5 MHz	Same as above	Z8430A	DS	4.0 MHz	Same as above
Z8430	CMB	2.5 MHz	Same as above	Z8430A	PE	4.0 MHz	Same as above
Z8430	CS	2.5 MHz	Same as above	Z8430A	PS	4.0 MHz	Same as above
Z8430	DE	2.5 MHz	Same as above	Z8430B	CE	6.0 MHz	Z80B CTC (28-pin)
Z8430	DS	2.5 MHz	Same as above	Z8430B	CM	6.0 MHz	Same as above
Z8430	PE	2.5 MHz	Same as above	Z8430B	CMB	6.0 MHz	Same as above
Z8430	PS	2.5 MHz	Same as above	Z8430B	CS	6.0 MHz	Same as above
Z8430A	CE	4.0 MHz	Z80A CTC (28-pin)	Z8430B	DE	6.0 MHz	Same as above
Z8430A	CM	4.0 MHz	Same as above	Z8430B	DS	6.0 MHz	Same as above
Z8430A	CMB	4.0 MHz	Same as above	Z8430B	PE	6.0 MHz	Same as above
Z8430A	CS	4.0 MHz	Same as above	Z8430B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8440
Z80® SIO Serial Input/Output Controller

Zilog

Product Specification

March 1981

Features

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA

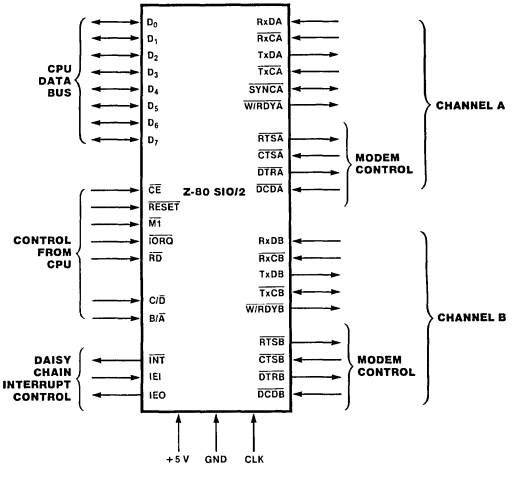


Figure 1. Z-80 SIO/2 Pin Functions

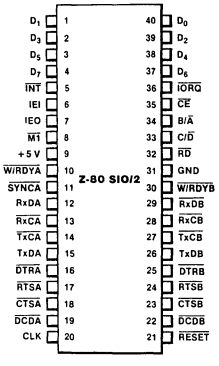


Figure 2. Z-80 SIO/2 Pin Assignments

General Description (Continued)

control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well suited to many other CPUs.

The Z-80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 family single-phase clock.

Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{SYNCB}$
- Z-80 SIO/1 lacks $\overline{DTRB}$
- Z-80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffer-

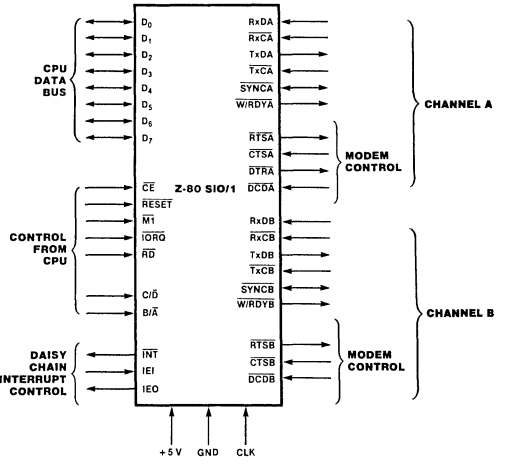


Figure 3. Z-80 SIO/1 Pin Functions

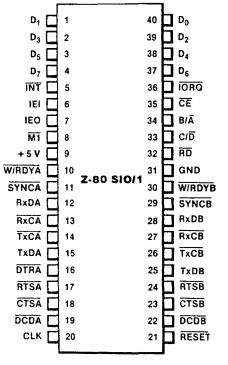


Figure 4. Z-80 SIO/1 Pin Assignments

Pin Description (Continued)

ing does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general-purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal 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 SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. If $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$ and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be

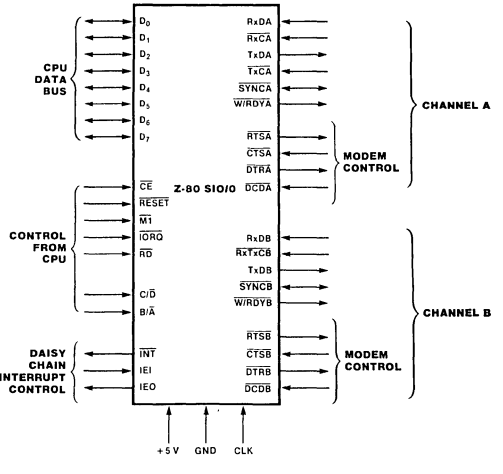


Figure 5. Z-80 SIO/0 Pin Functions

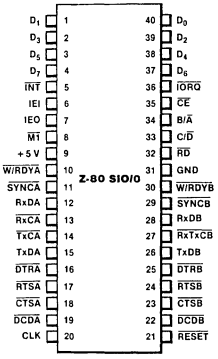


Figure 6. Z-80 SIO/0 Pin Assignments

Pin Description (Continued)

rewritten after the SIO is reset and before data is transmitted or received.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{\text{SYNCB}}$ is omitted.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{\text{TxC}}$.

$\overline{\text{W}}$/$\overline{\text{RDYA}}$, $\overline{\text{W}}$/$\overline{\text{RDYB}}$. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

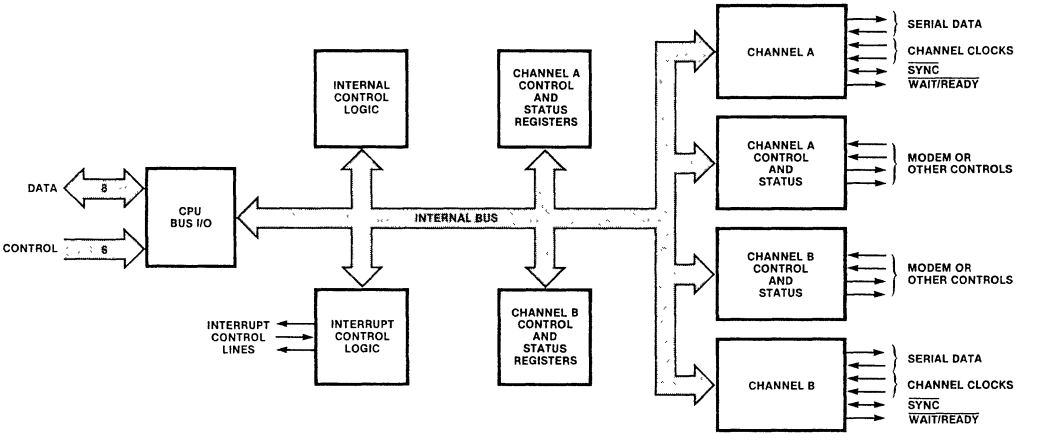


Figure 7. Block Diagram

Functional Description

The functional capabilities of the Z-80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

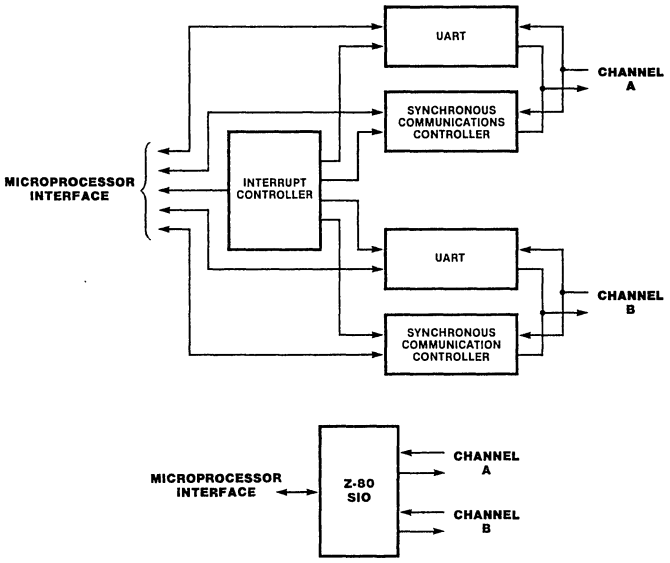


Figure 8. Conventional Devices Replaced by the Z-80 SIO

Data Communication Capabilities

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z-80 SIO Technical Manual.*

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with a Z-80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync

Data Communication Capabilities (Continued)

characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

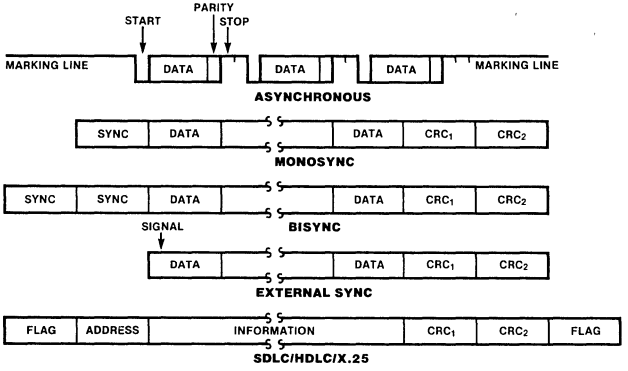


Figure 9. Some Z-80 SIO Protocols

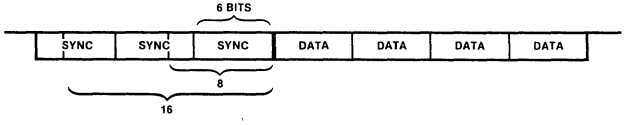


Figure 10.

I/O Interface Capabilities

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{\text{CTS}}$), Data Carrier Detect ($\overline{\text{DCD}}$) and Synchronization ($\overline{\text{SYNC}}$) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

I/O Interface Capabilities (Continued)

In a Z-80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z-80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{\text{WAIT}}$ line in the CPU block-transfer mode or as a $\overline{\text{READY}}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

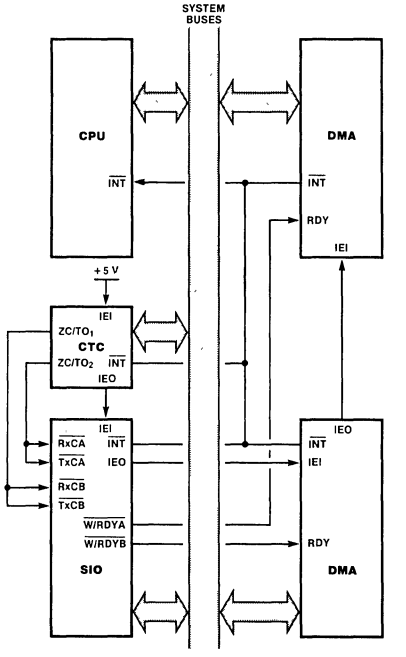


Figure 11. Typical Z-80 Environment

Internal Structure

The internal structure of the device includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Write Register Functions	
WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

Internal Structure (Continued)

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

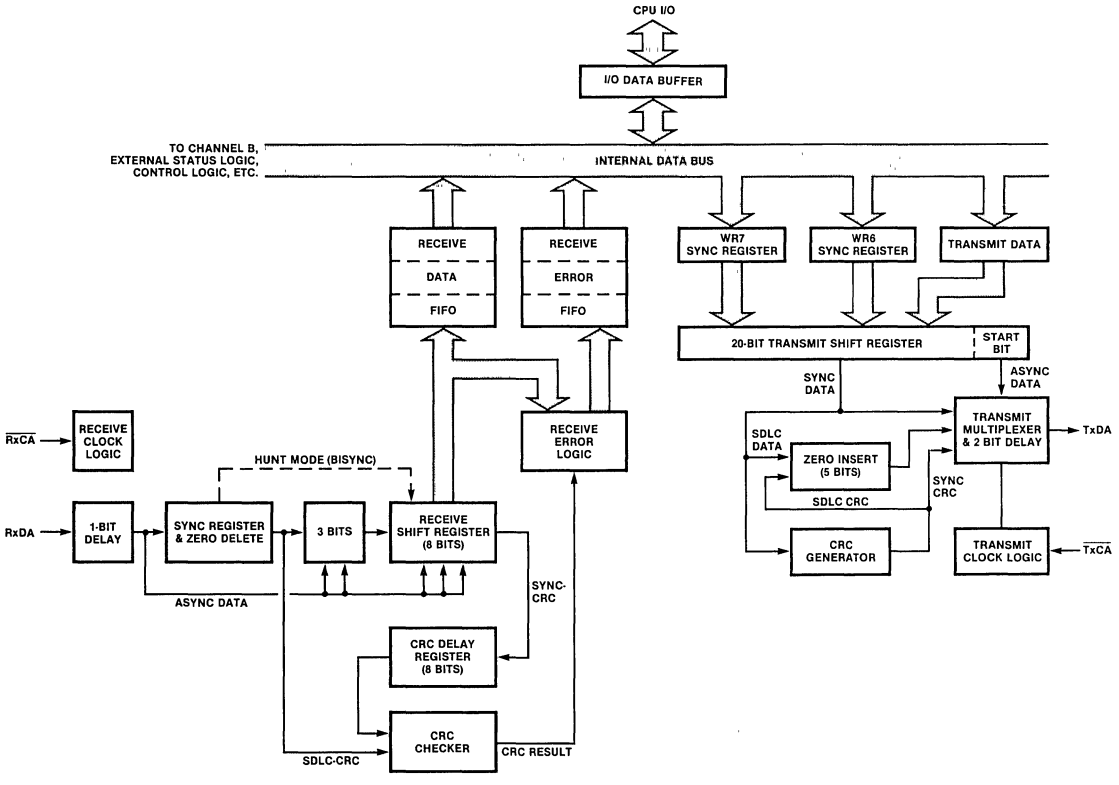


Figure 12. Transmit and Receive Data Path (Channel A)

Programming The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input ($B/\overline{A}$) and the control/data input ($C/\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

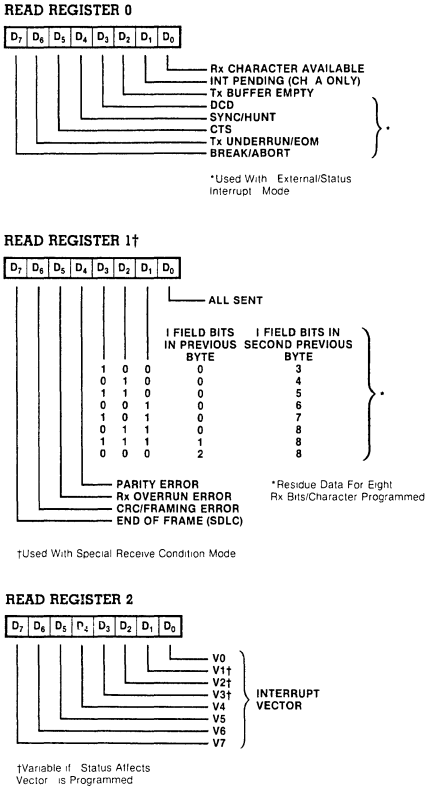


Figure 13. Read Register Bit Functions

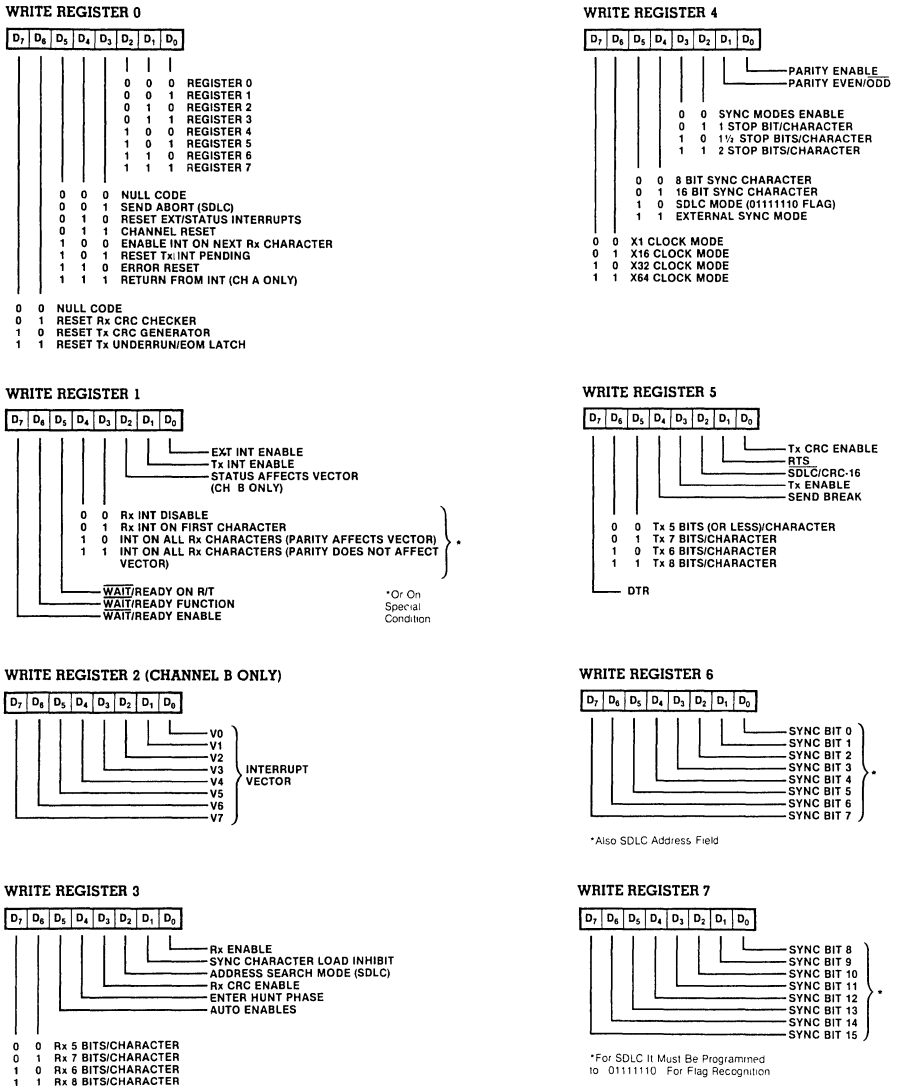


Figure 14. Write Register Bit Functions

Timing

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z-80 CPU sends an interrupt-acknowledge sequence ($\overline{M1}$ Low, and $\overline{IORQ}$ Low a few cycles later) as in Figure 17.

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z-80 CPU issues a RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z-80 CPU Product Specification.*

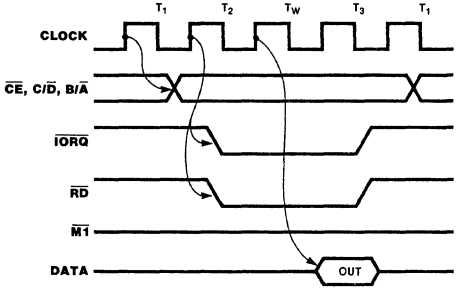


Figure 15. Read Cycle

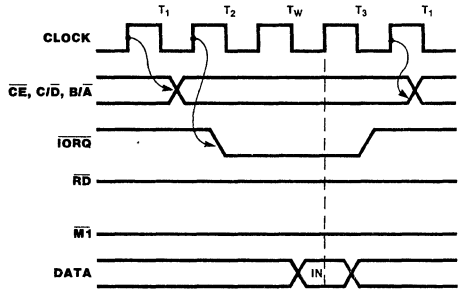


Figure 16. Write Cycle

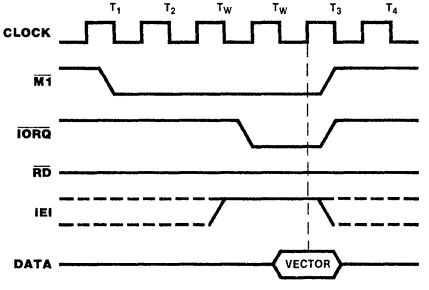


Figure 17. Interrupt Acknowledge Cycle

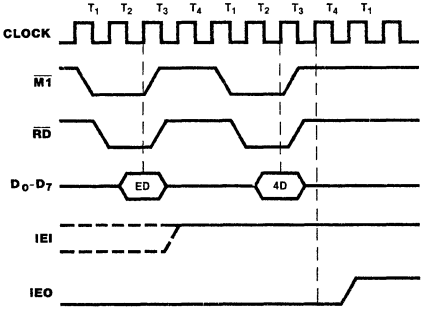


Figure 18. Return from Interrupt Cycle

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

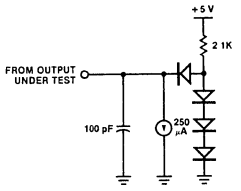
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- -40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- -55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+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} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 μA
I_{LI}	Input Leakage Current	-10	+10	μA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	μA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current	-40	+10	μA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100	mA	

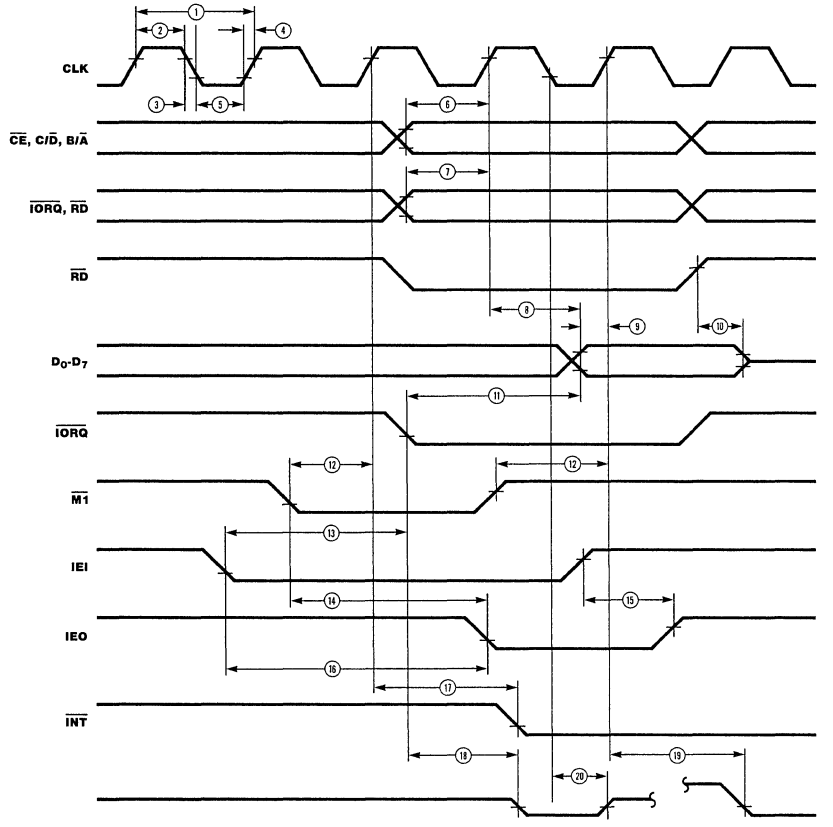
Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured
C_{IN}	Input Capacitance		5	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

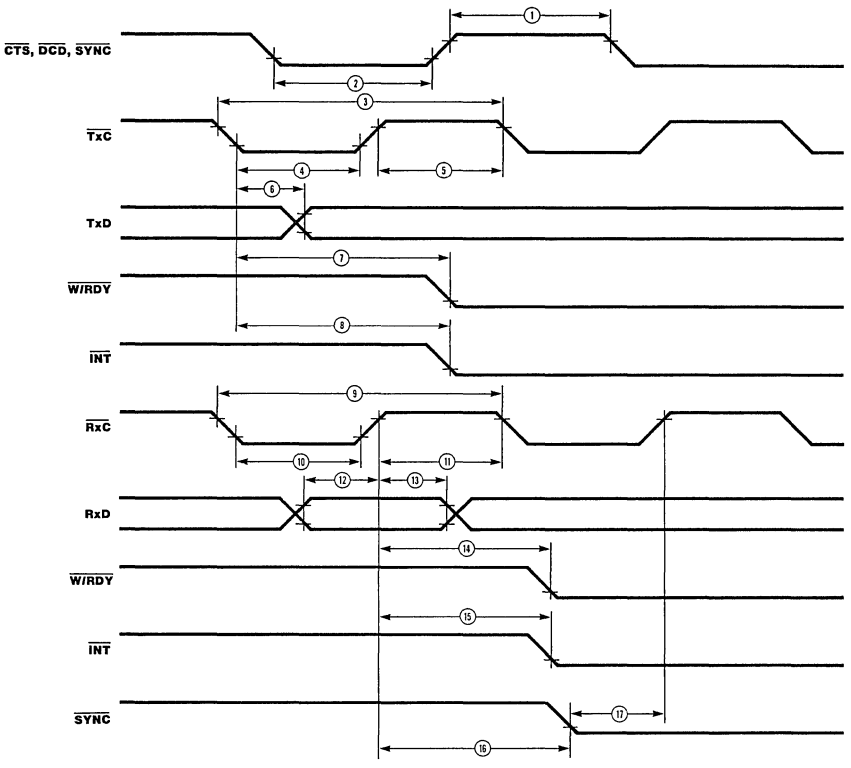
Over specified temperature range, f = 1MHz

AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO Min	Z-80B SIO Max	Unit
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000	ns
3	TfC	Clock Fall Time		30		30		15	ns
4	TrC	Clock Rise Time		30		30		15	ns
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000	ns
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60		ns
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150	ns
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		30		ns
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90	ns
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		100	ns
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120		ns
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190	160		ns
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70	ns
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70	ns
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200		150	ns
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to W/$\overline{RDY}$ ↓ Delay Wait Mode)		300		210		175	ns
19	TdC(W/RR)	Clock ↑ to W/$\overline{RDY}$ ↓ Delay (Ready Mode)		120		120		100	ns
20	TdC(W/RWz)	Clock ↓ to W/$\overline{RDY}$ Float Delay (Wait Mode)		150		130		110	ns
21	Th	Any unspecified Hold when Setup is specified	0		0		0		ns

AC Electrical Characteristics (Continued)



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO Min	Z-80B SIO Max	Unit
1	TwPh	Pulse Width (High)	200		200		200		ns
2	TwPl	Pulse Width (Low)	200		200		200		ns
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞	100	∞	ns
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay (x1 Mode)		400		300		220	ns
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	5	9	Clk Periods*
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	Clk Periods*
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	ns
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		ns
13	ThRxD(RxC)	$\overline{RxC}$ ↑ to RxD Hold Time (x1 Mode)	140		140		100		ns
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	10	13	Clk Periods*
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	10	13	Clk Periods*
16	TdRxC(SYNC)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ ↓ Delay (Output Modes)	4	7	4	7	4	7	Clk Periods*
17	TsSYNC(RxC)	$\overline{SYNC}$ ↓ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	-100		-100			100	ns

In all modes, the System Clock rate must be at least five times the maximum data rate
$\overline{RESET}$ must be active a minimum of one complete Clock Cycle
*System Clock

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8440	CE,CM	2.5 MHz	Z80 SIO/0 (40-pin)
Z8440	CMB,CS	2.5 MHz	Same as above
Z8440	DE,DS	2.5 MHz	Same as above
Z8440	PE,PS	2.5 MHz	Same as above
Z8440A	CE,CM	4.0 MHz	Z80A SIO/0 (40-pin)
Z8440A	CMB,CS	4.0 MHz	Same as above
Z8440A	DE,DS	4.0 MHz	Same as above
Z8440A	PE,PS	4.0 MHz	Same as above
Z8440B	CE,CM	6.0 MHz	Z80B SIO/0 (40-pin)
Z8440B	CMB,CS	6.0 MHz	Same as above
Z8440B	DE,DS	6.0 MHz	Same as above
Z8440B	PE,PS	6.0 MHz	Same as above
Z8441	CE,CM	2.5 MHz	Z80 SIO/1 (40-pin)
Z8441	CMB,CS	2.5 MHz	Same as above
Z8441	DE,DS	2.5 MHz	Same as above
Z8441	PE,PS	2.5 MHz	Same as above
Z8441A	CE,CM	4.0 MHz	Z80A SIO/1 (40-pin)
Z8441A	CMB,CS	4.0 MHz	Same as above
Z8441A	DE,DS	4.0 MHz	Z80A SIO/1 (40-pin)
Z8441A	PE,PS	4.0 MHz	Same as above
Z8441B	CE,CM	6.0 MHz	Z80B SIO/1 (40-pin)
Z8441B	CMB,CS	6.0 MHz	Same as above
Z8441B	DE,DS	6.0 MHz	Same as above
Z8441B	PE,PS	6.0 MHz	Same as above
Z8442	CE,CM	2.5 MHz	Z80 SIO/2 (40-pin)
Z8442	CMB,CS	2.5 MHz	Same as above
Z8442	DE,DS	2.5 MHz	Same as above
Z8442	PE,PS	2.5 MHz	Same as above
Z8442A	CE,CM	4.0 MHz	Z80A SIO/2 (40-pin)
Z8442A	CMB,CS	4.0 MHz	Same as above
Z8442A	DE,DS	4.0 MHz	Same as above
Z8442A	PE,PS	4.0 MHz	Same as above
Z8442B	CE,CM	6.0 MHz	Z80B SIO/2 (40-pin)
Z8442B	CMB,CS	6.0 MHz	Same as above
Z8442B	DE,DS	6.0 MHz	Same as above
Z8442B	PE,PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8449
Z80® SIO/9 Serial Input/Output Controller

Zilog

Product Specification

March 1981

Features

- One full-duplex channel, with separate control and status lines for a modem or other device.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity, overrun and framing error detection.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO/9 Serial Input/Output Controller is a single-channel data communication interface with extraordinary versatility and capability. Functionally this device is identical to the Z-80 SIO, except that it operates in one channel only. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions

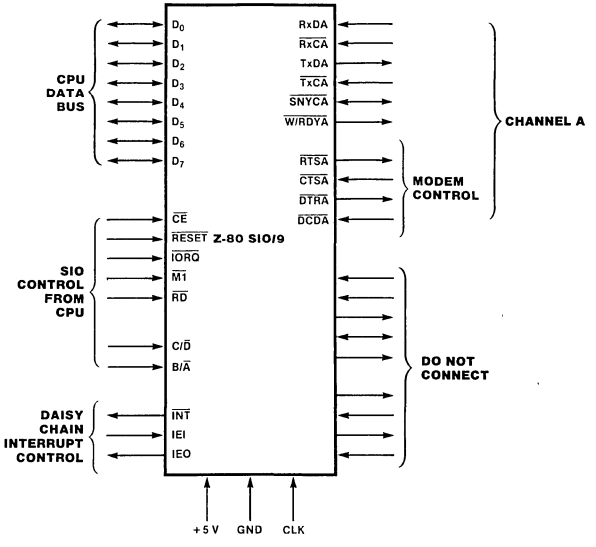


Figure 1. Z-80 SIO/9 Pin Functions

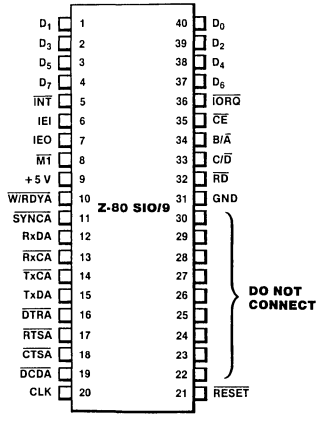


Figure 2. Z-80 SIO/9 Pin Assignments

General Description (Continued)

traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU and DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well-suited to many other CPUs.

The Z-80 SIO/9 is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and standard Z-80 family single-phase clock.

Refer to the *Z-80 SIO Product Specification* and the *Z-80 SIO Technical Manual* for detailed functional and electrical descriptions. All functional and electrical descriptions in these publications are applicable to the Z-80 SIO/9, except that Channel B cannot be used for data input or output and pins 22 through 30 must not be connected.

Write Register 2 (interrupt vector) and the Status Affects Vector bit in Write Register 1 are, however, still programmed by selecting Channel B with the B/$\overline{A}$ input. All other bits in Write Register 1 or Channel B must be programmed to 0.

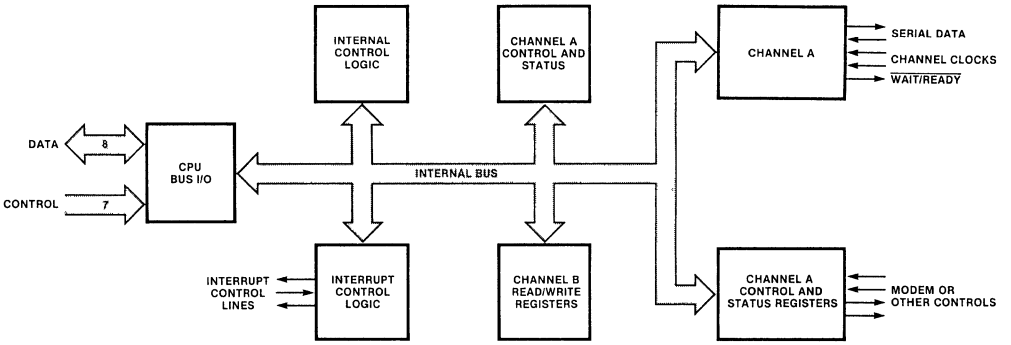


Figure 3. Block Diagram

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8449	CE	2.5 MHz	Z80 SIO/9 (40-pin)
Z8449	CM	2.5 MHz	Same as above
Z8449	CMB	2.5 MHz	Same as above
Z8449	CS	2.5 MHz	Same as above
Z8449	DE	2.5 MHz	Same as above
Z8449	DS	2.5 MHz	Same as above
Z8449	PE	2.5 MHz	Same as above
Z8449	PS	2.5 MHz	Same as above
Z8449A	CE	4.0 MHz	Z80A SIO/9 40-pin)
Z8449A	CM	4.0 MHz	Same as above
Z8449A	CMB	4.0 MHz	Same as above
Z8449A	CS	4.0 MHz	Same as above
Z8449A	DE	4.0 MHz	Z80A SIO/9 (40-pin)
Z8499A	DS	4.0 MHz	Same as above
Z8449A	PE	4.0 MHz	Same as above
Z8449A	PS	4.0 MHz	Same as above
Z8449B	CE	6.0 MHz	Z80B SIO/9 (40-pin)
Z8449B	CM	6.0 MHz	Same as above
Z8449B	CMB	6.0 MHz	Same as above
Z8449B	CS	6.0 MHz	Same as above
Z8449B	DE	6.0 MHz	Same as above
Z8449B	DS	6.0 MHz	Same as above
Z8449B	PE	6.0 MHz	Same as above
Z8449B	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8470 Z80® DART Dual Asynchronous Receiver/Transmitter

Zilog

Product Specification

March 1981

Features

- Two independent full-duplex channels with separate modem controls. Modem status can be monitored.
- Receiver data registers are quadruply buffered; the transmitter is doubly buffered.
- Interrupt features include a programmable interrupt vector, a "status affects vector" mode for fast interrupt processing, and the standard Z-80 peripheral daisy-chain interrupt structure that provides automatic interrupt vectoring with no external logic.
- In x1 clock mode, data rates are 0 to 500K bits/second with a 2.5 MHz clock, or 0 to 800K bits/second with a 4.0 MHz clock.
- Programmable options include 1, 1½ or 2 stop bits; even, odd or no parity; and x1, x16, x32 and x64 clock modes.
- Break generation and detection as well as parity-, overrun- and framing-error detection are available.

Description

The Z-80 DART (Dual-Channel Asynchronous Receiver/Transmitter) is a dual-channel multi-function peripheral component that satisfies a wide variety of asynchronous serial data communications requirements in microcomputer systems. The Z-80 DART is used as a serial-to-parallel, parallel-to-serial converter/controller in asynchronous applications. In addition, the device also provides modem controls for both channels. In applications where modem controls are not needed, these lines can be used for general-purpose I/O.

Zilog also offers the Z-80 SIO, a more versatile device that provides synchronous (Bisync, HDLC and SDLC) as well as asynchronous operation.

The Z-80 DART is fabricated with n-channel silicon-gate depletion-load technology, and is packaged in a 40-pin plastic or ceramic DIP.

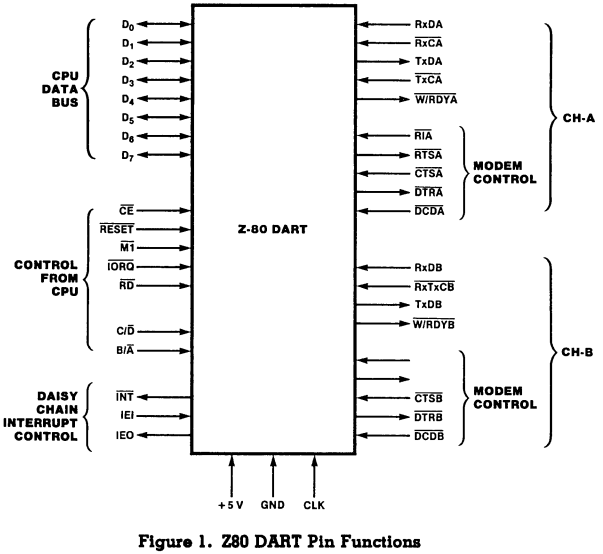


Figure 1. Z80 DART Pin Functions

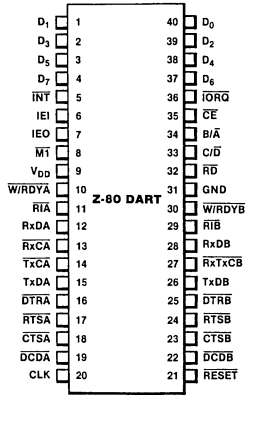


Figure 2. Pin Assignments

Pin Description

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the Z-80 DART.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input specifies the type of information (control or data) transferred on the data bus between the CPU and the Z-80 DART.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low at this input enables the Z-80 DART to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The Z-80 DART uses the standard Z-80 single-phase system clock to synchronize internal signals.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals.

D_0-D_7. *System Data Bus* (bidirectional, 3-state) transfers data and commands between the CPU and the Z-80 DART.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the Z-80 DART is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

IEI. *Interrupt Enable In* (input, active High) 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 Z-80 DART. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the Z-80 DART is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{M1}$. *Machine Cycle One* (input from Z-80 CPU, active Low). When $\overline{M1}$ and $\overline{RD}$ are both active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the Z-80 DART accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the Z-80 DART is the highest priority device that has interrupted the Z-80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z-80 DART. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$.

RxCA, RxCB. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate.

$\overline{RD}$. *Read Cycle Status.* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress.

RxDA, RxDB. *Receive Data* (inputs, active High).

$\overline{RESET}$. *Reset* (input, active Low). Disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts.

$\overline{RIA}$, $\overline{RIB}$. *Ring Indicator* (inputs, Active Low). These inputs are similar to $\overline{CTS}$ and $\overline{DCD}$. The Z-80 DART detects both logic level transitions and interrupts the CPU. When not used in switched-line applications, these inputs can be used as general-purpose inputs.

$\overline{RTSA}$, $\overline{RTSB}$. *Request to Send* (outputs, active Low). When the RTS bit is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset, the output goes High after the transmitter empties.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). TxD changes on the falling edge of $\overline{TxC}$. The Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered. Both the Receiver and Transmitter Clocks may be driven by the Z-80 CTC Counter Time Circuit for programmable baud rate generation.

TxDA, TxDB. *Transmit Data* (outputs, active High).

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the Z-80 DART data rate. The reset state is open drain.

Functional Description

The functional capabilities of the Z-80 DART can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of asynchronous data communications protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other Z-80 peripheral circuits, and shares the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the Z-80 DART offers valuable features such as non-vectored interrupts, polling and simple hand-shake capability.

The first part of the following functional description introduces Z-80 DART data communications capabilities; the second part describes the interaction between the CPU and the Z-80 DART.

A more detailed explanation of Z-80 DART operation can be found in the *Z-80 SIO Technical Manual* (Document Number 03-3033-01). Because this manual was written for the Z-80 SIO, it contains information about synchronous as well as asynchronous operation.

Communications Capabilities. The Z-80 DART provides two independent full-duplex channels for use as an asynchronous receiver/transmitter. The following is a short description of receiver/transmitter capabilities. For more details, refer to the Asynchronous Mode section of the *Z-80 SIO Technical Manual.* The Z-80 DART offers transmission and reception of five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one and a half or two stop bits per character and can provide a break output at any time. The receiver break detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the Receive Data input. If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the character on which they occurred. Vectored interrupts allow fast servicing of interrupting conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The Z-80 DART does not require symmetric Transmit and Receive Clock signals—a feature that allows it to be used with a Z-80 CTC or any other clock source. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the Receive and Transmit Clock inputs. When using Channel B, the bit rates for transmit and receive operations must be the same because $\overline{RxC}$ and $\overline{TxC}$ are bonded together ($\overline{RxTxCB}$).

I/O Interface Capabilities. The Z-80 DART offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

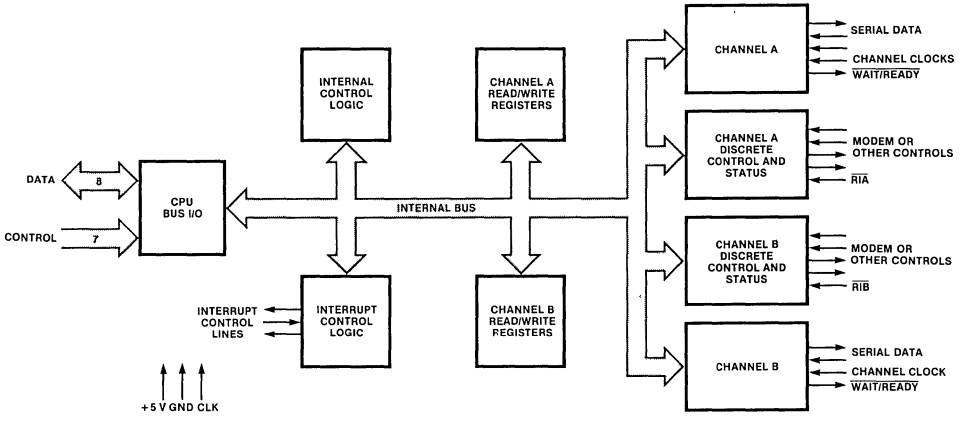


Figure 3. Block Diagram

Functional Description (Continued)

POLLING. There are no interrupts in the Polled mode. Status registers RR0 and RR1 are updated at appropriate times for each function being performed. All the interrupt modes of the Z-80 DART must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a data transfer is needed. The status also indicates Error or other special status conditions (see "Z-80 DART Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 are accompanied by a Receive Character Available status in RR0.

INTERRUPTS. The Z-80 DART offers an elaborate interrupt scheme that provides fast interrupt response in real-time applications. As a member of the Z-80 family, the Z-80 DART can be daisy-chained along with other Z-80 peripherals for peripheral interrupt-priority resolution. In addition, the internal interrupts of the Z-80 DART are nested to prioritize the various interrupts generated by Channels A and B. Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To eliminate the necessity of writing a status analysis routine, the Z-80 DART can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in RR2 is modified according to the assigned priority of the various interrupting conditions.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on the first received character
- Interrupt on all received characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters can optionally modify the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character basis. The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{RI}$ pins; however, an External/Status interrupt is also caused by the detection of a Break sequence in the data stream. The interrupt caused by the Break sequence has a special feature that allows the Z-80 DART to interrupt when the Break sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break condition.

CPU/DMA BLOCK TRANSFER. The Z-80 DART provides a Block Transfer mode to accommodate CPU block transfer functions and DMA block transfers (Z-80 DMA or other designs). The Block Transfer mode uses the $\overline{W/RDY}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{W/RDY}$ output can be defined under software control as a Wait line in the CPU Block Transfer mode or as a Ready line in the DMA Block Transfer mode.

To a DMA controller, the Z-80 DART Ready output indicates that the Z-80 DART is ready to transfer data to or from memory. To the CPU, the Wait output indicates that the Z-80 DART is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

Internal Architecture

The device internal structure includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated as follows:

WR0-WR5 — Write Registers 0 through 5
RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process.

The logic for both channels provides formats, bit synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$) and Ring Indicator ($\overline{RI}$) are monitored by the control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/Status interrupts are prioritized in that order within each channel.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 4 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service a Receive Character Available interrupt in a high-speed data transfer.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus, and a 9-bit transmit shift register that is loaded from the transmit data register.

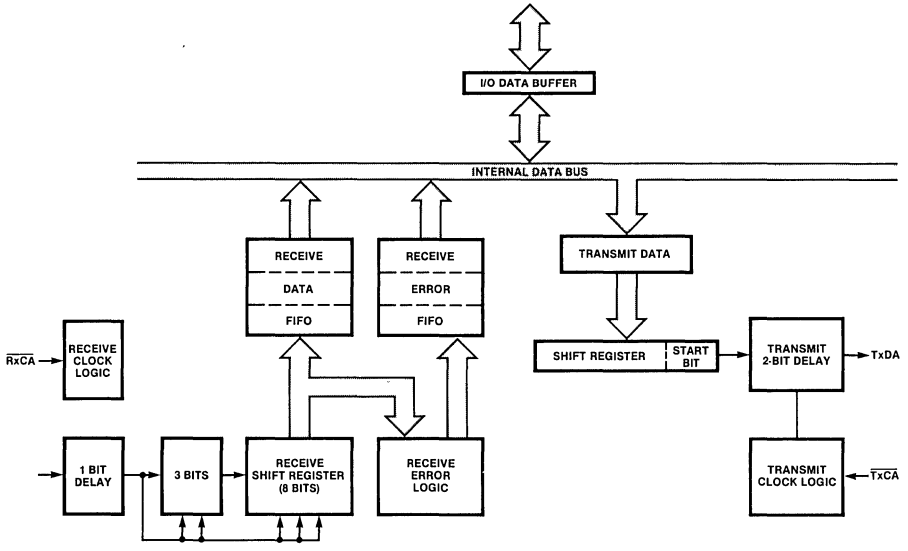


Figure 4. Data Path

Read, Write and Interrupt Timing

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a Data or Status byte from the Z-80 DART are illustrated in Figure 5a.

Write Cycle. Figure 5b illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a Data or Control byte into the Z-80 DART.

Interrupt Acknowledge Cycle. After receiving an Interrupt Request signal ($\overline{INT}$ pulled Low), the Z-80 CPU sends an Interrupt Acknowledge signal ($\overline{M1}$ and $\overline{IORQ}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, IEO = IEI. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Refer to the *Z-80 SIO Technical Manual* for additional details on the interrupt daisy chain and interrupt nesting.

Return From Interrupt Cycle. Normally, the Z-80 CPU issues an RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed.

When used with other CPUs, the Z-80 DART allows the user to return from the interrupt cycle with a special command called "Return From Interrupt" in Write Register 0 of Channel A. This command is interpreted by the Z-80 DART in exactly the same way it would interpret an RETI command on the data bus.

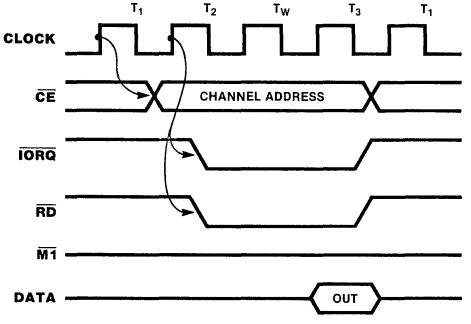


Figure 5a. Read Cycle

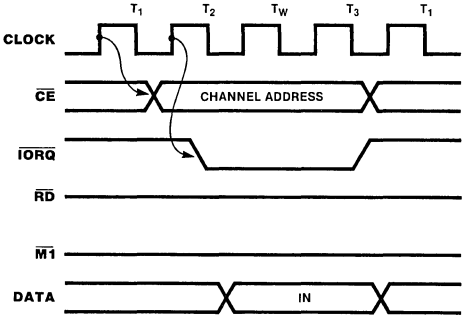


Figure 5b. Write Cycle

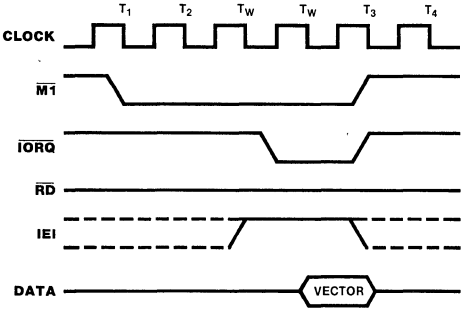


Figure 5c. Interrupt Acknowledge Cycle

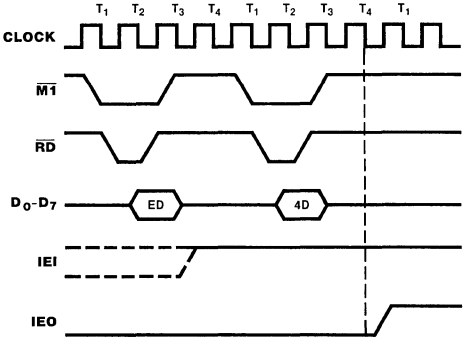


Figure 5d. Return from Interrupt Cycle

Z-80 DART Programming

To program the Z-80 DART, the system program first issues a series of commands that initialize the basic mode and then other commands that qualify conditions within the selected mode. For example, the character length, clock rate, number of stop bits, even or odd parity are first set, then the Interrupt mode and, finally, receiver or transmitter enable.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/$\overline{A}$) and the Control/Data input (C/$\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus.

Write Registers. The Z-80 DART contains six registers (WR0-WR5) in each channel that are programmed separately by the system program to configure the functional personality of the channels (Figure 4). With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z-80 DART.

WR0 is a special case in that all the basic commands (CMD_0-CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This means that a register cannot be pointed to in the same operation as a channel reset.

Write Register Functions	
WR0	Register pointers, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls

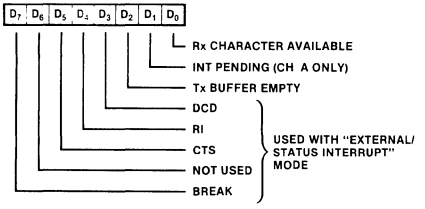
Read Registers. The Z-80 DART contains three registers (RR0-RR2) that can be read to obtain the status information for each channel (except for RR2, which applies to Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

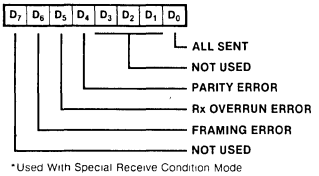
The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

READ REGISTER 0

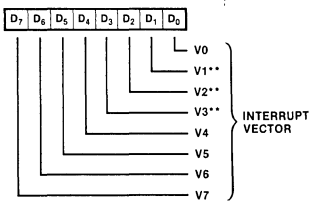


READ REGISTER 1*



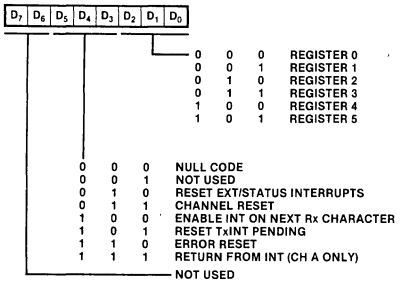
*Used With Special Receive Condition Mode

READ REGISTER 2

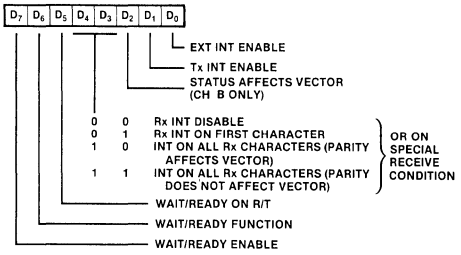


**Variable If Status Affects Vector Is Programmed

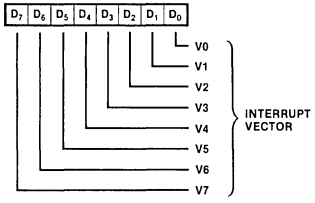
WRITE REGISTER 0



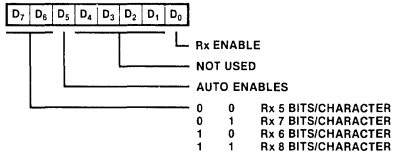
WRITE REGISTER 1



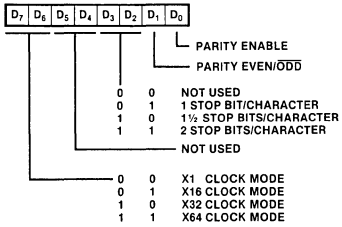
WRITE REGISTER 2 (CHANNEL B ONLY)



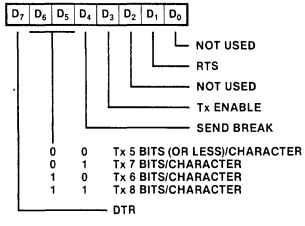
WRITE REGISTER 3



WRITE REGISTER 4



WRITE REGISTER 5



Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65 °C to +150 °C

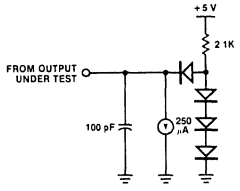
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- −40°C to +85°C, $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- −55°C to +125°C, $+4.5\ V \leq V_{CC} \leq +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section.

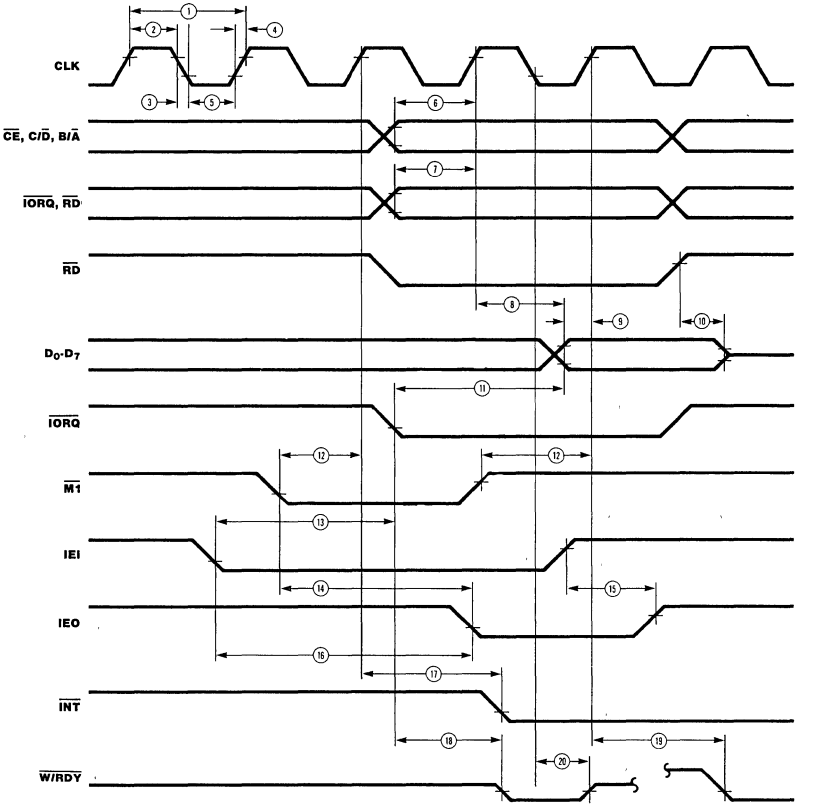


DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+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} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_L	Input/3-State Output Leakage Current	-10	+10	µA	0.4<V<2.4V
$I_{L(RI)}$	$\overline{RI}$ Pin Leakage Current	-40	+10	µA	0.4<V<2.4V
I_{CC}	Power Supply Current		100	mA	

T_A = 0°C to 70°C, V_{CC} = +5V, ±5%

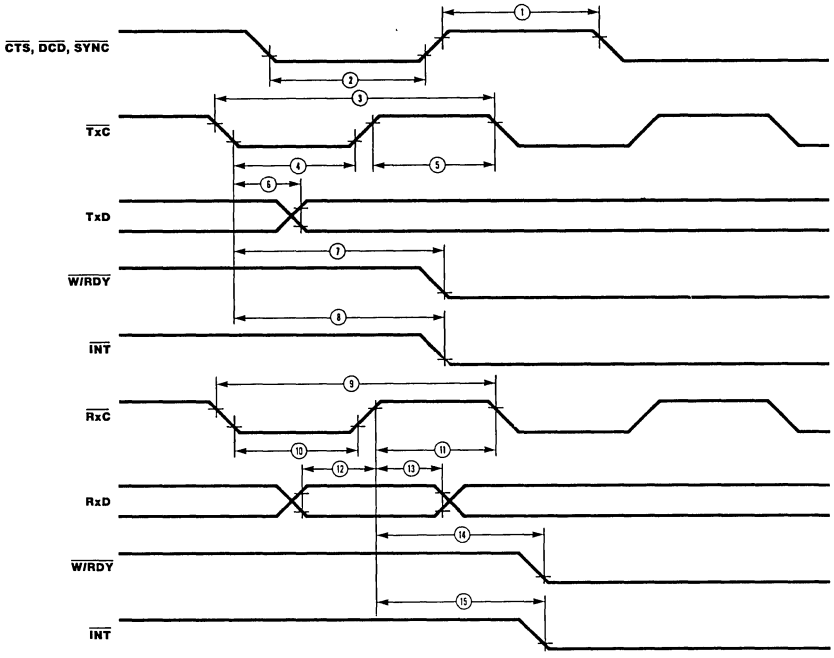
AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 DART Min	Z-80 DART Max	Z-80A DART Min	Z-80A DART Max	Z-80B DART* Min	Z-80B DART* Max	Unit
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000	ns
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000	ns
3	TfC	Clock Fall Time		30		30		15	ns
4	TrC	Clock Rise Time		30		30		15	ns
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000	ns
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60		ns
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60		ns
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150	ns
9	TsDI(C)	Data In to Clock ↑ Setup Time	50		50		30		ns
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90	ns
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160		100	ns
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75		ns
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTA Cycle)	200		140		120		ns
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt immediately preceding M1 ↓)		300		190	160		ns
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70	ns
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70	ns
17	TdC(INT)	Clock ↑ to INT ↓ Delay		200		200		150	ns
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Wait Mode)		300		210		175	ns
19	TdC(W/RR)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		120		120		100	ns
20	TdC(W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		150		130		110	ns

*Z-80B DART timings are preliminary and subject to change

AC Electrical Characteristics (Continued)



Number	Symbol	Parameter	Z-80 DART Min	Z-80 DART Max	Z-80A DART Min	Z-80A DART Max	Z-80B DART[1] Min	Z-80B DART[1] Max	Unit
1	TwPh	Pulse Width (High)	200		200		200		ns
2	TwPl	Pulse Width (Low)	200		200		200		ns
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞		220	ns
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay		400		300			ns
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	5	9	Clk Periods
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	Clk Periods
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	ns
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	ns
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	ns
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		ns
13	ThRxD(RxC)	RxD Hold Time (x1 Mode)	140		140		100		ns
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	10	13	Clk Periods
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	10	13	Clk Periods

NOTES.
1. Z-80B DART timings are preliminary and subject to change
2. In all modes, the Clock rate must be at least five times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete Clock Cycle

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8470	CE	2.5 MHz	Z80 DART (40-pin)
Z8470	CM	2.5 MHz	Same as above
Z8470	CMB	2.5 MHz	Same as above
Z8470	CS	2.5 MHz	Same as above
Z8470	DE	2.5 MHz	Same as above
Z8470	DS	2.5 MHz	Same as above
Z8470	PE	2.5 MHz	Same as above
Z8470	PS	2.5 MHz	Same as above
Z8470A	CE	4.0 MHz	Z80A DART (40-pin)
Z8470A	CM	4.0 MHz	Same as above
Z8470A	CMB	4.0 MHz	Same as above
Z8470A	CS	4.0 MHz	Same as above
Z8470A	DE	4.0 MHz	Same as above
Z8470A	DS	4.0 MHz	Z80A DART (40-pin)
Z8470A	PE	4.0 MHz	Same as above
Z8470A	PS	4.0 MHz	Same as above
Z8470B	CE	6.0 MHz	Z80B DART (40-pin)
Z8470B	CM	6.0 MHz	Same as above
Z8470B	CMB	6.0 MHz	Same as above
Z8470B	CS	6.0 MHz	Same as above
Z8470B	DE	6.0 MHz	Same as above
Z8470B	DS	6.0 MHz	Same as above
Z8470B	PE	6.0 MHz	Same as above
Z8470B	PS	6.0 MHz	Same as above

NOTES C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8000 Family

Zilog

Zilog Z8000™ Family

Zilog

The Art of Staying a Generation Ahead

March 1981

If you know about microprocessors, you know that Zilog became a technology leader by introducing the 8-bit Z80 Family. The soundness of that family design has been amply proven, and its popularity is still growing.

Naturally, you expect Zilog to come up with more than just an extension of an 8-bit architecture to 16 bits. And you are right. The new Z8000 Family of microprocessor components is a whole generation ahead. Now you can design advanced concepts from the mainframe and minicomputer worlds into microcomputer systems that do more at less cost than ever before.

You won't find anything like this anywhere but Zilog. The advanced architecture, high throughput, intelligent peripherals and system flexibility of the Z8000 Family are sure to make it the most popular microprocessor of the 1980s. Want to find out why? Get acquainted with the family. Read on.

System Flexibility. The Z-Family spans the gulf between simple stand-alone computers and complex multiple-processor systems. Your system can grow as your application matures or expands.

Even the smallest Z8000 systems offer high throughput and easy programming far superior to any existing microprocessor alternative. In mid-range applications, Z8000 components offer much more powerful solutions to the design problems of word processing, intelligent terminals, data communications, instrumentation and process control. And in a complex network of multiple processors, smart peripheral components, small local memories and a large common memory, the Z8000 Family provides performance and versatility exceeding that of much larger—and far more expensive—minicomputers.

Higher Throughput, Reduced Cost. The powerful instruction set, high execution speed, regular architecture and numerous special features of the Z8000 CPU dramatically increase system throughput. Intelligent Z8000 peripheral controllers unburden the CPU and boost throughput even more.

Simply put, the Z8000 Family offers more for less money. The Z8000 CPU gives mid-range minicomputer performance at microprocessor cost. At component prices, Z8000 peripheral controllers perform complex system functions that previously required an entire PC board. Memory costs are reduced by Z-BUS memories that require no external logic, and by the compact code and moderate clock rate of the Z8000 CPU.

The Z8000 Family is designed for multiple-processor operation—an economical way of greatly increasing system performance. Many special features for multiple Z8000 CPUs facilitate the design of multiple-processor systems that share access to a common memory. The Z8010 MMU Memory Management Unit can dynamically relocate code and protect memory areas. The Z8034 Z-UPC Universal Peripheral Controller, a complete slave microcomputer, can manipulate data off-line. Asynchronous parts of multiple-processor systems can be joined by the Z8038 Z-FIO FIFO Interface Unit. Z-BUS compatible memories make small local memory for dedicated CPUs an affordable item.

An Unmatched CPU. The Z8000 CPU is far more than a wider data path, more registers, more data types, more addressing modes, more instructions and more addressing space. It brings the big-machine concepts to the level of microprocessors. The instruction set of the Z8000 CPU is more powerful than that of many minicomputers and incorporates features previously found only in large mainframes. Its general-register architecture avoids bottlenecks associated with dedicated or implied registers. Special features support parallel processors, operating systems and compilers. For example, its ability to operate in both the System and Normal modes separates user programs from the operating system for better software security and modularity.

The Z8000 CPU is also a very fast machine. Its throughput is greater than that of any other 16-bit microprocessor and many minicomputers. And the Z8000 CPU achieves this at a moderate 4 MHz clock rate that allows you the choice of slow-access, low-cost memories.

How to Manage Your Memory Better. Modern trends are toward systems that have large and growing memories, multiple users, complex programs and requirements for effective system security. These design problems pose questions not sufficiently answered by other microprocessor families.

Another example of the Z-Family commitment to advanced architectural concepts, the MMU provides flexibility in code segment relocation and sophistication in memory protection found nowhere else in the microprocessor world. This unique device encourages modular software development—a necessary trend as programs reach new levels of complexity.

You are free from specifying where information is actually located in the physical memory because the MMU makes software addresses totally independent from the actual physical memory address. Some existing microprocessor CPUs do have internal CPU relocation registers, but they are dedicated and support few segments, and these CPUs restrict memory protection. Not true for the MMU. Various MMU configurations can randomly relocate all 128 segments output by the Z8000 CPU in all its six addressing spaces and with various translation tables for each space.

But the MMU is far more than a relocation device. It offers you a host of memory protection features that allow the system to protect its software from unwanted uses and users. Segments can be specified as read-only to protect them from being overwritten, as system-only to protect the operating system from inadvertent user access, as execute-only, and so on. A write warning zone is especially useful in stack operations so the operating system can deal with growing stacks.

Peripheral Problem Solvers. Z8000 peripheral components are not dumb I/O circuits. They perform intelligent, complicated tasks on their own. They unburden the CPU, reduce bus traffic and increase system throughput. Complex system tasks that previously required burdensome conglomerations of MSI, can now be handled off-line by Z-BUS peripherals with little CPU overhead.

Multifunction Z-BUS peripherals are extensively programmable, so each can be precisely tailored to its application. All share common interrupt and bus-request structures, as well as an I/O command structure that addresses up to 64 internal registers. Z-BUS peripherals can be operated in priority-interrupt or polled environments. They offer multiple channels to minimize chip count.

Counting, timing and parallel I/O problems seem less tiresome with the **Z8036 Z-CIO Counter and Parallel I/O** device. It has three 16-bit counter/timers, three I/O ports and can even double as a programmable interrupt-priority controller. Data communications are neatly handled by the **Z8030 Z-SCC Serial Communication Controller,** a dual-channel multiprotocol component that supports all popular communications formats. Direct memory access is amply supported by the **Z8016 DTC DMA Transfer Controller,** a fast dual-channel device that enhances the addressing power of the Z8000 CPU in stand-alone or parallel-processor environments. You can interface a variety of CRT displays with the **Z8052 CRT Controller,** which offers a vertical or horizontal split screen, oversize alphanumerics, smooth scrolling, and numerous other features. General-purpose control and data-manipulation problems are smoothly solved by the **Z8034 Z-UPC Universal Peripheral Controller,** a complete off-line microcomputer-on-a-chip with three I/O ports. This processor executes the same friendly, capable instruction set as our Z8 Microcomputer. Bits and pieces of asynchronous parallel-processing systems are interconnected by the **Z8038 Z-FIO FIFO Input/Output,** a surprisingly flexible device that can interface any major microprocessor and most peripherals to the Z-BUS. Its buffer depth can be expanded without limit using the **Z8060 FIFO.** Small local memory is affordable because the **Z6132 4K x 8 Quasi-Static Z-BUS RAM** requires no external interface circuitry, and it refreshes itself.

Z8001/2
Z8000™ CPU Central Processing Unit

Zilog

Product Specification

March 1981

Features

- Regular, easy-to-use architecture
- Instruction set more powerful than many minicomputers
- Directly addresses 8M bytes
- Eight user-selectable addressing modes
- Seven data types that range from bits to 32-bit long words and word strings
- System and Normal operating modes
- Separate code, data and stack spaces
- Sophisticated interrupt structure
- Resource-sharing capabilities for multiprocessing systems
- Multi-programming support
- Compiler support
- Memory management and protection provided by Z8010 Memory Management Unit
- 32-bit operations, including signed multiply and divide
- Z-BUS compatible
- 4 MHz clock rate

General Description

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially, a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; resources abundant in registers, data types, addressing modes and addressing range; and a regular architecture that enhances throughput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set exhibits a high degree of regularity: most instructions can use any of the five main addressing modes and can operate on byte, word and long-word data types.

The CPU can operate in either the system or normal modes. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination of instruction and hardware features; and compilers by multiple stacks, special instructions and addressing modes.

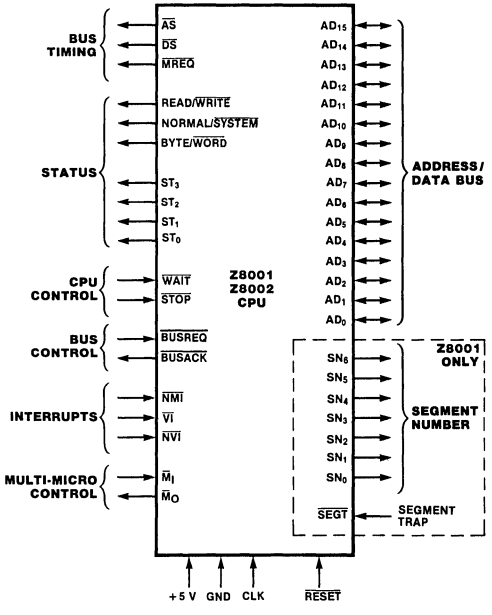


Figure 1. Z8000 CPU Pin Functions

General Description (Continued)

The Z8000 CPU is offered in two versions: the Z8001 48-pin segmented CPU and the Z8002 40-pin non-segmented CPU. The main difference between the two is in addressing range. The Z8001 can directly address 8 megabytes of memory; the Z8002 directly addresses 64 kilobytes. The two operating modes – system and normal – and the distinction between code, data and stack spaces within each mode allows memory extension up to 48 megabytes for the Z8001 and 384 kilobytes for the Z8002.

To meet the requirements of complex, memory-intensive applications, a companion memory-management device is offered for the Z8001. The Z8010 Memory Management Unit manages the large address space by providing features such as segment relocation and memory protection. The Z8001 can be used with or without the Z8010. If used by itself, the Z8001 still provides an 8 megabyte direct addressing range, extendable to 48 megabytes.

The Z8001, Z8002 and Z8010 are fabricated with high-density, high-performance scaled n-channel silicon-gate depletion-load technology, and are housed in dual in-line packages.

Register Organization

The Z8000 CPU is a register-oriented machine that offers sixteen 16-bit general-purpose registers and a set of special system registers. All general-purpose registers can be used as accumulators and all but one as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figures 2 and 3). For byte operations, the first eight 16-bit registers (R0...R7) are treated as sixteen 8-bit registers (RL0, RH0, ... , RL7, RH7). The sixteen 16-bit registers are grouped in pairs (RR0 ... RR14) to form 32-bit long-word registers. Similarly, the register set is grouped in quadruples (RQ0 ... RQ12) to form 64-bit registers.

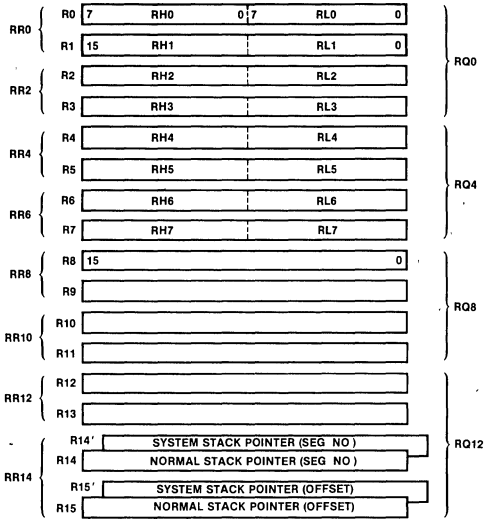


Figure 2. Z8001 General-Purpose Registers

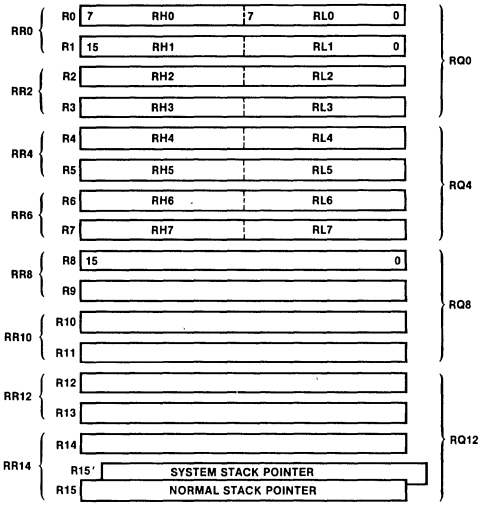


Figure 3. Z8002 General-Purpose Registers

Stacks

The Z8001 and Z8002 can use stacks located anywhere in memory. Call and Return instructions as well as interrupts and traps use implied stacks. The distinction between normal and system stacks separates system information from the application program information. Two stack pointers are available: the system stack pointer and the normal stack pointer. Because they are part of the general-purpose register group, the user can manipulate the stack pointers with any instruction available for register operations.

In the Z8001, register pair RR14 is the implied stack pointer. Register R14 contains the 7-bit segment number and R15 contains the 16-bit offset. In the Z8002, register R15 is the implied 16-bit stack pointer.

Refresh

The Z8000 CPU contains a counter that can be used to automatically refresh dynamic memory. The refresh counter register consists of a 9-bit row counter, a 6-bit rate counter and an enable bit (Figure 4). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler (n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed from 1 to 64 μs with a 4 MHz clock. Refresh can be disabled by programming the refresh enable/disable bit.

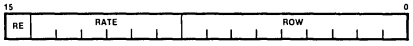


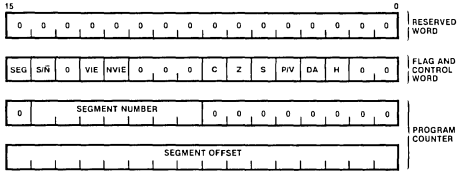
Figure 4. Refresh Counter

Program Status Information

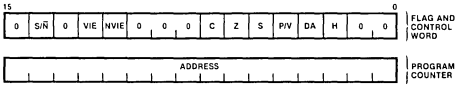
This group of status registers contains the program counter, flags and control bits. When an interrupt or trap occurs, the entire group is saved and a new program status group is loaded.

Figure 5 illustrates how the program status groups of the Z8001 and Z8002 differ. In the non-segmented Z8002, the program status group consists of two words: the program counter (PC), and the flag and control word (FCW). In the segmented Z8001, the program status group consists of four words: a two-word program counter, the flag and control word and an unused word reserved for future use. Seven bits of the first PC word designate one of the 128 memory segments. The second word supplies the 16-bit offset that designates a memory location within the segment.

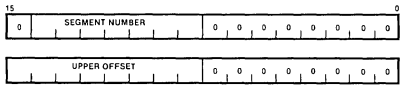
With the exception of the segment enable bit in the Z8001 program status group, the flags and control bits are the same for both CPUs.



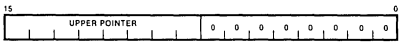
Z8001 Program Status Registers



Z8002 Program Status Registers



Z8001 Program Status Area Pointer



Z8002 Program Status Area Pointer

Figure 5. Z8000 CPU Special Registers

Interrupt and Trap Structure

The Z8000 provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored and non-vectored) and four traps (system call, unimplemented instruction, privileged instructions and segmentation trap). The vectored and non-vectored interrupts are maskable. Of the four traps, the only external one is the segmentation trap, which is generated by the Z8010.

The remaining traps occur when instructions limited to the system mode are used in the normal mode, or as a result of the System Call instruction, or for an unimplemented instruction. The descending order of priority for traps and interrupts is: internal traps, non-maskable interrupt, segmentation trap, vectored interrupt and non-vectored interrupt.

When an interrupt or trap occurs, the current program status is automatically pushed on the system stack. The program status consists of the processor status (PC and FCW) plus a 16-bit identifier. The identifier contains the reason or source of the trap or interrupt. For internal traps, the identifier is the first word of the trapped instruction. For external traps or interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle.

After saving the current program status, the new program status is automatically loaded from the program status area in system memory. This area is designated by the program status area pointer (PSAP).

Data Types

Z8000 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits), byte strings and word strings (up to 64 kilobytes long). Bits can be set, reset and tested; digits are used in BCD arithmetic operations; bytes are used for characters or small integer values; words are used for integer values, instructions and non-segmented addresses; long words are used for long integer values and segmented addresses. All data elements except strings can reside either in registers or memory. Strings are stored in memory only.

The basic data element is the byte. The number of bytes used when manipulating a data element is either implied by the operation or - for strings and multiple register operations - explicitly specified in the instruction.

Segmentation and Memory Management

High-level languages, sophisticated operating systems, large programs and data bases, and decreasing memory prices are all accelerating the trend toward larger memory requirements in microcomputer systems. The Z8001 meets this requirement with an eight megabyte addressing space. This large address space is directly accessed by the CPU using a segmented addressing scheme and can be managed by the Z8010 Memory Management Unit.

Segmented Addressing

A segmented addressing space - compared with linear addressing - is closer to the way a programmer uses memory because each procedure and data space resides in its own segment. The 8 megabytes of Z8001 addressing space is divided into 128 relocatable segments up to 64 kilobytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any location relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately. The segmented Z8001 can run any code written for the non-segmented Z8002 in any one of its 128 segments, provided it is set to the non-segmented mode.

In hardware, segmented addresses are contained in a register pair or long-word memory location. The segment number and offset can be manipulated separately or together by all the available word and long-word operations.

When contained in an instruction, a segmented address has two different representations: long offset and short offset. The long offset occupies two words, whereas the short offset requires only one and combines in one word the 7-bit segment number with an 8-bit offset (range 0-256). The short offset mode allows very dense encoding of addresses and minimizes the need for long addresses required by direct accesssing of this large address space.

Memory Management

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical* addresses. The Memory Management Unit takes the logical addresses and transforms them into the *physical* addresses required for accessing the memory (Figure 6). This address transformation process is called relocation. Segment relocation makes user software addresses independent of the physical memory so the user is freed from specifying where information is actually located in the physical memory.

The relocation process is transparent to user software. A translation table in the Memory Management Unit associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit offset is added to the physical base address to obtain the actual physical address. The system may dynamically reload translation tables as tasks are created, suspended or changed.

In addition to supporting dynamic segment relocation, the Memory Management Unit also provides segment protection and other segment management features. The protection features prevent illegal uses of segments, such as writing into a write-protected zone.

Each Memory Management Unit stores 64 segment entries that consist of the segment base address, its attributes, size and status. Segments are variable in size from 256 bytes to 64 kilobytes in increments of 256 bytes. Pairs of Management Units support the 128 segment numbers available for each of the six CPU address spaces. Within an address space, several Management Units can be used to create multiple translation tables.

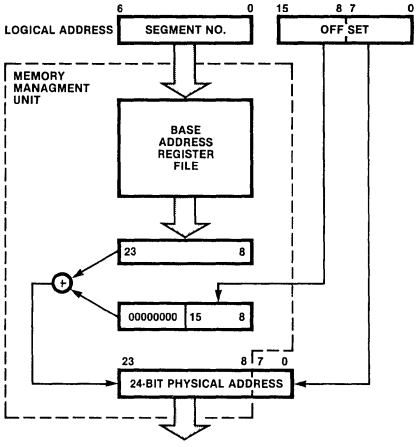


Figure 6. Logical-to-Physical Address Transformation

Extended Processing Architecture

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z8000 CPU. Features of the EPA include:

- Specialized instructions for external processors or software traps may be added to CPU instruction set.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Permits modular design of Z8000-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Simple interconnection between extended processing units and Z8000 CPU requires no additional external supporting logic.
- Supports debugging of suspect hardware against proven software.
- Standard feature on all Zilog Z8000 CPUs.

Specific benefits include:

- EPUs can be added as the system grows and as EPUs with specialized functions are developed.
- Control of EPUs is accomplished via a "single instruction stream" in the Z8000 CPU, eliminating many significant system software and bus contention management obstacles that occur in other multiprocessor (e.g., master-slave) organization schemes.

The processing power of the Zilog Z8000 16-bit microprocessor can be boosted beyond its intrinsic capability by Extended Processing Architecture. Simply stated, EPA allows the Z8000 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream.

The use of extended processors to boost the main CPU's performance capability has been proven with large mainframe computers and minicomputers. In these systems, specialized functions such as array processing, special input/output processing, and data communications processing are typically assigned to extended processor hardware. These extended processors are complex computers in their own right.

The Zilog Extended Processing Architecture combines the best concepts of these proven performance boosters with the latest in high-density MOS integrated-circuit design. The result is an elegant expansion of design capability—a powerful microprocessor architecture capable of connecting single-chip EPUs that permits very effective parallel processing and makes for a smoothly integrated instruction stream from the Z8000 programmer's point of view. A typical addition to the current Z8000 instruction set might be Floating Point Instructions.

The Extended Processing Units connect directly to the Z8000 BUS (Z-BUS) and continuously monitor the CPU instruction stream. When an extended instruction is detected, the appropriate EPU responds, obtaining or placing data or status information on the Z-BUS using the Z8000-generated control signals and performing its function as directed.

The Z8000 CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes instructions intended for it and executes them, using data supplied

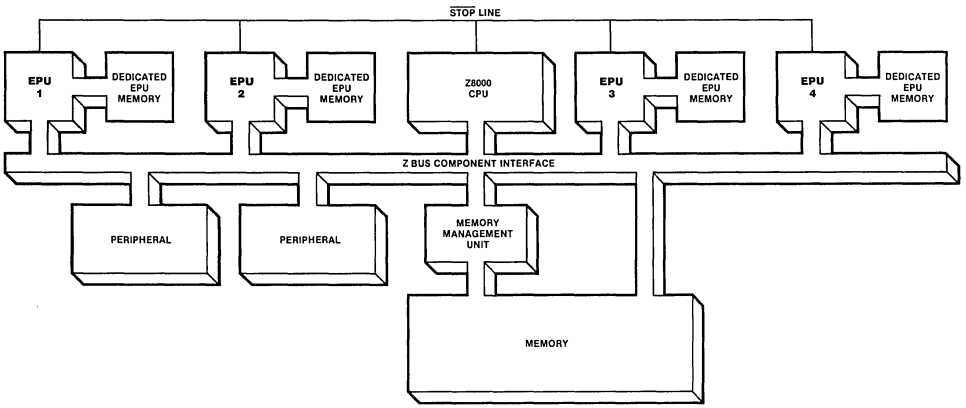


Figure 7. Typical Extended Processor Configuration

Extended Processing Architecture (Continued)

with the instruction and/or data within its internal registers. There are four classes of EPU instructions:

- Data transfers between main memory and EPU registers
- Data transfers between CPU registers and EPU registers
- EPU internal operations
- Status transfers between the EPUs and the Z8000 CPU Flag and Control Word register (FCW)

Four Z8000 addressing modes may be utilized with transfers between EPU registers and the CPU and main memory:

- Register
- Indirect Register
- Direct Address
- Indexed

In addition to the hardware-implemented capabilities of the Extended Processing Architecture, there is an extended instruction trap mechanism to permit software simulation of EPU functions. A control bit in the Z8000 FCW register indicates whether actual EPUs are present or not. If not, when an extended instruction is detected, the Z8000 traps on the instruction, so that a software "trap handler" can emulate the desired EPU function—a very useful development tool. The EPA software trap routine supports the debugging of suspect hardware against proven software. This feature will increase in significance as designers become familiar with the EPA capability of the Z8000 CPU.

This software trap mechanism facilitates the design of systems for later addition of EPUs: initially, the extended function is executed as a trap subroutine; when the EPU is finally attached, the trap subroutine is eliminated and the EPA control bit is set. Application software is unaware of the change.

Extended Processing Architecture also offers protection against extended instruction overlapping. Each EPU connects to the Z8000 CPU via the STOP line so that if an EPU is requested to perform a second extended instruction function before it has completed the previous one, it can put the CPU into the Stop/Refresh state until execution of the previous extended instruction is complete.

EPA and CPU instruction execution are shown in Figure 8. The CPU begins operation by fetching an instruction and determining whether it is a CPU or an EPU command. The EPU meanwhile monitors the Z-BUS for its own instructions. If the CPU encounters an EPU command, it checks to see whether an EPU is present; if not, the EPU may be simulated by an EPU instruction trap software routine; if an EPU is present, the necessary data and/or address is placed on the Z-BUS. If the EPU is free when the instruction and data for it appear, the extended instruction is executed. If the EPU is still processing a previous instruction, it activates the CPU's $\overline{\text{STOP}}$ line to lock the CPU off at the Z-BUS until execution is complete. After the instruction is finished, the EPU deactivates the $\overline{\text{STOP}}$ line and CPU transactions continue.

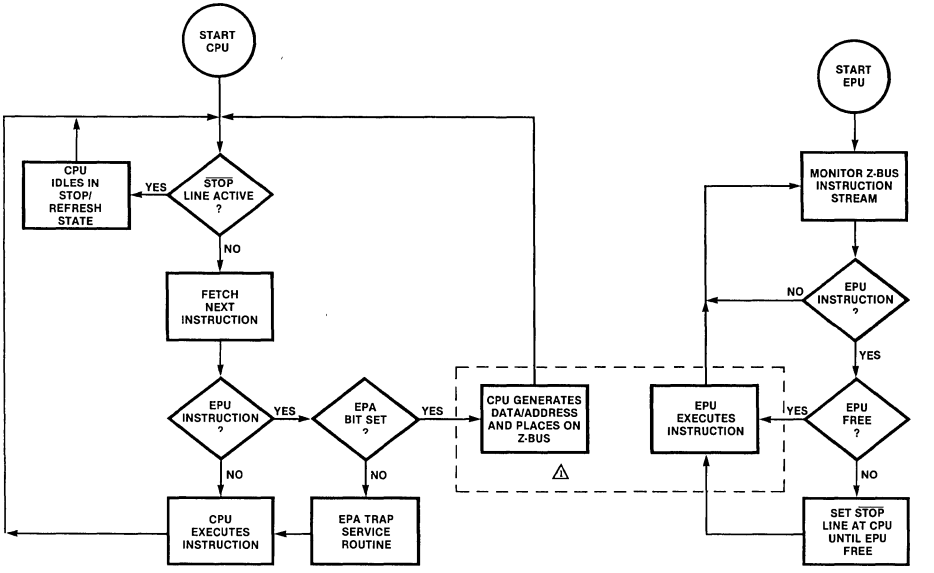


Figure 8. EPA and Z8000 CPU Instruction Execution

Addressing Modes

The information included in Z8000 instructions consists of the function to be performed, the type and size of data elements to be manipulated and the location of the data elements. Locations are designated by register addresses, memory addresses or I/O addresses. The addressing mode of a given instruction defines the address space it references and the method used to compute the address itself. Addressing modes are explicitly specified or implied by the instruction.

Figure 4 illustrates the eight addressing modes: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Indexed (X), Relative Address (RA), Base Address (BA) and Base Indexed (BX). In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address space are usually implied by the instruction.

Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory	
Register		REGISTER ADDRESS → OPERAND		The content of the register
Immediate	OPERAND			In the instruction
Indirect Register		REGISTER ADDRESS → ADDRESS	→ OPERAND	The content of the location whose address is in the register
Direct Address	ADDRESS		→ OPERAND	The content of the location whose address is in the instruction
Index		REGISTER ADDRESS, BASE ADDRESS; DISPLACEMENT; (+)	→ OPERAND	The content of the location whose address is the address in the instruction, offset by the content of the working register
Relative Address	DISPLACEMENT	PC VALUE; (+)	→ OPERAND	The content of the location whose address is the content of the program counter, offset by the displacement in the instruction
Base Address		REGISTER ADDRESS, DISPLACEMENT; BASE ADDRESS; (+)	→ OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the instruction
Base Index		REGISTER ADDRESS, REGISTER ADDRESS; BASE ADDRESS, DISPLACEMENT; (+)	→ OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the register

Figure 9. Addressing Modes

Input/ Output

A set of I/O instructions performs 8-bit or 16-bit transfers betwen the CPU and I/O devices. I/O devices are addressed with a 16-bit I/O port address. The I/O port address is similar to a memory address; however, I/O address space is not part of the memory address space. I/O port and memory addresses coexist on the same bus lines and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. Standard I/O instructions include a comprehensive set of In, Out and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status information distinguishes between standard and special I/O references.

Multi-Micro-Processor Support

Multi-microprocessor systems are supported in hardware and software. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors. The Multi-Micro Out pin issues a request for the resource, while the Multi-Micro In pin is used to recognize the state of the resource. Thus, any CPU in a multiple microprocessor system can exclude all other asynchronous CPUs from a critical shared resource.

Multi-microprocessor systems are supported in software by the instructions Multi-Micro Request, Test Multi-Micro In, Set Multi-Micro Out and Reset Multi-Micro Out. In addition, the eight megabyte CPU address space is beneficial in multiple microprocessor systems that have large memory requirements.

Instruction Set Summary

The Z8000 provides the following types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

Load and Exchange

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CLR	dst	R	7	-	-				**Clear**
CLRB		IR	8	-	-				dst ← 0
		DA	11	12	14				
		X	12	12	15				
EX	R, src	R	6	-	-				**Exchange**
EXB		IR	12	-	-				R ↔ src
		DA	15	16	18				
		X	16	16	19				
LD	R, src	R	3	-	-	5	-	-	**Load into Register**
LDB		IM	7	-	-	11	-	-	R ← src
LDL		IM	5 (byte only)						
		IR	7	-	-	11	-	-	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	-	-	17	-	-	
		BX	14	-	-	17	-	-	
LD	dst, R	IR	8	-	-	11	-	-	**Load into Memory** (Store)
LDB		DA	11	12	14	14	15	17	dst ← R
LDL		X	12	12	15	15	15	18	
		BA	14	-	-	17	-	-	
		BX	14	-	-	17	-	-	
LD	dst, IM	IR	11	-	-				**Load Immediate into Memory**
LDB		DA	14	15	17				dst ← IM
		X	15	15	18				

*** NS = Non-Segmented SS = Segmented Short Offset SL = Segmented Long Offset**

	Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Load and Exchange (Continued)	**LDA**	R, src	DA	12	13	15				**Load Address**
			X	13	13	16				R ← source address
			BA	15	-	-				
			BX	15	-	-				
	LDAR	R, src	RA	15	-	-				**Load Address Relative**
										R ← source address
	LDK	R, src	IM	5	-	-				**Load Constant**
										R ← n (n = 0 ... 15)
	LDM	R, src, n	IR	11	-	-	+3n			**Load Multiple**
			DA	14	15	17				R ← src (n consecutive words)
			X	15	15	18				(n = 1 ... 16)
	LDM	dst, R, n	IR	11	-	-	+3n			**Load Multiple** (Store Multiple)
			DA	14	15	17				dst ← R (n consecutive words)
			X	15	15	18				(n = 1 ... 16)
	LDR **LDRB** **LDRL**	R, src	RA	14	-	-	17	-	-	**Load Relative** R ← src (range -32768 ... +32767)
	LDR **LDRB** **LDRL**	dst, R	RA	14	-	-	17	-	-	**Load Relative** (Store Relative) dst ← R (range -32768 ... +32767)
	POP **POPL**	dst, IR	R	8	-	-	12	-	-	**Pop** dst ← IR Autoincrement contents of R
			IR	12	-	-	19	-	-	
			DA	16	16	18	23	23	25	
			X	16	16	19	23	23	26	
	PUSH **PUSHL**	IR, src	R	9	-	-	12	-	-	**Push** Autodecrement contents of R IR ← src
			IM	12	-	-	-	-	-	
			IR	13	-	-	20	-	-	
			DA	14	14	16	21	21	23	
			X	14	14	17	21	21	24	
Arithmetic	**ADC** **ADCB**	R, src	R	5	-	-				**Add with Carry** R ← R + src + carry
	ADD **ADDB** **ADDL**	R, src	R	4	-	-	8	-	-	**Add** R ← R + src
			IM	7	-	-	14	-	-	
			IR	7	-	-	14	-	-	
			DA	9	10	12	15	16	18	
			X	10	10	13	16	16	19	
	CP **CPB** **CPL**	R, src	R	4	-	-	8	-	-	**Compare with Register** R - src
			IM	7	-	-	14	-	-	
			IR	7	-	-	14	-	-	
			DA	9	10	12	15	16	18	
			X	10	10	13	16	16	19	
	CP **CPB**	dst, IM	IR	11	-	-				**Compare with Immediate** dst - IM
			DA	14	15	17				
			X	15	15	18				
	DAB	dst	R	5	-	-				**Decimal Adjust**
	DEC **DECB**	dst, n	R	4	-	-				**Decrement by n** dst ← dst - n (n = 1 ... 16)
			IR	11	-	-				
			DA	13	14	16				
			X	14	14	17				

Arithmetic (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
DIV **DIVL**	R, src	R	107	-	-	744	-	-	**Divide** (signed) Word: $R_{n+1} \leftarrow R_{n,n+1} \div src$ $R_n \leftarrow$ remainder Long Word: $R_{n+2,n+3} \leftarrow R_{n\ldots n+3} \div src$ $R_{n,n+1} \leftarrow$ remainder
		IM	107	-	-	744	-	-	
		IR	107	107	107	744	744	744	
		DA	108	109	111	745	746	748	
		X	109	109	112	746	746	749	
EXTS **EXTSB** **EXTSL**	dst	R	11	-	-	11	-	-	**Extend Sign** Extend sign of low order half of dst through high order half of dst
INC **INCB**	dst, n	R	4	-	-				**Increment by n** dst ← dst + n (n = 1 ... 16)
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
MULT **MULTL**	R, src	R	70	-	-	282*	-	-	**Multiply** (signed) Word: $R_{n,n+1} \leftarrow R_{n+1} \cdot src$ Long Word: $R_{n\ldots n+3} \leftarrow R_{n+2,n+3}$ *Plus seven cycles for each 1 in the multiplicand
		IM	70	-	-	282*	-	-	
		IR	70	-	-	282*	-	-	
		DA	71	72	74	283*	284*	286*	
		X	72	72	75	284*	284*	287*	
NEG **NEGB**	dst	R	7	-	-				**Negate** dst ← 0 - dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
SBC **SBCB**	R, src	R	5	-	-				**Subtract with Carry** R ← R - src - carry
SUB **SUBB** **SUBL**	R, src	R	4	-	-	8	-	-	**Subtract** R ← R - src
		IM	7	-	-	14	-	-	
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

Logical

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
AND **ANDB**	R, src	R	4	-	-				**AND** R ← R AND src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
COM **COMB**	dst	R	7	-	-				**Complement** dst ← NOT dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
OR **ORB**	R, src	R	4	-	-				**OR** R ← R OR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
TCC **TCCB**	cc, dst	R	5	-	-				**Test Condition Code** Set LSB if cc is true
TEST **TESTB** **TESTL**	dst	R	7	-	-	13	-	-	**Test** dst OR 0
		IR	8	-	-	13	-	-	
		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
XOR **XORB**	R, src	R	4	-	-				**Exclusive OR** R ← R XOR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				

	Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Program Control	**CALL**	dst	IR	10	-	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
			DA	12	18	20				
			X	13	18	21				
	CALR	dst	RA	10	-	15				**Call Relative** Autodecrement SP @ SP ← PC PC ← PC + dst (range -4094 to +4096)
	DJNZ **DBJNZ**	R, dst	RA	11	-	-				**Decrement and Jump if Non-Zero** R ← R - 1 If R ≠ 0: PC ← PC + dst (range -254 to 0)
	IRET*	-	-	13	-	16				**Interrupt Return** PS ← @ SP Autoincrement SP
	JP	cc, dst	IR	10	-	15	(taken)			**Jump Conditional** If cc is true: PC ← dst
			IR	7	-	7	(not taken)			
			DA	7	8	10				
			X	8	8	11				
	JR	cc, dst	RA	6	-	-				**Jump Conditional Relative** If cc is true: PC ← PC + dst (range -256 to +254)
	RET	cc	-	10	-	13	(taken)			**Return Conditional** If cc is true: PC ← @ SP Autoincrement SP
				7	-	7	(not taken)			
	SC	src	IM	33	-	39				**System Call** Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS
Bit Manipulation	**BIT** **BITB**	dst, b	R	4	-	-				**Test Bit Static** Z flag ← NOT dst bit specified by b
			IR	8	-	-				
			DA	10	11	13				
			X	11	11	14				
	BIT **BITB**	dst, R	R	10	-	-				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R
	RES **RESB**	dst, b	R	4	-	-				**Reset Bit Static** Reset dst bit specified by b
			IR	11	-	-				
			DA	13	14	16				
			X	14	14	17				
	RES **RESB**	dst, R	R	10	-	-				**Reset Bit Dynamic** Reset dst bit specified by contents R
	SET **SETB**	dst, b	R	4	-	-				**Set Bit Static** Set dst bit specified by b
			IR	11	-	-				
			DA	13	14	16				
			X	14	14	17				
	SET **SETB**	dst, R	R	10	-	-				**Set Bit Dynamic** Set dst bit specified by contents of R
	TSET **TSETB**	dst	R	7	-	-				**Test and Set** S flag ← MSB of dst dst ← all 1s
			IR	11	-	-				
			DA	14	15	17				
			X	15	15	18				

*Privileged instruction. Executed in system mode only.

	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Rotate and Shift	**RL** **RLB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left** by n bits (n = 1, 2)
	RLC **RLCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left through Carry** by n bits (n = 1, 2)
	RLDB	R, src	R	9	-	-				**Rotate Digit Left**
	RR **RRB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right** by n bits (n = 1, 2)
	RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right through Carry** by n bits (n = 1, 2)
	RRDB	R, src	R	9	-	-				**Rotate Digit Right**
	SDA **SDAB** **SDAL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Arithmetic** Shift dst left or right by contents of R
	SDL **SDLB** **SDLL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Logical** Shift dst left or right by contents of R
	SLA **SLAB** **SLAL**	dst, n	R	(13 + 3 n)			(13 +3 n)			**Shift Left Arithmetic** by n bits
	SLL **SLLB** **SLLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Logical** by n bits
	SRA **SRAB** **SRAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Arithmetic** by n bits
	SRL **SRLB** **SRLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Logical** by n bits
Block Transfer and String Manipulation	**CPD** **CPDB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Decrement** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$
	CPDR **CPDRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Decrement and Repeat** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
	CPI **CPIB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Increment** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$
	CPIR **CPIRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Increment and Repeat** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
	CPSD **CPSDB**	dst, src, R, cc	IR	25	-	-				**Compare String and Decrement** dst - src Autodecrement dst and src addresses R ← R - 1

Block Transfer and String Manipulation (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPSDR **CPSDRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Decr. and Repeat** dst - src Autodecrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
CPSI **CPSIB**	dst, src, R, cc	IR	25	-	-				**Compare String and Increment** dst - src Autoincrement dst and src addresses R ← R - 1
CPSIR **CPSIRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Incr. and Repeat** dst - src Autoincrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst, src, R	IR	20	-	-				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R - 1
LDDR **LDDRB**	dst, src, R	IR	(11 + 9 n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R - 1 Repeat until R = 0
LDI **LDIB**	dst, src, R	IR	20	-	-				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R - 1
LDIR **LDIRB**	dst, src, R	IR	(11 + 9 n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R - 1 Repeat until R = 0
TRDB	dst, src, R	IR	25	-	-				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R - 1
TRDRB	dst, src, R	IR	(11 + 14 n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R - 1 Repeat until R = 0
TRIB	dst, src, R	IR	25	-	-				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R - 1
TRIRB	dst, src, R	IR	(11 + 14 n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R - 1 Repeat until R = 0
TRTDB	src 1, src 2, R	IR	25	-	-				**Translate and Test, Decrement** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1

Block Transfer and String Manipulation (Continued)	Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
	TRTDRB	src 1, src 2, R	IR	(11 + 14 n)						**Translate and Test, Decr. and Repeat** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
	TRTIB	src 1, src 2, R	IR	25						**Translate and Test, Increment** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1
	TRTIRB	src 1, src 2, R	IR	(11 + 14 n)						**Translate and Test, Incr. and Repeat** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
Input/ Output	**IN***	R, src	IR	10	-	-				**Input** R ← src
	INB*		DA	12	-	-				
	IND* **INDB***	dst, src, R	IR	21	-	-				**Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
	INDR* **INDRB***	dst, src, R	IR	(11 + 10 n)						**Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	INI* **INIB***	dst, src, R	IR	21	-	-				**Input and Increment** dst ← src Autoincrement dst address R ← R - 1
	INIR* **INIRB***	dst, src, R	IR	(11 + 10 n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	OUT*	dst, R	IR	10	-	-				**Output** dst ← R
	OUTB*		DA	12	-	-				
	OUTD* **OUTDB***	dst, src, R	IR	21	-	-				**Output and Decrement** dst ← src Autodecrement src address R ← R - 1
	OTDR* **OTDRB***	dst, src, R	IR	(11 + 10 n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
	OUTI* **OUTIB***	dst, src, R	IR	21	-	-				**Output and Increment** dst ← src Autoincrement src address R ← R - 1
	OTIR* **OTIRB***	dst, src, R	IR	(11 + 10 n)						**Output, Increment and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0

*Privileged instructions. Executed in system mode only.

	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Input/Output (Continued)	**SIN*** **SINB***	R, src	DA	12	-	-				**Special Input** R ← src
	SIND* **SINDB***	dst, src, R	IR	21	-	-				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
	SINDR* **SINDRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	SINI* **SINIB***	dst, src, R	IR	21	-	-				**Special Input and Increment** dst ← src Autoincrement dst address R ← R - 1
	SINIR* **SINIRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	SOUT* **SOUTB***	dst, src	DA	12	-	-				**Special Output** dst ← src
	SOUTD* **SOUTDB***	dst, src, R	IR	21	-	-				**Special Output and Decrement** dst ← src Autodecrement src address R ← R - 1
	SOTDR* **SOTDRB***	dst, src, R	IR	(11 + 10 n)						**Special Output, Decr. and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
	SOUTI* **SOUTIB***	dst, src, R	IR	21	-	-				**Special Output and Increment** dst ← src Autoincrement src address R ← R - 1
	SOTIR* **SOTIRB***	dst, src, R	R	(11 + 10 n)						**Special Output, Incr. and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
CPU Control	**COMFLG**	flags	-	7	-	-				**Complement Flag** (Any combination of C, Z, S, P/V)
	DI*	int	-	7	-	-				**Disable Interrupt** (Any combination of NVI, VI)
	EI*	int	-	7	-	-				**Enable Interrupt** (Any combination of NVI, VI)
	HALT*	-	-	(8 + 3 n)						**HALT**
	LDCTL*	CTLR, src	R	7	-	-				**Load into Control Register** CTLR ← src
	LDCTL*	dst, CTLR	R	7	-	-				**Load from Control Register** dst ← CTLR

*Privileged instructions Executed in system mode only

CPU Control (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDCTLB	FLGR, src	R	7	-	-				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dst, FLGR	R	7	-	-				**Load from Flag Byte Register** dst ← FLGR
LDPS*	src	IR	12	-	16				**Load Program Status** PS ← src
		DA	16	20	22				
		X	17	20	23				
MBIT*	-	-	7	-	-				**Test Multi-Micro Bit** Set S if $\overline{M_I}$ is Low, reset S if $\overline{M_I}$ is High.
MREQ*	dst	R	(12 + 7 n)						**Multi-Micro Request**
MRES*	-	-	5	-	-				**Multi-Micro Reset**
MSET*	-	-	5	-	-				**Multi-Micro Set**
NOP	-	-	7	-	-				**No Operation**
RESFLG	flag	-	7	-	-				**Reset Flag** (Any combination of C, Z, S, P/V)
SETFLG	flag	-	7	-	-				**Set Flag** (Any combination of C, Z, S, P/V)

*Privileged instructions Executed in system mode only.

Condition Codes

Code	Meaning	Flag Settings	CC Field
	Always false	-	0000
	Always true	-	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No Carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	P/V = 1	0100
NOV	No overflow	P/V = 0	1100
PE	Parity is even	P/V = 1	0100
PO	Parity is odd	P/V = 0	1100
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
LT	Less than (signed)	(S XOR P/V) = 1	0001
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

Status Line Codes

ST_3-ST_0	Definition	ST_3-ST_0	Definition
0 0 0 0	Internal operation	1 0 0 0	Data memory request
0 0 0 1	Memory refresh	1 0 0 1	Stack memory request
0 0 1 0	I/O reference	1 0 1 0	Data memory request (EPU)
0 0 1 1	Special I/O reference (e.g., to an MMU)	1 0 1 1	Stack memory request (EPU)
0 1 0 0	Segment trap acknowledge	1 1 0 0	Program reference, nth word
0 1 0 1	Non-maskable interrupt acknowledge	1 1 0 1	Instruction fetch, first word
0 1 1 0	Non-vectored interrupt acknowledge	1 1 1 0	Extension processor transfer
0 1 1 1	Vectored interrupt acknowledge	1 1 1 1	Reserved

Pin Description

AD_0-AD_{15}. *Address/Data* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for I/O and to address memory.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates addresses are valid.

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

$\overline{BUSREQ}$. *Bus Request* (input, active Low). This line must be driven Low to request the bus from the CPU.

$\overline{DS}$. *Data Strobe* (output, active Low, 3-state). This line times the data in and out of the CPU.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). A Low on this line indicates that the address/data bus holds a memory address.

$\overline{M}_I$, $\overline{M}_O$. *Multi-Micro In, Multi-Micro Out* (input and output, active Low). These two lines form a resource-request daisy chain that allows one CPU in a multi-microprocessor system to access a shared resource.

$\overline{NMI}$. *Non-Maskable Interrupt* (edge triggered, input, active Low). A high-to-low transition on $\overline{NMI}$ requests a non-maskable interrupt. The NMI interrupt has the highest priority of the three types of interrupts.

$\overline{NVI}$. *Non-Vectored Interrupt* (input, active Low). A Low on this line requests a non-vectored interrupt.

CLK. *System Clock* (input). CLK is a 5V single-phase time-base input.

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

R/$\overline{W}$. *Read/Write* (output, Low = Write, 3-state). R/$\overline{W}$ indicates that the CPU is reading from or writing to memory or I/O.

SN_0-SN_6. *Segment Number* (outputs, active High, 3-state). These lines provide the 7-bit segment number used to address one of 128 segments by the Z8010 Memory Management Unit. Output by the Z8001 only.

$\overline{SEGT}$. *Segment Trap* (input, active Low). The Memory Management Unit interrupts the CPU with a Low on this line when the MMU detects a segmentation trap.

ST_0-ST_3. *Status* (outputs, active High, 3-state). These lines specify the CPU status (see table).

$\overline{STOP}$. *Stop* (input, active Low). This input can be used to single-step instruction execution.

$\overline{VI}$. *Vectored Interrupt* (input, active Low). A Low on this line requests a vectored interrupt.

$\overline{WAIT}$. *Wait* (input, active Low). This line indicates to the CPU that the memory or I/O device is not ready for data transfer.

B/$\overline{W}$. *Byte/Word* (output, Low = Word, 3-state). This signal defines the type of memory reference on the 16-bit address/data bus.

N/$\overline{S}$. *Normal/System Mode* (output, Low = System Mode, 3-state). N/$\overline{S}$ indicates the CPU is in the normal or system mode.

Reserved. Do not connect.

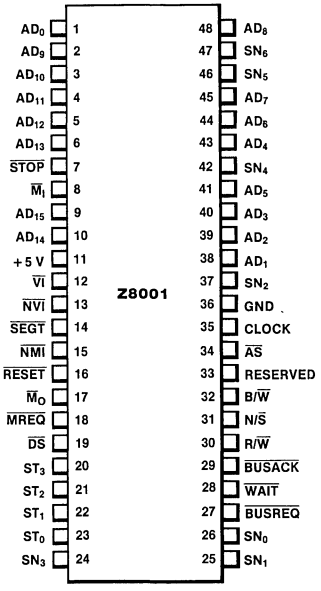


Figure 10. Z8001 Pin Assignments

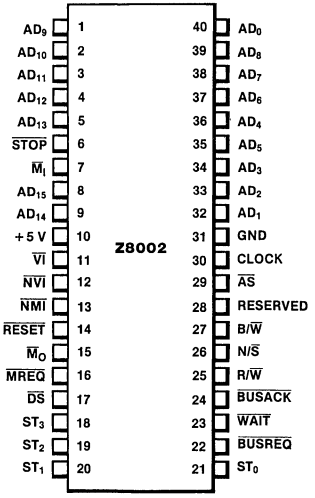


Figure 11. Z8002 Pin Assignments

Z8000 CPU Timing

The Z8000 CPU executes instructions by stepping through sequences of basic machine cycles, such as memory read or write, I/O device read or write, interrupt acknowledge, and internal execution. Each of these basic cycles requires three to ten clock cycles to execute. Instructions that require more clock cycles to execute are broken up into several machine cycles. Thus no machine cycle is longer than ten clock cycles and fast response to a Bus Request is guaranteed.

The instruction opcode is fetched by a normal memory read operation. A memory refresh cycle can be inserted just after the completion of any first instruction fetch (IF_1) cycle and can also be inserted while the following instructions are being executed: MULT, MULTL, DIV, DIVL, HALT, all Shift instructions, all Block Move instructions, and the Multi-Micro Request instruction (MREQ).

The following timing diagrams show the relative timing relationships of all CPU signals during each of the basic operations. When a machine cycle requires additional clock cycles for CPU internal operation, one to five clock cycles are added. Memory and I/O read and write, as well as interrupt acknowledge cycles, can be extended by activating the $\overline{WAIT}$ input. For exact timing information, refer to the composite timing diagram.

Note that the $\overline{WAIT}$ input is not synchronized in the Z8000 and that the setup and hold times for $\overline{WAIT}$ relative to the clock must be met. If asynchronous $\overline{WAIT}$ signals are generated, they must be synchronized with the CPU clock before entering the Z8000.

Memory Read and Write

Memory read and instruction fetch cycles are identical, except for the status information on the ST_0–ST_3 outputs. During a memory read cycle, a 16-bit address is placed on the AD_0–AD_{15} outputs early in the first clock period, as shown in Figure 12. (In the Z8001,

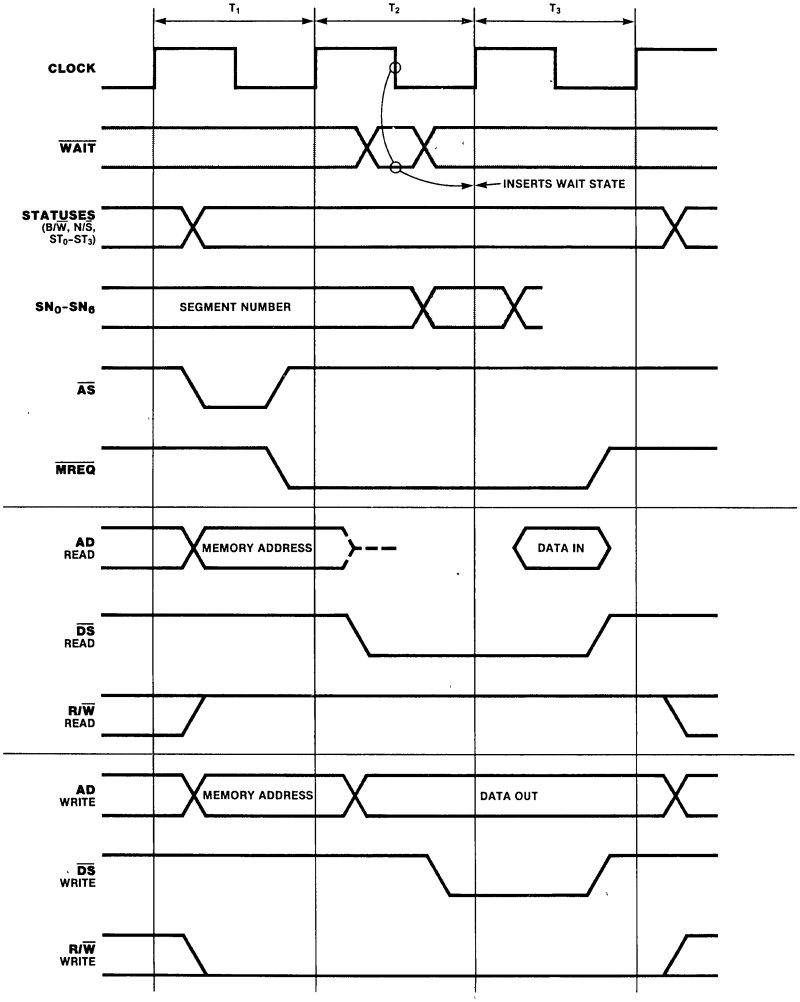


Figure 12. Memory Read and Write Timing

Memory Read and Write (Continued)

the 7-bit segment number is output on SN_0–SN_6 one clock period earlier than the 16-bit address offset to compensate for the delay in the memory management circuitry.)

A valid address is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The state of the $\overline{WAIT}$ input is sampled in the middle of the second clock cycle by the falling edge of Clock. If $\overline{WAIT}$ is Low, an additional clock period is added between T_2 and T_3. $\overline{WAIT}$ is sampled again in the middle of this wait cycle, and additional wait states can be inserted. This allows interfacing slow memories. No control outputs change during wait states.

Although Z8000 memory is word organized, memory is addressed as bytes. All instructions are word-aligned, using even addresses. Within a 16-bit word, the most significant byte (D_8–D_{15}) is addressed by the low-order address (A_0 = Low), and the least significant byte (D_0–D_7) is addressed by the high-order address (A_0 = High).

Input/ Output

I/O timing is similar to memory read/write timing, except that one wait state is automatically inserted between T_2 and T_3 (Figure 13). Both the segmented Z8001 and the non-segmented Z8002 use 16-bit I/O addresses.

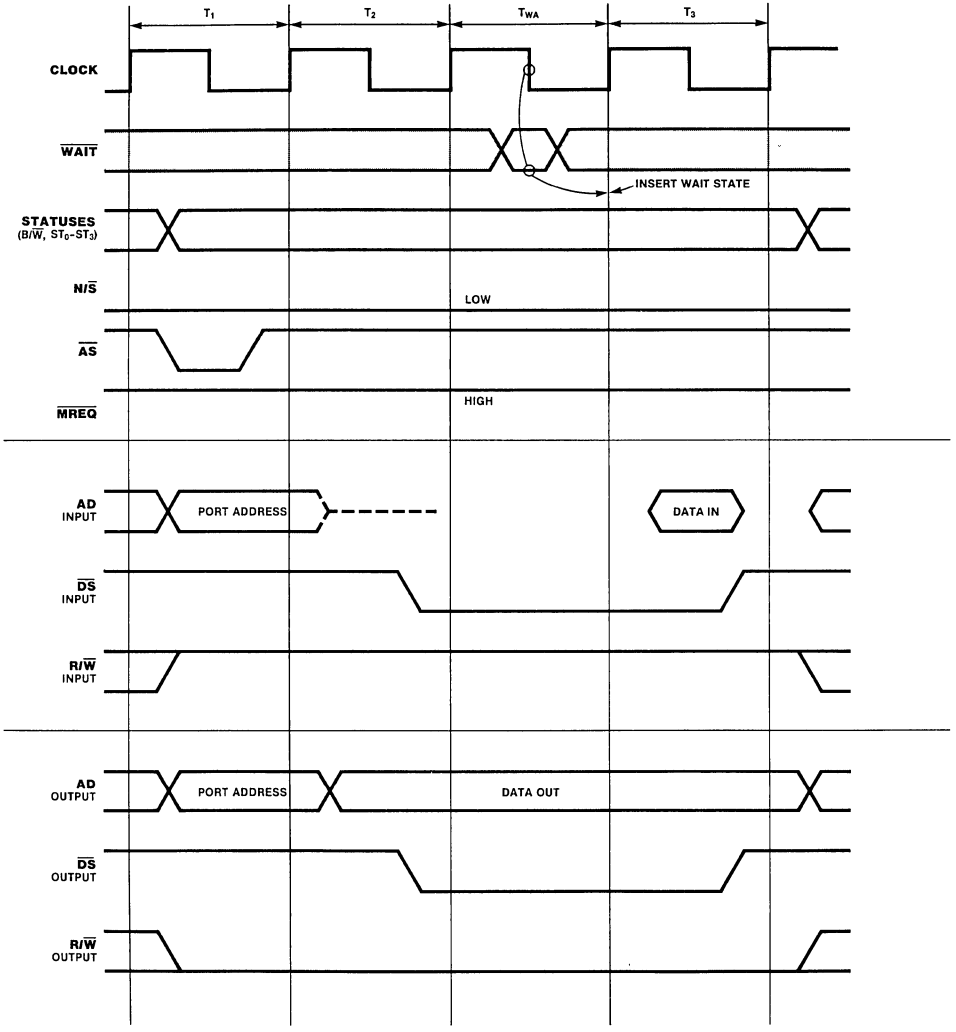


Figure 13. Input/Output Timing

Interrupt and Segment Trap Request and Acknowledge

The Z8000 CPU recognizes three interrupt inputs (non-maskable, vectored and non-vectored) and a segmentation trap input. Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge detected and sets the internal NMI latch. The $\overline{VI}$, $\overline{NVI}$ and $\overline{SEGT}$ inputs as well as the state of the internal NMI latch are sampled at the beginning of T_3 in the last machine cycle of any instruction.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised, but aborted. The program counter is not updated, but the system stack pointer is decremented.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, with additional wait states possible, as shown in Figure 14.

After the last wait state, the CPU reads the information on AD_0–AD_{15} and stores it temporarily, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For the non-vectored and non-maskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be extra user status. For the segmentation trap, the *high* byte is the Memory Management Unit identifier and the *low* byte is undefined.

After the acknowledge cycle, the N/S output indicates the automatic change to system mode.

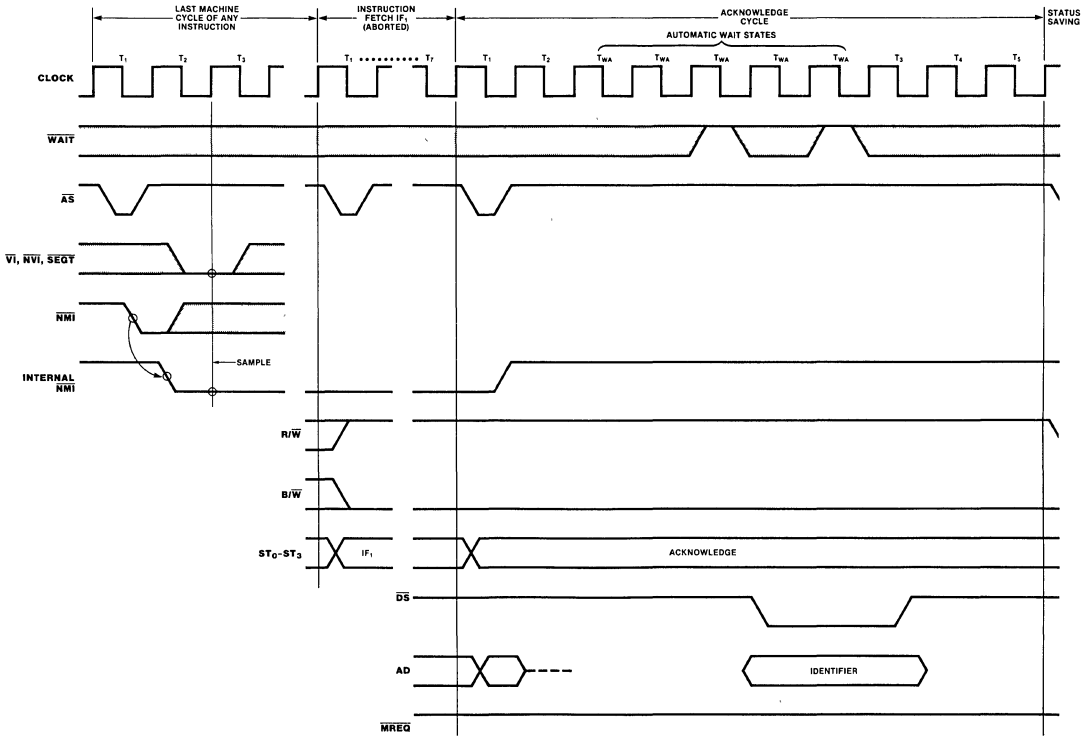


Figure 14. Interrupt and Segment Trap Request/Acknowledge Timing

Status Saving Sequence

The machine cycles following the interrupt acknowledge or segmentation trap acknowledge cycle push the old status information on the system stack in the following order: the 16-bit program counter; the 7-bit segment number (Z8001 only); the flag and control word; and finally the interrupt/trap identifier. Subsequent machine cycles fetch the new program status from the program status area, and then branch to the interrupt/trap service routine.

Bus Request Acknowledge Timing

A Low on the $\overline{BUSREQ}$ input indicates to the CPU that another device is requesting the Address/Data and Control buses. The asynchronous $\overline{BUSREQ}$ input is synchronized at the beginning of any machine cycle (Figure 15). If $\overline{BUSREQ}$ is Low, an internal synchronous $\overline{BUSREQ}$ signal is generated, which–after completion of the current machine cycle–causes the $\overline{BUSACK}$ output to go Low and all bus outputs to go into the high-impedance state. The

Bus Request/ Acknowledge (Continued)

requesting device-typically a DMA-can then control the bus.

When $\overline{BUSREQ}$ is released, it is synchronized with the rising clock edge and the $\overline{BUSACK}$ output goes High one clock period later, indicating that the CPU will again take control of the bus.

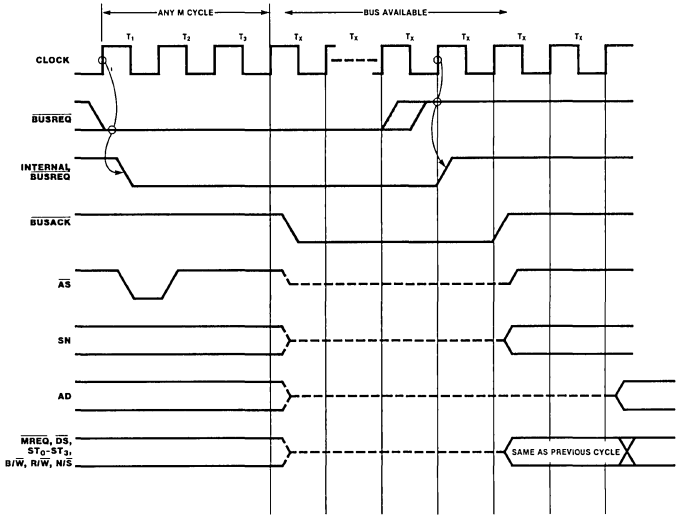


Figure 15. Bus Request/Acknowledge Timing

Stop

The $\overline{STOP}$ input is sampled by the last falling clock edge immediately preceeding any IF_1 cycle (Figure 16). If $\overline{STOP}$ is found Low, a stream of memory refresh cycles is inserted after T_3, again sampling the $\overline{STOP}$ input on each falling clock edge in the middle of the T_3 states. This refresh operation does not use the refresh prescaler or its divide-by-four clock prescaler; rather, it double-increments the refresh counter every three clock cycles. When $\overline{STOP}$ is found High again, the next refresh cycle is completed, any remaining T states of the IF_1 cycle are then executed and the CPU continues its operation.

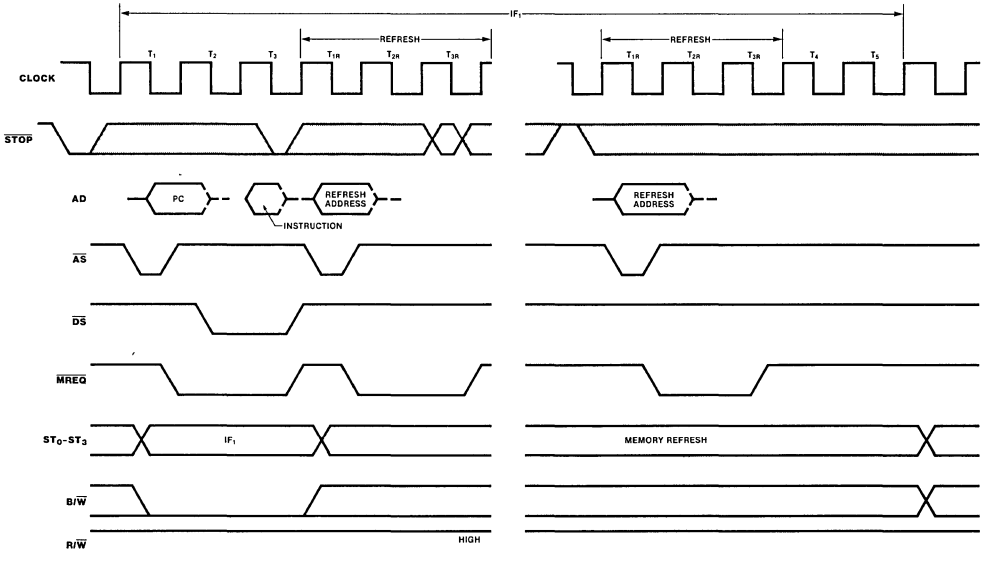


Figure 16. Stop Timing

Internal Operation

Certain extended instructions, such as Multiply and Divide, and some special instructions need additional time for the execution of internal operations. In these cases, the CPU goes through a sequence of internal operation machine cycles, each of which is three to eight clock cycles long (Figure 17). This allows fast response to Bus Request and Refresh Request, because bus request or refresh cycles can be inserted at the end of any internal machine cycle.

Although the address outputs during T_1 are undefined, Address Strobe is generated to satisfy the requirements of future Z-BUS compatible self-refresh dynamic memories.

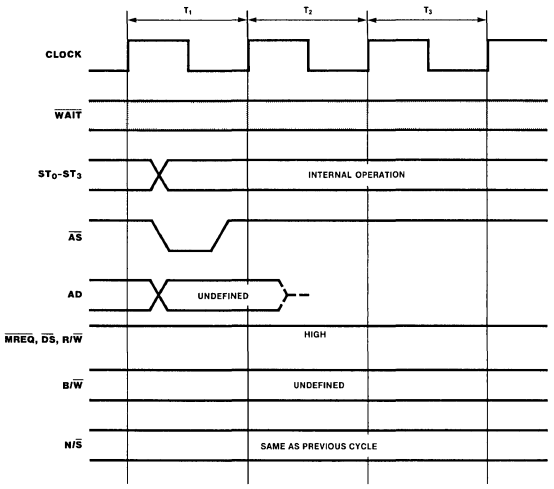


Figure 17. Internal Operation Timing

Memory Refresh

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle consisting of three T-states is started as soon as possible (that is, after the next IF_1 cycle or Internal Operation cycle).

The 9-bit refresh counter value is put on the low-order side of the address bus (AD_0–AD_8); AD_9–AD_{15} are undefined (Figure 18). Since the memory is word-organized, A_0 is always Low during refresh and the refresh counter is always incremented by two, thus stepping through 256 consecutive refresh addresses on AD_1–AD_8. Unless disabled, the presettable prescaler runs continuously and the delay in starting a refresh cycle is therefore not cumulative.

While the $\overline{STOP}$ input is Low, a continuous stream of memory refresh cycles, each three T-states long, is executed without using the refresh prescaler.

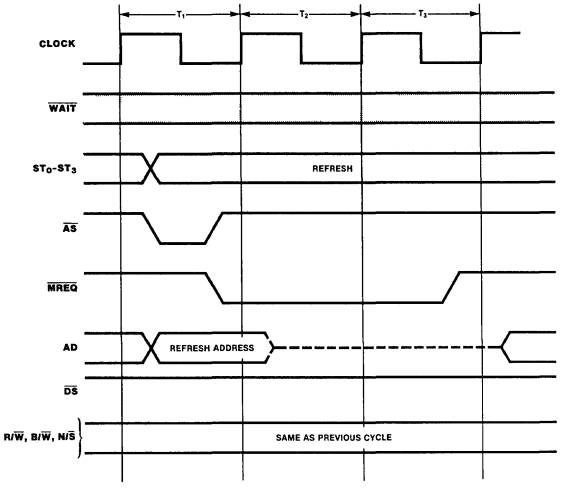


Figure 18. Memory Refresh Timing

Halt

A HALT instruction executes an unlimited number of 3-cycle internal operations, inter-spersed with memory refresh cycles whenever requested. An interrupt, segmentation trap or reset are the only exits from a HALT instruction.

The CPU samples the $\overline{VI}$, $\overline{NVI}$, $\overline{NMI}$ and $\overline{SEGT}$ inputs at the beginning of every T_3 cycle. If an input is found active during two consecutive samples, the subsequent IF_1 cycle is exercised, but aborted, and the normal interrupt acknowledge cycle is started.

Reset

A Low on the $\overline{RESET}$ input causes the following results within five clock cycles (Figure 19):

- AD_0–AD_{15} are 3-stated
- $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, $\overline{BUSACK}$ and $\overline{M}_O$ are forced High
- ST_0–ST_3 and SN_0–SN_6 are forced Low
- Refresh is disabled
- R/$\overline{W}$, B/$\overline{W}$ and N/$\overline{S}$ are not affected

When $\overline{RESET}$ has been High for three clock periods, two consecutive memory read cycles are executed in the system mode. In the Z8001, the first cycle reads the flag and control word from location 0002, the next reads the 7-bit program counter segment number from location 0004, the next reads the 16-bit PC offset from location 0006, and the following IF_1 cycle starts the program. In the Z8002, the first cycle reads the flag and control word from location 0002, the next reads the PC from location 0004 and the following IF_1 cycle starts the program.

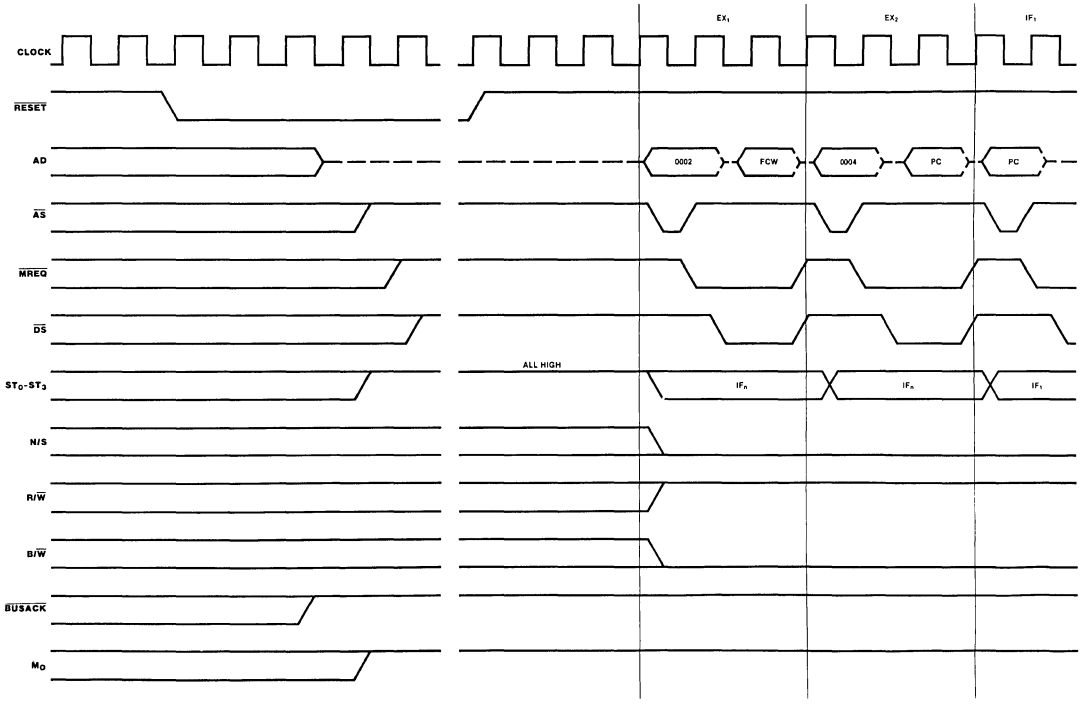
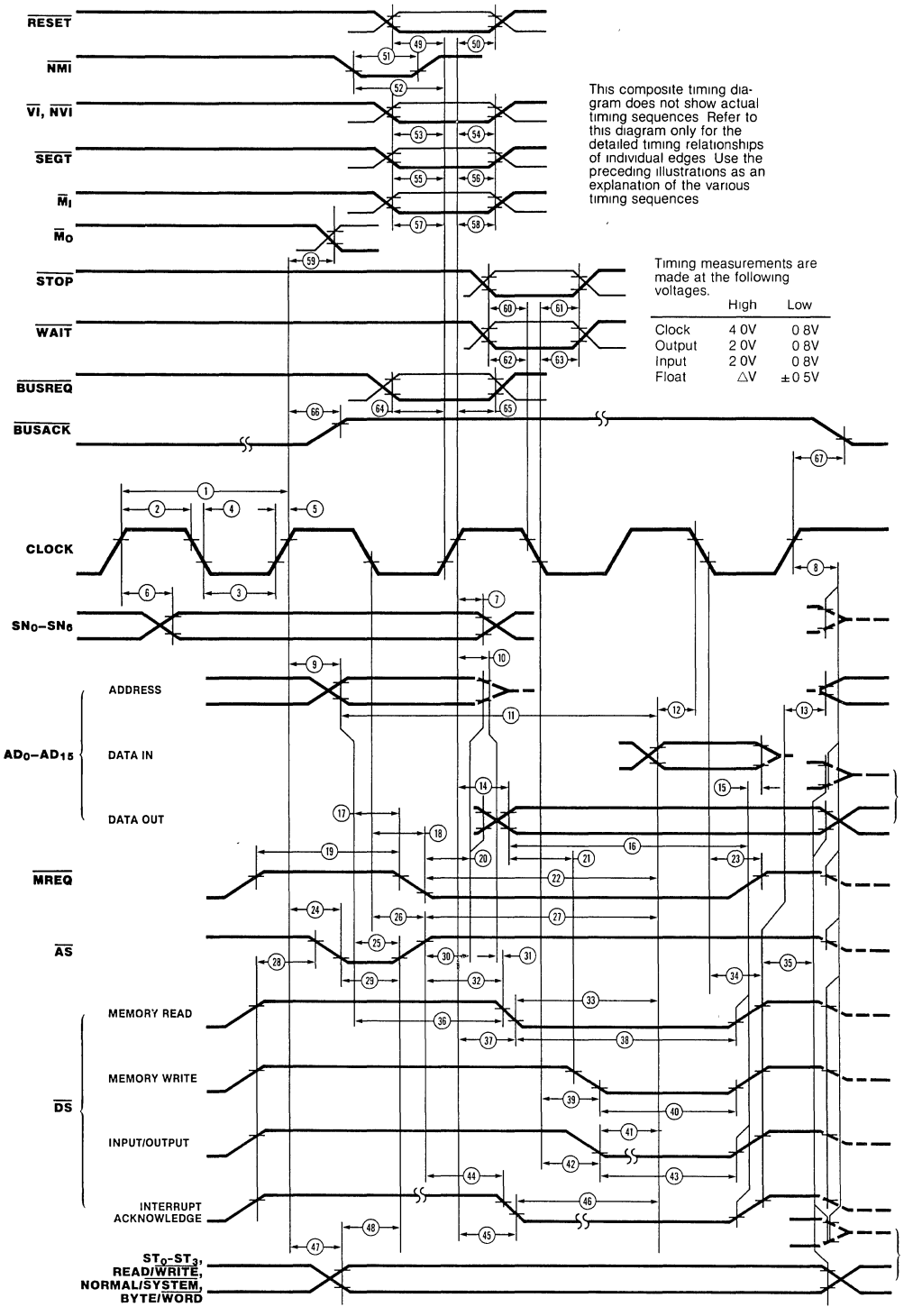


Figure 19. Reset Timing

Composite AC Timing Diagram



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages.

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	ΔV	±0.5V

Number	Symbol	Parameter	Z8001/Z8002 Min (ns)	Z8001/Z8002 Max (ns)	Z8001A/Z8002A Min (ns)	Z8001A/Z8002A Max (ns)
1	TcC	Clock Cycle Time	250	2000	165	2000
2	TwCh	Clock Width (High)	105	2000	70	2000
3	TwCl	Clock Width (Low)	105	2000	70	2000
4	TfC	Clock Fall Time		20		10
5	TrC	Clock Rise Time		20		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		110
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		10	
8	TdC(Bz)	Clock ↑ to Bus Float		65		55
9	TdC(A)	Clock ↑ to Address Valid		100		75
10	TdC(Az)	Clock ↑ to Address Float		65		55
11	TdA(DR)	Address Valid to Read Data Required Valid		455*		305*
12	TsDR(C)	Read Data to Clock ↓ Setup Time	50		20	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80*		40*	
14	TdC(DW)	Clock ↑ to Write Data Valid		100		75
15	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0	
16	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	295*		195*	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	(55)*		(35)*	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80		70
19	TwMRh	$\overline{MREQ}$ Width (High)	210*		135*	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70*		35*	
21	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	55*		35*	
22	TdMR(DR)	$\overline{MREQ}$ ↓ to Read Data Required Valid	350*		225*	
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		60
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		60
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55*		35*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80
27	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid	340*		215*	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70*		35*	
29	TwAS	$\overline{AS}$ Width (Low)	85*		55*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	60*		30*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	70*		35*	
33	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid	185*		130*	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65
35	TdDS(DW)	$\overline{DS}$ ↑ to Write Data and STATUS Not Valid	75*		45*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180*		110*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		85
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275*		185*	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		80
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185*		110*	
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid	320*		200*	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		100
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410*		255*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065*		690*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		85
46	TdDSA(DR)	$\overline{DS}$ (Ack.) ↓ to Read Data Required Delay	435*		295*	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		85
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	60*		30*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		70	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		70	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		70	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		50	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	0		0	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		55	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		0	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		110	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0	
59	TdC(MO)	Clock ↑ to MO Delay		120		85
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		70	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		0	
62	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		30	
63	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		10	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		80	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		10	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		75
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		75

*Clock-cycle-time-dependent characteristics See table on following page

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8001/Z8002 Equation	Z8001A/Z8002A Equation
11	TdA(DR)	2TcC + TwCh - 150 ns	2TcC + TwCh - 95 ns
13	TdDS(A)	TwCl - 25 ns	TWCl - 30 ns
16	TdDW(DS)	TcC + TwCh - 60 ns	TcC + TwCh - 40 ns
17	TdA(MR)	TwCh - 50 ns	TwCh - 35 ns
19	TwMRh	TcC - 40 ns	TcC - 30 ns
20	TdMR(A)	TwCl - 35 ns	TwCl - 35 ns
21	TdDW(DSW)	TwCh - 50 ns	TwCh - 35 ns
22	TdMR(DR)	2TcC - 150 ns	2TcC - 105 ns
25	TdA(AS)	TwCh - 50 ns	TwCh - 35 ns
27	TdAS(DR)	2TcC - 160 ns	2TcC - 115 ns
28	TdDS(AS)	TwCl - 35 ns	TwCl - 35 ns
29	TwAS	TwCh - 20 ns	TwCh - 15 ns
30	TdAS(A)	TwCl - 45 ns	TwCl - 40 ns
32	TdAS(DSR)	TwCl - 35 ns	TwCl - 35 ns
33	TdDSR(DR)	TcC + TwCh - 170 ns	TcC + TwCh - 105 ns
35	TdDS(DW)	TwCl - 30 ns	TwCl - 25 ns
36	TdA(DSR)	TcC - 70 ns	TcC - 55 ns
38	TwDSR	TcC + TwCh - 80 ns	TcC + TwCh - 50 ns
40	TwDSW	TcC - 65 ns	TcC - 55 ns
41	TdDSI(DR)	2TcC - 180 ns	2TcC - 130 ns
43	TwDS	2TcC - 90 ns	2TcC - 75 ns
44	TdAS(DSA)	4TcC + TwCl - 40 ns	4TcC + TwCl - 40 ns
46	TdDSA(DR)	2TcC + TwCh - 170 ns	2TcC + TwCh - 105 ns
48	TdS(AS)	TwCh - 55 ns	TwCh - 40 ns

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

Storage Temperature . . . . . -65°C to +150°C

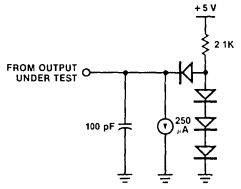
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating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

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- 0°C to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- −55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

The product number for each operating temperature range may be found in the ordering information section.



All ac parameters assume a load capacitance of 100 pF max, except for parameter 6 (50 pF max) Timing references between two output signals assume a load difference of 50 pF max

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
$V_{IH\ RESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	V_{CC} to .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	$0.4 \le V_{IN} \le +2.4$ V
$I_{IL\ SEGT}$	Input Leakage on SEGT pin	−100	100	µA	
I_{OL}	Output Leakage		±10	µA	$0.4 \le V_{IN} \le +2.4$ V
I_{CC}	V_{CC} Supply Current		300	mA	

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8001	CE	4.0 MHz	CPU (segmented, 40-pin)	Z8002	CE	4.0 MHz	CPU (nonsegmented, 40-pin)
Z8001	CM	4.0 MHz	Same as above	Z8002	CM	4.0 MHz	Same as above
Z8001	CMB	4.0 MHz	Same as above	Z8002	CMB	4.0 MHz	Same as above
Z8001	CS	4.0 MHz	Same as above	Z8002	CS	4.0 MHz	Same as above
Z8001	DE	4.0 MHz	Same as above	Z8002	DE	4.0 MHz	Same as above
Z8001	DS	4.0 MHz	Same as above	Z8002	DS	4.0 MHz	Same as above
Z8001	PE	4.0 MHz	Same as above	Z8002	PE	4.0 MHz	Same as above
Z8001	PS	4.0 MHz	Same as above	Z8002	PS	4.0 MHz	Same as above
Z8001A	CE	6.0 MHz	CPU (segmented, 40-pin)	Z8002A	CE	6.0 MHz	CPU (nonsegmented, 40-pin)
Z8001A	CS	6.0 MHz	Same as above	Z8002A	CS	6.0 MHz	Same as above
Z8001A	DE	6.0 MHz	Same as above	Z8002A	DE	6.0 MHz	Same as above
Z8001A	DS	6.0 MHz	Same as above	Z8002A	DS	6.0 MHz	Same as above
Z8001A	PE	6.0 MHz	Same as above	Z8002A	PE	6.0 MHz	Same as above
Z8001A	PS	6.0 MHz	Same as above	Z8002A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, CM = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8010
Z8000™ Z-MMU Memory Management Unit

Zilog

Product Specification

March 1981

Features

- Dynamic segment relocation makes software addresses independent of physical memory addresses.
- Sophisticated memory-management features include access validation that protects memory areas from unauthorized or unintentional access, and a write-warning indicator that predicts stack overflow.
- 64 variable-sized segments from 256 to 65,536 bytes can be mapped into a total physical address space of 16M bytes; all 64 segments are randomly accessible.
- Multiple MMUs can support several translation tables for each Z8001 address space.
- MMU architecture supports multi-programming systems and virtual memory implementations.

General Description

The Z8010 Memory Management Unit (MMU) manages the large 8M byte addressing spaces of the Z8001 CPU. The MMU provides dynamic segment relocation as well as numerous memory protection features.

Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually located in the physical memory. It also provides a flexible, efficient method for supporting multi-programming systems. The MMU uses a translation table to transform the 23-bit logical address output from the Z8001 CPU into a 24-bit address for the physical memory. (Only logical memory addresses go to an MMU for translation; I/O addresses and data, in general, must bypass this component.)

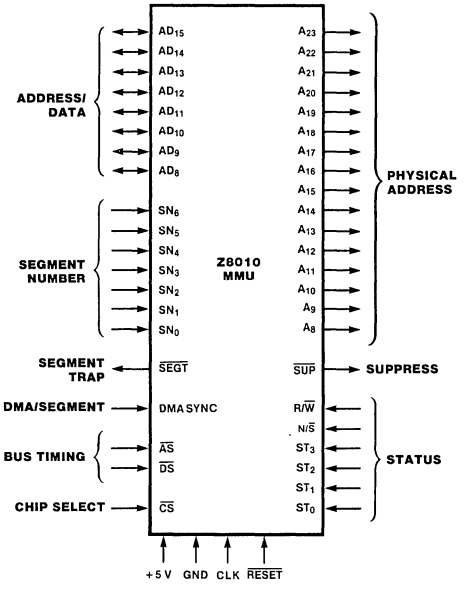


Figure 1. Pin Functions

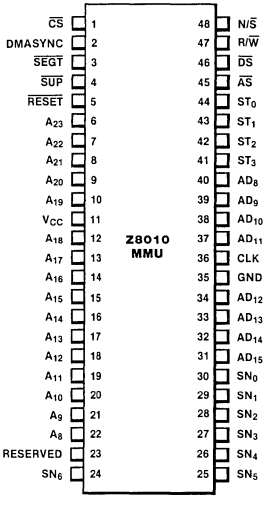


Figure 2. Pin Assignments

General Description (Continued)

Memory segments are variable in size from 256 bytes to 64K bytes, in increments of 256 bytes. Pairs of MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of MMUs can be used to accommodate multiple translation tables for System and Normal operating modes, or to support more sophisticated memory-management systems.

MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a number of attributes when its descriptor initially entered into the MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001 CPU. If a mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause.

Segments are protected by modes of permitted use, such as read only, system only, execute only and CPU-access only. Other segment management features include a write-warning zone useful for stack operations and status flags that record read or write accesses to each segment.

The MMU is controlled via 22 Special I/O instructions from the Z8001 CPU in System mode. With these instructions, system software can assign program segments to arbitrary memory locations, restrict the use of segments and monitor whether segments have been read or written.

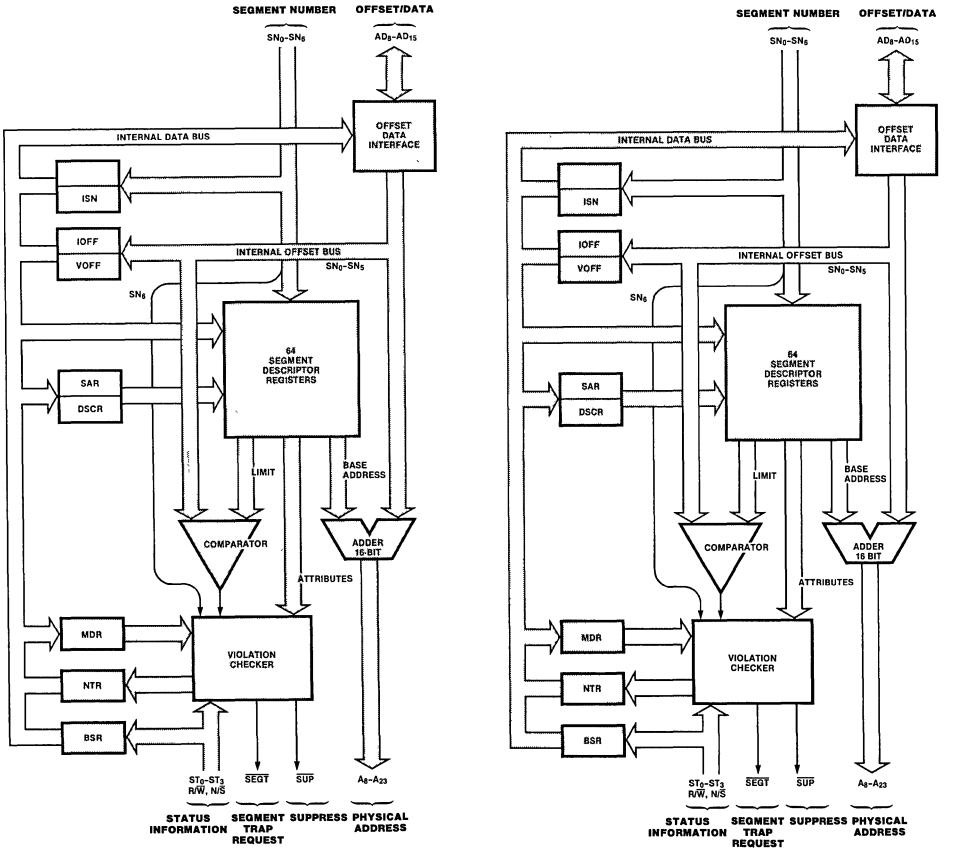


Figure 3. The shaded areas in these block diagrams illustrate the resources used in the two modes of MMU operation. In the Address Translation Mode shown on the left, addresses are translated automatically. In the Command Mode shown on the right, specific registers are accessed using Special I/O commands.

Segmented Addressing

A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data set can reside in its own segment.

The 8M byte Z8001 addressing spaces are divided into 128 relocatable segments of up to 64K bytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any byte relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately.

The MMU divides the physical memory into 256-byte blocks. Segments consist of physically contiguous blocks. Certain segments may be designated so that writes into the last block generate a warning trap. If such a segment is used as a stack, this warning can be used to increase the segment size and prevent a stack overflow error.

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical addresses.* The MMU takes the logical addresses and transforms them into the *physical addresses* required for accessing the memory (Figure 4). This address transformation process is called *relocation.*

The relocation process is transparent to user software. A translation table in the MMU associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit logical address offset is added to the physical base address to obtain the actual physical memory location. Because a base address always has a low byte equal to zero, only the high-order 16 bits are stored in the MMU and used in the addition. Thus the low-order byte of the physical memory location is the same as the low-order byte of the logical address offset. This low-order byte therefore bypasses the MMU, thus reducing the number of pins required.

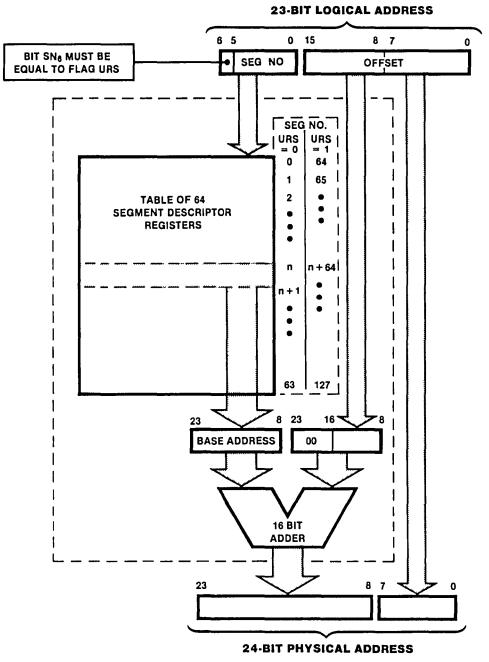


Figure 4. Logical-to-Physical Address Translation

Memory Protection

Each memory segment is assigned several attributes that are used to provide memory access protection. A memory request from the Z8001 CPU is accompanied by status information that indicates the attributes of the memory request. The MMU compares the memory request attributes with the segment attributes and generates a Trap Request whenever it detects an attribute violation. Trap Request informs the Z8001 CPU and the system control program of the violation so that appropriate action can be taken to recover. The MMU also generates the Suppress signal $\overline{SUP}$ in the event of an access violation. Suppress can be used by a memory system to inhibit stores into the memory and thus protect the contents of the memory from erroneous changes.

Five attributes can be associated with each segment. When an attempted access violates any one of the attributes associated with a segment, a Trap Request and a Suppress signal are generated by the MMU. These attributes are read only, execute only, system access only, inhibit CPU accesses and inhibit DMA accesses.

Segments are specified by a base address and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access that lies outside the segment is attempted, Trap Request and Suppress are generated.

Normally the legal range of offsets within a segment is from 0 to 256N + 255 bytes, where $0 \leq N \leq 255$. However, a segment may be specified so that legal offsets range from 256N to 65,535 bytes, where $0 \leq N \leq 255$. The latter type of segment is useful for stacks since the Z8001 stack manipulation instructions cause stacks to grow toward lower memory locations. Thus when a stack grows to the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

MMU Register Organization

The MMU contains three types of registers: Segment Descriptor, Control and Status. A set of 64 Segment Descriptor Registers supplies the information needed to map logical memory addresses to physical memory locations. The segment number of a logical address determines which Segment Descriptor Register is used in address translation. Each Descriptor Register also contains the necessary information for checking that the segment location referenced is within the bounds of the segment and that the type of reference is permitted. It also indicates whether the segment has been read or written.

In addition to the Segment Descriptor Registers, the Z8010 MMU contains three 8-bit control registers for programming the device and six 8-bit status registers that record information in the event of an access violation.

Segment Descriptor Registers. Each of the 64 Descriptor Registers contains a 16-bit base address field, an 8-bit limit field and an 8-bit attribute field (Figure 5). The base address field is subdivided into high- and low-order bytes that are loaded one byte at a time when the descriptor is initialized. The limit field contains a value N that indicates N + 1 blocks of 256 bytes have been allocated to the segment.*

The attribute field contains eight flags (Figure 6). Five are related to protecting the segment against certain types of access, one indicates the special structure of the segment, and two encode the types of accesses that have been made to the segment. A flag is set when its value is 1. The following brief descriptions indicate how these flags are used.

Read-Only (RD). When this flag is set, the segment is read only and is protected against any write access.

System-Only (SYS). When this flag is set, the segment can be accessed only in system mode, and is protected against any access in normal mode.

CPU-Inhibit (CPUI). When this flag is set, the segment is not accessible to the currently executing process, and is protected against any memory access by the CPU. The segment is, however, accessable under DMA.

Execute-Only (EXC). When this flag is set, the segment can be accessed only during an instruction fetch cycle, and thus is protected against any access during other cycles.

DMA-Inhibit (DMAI). When this flag is set, the segment can be accessed only by the CPU, and thus is protected against any access under DMA.

Direction and Warning (DIRW). When this flag is set, the segment memory locations are considered to be organized in descending order and each write to the segment is checked for access to the last 256-byte block. Such an access generates a trap to warn of potential segment overflow, but no Suppress signal is generated.

Changed (CHG). When this flag is set, the segment has been changed (written). This bit is set automatically during any write access to this segment if the write access does not cause any violation.

Referenced (REF). When this flag is set, the segment has been referenced (either read or written). This bit is set automatically during any access to the segment if the access does not cause a violation.

*In the stack mode, segment size is 64K-256N.

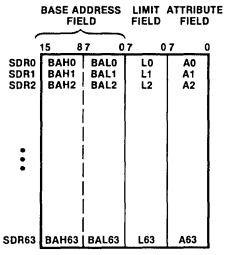


Figure 5. Segment Descriptor Registers

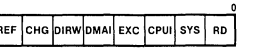


Figure 6. Attribute Field in Segment Descriptor Register

Control Registers. The three user-accessible 8-bit control registers in the MMU direct the functioning of the MMU (Figure 7). The Mode Register provides a sophisticated method for selectively enabling MMUs in multiple-MMU configurations. The Segment Address Register (SAR) selects a particular Segment Descriptor Register to be accessed during a control operation. The Descriptor Selection Counter Register points to a byte within the Segment Descriptor Register to be accessed during a control operation.

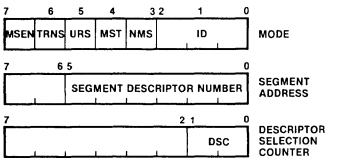


Figure 7. Control Registers

The Mode Register contains a 3-bit identification field (ID) that distinguishes among eight enabled MMUs in a multiple-MMU configuration. This field is used during the segment trap acknowledge sequence (refer to the section on Segment Trap and Acknowledge). In addition, the Mode Register contains five flags.

Multiple Segment Table (MST). This flag indicates whether multiple segment tables are present in the hardware configuration. When this flag is set, more than one table is present and the N/$\overline{S}$ line must be used to determine whether the MMU contains the appropriate table.

Normal Mode Select (NMS). This flag indicates whether the MMU is to translate addresses when the N/$\overline{S}$ line is High or Low. If the MST flag is set, the N/$\overline{S}$ line must match the NMS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

MMU Register Organization (Continued)

Upper Range Select (URS). This flag is used to indicate whether the MMU contains the lower-numbered segment descriptors or the higher-numbered segment descriptors. The most significant bit of the segment number must match the URS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

Translate (TRNS). This flag indicates whether the MMU is to translate logical program addresses to physical memory locations or is to pass the logical addresses unchanged to the memory and without protection checking. In the non-translation mode, the most significant byte of the output is the 7-bit segment number and the most significant bit is 0. When this flag is set, the MMU performs address translation and attribute checking.

Master Enable (MSEN). This flag enables or disables the MMU from performing its address translation and memory protection functions. When this flag is set, the MMU performs these tasks; when the flag is clear the Address lines of the MMU remain 3-stated.

The Segment Address Register (SAR) points to one of the 64 segment descriptors. Control commands to the MMU that access segment descriptors implicitly use this pointer to select one of the descriptors. This register has an auto-incrementing capability so that multiple descriptors can be accessed in a block read/write fashion.

The Descriptor Selection Counter Register holds a 2-bit counter that indicates which byte in the descriptor is being accessed during the reading or writing operation. A value of zero in this counter indicates the high-order byte of the base address field is to be accessed, one indicates the low-order byte of the base address, two indicates the limit field and three indicates the attribute field.

Status Registers. Six 8-bit registers contain information useful in recovering from memory access violations (Figure 8). The Violation Type Register describes the conditions that generated the trap. The Violation Segment Number and Violation Offset Registers record the most-significant 15 bits of the logical address that causes a trap. The Instruction Segment Number and Offset Registers record the most-significant 15 bits of the logical address of the last instruction fetched before the first accessing violation. These two registers can be used in conjunction with external circuitry that records the low-order offset byte. At the time of the addressing violation, the Bus Cycle Status Register records the bus cycle status (status code, read/write mode and normal/system mode).

The MMU generates a Trap Request for two general reasons: either it detects an access violation, such as an attempt to write into a read-only segment, or it detects a warning condition, which is a write into the lowest 256 bytes of a segment with the DIRW flag set. When a violation or warning condition is detected, the MMU generates a Trap Request and automatically sets the appropriate flags. The eight flags in the Violation Type Register describe the cause of a trap.

Read-Only Violation (RDV). Set when the CPU attempts to access a read-only segment and the R/$\overline{W}$ line is Low.

System Violation (SYSV). Set when the CPU accesses a system-only segment and the N/$\overline{S}$ line is High.

CPU-Inhibit Violation (CPUIV). Set when the CPU attempts to access a segment with the CPU-inhibit flag set.

Execute-Only Violation (EXCV). Set when the CPU attempts to access an execute-only segment in other than an instruction fetch cycle

Segment Length Violation (SLV). Set when an offset falls outside of the legal range of a segment.

Primary Write Warning (PWW). Set when an access is made to the lowest 256 bytes of a segment with the DIRW flag set.

Secondary Write Warning (SWW). Set when the CPU pushes data into the last 256 bytes of a system stack and EXCV, CPUIV, SLV, SYSV, RDV or PWW is set. Once this flag is set, subsequent write warnings for accessing the system stack do not generate a Segment Trap request.

Fatal Condition (FATL). Set when any other flag in the Violation Type Register is set and either a violation is detected or a write warning condition occurs in normal mode. This flag is not set during a stack push in system mode that results in a warning condition. This flag indicates a memory access error has occurred in the trap processing routine. Once set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

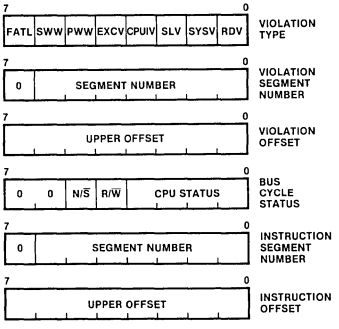


Figure 8. Status Registers

Segment Trap and Acknowledge

The Z8010 MMU generates a Segment Trap when it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress, which can be used to inhibit memory writes and to flag special data to be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a CPU-generated violation occurs, Suppress is asserted for that cycle and all subsequent CPU instruction execution cycles until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only—no Segment Trap Requests are ever generated during DMA cycles.

Segment traps to the Z8001 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU issues a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is aborted (the MMU has been designed so that this dummy cycle is ignored). During the acknowledge cycle all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap Request outputs a 1 on the A/D line associated with the number in its ID field; an MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a segment trap acknowledge cycle, an MMU uses A/D line 8+i if its ID field is i.

Following the acknowledge cycle the CPU automatically pushes the Program Status and Program Counter onto the system stack and loads another Program Status and Program Counter from the Program Status Area. The Segment Trap line is reset during the segment trap acknowledge cycle. Suppress is not generated during the stack push. If the store creates a write warning condition, a Segment Trap Request is generated and is serviced at the end of the context swap. The SWW flag is also set. Servicing this second Segment Trap Request also creates a write warning condition, but because the SWW flag is set, no Segment Trap Request is generated. If a violation rather than a write warning occurs during the context swap, the FATL flag is set rather than the SWW flag. Subsequent violations cause Suppress to be asserted but not Segment Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to process the trap they created.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the Violation Type Register reset.

Virtual Memory

Several features of the MMU can be used in conjunction with external circuitry to support virtual memory for the Z8001. Segment Trap Request can be used to signal the CPU in the event that a segment is not in primary memory. The CPU-Inhibit Flag can be used to indicate whether a segment is in the memory or in secondary storage. The Changed and Altered Flags in the attribute field for each segment can aid in implementing efficient segment management policies. The Status Registers can be used in recovering from virtual memory access faults.

Multiple MMUs

MMU architecture directly supports two methods for multiple MMU configurations. The first approach extends single-MMU capability for handling 64 segments to a dual-MMU configuration that manages the 128 different segments the Z8001 can address. This scheme uses the URS flag in the Mode Register in connection with the high-order bit of the segment number (SN_6).

The second approach uses several MMUs to implement multiple translation tables. Multiple tables can be used to reduce the time required to switch tasks by assigning separate tables to each task. Multiple translation tables for multi-task environments can use the Master Enable Flag to enable the appropriate MMUs through software. Multiple translation tables may also be used to extend the physical memory size beyond 16 megabytes by separating system from normal memory and/or program from data memory. The MST and NMS flags in the Mode Register can be used in conjunction with the $N/\overline{S}$ line to select the MMU that contains the appropriate table. Special external circuitry that monitors the CPU Status lines can manipulate the MMU $N/\overline{S}$ line to perform this selection.

DMA Operation

Direct memory access operations may occur between Z8001 instruction cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal mode of operation. For each memory access, the segment attributes are checked and if a violation is detected, Suppress is activated. Unlike a CPU violation that automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per memory access basis.

The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA, hence warning conditions are not signaled. Trap Requests are not issued because the CPU cannot acknowledge such a request.

At the start of a DMA cycle, DMASYNC must go Low, indicating to the MMU the beginning of a DMA cycle. A Low DMASYNC inhibits the MMU from using an indeterminate segment number on lines SN_0–SN_6. When the DMA logical memory address is valid, the DMASYNC line must be High on a rising edge of Clock and the MMU then performs its address translation and access protection functions. Upon the release of the bus at the termination of the DMA cycle the DMASYNC line must again be High. After two clock cycles of DMASYNC High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two cycles after the CPU regains control of the bus. During CPU cycles, DMASYNC should always be High.

MMU Commands

The various registers in the MMU can be read and written using Z8001 CPU special I/O commands. These commands have machine cycles that cause the Status lines to indicate an SIO operation is in progress. During these machine cycles the MMU enters command mode. In this mode, the rising edge of the Address Strobe indicates a command is present on the AD_8–AD_{15}. If this command indicates that data is to be written into one of the MMU registers, the data is read from AD_8–AD_{15} while Data Strobe is Low. If the command indicates that data is to be read from one of the MMU registers, the data is placed on AD_8–AD_{15} while Data Strobe is Low.

There are ten commands that read or write various fields in the Segment Descriptor Register. The status of the Read/Write line indicates whether the command is a read or a write.

The auto-incrementing feature of the Segment Address Register (SAR) can be used to block load segment descriptors using the repeat forms of the Special I/O instructions. The SAR is autoincremented at the end of the field. In accessing the base field, first the high-order byte is selected and then the low-order byte. The command accessing the entire Descriptor Register references the fields in the order of base address, limit and attribute.

Opcode (Hex)	Instruction
08	Read/Write Base Field
09	Read/Write Limit Field
0A	Read/Write Attribute Field
0B	Read/Write Descriptor (all fields)
0C	Read/Write Base Field; Increment SAR
0D	Read/Write Limit Field; Increment SAR
0E	Read/Write Attribute Field; Increment SAR
0F	Read/Write Descriptor; Increment SAR
15	Set All CPU-Inhibit Attribute Flags
16	Set All DMA-Inhibit Attribute Flags

Three commands are used to read and write the control registers.

Opcode (Hex)	Instruction
00	Read/Write Mode Register
01	Read/Write Segment Address Register
20	Read/Write Descriptor Selector Counter Register

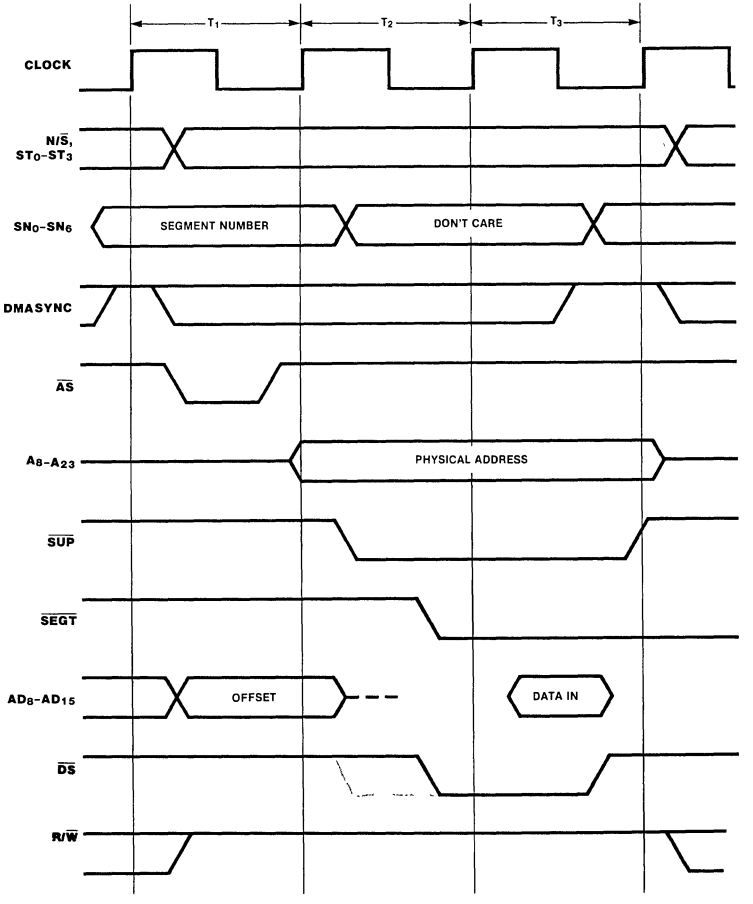
The Status Registers are read-only registers, although the Violation Type Register (VTR) can be reset. Nine instructions access these registers.

Opcode (Hex)	Instruction
02	Read Violation Type Register
03	Read Violation Segment Number Register
04	Read Violation Offset (High-byte) Register
05	Read Bus Status Register
06	Read Instruction Segment Number Register
07	Read Instruction Offset (High-byte) Register
11	Reset Violation Type Register
13	Reset SWW Flag in VTR
14	Reset FATL Flag in VTR

MMU Timing

The Z8010 translates addresses and checks for access violations by stepping through sequences of basic clock cycles corresponding to the cycle structure of the Z8001 CPU. The following timing diagrams show the relative timing relationships of MMU signals during the basic operations of memory read/write and MMU control commands. For exact timing information, refer to the composite timing diagram.

Memory Read and Write. Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 inputs. During a memory read cycle (Figure 9) the 7-bit segment number is input on SN_0-SN_6 one clock period earlier than the address offset; a High on DMASYNC during T_3 indicates that the segment offset data is valid. The most significant eight bits of the address offset are placed on the AD_0-AD_{15} inputs early in the



MMU Timing (Continued)

first clock period. Valid address offset data is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The most significant 16-bits of the address (physical memory location) remain valid until the end of T_3. Segment Trap Request and Suppress are asserted in T_2. Segment Trap Request remains Low until Segment Trap Acknowledge is received. Suppress is asserted during the current machine cycle and terminates during T_3. Suppress is repeatedly asserted during CPU instruction execution cycles until the current instruction has terminated.

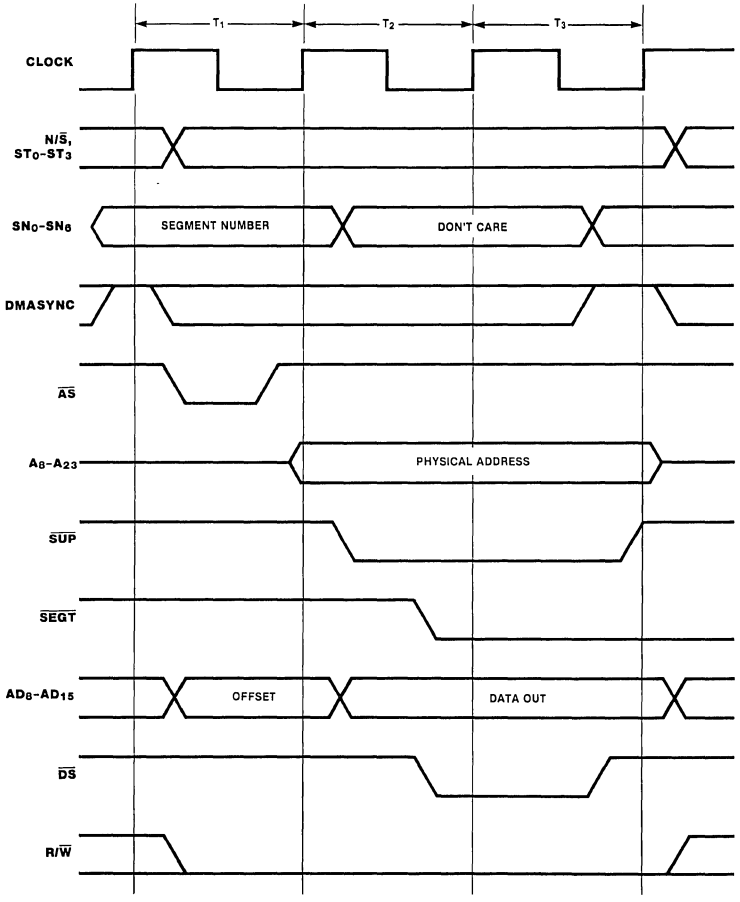


Figure 10. Memory Write Timing

MMU Timing (Continued)

MMU Command Cycle. During the command cycle of the MMU (Figure 11), commands are placed on the Address/Data lines during T_1. The Status lines indicate that a special I/O instruction is in progress, and the Chip Select line enables the appropriate MMU for that command. Data to be written to a register in the MMU must be valid on the Address/Data lines late in T_2. Data read from the MMU is placed on the Address/Data lines late in the T_{WA} cycle.

Input/Output and Refresh. Input/Output and Refresh operations are indicated by the status lines ST_0-ST_3. During these operations, the MMU refrains from any address translation or protection checking. The address lines A_8-A_{23} remain 3-stated.

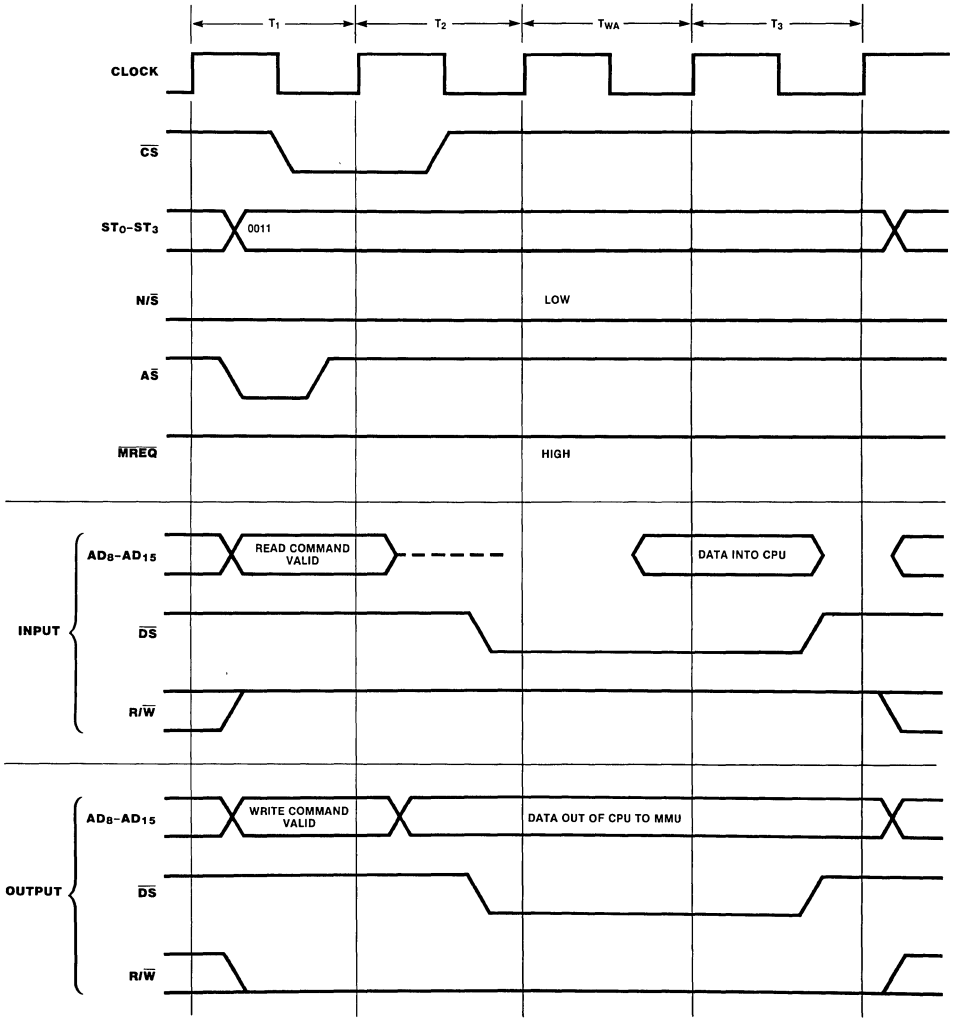


Figure 11. I/O Command Timing

MMU Timing (Continued)

Reset. The MMU can be reset by either hardware or software mechanisms. A hardware reset occurs on the falling edge of the Reset signal; a software reset is performed by a Z8001 special I/O command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low, the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the $\overline{AD_8}$-$\overline{AD_{15}}$ and A_8-A_{23} lines are 3-stated. The $\overline{SUP}$ and $\overline{SEGT}$ open-drain outputs are not driven. If the Master Enable flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with the Chip Select line.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state (as if no violations or warnings had occurred). Note that the hardware and software resets have different effects.

Segment Trap and Acknowledge. The Z8010 MMU generates a segment trap whenever it detects an access violation or a write into the lowest block of a segment with the DIRW flag set. In the case of an access violation, the MMU also activates Suppress. This Suppress signal can be used to inhibit memory writes and to flag special data to be returned on a read access. The Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU cycles until the end of the instruction; intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only, but no Trap Request is generated.

When the MMU issues a Segment Trap Request it awaits a Segment Trap Acknowledge. Subsequent violations occurring before the Trap Acknowledge is received are still detected and handled appropriately. During the Segment Trap Acknowledge cycle, the MMU drives one of its Address/Data lines High; the particular line selected is a function of the identification field of the mode register. After the Segment Trap has been acknowledged by the Z8001 CPU, the Violation Status Register should be read via the Special I/O commands in order to determine the cause of the trap. The Trap Type Register should also be reset so that subsequent traps will be recorded correctly.

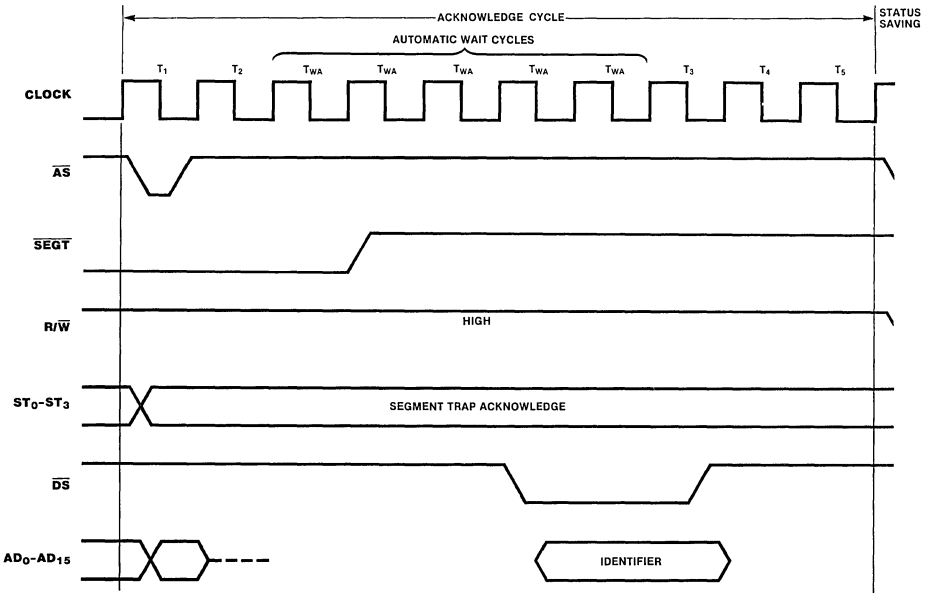


Figure 12. Segment Trap and Acknowledge Timing

Pin Description

A_8-A_{23}. *Address Bus* (outputs, active High, 3-state). These address lines are the 16 most significant bits of the physical memory location.

AD_8-AD_{15}. *Address/Data Bus* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for commands and for logical addresses intended for translation.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ indicates that AD_8–AD_{15}, ST_0–ST_3, CS, R/W and N/S are valid.

CLK. *System Clock* (input). CLK is the 5 V single-phase time-base input used for both the CPU and the MMU.

$\overline{CS}$. *Chip Select* (input, active Low). This line selects an MMU for a control command.

DMASYNC. *DMA/Segment Number Synchronization Strobe* (input, active High). A Low on this line indicates that a DMA access is occurring; a High indicates that the segment number is valid. It must always be High during CPU cycles.

$\overline{DS}$. *Data Strobe* (input, active Low). This line provides timing for the data transfer between the MMU and the Z8001 CPU.

N/$\overline{S}$. *Normal/System Mode* (input, Low = System mode). N/$\overline{S}$ indicates whether the Z8001 CPU or Z8016 DMA is in the Normal or System mode. The signal can also be used to switch between MMUs during different phases of an instruction.

Reserved. Do not connect.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the MMU.

R/$\overline{W}$. *Read/Write* (input, Low = write). R/$\overline{W}$ indicates the Z8001 CPU or Z8016 DMA is reading from or writing to memory or the MMU.

$\overline{SEGT}$. *Segment Trap Request* (output, active Low, open drain). The MMU interrupts the Z8001 CPU with a Low on this line when the MMU detects an access violation or write warning.

SN_0-SN_6. *Segment Number* (inputs, active High). The SN_0–SN_5 lines are used to address one of 64 segments in the MMU; SN_6 is used to selectively enable the MMU.

ST_0-ST_3. *Status* (inputs, active High). These lines specify the Z8001 CPU status.

ST_3-ST_0	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference (e.g., to an MMU)
0 1 0 0	Segment trap acknowledge
0 1 0 1	Non-maskable interrupt acknowledge
0 1 1 0	Non-vectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Status memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer
1 1 1 1	Reserved

$\overline{SUP}$. *Suppress* (output, active Low, open drain). This signal is asserted during the current bus cycle when any access violation except write warning occurs.

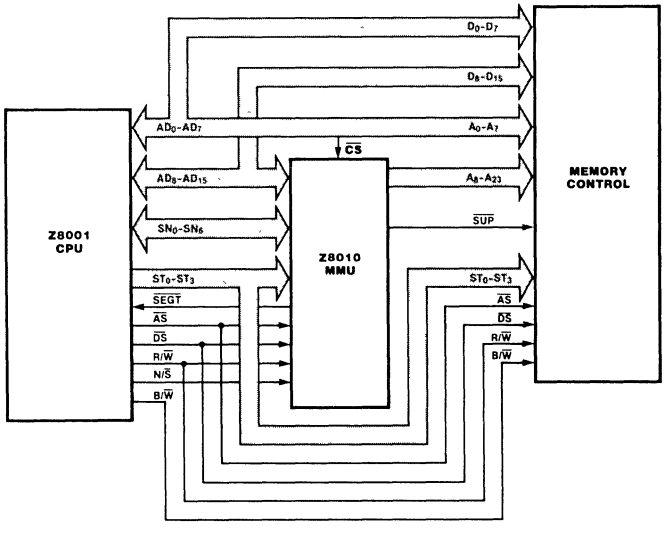


Figure 13. The MMU in a Z8001 System

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to + 70°C

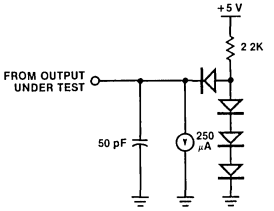
Storage Temperature . . . . . -65°C to +150°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- $0°C \leq T_A \leq +70°C$



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0 3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0 3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10	µA	$0\ 4 \leq V_{IN} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		300	mA	

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8010	CE	4.0 MHz	Z-MMU (48-pin)	Z8010	PS	4.0 MHz	Z-MMU (48-pin)
Z8010	CM	4.0 MHz	Same as above	Z8010A	CE	6.0 MHz	Z-MMU (48-pin)
Z8010	CMB	4.0 MHz	Same as above	Z8010A	CS	6.0 MHz	Same as above
Z8010	CS	4.0 MHz	Same as above	Z8010A	DE	6.0 MHz	Same as above
Z8010	DE	4.0 MHz	Same as above	Z8010A	DS	6.0 MHz	Same as above
Z8010	DS	4.0 MHz	Same as above	Z8010A	PE	6.0 MHz	Same as above
Z8010	PE	4.0 MHz	Same as above	Z8010A	PS	6.0 MHz	Same as above

NOTES C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C

AC Characteristics

Number	Symbol	Parameters	Min [3]	Max [3]	Notes
1	TcC	Clock Cycle Time	250		
2	TwCh	Clock Width (High)	105		
3	TwCl	Clock Width (Low)	105		
4	TfC	Clock Fall Time		25	
5	TrC	Clock Rise Time		25	
6	TdDSA(RDv)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		100	[1]
7	TdDSA(RDf)	$\overline{DS}$ ↑ (Acknowledge) to Read Data Float Delay	20	75	[1]
8	TdDSR(RDv)	$\overline{DS}$ ↓ (Read) to AD Output Driven Delay		100	[1]
9	TdDSR(RDf)	$\overline{DS}$ ↑ (Read) to Read Data Float Delay	20	75	[1]
10	TdC(WDv)	CLK ↑ to Write Data Valid Delay		160	
11	ThC(WDn)	CLK ↓ to Write Data Not Valid Hold Time	30		
12	TwAS	Address Strobe Width	60		
13	TsOFF(AS)	Offset Valid to $\overline{AS}$ ↑ Setup Time	45		
14	ThAS(OFFn)	$\overline{AS}$ ↑ to Offset Not Valid Hold Time	60		
15	TdAS(C)	$\overline{AS}$ ↓ to CLK ↑ Delay	110		
16	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	50		
18	TsSN(C)	SN Data Valid to CLK ↑ Setup Time	120		
19	ThC(SNn)	CLK ↑ to SN Data Not Valid Hold Time	0		
20	TdDMAS(C)	DMASYNC Valid to CLK ↑ Delay	120		
21	TdSTNR(AS)	Status (ST_0-ST_3, $N/\overline{S}$, $R/\overline{W}$) Valid to $\overline{AS}$ ↑ Delay	60		
22	TdC(DMA)	Clk ↑ to DMASYNC ↓ Delay	20		
23	TdST(C)	Status (ST_0-ST_3) Valid to CLK ↑ Delay	140		
24	TdDS(STn)	$\overline{DS}$ ↑ to Status Not Valid Delay	0		
25	TdOFF(Av)	Offset Valid to Address Output Valid Delay		175	[1,4,5]
26	TdST(Ad)	Status Valid to Address Output Driven Delay		155	[1,4,6]
27	TdDS(Af)	$\overline{DS}$ ↑ to Address Output Float Delay	30	160	[1]
28	TdAS(Ad)	$\overline{AS}$ ↓ to Address Output Driven Delay		145	[1, 4]
29	TdC(Av)	CLK ↑ to Address Output Valid Delay		255	[1, 4]
30	TdAS(SEGT)	$\overline{AS}$ ↑ to $\overline{SEGT}$ ↓ Delay		160	[1, 2]
31	TdC(SEGT)	CLK ↑ to $\overline{SEGT}$ ↑ Delay		300	[1, 2]
32	TdAS(SUP)	$\overline{AS}$ ↑ to $\overline{SUP}$ ↓ Delay		150	[1, 2]
33	TdDS(SUP)	$\overline{DS}$ ↑ to $\overline{SUP}$ ↑ Delay	30	155	[1, 2]
34	TsCS(AS)	Chip Select Input Valid to $\overline{AS}$ ↑ Setup Time	10		
35	ThAS(CSn)	$\overline{AS}$ ↑ to Chip Select Input Not Valid Hold Time	60		
36	TdAS(C)	$\overline{AS}$ ↑ to CLK ↑ Delay	0		
37	TsCS(RST)	Chip Select Input Valid to $\overline{RESET}$ ↑ Setup Time	150		
38	ThRST(CSn)	$\overline{RESET}$ ↑ to Chip Select Input Not Valid Hold Time	0		
39	TwRST	$\overline{RESET}$ Width (Low)	2TcC		
40	TdC(RDv)	CLK ↑ to Read Data Valid Delay		460	[1]
41	TdDS(C)	$\overline{DS}$ ↑ to CLK ↑ Delay	30		
42	TdC(DS)	CLK ↓ to $\overline{DS}$ ↑ Delay	0	110	

[1] 50 pF Load

[2] 2 2K Pull-up

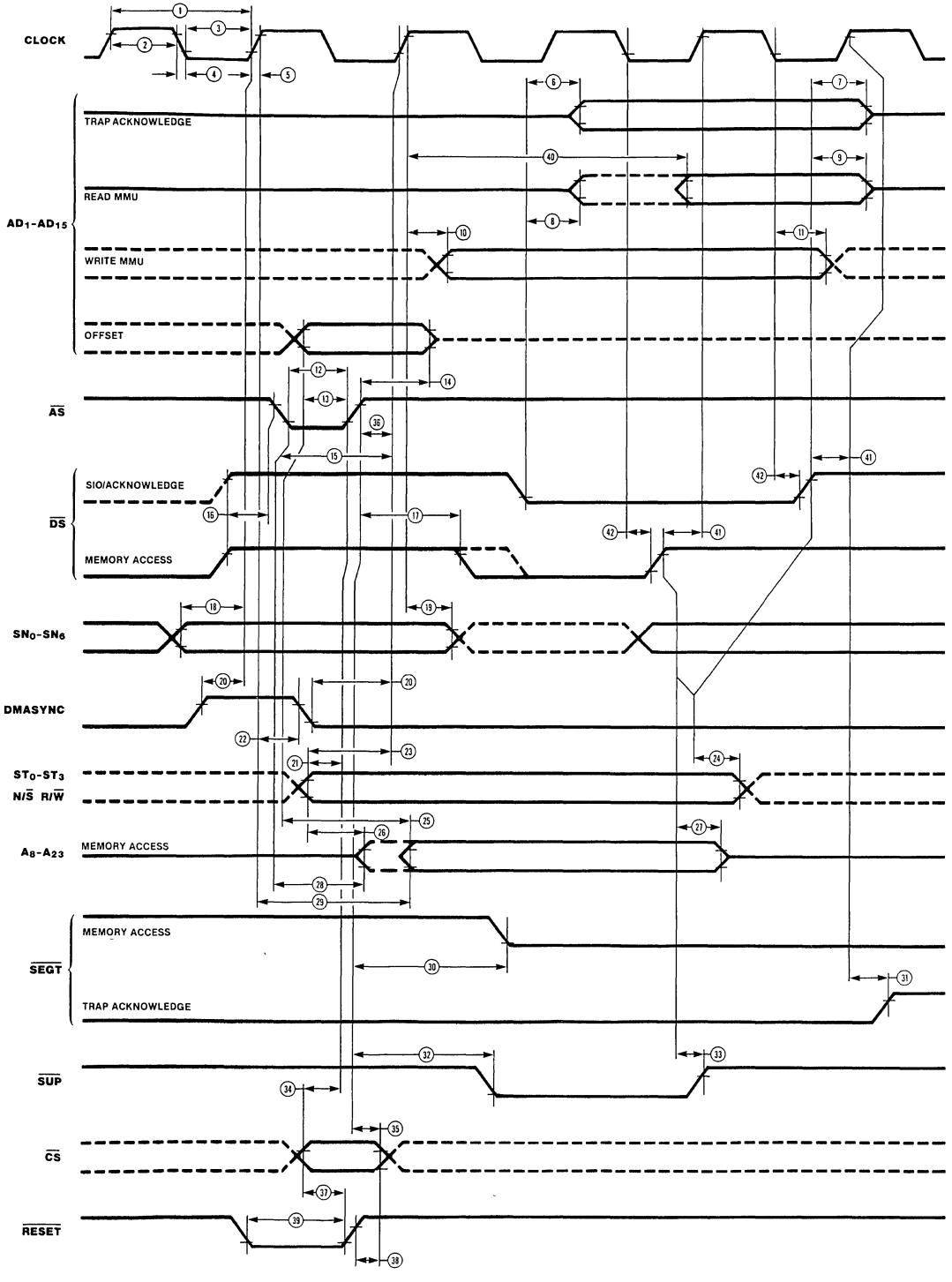
[3] All values in nanoseconds

[4] These values apply to the Z8010-3 version only A Z8010-2 version available soon will improve these values by 40 ns

[5] Parameter 9 (Clock ↑ to Address Valid) in the *Z8001/2 CPU Product Specification* is specified at 100 ns with 100 pF load With a load of 50 pF, this delay is reduced to 80 ns

[6] Parameter 47 (Clock ↑ to Status Valid Delay) in the *Z8001/2 CPU Product Specification* is specified at 110 ns with a 100 pF load With a 50 pF load, this delay is reduced to 100 ns

* Timings are preliminary and subject to change.


CLOCK
TRAP ACKNOWLEDGE
READ MMU
AD1-AD15
WRITE MMU
OFFSET
AS
SIO/ACKNOWLEDGE
DS
MEMORY ACCESS
SN0-SN6
DMASYNC
ST0-ST3
N/S R/W
A8-A23
MEMORY ACCESS
SEGT
TRAP ACKNOWLEDGE
SUP
CS
RESET

Z8030 Z8000™ Z-SCC Serial Communications Controller

Zilog

Product Specification

March 1981

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase-Locked Loop for clock recovery.
- Multi-protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.

General Description

The Z8030 Z-SCC Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with the Zilog Z-Bus. The Z-SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The Z-SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, Digital Phase-Locked Loops, and crystal oscillators that dramatically reduce the need for external logic.

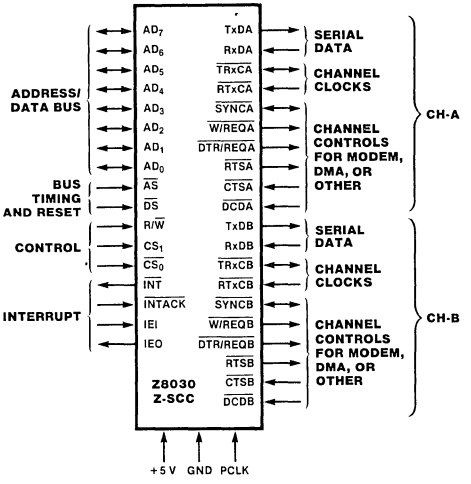


Figure 1. Pin Functions

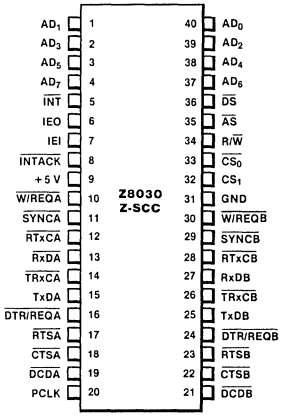


Figure 2. Pin Assignments

General Description (Continued)

The Z-SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The Z-SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-Bus daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8030 Z-SCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

Pin Description

The following section describes the pin functions of the Z-SCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

AD_0-AD_7. *Address/Data Bus* (bidirectional, active High, 3-state). These multiplexed lines carry register addresses to the Z-SCC as well as data or control information to and from the Z-SCC.

$\overline{AS}$. *Address Strobe* (input, active Low). Addresses on AD_0-AD_7 are latched by the rising edge of this signal.

$\overline{CS_0}$. *Chip Select 0* (input, active Low). This signal is latched concurrently with the addresses on AD_0-AD_7 and must be active for the intended bus transaction to occur.

CS_1. *Chip Select 1* (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS_1 must remain active throughout the transaction.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear to Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables their respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The Z-SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise-time signals. The Z-SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

$\overline{DS}$. *Data Strobe* (input, active Low). This signal provides timing for the transfer of data into and out of the Z-SCC. If $\overline{AS}$ and $\overline{DS}$ coincide, this is interpreted as a reset.

$\overline{DTR}/\overline{REQA}$, $\overline{DTR}/\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing a Z-SCC interrupt or the Z-SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the Z-SCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the Z-SCC interrupt daisy chain settles. When $\overline{DS}$ becomes active, the Z-SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of $\overline{AS}$.

PCLK. *Clock* (input). This is the master Z-SCC clock used to synchronize internal signals. PCLK is not required to have any phase relationship with the master system clock, although the frequency of this clock must be at least 90% of the CPU clock frequency for a Z8000. PCLK is a TTL level signal.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock of the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the

Pin Description (Continued)

$\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

R/$\overline{W}$. *Read/Write* (input). This signal specifies whether the operation to be performed is a read or a write.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit.

In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{TRxCA}$, $\overline{TRxCB}$. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{W/REQA}$, $\overline{W/REQB}$. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the Z-SCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the Z-SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a Z8000 Family peripheral, it interacts with the Z8000 CPU and other peripheral circuits and is part of the Z-Bus interrupt structure.

Data Communications Capabilities. The Z-SCC provides two independent full-duplex channels programmable for use in any common Asynchronous or Synchronous data-communication protocol. Figure 3 and the following description briefly detail these protocols.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a

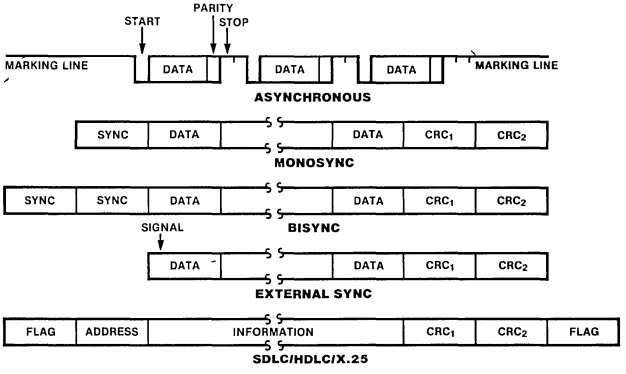


Figure 3. Some Z-SCC Protocols

Functional Description (Continued)

bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing or error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The Z-SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The Z-SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or with an external synchronization signal. Leading synchronous characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the Z-SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The Z-SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The Z-SCC supports Synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the Z-SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. The Z-SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the Z-SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s.

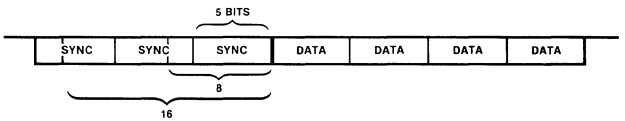


Figure 4. Detecting 5- or 7-Bit Synchronous Characters

Functional Description (Continued)

The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The Z-SCC can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the Z-SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The Z-SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the Z-SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The Z-SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the Z-SCC performs the functions of a secondary station while a Z-SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the Z-SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

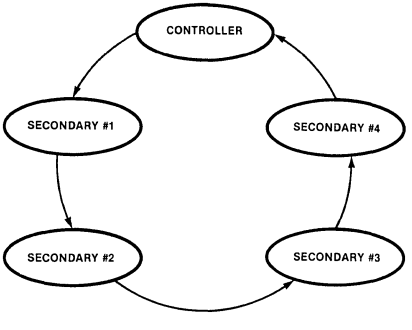


Figure 5. An SDLC Loop

Baud Rate Generator. Each channel in the Z-SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{baud rate} = \frac{1}{2\,(\text{time constant} + 2) \times (\text{BR clock period})}$$

Digital Phase-Locked Loop. The Z-SCC contains a Digital Phase-Locked Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the Z-SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the

Functional Description (Continued)

incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the Z-SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding The Z-SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark) a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the Z-SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0 the bit is a 1.

Auto Echo and Local Loopback. The Z-SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The Z-SCC is also capable of Local Loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The Z-SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the Z-SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be

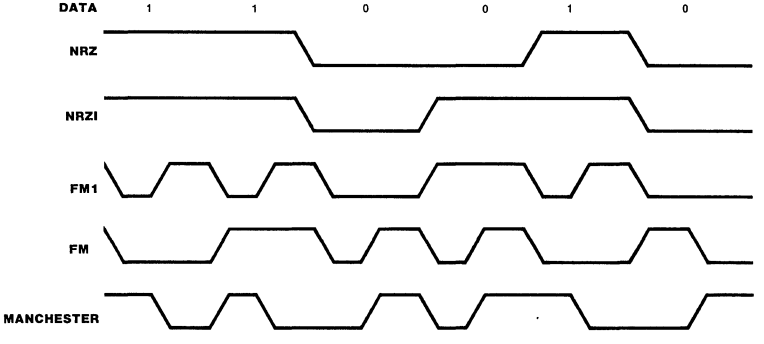


Figure 6. Data Encoding Methods

Functional Description (Continued)

read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. The Z-SCC interrupt scheme conforms to the Z-Bus specification. When a Z-SCC responds to an Interrupt Acknowledge signal ($\overline{\text{INTACK}}$) from the CPU, an interrupt vector may be placed on the A/D bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the Z-SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the Z-SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the Z-Bus interrupt priority chain (Figure 7). As a Z-Bus peripheral, the Z-SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{\text{INT}}$. The CPU then responds with $\overline{\text{INTACK}}$, and the interrupting device places the vector on the A/D bus.

In the Z-SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{\text{INT}}$ output is pulled Low, requesting an interrupt. In the Z-SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the Z-SCC and external to the Z-SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the Z-SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{\text{CTS}}$, $\overline{\text{DCD}}$, and $\overline{\text{SYNC}}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count

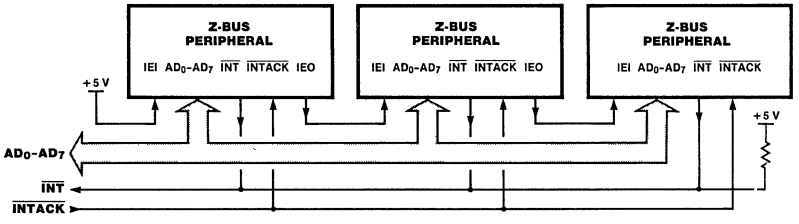


Figure 7. Z-Bus Interrupt Schedule

Functional Description (Continued)

in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the Z-SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The Z-SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the Z-SCC $\overline{\text{REQUEST}}$ output indicates that the Z-SCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the Z-SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}/\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The Z-SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to the Zilog Z-Bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides

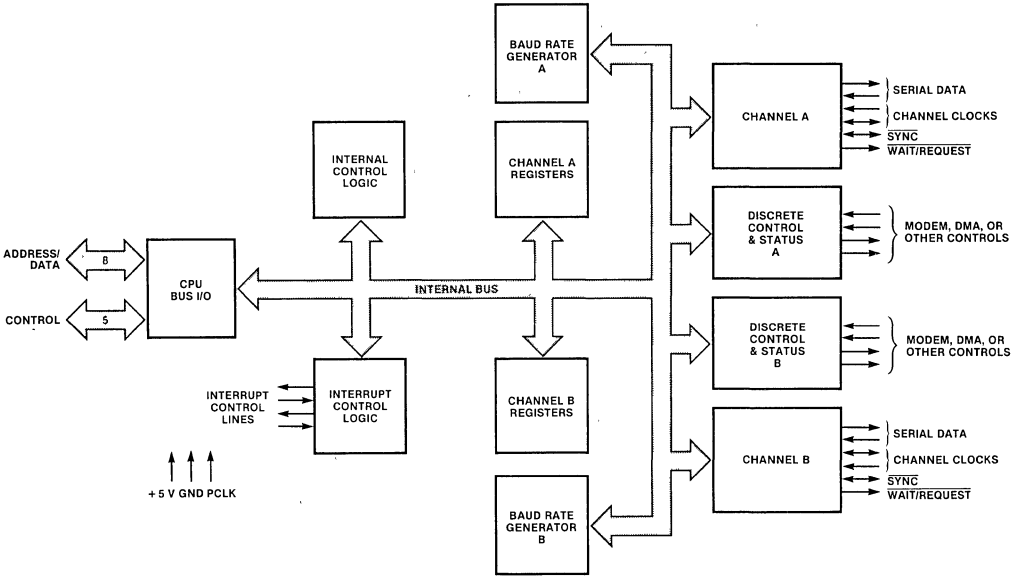


Figure 8. Block Diagram of Z-SCC Architecture

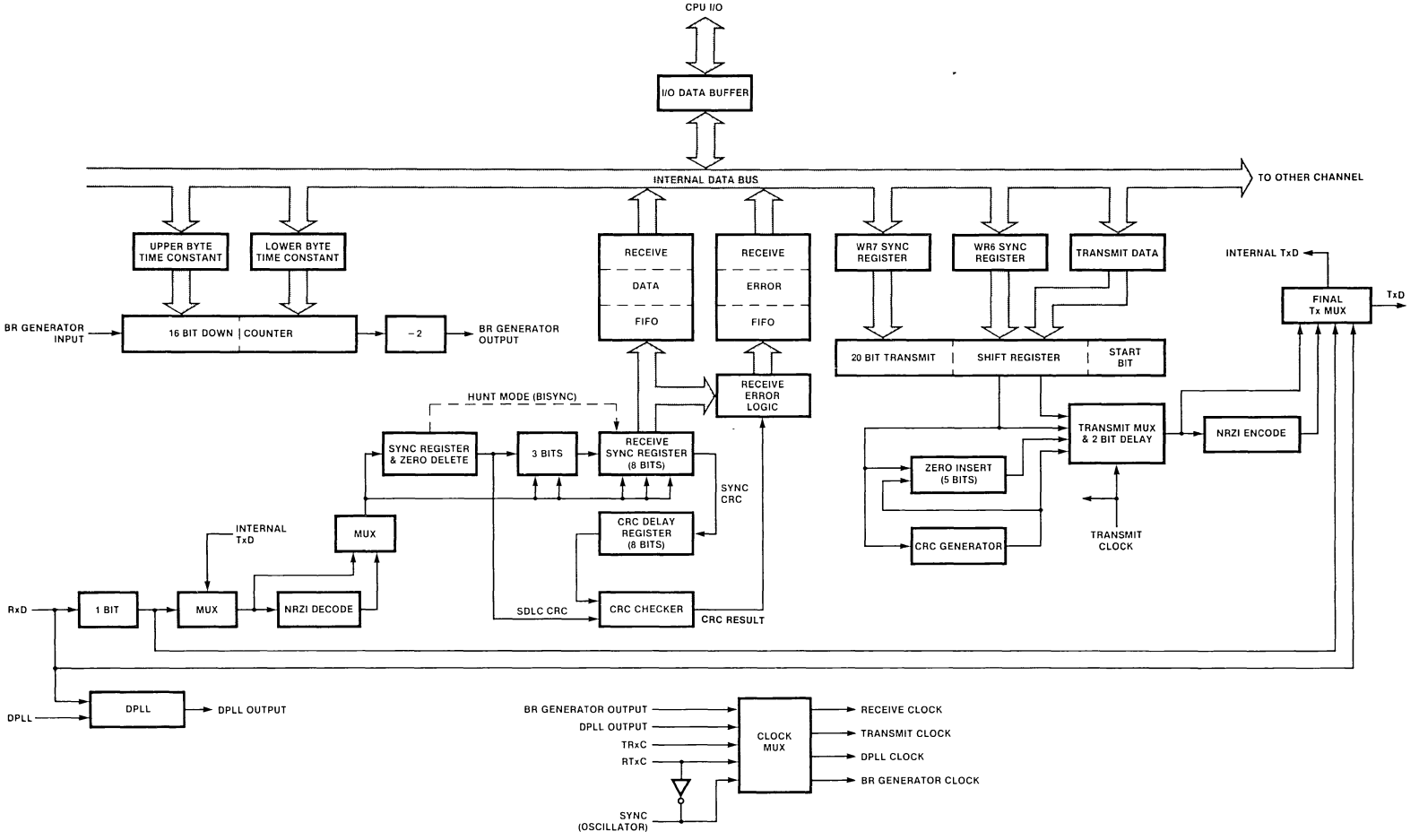


Figure 9. Data Path

Architecture (Continued)

formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two sync character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write-only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 — Write Registers 0 through 15.

RR0–RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The Z-SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20-bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD)

	Read Register Functions
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information
	Write Register Functions
WR0	CRC initialize, initialization commands for the various modes, shift right/shift left command
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The Z-SCC contains 13 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels. All of the registers in the Z-SCC are directly addressable. How the Z-SCC decodes the address placed on the address/data bus at the beginning of a Read or Write cycle is controlled by a command issued in WR0B.

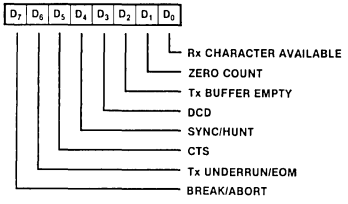
The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the Interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

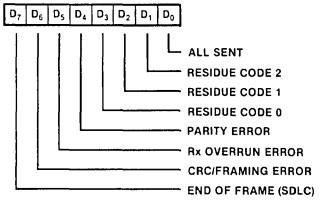
Read Registers. The Z-SCC contains eight read registers (actually nine, counting the receive buffer [RR8]) in each channel. Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

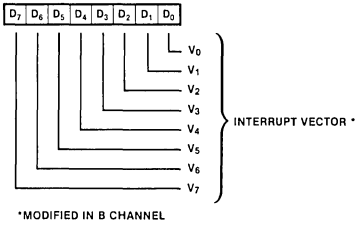
Read Register 0



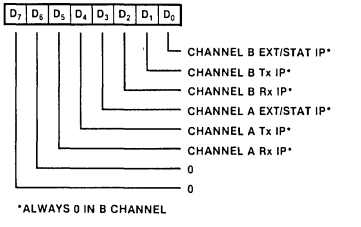
Read Register 1



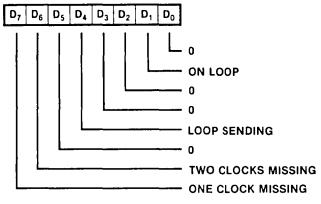
Read Register 2



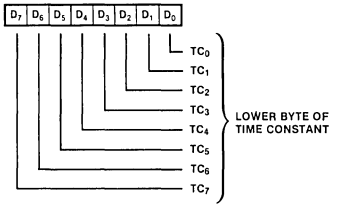
Read Register 3



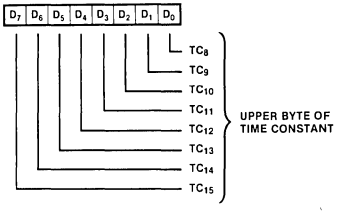
Read Register 10



Read Register 12



Read Register 13



Read Register 15

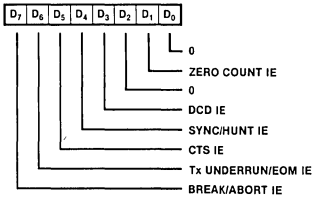
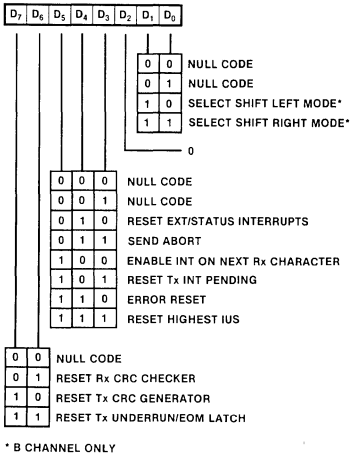


Figure 10. Read Register Bit Functions

Programming (Continued)

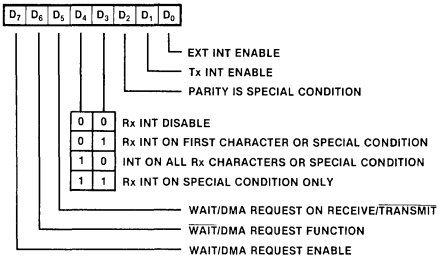
Write Registers. The Z-SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

Write Register 0

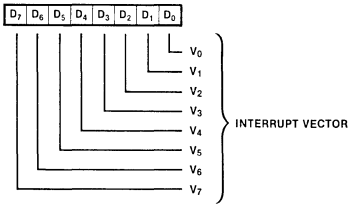


* B CHANNEL ONLY

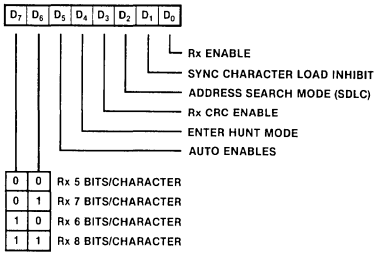
Write Register 1



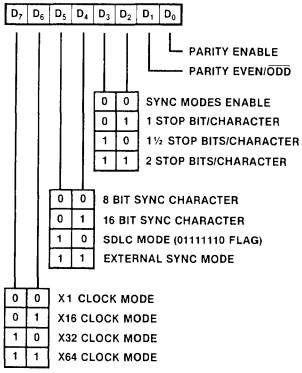
Write Register 2



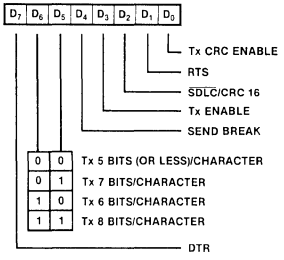
Write Register 3



Write Register 4



Write Register 5



Write Register 6

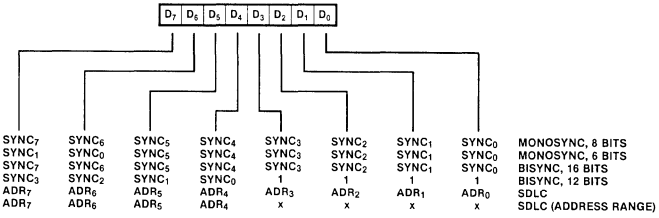
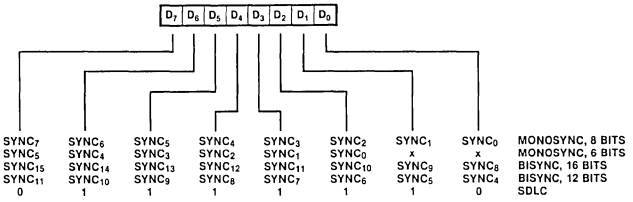
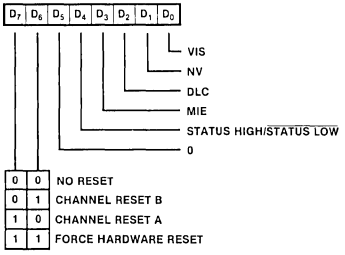


Figure 11. Write Register Bit Functions

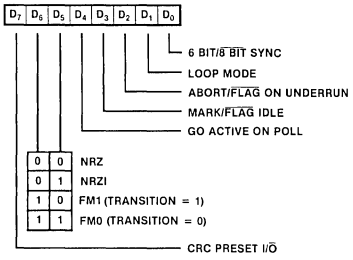
Write Register 7



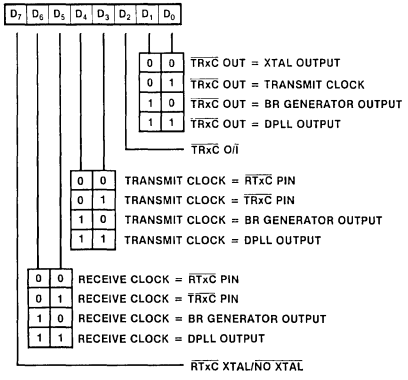
Write Register 9



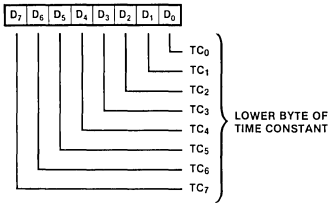
Write Register 10



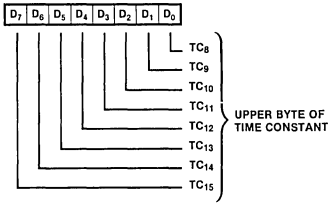
Write Register 11



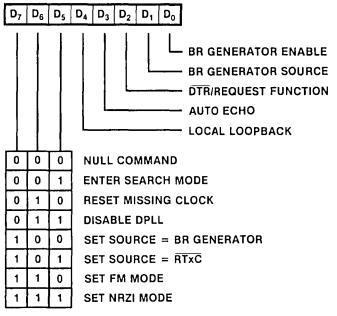
Write Register 12



Write Register 13



Write Register 14



Write Register 15

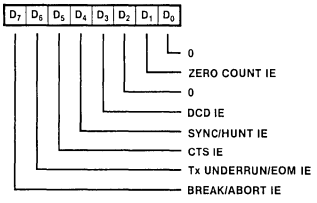


Figure 11. Write Register Bit Functions (Continued)

Timing

The Z-SCC generates internal control signals from $\overline{AS}$ and $\overline{DS}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{AS}$ and $\overline{DS}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the Z-SCC. The recovery time required for proper operation is specified from the rising edge of $\overline{DS}$ in the first transaction involving the Z-SCC to the falling edge of $\overline{DS}$ in the second transaction involving the Z-SCC. This time must be at least 6 PCLK cycles plus 200 ns.

Read Cycle Timing. Figure 12 illustrates read cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be High to indicate a Read cycle. CS_1 must also be High for the Read cycle to occur. The data bus drivers in the Z-SCC are then enabled while $\overline{DS}$ is Low.

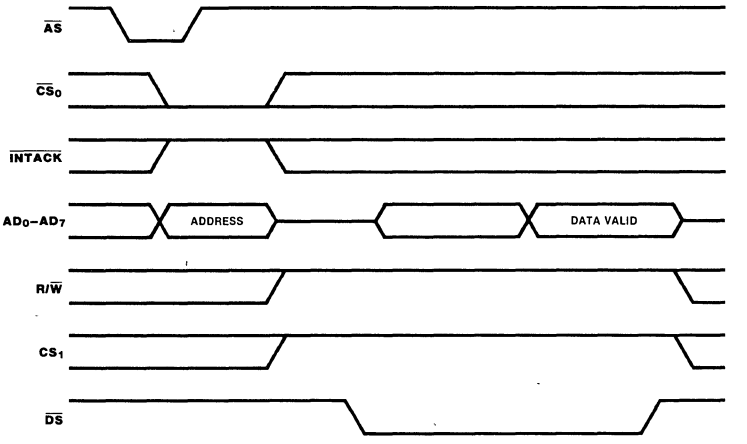


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be Low to indicate a Write cycle. CS_1 must be High for the Write cycle to occur. $\overline{DS}$ Low strobes the data into the Z-SCC.

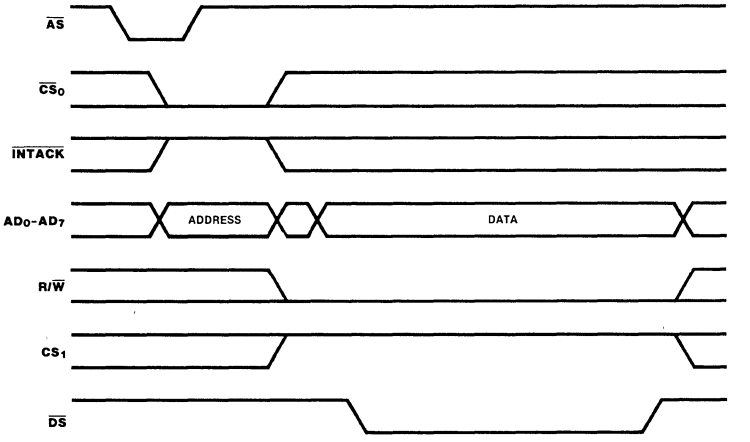


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. However, if $\overline{INTACK}$ is Low, the address and $\overline{CS_0}$ are ignored. The state of the R/$\overline{W}$ and CS_1 are also ignored for the duration of the Interrupt Acknowledge

Timing (Continued)

cycle. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the Z-SCC and IEI is High when $\overline{DS}$ falls, the Acknowledge cycle was intended for the Z-SCC. In this case, the Z-SCC may be programmed to respond to $\overline{DS}$ Low by placing its interrupt vector on AD_0–AD_7. It then sets the appropriate Interrupt-Under-Service latch internally.

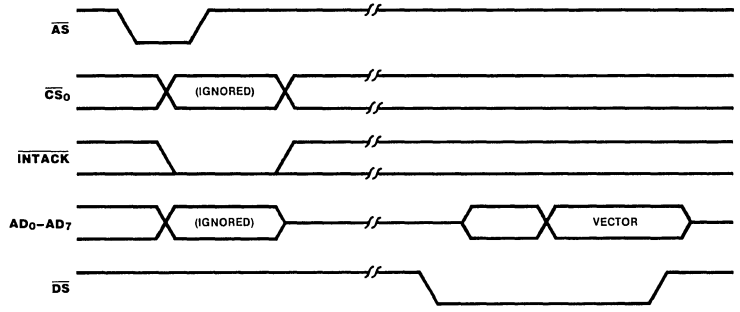


Figure 14. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in 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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \le V_{CC} \le +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

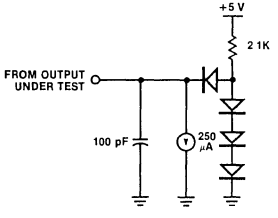


Figure 15. Standard Test Load

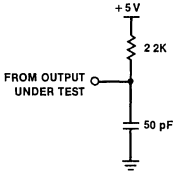


Figure 16. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \le V_{IN} \le +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \le V_{OUT} \le +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

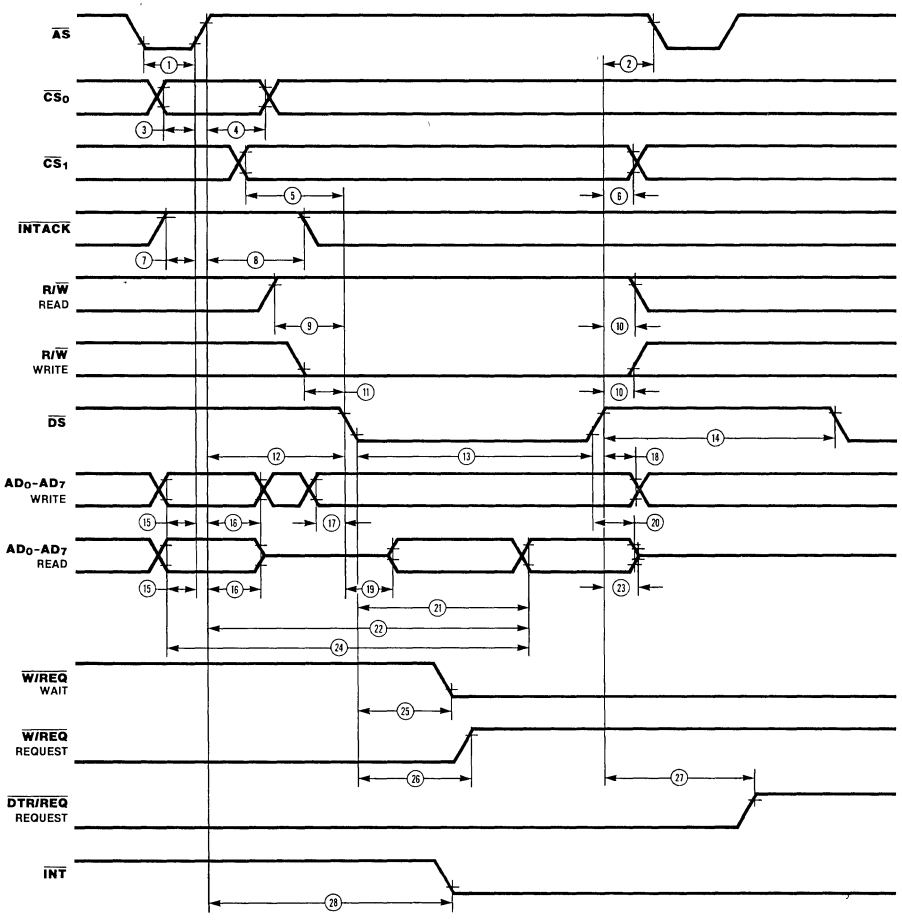
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range

Read and Write Timing

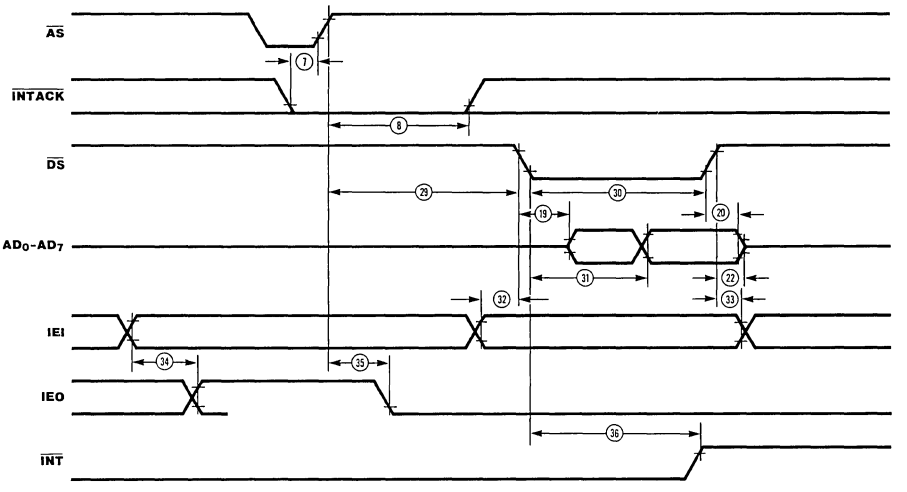


Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
1	TwAS	$\overline{AS}$ Low Width	70		
2	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		
3	TsCS0(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		1
4	ThCS0(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		1
5	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		1
6	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	60		1
7	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		
8	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		
10	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	60		
11	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		
13	TwDS1	$\overline{DS}$ Low Width	390		
14	TrC	Valid Access Recovery Time	6TcPC +200		2
15	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	10		1
16	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		1
17	TsDW(DS)	Write Data to $\overline{DS}$ ↓ Setup Time	30		
18	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	0		
19	TdDS(DA)	$\overline{DS}$ ↓ to Data Active Delay	0		
20	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		
21	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		255	
22	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		480	

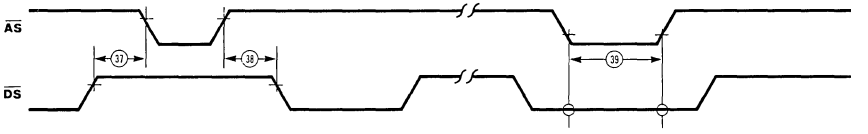
NOTES:
1 Parameter does not apply to Interrupt Acknowledge transactions.

2. Parameter applies only between transactions involving the Z-SCC.

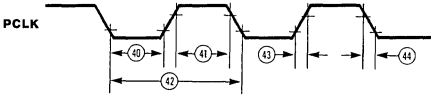
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing



Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
23	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70	3
24	TdA(DR)	Address Required Valid to Read Data Valid Delay		590	
25	TdDS(W)	$\overline{DS}$ ↓ to Wait Valid Delay		240	4
26	TdDSf(REQ)	$\overline{DS}$ ↓ to $\overline{W}/\overline{REQ}$ Not Valid Delay		240	
27	TdDSr(REQ)	$\overline{DS}$ ↓ to $\overline{DTR}/\overline{REQ}$ Not Valid Delay		5TcPC +300	
28	TdAS(INT)	$\overline{AS}$ ↑ to $\overline{INT}$ Valid Delay		500	4
29	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay			5
30	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		
31	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		360	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	120		
33	ThIEI(DSA)	IEI to $\overline{DS}$ ↑ (Acknowledge) Hold Time	0		
34	TdIEI(IEO)	IEI to IEO Delay		120	
35	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		250	6
36	TdDSA(INT)	$\overline{DS}$ ↓ (Acknowledge) to $\overline{INT}$ Inactive Delay		500	4
37	TdDS(ASQ)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay for No Reset	30		
38	TdASQ(DS)	$\overline{AS}$ ↑ to $\overline{DS}$↓ Delay for No Reset	30		
39	TwRES	$\overline{AS}$ and $\overline{DS}$ Coincident Low for Reset	250		7
40	TwPCl	PCLK Low Width	105	2000	
41	TwPCh	PCLK High Width	105	2000	
42	TcPC	PCLK Cycle Time	250	4000	
43	TrPC	PCLK Rise Time		20	
44	TfPC	PCLK Fall Time		20	

NOTES:

3. Float delay is defined as the time required for a ±0 5 V change in the output with a maximum dc load and minimum ac load.
4. Open-drain output, measured with open-drain test load
5. Parameter is system dependent For any Z-SCC in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the highest priority device in the daisy chain, TsIEI(DSA) for the Z-SCC, and TdIEIf(IEO) for each device separating them in the daisy chain
6. Parameter applies only to a Z-SCC pulling $\overline{INT}$ Low at the beginning of the Interrupt Acknowledge transaction
7. Internal circuitry allows for the reset provided by the Z8 to be recognized as a reset by the Z-SCC.

*Timings are preliminary and subject to change

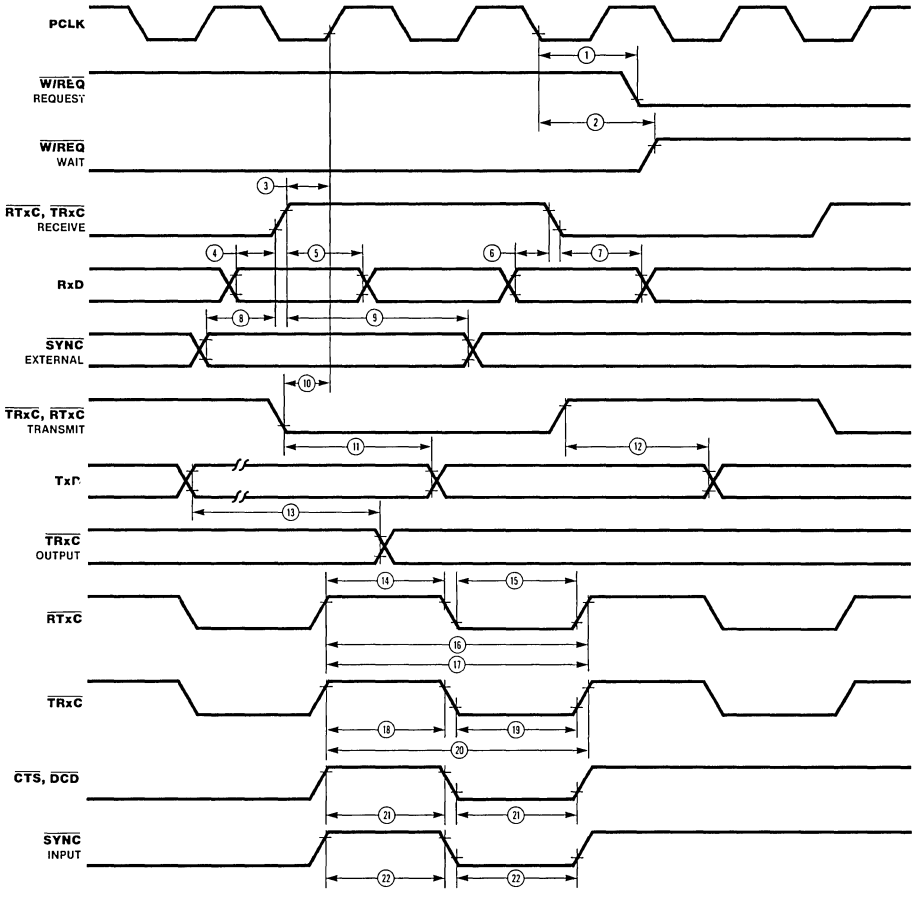
General Timing

Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
1	TdPC(REQ)	PCLK ↓ to W/$\overline{REQ}$ Valid Delay		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350	
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time	50		1,4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Setup Time (X1 Mode)	0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode)	150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode)	0		1,5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150		1,5
8	TsSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Setup Time	-200		1
9	ThSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Hold Time	3TcPC +200		1
10	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0		2,4
11	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)		300	2
12	TdTXCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode)		300	2,5
13	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)			
14	TwRTXh	$\overline{RTxC}$ High Width	180		
15	TwRTXl	$\overline{RTxC}$ Low Width	180		
16	TcRTX	$\overline{RTxC}$ Cycle Time	400		
17	TcRTXX	Crystal Oscillator Period	250	1000	3
18	TwTRXh	$\overline{TRxC}$ High Width	180		
19	TwTRXl	$\overline{TRxC}$ Low Width	180		
20	TcTRX	$\overline{TRxC}$ Cycle Time	400		
21	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ Pulse Width	200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200		

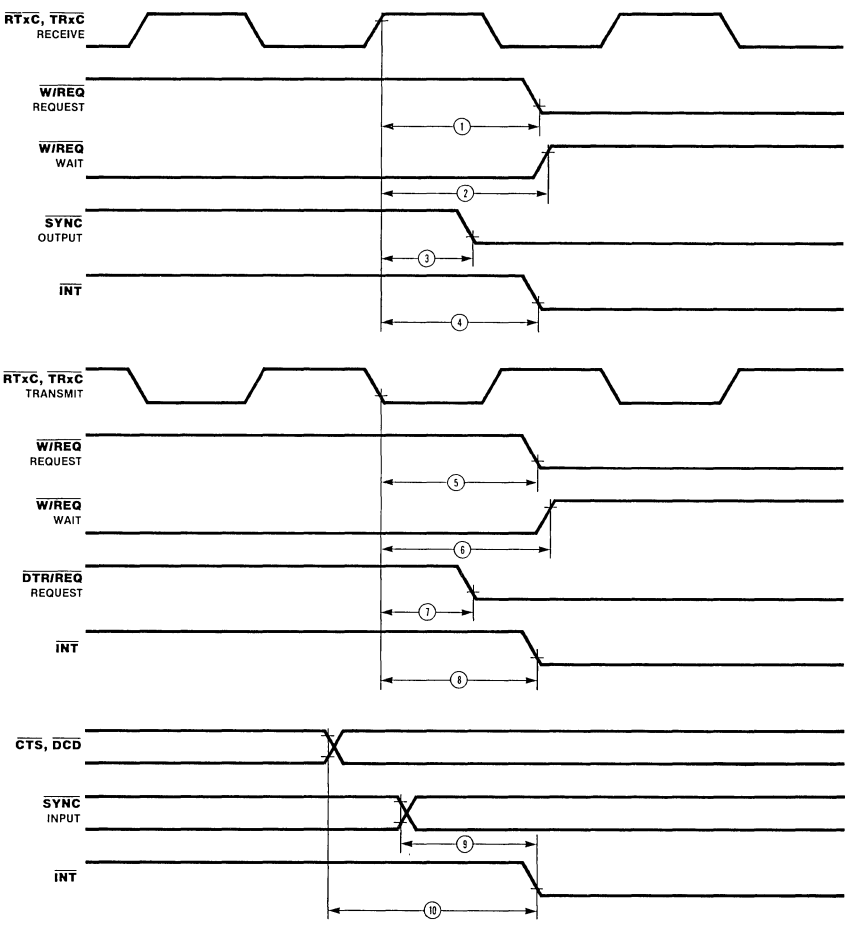
NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{SYNC}$ have 30 pF capacitors to the ground connected to them.
4. Parameter applies *only* if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.

*Timings are preliminary and subject to change

General Timing (Continued)



System Timing



Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdRXC (REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	TcPC	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	TcPC	1,2
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	TcPC	2
4	TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	8 +2	12 +3	TcPC ASt	1,2
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	TcPC	3
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	TcPC	1,3
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR/REQ}$ Valid Delay	4	7	TcPC	3
8	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	4 +2	6 +3	TcPC ASt	1,3
9	TdSY(INT)	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	3	ASt	1
10	TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	3	ASt	1

NOTES.

1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.

*Timings are preliminary and subject to change.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8030	CE	4.0 MHz	Z-SCC (40-pin)	Z8030A	CE	6.0 MHz	Z-SCC (40-pin)
Z8030	CS	4.0 MHz	Same as above	Z8030A	CS	6.0 MHz	Same as above
Z8030	DE	4.0 MHz	Same as above	Z8030A	DE	6.0 MHz	Same as above
Z8030	DS	4.0 MHz	Same as above	Z8030A	DS	6.0 MHz	Same as above
Z8030	PE	4.0 MHz	Same as above	Z8030A	PE	6.0 MHz	Same as above
Z8030	PS	4.0 MHz	Same as above	Z8030A	PS	6.0 MHz	Same as above

NOTES. C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, S = 0°C to +70°C.

Z8036 Z8000™ Z-CIO Counter/Timer and Parallel I/O Unit

Zilog

Product Specification

March 1981

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write and directly addressable.

General Description

The Z8036 Z-CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications.

The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique address so that it can be accessed directly—no special sequential operations are required. The Z-CIO is directly Z-Bus compatible.

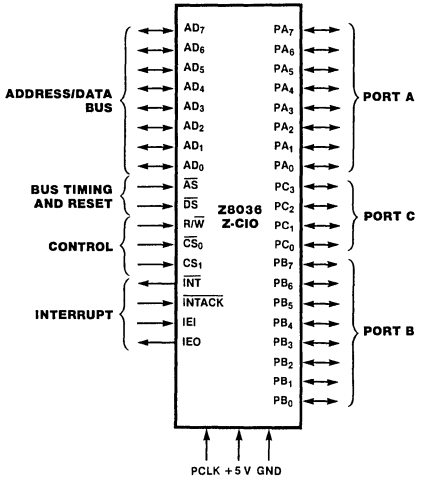


Figure 1. Pin Functions

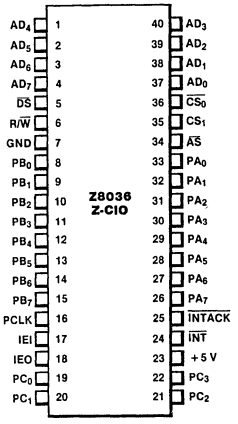


Figure 2. Pin Assignments

Pin Description

AD_0–AD_7. *Z-Bus Address/Data lines* (bidirectional/3-state). These multiplexed Address/Data lines are used for transfers between the CPU and Z-CIO.

$\overline{AS}$*. *Address Strobe* (input, active Low). Addresses, $\overline{INTACK}$, and $\overline{CS_0}$ are sampled while $\overline{AS}$ is Low.

$\overline{CS_0}$ and CS_1. *Chip Select 0* (input, active Low) *and Chip Select 1* (input, active High). $\overline{CS_0}$ and CS_1 must be Low and High, respectively, in order to select a device. $\overline{CS_0}$ is latched by $\overline{AS}$.

$\overline{DS}$*. *Data Strobe* (input, active Low). $\overline{DS}$ provides timing for the transfer of data into or out of the Z-CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting Z-CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

*When $\overline{AS}$ and $\overline{DS}$ are detected Low at the same time (normally an illegal condition), the Z-CIO is reset

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the Z-CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates to the Z-CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ is sampled while $\overline{AS}$ is Low.

PA_0–PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port A and external devices.

PB_0–PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0–PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the Z-CIO's Port C.

PCLK. *(input, TTL-compatible).* This is a peripheral clock that may be, but is not necessarily, the CPU clock. It is used with timers and REQUEST/$\overline{WAIT}$ logic. Maximum input frequency is 4 MHz.

R/$\overline{W}$. *Read/Write* (input). R/$\overline{W}$ indicates that the CPU is reading from (High) or writing to (Low) the Z-CIO.

Architecture

The Z8036 Z-CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a Z-Bus interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose

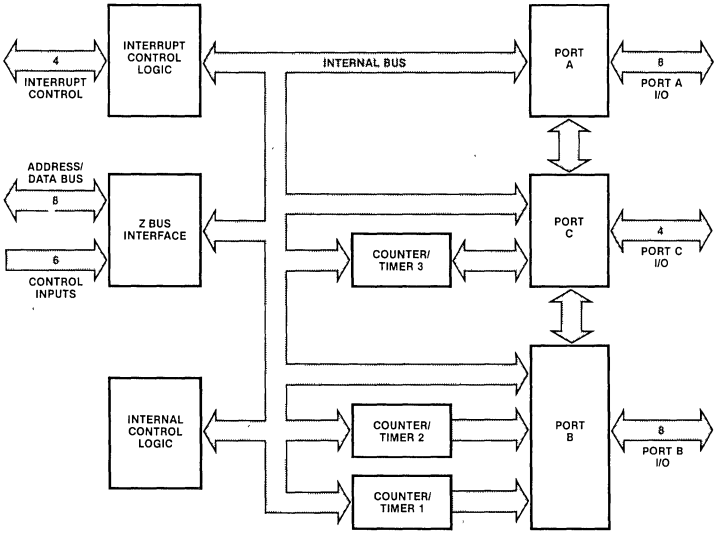


Figure 3. Z-CIO Block Diagram

Architecture (Continued)

4-bit port), three 16-bit counter/timers, an interrupt control logic block, and the internal control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

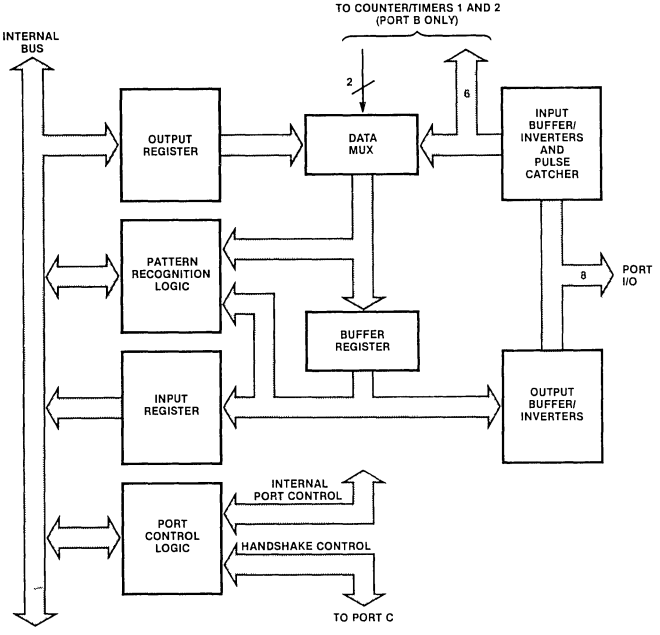


Figure 4. Ports A and B Block Diagram

Architecture (Continued)

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Counter register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

The interrupt control logic provides standard Z-Bus interrupt capabilities. There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

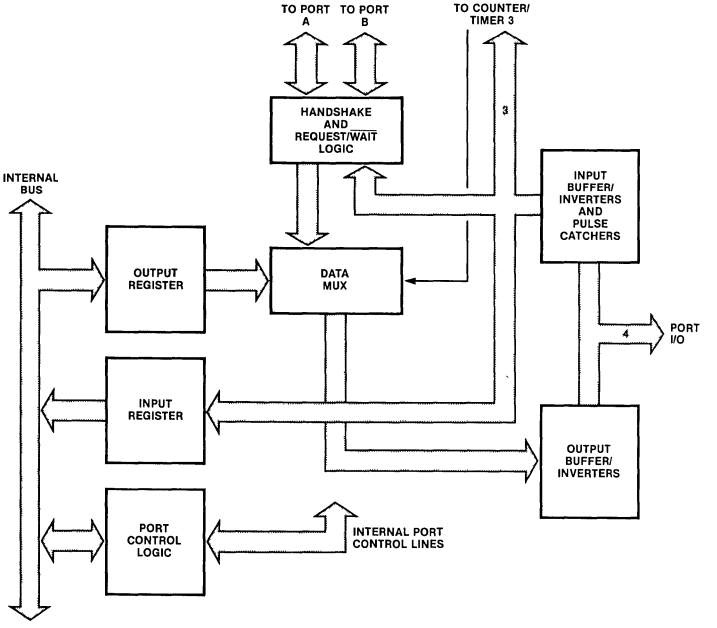


Figure 5. Port C Block Diagram

Architecture
(Continued)

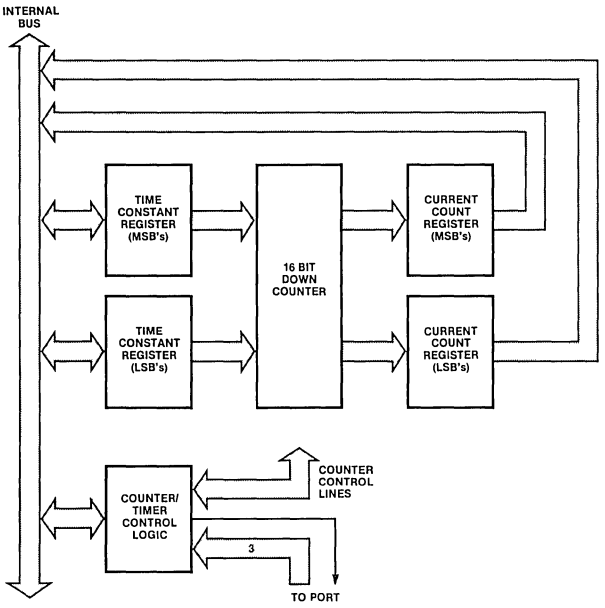


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the Z-CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that Z-CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to a 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The Z-CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the Z-CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the Z-CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another Z-CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$

Table 1. Port C Bit Utilization

Functional Description (Continued)

allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$ goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked Handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

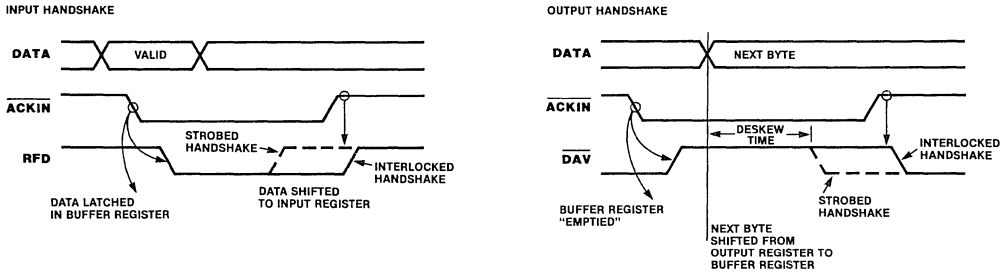


Figure 7. Interlocked and Strobed Handshakes

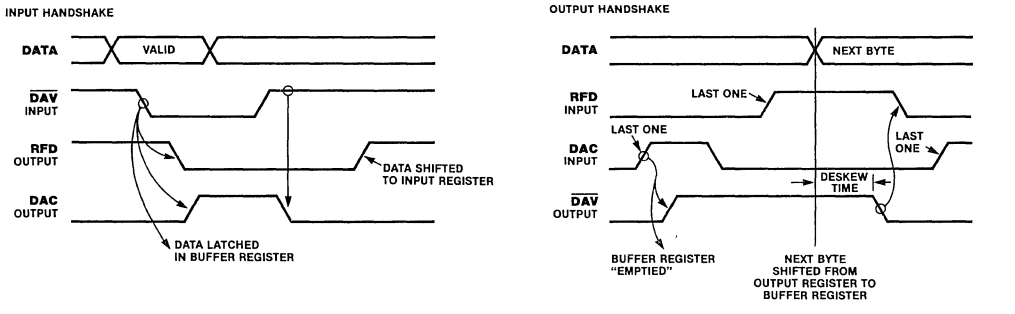


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the Z-Bus. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the Z-CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any non-masked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

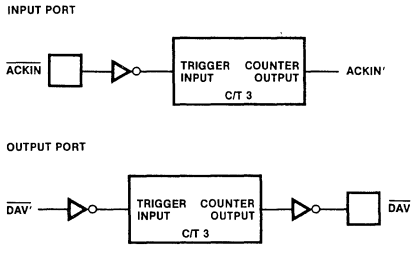


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: Transparent and Latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1, after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the Acknowledge cycle is initiated, the vector is

Functional Description (Continued)

frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern-match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type applications when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/

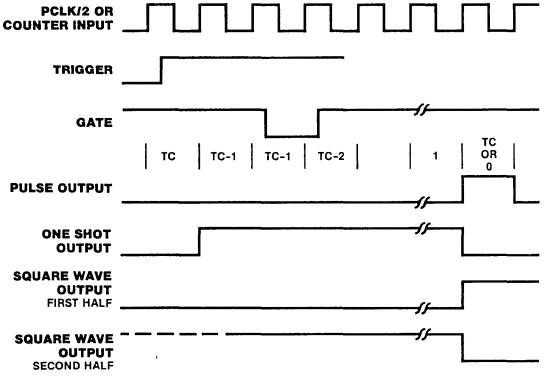


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

timer waveforms. When the Pulse mode is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/Single Cycle (C/$\overline{SC}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{SC}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{SC}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to a 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The interrupts generated by the Z-CIO follow the Z-Bus operation as described more fully in the Zilog *Z-Bus Summary*. The Z-CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The Z-CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device,

Functional Description (Continued)

the Z-CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the Z-CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS) status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, an interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the Z-CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the Z-CIO's IEO is forced Low, independent of the state of the Z-CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the Z-CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The Z-CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the Z-CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

Programming the Z-CIO entails loading control registers with bits to implement the desired operation. Individual enable bits are provided for the various major blocks so that erroneous operations do not occur while the part is being initialized. Before the ports are enabled, IPs cannot be set, REQUEST and $\overline{WAIT}$ cannot be asserted, and all outputs remain high-impedance. The handshake lines are ignored until Port C is enabled. The counter/timers cannot be triggered until their enable bits are set.

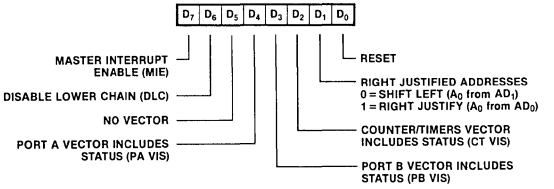
The Z-CIO is reset by forcing $\overline{AS}$ and $\overline{DS}$ Low simultaneously or by writing a 1 to the Reset bit. Once reset, the only thing that can be done is to read and write the Reset bit. Writes to all other bits are ignored and all reads return 0s. In this state, all control bits are forced to 0. Only after clearing the Reset bit (by writing to it) can the other command bits be programmed.

Register Addressing. The Z-CIO allows two schemes for register addressing. Both schemes use only six of the eight bits of the address/data bus. The scheme used is determined by the Right Justify Address (RJA) bit in the Master Interrupt Control register. When RJA equals 0, address bus bits 0 and 7 are ignored, and bits 1 through 6 are decoded for the register address (A_0 from AD_1). When RJA equals 1, bits 0 through 5 are decoded for the register address (A_0 from AD_0). In the following register descriptions, only six bits are shown for addresses and represent address/data bus bits 0 through 5 or 1 through 6, depending on the state of the RJA bit.

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

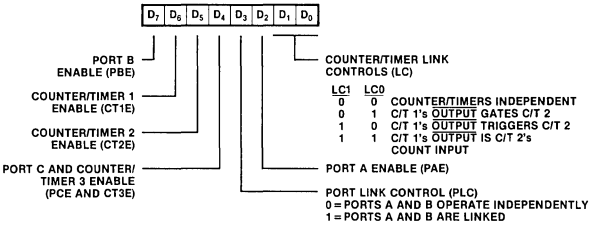
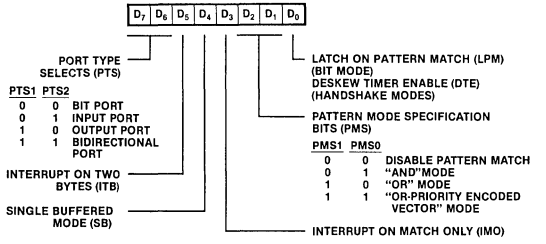


Figure 11. Master Control Registers

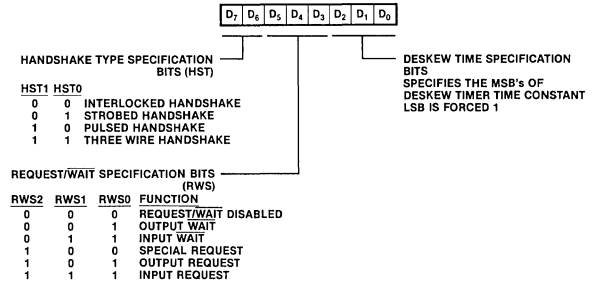
Port Mode Specification Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)



Port Handshake Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)

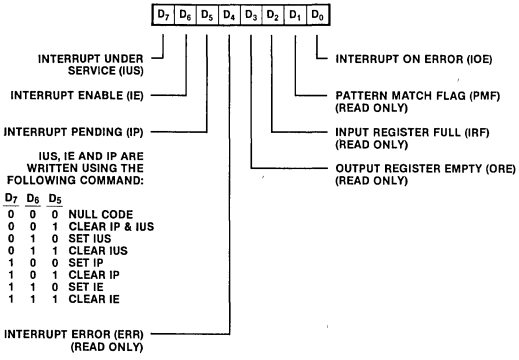
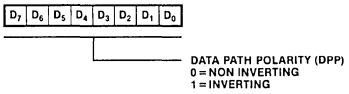


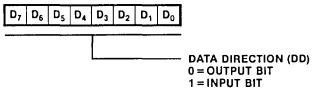
Figure 12. Port Specification Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

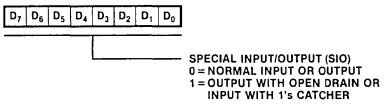
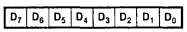


Figure 13. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A
001110 Port B
(Read/Write)



Port C Data Register
Address: 001111
(Read/Write)

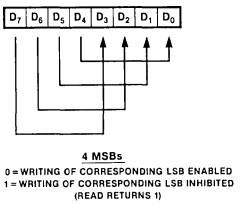
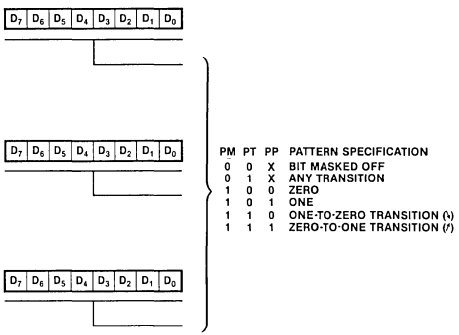


Figure 14. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)

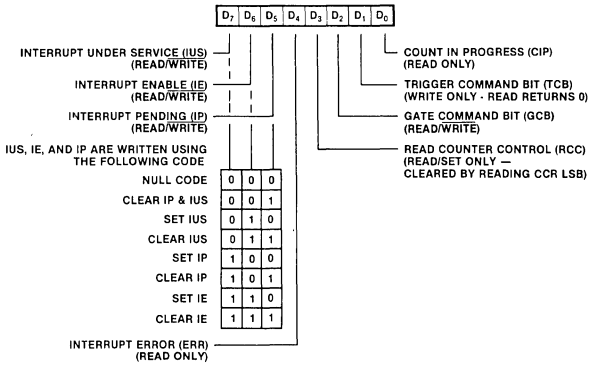


PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE-TO-ZERO TRANSITION (↓)
1	1	1	ZERO-TO-ONE TRANSITION (↑)

Figure 15. Pattern Definition Registers

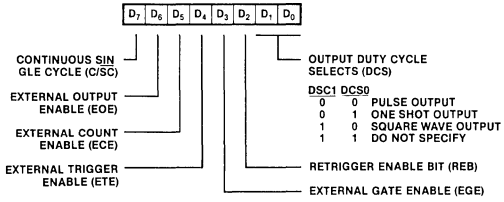
Counter/Timer Command and Status Registers

Addresses:. 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



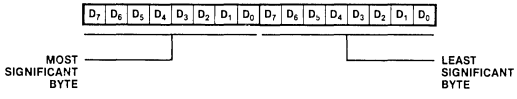
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

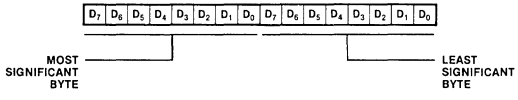
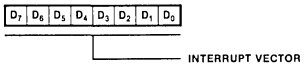


Figure 16. Counter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses. 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



PORT VECTOR STATUS

PRIORITY ENCODED VECTOR MODE

D3	D2	D1	
x	x	x	NUMBER OF HIGHEST PRIORITY BIT WITH A MATCH

ALL OTHER MODES

D3	D2	D1		
ORE	IRF	PMF	NORMAL	
0	0	0	ERROR	

COUNTER/TIMER STATUS

D2	D1	
0	0	C/T 3
0	1	C/T 2
1	0	C/T 1
1	1	ERROR

Current Vector Regis
Address: 011111
(Read Only)

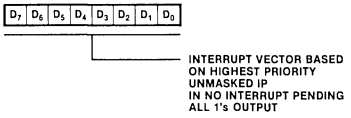


Figure 17. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address (AD_7-AD_0)	Register Name
000000XX	Master Interrupt Control
000001XX	Master Configuration Control
000010XX	Port A's Interrupt Vector
000011XX	Port B's Interrupt Vector
000100XX	Counter/Timer's Interrupt Vector
000101XX	Port C's Data Path Polarity
000110XX	Port C's Data Direction
000111XX	Port C's Special I/O Control

Most Often Accessed Registers

Address (AD_7-AD_0)	Register Name
001000XX	Port A's Command and Status
001001XX	Port B's Command and Status
001010XX	Counter/Timer 1's Control
001011XX	Counter/Timer 2's Control
001100XX	Counter/Timer 3's Control
001101XX	Port A's Data
001110XX	Port B's Data
001111XX	Port C's Data

Counter/Timer Related Registers

Address (AD_7-AD_0)	Register Name
010000XX	Counter/Timer 1's Current Count-MSBs
010001XX	Counter/Timer 1's Current Count-LSBs
010010XX	Counter/Timer 2's Current Count-MSBs
010011XX	Counter/Timer 2's Current Count-LSBs
010100XX	Counter/Timer 3's Current Count-MSBs
010101XX	Counter/Timer 3's Current Count-LSBs
010110XX	Counter/Timer 1's Time Constant-MSBs
010111XX	Counter/Timer 1's Time Constant-LSBs

Counter/Timer Related Registers (Continued)

Address (AD_7-AD_0)	Register Name
011000XX	Counter/Timer 2's Time Constant-MSBs
011001XX	Counter/Timer 2's Time Constant-LSBs
011010XX	Counter/Timer 3's Time Constant-MSBs
011011XX	Counter/Timer 3's Time Constant-LSBs
011100XX	Counter/Timer 1's Mode Specification
011101XX	Counter/Timer 2's Mode Specification
011110XX	Counter/Timer 3's Mode Specification
011111XX	Current Vector

Port A Specification Registers

Address (AD_7-AD_0)	Register Name
100000XX	Port A's Mode Specification
100001XX	Port A's Handshake Specification
100010XX	Port A's Data Path Polarity
100011XX	Port A's Data Direction
100100XX	Port A's Special I/O Control
100101XX	Port A's Pattern Polarity
100110XX	Port A's Pattern Transition
100111XX	Port A's Pattern Mask

Port B Specification Registers

Address (AD_7-AD_0)	Register Name
101000XX	Port B's Mode Specification
101001XX	Port B's Handshake Specification
101010XX	Port B's Data Path Polarity
101011XX	Port B's Data Direction
101100XX	Port B's Special I/O Control
101101XX	Port B's Pattern Polarity
101110XX	Port B's Pattern Transition
101111XX	Port B's Pattern Mask

Timing

Read Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The data from the register specified is strobed onto the address/data bus when the CPU issues a Data Strobe ($\overline{DS}$). If the register indicated by the address does not exist, the Z-CIO remains high-impedance.

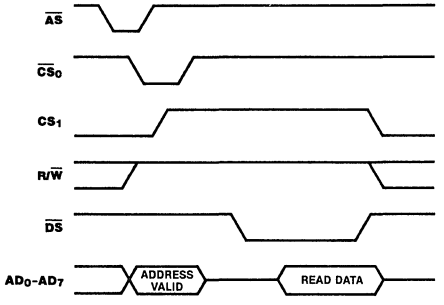


Figure 18. Read Cycle Timing

Write Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The CPU places the data on the address/data bus and strobes it into the Z-CIO register by issuing a Data Strobe ($\overline{DS}$).

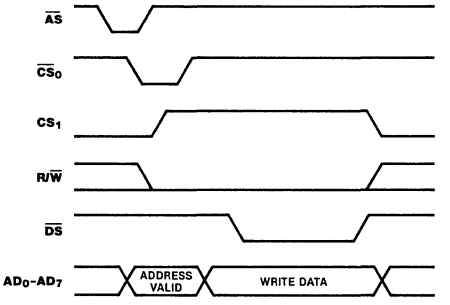
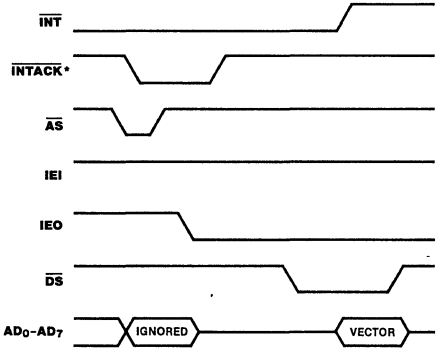


Figure 19. Write Cycle Timing

Interrupt Acknowledge Cycle. When one of the IP bits in the Z-CIO goes High and interrupts are enabled, the Z-CIO pulls its $\overline{INT}$ output line Low, requesting an interrupt. The CPU responds with an Interrupt Acknowledge cycle. When $\overline{INTACK}$ goes Low with IP set, the Z-CIO pulls its Interrupt Enable Out (IEO) Low, disabling all lower priority devices on the daisy chain. The CPU reads the Z-CIO interrupt vector by issuing a Low $\overline{DS}$, thereby strobing the interrupt vector onto the address/data bus. The IUS that corresponds to the IP is also set, which causes IEO to remain Low.



*$\overline{INTACK}$ is decoded from Z8000 status.

Figure 20. Interrupt Acknowledge Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in 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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

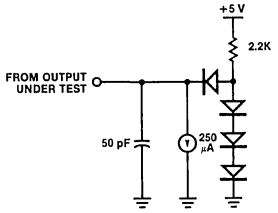


Figure 21. Standard Test Load

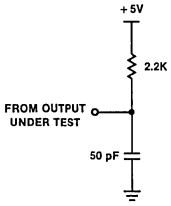


Figure 22. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		250	mA	

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

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

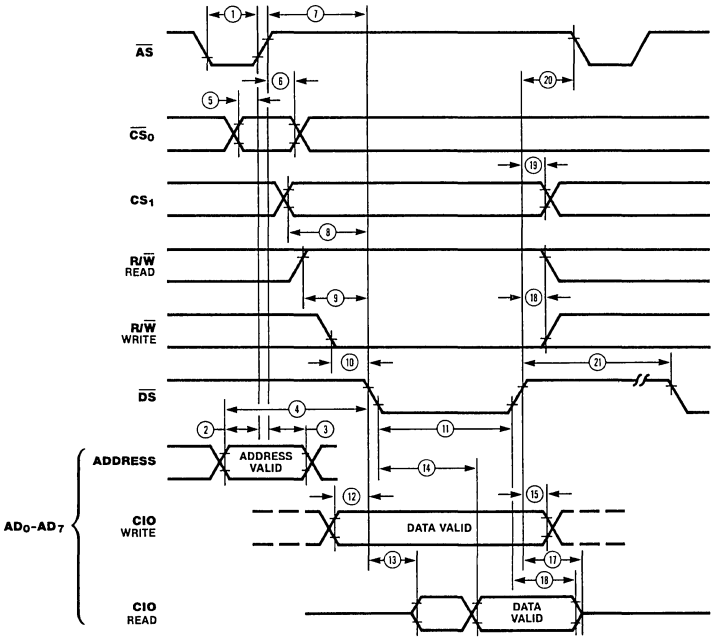
f = 1 MHz, over specified temperature range.

	Number	Symbol	Parameter	Min	Max	Units	Notes*
CPU Interface Timing	1	TwAS	$\overline{AS}$ Low Width	70	2000	ns	
	2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	10		ns	1
	3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		ns	1
	4	TsA(DS)	Address to $\overline{DS}$ ↓ Setup Time	120		ns	1
	5	TsCS0(AS)	$\overline{CS}_0$ to $\overline{AS}$ ↑ Setup Time	0		ns	1
	6	ThCS0(AS)	$\overline{CS}_0$ to $\overline{AS}$ ↑ Hold Time	60		ns	1
	7	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		ns	1
	8	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		ns	
	9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		ns	
	10	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0			
	11	TwDS	$\overline{DS}$ Low Width	390		ns	
	12	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		ns	
	13	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0			
	14	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		255	ns	
	15	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		ns	
	16	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0			
	17	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70	ns	2
	18	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	60		ns	
	19	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	60		ns	
	20	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		ns	
	21	Trc	Valid Access Recovery Time	1000		ns	3
Interrupt Timing	22	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		1	$\overline{AS}$ cycle + ns	
	23	TdACK(INT)	$\overline{ACKIN}$ to $\overline{INT}$ Delay (Port with Handshake)		4	$\overline{AS}$ cycle + ns	4
	24	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		1	$\overline{AS}$ cycle + ns	
	25	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		1	$\overline{AS}$ cycle + ns	
	26	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay			ns	
Interrupt Acknowledge Timing	27	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		ns	
	28	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		ns	
	29	TsAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Setup Time	350		ns	5
	30	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		360	ns	
	31	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		ns	
	32	TdAS(IEO)	$\overline{AS}$ ↓ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		350	ns	5
	33	TdIEI(IEO)	IEI to IEO Delay		150	ns	5
	34	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		ns	5
	35	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↑ Hold Time	100		ns	
	36	TdDSA(INT)	$\overline{DS}$ (Acknowledge) ↓ to $\overline{INT}$ ↑ Delay		600	ns	

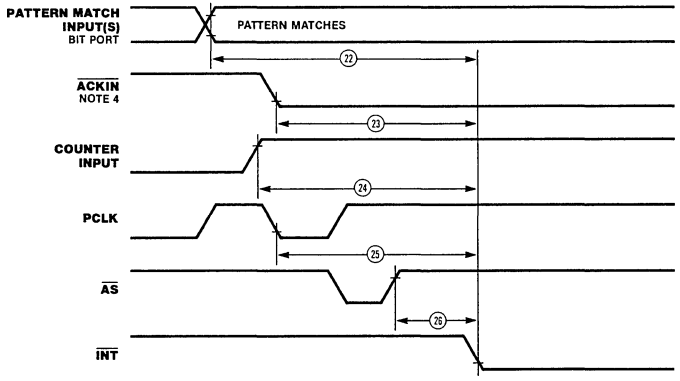
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{DS}$ ↑ of one CIO access to $\overline{DS}$ ↓ of another CIO access.
4. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake. One additional $\overline{AS}$ cycle is required for ports in the Single Buffered mode.
5. The parameters for the devices in any particular daisy chain must meet the following constraint: the delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.

*Timings are preliminary and subject to change

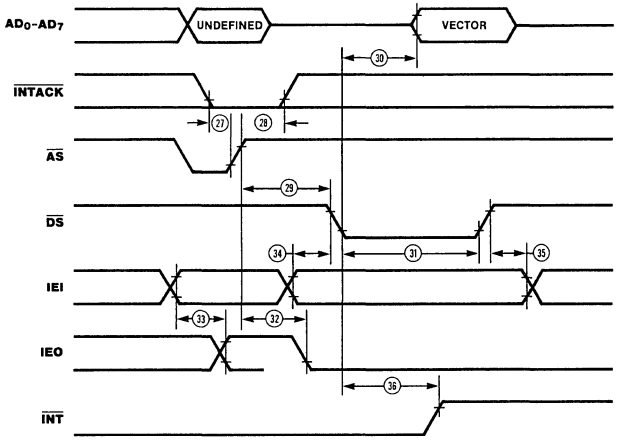
CPU Interface Timing



Interrupt Timing



Interrupt Acknowledge Timing

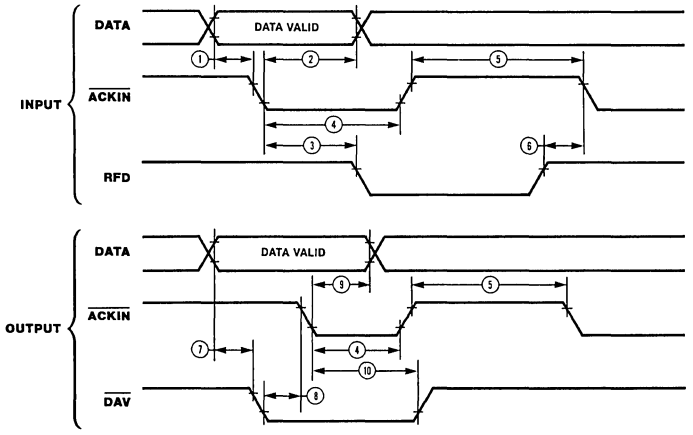


Handshake Timing

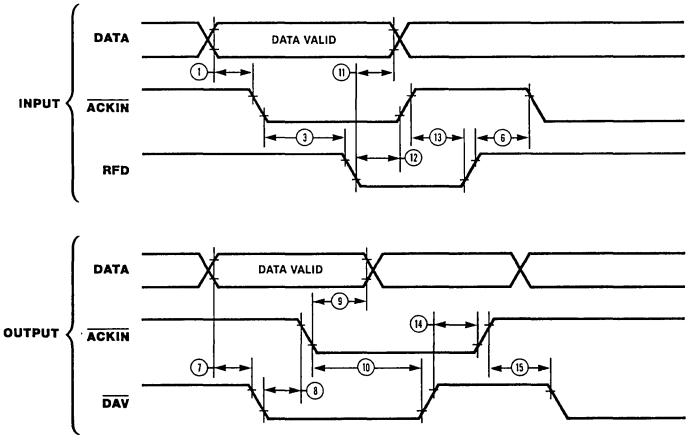
Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Setup Time	0		ns	
2	ThDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Hold Time—Strobed Handshake			ns	
3	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		ns	
4	TwACKl	$\overline{ACKIN}$ Low Width—Strobed Handshake			ns	
5	TwACKh	$\overline{ACKIN}$ High Width—Strobed Handshake			ns	
6	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		ns	
7	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		ns	1
8	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		ns	
9	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↓ Hold Time	2		TcPC	
10	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	2		TcPC	
11	ThDI(RFD)	Data Input to RFD ↓ Hold Time—Interlocked Handshake	0		ns	
12	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay—Interlocked Handshake	0		ns	
13	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handshake	0		ns	
14	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)—Interlocked and 3-Wire Handshake	0		ns	
15	TdACK(DAV)	$\overline{ACKIN}$ ↑ (RFD ↑) to $\overline{DAV}$ ↓ Delay—Interlocked and 3-Wire Handshake	0		ns	
16	TdDAVIf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Input 3-Wire Handshake	0		ns	
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time—3-Wire Handshake	0		ns	
18	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Input 3-Wire Handshake	0		ns	
19	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Input 3-Wire Handshake	0		ns	
20	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Output 3-Wire Handshake	0		ns	
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time—3-Wire Handshake	2		TcPC	
22	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Output 3-Wire Handshake	2		TcPC	
23	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Output 3-Wire Handshake	0		ns	

NOTES:
1. This time can be extended through the use of the deskew timers.
*Timings are preliminary and subject to change.

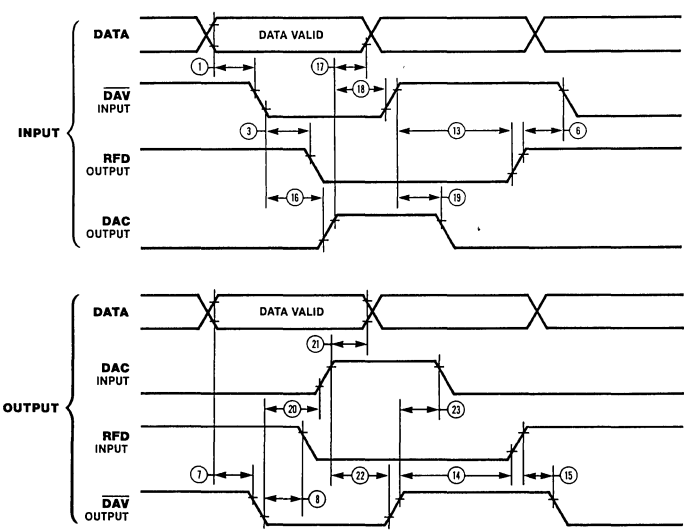
Strobed Handshake



Interlocked Handshake



3-Wire Handshake



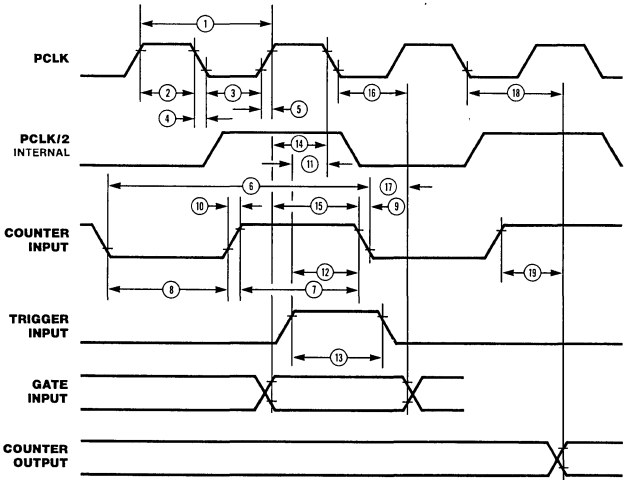
Counter/ Timer Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TcPC	PCLK Cycle Time	250	4000	ns	1
2	TwPCh	PCLK High Width	105	2000	ns	
3	TwPCl	PCLK Low Width	105	2000	ns	
4	TfPC	PCLK Fall Time		20	ns	
5	TrPC	PCLK Rise Time		20	ns	
6	TcCI	Counter Input Cycle Time	500		ns	
7	TCIh	Counter Input High Width	230		ns	
8	TwCIl	Counter Input Low Width	230		ns	
9	TfCI	Counter Input Fall Time		20	ns	
10	TrCI	Counter Input Rise Time		20	ns	
11	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)			ns	2
12	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)			ns	2
13	TwTI	Trigger Input Pulse Width (High or Low)			ns	
14	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)			ns	2
15	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)			ns	2
16	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)			ns	2
17	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)			ns	2
18	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)			ns	
19	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)			ns	

NOTES:
1 PCLK is only used with the counter/timers (in Timer mode), the deskew timers, and the REQUEST/WAIT logic If these functions are not used, the PCLK input can be held Low.

2. These parameters must be met to guarantee trigger or gate are valid for the next counter/timer cycle.

*Timings are preliminary and subject to change.

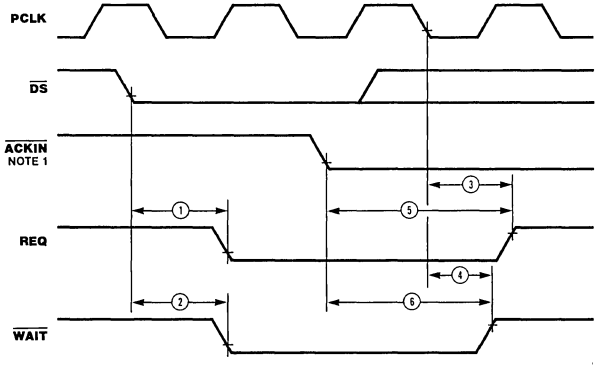


REQUEST/ WAIT Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdDS(REQ)	$\overline{DS}$ ↓ to REQ ↓ Delay			ns	
2	TdDS(WAIT)	$\overline{DS}$ ↓ to $\overline{WAIT}$ ↓ Delay			ns	
3	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay			ns	
4	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay			ns	
5	TdACK(REQ)	$\overline{ACKIN}$ ↓ to REQ ↑ Delay			$\overline{AS}$ cycles + PCLK cycles + ns	1
6	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay			PCLK cycles + ns	1

NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.

*Timings are preliminary and subject to change.

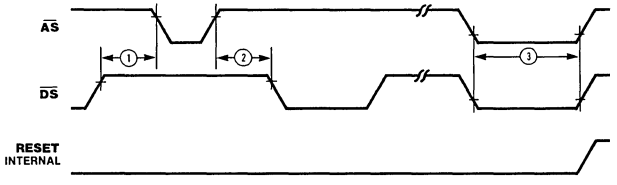


Reset Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		ns	
2	TdASQ(DS)	Delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		ns	
3	TwRES	Minimum Width of $\overline{AS}$ and $\overline{DS}$ both Low for Reset	250		ns	1

NOTES:
1. Internal circuitry allows for the reset provided by the Z8 ($\overline{DS}$ held Low while $\overline{AS}$ pulses) to be sufficient.

*Timings are preliminary and subject to change.

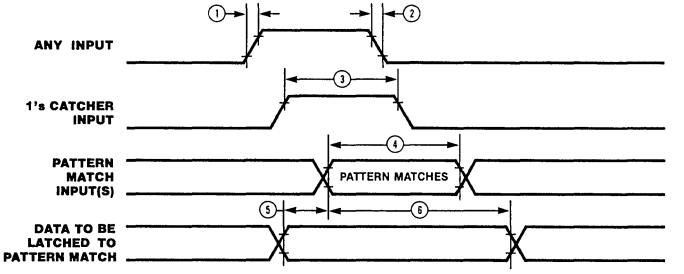


Miscellaneous Port Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TrI	Any Input Rise Time		100	ns	
2	TfI	Any Input Fall Time		100	ns	
3	Twl's	1's Catcher High Width	250		ns	1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		ns	
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		ns	
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		ns	

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

*Timings are preliminary and subject to change.



Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8036	CE	4.0 MHz	Z-CIO (40-pin)	Z8036A	CE	6.0 MHz	Z-CIO (40-pin)
Z8036	CS	4.0 MHz	Same as above	Z8036A	CS	6.0 MHz	Same as above
Z8036	DE	4.0 MHz	Same as above	Z8036A	DE	6.0 MHz	Same as above
Z8036	DS	4.0 MHz	Same as above	Z8036A	DS	6.0 MHz	Same as above
Z8036	PE	4.0 MHz	Same as above	Z8036A	PE	6.0 MHz	Same as above
Z8036	PS	4.0 MHz	Same as above	Z8036A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, S = 0°C to +70°C.

Z8038 Z8000™ Z-FIO FIFO Input/ Output Interface Unit

Zilog

Product Specification

March 1981

Z8038 Z-BUS™ Version FIO
Z8538 Universal Version FIO

Features

- 128-byte FIFO buffer provides asynchronous bidirectional CPU/CPU or CPU/peripheral interface, expandable to any width in byte increments by use of multiple FIOs.
- Interlocked 2-Wire or 3-Wire Handshake logic port mode; Z-BUS or non-Z-BUS interface.
- Pattern-recognition logic stops DMA transfers and/or interrupts CPU; preset byte count can initiate variable-length DMA transfers.
- Seven sources of vectored/nonvectored interrupt which include pattern-match, byte count, empty or full buffer status; a dedicated "mailbox" register with interrupt capability provides CPU/CPU communication.
- $\overline{\text{REQUEST}}/\overline{\text{WAIT}}$ lines control high-speed data transfers.
- All functions are software controlled via directly addressable read/write registers.

General Description

The Z8038 FIO provides an asynchronous 128-byte FIFO buffer between two CPUs or between a CPU and a peripheral device. This buffer interface expands to a 16-bit or wider data path and expands in depth to add as many Z8060 FIFOs (and an additional FIO) as are needed.

The FIO manages data transfers by assuming Z-BUS, non-Z-BUS microprocessor (a generalized microprocessor interface), Interlocked 2-Wire Handshake, and 3-Wire Handshake operating modes. These modes interface dissimilar CPUs or CPUs and peripherals running under differing speeds or protocols, allowing asynchronous data transactions and improving I/O overhead by as much as two orders of magnitude. Figures 1 and 2 show how the signals controlling these operating modes are mapped to the FIO pins.

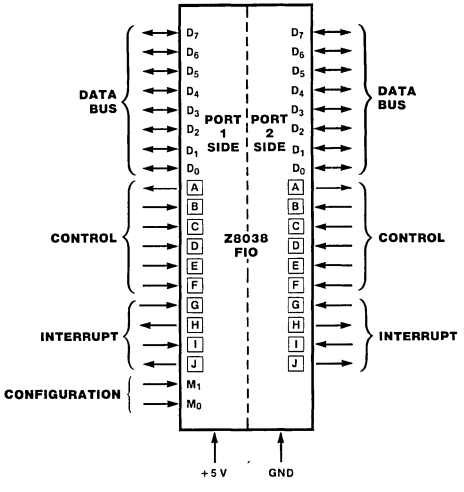


Figure 1. Pin Functions

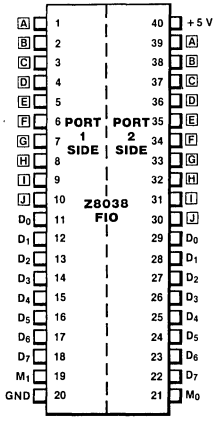


Figure 2. Pin Assignments

General Description (Continued)

The FIO supports the Z-BUS interrupt protocols, generating seven sources of interrupts upon any of the following events: a write to a message register, change in data direction, pattern match, status match, over/underflow error, buffer full and buffer empty status. Each interrupt source can be enabled or disabled, and can also place an interrupt vector on the port address/data lines.

The data transfer logic of the FIO has been specially designed to work with DMA (Direct Memory Access) devices for high-speed transfers. It provides for data transfers to or from memory each machine cycle, while the DMA device generates memory address and control signals. The FIO also supports the variably sized block length, improving system throughput when multiple variable length messages are transferred amongst several sources.

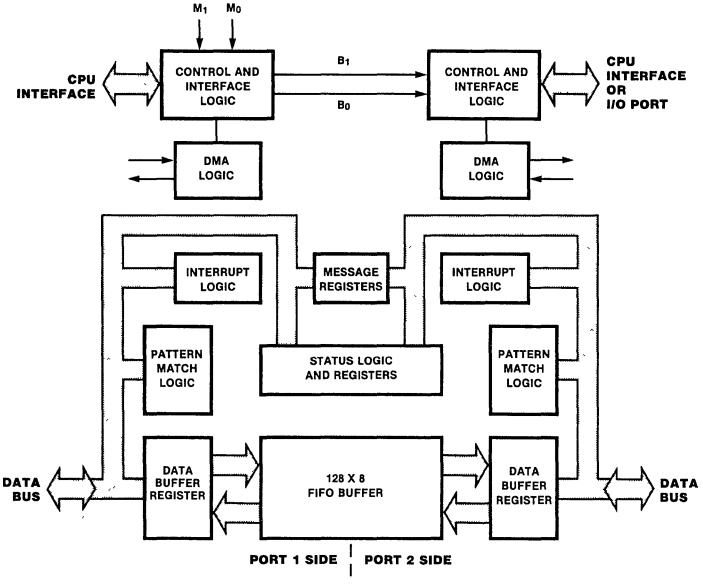


Figure 3. FIO Block Diagram

Functional Description

Operating Modes. Ports 1 and 2 operate in any of twelve combinations of operating modes, listed in Table 2. Port 1 functions in either the Z-BUS or non-Z-BUS microprocessor modes, while Port 2 functions in Z-BUS, non-Z-BUS, Interlocked 2-Wire Handshake, and 3-Wire Handshake modes. Table 1 describes the signals and their corresponding pins in each of these modes.

The pin diagrams of the FIO are identical, except for two pins on the Port 1 side, which select that port's operating mode. Port 2's operating mode is programmed by two bits in Port 1's Control register 0. Table 2 describes the combinations of operating modes; Table 3 describes the control signals mapped to pins A-J in the five possible operating modes.

Control Signal Pins	Z-BUS Low Byte	Z-BUS High Byte	Non-Z-BUS	Interlocked HS Port*	3-Wire HS Port*
A	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	RFD/$\overline{DAV}$	RFD/$\overline{DAV}$
B	$\overline{DMASTB}$	$\overline{DMASTB}$	$\overline{DACK}$	$\overline{ACKIN}$	$\overline{DAV}$/DAC
C	$\overline{DS}$	$\overline{DS}$	$\overline{RD}$	FULL	DAC/RFD
D	R/$\overline{W}$	R/$\overline{W}$	$\overline{WR}$	EMPTY	EMPTY
E	$\overline{CS}$	$\overline{CS}$	$\overline{CE}$	$\overline{CLEAR}$	$\overline{CLEAR}$
F	$\overline{AS}$	AS	C/$\overline{D}$	DATA DIR	DATA DIR
G	$\overline{INTACK}$	A_0	$\overline{INTACK}$	IN_0	IN_0
H	IEO	A_1	IEO	OUT_1	OUT_1
I	IEI	A_2	IEI	$\overline{OE}$	$\overline{OE}$
J	$\overline{INT}$	A_3	$\overline{INT}$	OUT_3	OUT_3

*2 side only.

Table 1. Pin Assignments

Functional Description (Continued)

Mode	M_1	M_0	B_1	B_0	Port 1	Port 2
0	0	0	0	0	Z-BUS Low Byte	Z-BUS Low Byte
1	0	0	0	1	Z-BUS Low Byte	Non-Z-BUS
2	0	0	1	0	Z-BUS Low Byte	3-Wire Handshake
3	0	0	1	1	Z-BUS Low Byte	2-Wire Handshake
4	0	1	0	0	Z-BUS High Byte	Z-BUS High Byte
5	0	1	0	1	Z-BUS High Byte	Non-Z-BUS
6	0	1	1	0	Z-BUS High Byte	3-Wire Handshake
7	0	1	1	1	Z-BUS High Byte	2-Wire Handshake
8	1	0	0	0	Non-Z-BUS	Z-BUS Low Byte
9	1	0	0	1	Non-Z-BUS	Non-Z-BUS
10	1	0	1	0	Non-Z-BUS	3-Wire Handshake
11	1	0	1	1	Non-Z-BUS	2-Wire Handshake

Table 2. Operating Modes

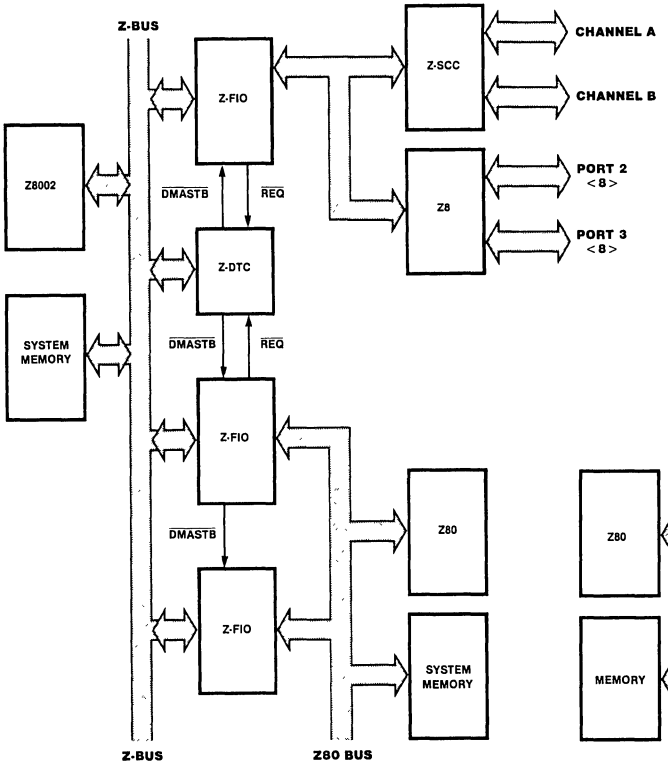


Figure 4. CPU to CPU Configuration

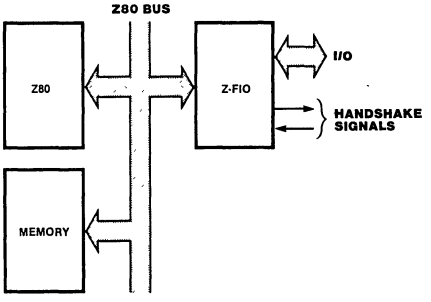


Figure 5. CPU to I/O Configuration

Pins Common To Both Sides

Pin Signals	Pin Names	Pin Numbers	Signal Description
M_0	M_0	21	M_1 and M_0 program Port 1 side CPU interface
M_1	M_1	19	
+5 Vdc	+5 Vdc	40	DC power source
GND	GND	20	DC power ground

Z-BUS Low Byte Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for data transfer to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input; active High signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ sampled while $\overline{AS}$ Low.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt. Latched on the rising edge of $\overline{AS}$.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt request to CPU.

Z-BUS High Byte Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for transfer of data to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input, active High. Signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while $\overline{AS}$ is Low.
A_0 (Address Bit 0)	G	7	33	Input, active High. With A_1, A_2, and A_3, addresses FIO internal registers.
A_1 (Address Bit 1)	H	8	32	Input, active High. With A_0, A_2, and A_3, addresses FIO internal registers.
A_2 (Address Bit 2)	I	9	31	Input, active High. With A_0, A_1, and A_3, addresses FIO internal registers.
A_3 (Address Bit 3)	J	10	30	Input, active High. With A_0, A_1, and A_2, addresses FIO internal registers.

Table 3. Signal/Pin Descriptions

Non-Z-BUS Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Port 4	Signal Description
D_0-D_7 (Data)	D_0-D_7	11-18	29-22	Bidirectional data bus.
$\overline{REQ}/\overline{WT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfer.
$\overline{DACK}$ (DMA Acknowledge)	B	2	38	Input, active Low. DMA acknowledge.
$\overline{RD}$ (Read)	C	3	37	Input, active Low. Signals CPU read from FIO.
$\overline{WR}$ (Write)	D	4	36	Input, active Low. Signals CPU write to FIO.
$\overline{CE}$ (Chip Select)	E	5	35	Input, active Low. Used to select FIO.
$C/\overline{D}$ (Control/Data)	F	6	34	Input, active High. Identifies control byte on D_0-D_7; active Low identifies data byte on D_0-D_7.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt to CPU.

Port 2—I/O Port Mode

Pin Signals	Pin Names	Pin Numbers	Mode	Signal Description
D_0-D_7 (Data)	D_0-D_7	29-22	2-Wire HS* 3-Wire HS	Bidirectional data bus.
RFD/$\overline{DAV}$ (Ready for Data/Data Available)	A	39	2-Wire HS 3-Wire HS	Output, RFD active High. Signals peripherals that FIO is ready to receive data. $\overline{DAV}$ active Low signals that FIO is ready to send data to peripherals.
$\overline{ACKIN}$ (Acknowledge Input)	B	38	2-Wire HS	Input, active Low. Signals FIO that output data is received by peripherals or that input data is valid.
$\overline{DAV}$/DAC (Data Available/Data Accepted)	B	38	3-Wire HS	Input; $\overline{DAV}$ (active Low) signals that data is valid on bus. DAC (active High) signals that output data is accepted by peripherals.
FULL	C	37	2-Wire HS	Output, open drain, active High. Signals that FIO buffer is full.
DAC/RFD (Data Accepted/Ready for Data)	C	37	3-Wire HS	Direction controlled by internal programming. Both active High. DAC (an output) signals that FIO has received data from peripheral; RFD (an input) signals that the listeners are ready for data.
EMPTY	D	36	2-Wire HS 3-Wire HS	Output, open drain, active High. Signals that FIFO buffer is empty.
$\overline{CLEAR}$	E	35	2-Wire HS 3-Wire HS	Programmable input or output, active Low. Clears all data from FIFO buffer.
DATA DIR (Data Direction)	F	34	2-Wire HS 3-Wire HS	Programmable input or output. Active High signals data input to Port 2; Low signals data output from Port 2.
IN_0	G	33	2-Wire HS 3-Wire HS	Input line to D_0 of Control Register 3.
OUT_1	H	32	2-Wire HS 3-Wire HS	Output line from D_1 of Control Register 3.
$\overline{OE}$ (Output Enable)	I	31	2-Wire HS 3-Wire HS	Input, active Low. When Low, enables bus drivers. When High, floats bus drivers at high impedance.
OUT_3	J	30	2-Wire HS 3-Wire HS	Output line from D_3 of Control register 3.

*Handshake

Table 3. Signal/Pin Descriptions (Continued)

Reset

The FIO can be reset under either hardware or software control by one of the following methods:

- By forcing both $\overline{AS}$ and $\overline{DS}$ Low simultaneously in Z-BUS mode (normally illegal).
- By forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously in non-Z-BUS mode.
- By writing a 1 to the Reset bit in Control register 0 for software reset.

In the Reset state, all control bits are cleared to 0. Only after clearing the Reset bit (by writing a 0 to it) can the other command bits be programmed. This action is true for both sides of the FIO when programmed as a CPU interface.

For proper system control, when Port 1 is reset, Port 2 is also reset. In addition, all Port 2's outputs are floating and all inputs are ignored. To initiate the data transfer, Port 2 must be enabled by Port 1. The Port 2 CPU can determine when it is enabled by reading Control register 0, which reads "floating" data bus if not enabled and "01_H" if enabled.

CPU Interfaces

The FIO is designed to work with both Z-BUS- and non-Z-BUS-type CPUs on both Port 1 and Port 2. The Z-BUS configuration interfaces CPUs with time-multiplexed address and data information on the same pins. The Z8001, Z8002, and Z8 are examples of this type of CPU. The $\overline{AS}$ (Address Strobe) pin is used to latch the address and chip select information sent out by the CPU. The R/$\overline{W}$ (Read/Write) pin and the $\overline{DS}$ (Data Strobe) pin are used for timing reads and writes from the CPU to the FIO (Figures 6 and 7).

The non-Z-BUS configuration is used for CPUs where the address and data buses are separate. Examples of this type of CPU are the Z80 and 8080. The $\overline{RD}$ (Read) and $\overline{WR}$ (Write) pins are used to time reads and writes from the CPU to the FIO (Figures 9 and 10). The C/$\overline{D}$ (Control/Data) pin is used to directly access the FIFO buffer (C/$\overline{D}$ = 0) and to access the other registers (C/$\overline{D}$ = 1). Read and write to all

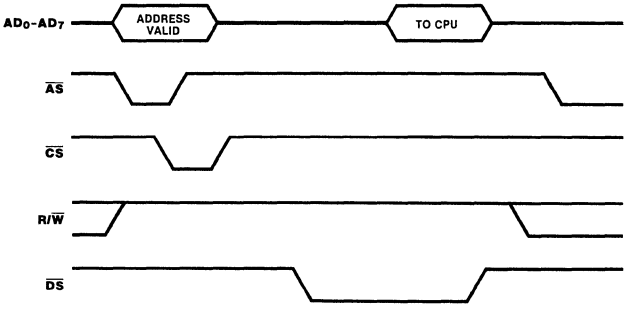


Figure 6. Z-BUS Read Cycle Timing

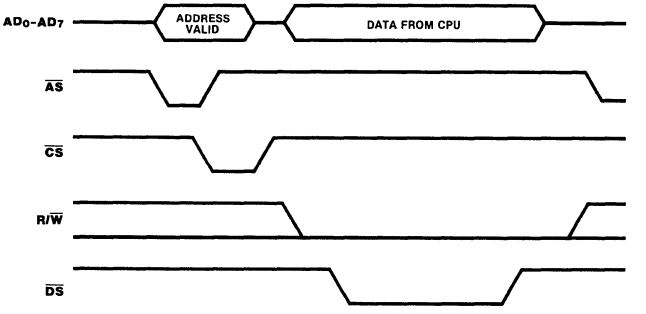


Figure 7. Z-BUS Write Cycle Timing

CPU Interfaces (Continued)

registers except the FIFO buffer[1] are two-step operations, described as follows (Figure 8). First, write the address (C/$\overline{D}$ = 1) of the register to be accessed into the Pointer Register (State 0); second, read or write (C/$\overline{D}$ = 1) to the register pointed at previously (State 1). Continuous status monitoring can be performed in State 1 by continuous Control Read operations (C/$\overline{D}$ = 1).

[1]The FIFO buffer can also be accessed by this two-step operation.

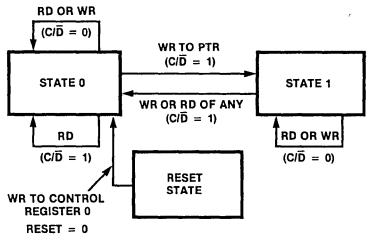


Figure 8. Register Access in Non-Z-BUS Mode

Figure 9. Non-Z-BUS Read Cycle Timing

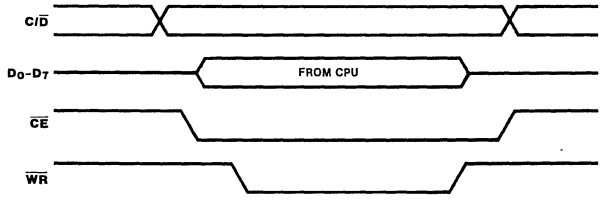


Figure 10. Non-Z-BUS Write Cycle Timing

$\overline{WAIT}$ Operation

When data is output by the CPU, the $\overline{REQ}$/$\overline{WT}$ ($\overline{WAIT}$) pin is active (Low) only when the FIFO buffer is full, the chip is selected, and the FIFO buffer is addressed. $\overline{WAIT}$ goes inactive when the FIFO buffer is not full.

When data is input by the CPU, the $\overline{REQ}$/$\overline{WT}$ pin becomes active (Low) only when the FIFO buffer is empty, the chip is selected, and the FIFO buffer is addressed. $\overline{WAIT}$ goes inactive when the FIFO buffer is not empty.

Interrupt Operation

The FIO supports Zilog's prioritized daisy chain interrupt protocol for both Z-BUS and non-Z-BUS operating modes (for more details refer to the Zilog *Z-BUS Summary*).

Each side of the FIO has seven sources of interrupt. The priorities of these devices are fixed in the following order (highest to lowest): Mailbox Message, Change in Data Direction, Pattern Match, Status Match, Overflow/Underflow Error, Buffer Full, and Buffer Empty. Each interrupt source has three bits that control how it generates the interrupt. These bits are Interrupt Pending (IP), Interrupt Enable (IE), and Interrupt Under Service (IUS).

In addition, each side of the FIO has an interrupt vector and four bits controlling the FIO interrupt logic. These bits are Vector

Interrupt Operation (Continued)

Includes Status (VIS), Master Interrupt Enable (MIE), Disable Lower Chain (DLC), and No Vector (NV).

A typical Interrupt Acknowledge cycle for Z-BUS operation is shown in Figure 11 and for non-Z-BUS operation in Figure 12. The only difference is that in Z-BUS mode, $\overline{INTACK}$ is latched by $\overline{AS}$, and in non-Z-BUS mode $\overline{INTACK}$ is not latched.

When MIE = 1, reading the vector always includes status, independent of the state of the VIS bit. In this way, when VIS = 0, all information can be obtained with one additional read, thus conserving vector space. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified.

In non-Z-BUS mode, IPs do not get set while in State 1. Therefore, in order to minimize interrupt latency, the FIO should be left in State 0.

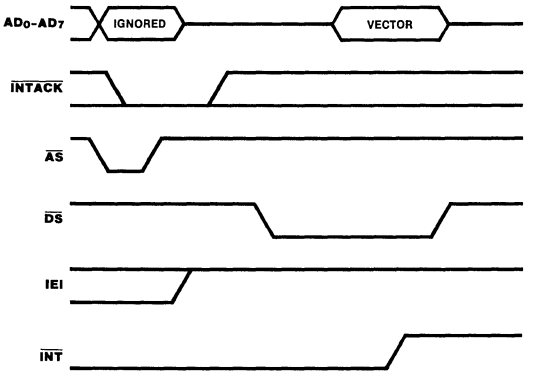


Figure 11. Z-BUS Interrupt Acknowledge Cycle

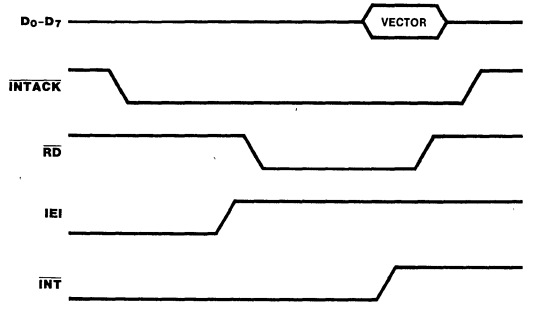


Figure 12. Non-Z-BUS Interrupt Acknowledge Cycle

CPU to CPU Operation

DMA Operation. The FIO is particularly well suited to work with a DMA in both Z-BUS and non-Z-BUS modes. A data transfer between the FIO and system memory can take place during every machine cycle on both sides of the FIO simultaneously.

In Z-BUS mode, the $\overline{DMASTB}$ pin (DMA Strobe) is used to read or write into the FIFO buffer. The R/$\overline{W}$ (Read/Write) and $\overline{DS}$ (Data Strobe) signals are ignored by the FIO; however, the $\overline{CS}$ (Chip Select) signal is not ignored and therefore must be kept invalid. Figures 13 and 14 show typical timing.

In Non-Z-BUS mode, the $\overline{DACK}$ pin (DMA Acknowledge) is used to tell the FIO that its DMA request is granted. After $\overline{DACK}$ goes Low, every read or write to the FIO goes into the FIFO buffer. Figures 15 and 16 show typical timing.

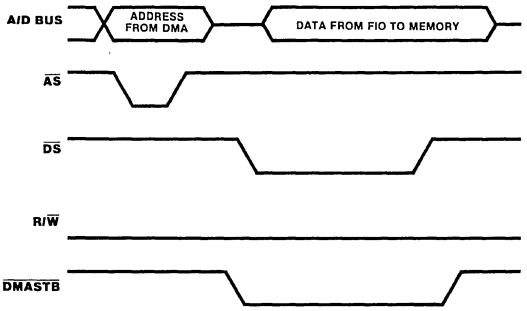


Figure 13. Z-BUS FIO to Memory Data Transaction

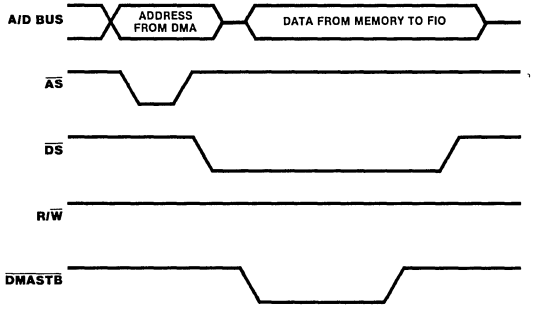


Figure 14. Z-BUS Memory to FIO Data Transaction

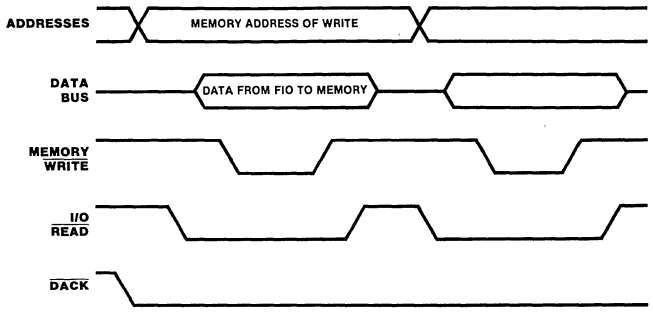


Figure 15. Non-Z-BUS FIO to Memory Transaction

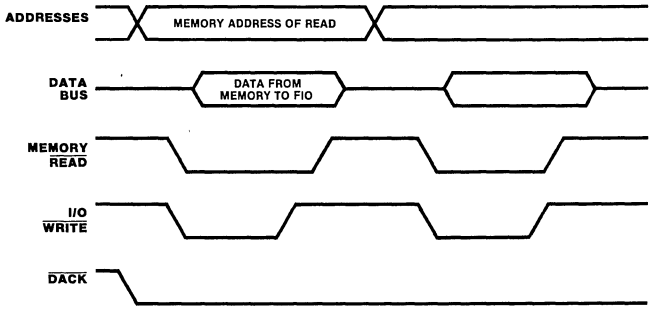
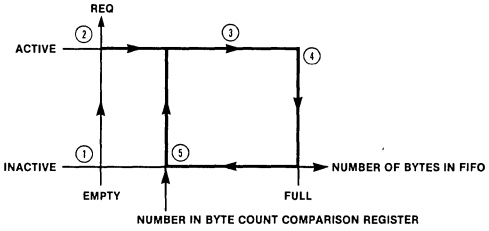


Figure 16. Non-Z-BUS Memory to FIO Data Transaction

CPU to CPU Operation (Continued)

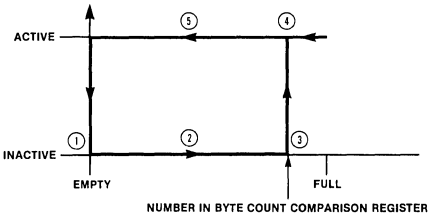
The FIO provides a special mode to enhance its DMA transfer capability. When data is written into the FIFO buffer, the REQ/WT (REQUEST) pin is active (Low) until the FIFO buffer is full. It then goes inactive and stays inactive until the number of bytes in the FIFO buffer is equal to the value programmed into the Byte Count Comparison register. Then the REQUEST signal goes active and the sequence starts over again (Figure 17).

When data is read from the FIO, the REQ/WT pin (REQUEST) is inactive until the number of bytes in the FIFO buffer is equal to the value programmed in the Byte Count Comparison register. The REQUEST signal then goes active and stays active until the FIFO buffer is empty. When empty, REQUEST goes inactive and the sequence starts over again (Figure 18).



NOTES:
1. FIFO empty.
2. REQUEST enabled, FIO requests DMA transfer.
3. DMA transfers data into the FIO.
4. FIFO full, REQUEST inactive.
5. The FIFO empties from the opposite port until the number of bytes in the FIFO buffer is the same as the number programmed in the Byte Count Comparison register.

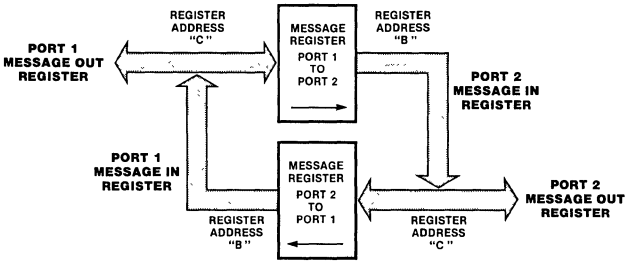
Figure 17. Byte Count Control: Write to FIO



NOTES:
1. FIFO empty.
2. CPU/DMA fills FIFO buffer from the opposite port.
3. Number of bytes in FIFO buffer is the same as the number of bytes programmed in the Byte Count Comparison register.
4. REQUEST goes active.
5. DMA transfers data out of FIFO until it is empty.

Figure 18. Byte Count Control: Read from FIO

Message Registers. Two CPUs can communicate through a dedicated "mailbox" register without involving the 128 × 8 bit FIFO buffer (Figure 19). This mailbox approach is useful for transferring control parameters between the interfacing devices on either side of the FIO without using the FIFO buffer. For example, when Port 1's CPU writes to the Message Out register, Port 2's message IP is set. If interrupts are enabled, Port 2's CPU is interrupted. Port 2's message IP status is readable from the Port 1 side. When Port 2's CPU reads the data from its Message In register, the Port 2 IP is cleared. Thus, Port 1's CPU can read when the message has been read and can now send another message or follow whatever protocol that is set up between the two CPU's. The same transfer can also be made from Port 2's CPU to Port 1's CPU.



NOTE: Usable only for CPU/CPU interface.

Figure 19. Message Register Operation

CPU to CPU Operation (Continued)

CLEAR (Empty) FIFO Operation. The $\overline{\text{CLEAR}}$ FIFO bit (active Low) clears the FIFO buffer of data. Writing a 0 to this bit empties the FIFO buffer, inactivates the REQUEST line, and disables the handshake (if programmed). The $\overline{\text{CLEAR}}$ bit does not affect any control or data register. To remove the $\overline{\text{CLEAR}}$ state, write a 1 to the $\overline{\text{CLEAR}}$ bit.

In CPU/CPU mode, under program control, only one of the ports can empty the FIFO by writing to its Control Register 3, bit 6. The Port 1 CPU must program bit 7 in Control Register 3 to determine which port controls the $\overline{\text{CLEAR}}$ FIFO operation (0 = Port 1 control; 1 = Port 2 control).

Direction of Data Transfer Operation. The Data Direction bit controls the direction of data transfer in the FIFO buffer. The Data Direction bit is defined as 0 = output from CPU and 1 = input to CPU. This bit reads correctly when read by either port's CPU. For example, if Port 1's CPU reads a 0 (CPU output) in its Data Direction bit, then Port 2's CPU reads a 1 (input to CPU) in its Data Direction bit.

In CPU/CPU mode, under program control, only one of the ports can control the direction of data transfer. The Port 1 CPU must program bit 5 in Control Register 3 to determine which port controls the data direction (0 = Port 1 control; 1 = Port 2 control). Figure 20 shows FIO data transfer options.

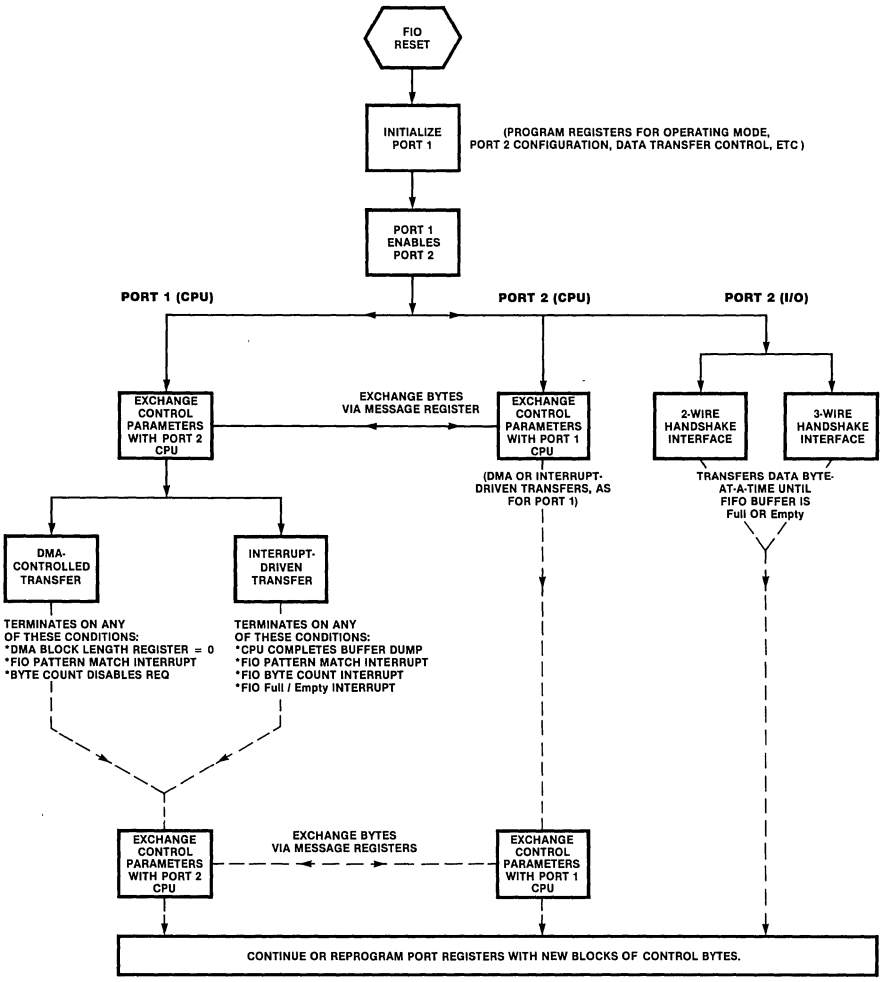


Figure 20. FIO Data Transfer Options

CPU to I/O Operation

When Port 2 is programmed in the Interlocked 2-Wire Handshake mode or the 3-Wire Handshake mode, and Port A is programmed in Z-BUS or non-Z-BUS Microprocessor mode, the FIO interfaces a CPU and a peripheral device. In the Interlocked 2-Wire Handshake mode, RFD/$\overline{\text{DAV}}$ and $\overline{\text{ACKIN}}$ strobe data to and from Port 2. In the 3-Wire Handshake mode, RFD/$\overline{\text{DAV}}$, $\overline{\text{DAV}}$/DAC, and DAC/RFD signals control data flow.

Interlocked 2-Wire Handshake. In the Interlocked Handshake, the action of the FIO must be acknowledged by the other half of the handshake before the next action can take place. In output mode, Port 2 does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, in input mode, Port 2 does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging Port 2's acceptance of the last byte. This allows the FIO to directly interface to a Z8's port, a CIO's port, a UPC's port, another FIO port, or another FIFO Z8060, with no external logic (Figures 21 and 22).

3-Wire Handshake. The 3-Wire Handshake is designed for applications in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate that an input port is ready for new data or that it has accepted the present data. In the 3-Wire Handshake, the rising edge of the RFD status line indicates that the port is ready for data, and the rising edge of the DAC status line indicates that the data has been accepted. With 3-Wire Handshake, the lines of many input ports can be bussed together with open-drain drivers and the output port knows when all of the ports are ready and have accepted the data. This handshake is the same handshake used in the IEEE-488 Instruments. Since the port's direction can be changed under software control, bidirectional IEEE-488-type transfers can be performed. Figures 23 and 24 show the timings associated with 3-Wire Handshake communications.

CLEAR FIFO Operation. In CPU-to-I/O operation, the CLEAR FIFO operation can be performed by the CPU side (Port 1) under software control as previously explained. The CLEAR FIFO operation can also be performed under hardware control by defining the $\overline{\text{CLEAR}}$ pin of Port 2 as an input (Control Register 3, bit 7 = 1).

For cascading purposes, the $\overline{\text{CLEAR}}$ pin can also be defined as an output (Control Register 3, bit 7 = 0), which reflects the current state of the $\overline{\text{CLEAR}}$ FIFO bit. It can then empty other FIOs or initialize other devices in the system.

Data Direction Control. In CPU-to-I/O mode, the direction of data transfer can be controlled by the CPU side (Port 1) under software control as previously explained. The data direction can also be determined by hardware control by defining the Data Direction pin of Port 2 as an input (Control Register 3, bit 5 = 1).

For cascading purposes, the Data Direction pin can also be defined as an output (Control Register 3, bit 5 = 0) pin which reflects the current state of the Data Direction bit. It can then be used to control the direction of data transfer for other FIOs or for external logic.

On the Port 2 side, when data direction is 0, Port 2 is in Output Handshake mode. When data direction is 1, Port 2 is in Input Handshake mode.

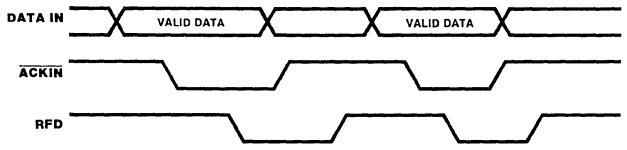


Figure 21. Interlocked Handshake Timing (Input) Port 2 Side Only

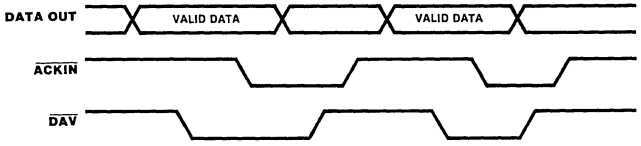


Figure 22. Interlocked Handshake Timing (Output) Port 2 Side Only

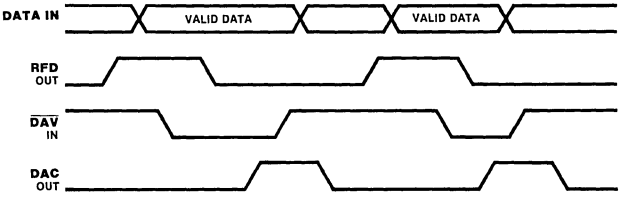


Figure 23. Input (Acceptor) Timing IEEE-488 HS Port: Port 2 Side Only

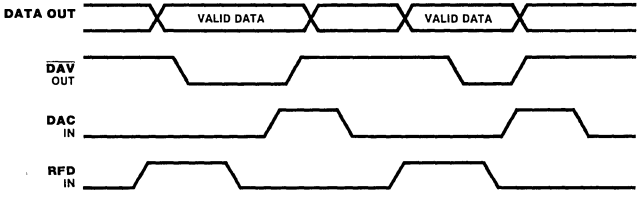


Figure 24. Output (Source) Timing IEEE-488 HS Port: Port 2 Side Only

Programming

The programming of the FIO is greatly simplified by the efficient grouping of the various operation modes in the control registers. Since all of the control registers are read/write, the need for maintaining their image in system memory is eliminated. Also, the read/write feature of the registers aids in system debugging.

Each side of the FIO has 16 registers. All 16 registers are used by the Port 1 side; Control register 2 is not used on the Port 2 side. All registers are addressable 0_H through F_H.

In the Z-BUS Low Byte mode, the FIO allows two methods for register addressing under control of the Right Justify Address (RJA) bit in Control register 0. When RJA = 0, address bus bits 1-4 are used for register addressing and bits 1, 5, 6, and 7 are ignored (Table 4). When RJA = 1, bits 0-3 are used for the register addresses, and bits 4-7 are ignored.

Control Registers. These four registers specify FIO operation. The Port 2 side control registers operate only if the Port 2 device is a CPU. The Port 2 CPU can control interface operations, including data direction, only when enabled by the setting of bit 0 in the Port 1 side of Control Register 2. A 1 in bit 1 of the same register enables the handshake logic.

Interrupt Status Registers. These four registers control and monitor the priority interrupt functions for the FIO.

Interrupt Vector Register. This register stores the interrupt service routine address. This vector is placed on D_0-D_7 when IUS is set by the Interrupt Acknowledge signal from the CPU. When bit 4 (Vector Includes Status) is set in Control Register 0, the reason for the interrupt is encoded within the vector address in bits 1, 2, and 3. If bit 5 is set in Control register 0, no vector is output by the FIO during an Interrupt Acknowledge cycle. However, IUS is set as usual.

Non Z-BUS	D_7-D_4	D_3	D_2	D_1	D_0	
Z-BUS High		A_3	A_2	A_1	A_0	
Z-BUS Low RJA=0	AD_7-AD_5	AD_4	AD_3	AD_2	AD_1	AD_0
Z-BUS Low RJA=1	AD_7-AD_4	AD_3	AD_2	AD_1	AD_0	
Description						
Control Register 0	x	0	0	0	0	x
Control Register 1	x	0	0	0	1	x
Interrupt Status Register 0	x	0	0	1	0	x
Interrupt Status Register 1	x	0	0	1	1	x
Interrupt Status Register 2	x	0	1	0	0	x
Interrupt Status Register 3	x	0	1	0	1	x
Interrupt Vector Register	x	0	1	1	0	x
Byte Count Register	x	0	1	1	1	x
Byte Count Comparison Register	x	1	0	0	0	x
Control Register 2*	x	1	0	0	1	x
Control Register 3	x	1	0	1	0	x
Message Out Register	x	1	0	1	1	x
Message In Register	x	1	1	0	0	x
Pattern Match Register	x	1	1	0	1	x
Pattern Mask Register	x	1	1	1	0	x
Data Buffer Register	x	1	1	1	1	x

x = Don't Care
*Register is only on Port 1 side

Table 4. FIO Register Address Summary

Programming (Continued)

Byte Count Compare Register. This register contains a value compared with the byte count in the Byte Count register. If the Byte Count Compare interrupt is enabled, an interrupt will occur upon compare.

Message Out Register. Either CPU can place a message in its Message Out register. If the opposite side Message register interrupt is enabled, the receiving side CPU will receive an interrupt request, advising that a message is present in its Message In register. Bit 5 in Control Register 1 on the initiating side is set when a message is written. It is cleared when the message is read by the receiving CPU.

Message In Register. This register receives a message placed in the Message Out register by the opposite side CPU.

Pattern Match Register. This register contains a bit pattern matched against the byte in the Data Buffer register. When these patterns match, a Pattern Match interrupt will be generated, if previously enabled.

Pattern Mask Register. The Pattern Mask register may be programmed with a bit pattern mask that limits comparable bits in the Pattern Match register to non-masked bits (1 = mask).

Data Buffer Register. This register contains the data to be read from or written to the FIFO buffer.

Byte Count Register. This is a read-only register, containing the byte count for the FIFO buffer. The byte count is derived by subtracting the number of bytes read from the buffer from the number of bytes written into the buffer. The count is "frozen" for an accurate reading by setting bit 6 (Freeze Status register) in Control Register 1. This bit is cleared when the Byte Count register read is completed.

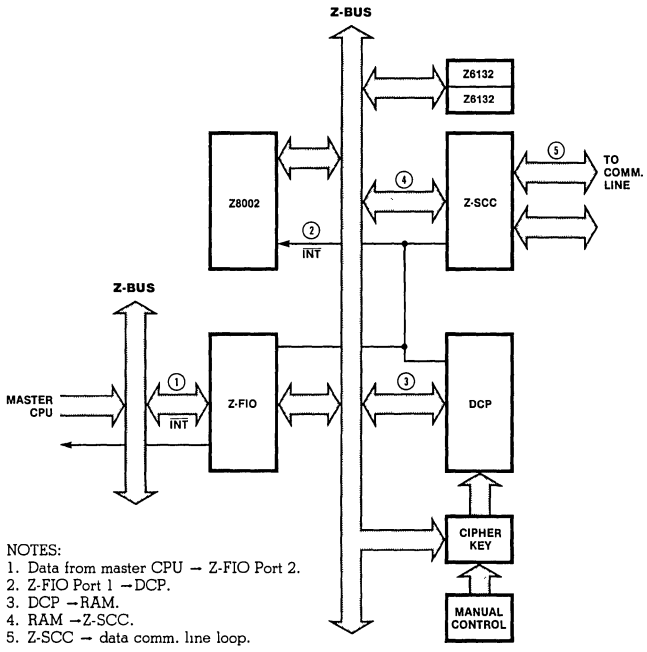
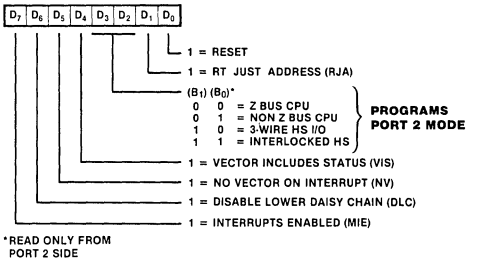


Figure 25. Typical Application: Node Controller

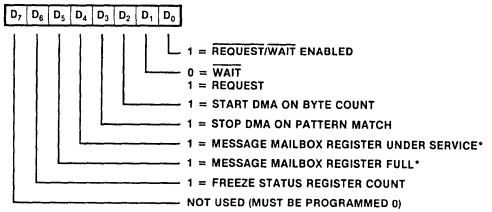
Registers

Control Register 0
Address: 0000
(Read/Write)



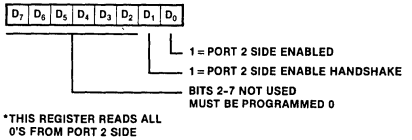
*READ ONLY FROM PORT 2 SIDE

Control Register 1
Address: 0001
(Read/Write)



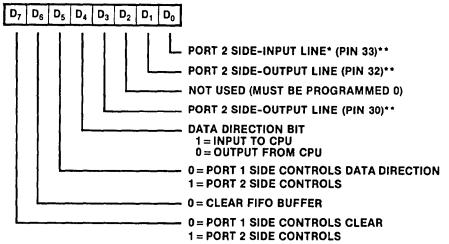
*READ ONLY BITS

Control Register 2*
Address: 1001
(Read/Write)



*THIS REGISTER READS ALL 0'S FROM PORT 2 SIDE

Control Register 3
Address: 1010
(Read/Write)



*READ-ONLY BITS
**ONLY WHEN PORT 2 IS AN I/O PORT

Figure 26. Control Registers

Interrupt Status Register 0
Address: 0010
(Read/Write)

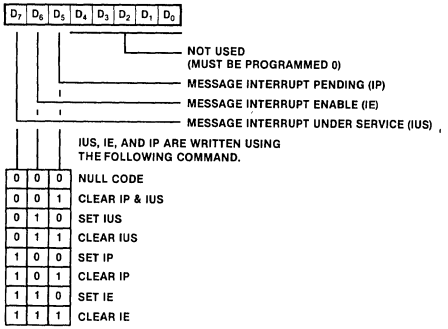
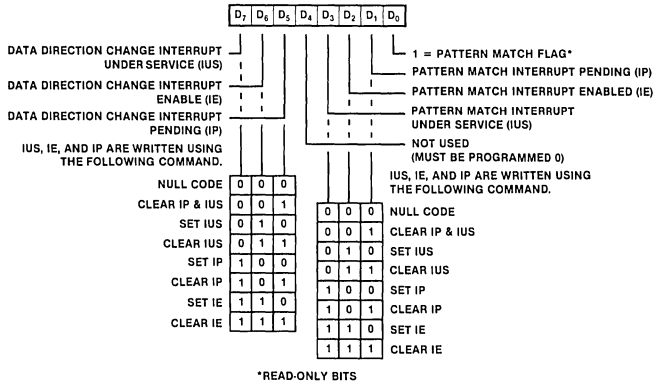


Figure 27. Interrupt Status Registers

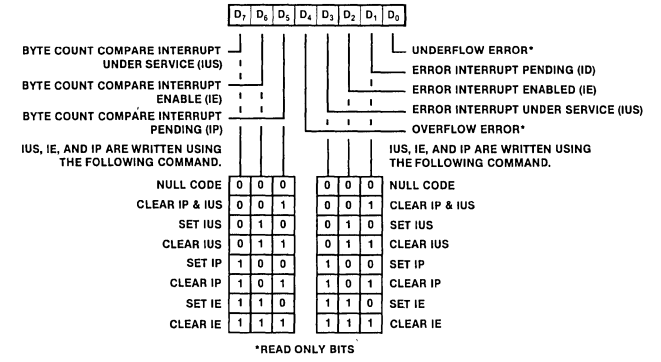
Interrupt Status Register 1

Address: 0011
(Read/Write)



Interrupt Status Register 2

Address: 0100
(Read/Write)



Interrupt Status Register 3

Address: 0101
(Read/Write)

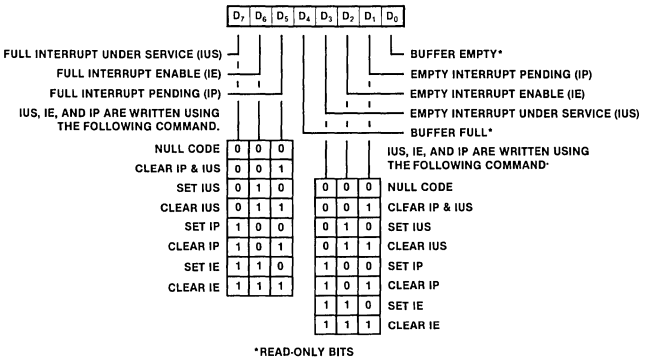


Figure 27. Interrupt Status Registers (Continued)

Registers
(Continued)

Byte Count Register
Address: 0111

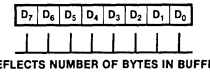


Figure 28. Byte Count Register

Interrupt Vector Register
Address: 0110
(Read/Write)

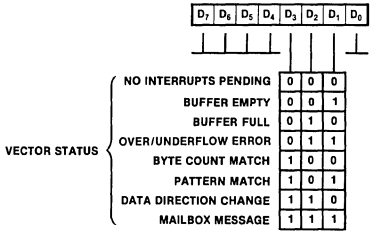


Figure 29. Interrupt Vector Register

Pattern Match Register
Address: 1011
(Read/Write)

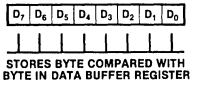


Figure 30. Pattern Match Register

Pattern Mask Register
Address: 1110
(Read/Write)

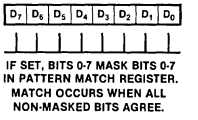


Figure 31. Pattern Mask Register

Data Buffer Register
Address: 1111
(Read/Write)

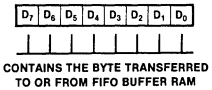


Figure 32. Data Buffer Register

Byte Count Comparison Register
Address: 1000
(Read/Write)

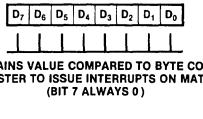


Figure 33. Byte Count Comparison Register

Message Out Register
Address: 1011
(Read/Write)

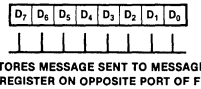


Figure 34. Message Out Register

Message In Register
Address: 1100
(Read Only)

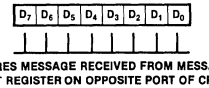


Figure 35. Message In Register

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

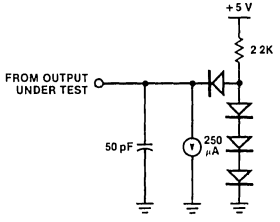


Figure 36. Standard Test Load

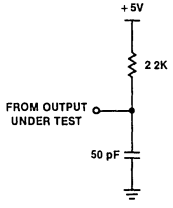


Figure 37. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

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

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

Inputs

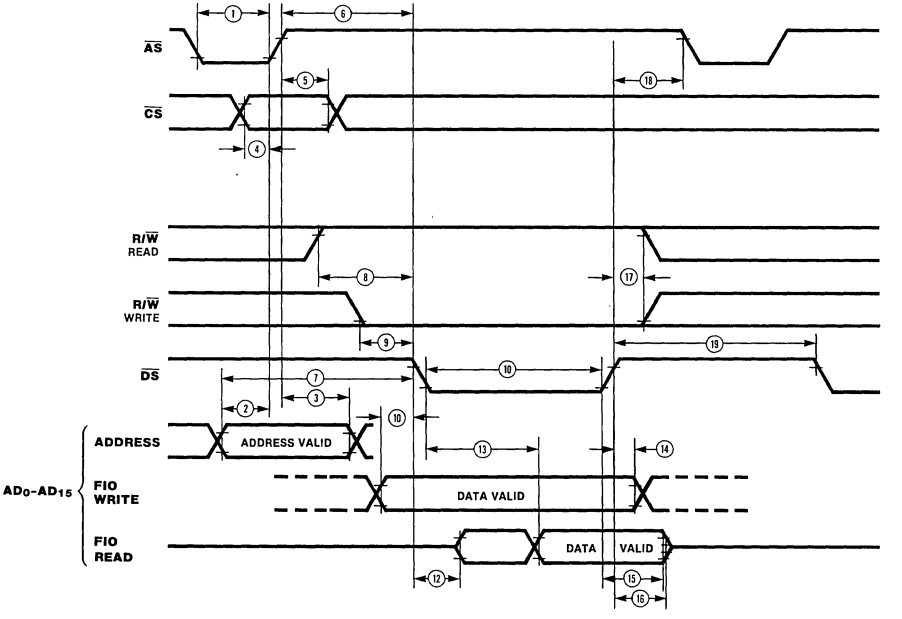
tr	Any Input Rise Time		100	ns	
tf	Any Input Fall Time		100	ns	

f = 1 MHz, over specified temperature range.

Z-BUS CPU Interface Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TwAS	$\overline{AS}$ Low Width	70		ns	
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	10		ns	1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		ns	1
4	TsCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		ns	1
5	ThCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		ns	1
6	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↑ Delay	60		ns	1
7	TsA(DS)	Address to $\overline{DS}$ ↓	120		ns	
8	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		ns	
9	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		ns	
10	TwDS	$\overline{DS}$ Low Width	390		ns	
11	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		ns	
12	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0		ns	
13	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		255	ns	
14	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		ns	
15	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		ns	
16	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70	ns	2
17	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	60		ns	
18	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		ns	
19	Trc	Valid Access Recovery Time	1000		ns	3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{DS}$ of one CIO access to $\overline{DS}$ of another FIO access (either read or write).

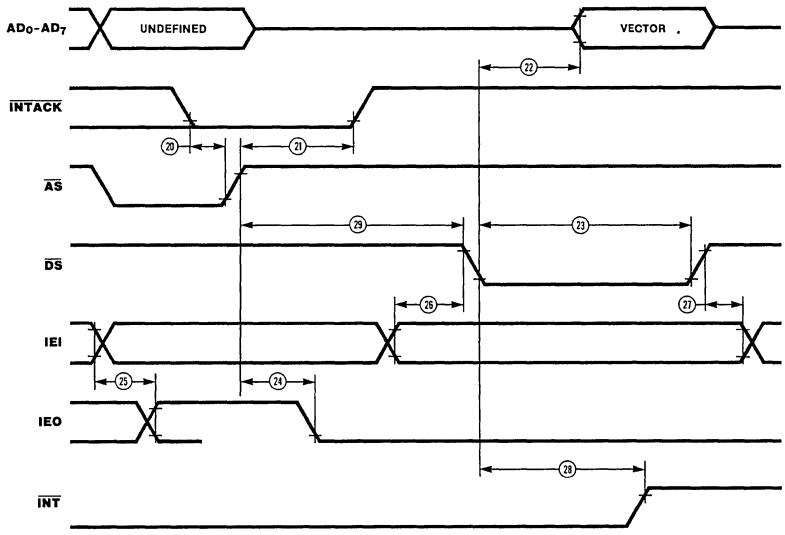


Z-BUS CPU Interrupt Acknowledge Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
20	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		ns	
21	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		ns	
22	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		360	ns	
23	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		ns	
24	TdAS(IEO)	$\overline{AS}$ ↓ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		350	ns	4
25	TdIEI(IEO)	IEI to IEO Delay		150	ns	4
26	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		ns	
27	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Hold Time	200		ns	4
28	TdDS(INT)	$\overline{DS}$ ($\overline{INTACK}$ Cycle) to $\overline{INT}$ Delay			ns	
29	TdDCST	Interrupt Daisy Chain Settle Time			ns	4

NOTES:

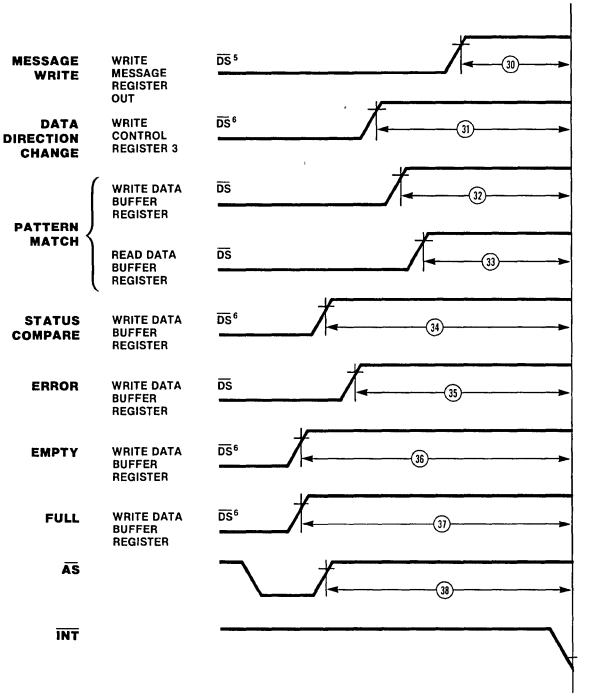
4. The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{AS}$ to $\overline{DS}$ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.



Z-BUS Interrupt Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
30	TdMW(INT)	Message Write to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	5
31	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	6
32	TdPMW(INT)	Pattern Match to $\overline{INT}$ Delay (Write Case)		1	$\overline{AS}$ Cycles + ns	
33	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	
34	TdSC(INT)	Status Compare to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	6
35	TdER(INT)	Error to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	
36	TdEM(INT)	Empty to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	6
37	TdFL(INT)	Full to $\overline{INT}$ Delay		1	$\overline{AS}$ Cycles + ns	6
38	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay			$\overline{AS}$ Cycles + ns	

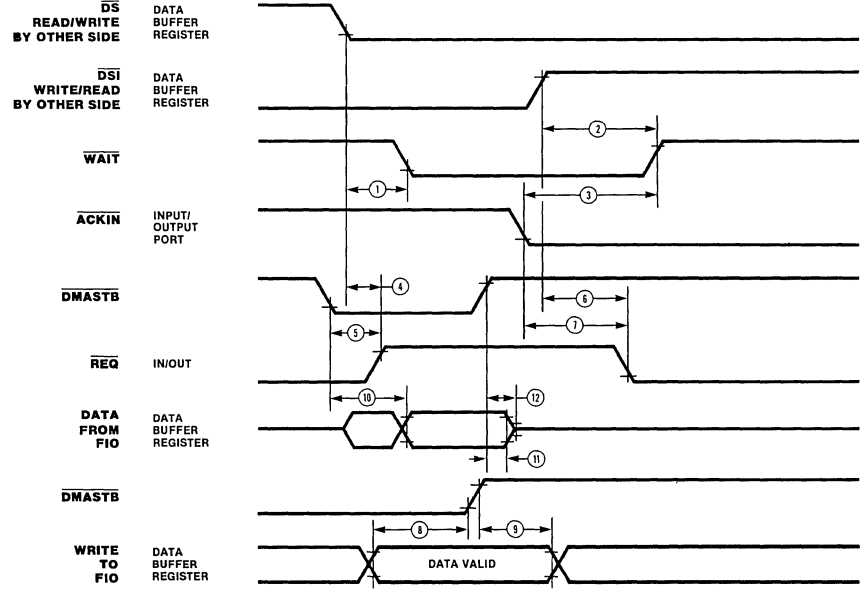
NOTES:
5. Write is from the other side of FIO.
6 Write can be from either side, depending on programming of FIO



Z-BUS Request/Wait Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdDS(WAIT)	$\overline{DS}$ ↓ to $\overline{WAIT}$ ↓ Delay			ns	
2	TdDSI(WAIT)	$\overline{DSI}$ ↓ to $\overline{WAIT}$ ↑ Delay			ns	
3	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay			ns	1
4	TdDS(REQ)	$\overline{DS}$ ↓ to $\overline{REQ}$ ↑ Delay			ns	
5	TdDMA(REQ)	$\overline{DMASTB}$ ↓ to $\overline{REQ}$ ↑ Delay			ns	
6	TdDSI(REQ)	$\overline{DSI}$ ↑ to $\overline{REQ}$ ↓ Delay			ns	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ ↓ Delay			ns	
8	TdSU(DMA)	Data Setup Time to $\overline{DMASTB}$	200		ns	
9	TdH(DMA)	Data Hold Time to $\overline{DMASTB}$	30		ns	
10	TdDMA(DR)	$\overline{DMASTB}$ ↓ to Valid Data			ns	
11	TdDMA(DRH)	$\overline{DMASTB}$ ↑ to Data Not Valid	0		ns	
12	TdDMA(DR2)	$\overline{DMASTB}$ ↑ to Data Bus Float		70	ns	

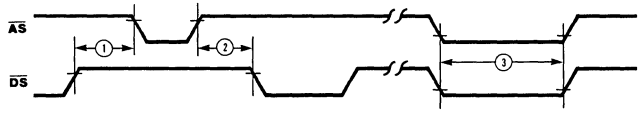
NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.



Z-BUS Reset Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		ns	
2	TdASQ(DS)	Delay for $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		ns	
3	Tw(AS+DS)	Minimum Width of $\overline{AS}$ and $\overline{DS}$ Both Low for Reset	500		ns	1

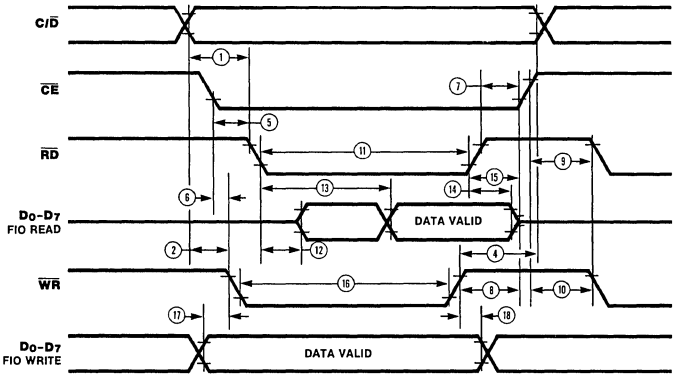
NOTES:
1. Internal circuitry allows for the reset provided by the Z8 ($\overline{DS}$ held Low while $\overline{AS}$ pulses) to be sufficient.



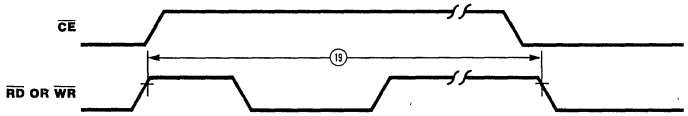
Non-Z-BUS CPU Interface Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TsA(RD)	Address Setup to $\overline{RD}$ ↓	80		ns	1
2	TsA(WR)	Address Setup to $\overline{WR}$ ↓	80		ns	
3	ThA(RD)	Address Hold Time to $\overline{RD}$ ↑	0		ns	1
4	ThA(WR)	Address Hold Time to $\overline{WR}$ ↑	0		ns	
5	TsCEI(RD)	$\overline{CE}$ Low Setup Time to $\overline{RD}$	0		ns	1
6	TsCEI(WR)	$\overline{CE}$ Low Setup Time to $\overline{WR}$	0		ns	
7	ThCEI(RD)	$\overline{CE}$ Low Hold Time to $\overline{RD}$	0		ns	1
8	ThCEI(WR)	$\overline{CE}$ Low Hold Time to $\overline{WR}$	0		ns	
9	TsCEh(RD)	$\overline{CE}$ High Setup Time to $\overline{RD}$	100		ns	1
10	TsCEh(WR)	$\overline{CE}$ High Setup Time to $\overline{WR}$	100		ns	
11	TwRD1	$\overline{RD}$ Low Width	400		ns	
12	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		ns	
13	TdRDf(DR)	$\overline{RD}$ ↓ to Valid Data Delay		300	ns	
14	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		ns	
15	TdRD(DRz)	$\overline{RD}$ ↑ to Data Bus Float		70	ns	2
16	TwWR1	$\overline{WR}$ Low Width	400		ns	
17	TsDW(WR)	Data Setup Time to $\overline{WR}$	0		ns	
18	ThDW(WR)	Data Hold Time to $\overline{WR}$	0		ns	
19	Trc	Valid Access Recovery Time	1000		ns	3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{RD}$ ↑ or $\overline{WR}$ ↑ of one FIO access to $\overline{RD}$ ↓ or $\overline{WR}$ ↓ of another FIO access.



Non-Z-BUS CPU Interface Timing



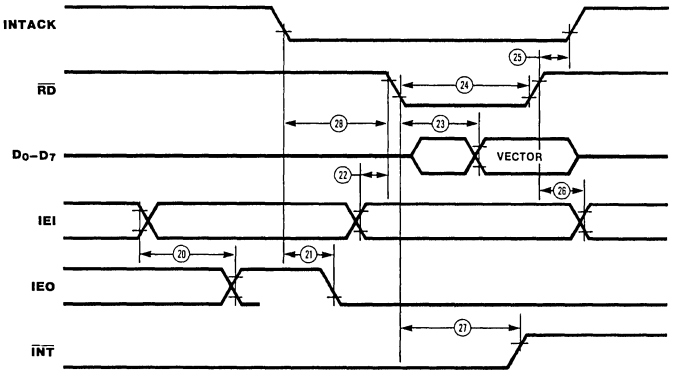
Non-Z-BUS Interface Timing

Non-Z-BUS Interrupt Acknowledge Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
20	TdIEI(IEO)	IEI to IEO Delay	150		ns	4
21	TdI(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay	350		ns	4
22	TsIEI(RDA)	IEI Setup Time to $\overline{RD}$ (Acknowledge)	200		ns	4
23	TdRD(DR)	$\overline{RD}$ ↓ to Vector Valid Delay		300	ns	
24	TwRDl(IA)	Read Low Width (Interrupt Acknowledge)	400		ns	
25	ThIA(RD)	$\overline{INTACK}$ ↑ to $\overline{RD}$ ↑ Hold Time	30		ns	
26	ThIEI(RD)	IEI Hold Time to $\overline{RD}$ ↑	100		ns	
27	TdRD(INT)	$\overline{RD}$ ↑ to $\overline{INT}$ ↑ Delay			ns	
28	TdDCST	Interrupt Daisy Chain Settle Time			ns	4

NOTES:

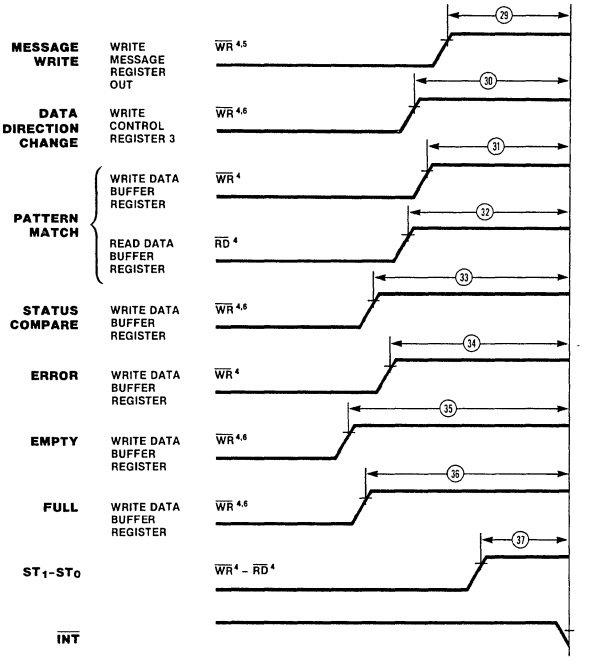
4. The parameter for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{INTACK}$ ↓ to $\overline{RD}$ ↓ must be greater than the sum of TdINA(IEO) for the highest priority peripheral, TsIEI(RD) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.



Non-Z-BUS Interrupt Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
29	TdMW(INT)	Message Write to $\overline{INT}$ Delay			ns	5,6
30	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay			ns	5,7
31	TdPMW(INT)	Pattern Match (Write Case) to $\overline{INT}$ Delay			ns	5
32	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay			ns	5
33	TdSC(INT)	Status Compare to $\overline{INT}$ Delay			ns	5,7
34	TdER(INT)	Error to $\overline{INT}$ Delay			ns	5,7
35	TdEM(INT)	Empty to $\overline{INT}$ Delay			ns	5,7
36	TdFL(INT)	Full to $\overline{INT}$ Delay			ns	5,7
37	TdS0(INT)	State 0 to $\overline{INT}$ Delay			ns	

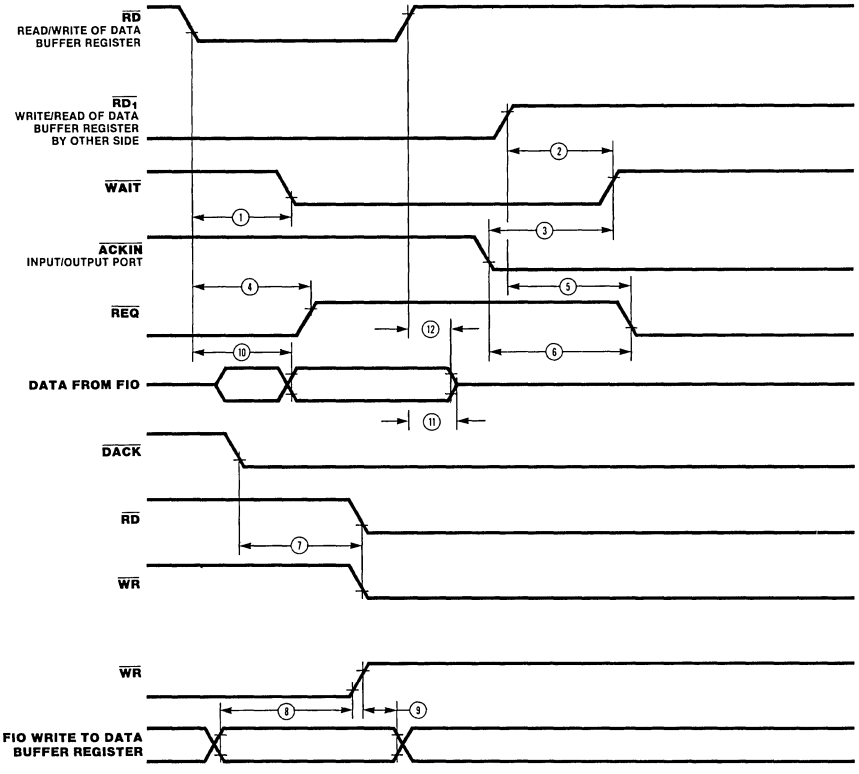
NOTES:
5. Delay number is valid for State 0 only.
6. Write is from other side of FIO.
7. Write can be from either side, depending on programming of FIO.



Non-Z-BUS Request/Wait Timing

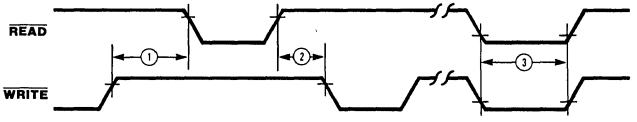
Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdRD(WT)	$\overline{RD}$ ↓ to $\overline{WAIT}$ Active			ns	
2	TdRD1(WT)	$\overline{RD1}$ ↓ to $\overline{WAIT}$ Inactive			ns	
3	TdACK(WT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ Inactive			ns	1
4	TdRD(REQ)	$\overline{RD}$ ↓ to $\overline{REQ}$ Inactive			ns	
5	TdRD1(REQ)	$\overline{RD1}$ ↓ to $\overline{REQ}$ Active			ns	
6	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ Active			ns	
7	TdDAC(RD)	$\overline{DACK}$ ↓ to $\overline{RD}$ ↓ or $\overline{WR}$ ↓			ns	
8	TSU(WR)	Data Setup Time to $\overline{WR}$			ns	
9	Th(WR)	Data Hold Time to $\overline{WR}$			ns	
10	TdDMA	$\overline{RD}$ ↓ to Valid Data			ns	2
11	TdDMA(DRH)	$\overline{RD}$ ↑ to Data Not Valid	0		ns	2
12	TdDMA(DRZ)	$\overline{RD}$ ↑ to Data Bus Float		70	ns	2

NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Input Handshake.
2. Only when $\overline{DACK}$ is active.



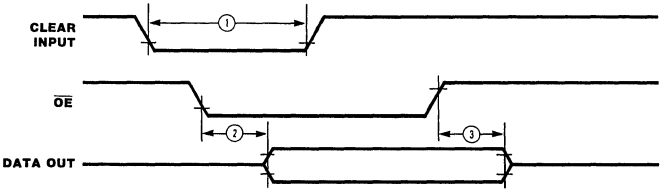
Non-Z-BUS Reset Timing

Number	Symbol	Parameter	Min	Max	Units
1.	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓	100		ns
2.	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓	100		ns
3.	TwRD + WR	Width of $\overline{RD}$ and $\overline{WR}$, both Low for Reset	500		ns



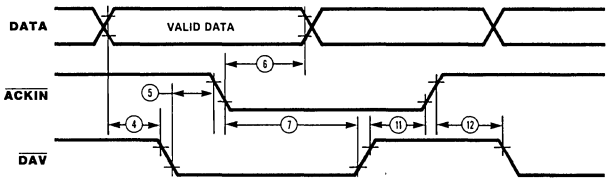
Port 2 Side Operation

Number	Symbol	Parameter	Min	Max	Units
1.	TwCLR	Width of Clear to Reset FIFO	700		ns
2.	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven	0		ns
3.	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float			ns

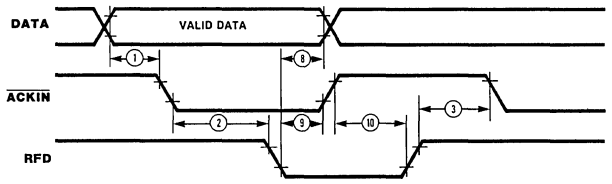


FIO 2-Wire Handshake Timing

Number	Symbol	Parameter	Min	Max	Units
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup Time			ns
2	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		ns
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		ns
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		ns
5	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		ns
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ Hold Time			ns
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	0		ns
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		ns
9	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		ns
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handshake	0		ns
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)	0		ns
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓	0		ns



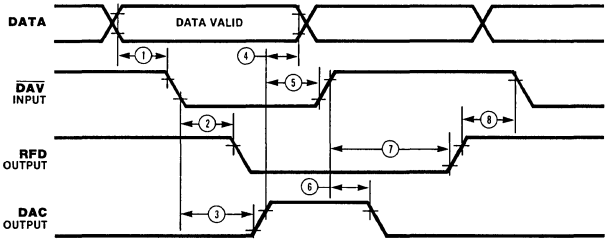
2-Wire Handshake (Port 2 Side Only) Output



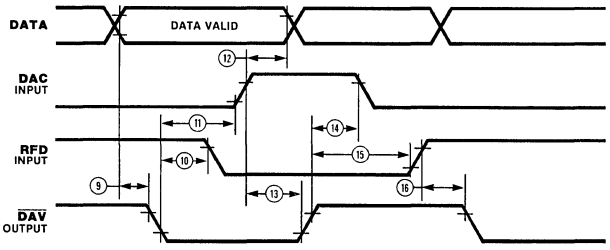
2-Wire Handshake (Port 2 Side Only) Input

3-Wire Handshake Timing

Number	Symbol	Parameter	Min	Max	Units
1	TsDI(DAV)	Data Input to $\overline{DAV}$ ↓ Setup Time			ns
2	TdDAVIf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0		ns
3	TdDAVf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0		ns
4	ThDI(DAC)	Data In to DAC ↑ Hold Time	0		ns
5	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay	0		ns
6	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0		ns
7	TdDAVIr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0		ns
8	TdRFDI(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0		ns
9	TsDO(DAC)	Data Out to $\overline{DAV}$ ↓			ns
10	TdDAVOf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0		ns
11	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0		ns
12	ThDO(DAC)	Data Out to DAC ↑ Hold Time			ns
13	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay			ns
14	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0		ns
15	TdDAVOr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0		ns
16	TdRFDO(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0		ns



3-Wire Handshake Input



3-Wire Handshake Output

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8038	CE	4.0 MHz	Z-FIO (Z-BUS compatible, 40-pin)
Z8038	CS	4.0 MHz	Same as above
Z8038	DE	4.0 MHz	Same as above
Z8038	DS	4.0 MHz	Same as above
Z8038	PE	4.0 MHz	Same as above
Z8038	PS	4.0 MHz	Same as above
Z8038A	CE	6.0 MHz	Z-FIO (Z-BUS compatible, 40-pin)
Z8038A	CS	6.0 MHz	Same as above
Z8038A	DE	6.0 MHz	Same as above
Z8038A	DS	6.0 MHz	Same as above
Z8038A	PE	6.0 MHz	Same as above
Z8038A	PS	6.0 MHz	Same as above
Z8538	CE	4.0 MHz	FIO (Universal, 40-pin)
Z8538	CS	4.0 MHz	Same as above
Z8538	DE	4.0 MHz	Same as above
Z8538	DS	4.0 MHz	Same as above
Z8538	PE	4.0 MHz	Same as above
Z8538	PS	4.0 MHz	Same as above
Z8538A	CE	6.0 MHz	FIO (Universal, 40-pin)
Z8538A	CS	6.0 MHz	Same as above
Z8538A	DE	6.0 MHz	Same as above
Z8538A	DS	6.0 MHz	Same as above
Z8538A	PE	6.0 MHz	Same as above
Z8538A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, S = 0°C to +70°C.

Z8060
Z8000™ FIFO Buffer Unit and Z-FIO Expander

Zilog

Product Brief

March 1981

Features

- Asynchronous, bidirectional first-in, first-out buffer.
- Extends depth of Z-FIO without limit.
- 128 × 8 organization.
- 3-state data outputs.
- Empty and Full status pins are wire-ORed among multiple stages.

Description

The Z-FIFO first-in, first-out buffer unit is a 128 × 8-bit memory with bidirectional data transfer capability and handshake logic. Its structure is similar to that of other FIFOs that are commonly available, such as the AM2812 and the 3351. The handshake logic used is compatible with that of the Z8, the Z-CIO, and Z-FIO. Z-FIFO buffers can be cascaded, end to end, without limit, their RFD/$\overline{DAV}$ and $\overline{ACKIN}$ signals daisy-chained, to make a FIFO array any desired number of words deep. Two such channels in parallel, suitably controlled, make up a 16-bit-wide buffer array.

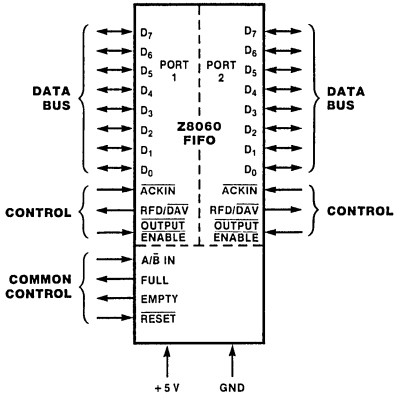


Figure 1. Pin Functions

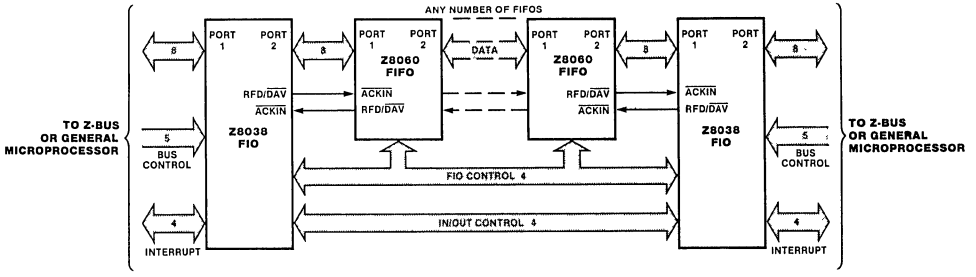


Figure 2. Using FIFOs to Extend FIOs

Z8065
Z8000™ Z-BEP
Burst Error Processor

Zilog

Product Brief

March 1981

Features

- Error detection and correction for high-speed data transfers.
- Effective data rates of up to 20M bits/second.
- Four selectable industry-standard polynomials: IBM 56- and 48-bit, and 35- and 32-bit polynomials.
- Three correction algorithms: full-period clock-around method, Chinese remainder theorem method, and reciprocal polynomial method.
- Allows correction of error bursts of up to 12 bits.
- Supports data stream of up to 585K bits.

General Description

The Z8065 Burst Error Processor (BEP) is a peripheral interface circuit for serial or parallel data error detection and correction. It is used in many applications, such as high performance disk systems. Four different generator polynomials are internally encoded to satisfy a broad range of applications.

Data is entered in 8-bit (byte) parallel format, and check bits are provided in the same parallel format. Write data is entered on the fly into the BEP while blocks are written to the associated disk, and check bits are extracted following the last data byte. A Read Normal mode extracts the error pattern and location while a Read High Speed mode allows direct division of data by the factors of the generator polynomial. A Divide mode generates output check bits and validates data. The Compute mode initiates a data correction process by locating the error pattern and outputting it for correction.

Operating with a single +5 V supply and a single phase clock, the Z8065 BEP supports data rates of up to 20M bits/second and data streams of over 585K bits in length. The BEP detects all errors and allows correction of error bursts of up to twelve bits in length from the first error bit location to the last. Outside of the burst error, the probability of overlooking an error is extremely small. (The expression $1/[2^N-1]$, where N is the degree of the detection polynomial selected, indicates the probability.)

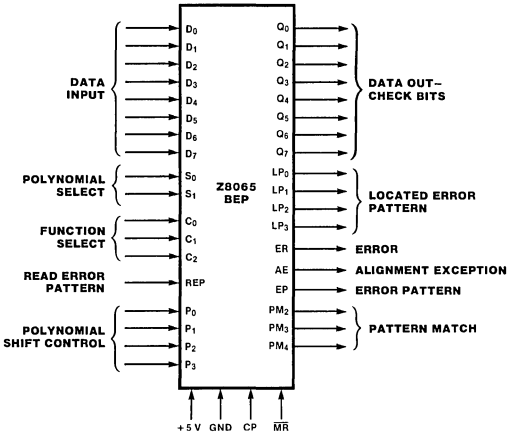


Figure 1. Pin Functions

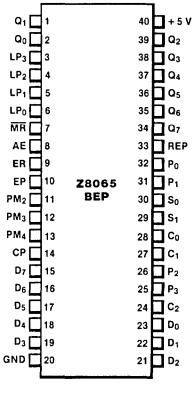


Figure 2. Pin Assignments

General Description (Continued)

The Z8065 BEP provides four standard polynomials, known as Fire codes, to satisfy a broad range of applications. These polynomials include the popular IBM 56- and 48-bit versions. During a write operation, the BEP divides the data stream by the selected polynomial using the rules of algebra in polynomial fields. The resulting remainder is the check word, which is then appended to the data stream for writing on the disc as a record. When reading the record back, the stream of data and check bytes is divided by the appropriate polynomial to obtain the syndrome.

If the syndrome is not zero, an error is detected and indicated by the ER (Error) output. This syndrome is used to correct any errors using a choice of two correction methodologies, depending on the type of polynomial selected: the "full-period clock-around" (normal method) or the "Chinese remainder theorem" (high-speed method). This extracts the burst error pattern and locates it in the data stream for external correction. A reciprocal polynomial method is used with the 48-bit code correction.

For even more flexibility, the BEP provides two read modes, normal and high-speed, which determine the correction methodology if an error is found. The normal method divides the data stream by the expanded form of the polynomial while the high-speed method performs parallel divisions using the factors of the polynomial. Both methods take the same amount of time during the read mode. However, the high-speed method can result in correction times differing by orders of magnitude.

Figure 3 shows the major sections of the Z8065 BEP. The Polynomial Divide Matrix is the heart of the BEP. The Control Logic decodes inputs to generate the necessary polynomial gating signals to the matrix. The matrix establishes connections with the Register Array such that a byte of data presented on the D_0–D_7 inputs is suitably divided by the selected generator polynomial.

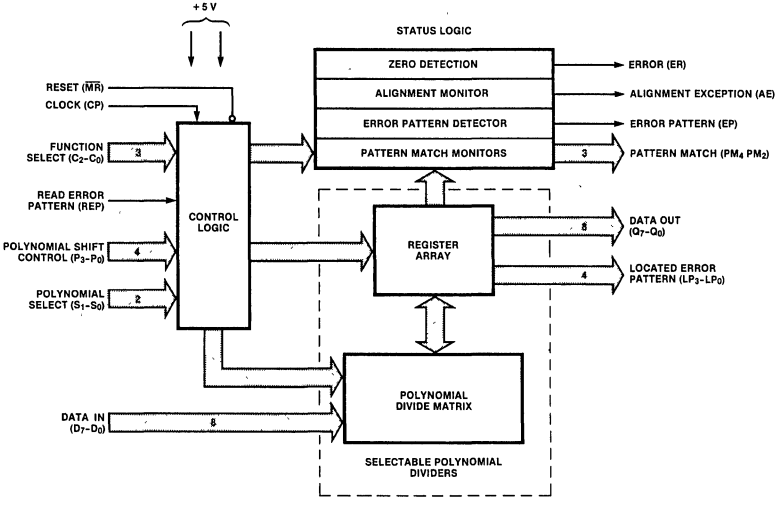


Figure 3. Block Diagram

Z8068
Z8000™ Z-DCP Data Ciphering Processor

Zilog

Product Brief

March 1981

Features

- Encrypts and decrypts data using the National Bureau of Standards encryption algorithm.
- Data rates greater than 1M bytes/second.
- Supports three standard ciphering options: Electronic Code Book, Cipher Feedback, and Chain Block.
- Three ports allow separate ports for the key, clear data, and enciphered data.
- Provides simultaneous input, output, and enciphering.
- Key parity check.
- Session keys and initialization vectors may be entered encrypted or clear.

General Description

The Z8068 Data Ciphering Processor (DCP) contains the circuitry necessary to encrypt and decrypt data using the National Bureau of Standards encryption algorithm. It is designed to be used in a variety of environments including dedicated controllers, communication concentrators, terminals, and peripheral task processors in general processor systems. The DCP provides a high throughput rate using Cipher Feedback, Electronic Code Book, or Chain Block Cipher operating modes. Separate ports are provided for key input, clear data, and enciphered data to enhance security.

The system communicates with the DCP using commands entered in the master port and through auxiliary control lines. Once set up, data can flow through the DCP at high speeds because input, output, and ciphering activities are all performed concurrently. External DMA control can easily be used to enhance throughput in some system configurations. This device is designed to interface directly to the Z-Bus.

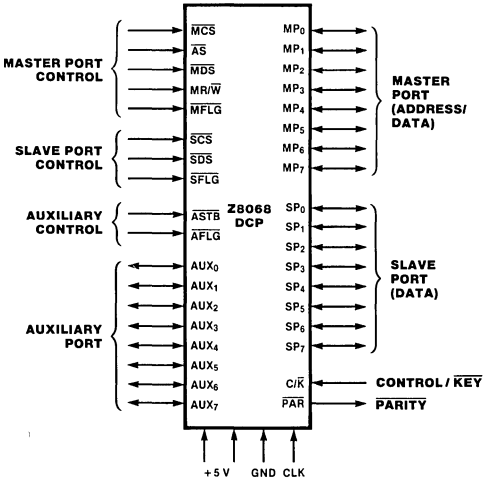


Figure 1. Pin Functions

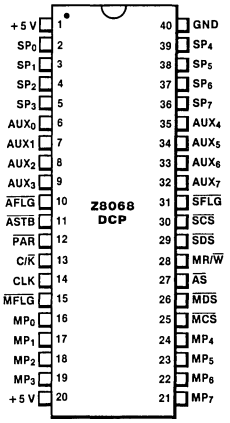


Figure 2. Pin Assignments

General Description (Continued)

The Z8068 can be configured in several ways: as a single-port system (Master Port only), as a dual-port system (master and slave) with either the master used for clear data and the slave for encrypted data or vice-versa, as an encrypting device, or as a descripting device. Figure 3 shows the major functional units of the DCP. The Algorithm Processor is the heart of the Z8068. Processing of data with the ciphering algorithm can be overlapped with input and output, thus maximizing data throughput.

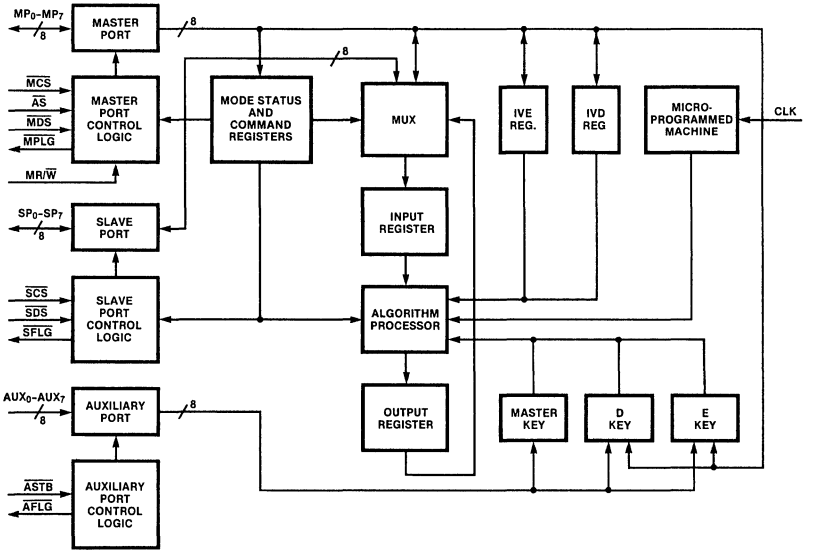


Figure 3. Block Diagram

Z8090
Z8000™ Z-UPC Universal Peripheral Controller

Zilog

Product Specification

March 1981

Features

- Complete slave microcomputer, for distributed processing Z-Bus use.
- Unmatched power of Z8 architecture and instruction set.
- Three programmable I/O ports, two with optional 2-Wire Handshake.
- Six levels of priority interrupts from eight sources: six from external sources and two from internal sources.
- Two programmable 8-bit counter/timers each with a 6-bit prescaler. Counter/Timer T0 is driven by an internal source, and Counter/Timer T1 can be driven by internal or external sources. Both counter/timers are independent of program execution.
- 256-byte register file, accessible by both the master CPU and Z-UPC, as allocated in the Z-UPC program.
- 2K bytes of on-chip ROM for efficiency and versatility.

General Description

The Z8090 Universal Peripheral Controller (Z-UPC) is an intelligent peripheral controller for distributed processing applications (Figure 3). The Z-UPC unburdens the host processor by assuming tasks traditionally done by the host (or by added hardware), such as performing arithmetic, translating or formatting data, and controlling I/O devices. Based on the Z8 microcomputer architecture and instruction set, the Z-UPC contains 2K bytes of internal program ROM, a 256-byte register file, three 8-bit I/O ports, and two counter/timers.

The Z-UPC offers fast execution time; an effective use of memory; and sophisticated interrupt, I/O, and bit manipulation. Using a powerful and extensive instruction set

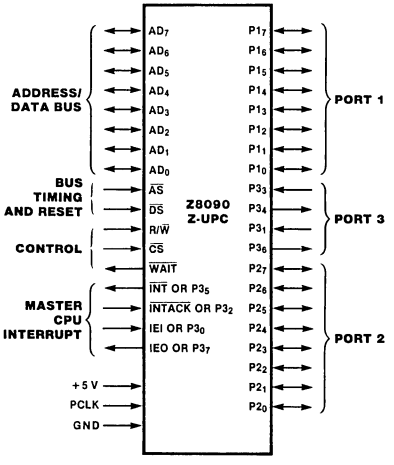


Figure 1. Z8090 Z-UPC Pin Functions

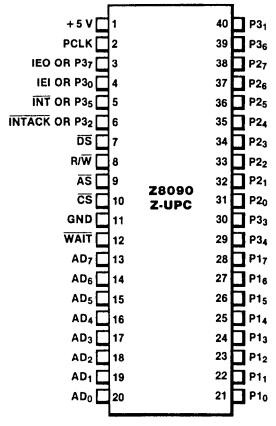


Figure 2. Z8090 Z-UPC Pin Assignments

General Description (Continued)

combined with an efficient internal addressing scheme, the Z-UPC speeds program execution and efficiently packs program code into the on-chip ROM.

An important feature of the Z-UPC is an internal register file containing I/O port and control registers accessed both by the Z-UPC program and indirectly by its associated master CPU. This architecture results in both byte and programming efficiency, because Z-UPC instructions can operate directly on I/O data without moving it to and from an accumulator. Such a structure allows the user to allocate as many general-purpose registers as the application requires for data buffers between the CPU and peripheral devices. All general-purpose registers can be used as address pointers, index registers, data buffers, or stack space.

The register file is logically divided into 16 groups, each consisting of 16 working registers. A Register Pointer is used in conjunction with short format instructions, resulting in tight, fast code and easy task switching.

Communication between the master CPU and the register file takes place via one group of 19 interface registers addressed directly by both the master CPU and the Z-UPC, or via a block transfer mechanism. Access by the master CPU is controlled by the Z-UPC to allow independence between the master CPU and Z-UPC software.

The Z-UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of input or output lines, with or without handshake, and with push-pull or open-drain outputs. Ports 1 and 2 are bit-programmable; Port 3 has four fixed inputs and four outputs.

To relieve software from coping with real-time counting and timing problems, the Z-UPC has two 8-bit hardware counter/timers, each with a fixed divide-by-four, and a 6-bit programmable prescaler. Various counting modes may be selected.

In addition to the 40-pin standard configuration, the Z-UPC is available in four special configurations:

- A 64-pin RAM development version with external interface for up to 4K bytes of RAM and 36 bytes of internal ROM permitting down-loading from the master CPU.
- A Protopack RAM version with a socket for up to 2K bytes of RAM, with 36 bytes of internal ROM permitting down-loading from the master CPU.
- A 64-pin ROM development version with external interface for up to 4K bytes of ROM and no internal ROM.
- A Protopack ROM version with a socket for 2K bytes of ROM and no internal ROM.

This range of versions and configurations makes the Z-UPC compatible with most system peripheral device control considerations.

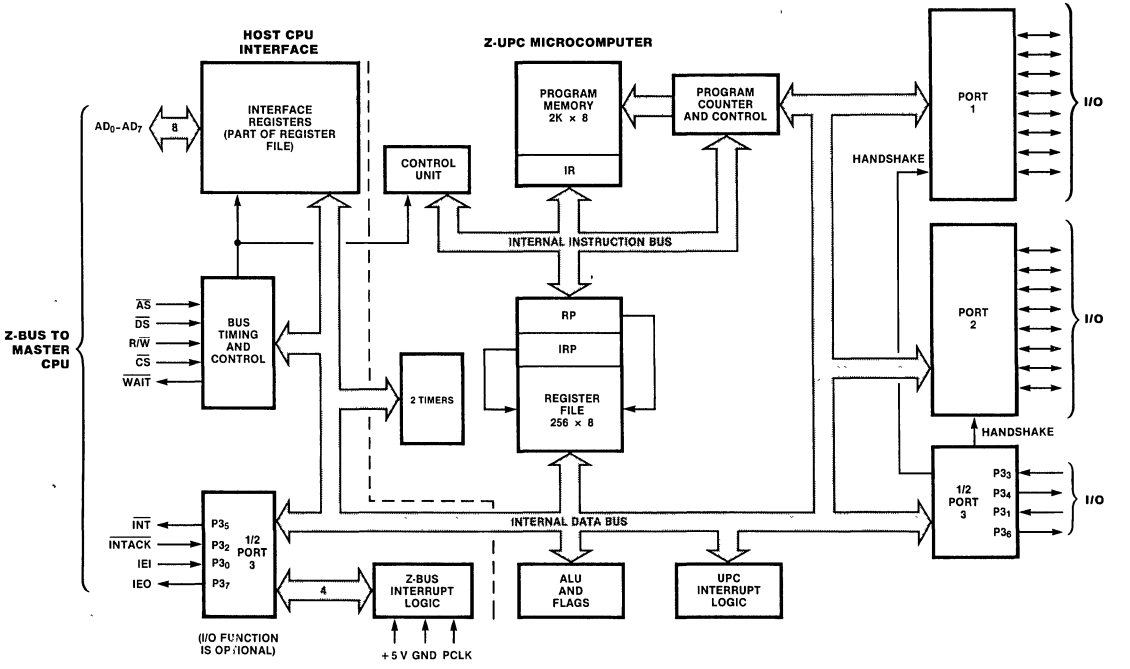


Figure 3. Functional Block Diagram

Pin Description

AD_0–AD_7. *Z-Bus Address/Data Lines* (bidirectional). These multiplexed address and data lines are used to transfer information between the master CPU and the slave Z-UPC.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ initiates the beginning of a transaction and indicates that the Address, Status, R/$\overline{W}$, and $\overline{CS}$ signals must be valid.

PCLK. *Clock* (input). TTL-compatible clock input, 4 MHz maximum. This signal does not need to be related to the master CPU clock.

$\overline{CS}$. *Chip Select* (input, active Low). A Low on this line during the rising edge of $\overline{AS}$ enables the Z-UPC to accept address or data information from the bus during a master CPU write cycle or to transmit data to the bus during a read cycle.

$\overline{DS}$. *Data Strobe* (input, active Low). $\overline{DS}$ provides timing for data movement to the bus master. A simultaneous Low on $\overline{AS}$ and $\overline{DS}$ resets the Z-UPC. It is held in reset as long as $\overline{DS}$ is Low.

$P1_0$–$P1_7$, $P2_0$–$P2_7$, $P3_0$–$P3_7$. *I/O Port Lines* (inputs/outputs, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports and may be configured in the following ways under program control:

$P1_0$–$P1_7$. *Port 1* (input/output—as output it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P2_0$–$P2_7$. *Port 2* (input/output—as output, it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P3_0$–$P3_7$. *Port 3* (four inputs, four outputs). Parallel I/O, handshake control, timer I/O, or interrupt control.

R/$\overline{W}$. *Read/Write* (input). This status signal indicates that the master CPU is executing a Read cycle if High, and a Write cycle if Low.

$\overline{WAIT}$. *Wait* (output, active Low, open-drain). When the CPU accesses the Z-UPC register file, this signal requests the master CPU to wait until the Z-UPC can complete its part of the transaction.

Functional Description

Address Space. On the 40-pin Z-UPC, all address space is committed to on-chip memory. There are 2048 bytes of mask-programmed ROM and 256 bytes of register file. I/O is memory-mapped to three registers in the register file. Only the Protopack and 64-pin versions of the Z-UPC can access external program memory. See the section entitled "Special Configurations" for complete descriptions of the Protopack and 64-pin versions.

Program Memory. Figure 4 is a map of the 2K on-chip program ROM. Even though the architecture allows addresses from 0 to 4K, behavior of the device above program address 2047 (7FFH) is not defined. The first 12 bytes of program memory are reserved for the Z-UPC interrupt vectors. For the Protopack and 64-pin versions, the address space is extended to 4096 bytes. In the RAM versions, addresses 0CH through 2FH are reserved for on-chip ROM.

Register File. This 256-byte file includes three I/O port registers (1-3H), 234 general-purpose registers (6-EEH), and 19 control, status and special I/O registers (0H, 4H, 5H, and F0-FFH). The functions and mnemonics assigned to these register address locations are shown in Figure 5. Of the 256 Z-UPC registers, 19 can be directly accessed by the master CPU; the others are accessed indirectly via the block transfer mechanism.

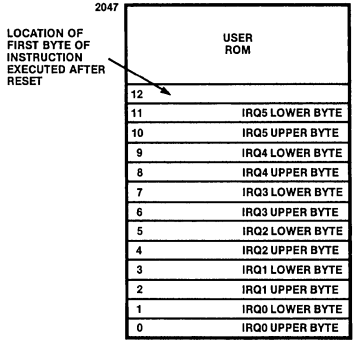


Figure 4. Program Memory Map

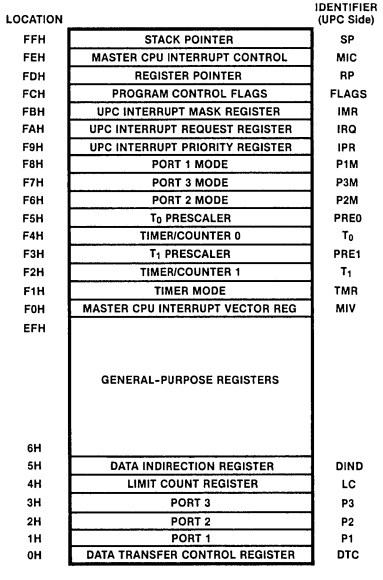


Figure 5. Register File Organization

Functional Description (Continued)

The I/O port and control registers are included in the register file without differentiation. This allows any Z-UPC instruction to process I/O or control information, thereby eliminating the need for special I/O and control instructions. All general-purpose registers can function as accumulators, address pointers, or index registers. In instruction execution, the registers are read when they are defined as sources and written when defined as destinations.

Z-UPC instructions may access registers directly or indirectly using an 8-bit address mode or a 4-bit address mode and a Register Pointer. For the 4-bit addressing mode, the file is divided into 16 working register groups, each occupying 16 contiguous locations (Figure 6). The Register Pointer (RP) in address location FDH addresses the starting point of the active working-register group, and the 4-bit register designator supplied by the instruction specifies the register within the group. Any instruction altering the contents of the register file can also alter the Register Pointer. The Z-UPC instruction set has a special Set Register Pointer (SRP) instruction for initializing or altering the pointer contents.

Stacks. An 8-bit Stack Pointer (SP), register R255, is used for addressing the stack, residing within the 234 general-purpose registers, address location 6H through EFH. PUSH and POP instructions can save and restore any register in the register file on the stack. During CALL instructions, the Program Counter is automatically saved on the stack. During Z-UPC interrupt cycles, the Program Counter and the Flag register are automatically saved on the stack. The RET and IRET instructions pop the saved values of the Program Counter and Flag register.

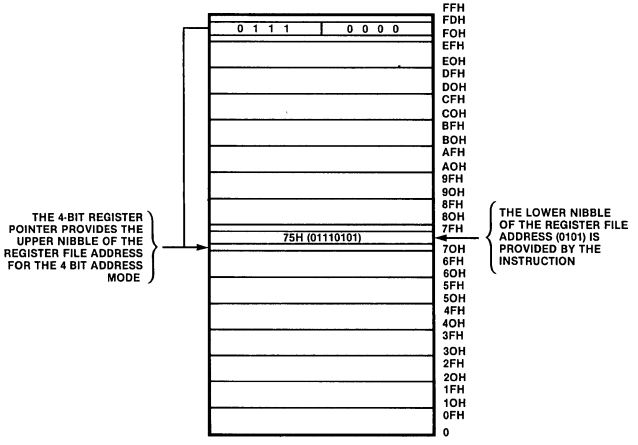


Figure 6. Register Pointer Mechanism

Ports. The Z-UPC has 24 lines dedicated to input and output. These are grouped into three ports of eight lines each and can be configured under software control as inputs, outputs, or special control signals. They can be programmed to provide Parallel I/O with or without handshake and timing signals. All outputs can have active pullups and pulldowns, compatible with TTL loads. In addition, they may be configured as open-drain outputs.

Port 1. Individual bits of Port 1 can be configured as input or output by programming Port 1 Mode register (P1M) F8H. This port is accessed by the Z-UPC program as general register 1H. It is written by specifying address 1H as the destination of any instruction used to store data in the output register. The port is read by specifying address 1H as the source of an instruction.

Port 1 may be placed under handshake control by programming Port 3 Mode register (P3M) F7H. This configures Port 3 pins $P3_3$ and $P3_4$ as handshake control lines $\overline{DAV}_1$ and RDY_1 for input handshake, or RDY_1 and $\overline{DAV}_1$ for output handshake, as determined by the direction (input or output) assigned to bit 7 of Port 1. The Port 3 Mode register also has a bit that programs Port 1 for open-drain output.

Port 2. Individual bits of Port 2 can be configured as inputs or outputs by programming Port 2 Mode register (P2M) F6H. This port is accessed by the Z-UPC program as general register 2H, and its functions and methods of programming are the same as those of Port 1. Port 3 pins $P3_1$ and $P3_6$ are the handshake lines $\overline{DAV}_2$ and RDY_2, with the direction (input or output) determined by the state of bit 7 of the port. The Port 3 Mode register also has a bit used to program Port 2 for open-drain output.

Function	Line	Direction	Signal
Handshake	$P3_1$	In	$\overline{DAV}_2/RDY_2$
	$P3_3$	In	$\overline{DAV}_1/RDY_1$
	$P3_4$	Out	$RDY_1/\overline{DAV}_1$
	$P3_6$	Out	$RDY_2/\overline{DAV}_2$
Z-UPC Interrupt Request*	$P3_0$	In	IRQ_3
	$P3_1$	In	IRQ_2
	$P3_3$	In	IRQ_1
Counter/Timer	$P3_1$	In	T_{IN}
	$P3_6$	Out	T_{OUT}
Master CPU	$P3_5$	Out	$\overline{INT}$
	$P3_2$	In	$\overline{INTACK}$
	$P3_0$	In	IEI
	$P3_7$	Out	IEO
Test Mode	$P3_5$	Out	$\overline{AS}$

*$P3_0$, $P3_1$, and $P3_3$ can always be used as UPC interrupt request inputs, regardless of the configuration programmed.

Table 1. Port 3 Control Functions

Functional Description (Continued)

Port 3. This port can be configured as I/O or control lines by programming the Port 3 Mode register. Port 3 is accessed as general register 3H. The directions of the eight data lines are fixed. Four lines, $P3_0$ through $P3_3$, are inputs, and the other four, $P3_4$ through $P3_7$, are outputs. The control functions performed by Port 3 are listed in Table 1.

Counter/Timers. The Z-UPC contains two 8-bit programmable counter/timers, each driven by an internal 6-bit programmable prescaler.

The T1 prescaler can be driven by internal or external clock sources. The T0 prescaler is driven by an internal clock source. Both counter/timers operate independently of the processor instruction sequence to relieve the program from time-critical operations like event counting or elapsed-time calculation. T0 Prescaler register (PRE0) F5H and T1 Prescaler register (PRE1) F3H can be programmed to divide the input frequency of the source being counted by any number from 1 to 64. A Counter register (F2H or F4H) is loaded with a number from 1 to 256. The corresponding counter is decremented from this number each time the prescaler reaches end-of-count. When the count is complete, the counter issues a timer interrupt request; IRQ_4 for T0 or IRQ_5 for T1. Loading either counter with a number (n) results in the interruption of the Z-UPC at the nth count.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. They can be programmed to stop upon reaching end-of-count (Single-Pass mode) or to automatically reload the initial value and continue counting (Modulo-n Continuous mode). The counters and prescalers can be read at any time without disturbing their values or changing their counts. The clock sources for both timers can be defined as any one of the following:

- Z-UPC internal clock (4 MHz maximum) divided by four.
- External clock input to Counter/Timer T1 via $P3_1$ (1 MHz maximum).
- Retriggerable trigger input for the Z-UPC internal clock divided by four.
- Nonretriggerable trigger input for the Z-UPC internal clock divided by four.
- External gate input for the Z-UPC internal clock divided by four.

Interrupts. The Z-UPC allows six interrupts from eight different sources as follows:

- Port 3 lines $P3_0$, $P3_2$, and $P3_3$.
- The master CPU(3).
- The two counter/timers.

These interrupts can be masked and globally enabled or disabled using Interrupt Mask Register (IMR) FBH. Interrupt Priority Register (IPR) F9H specifies the order of their priority. All Z-UPC interrupts are vectored.

Table 2 lists the Z-UPC's interrupt sources, their types, and their vector locations in program ROM. Interrupt Request IRQ_6 is dedicated to master CPU communications. Interrupt Requests IRQ_1, IRQ_2, and IRQ_3 are generated on the falling transitions of external inputs $P3_3$, $P3_1$, and $P3_0$. Interrupt Requests IRQ_4 and IRQ_5 are generated upon the timeout of the Z-UPC's two counter/timers. When an interrupt request is granted, the Z-UPC enters an interrupt machine cycle. This cycle disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

The Z-UPC also supports polled systems. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Following any hardware reset operation, an EI instruction must be executed to enable the setting of any interrupt request bit in the IRQ register. Interrupts must be disabled prior to changing the content of either the IPR (F9H) or the IMR (FBH). DI is the only instruction that should be used to globally disable interrupts.

Name	Source	Vector Location	Comments
IRQ_0	EOM, XERR, LERR	0,1	Internal (R0 Bits 0, 1, 2)
IRQ_1	$\overline{DAV}_1$, IRQ_1	2,3	External ($P3_3$) ↓ Edge Triggered
IRQ_2	$\overline{DAV}_2$, IRQ_2, T_{IN}	4,5	External ($P3_1$) ↓ Edge Triggered
IRQ_3	IRQ_3, IEI	6,7	External ($P3_0$) ↓ Edge Triggered
IRQ_4	T0	8,9	Internal
IRQ_5	T1	10,11	Internal

Table 2. Interrupt Types, Sources, and Vector Locations

Functional Description (Continued)

Master CPU Register File Access. There are two ways in which the master CPU can access the Z-UPC register file: direct access and block access.

Direct Access. Three Z-UPC registers—the Data Transfer Control (0H), the Master Interrupt Vector (F0H), and the Master Interrupt Control (FEH)—are mapped directly into the master CPU address space. The master CPU accesses these registers via the addresses shown in Table 3.

The master CPU also has direct access to 16 registers known as the DSC (Data, Status, Command) registers. The DSC Registers are numbered 0 through F (DSC0-DSCF). These registers can be any 16 contiguous register file registers beginning on a 16-byte boundary. The base address of the DSC register group is designated by the IRP (I/O Register Pointer), which is bits D_4–D_7 of the Data Transfer Control register (0H). Figure 7 shows how the register address is made up of the 4-bit IRP field, concatenated with the low order 4-bits of the address from the master CPU.
read or written. The Data Indirection register is incremented, and the Limit Count register is

Block Access. The master CPU may transmit or receive blocks of data via address xxx11111 (xx111111x shifted). When the master CPU accesses this address, the Z-UPC register pointed to by the Data Indirection register is decremented, for example, when the master CPU issues a read or write to address xxx11111 while the Data Indirection register contains the value 33H. The operation causes register 33H to be read or written and the Data Indirection register to be incremented to 34H.

The Limit Count register (04H) is decremented and is used to control the number of bytes to be transferred by master CPU block accesses. If the master CPU attempts a read or write to the Z-UPC after the Limit Count register reaches 0, the access is not completed, the LERR bit (D_2) of the Data Transfer Control register is set (indicating a limit error), and the LERR error causes an IRQ_0 interrupt request.

The IRP field of the Data Transfer Control register, the Data Indirection register, and the Limit Count register are not directly accessible to the master CPU and therefore must be set by the Z-UPC. This allows the Z-UPC to protect itself from master CPU errors and frees the master CPU from tracking the Z-UPC's internal data layout.

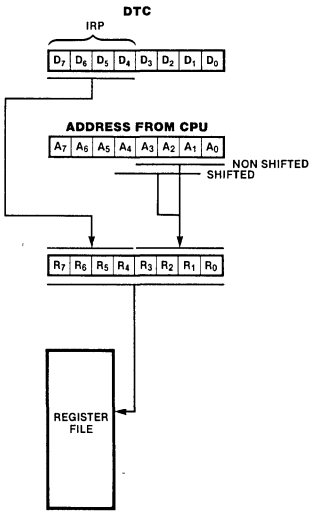


Figure 7. DSC Register Addressing Scheme

Z-UPC Address Decimal	Hex	Identifier	No-Shift Address	Shift Address
0	0H	DTC	xxx11000	xx11000x
5	5H	DIND		
@5**	@5H**		xxx11111	xx11111x
240	F0H	MIV	xxx10000	xx10000x
254	FEH	MIC	xxx11110	xx11110x
*n		DSC0	xxx00000	xx00000x
n+1		DSC1	xxx00001	xx00001x
n+2		DSC2	xxx00010	xx00010x
n+3		DSC3	xxx00011	xx00011x
n+4		DSC4	xxx00100	xx00100x
n+5		DSC5	xxx00101	xx00101x
n+6		DSC6	xxx00110	xx00110x
n+7		DSC7	xxx00111	xx00111x
n+8		DSC8	xxx01000	xx01000x
n+9		DSC9	xxx01001	xx01001x
n+10		DSCA	xxx01010	xx01010x
n+11		DSCB	xxx01011	xx01011x
n+12		DSCC	xxx01100	xx01100x
n+13		DSCD	xxx01101	xx01101x
n+14		DSCE	xxx01110	xx01110x
n+15		DSCF	xxx01111	xx01111x

x = don't care
*n is the value in the IRP x 16
**Master CPU accesses the register address in Register 5

Table 3. Master CPU/Z-UPC Register Map

Special Configurations

There are two Protopack and two 64-pin versions of the Z-UPC. These versions are identical to the 40-pin Z-UPC with the following exceptions:

- Internal ROM is totally omitted from the 64-pin development and ROM Protopack versions.
- All but 36 bytes of internal ROM are omitted from the 64-pin RAM and Protopack RAM versions.
- The memory address and data lines are buffered and brought out to external pins or to the socket on the Protopack.
- Control lines for the external memory are also provided.

The 64-pin version of the Z-UPC allows the user to prototype the system in hardware with an actual Z-UPC device and to develop the code intended to be mask-programmed into the on-chip ROM of the 40-pin Z-UPC for the production system. The 64-pin or Protopack RAM/ROM versions of the Z-UPC are extremely versatile parts. Memory space can be extended to 4K bytes on the 64-pin version by using external RAM/ROM for all but 36 bytes of the Z-UPC's memory space. This memory can then be down-loaded from the master CPU using a bootstrap program stored in the 36 bytes (C–2F). Figure 8 is a memory map for the 64-pin RAM version.

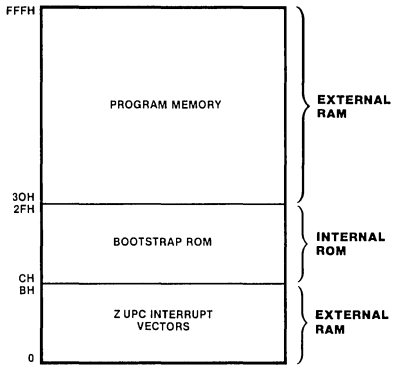


Figure 8. Z-UPC RAM Version Memory Map

64-Pin and Protopack Pin Functions. Forty of the pins on the 64-pin and Protopack versions have functions identical to those of the 40-pin version. The remaining 24 pins have additional functions described below. (Figures 9 through 11 show the 64-pin and Protopack versions' pin functions and pin assignments.)

A_0-A_{11}. *Program Memory Address Lines* (output). These lines are identical in all 64-pin and RAM versions in the Protopack. They are used to address 4K bytes of external Z-UPC memory.

D_0-D_7. *Program Data* (input). Data is read in from the external memory on these lines. The RAM version also writes external memory through this bus.

IACK. *Interrupt Acknowledge* (output, active High). This signal is active whenever an internal Z-UPC interrupt cycle is in process.

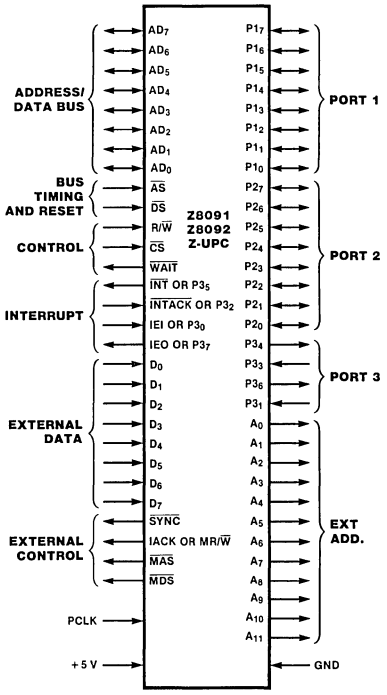


Figure 9. Z8091/Z8092 Z-UPC Pin Functions

Special Configurations (Continued)

MAS. *Memory Address Strobe* (output, active Low). This address strobe is pulsed once for each memory fetch to interface with quasi-static RAM.

MDS. *Memory Data Strobe* (output, active Low). This signal is Low during an instruction fetch or memory write.

MR/W. *Memory Read/Write* (output RAM versions only). This signal is High when the Z-UPC is fetching an instruction and Low when it is loading external memory.

SYNC. *Instruction Sync* (output, active Low). This signal is Low during the clock cycle just preceding an opcode fetch.

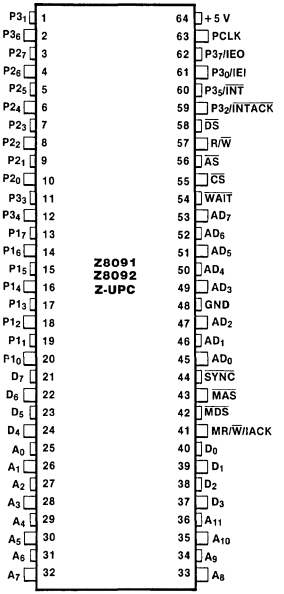


Figure 10. Z8091/Z8092 Z-UPC Pin Assignments

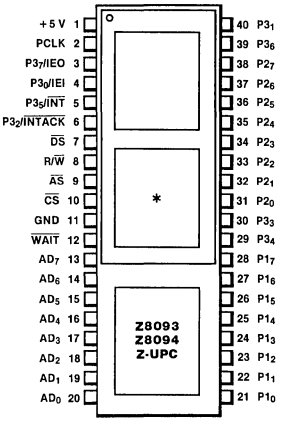


Figure 11. Z8093/Z8094 Protopack Pin Assignments

Addressing Modes

The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working-register pair address
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate

Symbols

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack Pointer (control register FFH)
PC	Program Counter
FLAGS	Flag register (control register FCH)
RP	Register Pointer (control register FDH)
IMR	Interrupt Mask register (control register FBH)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags

Control Register FCH contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

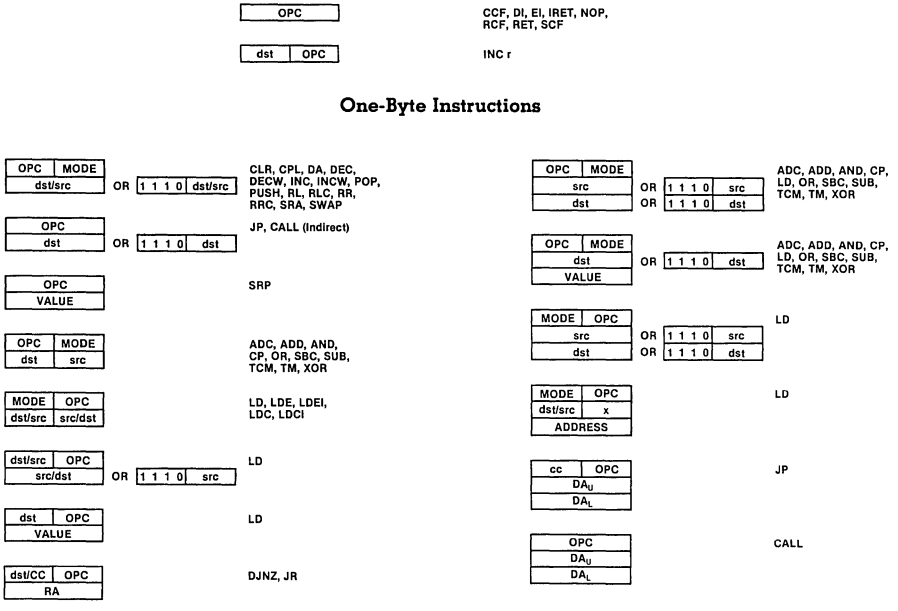
Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

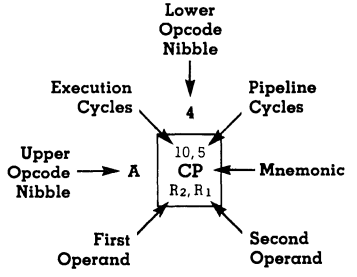
Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

Instruction Formats



Opcode Map

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, Ir_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	6,7 **SWAP** R_1	6,7 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction.

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst	R IR		90 91	* * * * - -
RLC dst	R IR		10 11	* * * * - -
RR dst	R IR		E0 E1	* * * * - -
RRC dst	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst,src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

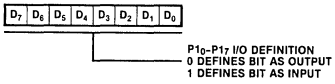
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

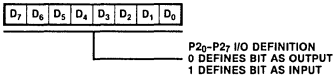
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

R248 P1M
Port 1 Mode Register
Z-UPC register address (Hex): F8



R246 P2M
Port 2 Mode Register
Z-UPC register address (Hex): F6



R247 P3M
Port 3 Mode Register
Z-UPC register address (Hex): F7

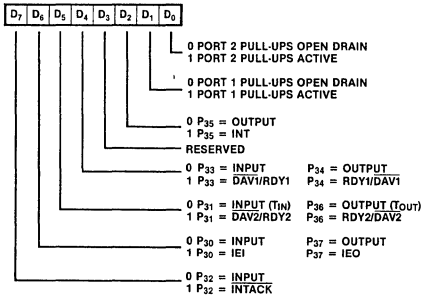
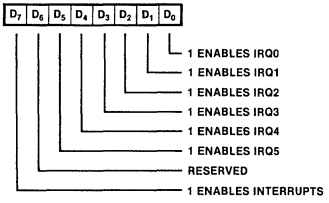
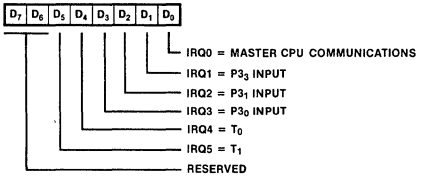


Figure 12. Port Mode Registers

R251 IMR
Interrupt Mask Register
Z-UPC register address (Hex): FB



R250 IRQ
Interrupt Request Register
Z-UPC register address (Hex): FA



R249 IPR
Interrupt Priority Register
Z-UPC register address (Hex): F9 (Write Only)

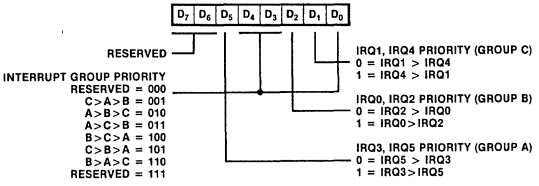
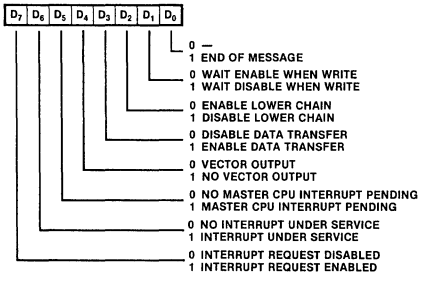


Figure 13. Interrupt Control Registers

R254 MIC
Master CPU Interrupt Control Register
Z-UPC register address (Hex): FE



R240 MIV
Master CPU Interrupt Vector Register
Z-UPC register address (Hex): F0

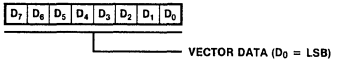
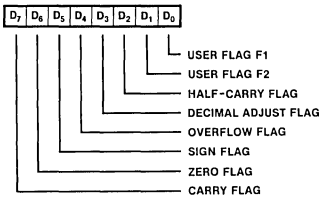
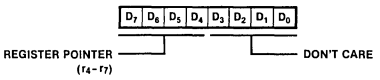


Figure 14. Master CPU Interrupt Registers

R252 FLAGS
Flag Register
Z-UPC register address (Hex): FC



R253 RP
Register Pointer
Z-UPC register address (Hex): FD



R255 SP
Stack Pointer
Z-UPC register address (Hex): FF

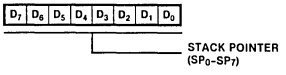
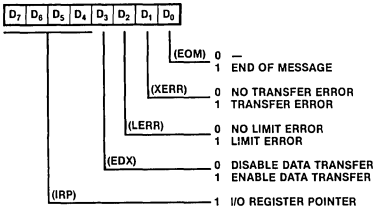
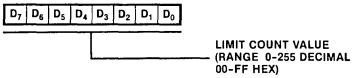


Figure 15. Z-UPC Control Registers

R0 DTC
Data Transfer Control Register
Z-UPC register address (Hex): 00



R4 LC
Limit Count Register
Z-UPC register address (Hex): 04



R5 DIND
Data Indirection Register
Z-UPC register address (Hex): 05

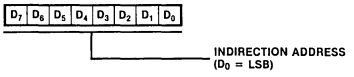
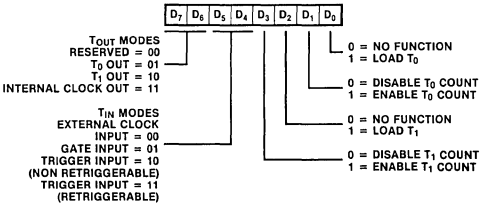
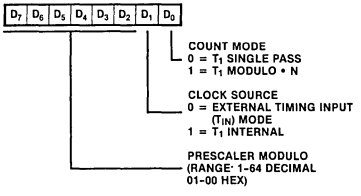


Figure 16. Master CPU-Z-UPC Data Transfer Registers

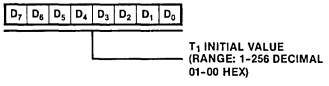
R241 TMR
Timer Mode Register
Z-UPC register address (Hex): F1



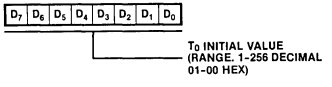
R243 PRE1
Prescaler 1 Register
Z-UPC register address (Hex): F3



R242 T1
Counter/Timer 1 Register
Z-UPC register address (Hex): F2



R244 T0
Counter/Timer 0 Register
Z-UPC register address (Hex): F4



R245 PRE0
Prescaler 0 Register
Z-UPC register address (Hex): F5

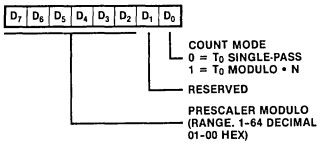


Figure 17. Z-UPC Counter/Timer Registers

Registers (Continued)

Control Register	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Comments
00_H Data Transfer Control Register	X	X	X	X	0	0	0	0	Disable data transfer from master CPU
04_H Limit Count Register	Not Defined								
05_H Data Indirection Register	Not Defined								
$F0_H$ Interrupt Vector Register	Not Defined								
$F1_H$ Timer Mode	0	0	0	0	0	0	0	0	Stops T0 and T1
$F2_H$ T0 Register	Not Defined								
$F3_H$ T0 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode
$F4_H$ T1 Register	Not Defined								
$F5_H$ T1 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode External clock source
$F6_H$ Port 2 Mode	1	1	1	1	1	1	1	1	Port 2 lines defined as inputs
$F7_H$ Port 3 Mode	0	0	0	0	X	1	0	0	Port 1, 2 open drain; $P3_5$ = INT; $P3_0$, $P3_1$, $P3_2$, $P3_3$ defined as input; $P3_4$, $P3_6$, $P3_7$ defined as output.
$F8_H$ Port 1 Mode	1	1	1	1	1	1	1	1	Port 1 lines defined as inputs
$F9_H$ Interrupt Priority	Not Defined								
FA_H Interrupt Request	X	X	0	0	0	0	0	0	Reset Interrupt Request
FB_H Interrupt Mask	0	X	X	X	X	X	X	X	Interrupts disabled
FC_H Flag Register	Not Defined								
FD_H Register Pointer	Not Defined								
FE_H Master CPU Interrupt Control Register	0	0	0	0	0	0	0	0	Master CPU interrupt disabled; wait enable when write; lower chain enabled
FF_H Stack Pointer	Not Defined								

NOTE: X means not defined.

Table 4. Control Register Reset Conditions

Absolute Maximum Ratings

Voltages on all pins (except V_{BB}) with respect to GND . . . . . -0.5 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- $V_{SS} = GND = 0\ V$
- $0°C \leq T_A \leq +70°C$

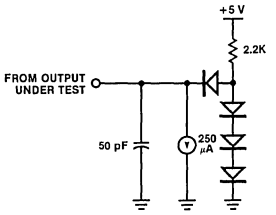


Figure 18. Test Load 1

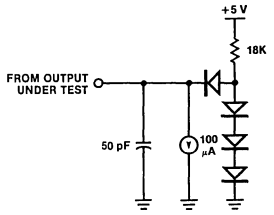


Figure 19. Test Load 2

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	2.4	V_{CC}	V		
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V		
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$	1
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ mA$	1
I_{IL}	Input Leakage	-10	10	µA	$0 \leq V_{IN} \leq +5.25\ V$	
I_{OL}	Output Leakage	-10	10	µA	$0 \leq V_{IN} \leq +5.25\ V$	
I_{CC}	V_{CC} Supply Current		180	mA		

1 For A_0-A_{11} and D_0-D_7, $\overline{MDS}$, $\overline{SYNC}$, $\overline{MAS}$, and MR/$\overline{W}$/IACK on the 64-pin versions. I_{OH} = 100 µA and I_{OL} = 1.0 mA.

Master CPU Interface Timing

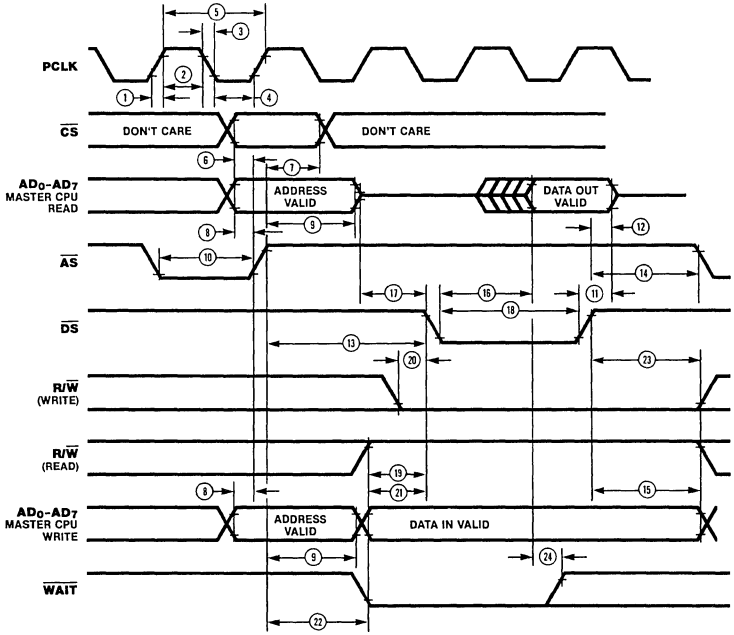
Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TrC	Clock Rise Time		20	
2	TwCh	Clock High Width	105	1855	
3	TfC	Clock Fall Time		20	
4	TwCl	Clock Low Width	105	1855	
5	TpC	Clock Period	250	2000	
6	TsCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		1
7	ThCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		1
8	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	10		1
9	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		1
10	TwAS	$\overline{AS}$ Low Width	70		
11	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Not Valid	0		
12	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70	2
13	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60	2095	
14	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		
15	ThDW(DS)	Write Data to $\overline{DS}$ Hold Time	30		1
16	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay			3
17	TdAz(DS)	Address Float to $\overline{DS}$ Delay	0		
18	TwDS	$\overline{DS}$ Low Width	390		
19	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		
20	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		
21	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		
22	TdAS(W)	$\overline{AS}$ ↓ to $\overline{WAIT}$ ↓ Valid Delay		195	
23	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	60		
24	TsDR(W)	Read Data Valid to $\overline{WAIT}$	0		

Interrupt Acknowledge Timing

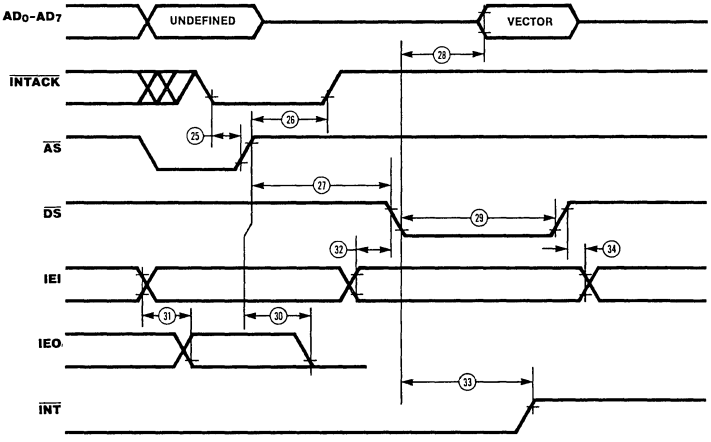
Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
25	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		
26	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		
27	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	940		
28	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		360	
29	TwDSA	$\overline{DS}$ ↓ (Acknowledge) Low Width	475		
30	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		290	
31	TdIEIf(IEO)	IEI to IEO Delay		120	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	150		
33	TdDS(INT)	$\overline{DS}$ ↓ to $\overline{INT}$ Delay		500	
34	ThIEI(DS)	IEI to $\overline{DS}$ ↑ Hold Time	100		

NOTES
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. The maximum value for TdAS(DS) does not apply to Interrupt Acknowledge transactions
3 This parameter is dependent on the state of UPC at the time of master CPU access
4. The timing characteristics given reference 2.0 V as High and 0.8 V as Low.
5. All output ac parameters use test load 1.
*Timings are preliminary and subject to change.

Master CPU Interface Timing



Interrupt Acknowledge Timing



Handshake Timing

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TsDI(DA)	Data In Setup Time	0		
2	ThDA(DI)	Data In Hold Time	230		
3	TwDA	Data Available Width	175		1,2
4	TdDAL(RY)	Data Available Low to Ready Delay Time	20 0	175	1,2 2,3
5	TdDAH(RY)	Data Available High to Ready Delay Time	 0	150	1,2 2,3
6	TdDO(DA)	Data Out to Data Available Delay Time	50		2
7	TdRY(DA)	Ready to Data Available Delay Time	0	205	2

Reset Timing

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TdRDQ(WR)	Delay from DS ↑ to AS ↓ for No Reset	40		
2	TdWRQ(RD)	Delay from DS ↑ to AS ↓ for No Reset	50		
3	TwRES	Minimum Width of AS and DS both Low for Reset	250		4

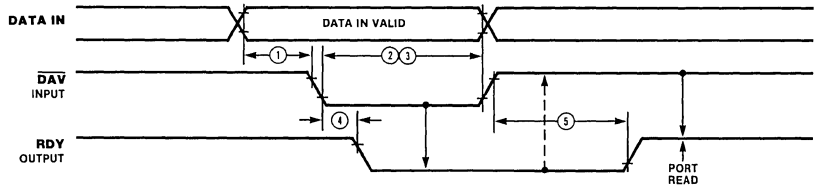
RAM Version Program Memory Timing

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TwMAS	Memory Address Strobe Width	60		5
2	TdA(MAS)	Address Valid to Memory Address Strobe ↑ Delay	30		5
3	TdMR/W (MAS)	Memory Read/Write to Memory Address Strobe ↑ Delay	30		5
4	TdMDS(A)	Memory Data Strobe ↑ to Address Change Delay	60		
5	TdMDS (MR/W)	Memory Data Strobe ↑ to Memory Read/Write Not Valid Delay	80		
6	Tw(MDS)	Memory Data Strobe Width (Write Case)	160		6
7	TdD0(MDS)	Data Out Valid to Memory Data Strobe ↓ Delay	30		5
8	TdMDS(D0)	Memory Data Strobe ↑ to Data Out Change Delay	30		5
9	Tw(MDS)	Memory Data Strobe Width (Read Case)	230		6
10	TdMDS(DI)	Memory Data Strobe ↓ to Data In Valid Delay		160	7
11	TdMAS(DI)	Memory Address Strobe ↓ to Data In Valid Delay		280	7
12	ThMDS(DI)	Memory Data Strobe ↑ to Data In Hold Time	0		
13	TwSY	Instruction Sync Out Width	160		
14	TdSY(MDS)	Instruction Sync Out to Memory Data Strobe Delay	200		
15	TwI	Interrupt Request via Port 3 Input Width	100		

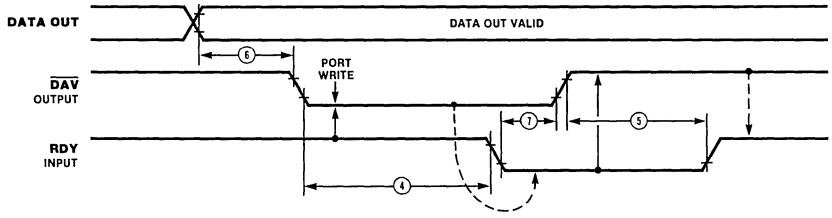
NOTES:
1. Input Handshake.
2. Test Load 1.
3. Output Handshake.
4. Internal reset signal is ½ to 2 clock delays from external reset condition.
5. Delay times are specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in input clock period must be added to the specified delay time.
6. Data strobe width is specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in three input clock periods must be added to the specified width. Data strobe width varies according to the instruction being executed.
7. Address strobe and data strobe to data in valid delay times represent memory system access times and are given for a 4 MHz input frequency.
8. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."
9. All output ac parameters use test load 2.

†Timings are preliminary and subject to change.

Handshake Timing

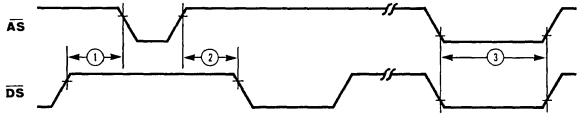


Input Handshake

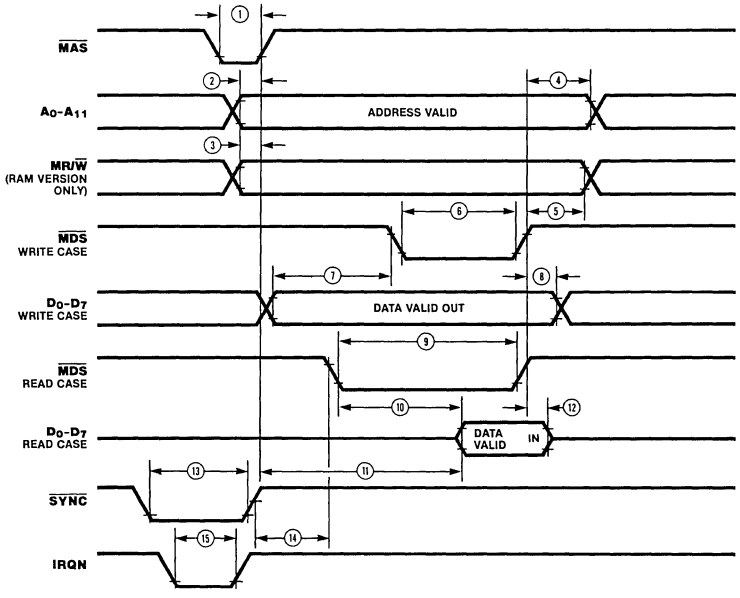


Output Handshake

Reset Timing



RAM Version Program Memory Timing



Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8090	CE	4.0 MHz	Z-UPC Universal Peripheral Controller (40-pin)
Z8090	CS	4.0 MHz	Same as above
Z8090	DE	4.0 MHz	Same as above
Z8090	DS	4.0 MHz	Same as above
Z8090	PE	4.0 MHz	Same as above
Z8090	PS	4.0 MHz	Same as above
Z8091	QS	4.0 MHz	Z-UPC External ROM-based Program Memory (64-pin)
Z8092	QS	4.0 MHz	Z-UPC External RAM-based Program Memory (64-pin)
Z8093	RS	4.0 MHz	Z-UPC 2716 EPROM Program Memory (40-pin)
Z8094	RS	4.0 MHz	Z-UPC RAM Program Memory (40-pin)

NOTES: C = Ceramic, D = Cerdip, P = Plastic, Q = Quip, R = Protopack; E = −40°C to +85°C, M = −55°C to +125°C, S = 0°C to +70°C.

More to Come

The components described in the foregoing documents exist now or are well along in their development. These components represent the first step of an ongoing commitment to support the Z8000 Family. The next step is a series of specialized processors and peripherals that includes the Z8016 DMA Transfer Controller and Z8052 CRT Controller.

Z8016 DMA Transfer Controller (DTC). This high-speed (2M byte/sec) versatile dual-channel DMA controller matches the power and addressing capability of the Z8000 CPUs. It supports a variety of system implementations, ranging from dedicated single-DMA configurations to distributed multiple-DMA configurations found in multi-user, multi-tasking environments.

The Z8016 DTC takes full advantage of the Z8000 memory management scheme because it interfaces directly to the Z8010 Memory Management Unit (MMU). Consequently, 8M bytes of logical address range are provided for each CPU address space. Alternatively, the Z8016 DTC can operate independently of the Z8010 MMU and directly address up to 16M bytes of physical address space.

The ability to self-load control parameters from memory enables chained DMA operations of different types, and provides a high degree of independence from the CPU. Memory-to-memory data transfers, as well as the more conventional data transfer between I/O and memory, can be executed. Data transfers can be in the form of single byte, double byte or word, and a number of search and match operations can be performed on the data.

Several different interrupt stimuli can be enabled under program control. These include interrupts on Terminal Count (TC), End of Process (EOP), or a number or condition bits in the Channel Mode Register. The Z8016 DTC operates within the Z8000 daisy-chain vectored-priority interrupt scheme.

Z8052 CRT Controller (CRTC). The Z8052 CRTC is a broad-application, raster scan CRT controller specifically designed to interface the Z8000 CPU to a variety of CRT displays.

Its numerous advanced features suit a wide variety of applications including general business and scientific data processing, word processing and graphics.

Text-editing software implementation is simplified by means of variable vertical and horizontal split-screen capability. Double character-cell display with vertical adjustment of character position enhances word processing as well as scientific and mathematical notation. Character justification along with mixing of fonts is achieved by external synchronization of vertical retrace and variable character clock frequency. Typewriter formatted single- or multiple-line spacing along with oversized alphanumerics or memory bit-mapped displays can be generated with variable scan lines per character row. Simple line drawing capability is also made available with the line attributes provided.

The Z8052 CRTC's programmable register architecture allows easy operating configuration and mode changes under software program control. This contrasts favorably to the time consuming and costly ROM programming methods used by other CRT controllers.

These advanced display features are achieved with the ability to change attributes dynamically on a real-time basis. The ability to change row attributes in real time allows smooth scrolling. Character attributes changing in real time allow multiple cursors and various underline combinations, all with individual blink rates. The character attributes control normal or double character cells for subscripted and superscripted display as well as the line attributes used for line drawing. This allows intermixing of forms and text displays for various applications.

Dynamic change of attributes is achieved by storing the data for changing attributes in parallel with the displayed character data. Data transfers from main memory to the CRTC are under the control of a flexible DMA function built into the Z8052. Bus activity is minimized by the inclusion of full two-line buffering on the Z8052 CRT Controller.

Universal Peripherals

Zilog

Universal Peripherals

Zilog

Two Versions Extend Range of Applications

March 1981

Zilog's Universal Peripheral Components Family is more than a group of simple I/O circuits—they are intelligent, fully programmable devices capable of performing complicated tasks independently. Their capabilities unburden the master CPU, reduce bus traffic, increase system throughput, and greatly simplify overall system hardware design requirements.

The peripheral components, where needed, are produced in two versions to increase their range of application. One version, identified by the number Z80xx, is capable of interfacing with Zilog's multiplexed Z-BUS only or with both the Z-BUS and conventional multiplexed buses. The second version, identified by the number Z85xx, is capable of interfacing with conventional non-multiplexed buses.

All of the peripheral components are extensively programmable to permit each to be tailored to its own application(s). All Z-BUS peripherals share common interrupt and bus-request structures; they can also be operated in either a priority-interrupt or polled environment.

Counting, timing, and parallel I/O transfer problems are easily solved using the **Z8036/Z8536 CIO Counter/Timer and I/O Unit.** This component has three 16-bit counter/timers, three I/O ports, and can double as a programmable priority-interrupt controller.

Data communications problems are neatly handled by the **Z8030/ Z8530 SCC Serial Communications Controller.** This device is a serial, dual-channel, multi-protocol controller which supports all popular communications formats. The SCC supports virtually all serial data transfer applications.

Interface problems with the interconnection of major components within an asynchronous, parallel processor system can be solved using the **Z8038 Z-FIO FIFO I/O Interface Unit.** This general-purpose interface unit provides expandable, bidirectional buffering between asynchronous CPUs in a parallel processing network, or between a CPU and peripheral circuits and/or devices. The Z-FIO can be used with systems having either multiplexed or non-multiplexed buses.

General-purpose control and data manipulation problems are easily handled by the **Z8034/Z8534 UPC Universal Peripheral Controller.** The UPC is a complete microcomputer designed for off-line applications. This microcomputer executes the same friendly, capable instruction set as Zilog's Z8 microcomputer; it has three I/O ports, six levels of priority-interrupt, and 2K bytes of memory on chip. The UPC is intended for applications that require an intelligent peripheral controller which can assume many of the tasks normally required of the master CPU.

Z8530 SCC Serial Communications Controller

Zilog

Product Specification

February 1981

SCC

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase-Locked Loop for clock recovery.
- Multi-protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.

General Description

The Z8530 SCC Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with conventional non-multiplexed buses. The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, Digital Phase-Locked Loops, and crystal oscillators that dramatically reduce the need for external logic.

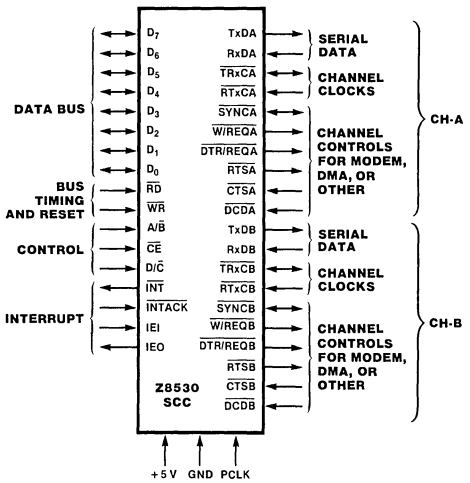


Figure 1. Pin Functions

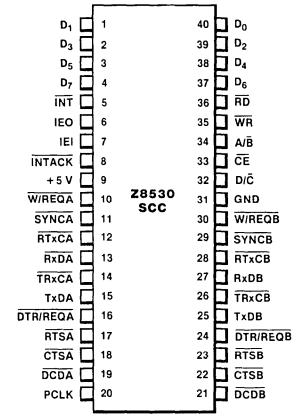


Figure 2. Pin Assignments

General Description (Continued)

The SCC handles asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-Bus daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8530 SCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

Pin Description

The following section describes the pin functions of the SCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

A/$\overline{B}$. *Channel A/Channel B Select* (input). This signal selects the channel in which the read or write operation occurs.

$\overline{CE}$. *Chip Enable* (input, active Low). This signal selects the SCC for a read or write operation.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

D/$\overline{C}$. *Data/Control Select* (input). This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

D_0-D_7. *Data Bus* (bidirectional, 3-state). These lines carry data and commands to and from the SCC.

$\overline{DTR}$/$\overline{REQA}$, $\overline{DTR}$/$\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the SCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{RD}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of PCLK.

PCLK. *Clock* (input). This is the master SCC clock used to synchronize internal signals PCLK is a TTL level signal.

$\overline{RD}$. *Read* (input, active Low). This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the $\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto

Pin Description (Continued)

Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{\text{SYNC}}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{\text{SYNC}}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{\text{TRxCA}}$, $\overline{\text{TRxCB}}$. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{\text{TRxC}}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{\text{WR}}$. *Write* (input, active Low). When the SCC is selected, this signal indicates a write operation. The coincidence of $\overline{\text{RD}}$ and $\overline{\text{WR}}$ is interpreted as a reset.

$\overline{\text{W}}$/$\overline{\text{REQA}}$, $\overline{\text{W}}$/$\overline{\text{REQB}}$. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the SCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The SCC provides two independent full-duplex channels programmable for use in any common Asynchronous or Synchronous data-communication protocol. Figure 3 and the following description briefly detail these protocols.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection

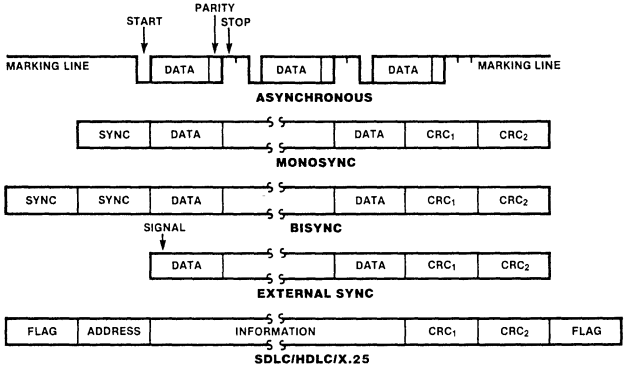


Figure 3. Some SCC Protocols

Functional Description (Continued)

mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing or error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or with an external synchronous signal. Leading sync characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports Synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s.

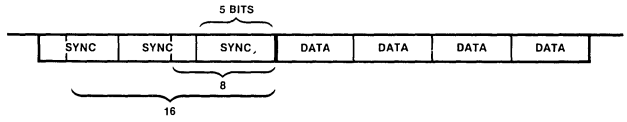


Figure 4. Detecting 5- or 7-Bit Synchronous Characters

Functional Description (Continued)

The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The SCC can be conveniently used under DMA control to provide high speed reception or transmission. In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

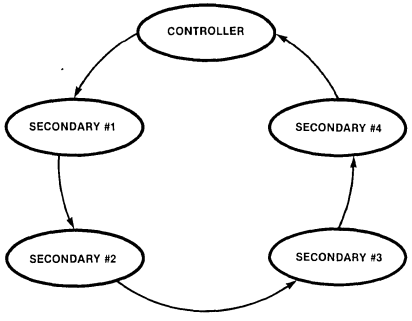


Figure 5. An SDLC Loop

Baud Rate Generator. Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{baud rate} = \frac{1}{2\,(\text{time constant} + 2) \times (\text{BR clock period})}$$

Digital Phase-Locked Loop. The SCC contains a Digital Phase-Locked-Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the

Functional Description (Continued)

incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Auto Echo and Local Loopback. The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be

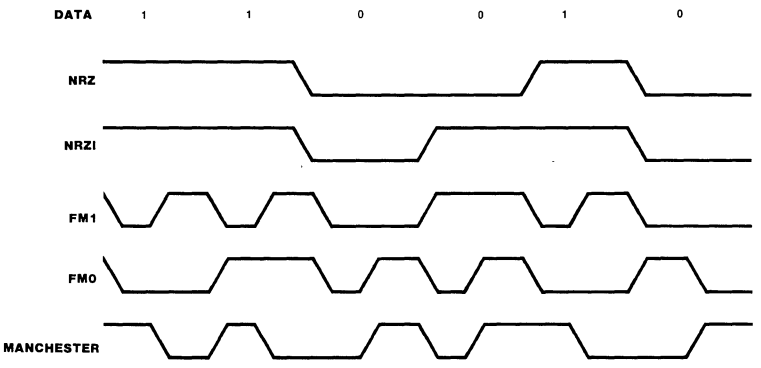


Figure 6. Data Encoding Methods

Functional Description (Continued)

read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an SCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 7). As a microprocessor peripheral, the SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{SYNC}$ pins; however, an

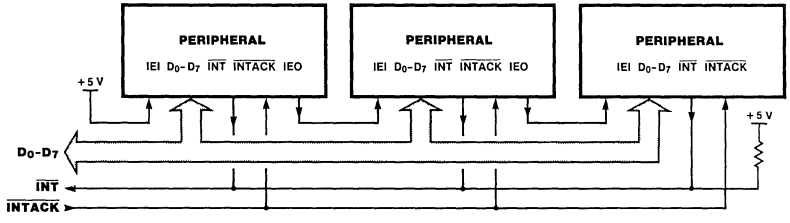


Figure 7. Interrupt Schedule

Functional Description (Continued)

External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}$/$\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}$/$\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC $\overline{\text{REQUEST}}$ output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}$/$\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two sync-character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a

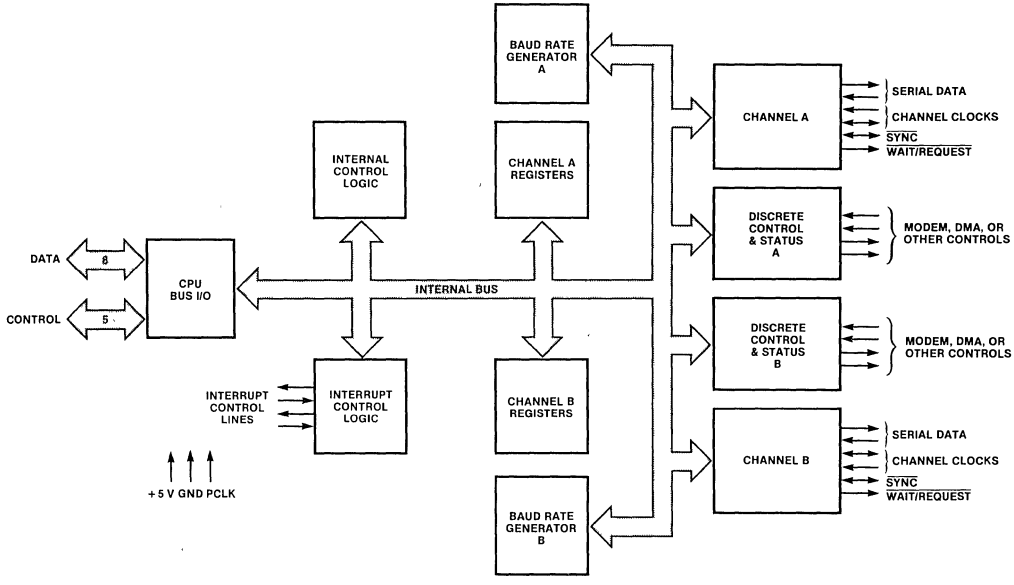


Figure 8. Block Diagram of SCC Architecture

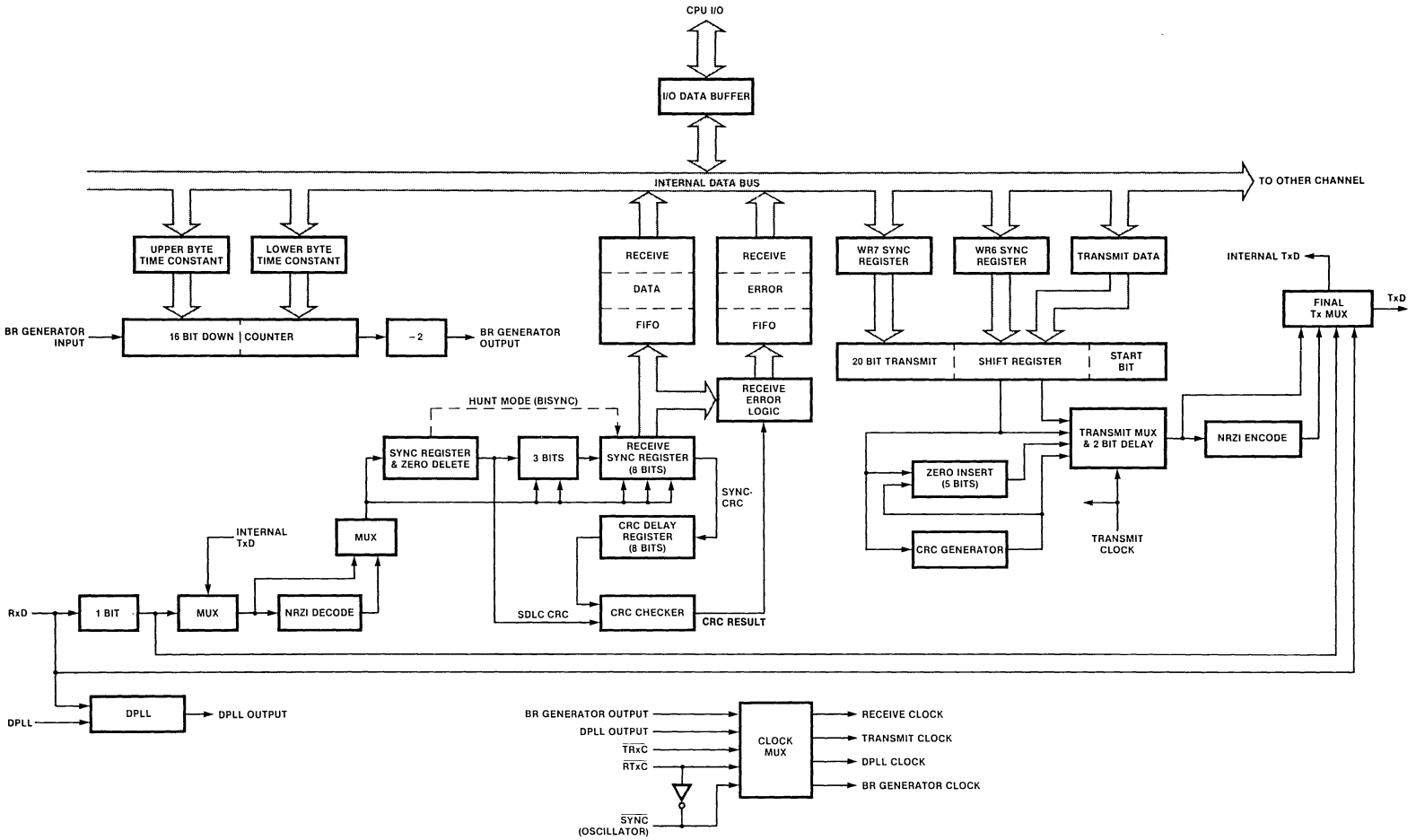


Figure 9. Data Path

Architecture (Continued)

write only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 — Write Registers 0 through 15.

RR0–RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20-bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD)

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information

Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, Register Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The SCC contains 13 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

In the SCC, register addressing is direct for the data registers only, which are selected by a High on the D/$\overline{C}$ pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the SCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

Read Registers. The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

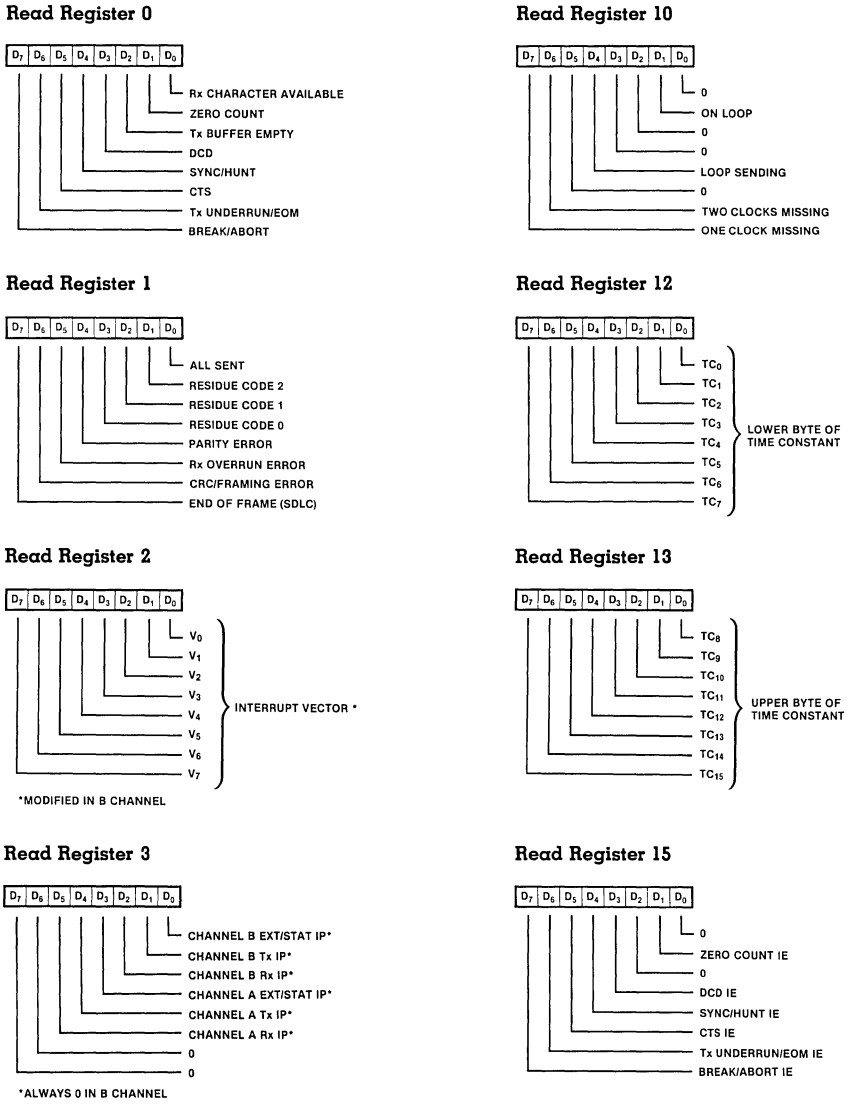
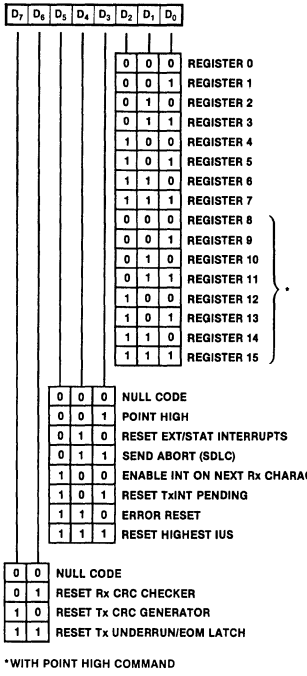


Figure 10. Read Register Bit Functions

Programming (Continued)

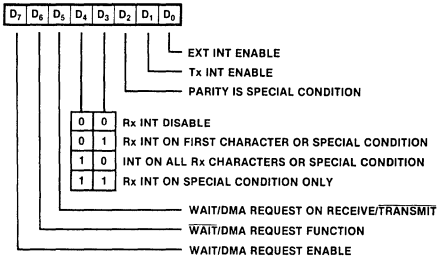
Write Registers. The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

Write Register 0

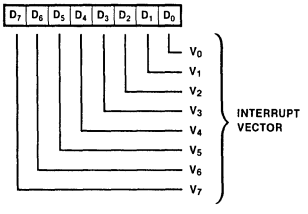


*WITH POINT HIGH COMMAND

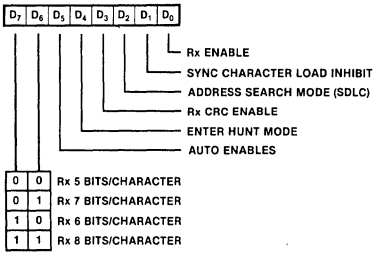
Write Register 1



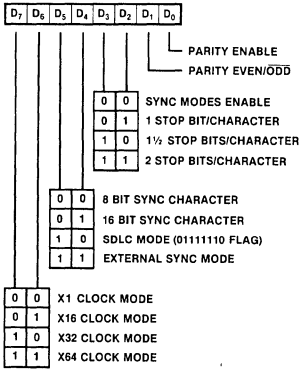
Write Register 2



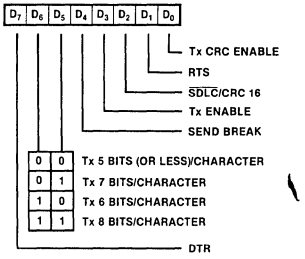
Write Register 3



Write Register 4



Write Register 5



Write Register 6

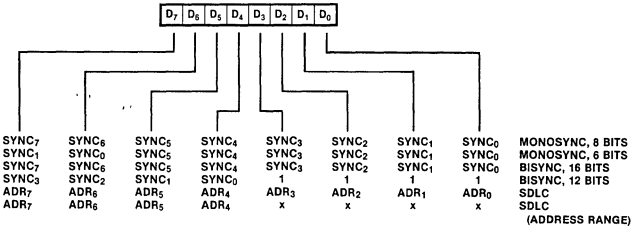
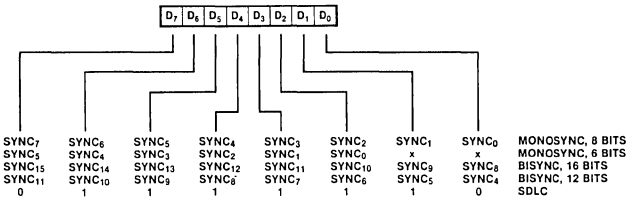
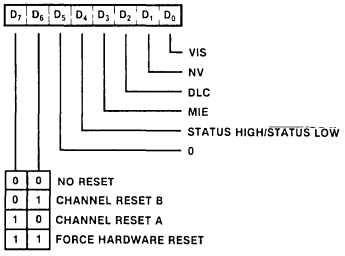


Figure 11. Write Register Bit Functions

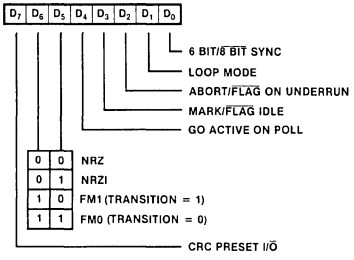
Write Register 7



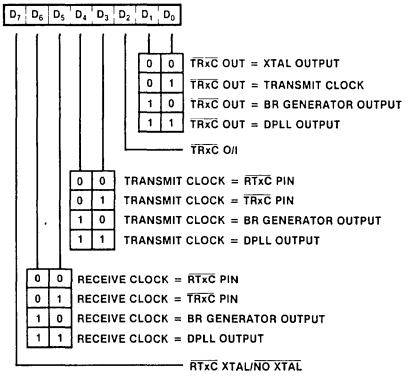
Write Register 9



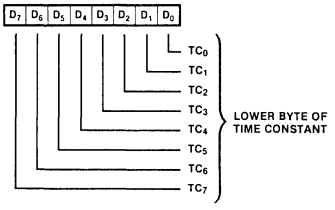
Write Register 10



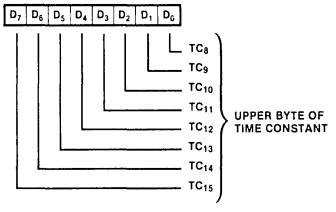
Write Register 11



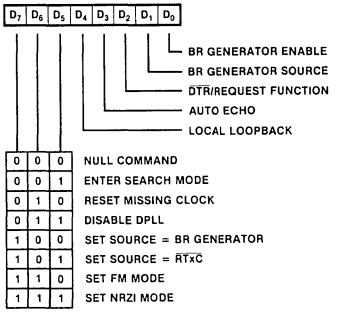
Write Register 12



Write Register 13



Write Register 14



Write Register 15

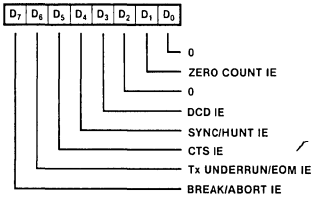


Figure 11. Write Register Bit Functions (Continued)

Timing

The SCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the rising edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the SCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the SCC. This time must be at least 6 PCLK cycles plus 200 ns.

Read Cycle Timing. Figure 12 illustrates Read cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls or if it rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

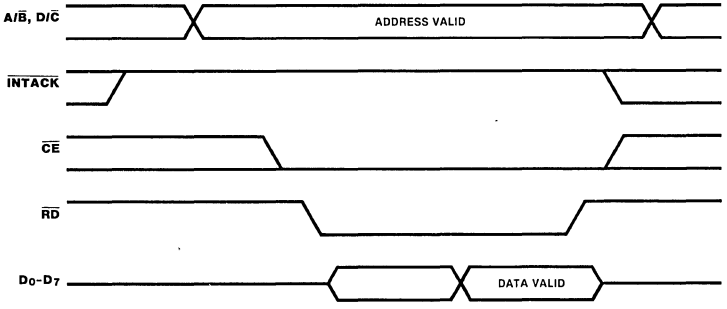


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or if it rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened.

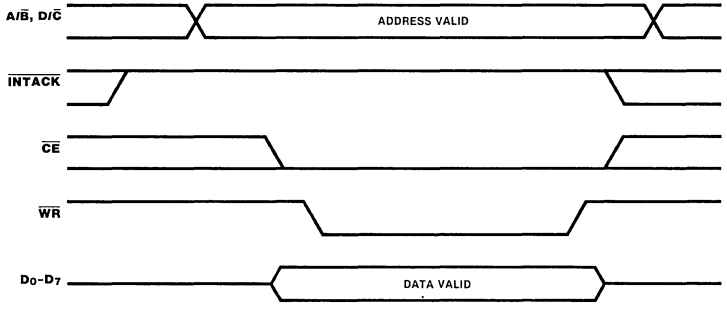


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. Between the time $\overline{INTACK}$ goes Low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{RD}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0-D_7 and it then sets the appropriate Interrupt-Under-Service latch internally.

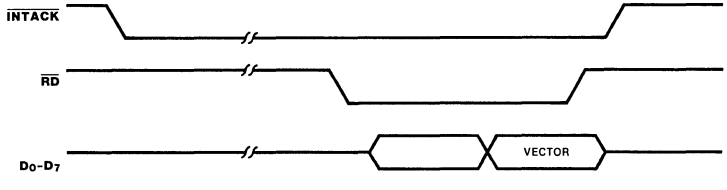


Figure 14. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in 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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

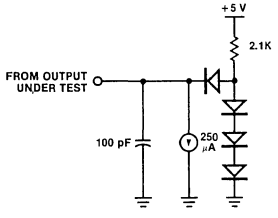


Figure 15. Standard Test Load

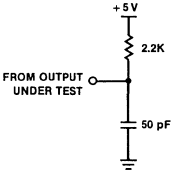


Figure 16. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

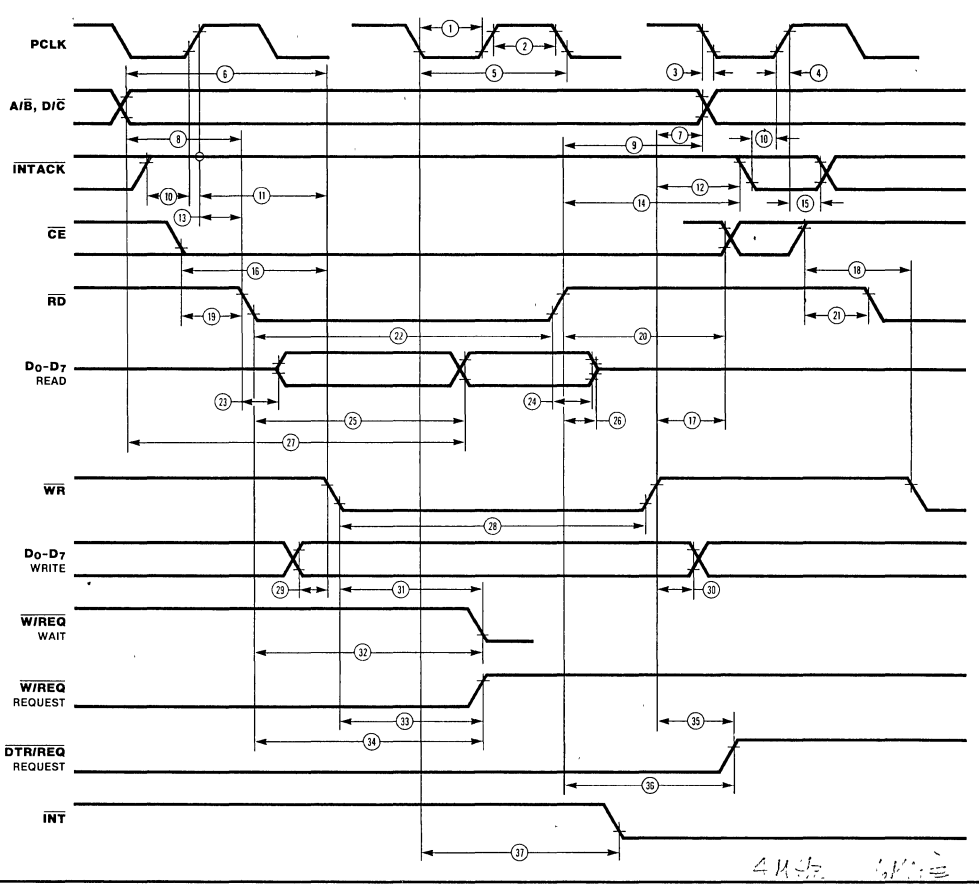
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

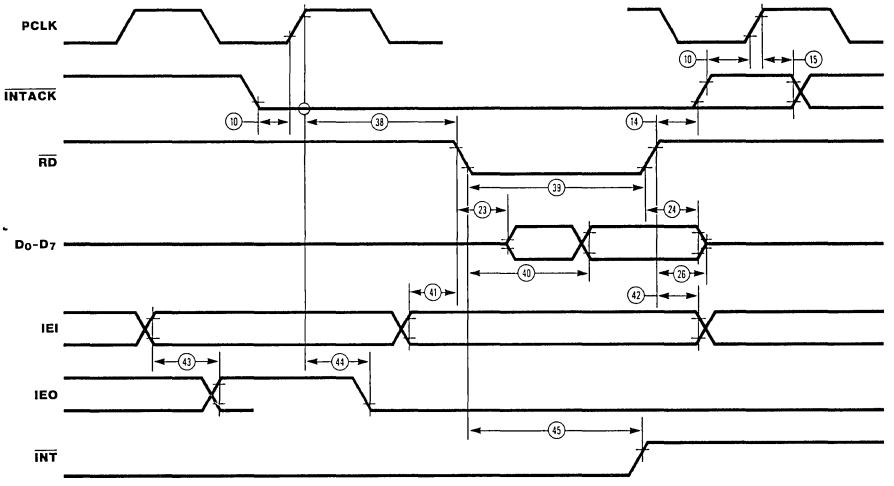
Read and Write Timing



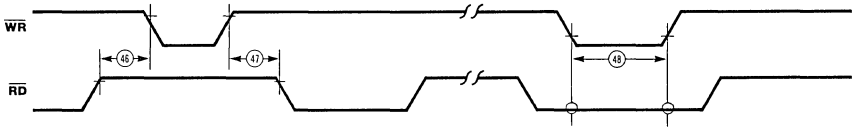
Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
1	TwPC1	PCLK Low Width	105	2000	
2	TwPCh	PCLK High Width	105	2000	
3	TfPC	PCLK Fall Time		20	
4	TrPC	PCLK Rise Time		20	
5	TcPC	PCLK Cycle Time	250	4000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		
10	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	0		
11	TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		
13	TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	100		
16	TsCE1(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		
19	TsCE1(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		1
22	TwRD1	$\overline{RD}$ Low Width	390		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		255	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70	2

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions
2. Float delay is defined as the time required for a ±0.5 V change in the output with a maximum dc load and minimum ac load.

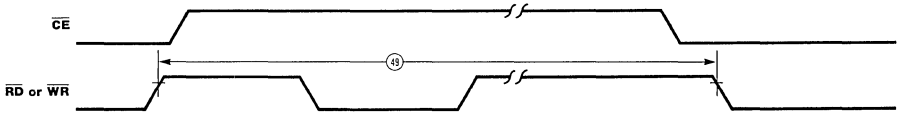
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing



Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		590	
28	TwWRl	$\overline{WR}$ Low Width	390		
29	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	0		
30	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		
31	TdWR(W)	$\overline{WR}$ ↓ to Wait Valid Delay		240	4
32	TdRD(W)	$\overline{RD}$ ↓ to Wait Valid Delay		240	4
33	TdWRf(REQ)	$\overline{WR}$ ↓ to W/$\overline{REQ}$ Not Valid Delay		240	
34	TdRDf(REQ)	$\overline{RD}$ ↓ to W/$\overline{REQ}$ Not Valid Delay		240	
35	TdWRr(REQ)	$\overline{WR}$ ↑ to $\overline{DTR}$/$\overline{REQ}$ Not Valid Delay		5TcPC +300	
36	TdRDr(REQ)	$\overline{RD}$ ↑ to $\overline{DTR}$/$\overline{REQ}$ Not Valid Delay		5TcPC +300	
37	TdPC(INT)	PCLK ↓ to $\overline{INT}$ Valid Delay		500	4
38	TdIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Delay			5
39	TwRDA	$\overline{RD}$ (Acknowledge) Width	285		
40	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		190	
41	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	120		
42	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	0		
43	TdIEI(IEO)	IEI to IEO Delay Time		120	
44	TdPC(IEO)	PCLK ↑ to IEO Delay		250	
45	TdRDA(INT)	$\overline{RD}$ ↓ to $\overline{INT}$ Inactive Delay		500	4
46	TdRD(WRQ)	$\overline{RD}$ ↑ to $\overline{WR}$ ↓ Delay for No Reset	30		
47	TdWRQ(RD)	$\overline{WR}$ ↑ to $\overline{RD}$ ↓ Delay for No Reset	30		
48	TwRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset	250		
49	Trc	Valid Access Recovery Time	6TcPC +200		3

NOTES:

3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.

*Timings are preliminary and subject to change.

4 & 6 MHz

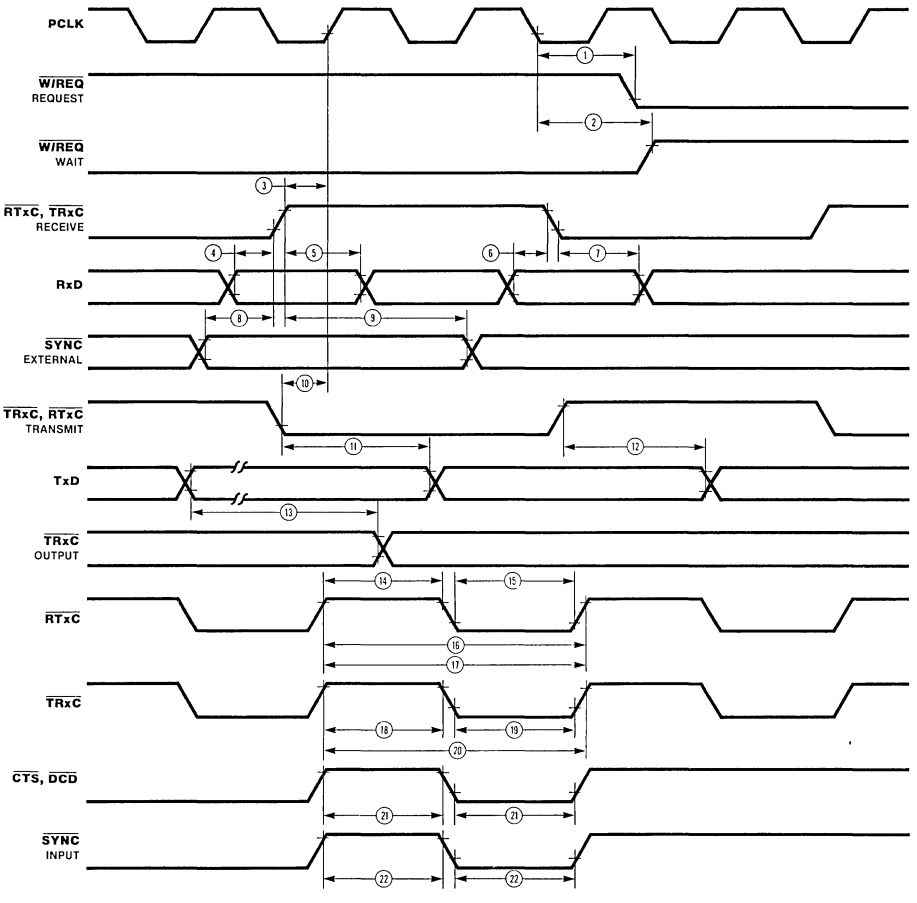
General Timing

Number	Symbol	Parameters	Min(ns)	Max(ns)	Notes*
1	TdPC(REQ)	PCLK ↓ to $\overline{W/REQ}$ Valid Delay		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350	
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time	50		1,4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Setup Time (X1 Mode)	0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode)	150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode)	0		1,5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150		1,5
8	TsSY(RXC)	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Setup Time	-200		1
9	ThSY(RXC)	$\overline{SYNC}$ to $\overline{RXC}$↑ Hold Time	3TcPC +200		1
10	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0		2,4
11	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)		300	2
12	TdTXCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode)		300	2,5
13	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)			
14	TwRTXh	$\overline{RTxC}$ High Width	180		
15	TwRTXl	$\overline{RTxC}$ Low Width	180		
16	TcRTX	$\overline{RTxC}$ Cycle Time	400		
17	TcRTXX	Crystal Oscillator Period	250	1000	3
18	TwTRXh	$\overline{TRxC}$ High Width	180		
19	TwTRXl	$\overline{TRxC}$ Low Width	180		
20	TcTRX	$\overline{TRxC}$ Cycle Time	400		
21	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ Pulse Width	200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200		

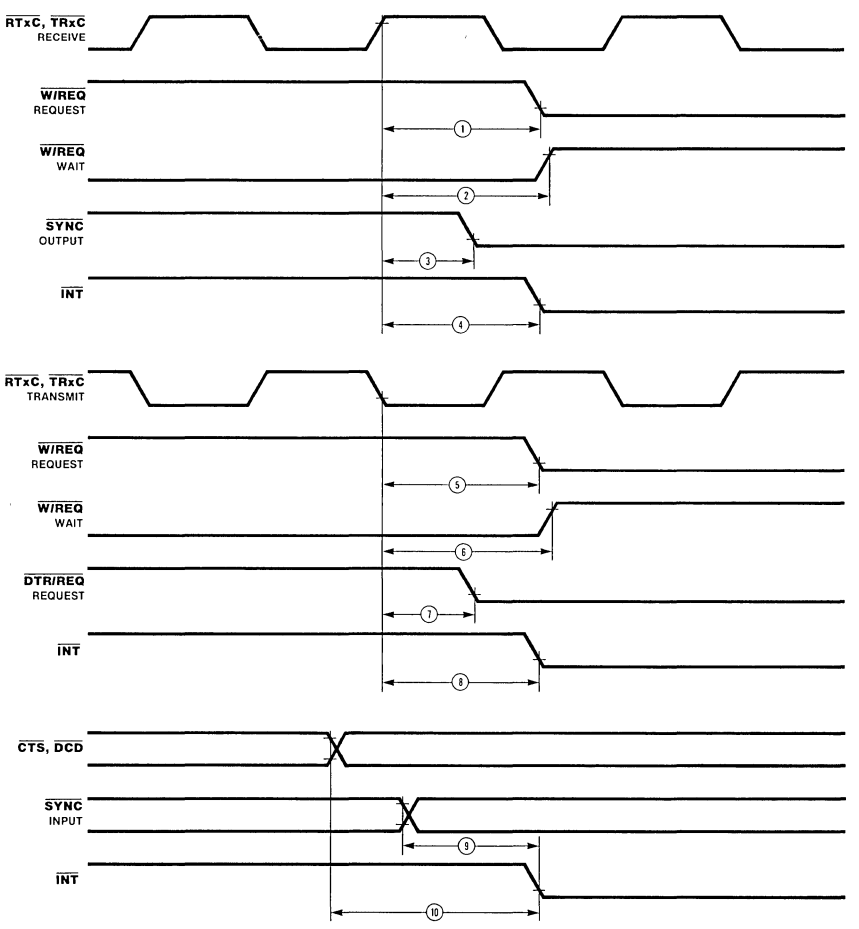
NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{SYNC}$ have 30 pF capacitors to ground connected to them.
4. Parameter applies *only* if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.

*Timings are preliminary and subject to change.

General Timing (Continued)



System Timing



Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdRXC (REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	TcPC	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	TcPC	1,2
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	TcPC	2
4	TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	TcPC	1,2
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	TcPC	3
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	TcPC	1,3
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR/REQ}$ Valid Delay	4	7	TcPC	3
8	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	TcPC	1,3
9	TdSY(INT)	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	6	TcPC	1
10	TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	6	TcPC	1

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.

*Timings are preliminary and subject to change.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8530	CE	4.0 MHz	SCC (40-pin)	Z8530A	CE	6.0 MHz	SCC (40-pin)
Z8530	CS	4.0 MHz	Same as above	Z8530A	CS	6.0 MHz	Same as above
Z8530	DE	4.0 MHz	Same as above	Z8530A	DE	6.0 MHz	Same as above
Z8530	DS	4.0 MHz	Same as above	Z8530A	DS	6.0 MHz	Same as above
Z8530	PE	4.0 MHz	Same as above	Z8530A	PE	6.0 MHz	Same as above
Z8530	PS	4.0 MHz	Same as above	Z8530A	PS	6.0 MHz	Same as above

NOTES. C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, S = 0°C to +70°C.

Z8536 CIO Counter/Timer and Parallel I/O Unit

Zilog

Product Specification

March 1981

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write.

General Description

The Z8536 CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique internal address, so that any register can be accessed in two operations. All data registers can be directly accessed in a single operation. The CIO is easily interfaced to all popular microprocessors.

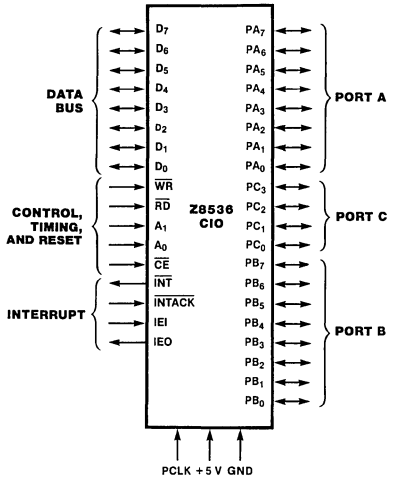


Figure 1. Pin Functions

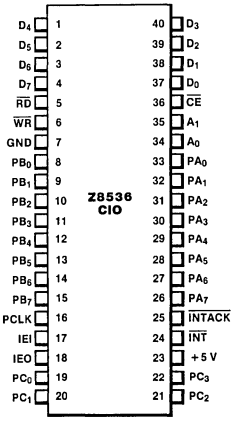


Figure 2 Pin Assignments

Pin Description

A_0-A_1. *Address Lines* (input). These two lines are used to select the register involved in the CPU transaction: Port A's Data register, Port B's Data register, Port C's Data register, or a control register.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level on this input enables the CIO to be read from or written to.

D_0-D_7. *Data Bus* (bidirectional 3-state). These eight data lines are used for transfers between the CPU and the CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This input indicates to the CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

PA_0-PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port A and external devices.

PB_0-PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0-PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO's Port C.

PCLK. *Peripheral Clock* (input, TTL-compatible). This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.

$\overline{RD}$*. *Read* (input, active Low). This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.

$\overline{WR}$*. *Write* (input, active Low). This signal indicates a CPU write to the CIO.

*When $\overline{RD}$ and $\overline{WR}$ are detected Low at the same time (normally an illegal condition), the CIO is reset.

Architecture

The CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a CPU interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose 4-bit port),

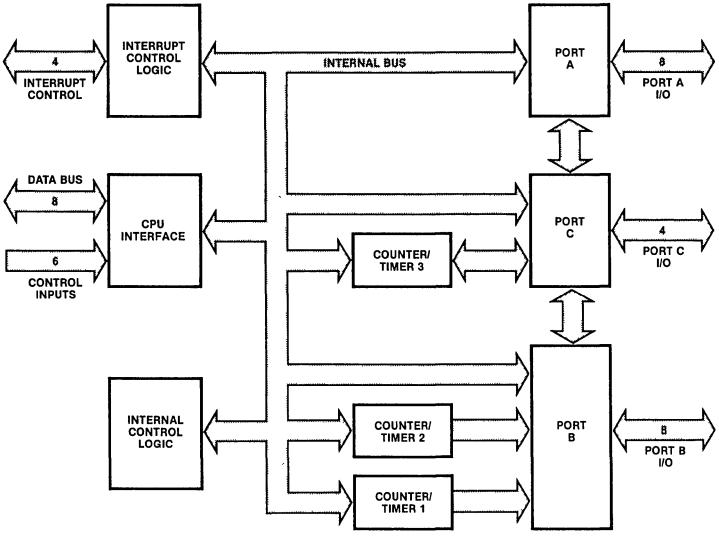


Figure 3. CIO Block Diagram

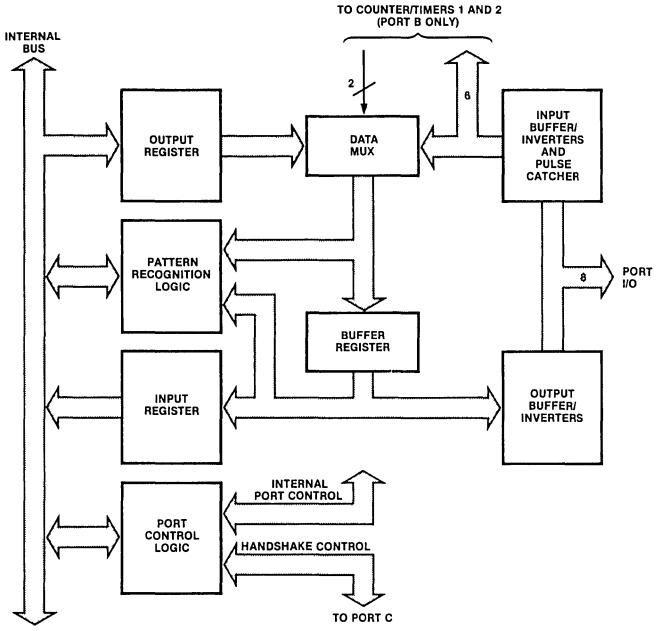


Figure 4. Ports A and B Block Diagram

three 16-bit counter/timers, an interrupt-control logic block, and the internal-control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

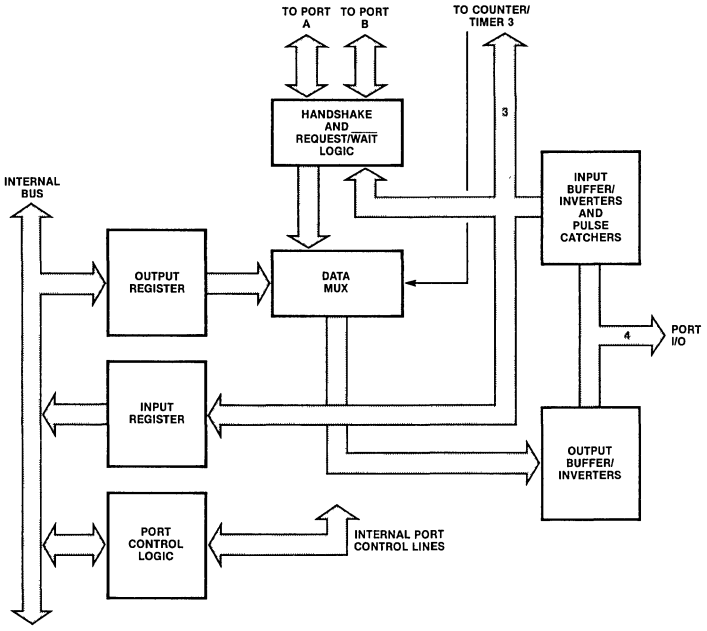


Figure 5. Port C Block Diagram

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

Architecture
(Continued)

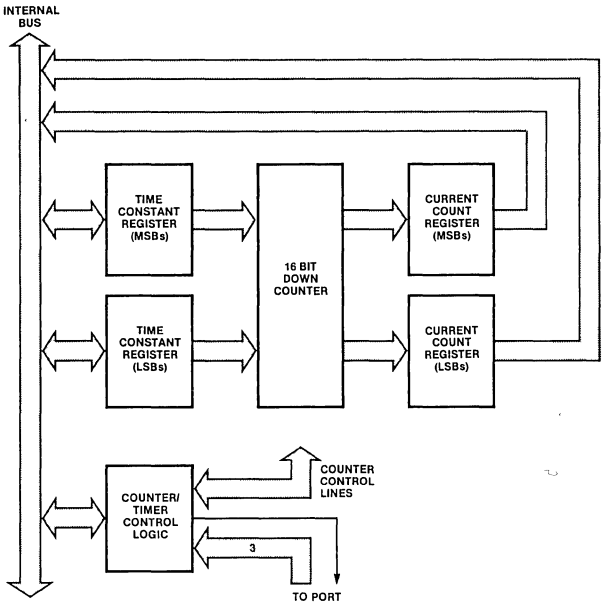


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data-direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$.

Table 1. Port C Bit Utilization

Functional Description (Continued)

goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

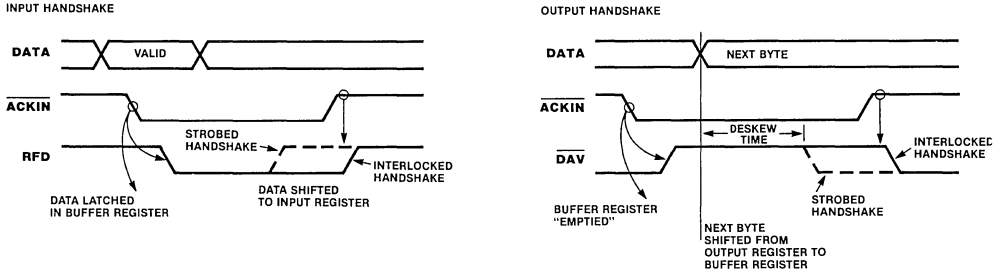


Figure 7. Interlocked and Strobed Handshakes

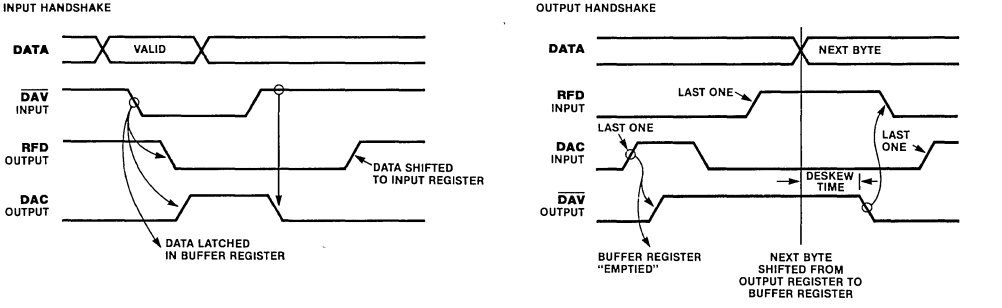


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the CPU interface. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any nonmasked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

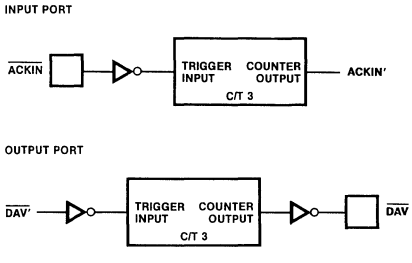


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: transparent and latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1 after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the

Functional Description (Continued)

Acknowledge cycle is initiated, the vector is frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode, and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern-Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type application when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/timer waveforms. When the Pulse mode

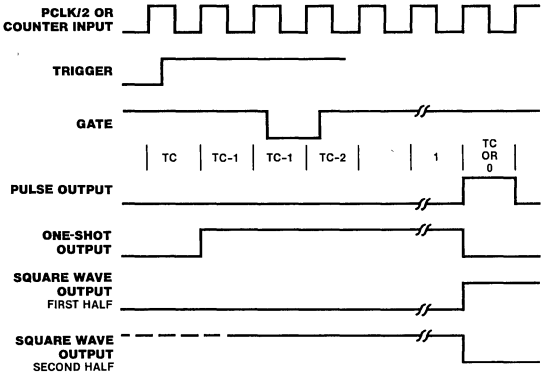


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/Single Cycle (C/$\overline{SC}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{SC}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{SC}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device, the CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS)

Functional Description (Continued)

status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the CIO's IEO is forced Low, independent of the state of the CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

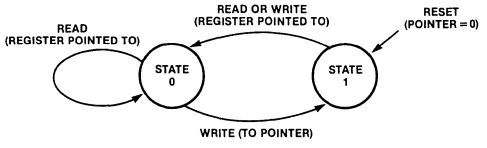
The data registers within the CIO are directly accessed by address lines A_0 and A_1 (Table 3). All other internal registers are accessed by the following two-step sequence, with the address lines specifying a control operation. First, write the address of the target register to an internal 6-bit Pointer Register; then read from or write to the target register. The Data registers can also be accessed by this method.

An internal state machine determines if accesses with A_0 and A_1 equalling 1 are to the Pointer Register or to an internal control register (Figure 11). Following any control read operation, the state machine is in State 0 (the next control access is to the Pointer Register). This can be used to force the state machine into a known state. Control reads in State 0 return the contents of the last register pointed to. Therefore, a register can be read continuously without writing to the Pointer. While the CIO is in State 1 (next control access is to the register pointed to), many internal operations are suspended—no IPs are set and internal status is frozen. Therefore, to minimize interrupt latency and to allow continuous status updates, the CIO should not be left in State 1.

The CIO is reset by forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously (normally an illegal condition) or by writing a 1 to the Reset bit. Reset disables all functions except a read from or write to the Reset bit; writes to all other bits are ignored, and all reads return 01_H. In this state, all control bits are forced to 0 and may be programmed only after clearing the Reset bit (by writing a 0 to it).

A_1	A_0	Register
0	0	Port C's Data Register
0	1	Port B's Data Register
1	0	Port A's Data Register
1	1	Control Registers

Table 3. Register Selection



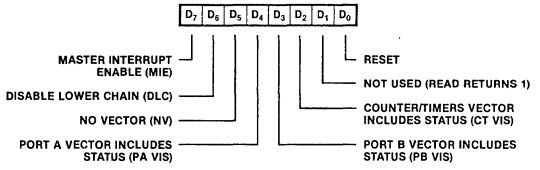
NOTE: State changes occur only when $A_0 = A_1 = 1$. No other accesses have effect

Figure 11. State Machine Operation

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

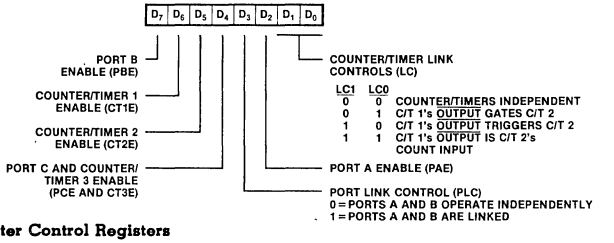
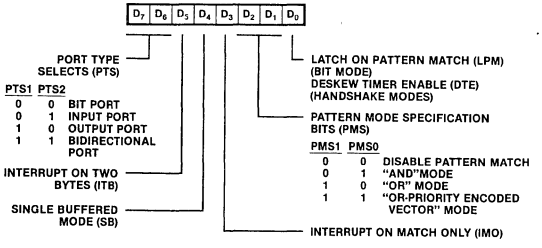


Figure 12. Master Control Registers

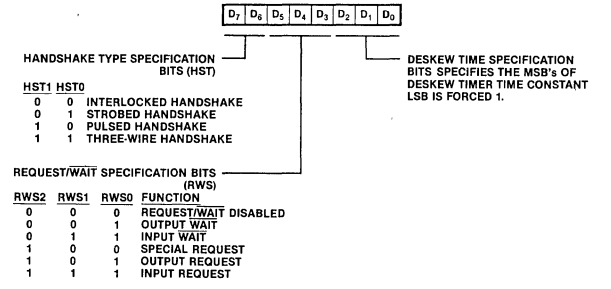
Port Mode Specification Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)



Port Handshake Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)

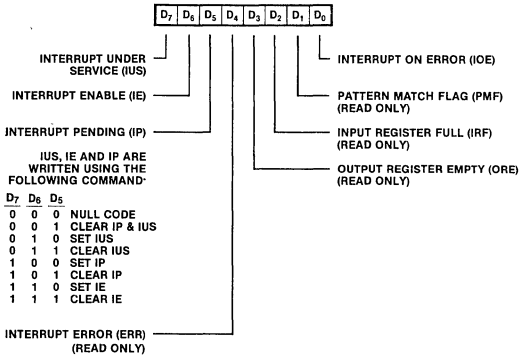
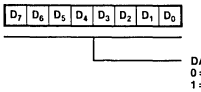


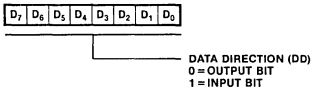
Figure 13. Port Specification Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

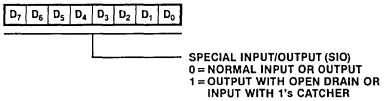
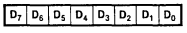


Figure 14. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A*
001110 Port B*
(Read/Write)



*These registers can be addressed directly.

Port C Data Register
Address: 001111*
(Read/Write)

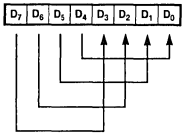
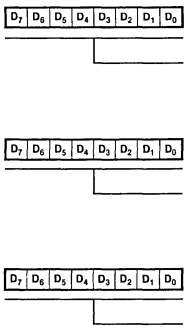


Figure 15. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)



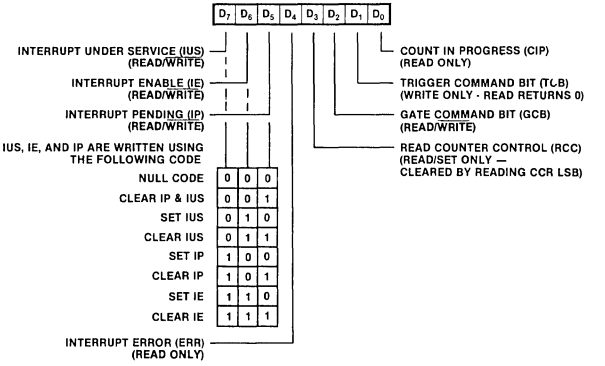
PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE-TO ZERO TRANSITION (\)
1	1	1	ZERO-TO-ONE TRANSITION (/)

Figure 16. Pattern Definition Registers

CIO

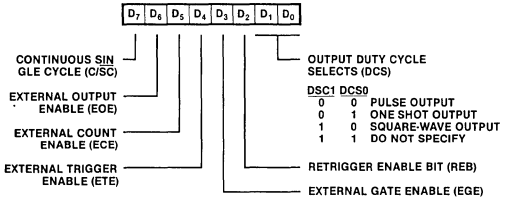
Counter/Timer Command and Status Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



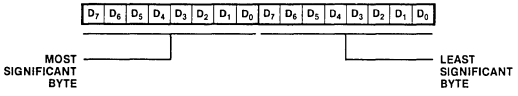
Counter/Timer Mode Specification Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

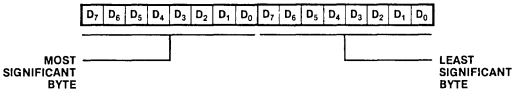
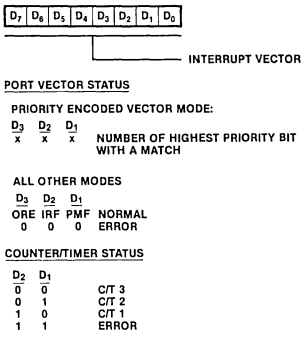


Figure 17. Counter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read only)

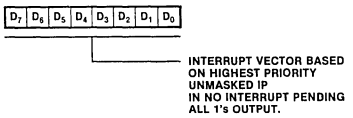


Figure 18. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address (AD_7-AD_0)	Register Name
000000XX	Master Interrupt Control
000001XX	Master Configuration Control
000010XX	Port A's Interrupt Vector
000011XX	Port B's Interrupt Vector
000100XX	Counter/Timer's Interrupt Vector
000101XX	Port C's Data Path Polarity
000110XX	Port C's Data Direction
000111XX	Port C's Special I/O Control

Most Often Accessed Registers

Address (AD_7-AD_0)	Register Name
001000XX	Port A's Command and Status
001001XX	Port B's Command and Status
001010XX	Counter/Timer 1's Control
001011XX	Counter/Timer 2's Control
001100XX	Counter/Timer 3's Control
001101XX	Port A's Data (can be accessed directly)
001110XX	Port B's Data (can be accessed directly)
001111XX	Port C's Data (can be accessed directly)

Counter/Timer Related Registers

Address (AD_7-AD_0)	Register Name
010000XX	Counter/Timer 1's Current Count-MSBs
010001XX	Counter/Timer 1's Current Count-LSBs
010010XX	Counter/Timer 2's Current Count-MSBs
010011XX	Counter/Timer 2's Current Count-LSBs
010100XX	Counter/Timer 3's Current Count-MSBs
010101XX	Counter/Timer 3's Current Count-LSBs
010110XX	Counter/Timer 1's Time Constant-MSBs
010111XX	Counter/Timer 1's Time Constant-LSBs

Counter/Timer Related Registers (Continued)

Address (AD_7-AD_0)	Register Name
011000XX	Counter/Timer 2's Time Constant-MSBs
011001XX	Counter/Timer 2's Time Constant-LSBs
011010XX	Counter/Timer 3's Time Constant-MSBs
011011XX	Counter/Timer 3's Time Constant-LSBs
011100XX	Counter/Timer 1's Mode Specification
011101XX	Counter/Timer 2's Mode Specification
011110XX	Counter/Timer 3's Mode Specification
011111XX	Current Vector

Port A Specification Registers

Address (AD_7-AD_0)	Register Name
100000XX	Port A's Mode Specification
100001XX	Port A's Handshake Specification
100010XX	Port A's Data Path Polarity
100011XX	Port A's Data Direction
100100XX	Port A's Special I/O Control
100101XX	Port A's Pattern Polarity
100110XX	Port A's Pattern Transition
100111XX	Port A's Pattern Mask

Port B Specification Registers

Address (AD_7-AD_0)	Register Name
101000XX	Port B's Mode Specification
101001XX	Port B's Handshake Specification
101010XX	Port B's Data Path Polarity
101011XX	Port B's Data Direction
101100XX	Port B's Special I/O Control
101101XX	Port B's Pattern Polarity
101110XX	Port B's Pattern Transition
101111XX	Port B's Pattern Mask

Timing

Read Cycle. At the beginning of a read cycle, the CPU places an address on the address bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When Read ($\overline{RD}$) goes Low, data from the specified register is gated onto the data bus.

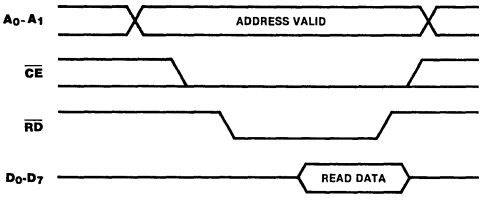


Figure 19. Read Cycle Timing

Write Cycle. At the beginning of a write cycle, the CPU places an address on the data bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When $\overline{WR}$ goes Low, data placed on the bus by the CPU is strobed into the specified CIO register.

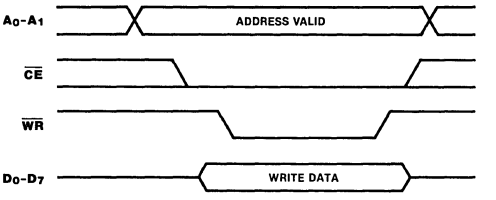


Figure 20. Write Cycle Timing

Interrupt Acknowledge. The CIO pulls its Interrupt Request ($\overline{INT}$) line Low, requesting interrupt service from the CPU, if an Interrupt Pending (IP) bit is set and interrupts are enabled. The CPU responds with an Interrupt Acknowledge cycle. When Interrupt Acknowledge ($\overline{INTACK}$) goes true and the IP is set, the CIO forces Interrupt Enable Out (IEO) Low, disabling all lower priority devices in the interrupt daisy chain. If the CIO is the highest priority device requesting service (IEI is High), it places its interrupt vector on the data bus and sets the Interrupt Under Service (IUS) bit when Read ($\overline{RD}$) goes Low.

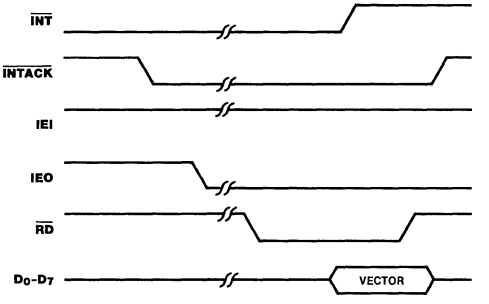


Figure 21. Interrupt Acknowledge Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in 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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

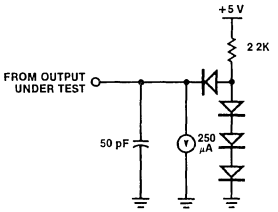


Figure 22. Standard Test Load

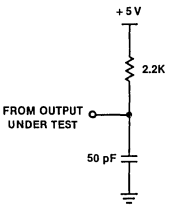


Figure 23. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		250	mA	

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

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

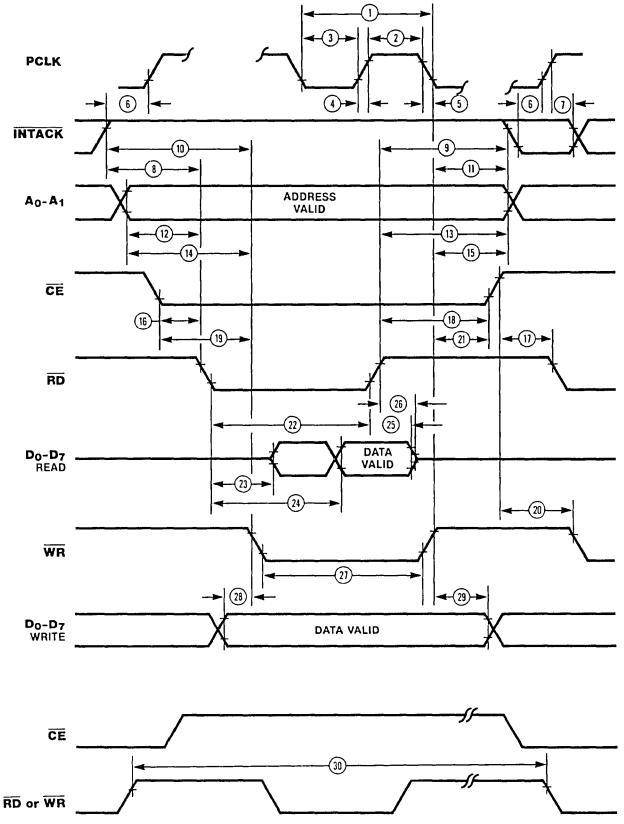
f = 1 MHz, over specified temperature range.

	Number	Symbol	Parameter	Min	Max	Units	Notes*
CPU Interface Timing	1	TcPC	PCLK Cycle Time	250	4000	ns	
	2	TwPCh	PCLK Width (High)	105	2000	ns	
	3	TwPCl	PCLK Width (Low)	105	2000	ns	
	4	TrPC	PCLK Rise Time		20	ns	
	5	TfPC	PCLK Fall Time		20	ns	
	6	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	100		ns	
	7	ThIA(PC)	$\overline{INTACK}$ To PCLK ↑ Hold Time	0		ns	
	8	TsIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		ns	1
	9	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		ns	
	10	TsIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		ns	
	11	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		ns	
	12	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		ns	
	13	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		ns	
	14	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		ns	
	15	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		ns	
	16	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		ns	1
	17	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		ns	1
	18	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		ns	1
	19	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		ns	
	20	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		ns	
	21	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		ns	
	22	TwRDl	$\overline{RD}$ Low Width	390		ns	1
	23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		ns	
	24	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		255	ns	
	25	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		ns	
	26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70	ns	2
	27	TwWRl	$\overline{WR}$ Low Width	390		ns	
	28	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	0		ns	
	29	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		ns	
	30	Trc	Valid Access Recovery Time	1000*		ns	3
Interrupt Timing	31	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		2	TcPC + ns	
	32	TdACK(INT)	$\overline{ACKIN}$ to $\overline{INT}$ Delay (Port with Handshake)		10	TcPC + ns	4
	33	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		2	TcPC + ns	
	34	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		3	TcPC + ns	
Interrupt Acknowledge Timing	35	TsIA(RDA)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Setup Time	350		ns	5
	36	TwRDA	$\overline{RD}$ (Acknowledge) Width)	350		ns	
	37	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		255	ns	
	38	TdIA(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350	ns	5
	39	TdIEI(IEO)	IEI to IEO Delay		150	ns	5
	40	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	100		ns	5
	41	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	100		ns	
	42	TdRDA(INT)	$\overline{RD}$ ↓ (Acknowledge) to $\overline{INT}$ ↑ Delay		600	ns	

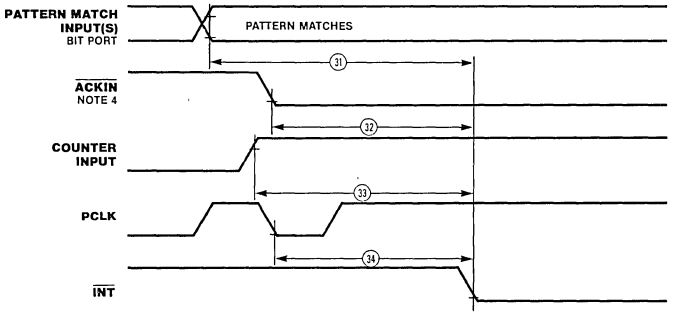
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V with minimum ac load and maximum dc load
3 Trc is 1μS or 3 TcPC, whichever is longer
4. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake
The delay is from DAC ↑ for 3-Wire Output Handshake.
5 The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{INTACK}$ ↓ to $\overline{RD}$ ↓ must be greater than the sum of TdIA(IEO) for the highest priority peripheral, TsIEI(RDA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain

*Timings are preliminary and subject to change.

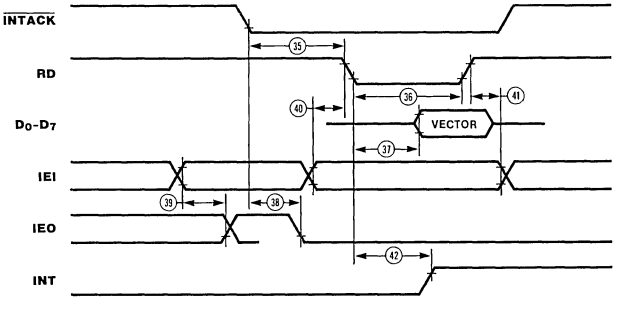
CPU Interface Timing



Interrupt Timing



Interrupt Acknowledge Timing



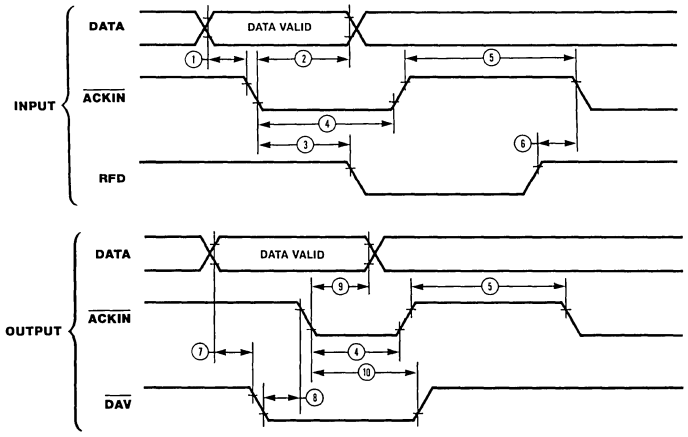
CIO

Handshake Timing

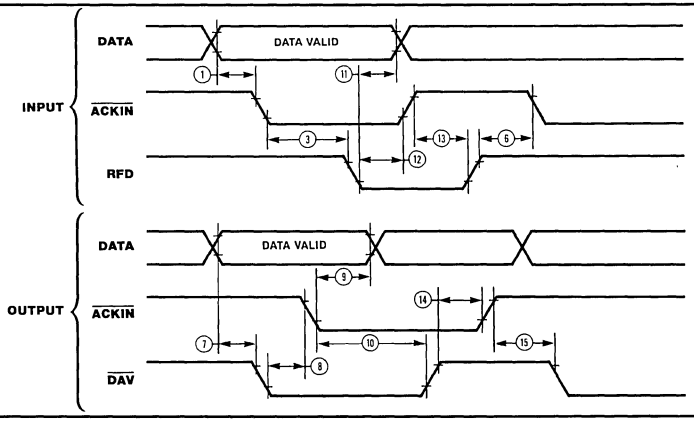
Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TsDI(ACK)	Data Input to $\overline{\text{ACKIN}}$ ↓ Setup Time	0		ns	
2	ThDI(ACK)	Data Input to $\overline{\text{ACKIN}}$ ↓ Hold Time—Strobed Handshake			ns	
3	TdACKf(RFD)	$\overline{\text{ACKIN}}$ ↓ to RFD ↓ Delay	0		ns	
4	TwACKl	$\overline{\text{ACKIN}}$ Low Width—Strobed Handshake			ns	
5	TwACKh	$\overline{\text{ACKIN}}$ High Width—Strobed Handshake			ns	
6	TdRFDr(ACK)	RFD ↑ to $\overline{\text{ACKIN}}$ ↓ Delay	0		ns	
7	TsDO(DAV)	Data Out to $\overline{\text{DAV}}$ ↓ Setup Time	25		ns	1
8	TdDAVf(ACK)	$\overline{\text{DAV}}$ ↓ to $\overline{\text{ACKIN}}$ ↓ Delay	0		ns	
9	ThDO(ACK)	Data Out to $\overline{\text{ACKIN}}$ ↓ Hold Time	2		TcPC	
10	TdACK(DAV)	$\overline{\text{ACKIN}}$ ↓ to $\overline{\text{DAV}}$ ↑ Delay	2		TcPC	
11	ThDI(RFD)	Data Input to RFD ↓ Hold Time—Interlocked Handshake	0		ns	
12	TdRFDf(ACK)	RFD ↓ to $\overline{\text{ACKIN}}$ ↑ Delay—Interlocked Handshake	0		ns	
13	TdACKr(RFD)	$\overline{\text{ACKIN}}$ ↑ ($\overline{\text{DAV}}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handshake	0		ns	
14	TdDAVr(ACK)	$\overline{\text{DAV}}$ ↑ to $\overline{\text{ACKIN}}$ ↑ (RFD ↑)—Interlocked and 3-Wire Handshake	0		ns	
15	TdACK(DAV)	$\overline{\text{ACKIN}}$ ↑ (RFD ↑) to $\overline{\text{DAV}}$ ↓ Delay—Interlocked and 3-Wire Handshake	0		ns	
16	TdDAVIf(DAC)	$\overline{\text{DAV}}$ ↓ to DAC ↑ Delay—Input 3-Wire Handshake	0		ns	
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time—3-Wire Handshake	0		ns	
18	TdDACOr(DAV)	DAC ↑ to $\overline{\text{DAV}}$ ↑ Delay—Input 3-Wire Handshake	0		ns	
19	TdDAVIr(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↓ Delay—Input 3-Wire Handshake	0		ns	
20	TdDAVOf(DAC)	$\overline{\text{DAV}}$ ↓ to DAC ↑ Delay—Output 3-Wire Handshake	0		ns	
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time—3-Wire Handshake	2		TcPC	
22	TdDACIr(DAV)	DAC ↑ to $\overline{\text{DAV}}$ ↑ Delay—Output 3-Wire Handshake	2		TcPC	
23	TdDAVOr(DAC)	$\overline{\text{DAV}}$ ↑ to DAC ↓ Delay—Output 3-Wire Handshake	0		ns	

NOTES:
1. This time can be extended through the use of the deskew timers *Timings are preliminary and subject to change

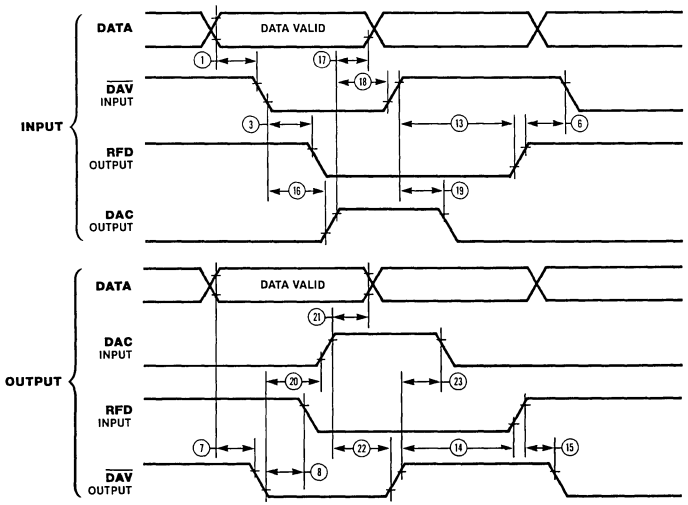
Strobed Handshake



Interlocked Handshake



3-Wire Handshake



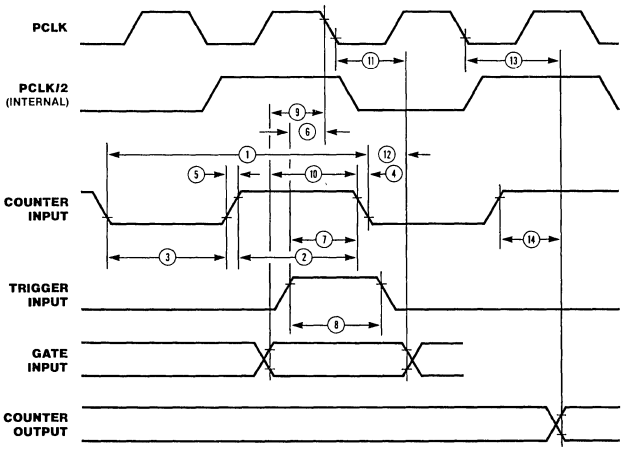
CIO

Counter/ Timer Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TcCI	Counter Input Cycle Time	500		ns	
2	TCIh	Counter Input High Width	230		ns	
3	TwCIl	Counter Input Low Width	230		ns	
4	TfCI	Counter Input Fall Time		20	ns	
5	TrCI	Counter Input Rise Time		20	ns	
6	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)			ns	1
7	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)			ns	1
8	TwTI	Trigger Input Pulse Width (High or Low)			ns	
9	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)			ns	1
10	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)			ns	1
11	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)			ns	1
12	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)			ns	1
13	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)			ns	
14	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)			ns	

NOTES:
1. These parameters must be met to guarantee trigger or gate are valid for the next counter/timer cycle.

*Timings are preliminary and subject to change.

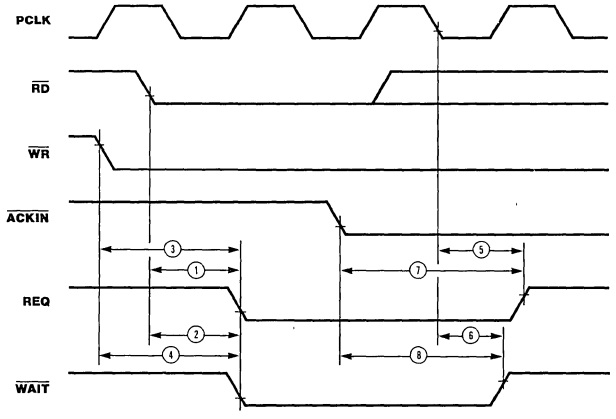


REQUEST/ WAIT Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdRD(REQ)	$\overline{RD}$ ↓ to REQ ↓ Delay			ns	
2	TdRD(WAIT)	$\overline{RD}$ ↓ to $\overline{WAIT}$ ↓ Delay			ns	
3	TdWR(REQ)	$\overline{WR}$ ↓ to REQ ↓ Delay			ns	
4	TdWR(WAIT)	$\overline{WR}$ ↓ to $\overline{WAIT}$ ↓ Delay			ns	
5	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay			ns	
6	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay			ns	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to REQ ↑ Delay			TcPC +ns	1
8	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay			TcPC +ns	1

NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake.
   The delay is from DAC ↑ for 3-Wire Output Handshake.

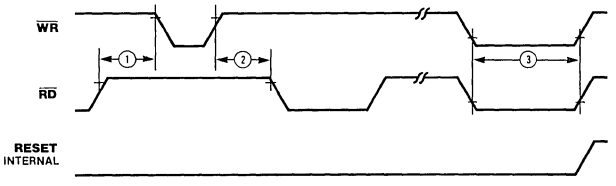
*Timings are preliminary and subject to change.



Reset Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓ for No Reset	50		ns	
2	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓ for No Reset	50		ns	
3	TwRES	Minimum Width of $\overline{RD}$ and $\overline{WR}$ both Low for Reset	250		ns	

*Timings are preliminary and subject to change

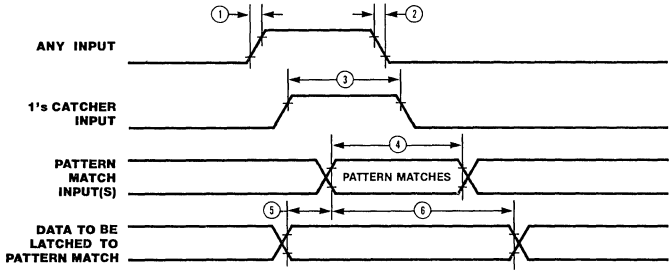


Miscellaneous Port Timing

Number	Symbol	Parameter	Min	Max	Units	Notes*
1	TrI	Any Input Rise Time		100	ns	
2	TfI	Any Input Fall Time		100	ns	
3	Twl's	1's Catcher High Width	250		ns	1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		ns	
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		ns	
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		ns	

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

*Timings are preliminary and subject to change.



Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8536	CE	4.0 MHz	CIO (40-pin)	Z8536A	CE	6.0 MHz	CIO (40-pin)
Z8536	CS	4.0 MHz	Same as above	Z8536A	CS	6.0 MHz	Same as above
Z8536	DE	4.0 MHz	Same as above	Z8536A	DE	6.0 MHz	Same as above
Z8536	DS	4.0 MHz	Same as above	Z8536A	DS	6.0 MHz	Same as above
Z8536	PE	4.0 MHz	Same as above	Z8536A	PE	6.0 MHz	Same as above
Z8536	PS	4.0 MHz	Same as above	Z8536A	PS	6.0 MHz	Same as above

NOTES C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, S = 0°C to +70°C.

Z8590
UPC Universal Peripheral Controller

Zilog

Product Specification

March 1981

Features

- Complete slave microcomputer, for distributed processing use.
- Unmatched power of Z8 architecture and instruction set.
- Three programmable I/O ports, two with optional 2-Wire Handshake.
- Six levels of priority interrupts from eight sources: six from external sources and two from internal sources.
- Two programmable 8-bit counter/timers each with a 6-bit prescaler. Counter/Timer T0 is driven by an internal source, and Counter/Timer T1 can be driven by internal or external sources. Both counter/timers are independent of program execution.
- 256-byte register file, accessible by both the master CPU and UPC, as allocated in the UPC program.
- 2K bytes of on-chip ROM for efficiency and versatility.

General Description

The Z8590 Universal Peripheral Controller (UPC) is an intelligent peripheral controller for distributed processing applications (Figure 3). The UPC unburdens the host processor by assuming tasks traditionally done by the host (or by added hardware), such as performing arithmetic, translating or formatting data, and controlling I/O devices. Based on the Z8 microcomputer architecture and instruction set, the UPC contains 2K bytes of internal program ROM, a 256-byte register file, three 8-bit I/O ports, and two counter/timers.

The UPC offers fast execution time, an effective use of memory, and sophisticated interrupt, I/O, and bit manipulation. Using a powerful and extensive instruction set

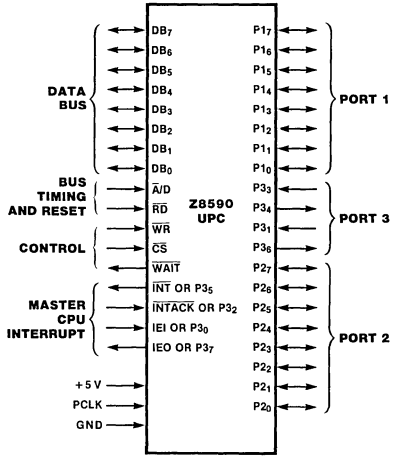


Figure 1. Z8590 UPC Pin Functions

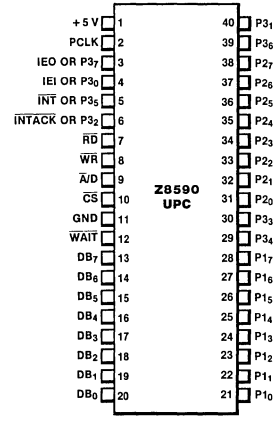


Figure 2. Z8590 UPC Pin Assignments

General Description (Continued)

combined with an efficient internal addressing scheme, the UPC speeds program execution and efficiently packs program code into the on-chip ROM.

An important feature of the UPC is an internal register file containing I/O port and control registers accessed both by the UPC program and indirectly by its associated master CPU. This architecture results in both byte and programming efficiency, because UPC instructions can operate directly on I/O data without moving it to and from an accumulator. Such a structure allows the user to allocate as many general purpose registers as the application requires for data buffers between the CPU and peripheral devices. All general-purpose registers can be used as address pointers, index registers, data buffers, or stack space.

The register file is logically divided into 16 groups, each consisting of 16 working registers. A Register Pointer is used in conjunction with short format instructions, resulting in tight, fast code and easy task switching.

Communication between the master CPU and the register file takes place via one group of 19 interface registers addressed directly by both the master CPU and the UPC, or via a block transfer mechanism. Access by the master CPU is controlled by the UPC to allow independence between the master CPU and UPC software.

The UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of input or output lines, with or without handshake, and with push-pull or open-drain outputs. Ports 1 and 2 are bit-programmable; Port 3 has four fixed inputs and four outputs.

To relieve software from coping with real-time counting and timing problems, the UPC has two 8-bit hardware counter/timers, each with a fixed divide-by-four, and a 6-bit programmable prescaler. Various counting modes may be selected.

In addition to the 40-pin standard configuration, the UPC is available in four special configurations:

- A 64-pin RAM development version with external interface for up to 4K bytes of RAM and 36 bytes of internal ROM permitting down-loading from the master CPU.
- A Protopack RAM version with a socket for up to 2K bytes of RAM, with 36 bytes of internal ROM permitting down-loading from the master CPU.
- A 64-pin ROM development version with external interface for up to 4K bytes of ROM and no internal ROM.
- A Protopack ROM version with a socket for 2K bytes of ROM and no internal ROM.

This range of versions and configurations makes the UPC compatible with most system peripheral device control considerations.

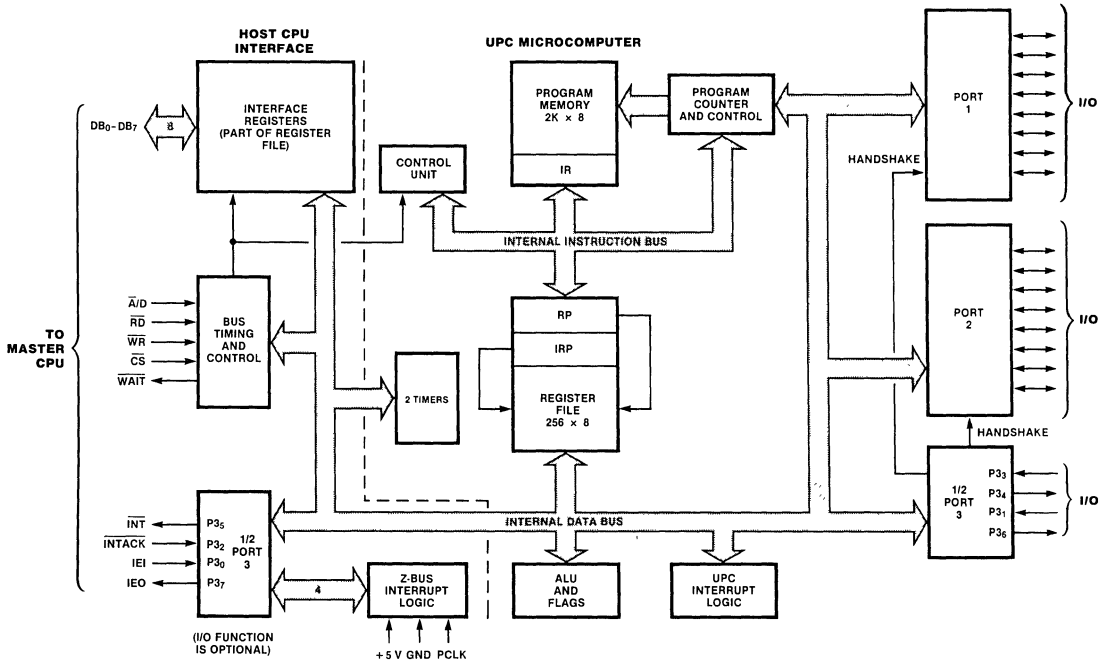


Figure 3. Functional Block Diagram

Pin Description

$\overline{A}$/D. *Address/Data* (input). A Low on this pin defines information on the data bus as an address. A High defines the information as data.

$\overline{CS}$. *Chip Select* (input, active Low). A Low enables the UPC to accept address or data information from the master CPU during a write cycle or to transmit data to the master CPU during a read cycle. This line is usually generated from higher bits of the address lines.

DB_0-DB_7. *Data Bus* (bidirectional). This bus is used to transfer address and data information between the master CPU and the UPC.

$P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (bidirectional, TTL compatible). These 24 lines are divided into three 8-bit I/O ports and may be configured in the following ways under program control:

$P1_0$-$P1_7$. *Port 1* (input/output—as output it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P2_0$-$P2_7$. *Port 2* (input/output—as output, it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P3_0$-$P3_7$. *Port 3* (four inputs, four outputs). Parallel I/O, handshake control, timer I/O, or interrupt control.

PCLK. *Clock* (input). TTL-compatible clock input, 4 MHz maximum. This signal does not need to be related to the master CPU clock.

$\overline{RD}$. *Read* (input, active Low). A Low enables the master CPU to read information from the UPC. Raising the voltage on this pin above V_{DD} will force the UPC into test mode.

$\overline{WAIT}$. *Wait* (output, active Low, open-drain). When the CPU accesses the UPC register file, this signal requests the master CPU to wait until the UPC can complete its part of the transaction.

$\overline{WR}$. *Write* (input, active Low). A Low on this pin enables the master CPU to write information to the UPC. A simultaneous Low on $\overline{RD}$ and $\overline{WR}$ resets the UPC. It is held in reset as long as $\overline{WR}$ is Low.

Functional Description

Address Space. On the 40-pin UPC, all address space is committed to on-chip memory. There are 2048 bytes of mask-programmed ROM and 256 bytes of register file. I/O is memory-mapped to three registers in the register file. Only the Protopack and 64-pin versions of the UPC can access external program memory. See the section entitled "Special Configurations" for complete descriptions of the Protopack and 64-pin versions.

Program Memory. Figure 4 is a map of the 2K on-chip program ROM. Even though the architecture allows addresses from 0 to 4K, behavior of the device above program address 2047 (7FFH) is not defined. The first 12 bytes of program memory are reserved for the UPC interrupt vectors. For the Protopack and 64-pin versions, the address space is extended to 4096 bytes. In the RAM versions, addresses 0CH through 2FH are reserved for on-chip ROM.

Register File. This 256-byte file includes three I/O port registers (1-3H), 234 general-purpose registers (6-EEH), and 19 control, status and special I/O registers (0H, 4H, 5H, and F0-FFH). The functions and mnemonics assigned to these register address locations are shown in Figure 5. Of the 256 UPC registers, 19 can be directly accessed by the master CPU; the others are accessed indirectly via the block transfer mechanism.

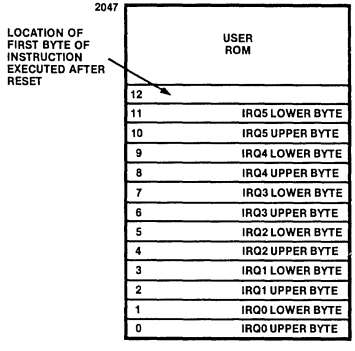


Figure 4. Program Memory Map

LOCATION		IDENTIFIER (UPC Side)
FFH	STACK POINTER	SP
FEH	MASTER CPU INTERRUPT CONTROL	MIC
FDH	REGISTER POINTER	RP
FCH	PROGRAM CONTROL FLAGS	FLAGS
FBH	UPC INTERRUPT MASK REGISTER	IMR
FAH	UPC INTERRUPT REQUEST REGISTER	IRQ
F9H	UPC INTERRUPT PRIORITY REGISTER	IPR
F8H	PORT 1 MODE	P1M
F7H	PORT 3 MODE	P3M
F6H	PORT 2 MODE	P2M
F5H	T_0 PRESCALER	PRE0
F4H	TIMER/COUNTER 0	T_0
F3H	T_1 PRESCALER	PRE1
F2H	TIMER/COUNTER 1	T_1
F1H	TIMER MODE	TMR
F0H	MASTER CPU INTERRUPT VECTOR REG	MIV
EFH	GENERAL-PURPOSE REGISTERS	
6H		
5H	DATA INDIRECTION REGISTER	DIND
4H	LIMIT COUNT REGISTER	LC
3H	PORT 3	P3
2H	PORT 2	P2
1H	PORT 1	P1
0H	DATA TRANSFER CONTROL REGISTER	DTC

Figure 5. Register File Organization

Functional Description (Continued)

The I/O port and control registers are included in the register file without differentiation. This allows any UPC instruction to process I/O or control information, thereby eliminating the need for special I/O and control instructions. All general-purpose registers can function as accumulators, address pointers, or index registers. In instruction execution, the registers are read when they are defined as sources and written when defined as destinations.

UPC instructions may access registers directly or indirectly using an 8-bit address mode or a 4-bit address mode and a Register Pointer. For the 4-bit addressing mode, the file is divided into 16 working register groups, each occupying 16 contiguous locations (Figure 6). The Register Pointer (RP) addresses the starting point of the active working-register group, and the 4-bit register designator supplied by the instruction specifies the register within the group. Any instruction altering the contents of the register file can also alter the Register Pointer. The UPC instruction set has a special Set Register Pointer (SRP) instruction for initializing or altering the pointer contents.

Stacks. An 8-bit Stack Pointer (SP), register R255, is used for addressing the stack, residing within the 234 general-purpose registers, address location 6H through EFH. PUSH and POP instructions can save and restore any register in the register file on the stack. During CALL instructions, the Program Counter is automatically saved on the stack. During UPC interrupt cycles, the Program Counter and the Flag register are automatically saved on the stack. The RET and IRET instructions pop the saved values of the Program Counter and Flag register.

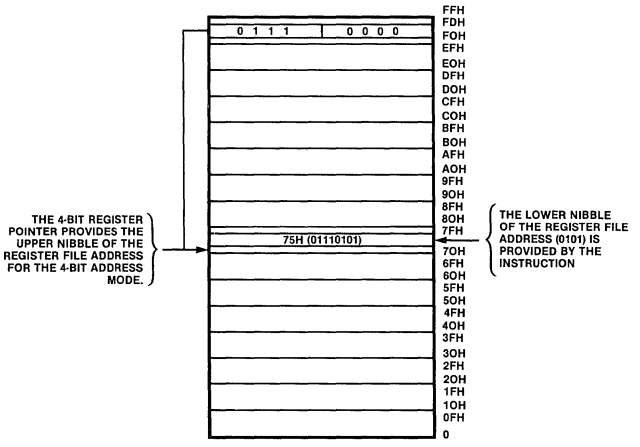


Figure 6. Register Pointer Mechanism

Ports. The UPC has 24 lines dedicated to input and output. These are grouped into three ports of eight lines each and can be configured under software control as inputs, outputs, or special control signals. They can be programmed to provide Parallel I/O with or without handshake and timing signals. All outputs can have active pullups and pulldowns, compatible with TTL loads. In addition, they may be configured as open-drain outputs.

Port 1. Individual bits of Port 1 can be configured as input or output by programming Port 1 Mode register (P1M) F8H. This port is accessed by the UPC program as general register 1H. It is written by specifying address 1H as the destination of any instruction used to store data in the output register. The port is read by specifying address 1H as the source of an instruction.

Port 1 may be placed under handshake control by programming Port 3 Mode register (P3M) F7H. This configures Port 3 pins $P3_3$ and $P3_4$ as handshake control lines $\overline{DAV}_1$ and RDY_1 for input handshake, or RDY_1 and $\overline{DAV}_1$ for output handshake, as determined by the direction (input or output) assigned to bit 7 of Port 1. The Port 3 Mode register also has a bit that programs Port 1 for open-drain output.

Port 2. Individual bits of Port 2 can be configured as inputs or outputs by programming Port 2 Mode register (P2M) F6H. This port is accessed by the UPC program as general register 2H, and its functions and methods of programming are the same as those of Port 1. Port 3 pins $P3_1$ and $P3_6$ are the handshake lines $\overline{DAV}_2$ and RDY_2, with the direction (input or output) determined by the state of bit 7 of the port. The Port 3 Mode register also has a bit used to program Port 2 for open-drain output.

Function	Line	Direction	Signal
Handshake	$P3_1$	In	$\overline{DAV}_2/RDY_2$
	$P3_3$	In	$\overline{DAV}_1/RDY_1$
	$P3_4$	Out	$RDY_1/\overline{DAV}_1$
	$P3_6$	Out	$RDY_2/\overline{DAV}_2$
UPC Interrupt Request*	$P3_0$	In	IRQ_3
	$P3_1$	In	IRQ_2
	$P3_3$	In	IRQ_1
Counter/Timer	$P3_1$	In	T_{7N}
	$P3_6$	Out	T_{OUT}
Master CPU	$P3_5$	Out	$\overline{INT}$
	$P3_2$	In	$\overline{INTACK}$
	$P3_0$	In	IEI
	$P3_7$	Out	IEO
Test Mode	$P3_5$	Out	$\overline{A}/D$

*$P3_0$, $P3_1$, and $P3_3$ can always be used as UPC interrupt request inputs, regardless of the configuration programmed.

Table 1. Port 3 Control Functions

Functional Description (Continued)

Port 3. This port can be configured as I/O or control lines by programming the Port 3 Mode register. Port 3 is accessed as general register 3H. The directions of the eight data lines are fixed. Four lines, $P3_0$ through $P3_3$, are inputs, and the other four, $P3_4$ through $P3_7$, are outputs. The control functions performed by Port 3 are listed in Table 1.

Counter/Timers. The UPC contains two 8-bit programmable counter/timers, each driven by an internal 6-bit programmable prescaler.

The T1 prescaler can be driven by internal or external clock sources. The T0 prescaler is driven by an internal clock source. Both counter/timers operate independently of the processor instruction sequence to relieve the program from time-critical operations like event counting or elapsed-time calculation. T0 Prescaler register (PRE0) F5H and T1 Prescaler register (PRE1) F3H can be programmed to divide the input frequency of the source being counted by any number from 1 to 64. A counter register (F2H or F4H) is loaded with a number from 1 to 256. The corresponding counter is decremented from this number each time the prescaler reaches end-of-count. When the count is complete, the counter issues a timer interrupt request; IRQ_4 for T0 or IRQ_5 for T1. Loading either counter with a number (n) results in the interruption of the UPC at the nth count.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. They can be programmed to stop upon reaching end-of-count (Single-Pass mode) or to automatically reload the initial value and continue counting (Modulo-n Continuous mode). The counters and prescalers can be read at any time without disturbing their values or changing their counts. The clock sources for both timers can be defined as any one of the following:

- UPC internal clock (4 MHz maximum) divided by four.
- External clock input to Counter/Timer T1 via $P3_1$ (1 MHz maximum).
- Retriggerable trigger input for the UPC internal clock divided by four.
- Nonretriggerable trigger input for the UPC internal clock divided by four.
- External gate input for the UPC internal clock divided by four.

Interrupts. The UPC allows six interrupts from eight different sources as follows:

- Port 3 lines $P3_0$, $P3_2$, and $P3_3$.
- The master CPU(3).
- The two counter/timers.

These interrupts can be masked and globally enabled or disabled using Interrupt Mask Register (IMR) FBH. Interrupt Priority Register (IPR) F9H specifies the order of their priority. All UPC interrupts are vectored.

Table 2 lists the UPC's interrupt sources, their types, and their vector locations in program ROM. Interrupt Request IRQ_6 is dedicated to master CPU communications. Interrupt Requests IRQ_1, IRQ_2, and IRQ_3 are generated on the falling transitions of external inputs $P3_3$, $P3_1$, and $P3_0$. Interrupt Requests IRQ_4 and IRQ_5 are generated upon the timeout of the UPC's two counter/timers. When an interrupt request is granted, the UPC enters an interrupt machine cycle. This cycle disables all subsequent interrupts, saves the Program Counter and Status Flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

The UPC also supports polled systems. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Following any hardware reset operation, an EI instruction must be executed to enable the setting of any interrupt request bit in the IRQ register. Interrupts must be disabled prior to changing the content of either the IPR (F9H) or the IMR (FBH). DI is the only instruction that should be used to globally disable interrupts.

Name	Source	Vector Location	Comments
IRQ_0	EOM, XERR, LERR	0,1	Internal (R0 Bits 0, 1, 2)
IRQ_1	$\overline{DAV}_1$, IRQ_1	2,3	External ($P3_3$) ↓ Edge Triggered
IRQ_2	$\overline{DAV}_2$, IRQ_2, T_{IN}	4,5	External ($P3_1$) ↓ Edge Triggered
IRQ_3	IRQ_3, IEI	6,7	External ($P3_0$) ↓ Edge Triggered
IRQ_4	T0	8,9	Internal
IRQ_5	T1	10,11	Internal

Table 2. Interrupt Types, Sources, and Vector Locations

Functional Description (Continued)

Master CPU Register File Access. There are two ways in which the master CPU can access the UPC register file: direct access and block access.

Direct Access. Three UPC registers—the Data Transfer Control (0H), the Master Interrupt Vector (F0H), and the Master Interrupt Control (FEH)—are mapped directly into the master CPU address space. The master CPU accesses these registers via the addresses shown in Table 3.

The master CPU also has direct access to 16 registers known as the DSC (Data, Status, Command) registers. The DSC registers are numbered 0 through F (DSC0-DSCF). These registers can be any 16 contiguous register file registers beginning on a 16-byte boundary. The base address of the DSC register group is designated by the IRP (I/O Register Pointer), which is bits D_4–D_7 of the Data Transfer Control register (0H). Figure 7 shows how the register address is made up of the 4-bit IRP field, concatenated with the low order 4-bits of the address from the master CPU.

Block Access. The master CPU may transmit or receive blocks of data via address xxx11111. When the master CPU accesses this address, the UPC register pointed to by the Data Indirection register is read or written. The Data Indirection register is incremented, and the Limit Count register is decremented, for example, when the master CPU issues a read or write to address xxx11111 while the Data Indirection register contains the value 33H. The operation causes register 33H to be read or written and the Data Indirection register to be incremented to 34H. This scheme is well suited to Block I/O Instructions and allows the master CPU to efficiently read or write a block of data to or from the UPC.

The Limit Count register (04H) is decremented and is used to control the number of bytes to be transferred by master CPU block accesses. If the master CPU attempts a read or write to the UPC after the Limit Count register reaches 0, the access is not completed, the LERR bit (D_2) of the Data Transfer Control register is set (indicating a limit error), and the LERR error causes an IRQ_0 interrupt request.

The IRP field of the Data Transfer Control register, the Data Indirection register, and the Limit Count register are not directly accessible to the master CPU and therefore must be set by the UPC. This allows the UPC to protect itself from master CPU errors and frees the master CPU from tracking the UPC's internal data layout.

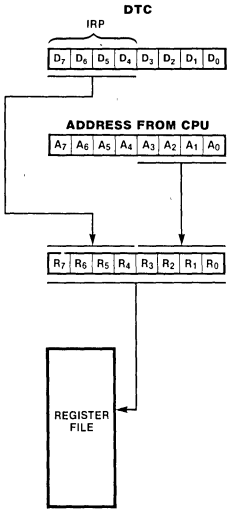


Figure 7. DSC Register Addressing Scheme

UPC Address Decimal	Hex	Identifier	Address
0	0H	DTC	xxx11000
5	5H	DIND	
@5**	@5H**		xxx11111
240	F0H	MIV	xxx10000
254	FEH	MIC	xxx11110
*n		DSC0	xxx00000
n+1		DSC1	xxx00001
n+2		DSC2	xxx00010
n+3		DSC3	xxx00011
n+4		DSC4	xxx00100
n+5		DSC5	xxx00101
n+6		DSC6	xxx00110
n+7		DSC7	xxx00111
n+8		DSC8	xxx01000
n+9		DSC9	xxx01001
n+10		DSCA	xxx01010
n+11		DSCB	xxx01011
n+12		DSCC	xxx01100
n+13		DSCD	xxx01101
n+14		DSCE	xxx01110
n+15		DSCF	xxx01111

x = don't care
*n is the value in the IRP x 16
**Master CPU accesses the register address in Register 5

Table 3. Master CPU/UPC Register Map

Special Configurations

There are two Protopack and two 64-pin versions of the UPC. These versions are identical to the 40-pin UPC with the following exceptions:

- Internal ROM is totally omitted from the 64-pin development and ROM Protopack versions.
- All but 36 bytes of internal ROM are omitted from the 64-pin RAM and Protopack RAM versions.
- The memory address and data lines are buffered and brought out to external pins or to the socket on the Protopack.
- Control lines for the external memory are also provided.

The 64-pin version of the UPC allows the user to prototype the system in hardware with an actual UPC device and to develop the code intended to be mask programmed into the on-chip ROM of the 40-pin UPC for the production system. The 64-pin or Protopack RAM versions of the UPC are extremely versatile parts. Memory space can be extended to 4K bytes on the 64-pin version by using external RAM/ROM for all but 36 bytes of the UPC's memory space. This memory can then be down-loaded from the master CPU using a bootstrap program stored in the 36 bytes (C-2F). Figure 8 is a memory map for the 64-pin RAM version.

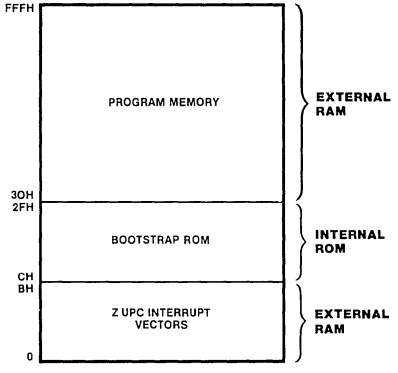


Figure 8. UPC RAM Version Memory Map

64-Pin and Protopack Pin Functions. Forty of the pins on the 64-pin and Protopack versions have functions identical to those of the 40-pin version. The remaining 24 pins have additional functions described below. (Figures 9 through 11 show the 64-pin and Protopack versions' pin functions and pin assignments.)

A_0-A_{11}. *Program Memory Address Lines* (output). These lines are identical in all 64-pin and RAM versions in the Protopack. They are used to address 4K bytes of external UPC memory.

D_0-D_7. *Program Data* (input). Data is read in from the external memory on these lines. The RAM version also writes external memory through this bus.

IACK. *Interrupt Acknowledge* (output, active High). This signal is active whenever an internal UPC interrupt cycle is in process.

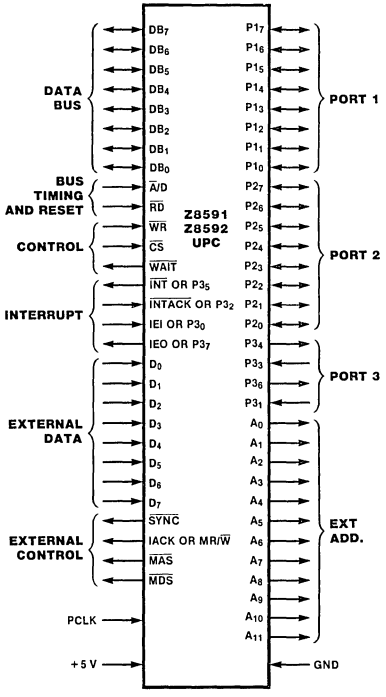


Figure 9. Z8591/Z8592 UPC Pin Functions

Special Configurations (Continued)

$\overline{\text{MAS}}$. *Memory Address Strobe* (output, active Low). This address strobe is pulsed once for each memory fetch to interface with quasi-static RAM.

$\overline{\text{MDS}}$. *Memory Data Strobe* (output, active Low). This signal is Low during an instruction fetch or memory write.

MR/$\overline{\text{W}}$. *Memory Read/Write* (output RAM versions only). This signal is High when the UPC is fetching an instruction and Low when it is loading external memory.

$\overline{\text{SYNC}}$. *Instruction Sync* (output, active Low). This signal is Low during the clock cycle just preceding an opcode fetch.

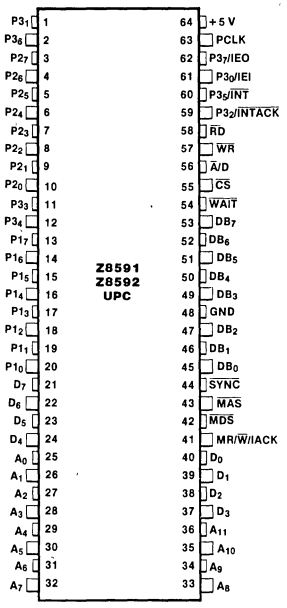
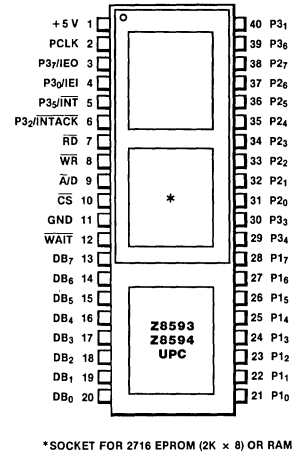


Figure 10. Z8591/Z8592 UPC Pin Assignments



*SOCKET FOR 2716 EPROM (2K × 8) OR RAM

Figure 11. Z8593/Z8594 UPC Protopack Pin Assignments

Addressing Modes

The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working-register pair address
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate

Additional Symbols

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack Pointer (control register FFH)
PC	Program Counter
FLAGS	Flag register (control register FCH)
RP	Register Pointer (control register FDH)
IMR	Interrupt Mask register (control register FBH)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags

Control Register FCH contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

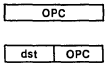
Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

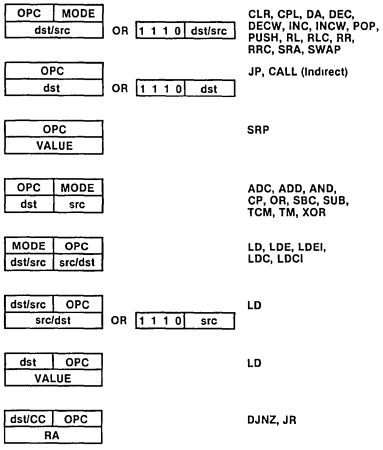
Instruction Formats



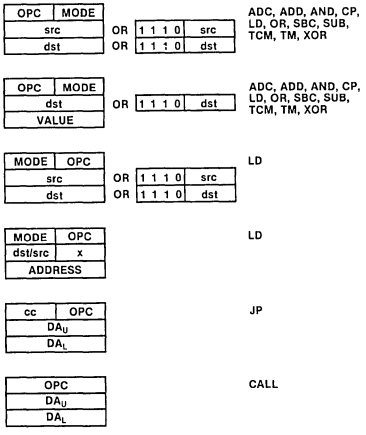
CCF, DI, EI, IRET, NOP, RCF, RET, SCF

INC r

One-Byte Instructions



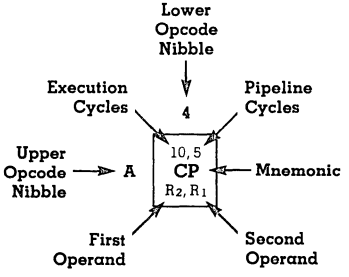
Two-Byte Instructions



Three-Byte Instructions

Opcode Map

Upper Nibble (Hex)	Lower Nibble (Hex) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, Ir_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	6,7 **SWAP** R_1	6,7 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR IR	IM R r X r Ir r R IR IM IM R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst	R IR		90 91	* * * * - -
RLC dst	R IR		10 11	* * * * - -
RR dst	R IR		E0 E1	* * * * - -
RRC dst	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst,src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

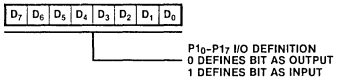
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

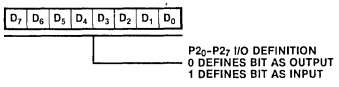
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

R248 P1M
Port 1 Mode Register
UPC register address (Hex): F8



R246 P2M
Port 2 Mode Register
UPC register address (Hex): F6



R247 P3M
Port 3 Mode Register
UPC register address (Hex): F7

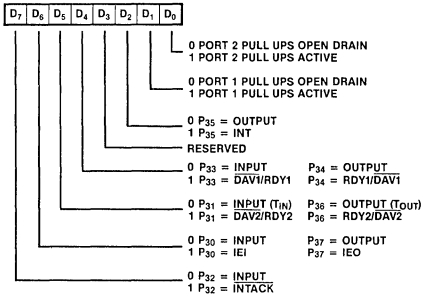
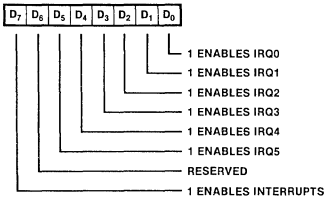
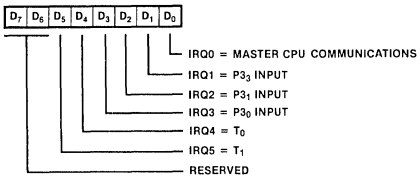


Figure 12. Port Mode Registers

R251 IMR
Interrupt Mask Register
UPC register address (Hex): FB



R250 IRQ
Interrupt Request Register
UPC register address (Hex): FA



R249 IPR
Interrupt Priority Register
UPC register address (Hex): F9 (Write Only)

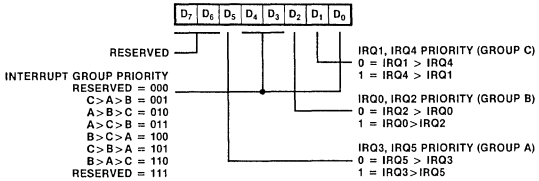
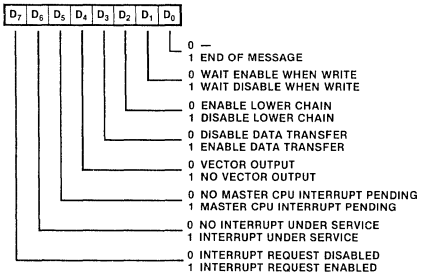


Figure 13. Interrupt Control Registers

R254 MIC
Master CPU Interrupt Control Register
UPC register address (Hex): FE



R240 MIV
Master CPU Interrupt Vector Register
UPC register address (Hex): F0

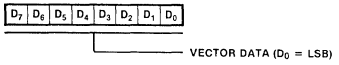
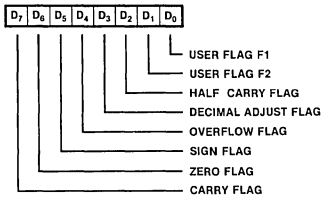


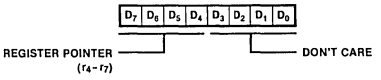
Figure 14. Master CPU Interrupt Registers

Registers
(Continued)

R252 FLAGS
Flag Register
UPC register address (Hex): FC



R253 RP
Register Pointer
UPC register address (Hex): FD



R255 SP
Stack Pointer
UPC register address (Hex): FF

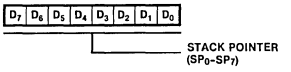
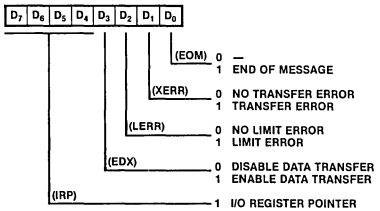
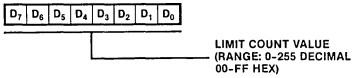


Figure 15. UPC Control Registers

R0 DTC
Data Transfer Control Register
UPC register address (Hex): 00



R4 LC
Limit Count Register
UPC register address (Hex): 04



R5 DIND
Data Indirection Register
UPC register address (Hex): 05

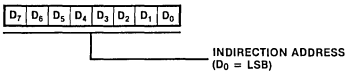
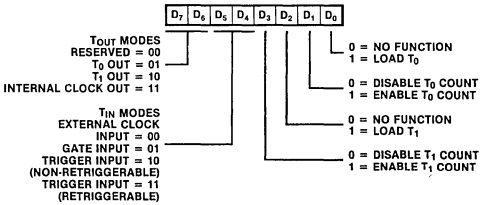
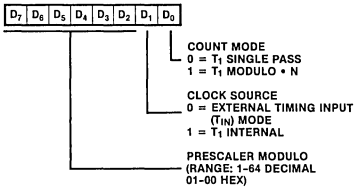


Figure 16. Master CPU-UPC Data Transfer Registers

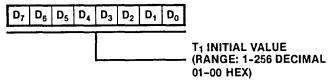
R241 TMR
Timer Mode Register
UPC register address (Hex): F1



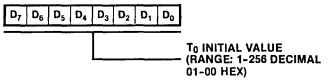
R243 PRE1
Prescaler 1 Register
UPC register address (Hex): F3



R242 T1
Counter/Timer 1 Register
UPC register address (Hex): F2



R244 T0
Counter/Timer 0 Register
UPC register address (Hex): F4



R245 PRE0
Prescaler 0 Register
UPC register address (Hex): F5

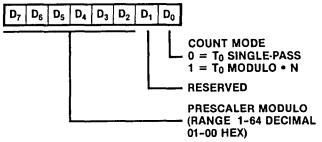


Figure 17. UPC Counter/Timer Registers

Registers (Continued)

Control Register	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Comments
00_H Data Transfer Control Register	X	X	X	X	0	0	0	0	Disable data transfer from master CPU
04_H Limit Count Register	Not Defined								
05_H Data Indirection Register	Not Defined								
$F0_H$ Interrupt Vector Register	Not Defined								
$F1_H$ Timer Mode	0	0	0	0	0	0	0	0	Stops T0 and T1
$F2_H$ T0 Register	Not Defined								
$F3_H$ T0 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode
$F4_H$ T1 Register	Not Defined								
$F5_H$ T1 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode External clock source
$F6_H$ Port 2 Mode	1	1	1	1	1	1	1	1	Port 2 lines defined as inputs
$F7_H$ Port 3 Mode	0	0	0	0	X	1	0	0	Port 1, 2 open drain; $P3_5$ = INT; $P3_0$, $P3_1$, $P3_2$, $P3_3$ defined as input; $P3_4$, $P3_6$, $P3_7$ defined as output.
$F8_H$ Port 1 Mode	1	1	1	1	1	1	1	1	Port 1 lines defined as inputs
$F9_H$ Interrupt Priority	Not Defined								
FA_H Interrupt Request	X	X	0	0	0	0	0	0	Reset Interrupt Request
FB_H Interrupt Mask	0	X	X	X	X	X	X	X	Interrupts disabled
FC_H Flag Register	Not Defined								
FD_H Register Pointer	Not Defined								
FE_H Master CPU Interrupt Control Register	0	0	0	0	0	0	0	0	Master CPU interrupt disabled; wait enable when write; lower chain enabled
FF_H Stack Pointer	Not Defined								

NOTE: X means not defined

Table 4. Control Register Reset Conditions

Absolute Maximum Ratings

Voltages on all pins (except V_{BB}) with respect to GND . . . . . -0.5 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- $V_{SS} = GND = 0\ V$
- $0°C \leq T_A \leq +70°C$

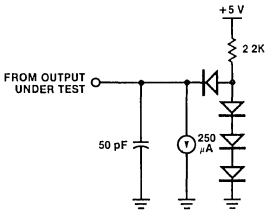


Figure 18. Test Load 1

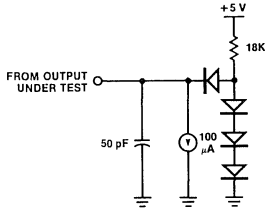


Figure 19. Test Load 2

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	2.4	V_{CC}	V		
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V		
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$	1
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA	1
I_{IL}	Input Leakage	-10	10	μA	$0 \leq V_{IN} \leq +5.25$ V	
I_{OL}	Output Leakage	-10	10	μA	$0 \leq V_{IN} \leq +5.25$ V	
I_{CC}	V_{CC} Supply Current		180	mA		

1 For A_0-A_{11} and D_0-D_7, $\overline{MDS}$, $\overline{SYNC}$, $\overline{MAS}$, and MR/$\overline{W}$/IACK on the 64-pin versions. $I_{OH} = 100\ \mu A$ and $I_{OL} = 1.0$ mA

Master CPU Interface Timing

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TrC	Clock Rise Time		20	
2	TwCh	Clock High Width	105	1855	
3	TfC	Clock Fall Time		20	
4	TwCl	Clock Low Width	105	1855	
5	TpC	Clock Period	250	2000	
6	TsA/D(WR)	$\overline{A}$/D to $\overline{WR}$ ↓ Setup Time	80		
7	TsA/D(RD)	$\overline{A}$/D to $\overline{RD}$ ↓ Setup Time	80		
8	ThA/D(WR)	$\overline{A}$/D to $\overline{WR}$ ↑ Hold Time	30		
9	ThA/D(RD)	$\overline{A}$/D to $\overline{RD}$ ↑ Hold Time	30		
10	TsCSf(WR)	$\overline{CS}$ ↓ to$\overline{WR}$ ↓ SetupTime	0		
11	TsCSf(RD)	$\overline{CS}$ ↓ to $\overline{RD}$ ↓ Setup Time	0		
12	TsCSr(WR)	$\overline{CS}$ ↑ to $\overline{WR}$ ↓ Setup Time	60		
13	TsCSr(RD)	$\overline{CS}$ ↑ to $\overline{RD}$ ↓ Setup Time	60		
14	ThCS(WR)	$\overline{CS}$ to $\overline{WR}$ ↓ Hold Time	0		
15	ThCS(RD)	$\overline{CS}$ to $\overline{RD}$ ↓ Hold Time	0		
16	TsDI(WR)	Data in to $\overline{WR}$ ↓ Setup Time	0		
17	Tw(WR)	$\overline{WR}$ Low Width	390		
18	Tw(RD)	$\overline{RD}$ Low Width	390		
19	ThWR(DI)	Data in to $\overline{WR}$ ↑ Hold Time	0		
20	TdRD(DI)	Data Valid from $\overline{RD}$ ↓ Delay			1
21	ThRD(DI)	Data Valid to $\overline{RD}$ ↑ Hold Time	0		
22	TdRD(DI_Z)	Data Bus Float Delay from $\overline{RD}$ ↑		70	
23	TdRD(DB_A)	$\overline{RD}$ ↓ to Read Data Active Delay	0		
24	TdWR(W)	$\overline{WR}$ ↓ to $\overline{WAIT}$ ↓ Delay		150	
25	TdRD(W)	$\overline{RD}$ ↓ to $\overline{WAIT}$ ↓ Delay		150	
26	TdDI(W)	Data Valid to $\overline{WAIT}$ ↑ Delay	0		

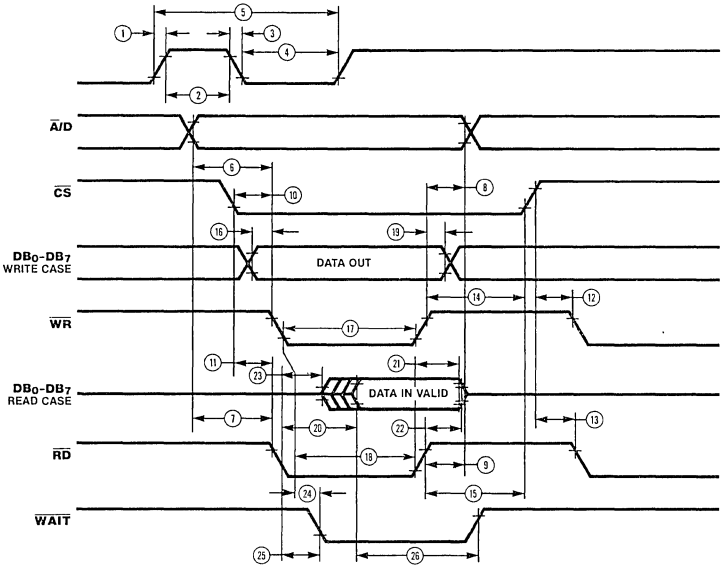
Interrupt Acknowledge Transactions

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
27	TsACK(RD)	$\overline{INTACK}$ ↓ to $\overline{RD}$ ↓ Setup Time	90		2
28	TdRD(DI)	$\overline{RD}$ ↓ to Vector Valid Delay		255	
29	ThRD(ACK)	$\overline{RD}$ ↑ to $\overline{INTACK}$ ↑ Hold Time	0		
30	ThIEI(RD)	IEI to $\overline{RD}$ ↓ Hold Time	100		
31	TwRDl	$\overline{RD}$ (Acknowledge) Low Width	255		
32	TdIEI(IEO)	IEI to IEO Delay		120	
33	TsIEI(RD)	IEI to $\overline{RD}$ ↓ Setup Time	150		
34	$TdACK_f$(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		250	
35	$TdACK_r$(IEO)	$\overline{INTACK}$ ↑ to IEO ↑ Delay		250	

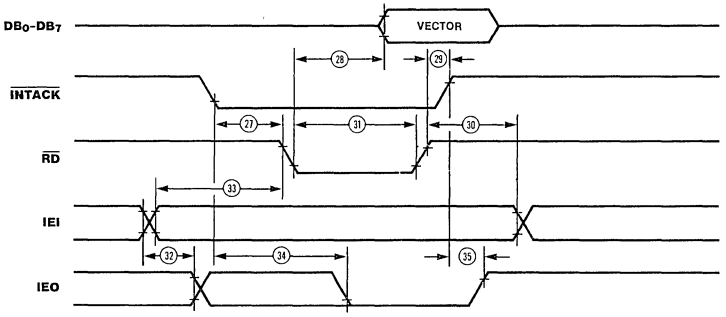
NOTES:
1. This parameter is dependent on the state of the UPC at the time of master CPU access.
2. In case where daisy chain is not used.
3. The timing characteristics given reference 2.0 V as High and 0.8 V as Low
4. All output ac parameters use test load 1

*Timings are preliminary and subject to change.

Master CPU Interface Timing



Interrupt Acknowledge Timing



Handshake Timing

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TsDI(DA)	Data In Setup Time	0		
2	ThDA(DI)	Data In Hold Time	230		
3	TwDA	Data Available Width	175		1,2
4	TdDAL(RY)	Data Available Low to Ready Delay Time	20 0	175	1,2 2,3
5	TdDAH(RY)	Data Available High to Ready Delay Time	 0	150	1,2 2,3
6	TdDO(DA)	Data Out to Data Available Delay Time	50		2
7	TdRY(DA)	Ready to Data Available Delay Time	0	205	2

Reset Timing

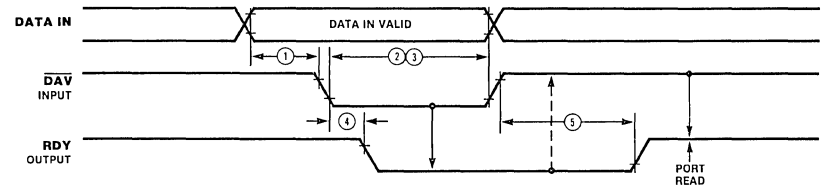
Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TdRDQ(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓ for No Reset	40		
2	TdWRQ(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓ for No Reset	50		
3	TwRES	Minimum Width of $\overline{WR}$ and $\overline{RD}$ both Low for Reset	250		4

RAM Version Program Memory Timing

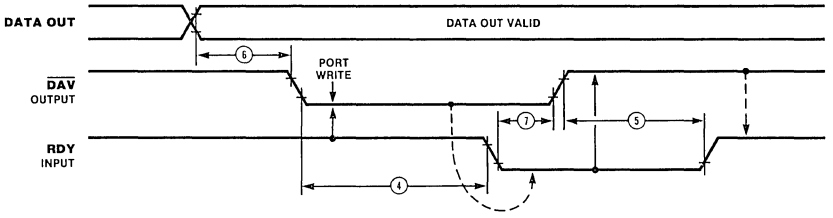
Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes*
1	TwMAS	Memory Address Strobe Width	60		5
2	TdA(MAS)	Address Valid to Memory Address Strobe ↑ Delay	30		5
3	TdMR/W (MAS)	Memory Read/Write to Memory Address Strobe ↑ Delay	30		5
4	TdMDS(A)	Memory Data Strobe ↑ to Address Change Delay	60		
5	TdMDS (MR/W)	Memory Data Strobe ↑ to Memory Read/Write Not Valid Delay	80		
6	Tw(MDS)	Memory Data Strobe Width (Write Case)	160		6
7	TdD0(MDS)	Data Out Valid to Memory Data Strobe ↓ Delay	30		5
8	TdMDS(D0)	Memory Data Strobe ↑ to Data Out Change Delay	30		5
9	Tw(MDS)	Memory Data Strobe Width (Read Case)	230		6
10	TdMDS(DI)	Memory Data Strobe ↓ to Data In Valid Delay		160	7
11	TdMAS(DI)	Memory Address Strobe ↓ to Data In Valid Delay		280	7
12	ThMDS(DI)	Memory Data Strobe ↑ to Data In Hold Time	0		
13	TwSY	Instruction Sync Out Width	160		
14	TdSY(MDS)	Instruction Sync Out to Memory Data Strobe Delay	200		
15	TwI	Interrupt Request via Port 3 Input Width	100		

NOTES
1. Input Handshake
2 Test Load 1
3 Output Handshake
4 Internal reset signal is ½ to 2 clock delays from external reset condition.
5. Delay times are specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in input clock period must be added to the specified delay time.
6. Data strobe width is specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in three input clock periods must be added to the specified width. Data strobe width varies according to the instruction being executed.
7. Address strobe and data strobe to data in valid delay times represent memory system access times and are given for a 4 MHz input frequency.
8 All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."
9 All output ac parameters use test load 2.
*Timings are preliminary and subject to change

Handshake Timing

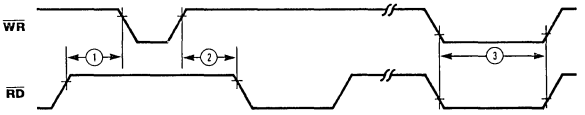


Input Handshake

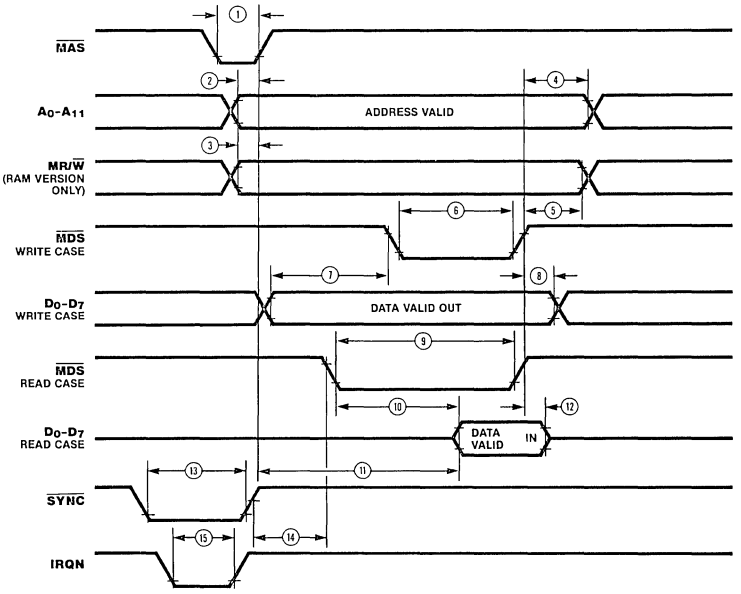


Output Handshake

Reset Timing



RAM Version Program Memory Timing



Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8590	CE	4.0 MHz	UPC (40-pin)
Z8590	CS	4.0 MHz	Same as above
Z8590	DE	4.0 MHz	Same as above
Z8590	DS	4.0 MHz	Same as above
Z8590	PE	4.0 MHz	Same as above
Z8590	PS	4.0 MHz	Same as above
Z8591	QS	4.0 MHz	UPC External ROM-based Program Memory (64-pin)
Z8592	QS	4.0 MHz	UPC External RAM-based Program Memory (64-pin)
Z8593	RS	4.0 MHz	UPC 2716 EPROM Program Memory (40-pin)
Z8594	RS	4.0 MHz	UPC RAM Program Memory (40-pin)

NOTES C = Ceramic, D = Cerdip, P = Plastic, Q = Quip, R = Protopack; E = −40°C to +85°C, S = 0°C to +70°C.

Z8 Family

Zilog

Zilog Z8™ Family

Zilog

The New Standard For Single-Chip Microcomputers

March 1981

The Z8 Family of microcomputers offers the most sophisticated processing capability available on a single chip. As an extension of earlier generations of microcomputers, the Z8 Family provides standard on-chip functions, such as:

- 2K or 4K bytes of ROM
- 144 8-bit registers
- 32 lines of programmable I/O
- Clock oscillator

In addition, the Z8 Family offers advanced on-chip features, including:

- Two counter/timers
- Six vectored interrupts
- UART for serial I/O communication
- Stack functions
- Power-down option
- TTL compatibility

The capability of the Z8 Family of microcomputers is expandable off-chip to provide an additional 62K bytes of program memory and 62K bytes of data memory for the 2K-byte ROM version. It provides an additional 60K bytes of program memory and 60K bytes of data memory for the 4K-byte ROM version. The interface to external memory is accomplished through one, one and one-half, or two of the 8-bit I/O ports, depending on the number of address bits required for the external functions. The Z-BUS protocol allows easy interface to external functions including Zilog's family of peripheral chips.

With the third-generation Z8 Family, Zilog is pushing the capability of microcomputers beyond the first and second generation of computers. The Z8 Family challenges the "multi-chip solution" design currently implemented by general-purpose microprocessors. Designs based on Z8-Family microcomputers offer a minimum chip-count configuration that can easily be expanded to meet requirements for enhancement options and for future improvements.

Optimized Instruction Set. The instruction set of the Z8-Family microcomputers is optimized for high-code density and reduced execution time. This feature is supported by a "working register area" concept that uses short (4-bit) register addresses. The general-purpose registers can be used as accumulators, as address pointers for indirect addressing, as index registers, or for implementing an on-chip stack.

The 47 instruction types and six addressing modes—together with the ability to operate on bits, 4-bit BCD digits, 8-bit bytes, and 16-bit words—offer unique programming capability and flexibility.

Growing Family. The Z8 Family of microcomputers is growing to meet the needs of more complex designs. The 4K ROM version of the Z8 microcomputer (the Z8610 series) offers all the features of the Z8 Family, plus 4K bytes of on-chip ROM. The increased ROM allows the designer to take advantage of the code optimization inherent in the Z8 instruction set when using between 2K and 4K bytes of program memory.

The ROMless microcomputer provides an alternative for designers seeking to take advantage of the on-chip features of the Z8601 in applications that require external program memory. A Z8681 microcomputer can be used to control a system that addresses up to 128K bytes of off-chip memory.

The Z8671 microcomputer is a Z8-based BASIC/debug interpreter on a chip. The BASIC used in the Z8671 is a subset of Dartmouth BASIC with the added capability of interaction between the interpreter and its environment through the debug facility. The BASIC/debug interpreter resides in the 2K of on-chip ROM, with all the features of the Z8 microcomputer at its disposal.

Expanded Applications. The Z8 Family of microcomputers is finding its way into increasingly sophisticated designs. In addition to the low-end capability applications commonly used with microcomputers, the Z8 Family of microcomputers can be used effectively in such applications as:

- Computer peripheral controllers
- Smart terminals
- Dumb terminals
- Telephone switching systems
- Arcade games and intelligent home games
- Process control
- Intelligent instrumentation
- Automotive mechanisms

An example of how a Z8 might be used in the design of an intelligent terminal is shown in Figure 1. The features of such a terminal depend on its specific requirements, but it is clear that the Z8 microcomputers offer unprecedented capability and flexibility to the microcomputer designer.

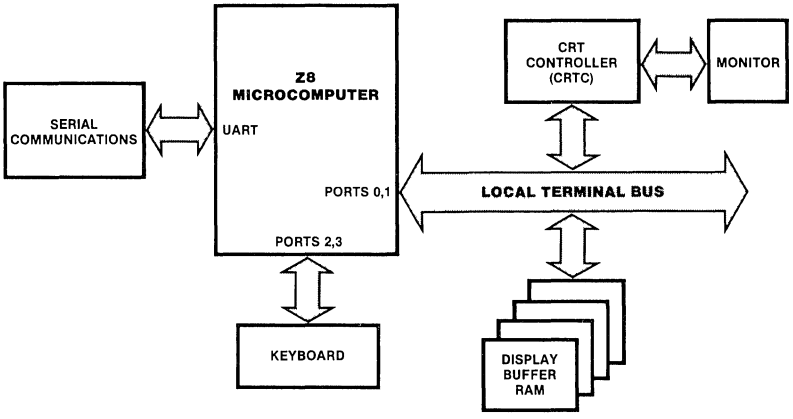


Figure 1. Z8-Based Intelligent Terminal

Z8™ Family of Microcomputers Z8601 · Z8602 · Z8603

Zilog

Product Specification

March 1981

Z8601 Single-Chip Microcomputer with 2K ROM
Z8602 Development Device with Memory Interface
Z8603 Prototyping Device with EPROM Interface

Features

- Complete microcomputer, 2K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 62K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Average instruction execution time of 2.2 μs, maximum of 4.25 μs.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1.5 μs.
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.

General Description

The Z8601 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8601 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8601 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K bytes of internal ROM, a traditional microprocessor that manages up to 124K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

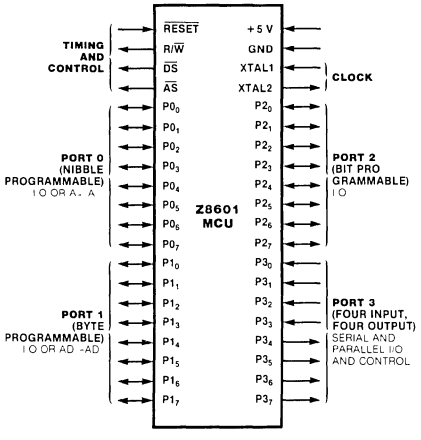


Figure 1. Z8601 MCU Pin Functions

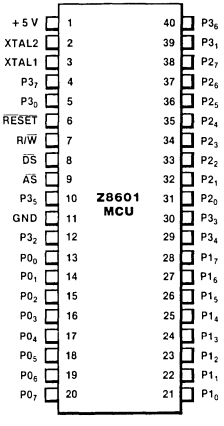


Figure 2. Z86(MCU Pin Assignments

Architecture

Z8601 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8601 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8601 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

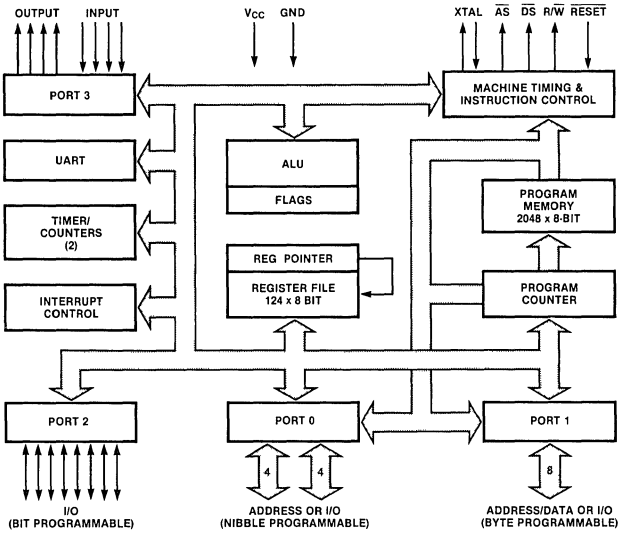


Figure 3. Functional Block Diagram

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8601. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8601 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum) or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

Address Spaces

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 2048 bytes consist of on-chip mask-programmed ROM. At addresses 2048 and greater, the Z8601 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8601 can address 62K bytes of external data memory beginning at locations 2048 (Figure 5). External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and 16 control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8601 instructions can access registers

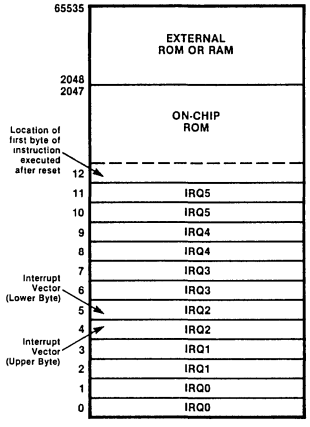


Figure 4. Program Memory Map

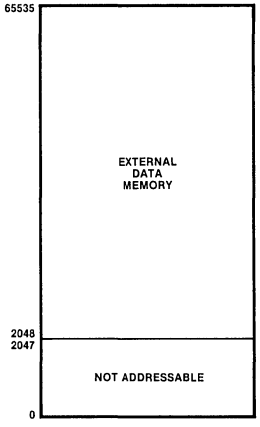


Figure 5. Data Memory Map

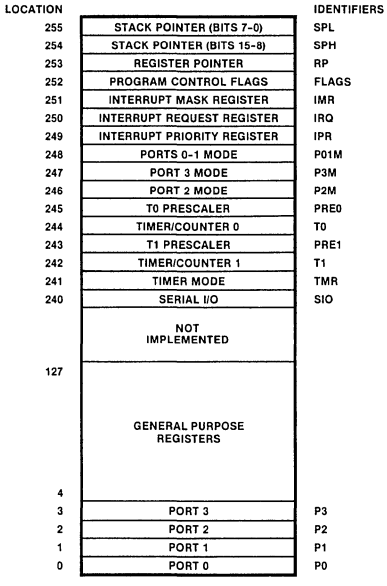


Figure 6. The Register File

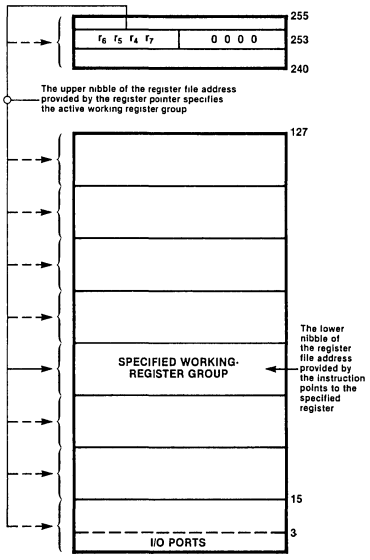


Figure 7. The Register Pointer

Address Spaces (Continued)

directly or indirectly with an 8-bit address field. The Z8601 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4–R127).

Serial Input/ Output

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second.

The Z8601 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.

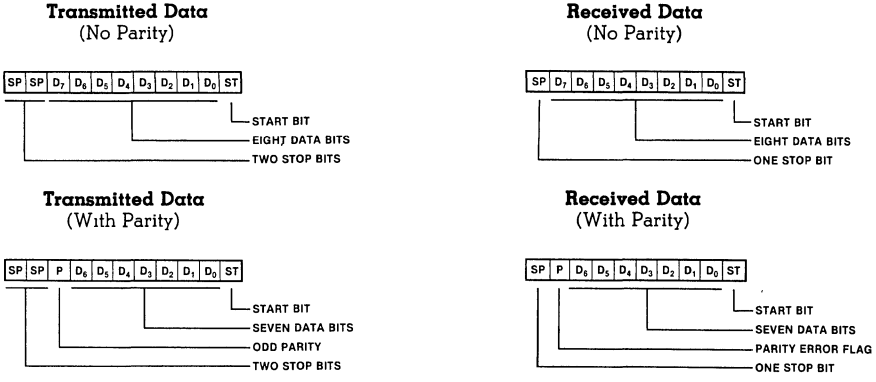


Figure 8. Serial Data Formats

Counter/ Timers

The Z8601 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8601 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8600 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input and $P3_4$ as a Bus Request output.

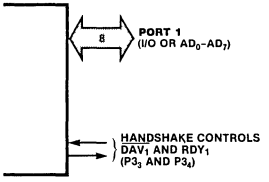


Figure 9a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

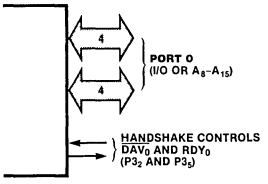


Figure 9b. Port 0

Port 2 bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

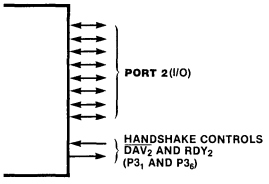


Figure 9c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0-IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

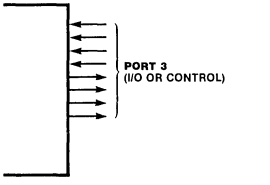


Figure 9d. Port 3

Interrupts

The Z8601 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8601 interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, series-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors (C_1 = 15 pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, series resonant
- Fundamental type, 8 MHz maximum
- Series resistance, $R_S \leq 100\ \Omega$

Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the V_{MM} (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.

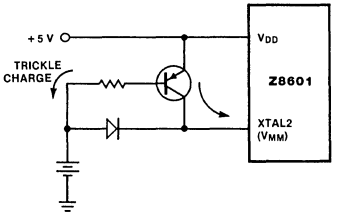


Figure 10. Recommended Driver Circuit for Power Down Operation

Z8602 Development Device

This 64-pin development version of the 40-pin mask-programmed Z8601 (Figure 11) allows the user to prototype the system in hardware with an actual device and to develop the code that is eventually mask-programmed into the on-chip ROM of the Z8601.

The Z8602 is identical to the Z8601 with the following exceptions:

- The internal ROM has been removed.
- The ROM address lines and data lines are buffered and brought out to external pins.
- Control lines for the new memory have been added.

Pin Description. The functions of the Z8602 I/O lines, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, XTAL1, XTAL2 and $\overline{RESET}$ are identical to those of their Z8601 counterparts. The functions of the remaining 24 pins are as follows:

A_0-A_{11}. *Program Memory Address* (outputs). A_0-A_{11} access the first 2K bytes of program memory. A_{11} is a reserved pin.

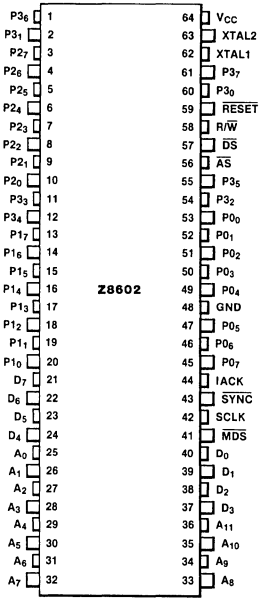


Figure 11. Z8602 Pin Assignments

Z8602 Development Device (Continued)

D_0-D_7. *Program Data* (inputs). Program data from the first 2K bytes of program memory is input through pins D_0-D_7.

IACK. *Interrupt Acknowledge* (output, active High). IACK is driven High in response to an interrupt during the interrupt machine cycle.

$\overline{MDS}$. *Program Memory Data Strobe* (output, active Low). $\overline{MDS}$ is Low during an instruction fetch cycle when the first 2K bytes of program memory are being accessed.

SCLK. *System Clock* (output). SCLK is the internal clock output through a buffer. The clock rate is equal to one-half the crystal frequency.

$\overline{SYNC}$. *Instruction Sync* (output, active Low). This strobe output is forced Low during the internal clock period preceding an opcode fetch.

Z8603 Protopack Emulator

The Z8603 MPE (Protopack) is used for prototype development and preproduction of mask-programmed applications. The Protopack is a ROMless version of the standard Z8601, housed in a pin-compatible 40-pin package (Figure 12).

To provide pin compatibility and interchangeability with the standard mask-programmed device, the Protopack carries (piggy-backs) a 24-pin socket for a direct interface to program memory (Figure 1). The 24-pin socket is equipped with 12 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2716 EPROM for the first 2K bytes of program memory.

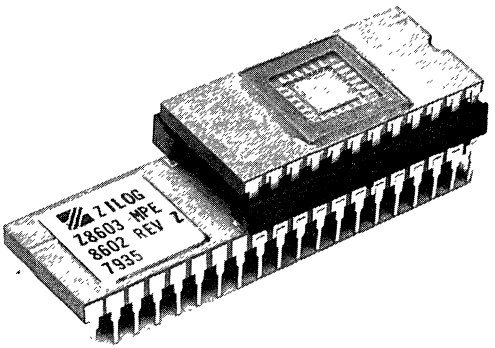


Figure 12. The Z8603 Microcomputer Protopack Emulator

Pin compatibility allows the user to design the pc board for a final 40-pin mask-programmed Z8601, and, at the same time, allows the use of the Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask-programmed Z8601 for large volume production. The Protopack is also useful in small volume applications where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage. For instance, in applications where the same hardware configuration is used with more than one program, the Z8603 Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Instruction Set Notation (Continued)

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

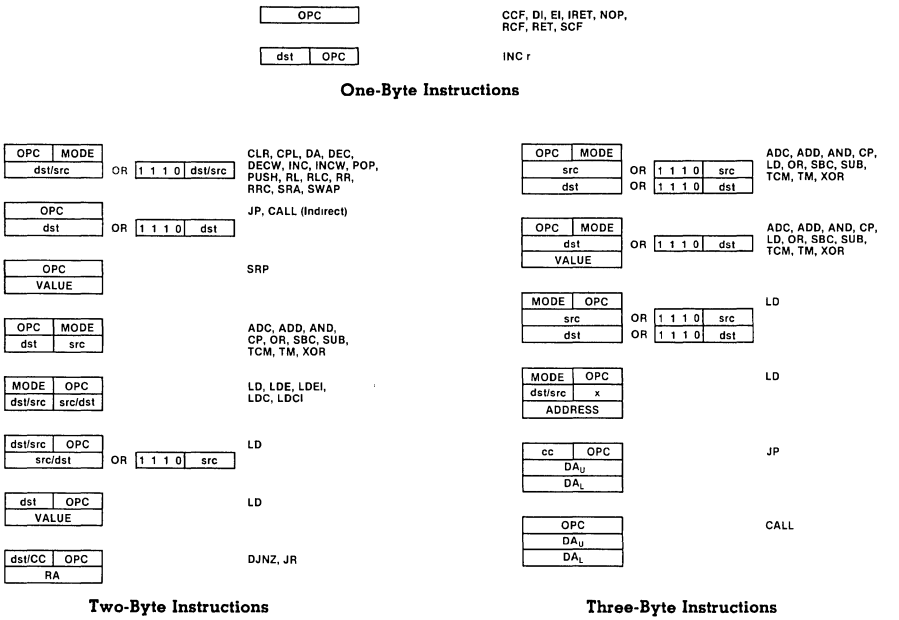
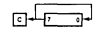
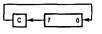
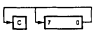
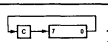
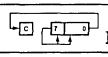
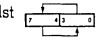


Figure 13. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r r Ir R R R IR IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst 	R IR		90 91	* * * * - -
RLC dst 	R IR		10 11	* * * * - -
RR dst 	R IR		E0 E1	* * * * - -
RRC dst 	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst 	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

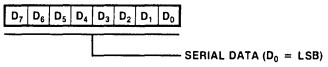
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

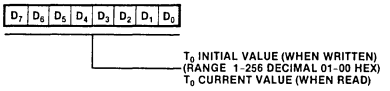
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

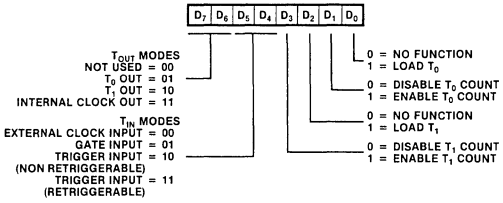
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



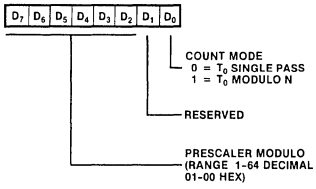
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



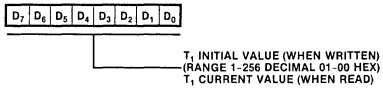
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



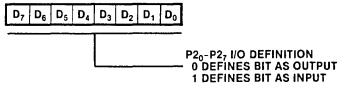
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



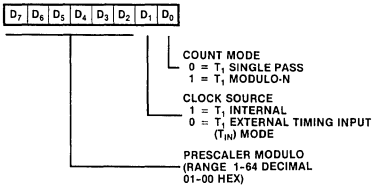
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

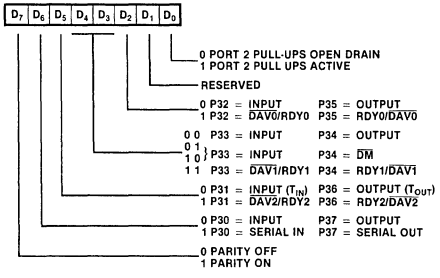
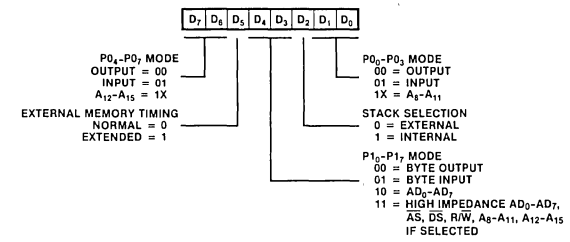
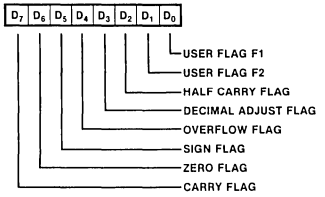


Figure 14. Control Registers

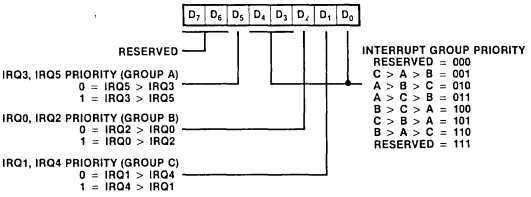
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



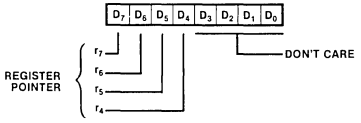
R252 FLAGS
Flag Register
(FC_H; Read/Write)



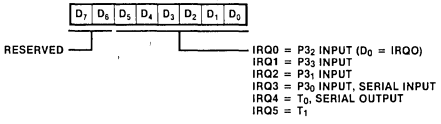
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



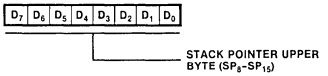
R253 RP
Register Pointer
(FD_H; Read/Write)



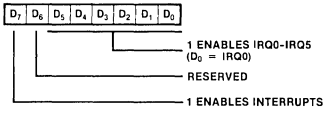
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

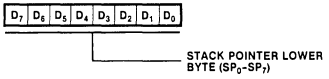
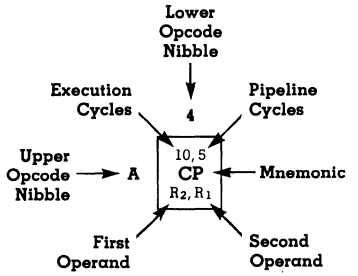


Figure 14. Control Registers

Z8601 Opcode Map

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R1	6,5 **DEC** IR1	6,5 **ADD** r1,r2	6,5 **ADD** r1,Ir2	10,5 **ADD** R2,R1	10,5 **ADD** IR2,R1	10,5 **ADD** R1,IM	10,5 **ADD** IR1,IM	6,5 **LD** r1,R2	6,5 **LD** r2,R1	12/10,5 **DJNZ** r1,RA	12/10,0 **JR** cc,RA	6,5 **LD** r1,IM	12/10,0 **JP** cc,DA	6,5 **INC** r1	
1	6,5 **RLC** R1	6,5 **RLC** IR1	6,5 **ADC** r1,r2	6,5 **ADC** r1,Ir2	10,5 **ADC** R2,R1	10,5 **ADC** IR2,R1	10,5 **ADC** R1,IM	10,5 **ADC** IR1,IM								
2	6,5 **INC** R1	6,5 **INC** IR1	6,5 **SUB** r1,r2	6,5 **SUB** r1,Ir2	10,5 **SUB** R2,R1	10,5 **SUB** IR2,R1	10,5 **SUB** R1,IM	10,5 **SUB** IR1,IM								
3	8,0 **JP** IRR1	6,1 **SRP** IM	6,5 **SBC** r1,r2	6,5 **SBC** r1,Ir2	10,5 **SBC** R2,R1	10,5 **SBC** IR2,R1	10,5 **SBC** R1,IM	10,5 **SBC** IR1,IM								
4	8,5 **DA** R1	8,5 **DA** IR1	6,5 **OR** r1,r2	6,5 **OR** r1,Ir2	10,5 **OR** R2,R1	10,5 **OR** IR2,R1	10,5 **OR** R1,IM	10,5 **OR** IR1,IM								
5	10,5 **POP** R1	10,5 **POP** IR1	6,5 **AND** r1,r2	6,5 **AND** r1,Ir2	10,5 **AND** R2,R1	10,5 **AND** IR2,R1	10,5 **AND** R1,IM	10,5 **AND** IR1,IM								
6	6,5 **COM** R1	6,5 **COM** IR1	6,5 **TCM** r1,r2	6,5 **TCM** r1,Ir2	10,5 **TCM** R2,R1	10,5 **TCM** IR2,R1	10,5 **TCM** R1,IM	10,5 **TCM** IR1,IM								
7	10/12,1 **PUSH** R2	12/14,1 **PUSH** IR2	6,5 **TM** r1,r2	6,5 **TM** r1,Ir2	10,5 **TM** R2,R1	10,5 **TM** IR2,R1	10,5 **TM** R1,IM	10,5 **TM** IR1,IM								
8	10,5 **DECW** RR1	10,5 **DECW** IR1	12,0 **LDE** r1,Irr2	18,0 **LDEI** Ir1,Irr2												6,1 **DI**
9	6,5 **RL** R1	6,5 **RL** IR1	12,0 **LDE** r2,Irr1	18,0 **LDEI** Ir2,Irr1												6,1 **EI**
A	10,5 **INCW** RR1	10,5 **INCW** IR1	6,5 **CP** r1,r2	6,5 **CP** r1,Ir2	10,5 **CP** R2,R1	10,5 **CP** IR2,R1	10,5 **CP** R1,IM	10,5 **CP** IR1,IM								14,0 **RET**
B	6,5 **CLR** R1	6,5 **CLR** IR1	6,5 **XOR** r1,r2	6,5 **XOR** r1,Ir2	10,5 **XOR** R2,R1	10,5 **XOR** IR2,R1	10,5 **XOR** R1,IM	10,5 **XOR** IR1,IM								16,0 **IRET**
C	6,5 **RRC** R1	6,5 **RRC** IR1	12,0 **LDC** r1,Irr2	18,0 **LDCI** Ir1,Irr2				10,5 **LD** r1,x,R2								6,5 **RCF**
D	6,5 **SRA** R1	6,5 **SRA** IR1	12,0 **LDC** r2,Irr1	18,0 **LDCI** Ir2,Irr1	20,0 **CALL*** IRR1		20,0 **CALL** DA	10,5 **LD** r2,x,R1								6,5 **SCF**
E	6,5 **RR** R1	6,5 **RR** IR1		6,5 **LD** r1,Ir2	10,5 **LD** R2,R1	10,5 **LD** IR2,R1	10,5 **LD** R1,IM	10,5 **LD** IR1,IM								6,5 **CCF**
F	6,7 **SWAP** R1	6,7 **SWAP** IR1		6,5 **LD** Ir1,r2		10,5 **LD** R2,IR1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction, fetch cycle appears as a 3-byte instruction

8085-002

Absolute Maximum Ratings

Voltages on all pins with respect to GND.....-0.3 V to +7.0 V

Operating Ambient Temperature.....0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- □ +4.75 V ≤ V_{CC} ≤ +5.25 V
- □ GND = 0 V
- □ 0°C ≤ T_A ≤ +70°C

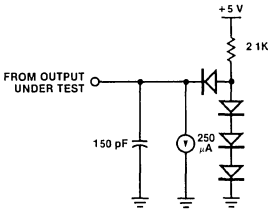


Figure 15. Test Load 1

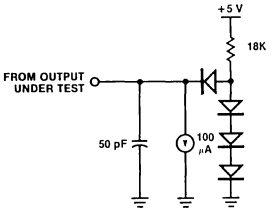


Figure 16. Test Load 2

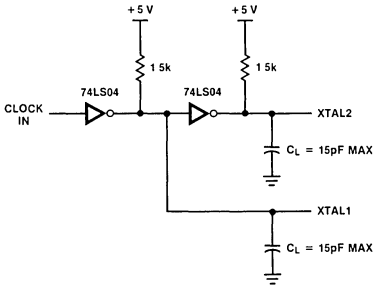


Figure 17. External Clock Interface Circuit

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V		
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA	1
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA	1
I_{IL}	Input Leakage	-10	10	μA	0 V≤ V_{IN} ≤ +5.25 V	
I_{OL}	Output Leakage	-10	10	μA	0 V≤ V_{IN} ≤ +5.25 V	
I_{IR}	Reset Input Current		-50	μA	V_{CC} = +5.25 V, V_{RL} = 0 V	
I_{CC}	V_{CC} Supply Current		180	mA		
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode	
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down	

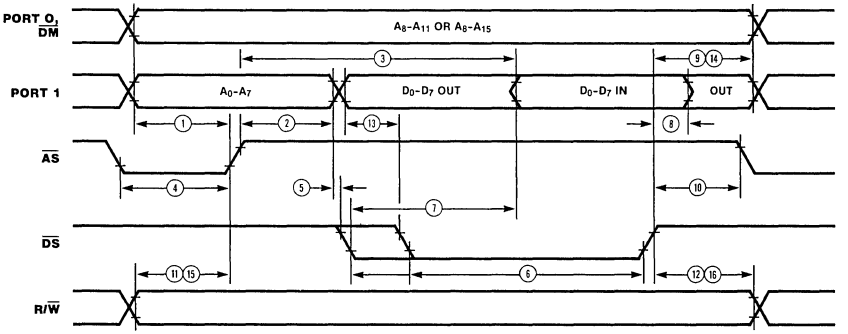
1 For A_0-A_{11}, $\overline{MDS}$, $\overline{SYNC}$, SCLK and IACK on the Z8612 version, I_{OH} = -100 μA and I_{OL} = 1.0 mA.

External I/O or Memory Read and Write Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TdA(AS)	Address Valid to Address Strobe Delay	50		ns	1,2
2	TdAS(A)	Address Strobe to Address Float Delay	60		ns	1,2
3	TdAS(DI)	Address Strobe to Data In Valid Delay		320	ns	1,4
4	TwAS	Address Strobe Width	80		ns	1,2
5	TdA(DS)	Address Float to Data Strobe Delay	0		ns	1
6a	TwDS	Data Strobe Width Read	250		ns	1,3
6b	TwDS	Data Strobe Width Write	160		ns	1,3
7	TdDS(DI)	Data Strobe to Data In Valid Delay		200	ns	1,4
8	ThDS(DI)	Data In Hold Time	0		ns	
9	TdDS(A)	Data Strobe to Address Change Delay	80		ns	1,2
10	TdDS(AS)	Data Strobe to Address Strobe Delay	70		ns	1,2
11	TdR(AS)	Read Valid to Address Strobe Delay	50		ns	1,2
12	TdDS(R)	Data Strobe to Read Change Delay	60		ns	1,2
13	TdDO(DS)	Data Out Valid to Data Strobe Delay	50		ns	1,2
14	TdDS(DO)	Data Strobe to Data Out Change Delay	80		ns	1,2
15	TdW(AS)	Write Valid to Address Strobe Delay	50		ns	1,2
16	TdDS(W)	Data Strobe to Write Change Delay	60		ns	1,2

NOTES:

1. Test Load 1.
2. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
3. Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Data Strobe Width varies according to the instruction being executed.
4. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies; the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).
5. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."

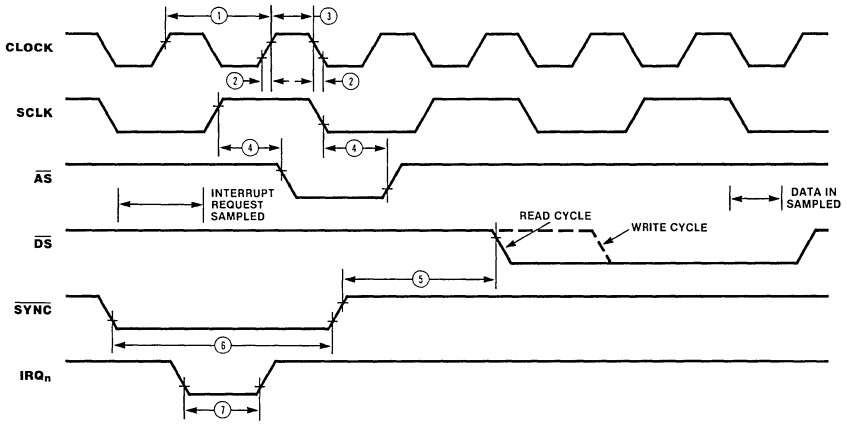


Additional Timing Table

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TpC	Input Clock Period	125	1000	ns	
2	TrC, TfC	Input Clock Rise and Fall Times		25	ns	3
3	TwC	Input Clock Width	37		ns	3
4	TdSC(AS)	System Clock Out to Address Strobe Delay Time			ns	1
5	TdSY(DS)	Instruction Sync Out to Data Strobe Delay Time	200		ns	1,2
6	TwSY	Instruction Sync Out Width	160		ns	1, 2
7	TwI	Interrupt Request via Port 3 Input Width	100		ns	

NOTES:

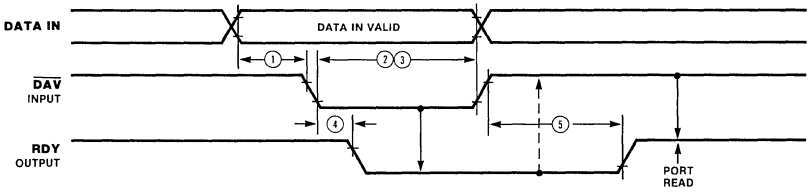
1. Test Conditions use Test Load 1 for SCLK when output through the Port 3 pins and Test Load 2 on the SCLK and SYNC direct outputs on Z8612.
2. Times given assume an 8 MHz crystal input frequency. For lower frequencies, the change in two clock periods must be added.
3. From external clock generator.
4. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."



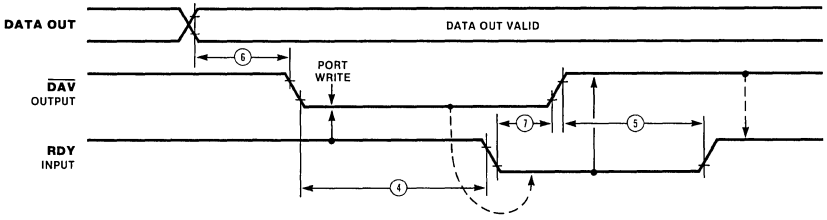
Handshake Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TsDI(DA)	Data In Setup Time	0		ns	
2	ThDA(DI)	Data In Hold Time	230		ns	
3	TwDA	Data Available Width	175		ns	1,2
4a	TdDAL(RY)	Data Available Low to Ready Delay Time	20	175	ns	1,2
4b			0		ns	1,3
5a	TdDAH(RY)	Data Available High to Ready Delay Time		150	ns	1,2
5b			0		ns	1,3
6	TdDO(DA)	Data Out to Data Available Delay Time	50		ns	1
7	TdRY(DA)	Ready to Data Available Delay Time	0	205	ns	1

NOTES
1 Test Load 1
2 Input Handshake
3 Output Handshake



Input Handshake

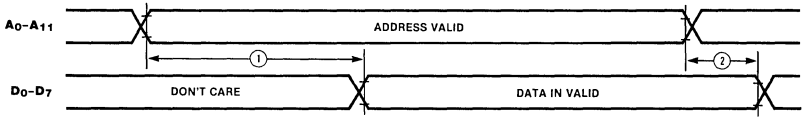


Output Handshake

Z8602, Z8603 Memory Port Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TdA(DI)	Address Valid to Data In Valid Delay Time		460	ns	1
2	ThDI(A)	Data in Hold Time	0		ns	

NOTES
1. Test Load 2
2 Delay times are specified for an input clock frequency of 8 MHz.
3. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".



Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8601	CE	8.0 MHz	Z8 MCU (2K ROM, 40-pin)
Z8601	CS	8.0 MHz	Same as above
Z8601	DE	8.0 MHz	Same as above
Z8601	DS	8.0 MHz	Same as above
Z8601	PE	8.0 MHz	Same as above
Z8601	PS	8.0 MHz	Same as above
Z8602	QE	8.0 MHz	Z8 MCU (2K XROM, 64-pin)
Z8602	QS	8.0 MHz	Same as above
Z8602	QE	8.0 MHz	Same as above
Z8603	RS	8.0 MHz	Z8 MCU (2K XROM, Prototype Device, 40-pin)

NOTES: C = Ceramic, D = Cerdip, P = Plastic, Q = Quip, R = Protopack; E = -40°C to +85°C, S = 0°C to +70°C.

Z8™ Family of Microcomputers Z8611 · Z8612 · Z8613

Product Specification

March 1981

Z8611 Single-Chip Microcomputer with 4K ROM
Z8612 Development Device with Memory Interface
Z8613 Prototyping Device with EPROM Interface

Features

- Complete microcomputer, 4K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 60K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Average instruction execution time of 2.2 μs, maximum of 4.25 μs.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working-register groups in 1.5 μs.
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL compatible.

General Description

The Z8611 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8611 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8611 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 4K bytes of internal ROM, a traditional microprocessor that manages up to 120K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

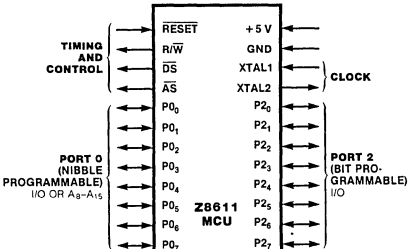


Figure 1. Z8611 MCU Pin Functions

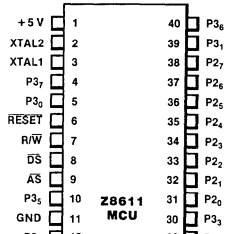


Figure 2. Z8611 MCU Pin Assignments

Architecture

Z8611 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8611 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8611 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 120K bytes of external memory (Figure 3).

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

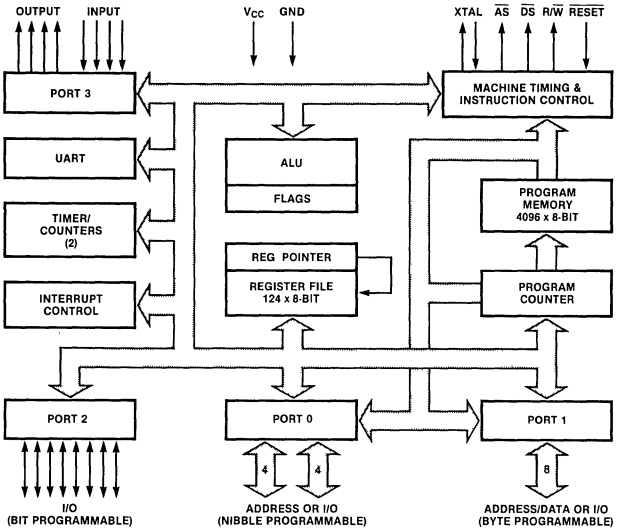


Figure 3. Functional Block Diagram

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8611. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8611 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum) or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

Address Spaces

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 4096 bytes consist of on-chip mask-programmed ROM. At addresses 4096 and greater, the Z8611 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8611 can address 60K bytes of external data memory beginning at locations 4096 (Figure 5). External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and 16 control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8611 instructions can access registers

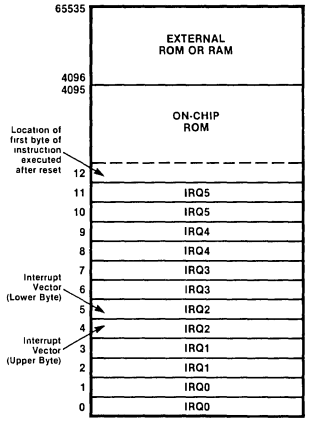


Figure 4. Program Memory Map

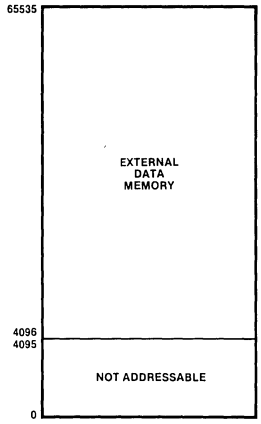


Figure 5. Data Memory Map

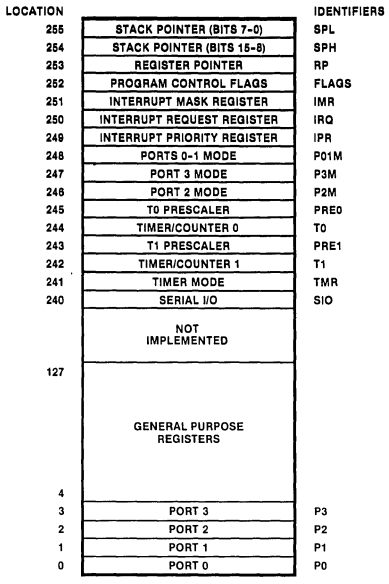


Figure 6. The Register File

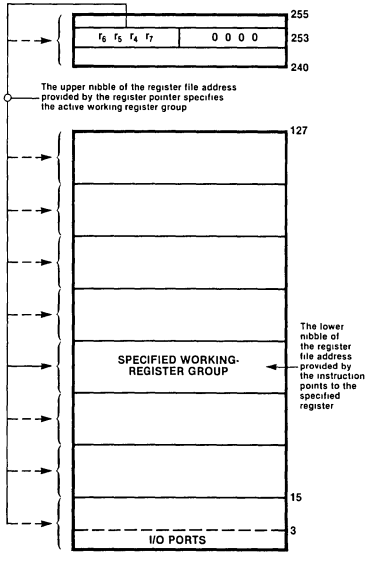


Figure 7. The Register Pointer

Address Spaces (Continued)

directly or indirectly with an 8-bit address field. The Z8611 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 4096 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4–R127).

Serial Input/ Output

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second.

The Z8611 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.

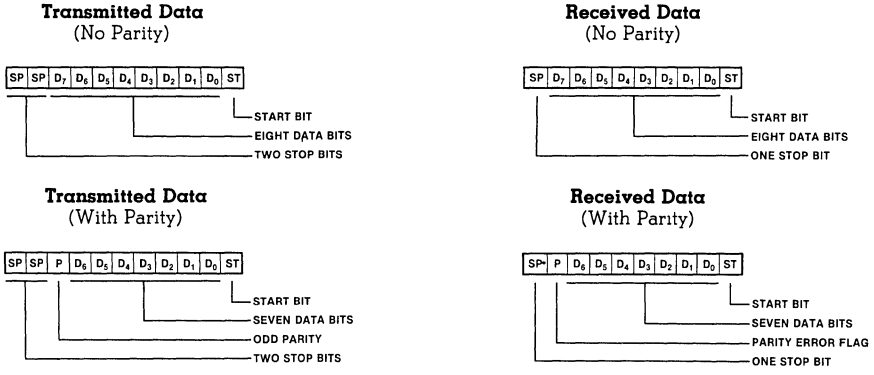


Figure 8. Serial Data Formats

Counter/ Timers

The Z8611 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8611 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8611 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input, and $P3_4$ as a Bus Request output.

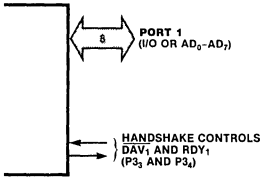


Figure 9a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

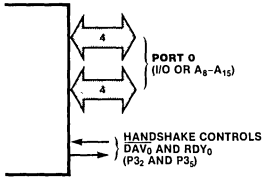


Figure 9b. Port 0

Port 2 bits can be programmed independently as input or output. This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

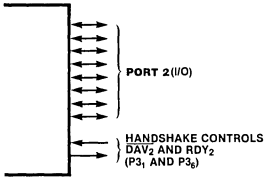


Figure 9c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0-IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

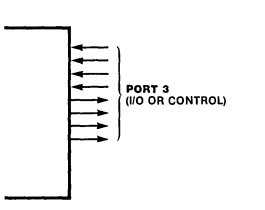


Figure 9d. Port 3

Interrupts

The Z8611 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$–$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8611 interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, series-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors (C_1 = 15 pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, series resonant
- Fundamental type, 8 MHz maximum
- Series resistance, $R_S \leq 100\ \Omega$

Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the V_{MM} (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.

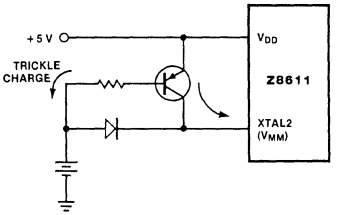


Figure 10. Recommended Driver Circuit for Power Down Operation

Z8612 Development Device

This 64-pin development version of the 40-pin mask-programmed Z8611 (Figure 11) allows the user to prototype the system in hardware with an actual device and to develop the code that is eventually mask-programmed into the on-chip ROM of the Z8611.

The Z8612 is identical to the Z8611 with the following exceptions:

- The internal ROM has been removed.
- The ROM address lines and data lines are buffered and brought out to external pins.
- Control lines for the new memory have been added.

Pin Description. The functions of the Z8612 I/O lines, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, XTAL1, XTAL2 and $\overline{RESET}$ are identical to those of their Z8611 counterparts. The functions of the remaining 24 pins are as follows:

A_0-A_{11}. *Program Memory Address* (outputs). A_0-A_{11} access the first 4K bytes of program memory.

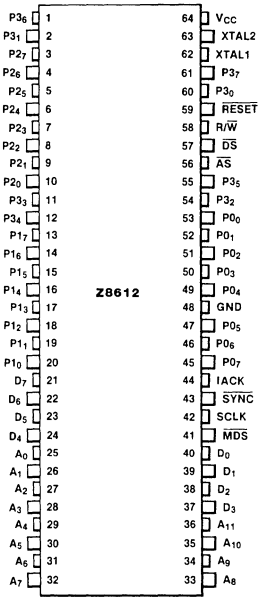


Figure 11. Z8612 Pin Assignments

Z8612 Development Device (Continued)

D_0-D_7. *Program Data* (inputs). Program data from the first 4K bytes of program memory is input through pins D_0-D_7.

IACK. *Interrupt Acknowledge* (output, active High). IACK is driven High in response to an interrupt during the interrupt machine cycle.

$\overline{MDS}$. *Program Memory Data Strobe* (output, active Low). $\overline{MDS}$ is Low during an instruction fetch cycle when the first 4K bytes of program memory are being accessed.

SCLK. *System Clock* (output). SCLK is the internal clock output through a buffer. The clock rate is equal to one-half the crystal frequency.

$\overline{SYNC}$. *Instruction Sync* (output, active Low). This strobe output is forced Low during the internal clock period preceding an opcode fetch.

Z8613 Protopack Emulator

The Z8613 MPE (Protopack) is used for prototype development and preproduction of mask-programmed applications. The Protopack is a ROMless version of the standard Z8611, housed in a pin-compatible 40-pin package (Figure 12).

To provide pin compatibility and interchangeability with the standard mask-programmed device, the Protopack carries (piggy-backs) a 24-pin socket for a direct interface to program memory (Figure 1). The 24-pin socket is equipped with 12 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2732 EPROM for the first 4K bytes of program memory.

Pin compatibility allows the user to design the pc board for a final 40-pin mask-programmed Z8611, and, at the same time, allows the use of the Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask-programmed Z8611 for large volume production. The Protopack is also useful in small volume applications where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage: For instance, in applications where the same hardware configuration is used with more than one program, the Z8613 Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

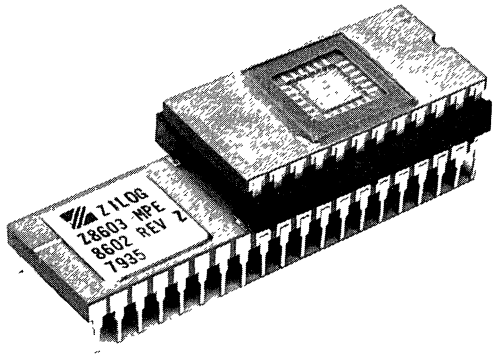


Figure 12. The Z8613 Microcomputer Protopack Emulator

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Instruction Set Notation (Continued)

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

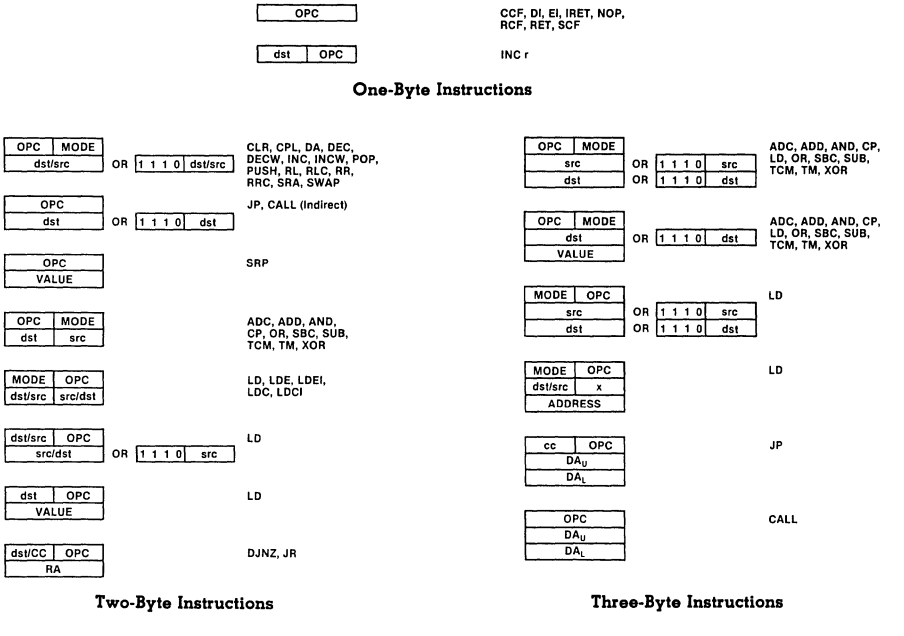


Figure 13. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst	R IR		90 91	* * * * - -
RLC dst	R IR		10 11	* * * * - -
RR dst	R IR		E0 E1	* * * * - -
RRC dst	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

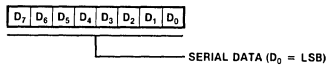
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

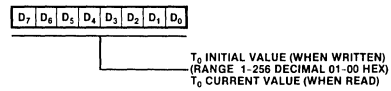
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

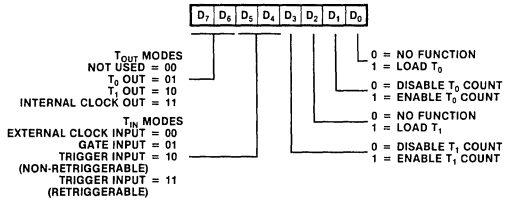
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



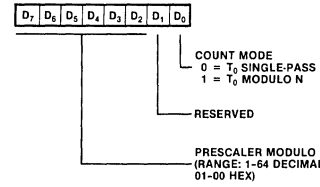
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



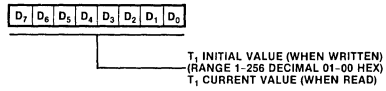
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



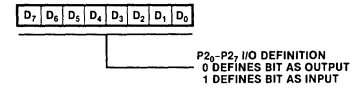
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



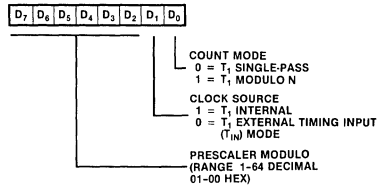
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

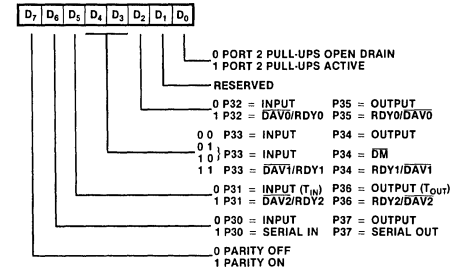
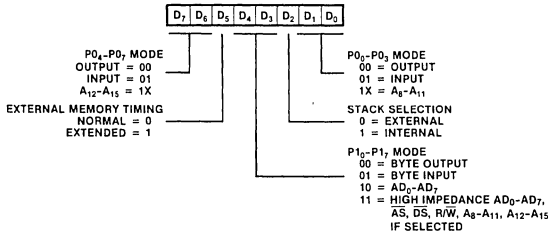
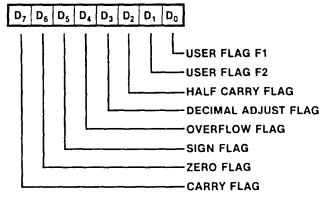


Figure 14. Control Registers

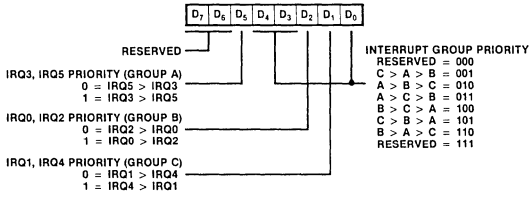
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



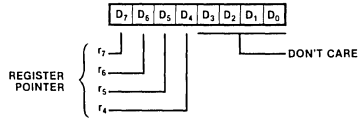
R252 FLAGS
Flag Register
(FC_H; Read/Write)



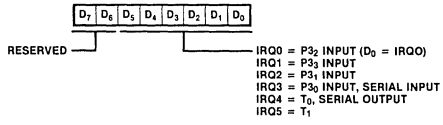
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



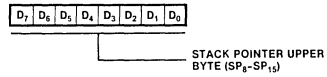
R253 RP
Register Pointer
(FD_H; Read/Write)



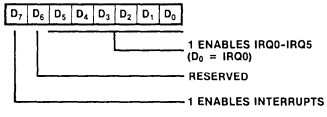
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

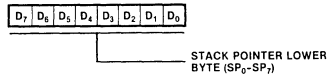
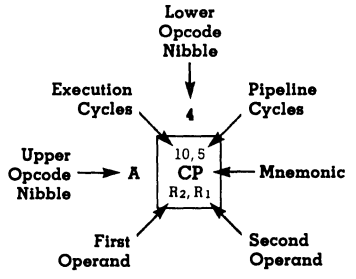


Figure 14. Control Registers

Opcode Map

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R1	6,5 **DEC** IR1	6,5 **ADD** r1, r2	6,5 **ADD** r1, Ir2	10,5 **ADD** R2, R1	10,5 **ADD** IR2, R1	10,5 **ADD** R1, IM	10,5 **ADD** IR1, IM	6,5 **LD** r1, R2	6,5 **LD** r2, R1	12/10,5 **DJNZ** r1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r1	
1	6,5 **RLC** R1	6,5 **RLC** IR1	6,5 **ADC** r1, r2	6,5 **ADC** r1, Ir2	10,5 **ADC** R2, R1	10,5 **ADC** IR2, R1	10,5 **ADC** R1, IM	10,5 **ADC** IR1, IM								
2	6,5 **INC** R1	6,5 **INC** IR1	6,5 **SUB** r1, r2	6,5 **SUB** r1, Ir2	10,5 **SUB** R2, R1	10,5 **SUB** IR2, R1	10,5 **SUB** R1, IM	10,5 **SUB** IR1, IM								
3	8,0 **JP** IRR1	6,1 **SRP** IM	6,5 **SBC** r1, r2	6,5 **SBC** r1, Ir2	10,5 **SBC** R2, R1	10,5 **SBC** IR2, R1	10,5 **SBC** R1, IM	10,5 **SBC** IR1, IM								
4	8,5 **DA** R1	8,5 **DA** IR1	6,5 **OR** r1, r2	6,5 **OR** r1, Ir2	10,5 **OR** R2, R1	10,5 **OR** IR2, R1	10,5 **OR** R1, IM	10,5 **OR** IR1, IM								
5	10,5 **POP** R1	10,5 **POP** IR1	6,5 **AND** r1, r2	6,5 **AND** r1, Ir2	10,5 **AND** R2, R1	10,5 **AND** IR2, R1	10,5 **AND** R1, IM	10,5 **AND** IR1, IM								
6	6,5 **COM** R1	6,5 **COM** IR1	6,5 **TCM** r1, r2	6,5 **TCM** r1, Ir2	10,5 **TCM** R2, R1	10,5 **TCM** IR2, R1	10,5 **TCM** R1, IM	10,5 **TCM** IR1, IM								
7	10/12, 1 **PUSH** R2	12/14, 1 **PUSH** IR2	6,5 **TM** r1, r2	6,5 **TM** r1, Ir2	10,5 **TM** R2, R1	10,5 **TM** IR2, R1	10,5 **TM** R1, IM	10,5 **TM** IR1, IM								
8	10,5 **DECW** RR1	10,5 **DECW** IR1	12,0 **LDE** r1, Irr2	18,0 **LDEI** Ir1, Irr2												6,1 **DI**
9	6,5 **RL** R1	6,5 **RL** IR1	12,0 **LDE** r2, Irr1	18,0 **LDEI** Ir2, Irr1												6,1 **EI**
A	10,5 **INCW** RR1	10,5 **INCW** IR1	6,5 **CP** r1, r2	6,5 **CP** r1, Ir2	10,5 **CP** R2, R1	10,5 **CP** IR2, R1	10,5 **CP** R1, IM	10,5 **CP** IR1, IM								14,0 **RET**
B	6,5 **CLR** R1	6,5 **CLR** IR1	6,5 **XOR** r1, r2	6,5 **XOR** r1, Ir2	10,5 **XOR** R2, R1	10,5 **XOR** IR2, R1	10,5 **XOR** R1, IM	10,5 **XOR** IR1, IM								16,0 **IRET**
C	6,5 **RRC** R1	6,5 **RRC** IR1	12,0 **LDC** r1, Irr2	18,0 **LDCI** Ir1, Irr2				10,5 **LD** r1, x, R2								6,5 **RCF**
D	6,5 **SRA** R1	6,5 **SRA** IR1	12,0 **LDC** r2, Irr1	18,0 **LDCI** Ir2, Irr1	20,0 **CALL*** IRR1		20,0 **CALL** DA	10,5 **LD** r2, x, R1								6,5 **SCF**
E	6,5 **RR** R1	6,5 **RR** IR1		6,5 **LD** r1, Ir2	10,5 **LD** R2, R1	10,5 **LD** IR2, R1	10,5 **LD** R1, IM	10,5 **LD** IR1, IM								6,5 **CCF**
F	6,7 **SWAP** R1	6,7 **SWAP** IR1		6,5 **LD** Ir1, r2		10,5 **LD** R2, IR1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- □ $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- □ GND = 0 V
- □ $0°C \leq T_A \leq +70°C$

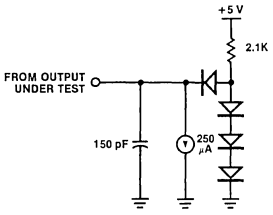


Figure 15. Test Load 1

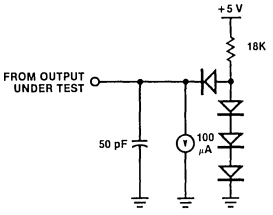


Figure 16. Test Load 2

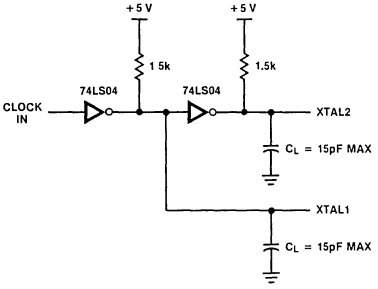


Figure 17. External Clock Interface Circuit

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V		
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA	1
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA	1
I_{IL}	Input Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$	
I_{OL}	Output Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$	
I_{IR}	Reset Input Current		-50	µA	V_{CC} = +5.25 V, V_{RL} = 0 V	
I_{CC}	V_{CC} Supply Current		180	mA		
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode	
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down	

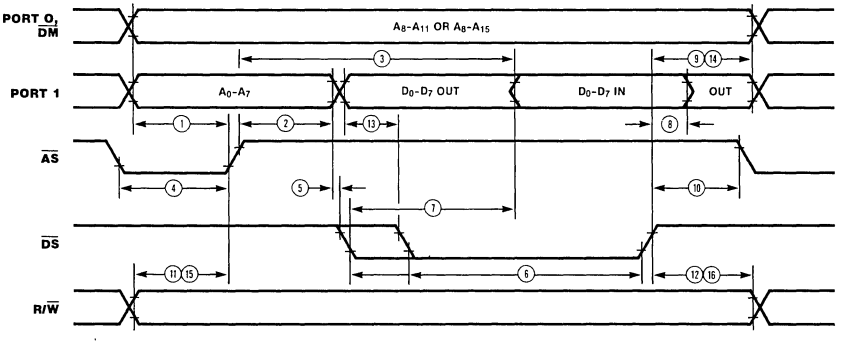
1. For A_0-A_{11}, $\overline{MDS}$, $\overline{SYNC}$, SCLK and IACK on the Z8612 version, I_{OH} = -100 µA and I_{OL} = 1 0 mA

External I/O or Memory Read and Write Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TdA(AS)	Address Valid to Address Strobe Delay	50		ns	1,2
2	TdAS(A)	Address Strobe to Address Float Delay	60		ns	1,2
3	TdAS(DI)	Address Strobe to Data In Valid Delay		320	ns	1,4
4	TwAS	Address Strobe Width	80		ns	1,2
5	TdA(DS)	Address Float to Data Strobe Delay	0		ns	1
6a	TwDS	Data Strobe Width Read	250		ns	1,3
6b	TwDS	Data Strobe Width Write	160		ns	1,3
7	TdDS(DI)	Data Strobe to Data In Valid Delay		200	ns	1,4
8	ThDS(DI)	Data In Hold Time	0		ns	
9	TdDS(A)	Data Strobe to Address Change Delay	80		ns	1,2
10	TdDS(AS)	Data Strobe to Address Strobe Delay	70		ns	1,2
11	TdR(AS)	Read Valid to Address Strobe Delay	50		ns	1,2
12	TdDS(R)	Data Strobe to Read Change Delay	60		ns	1,2
13	TdDO(DS)	Data Out Valid to Data Strobe Delay	50		ns	1,2
14	TdDS(DO)	Data Strobe to Data Out Change Delay	80		ns	1,2
15	TdW(AS)	Write Valid to Address Strobe Delay	50		ns	1,2
16	TdDS(W)	Data Strobe to Write Change Delay	60		ns	1,2

NOTES.

1 Test Load 1.
2. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
3 Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width The Data Strobe Width varies according to the instruction being executed.
4. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies; the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI)
5. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."

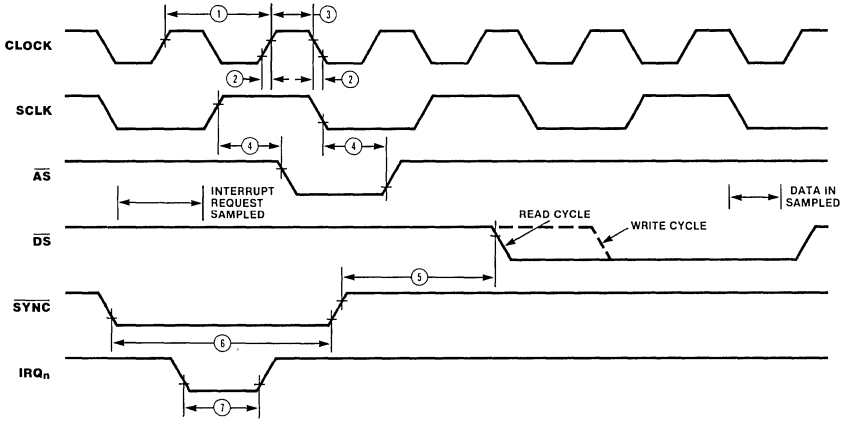


Additional Timing Table

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TpC	Input Clock Period	125	1000	ns	
2	TrC, TfC	Input Clock Rise and Fall Times		25	ns	3
3	TwC	Input Clock Width	37		ns	3
4	TdSC(AS)	System Clock Out to Address Strobe Delay Time			ns	1
5	TdSY(DS)	Instruction Sync Out to Data Strobe Delay Time	200		ns	1,2
6	TwSY	Instruction Sync Out Width	160		ns	1, 2
7	TwI	Interrupt Request via Port 3 Input Width	100		ns	

NOTES:

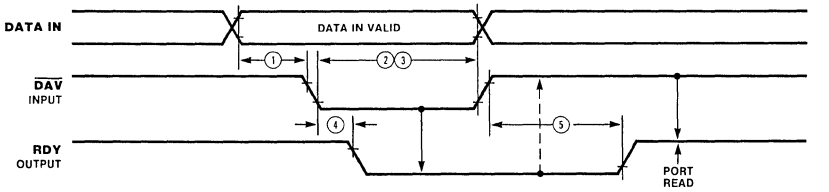
1. Test Conditions use Test Load 1 for SCLK when output through the Port 3 pins and Test Load 2 on the SCLK and SYNC direct outputs on Z8612.
2. Times given assume an 8 MHz crystal input frequency. For lower frequencies, the change in two clock periods must be added.
3. From external clock generator
4. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0."



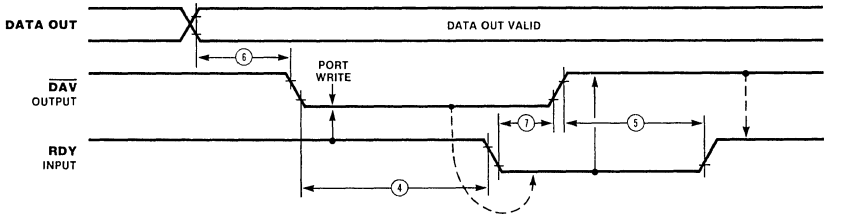
Handshake Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TsDI(DA)	Data In Setup Time	0		ns	
2	ThDA(DI)	Data In Hold Time	230		ns	
3	TwDA	Data Available Width	175		ns	1,2
4a	TdDAL(RY)	Data Available Low to Ready Delay Time	20	175	ns	1,2
4b			0		ns	1,3
5a	TdDAH(RY)	Data Available High to Ready Delay Time		150	ns	1,2
5b			0		ns	1,3
6	TdDO(DA)	Data Out to Data Available Delay Time	50		ns	1
7	TdRY(DA)	Ready to Data Available Delay Time	0	205	ns	1

NOTES:
1. Test Load 1
2. Input Handshake
3. Output Handshake



Input Handshake

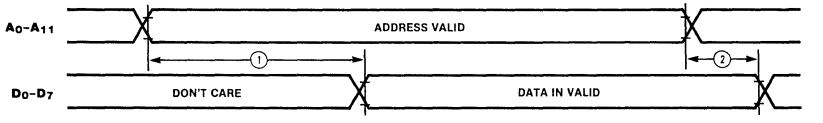


Output Handshake

Z8612, Z8613 Memory Port Timing

Number	Symbol	Parameter	Min	Max	Unit	Notes
1	TdA(DI)	Address Valid to Data In Valid Delay Time		460	ns	1
2	ThDI(A)	Data in Hold Time	0		ns	

NOTES.
1 Test Load 2
2. Delay times are specified for an input clock frequency of 8 MHz.
3. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".



Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8611	CE	8.0 MHz	Z8 MCU (4K ROM, 40-pin)
Z8611	CS	8.0 MHz	Same as above
Z8611	DE	8.0 MHz	Same as above
Z8611	DS	8.0 MHz	Same as above
Z8611	PE	8.0 MHz	Same as above
Z8611	PS	8.0 MHz	Same as above
Z8611	QE	8.0 MHz	Z8 MCU (4K XROM, 64-pin)
Z8612	QS	8.0 MHz	Same as above
Z8612	QE	8.0 MHz	Same as above
Z8613	RS	8.0 MHz	Z8 MCU (4K XROM, Prototype Device, 40-pin)

NOTES: C = Ceramic, D = Cerdip, P = Plastic, Q = Quip, R = Protopack, E = -40°C to +85°C, S = 0°C to +70°C

Z8™ Family
Z8681 Microcomputer

Zilog

Product Brief

March 1981

Features

- "ROMless" version of the Z8601 single-chip microcomputer, capable of addressing up to 128K bytes of external memory space.
- Up to 24 programmable I/O lines.
- 40-pin package, single +5 V supply, all pins TTL compatible.

General Description

The Z8681 MCU is the "ROMless" version of the Z8601 single-chip microcomputer and offers all the outstanding features of the Z8 Family architecture. Using the Z8681, it is possible to design a powerful microprocessor system incorporating a minimum number of support devices.

Port 1 is configured to function as a multiplexed Address/Data bus (AD_0–AD_7), while Port 0 is software configurable to output address bits A_8–A_{15}. This provides for program memory and data memory space of up to 64K bytes each.

Located on-chip are 144 bytes of RAM, organized as a register file of 124 general-purpose registers, 16 control and status registers, and three I/O port registers. (Port 1 cannot be utilized as an I/O register.) This file is divided into groups of working registers in such a way that short format instructions may be used to quickly access a register within a certain group.

Functional Description

Register File. The internal register organization of the Z8681 centers around a 144-byte random-access register file composed of 124 general-purpose registers, 16 control registers, and the three I/O port registers. Any general-purpose register can be an accumulator, address pointer, index register, or part of the internal stack. The register file is divided into nine groups of 16 working registers. A register pointer uses short-format instructions to quickly access any one of the nine groups, resulting in fast and easy task-switching.

I/O Ports. The I/O ports (Ports 0, 2, and 3) are software configurable as input, output, or additional address lines. These ports can also provide timing, status signals, and serial or parallel I/O (with or without handshake).

I/O port space is mapped into the register file, creating an efficient and convenient means of moving data.

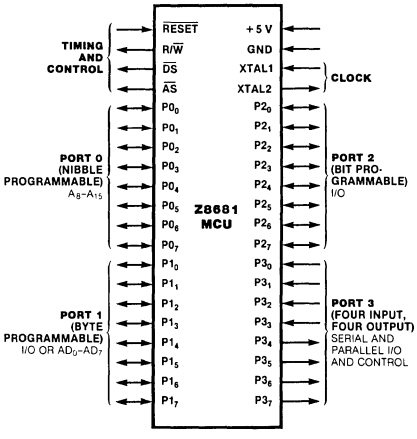


Figure 1. Pin Functions

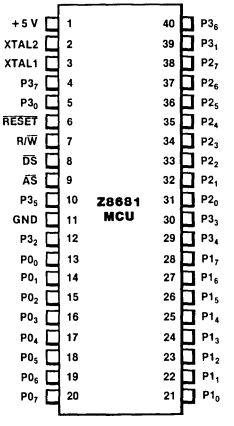


Figure 2. Pin Assignments

Functional Description (Continued)

Interrupts. The Z8681 can respond to six separate interrupts from eight sources. The interrupts are maskable and prioritized by software control, thus allowing greater design flexibility.

Using vectored interrupts, control is automatically passed to the appropriate service routine. The interrupts are organized as four external lines and four internal status signals. The internal interrupts control the serial port handshake and the two counter/timers.

UART. The Z8681 also offers the serial I/O capability of interfacing to asynchronous data communications. The on-chip counter (T0) is used to supply the baud rate for the serial data transfer. The UART is capable of transferring data at a rate of up to 62.5K b/s.

Counter/Timers. Also on-chip are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. Both counter/timers can operate independently of the processor instruction sequence, thereby unburdening the program from such time-critical operations as event-counting or elapsed-time calculations. The counters can be started, stopped, continued, or restarted from the initial value by program control.

Instruction Set for the Z8681

The basic instruction set for the Z8681 consists of 47 instruction types and utilizes seven addressing modes. The instructions can operate on several types of data elements, including individual bits, 4-bit BCD characters, bytes, or words.

All 124 general-purpose registers can be used as accumulators, address pointers, index registers, or as internal stack, resulting in fast data manipulation for real-time applications. The internal pipelining of instructions dramatically increases throughput by allowing instruction fetches during the previous instruction execution cycles.

Z8681 Applications

The Z8681 is a Z-BUS-compatible device and can be interfaced to various Z-BUS peripherals such as the Z-CIO, Z-SCC, or FIO. Due to the flexibility of Port 0 and the data memory select feature, the Z8681 can also support a great variety of memory configurations. Figures 3 and 4 illustrate two design approaches using the Z8681.

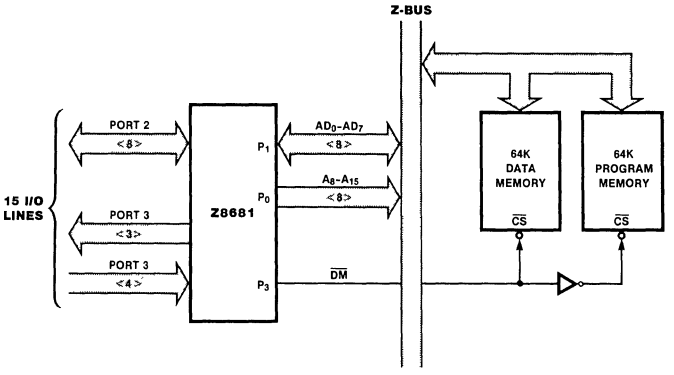


Figure 3. Z8681 Interfacing to Memory-Mapped I/O

Z8681 Applications (Continued)

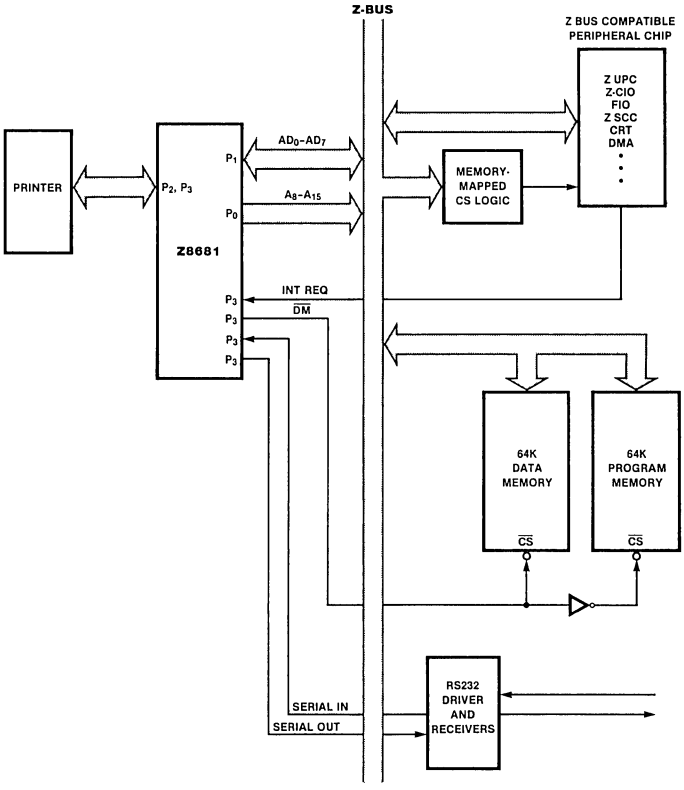


Figure 4. Z8681 Interfacing to External Memory

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8681	CE	8.0 MHz	Z8 MCU (ROMless, 40-pin)	Z8681	DS	8.0 MHz	Z8 MCU (ROMless, 40-pin)
Z8681	CS	8.0 MHz	Same as above	Z8681	PE	8.0 MHz	Same as above
Z8681	DE	8.0 MHz	Same as above	Z8681	PS	8.0 MHz	Same as above

NOTES C = Ceramic, D = Cerdip, P = Plastic, E = -40°C to +85°C, S = 0°C to +70°C

3

Memory

Zilog

Z6132 4K x 8 Quasi-Static RAM

Zilog

Product Specification

March 1981

Description

The Zilog Z6132 is a +5 V intelligent MOS dynamic RAM organized as 4096 words by eight bits. Although it uses single-transistor dynamic storage cells, the Z6132 effectively functions as a static RAM because it performs and controls its own refresh. This eliminates the need for external refresh circuitry and combines the convenience of a static RAM with the high density and low power consumption normally associated with a dynamic RAM.

The Z6132 is particularly suited for microprocessor and minicomputer applications where its byte-wide organization, transparent self-refresh and single supply voltage reduce the parts count and simplify the design.

The Z6132 uses high-performance depletion-load double-poly n-channel silicon-gate MOS technology with a mixture of static and dynamic circuitry that provides a small memory cell, fast access and low power consumption. The Z6132 has separate pins for addresses and bidirectional data I/O to provide maximum flexibility in its application.

The circuit is packaged in an industry-standard 28-pin DIP and pin compatible with the proposed JEDEC standard.

The Z6132 conforms with the Z-Bus specification used by the new generation of Zilog microprocessors, the Z8 and Z8000.

Features

- Byte-wide organization: 4096 words by eight bits
- Access and cycle times guaranteed over voltage and temperature range:

Part Number	Access Time	Cycle Time
Z6132-3	200 ns	350 ns
Z6132-4	250 ns	375 ns
Z6132-5	300 ns	425 ns
Z6132-6	350 ns	450 ns

- All inputs and outputs are TTL compatible
- Low power consumption: 250 mW active, 125 mW stand-by.
- Industry-standard 28-pin DIP with JEDEC-recommended pinout
- Automatic self-refresh scheme with slow-and fast-cycle modes.
- On-chip substrate bias generator.
- Interfaces readily to Z8 and Z8000.

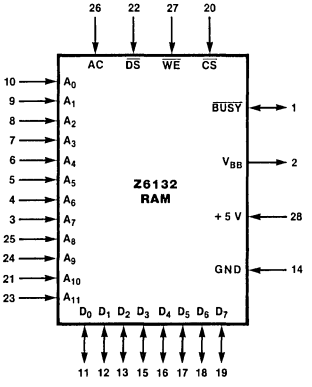


Figure 1. Logic Symbol

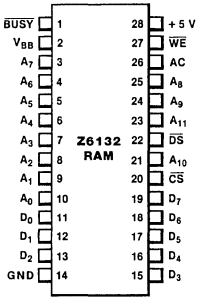


Figure 2. Pin Assignments

Pin Names

A_0-A_{11}	Address inputs
D_0-D_7	Data Inputs/Outputs (3-state)
$\overline{AC}$	Address Clock input (rising edge)
$\overline{DS}$	Data Strobe input (active Low)
$\overline{WE}$	Write Enable input (active Low)
$\overline{CS}$	Chip Select input (active Low)
$\overline{BUSY}$	Busy output (active Low; open drain) and Refresh Mode Control input
V_{BB}	Negative Substrate Bias output
V_{CC}	+5 V supply connection
V_{SS}	0 V = Ground connection

Functional Description

The Z6132 4K x 8 quasi-static RAM is organized as two separate blocks, each having two sets of 64 rows on either side of the 128 sense amplifiers (Figure 3). Both blocks have separate and independent row address buffers and decoders, but they share the column decoder and the internal 8-bit wide data path. The two sets of row address decoders are addressed either by the address inputs A_1–A_7 or by the internal 7-bit refresh counter. The least significant address input (A_0) selects one of the two blocks for external access. While the selected block performs a read or write operation, the other memory block uses the refresh counter address to refresh one row. Details of the self-refresh mechanism are explained later.

A memory cycle starts when the rising edge of Address Clock (AC) clocks in Chip Select ($\overline{CS}$), A_0, and Write Enable ($\overline{WE}$). If the chip is not selected ($\overline{CS}$ = High), all other inputs are ignored for the rest of the cycle (that is, until the next rising edge of AC). Both memory blocks are self refreshed by the 7-bit refresh counter. If the chip is selected ($\overline{CS}$ = Low), the 12 address bits and the Write Enable bit are clocked into their registers. A_0 determines which block is addressed by A_1–A_{11}; the other block is refreshed by the 7-bit refresh counter.

The Chip Select and Address inputs must be valid only during a short hold time after the rising edge of AC. This allows address/data multiplexing, because data I/O is controlled by a separate control input Data Strobe ($\overline{DS}$).

Read Cycle

A read cycle is initiated by the rising edge of Address Clock (AC) while Chip Select ($\overline{CS}$) is Low and Write Enable ($\overline{WE}$) High. A Low level on the Data Strobe ($\overline{DS}$) input activates the Data outputs after a specified delay from the rising edge of AC as well as the falling edge of $\overline{DS}$, whichever comes later. During a read operation, $\overline{DS}$ is nothing but a static Output Enable signal.

Write Cycle

A write cycle is initiated by the rising edge of Address Clock (AC) while Chip Select ($\overline{CS}$) is Low and Write Enable ($\overline{WE}$) is Low.

The $\overline{WE}$ input is checked again at the beginning (falling edge) of Data Strobe ($\overline{DS}$). If $\overline{WE}$ is still Low, this falling edge of $\overline{DS}$ edge-triggers the data on the D_0–D_7 inputs into the addressed memory location. Data must be valid only during a short hold time after the falling edge of $\overline{DS}$.

Write Inhibit Cycle

After a write cycle has been initiated, the actual write operation can still be aborted by pulling $\overline{WE}$ High again before the falling edge of $\overline{DS}$. This write inhibit cycle is a special feature that permits starting a write cycle early at AC time, but still allows the option of inhibiting the write operation later at $\overline{DS}$ time.

Note: *Whenever a write cycle has been initiated, it must be accompanied by a High-to-Low transition on the Data Strobe input.*

Maximum Cycle Time

The maximum read or write cycle time requirements (15,000 and 800 ns) do not apply to any individual cycle. They are specified to guarantee a complete refresh in a 2 ms period.

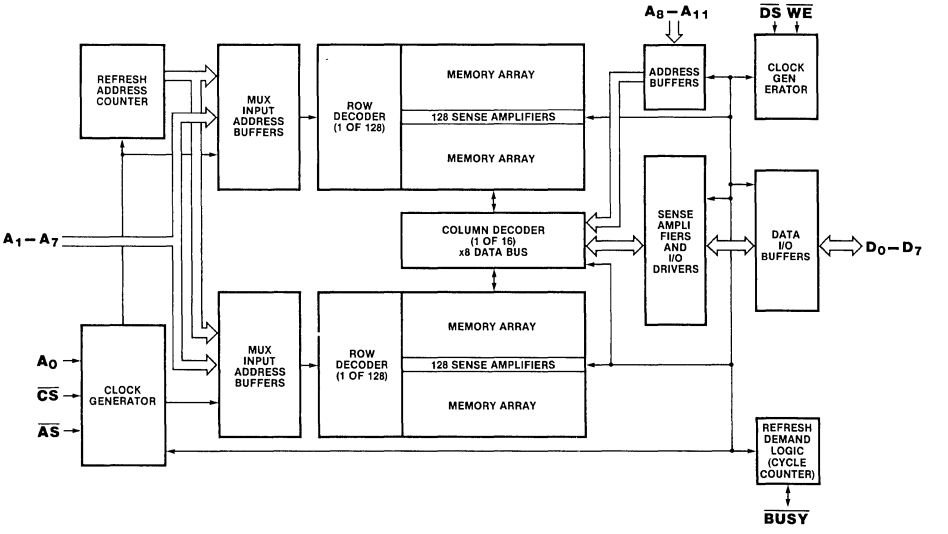


Figure 3. Z6132 Block Diagram

Self-Refresh Operation

The Z6132 stores data in single-transistor dynamic cells that must be refreshed at least every 2 ms. Each of the two memory blocks contains 16,384 cells and requires 128 refresh cycles to completely refresh the array.

The Z6132 operates in one of two user-selectable self-refresh modes, each satisfying the refresh time requirements. On the basis of the available memory cycle time, the user can decide to use either the Long Cycle-Time Refresh Mode or the Short Cycle-Time Refresh Mode.

Long Cycle-Time Refresh Mode

This is the simplest self-refresh mode, and is selected by permanently grounding the $\overline{BUSY}$ output pin. Every memory cycle in this mode consists of a memory operation followed by a refresh operation on both blocks, after which the refresh counter is incremented. Internally, the complete cycle consists of a 4-phase sequence: 1. Memory read, write, or write inhibit. 2. Precharge. 3. Refresh. 4. Precharge. These internal operations are automatic and transparent to the user. When the chip is not selected ($\overline{CS}$ = High when AC goes High), the first two phases are omitted.

There are two important requirements: the memory cycle time must always be longer than the TC (Min) value specified for $\overline{BUSY}$ = Low and there must be at least 128 Address Clocks in any 2 ms period.

The Long Cycle-Time Refresh mode is the one most practical for microprocessor applications, where the cycle time usually exceeds 700 ns.

Short Cycle-Time Refresh Mode

This is a more sophisticated self-refresh mode that allows operation at any cycle time down to the specified minimum value.

The user selects this mode by pulling the $\overline{BUSY}$ output pin High through a pull-up resistor (typically 1 kΩ) to V_{CC}. The $\overline{BUSY}$ outputs of several Z6132 chips can be or-tied together.

In this mode, the Z6132 always performs a refresh operation on the memory block that is not being addressed from the outside. The refresh counter is incremented whenever it is meaningful, as explained in the following text.

Deselect Self-Refresh. If the chip is deselected ($\overline{CS}$ = High when AC goes High), both blocks are refreshed and the refresh counter is incremented after every cycle.

Odd/Even Self-Refresh. If the chip is selected ($\overline{CS}$ = Low when AC goes High), the refresh counter refreshes the block that is not addressed by A_0. The refresh counter is incremented after an even and an odd address have occured. This self-refresh scheme takes advantage of the inherent sequential nature of most memory addressing.

Cycle-Count Self-Refresh. Normally the deselect and odd/even self-refresh schemes step through 128 refresh addresses in less than 2 ms. To guarantee proper refresh operation even in the exceptional case when the memory is continually selected and addressed by a long string of all even or all odd addresses, a built-in cycle counter activates the $\overline{BUSY}$ output and requests one longer memory cycle to append a refresh operation. This internal cycle counter is reset whenever the refresh counter is incremented. The cycle counter then counts memory cycles and activates the $\overline{BUSY}$ output when it reaches a count of 17.

$\overline{BUSY}$ is fed into the $\overline{WAIT}$ input of most microprocessors. $\overline{BUSY}$ is a request to the CPU for a longer memory cycle and is kept Low until the refresh cycle has started. $\overline{BUSY}$ only becomes active when the Z6132 has been selected and addressed with all odd or all even addresses for 17 consecutive Address Strobes.

Mixed Cycle Time Refresh Mode

External logic can be used to select between Long and Short Cycle Time Refresh modes by controlling the $\overline{BUSY}$ pin as an input. The Timing Diagram (parameters 25 through 27) shows when the internal logic interrogates the $\overline{BUSY}$ input.

When $\overline{BUSY}$ is Low the cycle must be long, both blocks are refreshed and the refresh counter is incremented every cycle.

When $\overline{BUSY}$ is High, the cycle can be short and the refresh operation is performed as described under Short Cycle Time Refresh Mode.

The external logic must guarantee proper refresh timing. If the Z6132 received a sequence of 17 consecutive all odd or all even addresses while it was continuously selected and $\overline{BUSY}$ was held High, the $\overline{BUSY}$ output will go Low as described before.

A current limiting resistor of ~ 1kΩ should be inserted if the $\overline{BUSY}$ pin is driven by TTL logic.

External logic, as shown in Figure 4 can detect the fact that the memory requires a long cycle time and can pull the CPU $\overline{WAIT}$ input Low.

Note that the cycle time in most microprocessor applications is so long that the simple Long Cycle Time Refresh Mode is sufficient.

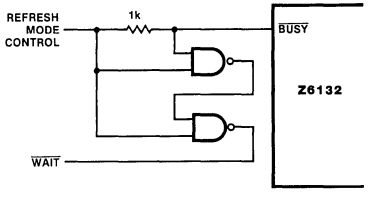
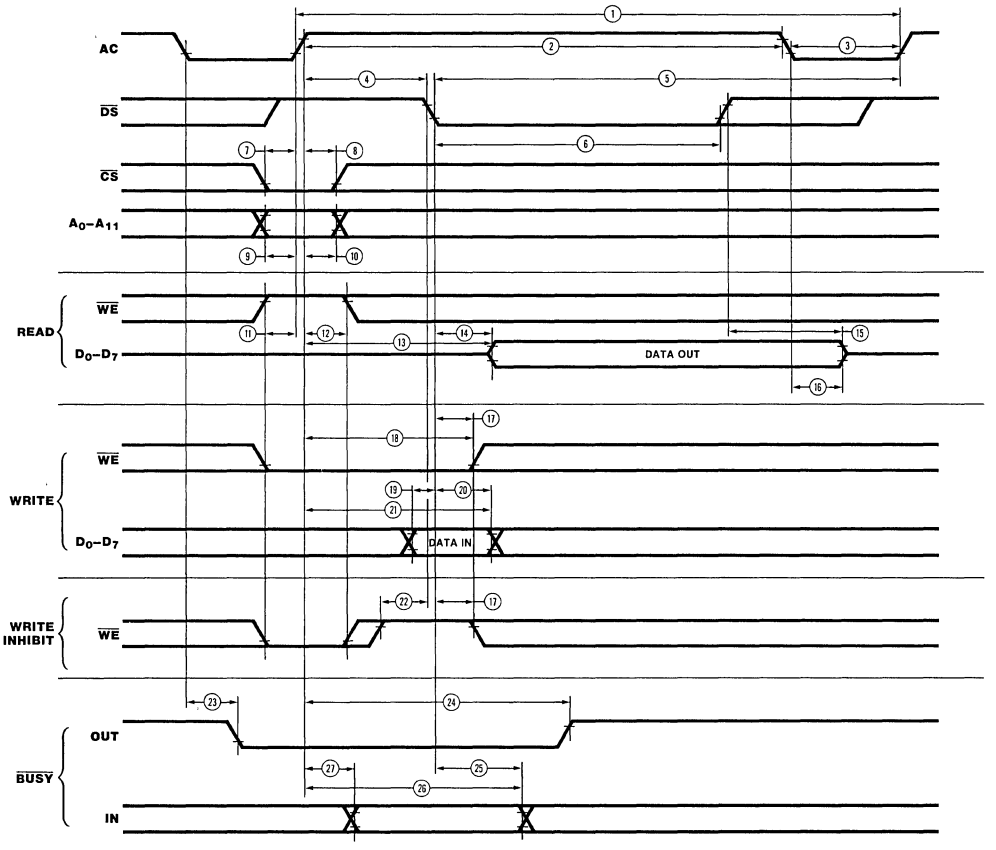


Figure 4. External $\overline{WAIT}$ Generation

AC Electrical Characteristics



No.	Symbol	Parameter	Z6132-3[7] Min (ns)	Z6132-3[7] Max (ns)	Z6132-4 Min (ns)	Z6132-4 Max (ns)	Z6132-5 Min (ns)	Z6132-5 Max (ns)	Z6132-6 Min (ns)	Z6132-6 Max (ns)	Notes
1	TC	Read or Write Cycle Time	650	15000	700	15000	725	15000	750	15000	1
			350	800	375	800	425	800	450	800	2
2	TwACh	AC Width (High)	480		510		550		610		1
			230		260		280		310		2
3	TwACl	AC Width (Low)	40		50		60		60		
4	TdAC(DS)	$\overline{AC}$ ↑ to $\overline{DS}$ ↓	10		10		10		10		
5	TdDS(AC)	$\overline{DS}$ ↓ to AC ↑	550		580		610		640		1
			250		275		310		340		2
6	TwDS	$\overline{DS}$ Width (Low)	120		140		160		180		
7	TsCS(AC)	$\overline{CS}$ Setup Time to AC ↑	0		0		0		0		
8	ThCS(AS)	$\overline{CS}$ Hold Time to AC ↑	40		45		50		55		
9	TsA(AC)	Address Setup Time to AC ↑	0		0		0		0		
10	ThA(AC)	Address Hold Time to AC ↑	40		45		50		55		
11	TsW(AC)	$\overline{WE}$ Setup Time to AC ↑	-10		-15		-20		-25		
12	ThW(AC)	$\overline{WE}$ Hold Time to AC ↑	60		70		80		80		
13	TdAC(DO)	$\overline{AC}$ ↑ to Data Out		200		250		300		350	3
14	TdDS(DO)	$\overline{DS}$ ↓ to Data Out		70		80		90		100	3
15	TdDS(DOz)	$\overline{DS}$ ↑ to Data Out Float	30	70	35	80	40	90	45	100	4
16	TdAC(DOz)	$\overline{AC}$ ↓ to Data Out Float	30	70	35	80	40	90	45	100	4
17	ThW(DS)	$\overline{WE}$ Hold Time to $\overline{DS}$ ↓	60		70		80		90		3
18	TsW(AC)	$\overline{WE}$ Hold Time to AC ↑	120		130		140		150		3
19	TsDI(DS)	Data In Setup Time to $\overline{DS}$ ↓	0		0		0		0		
20	ThDI(AC)	Data In Hold Time to $\overline{DS}$ ↓	45		50		60		70		3
21	ThDI(AC)	Data In Hold Time to $\overline{AC}$ ↑	120		130		140		150		3
22	TsWh(DS)	$\overline{WE}$ High Setup Time to $\overline{DS}$ ↓	10		10		10		10		
23	TdAC(Bl)	$\overline{AC}$ ↓ to $\overline{BUSY}$ Out ↓		80		90		100		110	
24	TdAC(Bh)	$\overline{AC}$ ↑ to $\overline{BUSY}$ Out ↑		400		450		500		550	5
				80		90		100		110	6
25	ThB(DS)	$\overline{BUSY}$ In Hold Time to $\overline{DS}$ ↓	70		80		90		100		3
26	ThB(AC)	$\overline{BUSY}$ In Hold Time to AC ↑	150		160		170		180		3
27	TsB(AC)	$\overline{BUSY}$ In Setup Time to AC ↑	-40		-50		-60		-70		

NOTES:
1 $\overline{BUSY}$ = Low.
2 $\overline{BUSY}$ = High
3 Whichever is later.
4 Whichever is earlier
5 Selected.
6 Deselected
7 Available second half of 1981.

Substrate Bias Generator

The Z6132 contains an on-chip negative substrate-bias generator, which is a simple dc-to-dc converter that generates a substrate-bias voltage of -2.5 to -3 V. This reduces parasitic junction capacitances and thus increases circuit speed. The substrate bias output V_{BB} should be decoupled externally with an $\approx 0.1\ \mu F$ ceramic capacitor to V_{SS} (ground).

Power-Up

After applying V_{CC}, it is necessary to wait 20 ms to charge the substrate bias decoupling capacitor. Moreover, the 6132 requires sixteen selected or deselected memory cycles before proper operation is attained.

Absolute Maximum Ratings

Voltages on all pins (except V_{BB}) with respect to GND.....-0.5 V to +7.0 V

Operating Ambient Temperature.....0°C to +70°C

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 characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the refer-ence pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- $V_{SS} = GND = 0\ V$
- $0°C \leq T_A \leq +70°C$

DC Electrical Characteristics

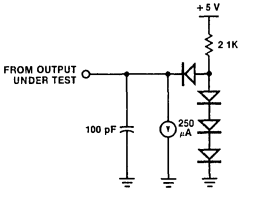
Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.2	7.0	V	
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250 μA (except $\overline{BUSY}$)
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +3.5 mA for D_0-D_7
			0.4	V	I_{OL} = +5 mA for $\overline{BUSY}$
I_{IL}	Input Leakage		±10	μA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10	μA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		30	mA	Standby, AC = Static
			45	mA	Fast Cycle Operation, TC = 400 ns

Capacitance

Capacitance of input or output pins, except $\overline{BUSY}$: 5 pF (max)

Capacitance of $\overline{BUSY}$ input/output: 10 pF (max)

All ac parameters assume a load capacitance of 100 pF maximum.



Interfacing the Z6132 to a Z8000

The Z8001 or Z8002 CPU addresses memory as bytes, but can access either 8-bit bytes or 16-bit words. When writing a byte, A_0 selects the byte within a word; in all other cases the Z8000 always accesses a word, and A_0 is ignored. (When reading a byte, the memory reads a word and the CPU selects the appropriate byte internally.)

The odd- and even-byte memory banks use separate Chip Select decoders. The LS157 multiplexer is used as a function generator and activates either the odd bank, the even bank or both, as determined by A_0, Read/Write (R/$\overline{W}$) and Byte/Word (B/$\overline{W}$).

Address labels A_0-A_{11} and Data labels D_0-D_7 are used only to illustrate this example. Obviously, all address pins as well as all Data pins can be arranged arbitrarily to accommodate the PC board layout. A_0 must, however, always be connected to AD_1 to enhance the self-refresh operation.

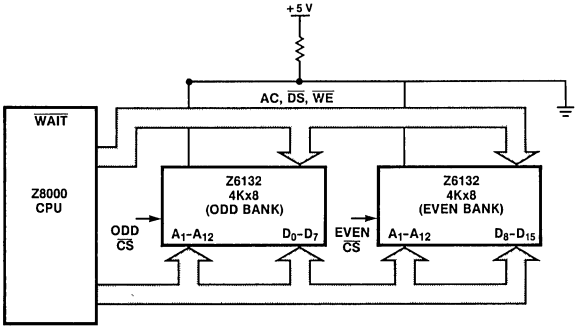


Figure 5. Block Diagram

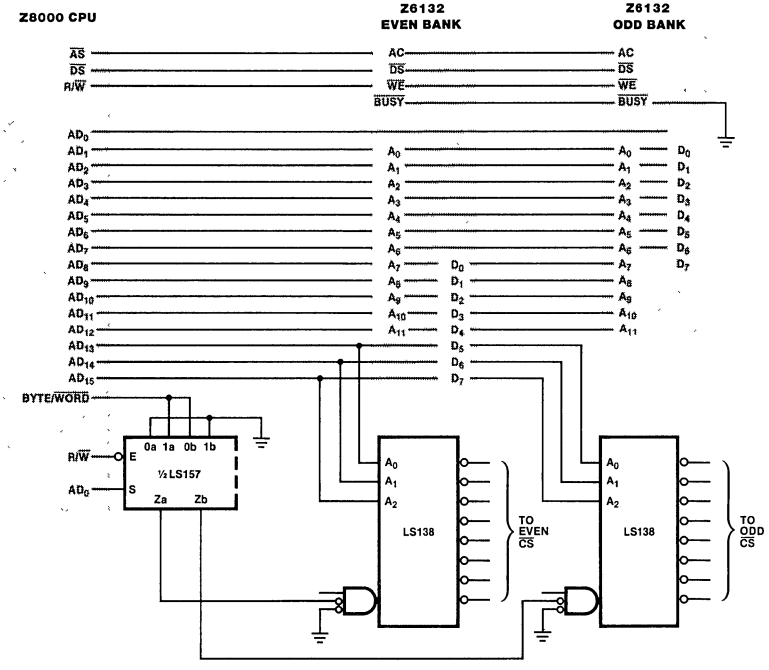


Figure 6. Connection Diagram

Interfacing the Z6132 to a Z-80

When interfacing the Z80 CPU to the Z6132, the complexity of the required logic depends somewhat on the speed at which the CPU is operating and the speed selection of the Z6132. For the interface example shown, the assumptions are that the Z80 CPU is operating at 4 MHz and the Z6132 has a 300 ns access time (i.e., a Z6132-5). Note that the Z6132 is used in the Short Cycle Time Refresh mode.

The $\overline{MREQ}$ and $\overline{M1}$ lines from the Z80 CPU are used to generate the Address Clock (AC) to the Z6132. The $\overline{M1}$ line is latched with the Z80 CPU Clock (CLK) so that the memory transaction is started as early as possible during opcode fetch cycles. This is done to provide the required access time for the Z6132 during Z80 opcode fetch cycles, which are one-half clock cycle shorter than data memory cycles.

The D-type flip-flop is used to "remove" the short cycle from the $\overline{MREQ}$ signal during Z80 CPU-initiated memory refresh cycles. If this were not done, an AC pulse would be generated that would not meet the AC width specification of the Z6132-5.

Memory select can be accomplished with a single address line (A12). If more than one Z6132 is used in a system, an address decoder (such as the 74LS138 shown) can be used to drive the $\overline{CS}$ line on the Z6132. The $\overline{CS}$ line is sampled with the Low-to-High transition of AC, so there will not be a problem with erroneous chip selects in the absence of a valid AC strobe.

During the write cycles, the $\overline{WE}$ line is sampled by the $\overline{DS}$ line. During the read cycles, the Low-to-High transition of AC is the critical sample time for the $\overline{WE}$ line, and $\overline{DS}$ becomes an output enable control. Therefore, $\overline{RD}$ and $\overline{WR}$ from the Z80 are OR'ed to generate $\overline{DS}$ to the Z6132, and $\overline{RD}$ and $\overline{M1}$ are OR'ed to generate $\overline{WE}$.

Normally, $\overline{WE}$ can be driven through an inverter by $\overline{RD}$ only. Due to the shortened Z80 cycle, however, AC is generated as early as possible during opcode fetches. Consequently, $\overline{WE}$ must be generated early by OR'ing $\overline{RD}$ and $\overline{M1}$.

The logic shown allows the Z6132 to be used for program or data memory and to work with 4 MHz clocks. Since all internal timing, including refresh, is derived from the rising edge of AC, it is necessary that AC be supplied at all times. The CPU does not have to access the Z6132 in order to do this because any $\overline{MREQ}$ signal will activate AC. In the unlikely event that only even or only odd addresses are presented to the Z6132 for more than 17 AC strobes, the Z6132 will do an internal refresh on the next AC. If the CPU tries to access the Z6132 during this time, the CPU is placed in the Wait mode. The BUSY line on the Z6132 is activated until the internal refresh operation is completed.

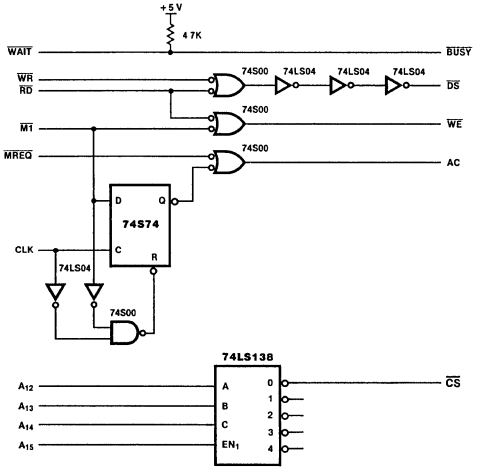


Figure 7. Z-80 Connection Diagram

Interfacing the Z6132 to a Z8

The Z6132 interfaces directly with the single chip Z8 microcomputer. Port 1 provides the 8-bit multiplexed Address/Data bus, and the more significant address bits are provided by Port 0.

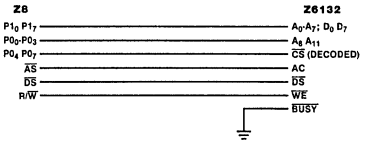


Figure 8. Z8 Connection Diagram

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z6132-3	CS,PS,DS	200 ns	Z6132-3 4096x8-Bit Quasi-Static RAM	Z6132-5	CS,PS,DS	300 ns	Z6132-5 4096x8-Bit Quasi-Static RAM
Z6132-4	CS,PS,DS	250 ns	Z6132-4 4096x8-Bit Quasi-Static RAM	Z6132-6	CS,PS,DS	350 ns	Z6132-6 4096x8-Bit Quasi-Static RAM

NOTES C = Ceramic, D = Cerdip, P = Plastic; S = 0°C to +70°C.

Additional Information

Zilog

Z-BUS™ Component Interconnect

Zilog

Summary

March 1981

Features

- Multiplexed address/data bus shared by memory and I/O transfers.
- 16 or more memory address bits; 16-bit I/O addresses; 8 or 16 data bits.
- Supports polling and vectored or non-vectored interrupts.
- Daisy-chain interrupt structure services interrupts without a separate priority controller.
- Direct addressing of registers within a peripheral facilitates I/O programming.
- Bus signals allow asynchronous CPU and peripheral clocks.
- Daisy-chain bus-request structure supports distributed control of the bus.
- Shared resources can be managed by a general-purpose, distributed resource-request mechanism.

General Description

The Z-BUS is a high-speed parallel shared bus that links components of the Z8000 Family. It provides family members with a common communication interface that supports the following kinds of interactions:

- *Data Transfer.* Data can be moved between bus controllers (such as a CPU) and memories or peripherals.
- *Interrupts.* Interrupts can be generated by peripherals and serviced by CPUs over the bus.
- *Resource Control.* Distributed management of shared resources (including the bus itself) is supported by a daisy-chain priority mechanism.

The heart of the Z-BUS is a set of multiplexed address/data lines and the signals that control these lines. Multiplexing data and addresses onto the same lines makes more efficient use of pins and facilitates expansion of the number of data and address bits. Multiplexing also allows straightforward addressing of a peripheral's internal registers, which greatly simplifies I/O programming.

A daisy-chained priority mechanism resolves interrupt and resource requests, thus allowing distributed control of the bus and eliminating the need for separate priority controllers. The resource-control daisy chain allows wide physical separation of components.

The Z-BUS is asynchronous in the sense that peripherals do not need to be synchronized with the CPU clock. All timing information is provided by Z-BUS signals.

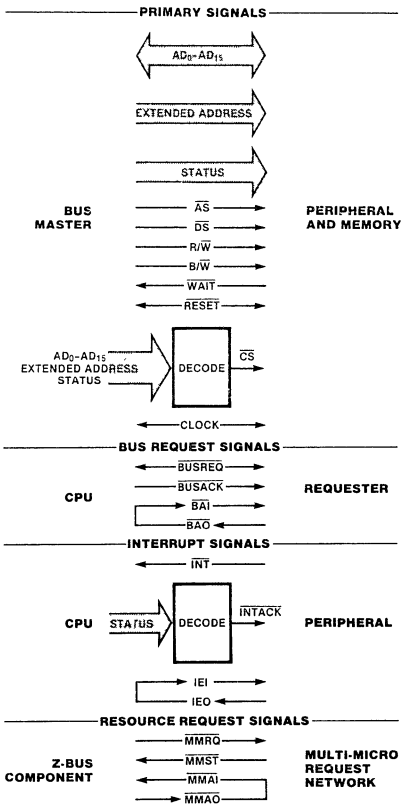


Figure 1. Z-BUS Signals

Z-BUS Components

A Z-BUS component is one that uses Z-BUS signals and protocols, and meets the specified ac and dc characteristics. Most components in the Z8000 Family are Z-BUS components. The four categories of Z-BUS components are as follows:

CPUs. A Z-BUS system contains one CPU, and this CPU has default control of the bus and typically initiates most bus transactions. Besides generating bus transactions, it handles interrupt and bus-control requests. The Z8001 Segmented CPU and Z8002 Non-Segmented CPU are Z-BUS CPUs.

Peripherals. A Z-BUS peripheral is a component capable of responding to I/O transactions and generating interrupt requests. The *Z8036 Counter Input/Output Circuit (Z-CIO), Z8038 FIFO Input/Output, Interface Unit (Z-FIO),* the *Z8030 Serial Communication Controller (Z-SCC),* the *Z8090 Universal Peripheral Controller (Z-UPC),* and the *Z8052 CRT Controller (Z-CRT)* are all Z-BUS peripherals.

Requesters. A Z-BUS requester is any component capable of requesting control of the bus and initiating transactions on the bus. A Z-BUS requester is usually also a peripheral. The *Z8016 DMA Transfer Controller (Z-DTC)* is a Z-BUS requester and a peripheral.

Memories. A Z-BUS memory is one that interfaces directly to the Z-BUS and is capable of fetching and storing data in response to Z-BUS memory transactions. The *Z6132 Quasi-Static RAM* is a Z-BUS memory.

Other Components

The *Z8 Microcomputer*—in its microprocessor configuration—conforms to Z-BUS timing (which allows it to use Z-BUS peripherals and memories), but is missing a wait input and certain status outputs.

The *Z8010 Memory Management Unit (Z-MMU)* is a Z8000 CPU support component that interfaces with part of the Z-BUS on the CPU side and provides demultiplexed addresses on the memory side.

The *Z8060 First-In-First-Out Buffer (Z-FIFO)* is not a Z-BUS component; rather, it is used to expand the buffer depth of the Z-FIO or to interface the I/O ports of the Z-UPC, Z-CIO, or Z-FIO to user equipment.

Z-80 Family components, while not Z-BUS compatible, are easily interfaced to Z-BUS CPUs.

Operation

Two kinds of operations can occur on the Z-BUS: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the Z-BUS and is known as the *bus master.* A transaction is initiated by a bus master and is responded to by some other device on the bus. Four kinds of transactions occur in Z-BUS systems:

- ***Memory.*** Transfers 8 or 16 bits of data to or from a memory location.
- ***I/O.*** Transfers 8 or 16 bits of data to or from a peripheral.
- ***Interrupt Acknowledge.*** Acknowledges an interrupt and transfers an identification/status vector from the interrupting peripheral.
- ***Null.*** Does not transfer data. Typically used for refreshing memory.

Only one transaction can proceed on the bus at a time, and it must be initiated by the bus master. A request, however, may be initiated by a component that does not have control of the bus. There are three kinds of requests:

- ***Interrupt.*** Requests the attention of the Z-BUS CPU.
- ***Bus.*** Requests control of the Z-BUS to initiate transactions.
- ***Resource.*** Requests control of a particular resource.

When a request is made, it is answered according to its type: for interrupt requests an interrupt-acknowledge transaction is initiated; for bus and resource requests an acknowledge signal is sent. In all cases a daisy-chain priority mechanism provides arbitration between simultaneous requests.

Signal Lines

The Z-BUS consists of a set of common signal lines that interconnect bus components (Figure 1). The signals on these lines can be grouped into four catagories, depending on how they are used in transactions and requests.

Primary Signals. These signals provide timing, control, and data transfer for Z-BUS transactions.

AD_0-AD_{15}. Address/Data (active High). These multiplexed data and address lines carry I/O addresses, memory addresses, and data during Z-BUS transactions. A Z-BUS may have 8 or 16 bits of data depending on the type of CPU. In the case of an 8-bit Z-BUS, data is transferred on AD_0-AD_7.

Extended Address. (active High). These lines extend AD_0-AD_{15} to support memory addresses greater than 16 bits. The number of lines and the type of address information carried is dependent on the CPU.

Status. (active High). These lines designate the kind of transaction occurring on the bus and certain additional information about the transaction (such as program or data memory access or System versus Normal Mode).

$\overline{AS}$. Address Strobe (active Low). The rising edge of $\overline{AS}$ indicates the beginning of a transaction and that the Address, Status, R/$\overline{W}$, and B/$\overline{W}$ signals are valid.

$\overline{DS}$. Data Strobe (active Low). $\overline{DS}$ provides timing for data movement to or from the bus master.

R/$\overline{W}$. Read/Write (Low = write). This signal determines the direction of data transfer for memory or I/O transactions.

B/$\overline{W}$. Byte/Word (Low = word). This signal indicates whether a byte or word of data is to be transmitted on a 16-bit bus. This signal is not present on an 8-bit bus.

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

$\overline{RESET}$. (active Low). A Low on this line resets the CPU and bus users. Peripherals may be reset by $\overline{RESET}$ or by holding $\overline{AS}$ and $\overline{DS}$ Low simultaneously.

$\overline{CS}$. Chip Select (active Low). Each peripheral or memory component has a $\overline{CS}$ line that is decoded from the address and status lines. A Low on this line indicates that the peripheral or memory component is being addressed by a transaction. The Chip Select information is latched on the rising edge of $\overline{AS}$.

CLOCK. This signal provides basic timing for bus transactions. Bus masters must provide all signals synchronouly to the clock. Peripherals and memories do not need to be synchronized to the clock.

Bus Request Signals. These signals make bus requests and establish which component should obtain control of the bus.

$\overline{BUSREQ}$. Bus Request (active Low). This line is driven by all bus requesters. A Low indicates that a bus requester has or is trying to obtain control of the bus.

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

$\overline{BAI}$, $\overline{BAO}$. Bus Acknowledge In, Bus Acknowledge Out (active Low). These signals form the bus-request daisy chain.

Z-BUS Connections

Signal	CPU	Requester	Peripheral	Memory
AD_0-AD_{15}	Bidirectional[2] 3-state	Bidirectional[2] 3-state	Bidirectional[1] 3-state	Bidirectional[2] 3-state
Extended Address[8]	Output 3-state	Output 3-state	□	Input
Status	Output 3-state	Output 3-state	Input[10]	□
R/$\overline{W}$	Output 3-state	Output 3-state	Input	Input
B/$\overline{W}$[9]	Output	Output	Input[3]	Input
$\overline{WAIT}$	Input	Input	Output[8] Open Drain	Output[8] Open Drain
$\overline{AS}$	Output 3-state	Output 3-state	Input	Input
$\overline{DS}$	Output 3-state	Output 3-state	Input	Input
$\overline{CS}$[4]	□	□	Input	Input
$\overline{RESET}$	Input	Input[13]	Input[5]	□
CLOCK[14]	Input	Input	Input[8]	Input[8]
$\overline{BUSREQ}$	Input	Bidirectional Open Drain	□	□
$\overline{BUSACK}$	Output	□	□	□
$\overline{BAI}$[7]	□	Input	□	□
$\overline{BAO}$[7]	□	Output	□	□
$\overline{INT}$	Input	□	Output Open Drain	□
$\overline{INTACK}$[6]	□	□	Input[11]	□
IEI[7]	□	□	Input	□
IEO[7]	□	□	Output	□
$\overline{MMRQ}$[12]	Output Open Drain			
$\overline{MMST}$[12]	Input			
$\overline{MMAI}$[7, 12]	Input			
$\overline{MMAO}$[7, 12]	Output			

1. Only AD_0-AD_7, unless peripheral is 16-Bit
2. For an 8-bit bus, only AD_0-AD_7 are bidirectional.
3. Only for a 16-bit peripheral.
4. Derived signal, one for each peripheral or memory, decoded from status and address lines.
5. Optional—peripherals are typically reset by $\overline{AS}$ and $\overline{DS}$ being Low simultaneously, however, they can have a reset input
6. Derived signal, decoded from status lines.
7. Daisy-chain lines
8. Optional signal(s).
9. For 16-bit data bus only
10. Optional—usually only input on peripherals that are also requesters
11. May be omitted if peripheral inputs status lines.
12. Optional signal; any component may attach to the resource request lines.
13. Optional signal; a bus requestor may also be reset by $\overline{AS}$ and $\overline{DS}$ going Low and $\overline{BAI}$ being High simultaneously.
14. This signal is optional if there are no requesters on the bus. CPU timing can be provided by alternate means such as crystal oscillator inputs

Table 1. Z-BUS Component Connections to Signal Lines. *This table shows how the various Z-BUS components attach to each signal line. When a device is both a bus requester and a peripheral, the attributes in both columns of the table should be combined (e.g., input combined with output and 3-state becomes bidirectional and 3-state.)*

□ No Connection

Signal Lines (Continued)

Interrupt Signals. These signals are used for interrupt requests and for determining which interrupting component is to respond to an acknowledge. To support more than one type of interrupt, the lines carrying these signals can be replicated. (The Z8000 CPU supports three types of interrupts: non-maskable, vectored, and non-vectored.)

$\overline{INT}$. Interrupt (active Low). This signal can be driven by any peripheral capable of generating an interrupt. A Low on $\overline{INT}$ indicates that an interrupt request is being made.

$\overline{INTACK}$. Interrupt Acknowledge (active Low). This signal is decoded from the status lines. A Low indicates an interrupt acknowledge transaction is in progress. This signal is latched by the peripheral on the rising edge of $\overline{AS}$.

IEI, IEO. Interrupt Enable In, Interrupt Enable Out (active High). These signals form the interrupt daisy chain.

Resource Request Signals. These signals are used for resource requests. To manage more than one resource, the lines carrying these signals can be replicated. (The Z8000 supports one set of resource request lines.)

$\overline{MMRQ}$. Multi-Micro Request (active Low). This line is driven by any device that can use the shared resource. A Low indicates that a request for the resource has been made or granted.

$\overline{MMST}$. Multi-Micro Status (active Low). This pin allows a device to observe the value of the $\overline{MMRQ}$ line. An input pin other than $\overline{MMRQ}$ facilitates the use of line drivers for $\overline{MMRQ}$.

$\overline{MMAI}$, $\overline{MMAO}$. Multi-Micro Acknowledge In, Multi-Micro Acknowledge Out (active Low). These lines form the resource-request daisy chain.

Transactions

All transactions start with Address Strobe being driven Low and then raised High by the bus master (Figure 2). The Status lines are valid on the rising edge of Address Strobe and indicate the type of transactions being initiated. If the transaction requires an address, it must also be valid on the rising edge of Address Strobe.

For all transactions except null transactions (which do nothing beyond this point), data is then transferred to or from the bus master. The bus master uses Data Strobe to time the movement of data. For a read (R/$\overline{W}$ = High), the bus master makes AD_0-AD_{15} inactive before driving Data Strobe Low so that the addressed memory or peripheral can put its data on the bus. The bus master samples this data just before raising Data Strobe High. For a write (R/$\overline{W}$ = Low), the bus master puts the data to be written on AD_0-AD_{15} before forcing Data Strobe Low.

For an 8-bit Z-BUS, data is transferred on AD_0-AD_7. Address bits may remain on AD_8-AD_{15} while $\overline{DS}$ is Low.

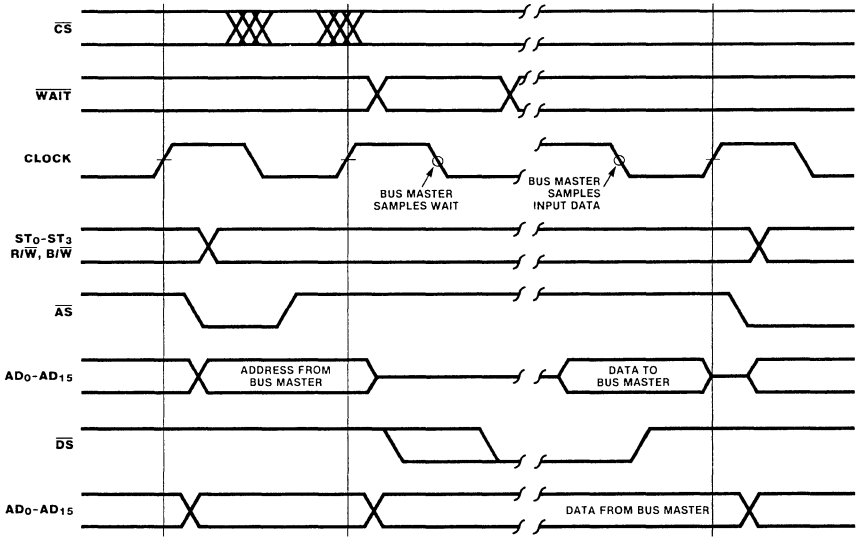


Figure 2. Typical Transaction Timing

Memory Transactions

For a memory transaction, the Status lines distinguish among various address spaces, such as program and data or system and normal, as well as indicating the type of transaction. The memory address is put on AD_0–AD_{15} and on the extended address lines.

For a Z-BUS with 16-bit data, the memory is organized as two banks of eight bits each (Figure 3). One bank contains all the upper bytes of all the addressable 16-bit words. The other bank contains all the lower bytes. When a single byte is written (R/$\overline{W}$ = Low, B/$\overline{W}$ = High), only the bank indicated by address bit A_0 is enabled for writing.

For a Z-BUS with 8-bit data, the memory is organized as one bank which contains all bytes. This bank always inputs and outputs its data on AD_0–AD_7.

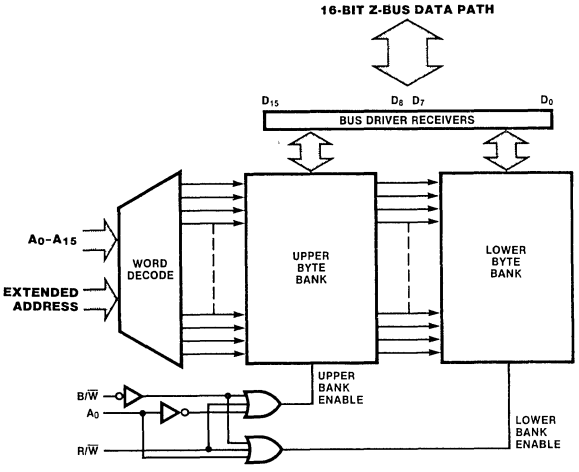


Figure 3. Byte/Word Memory Organization

I/O Transactions

I/O transactions are similar to memory transactions with two important differences. The first is that I/O transactions take an extra clock cycle to allow for slow peripheral operation. The second is that byte data (indicated by B/$\overline{W}$ High on a 16-bit bus) is always transmitted on AD_0–AD_7, regardless of the I/O address. (AD_8–AD_{15} contain arbitrary data in this case.) For an I/O transaction, the address indicates a peripheral and a particular register or function within that peripheral.

Null Transactions

The two kinds of null transactions are distinguished by the Status lines: internal operation and memory refresh. Both transactions look like a memory read transaction except that Data Strobe remains High and no data is transferred.

For an internal operation transaction, the Address lines contain arbitrary data when Address Strobe goes High. This transaction is initiated to maintain a minimum transaction rate when a bus master is doing a long internal operation (to support memories which generate refresh cycles from Address Strobe).

For a memory refresh transaction, the Address lines contain a refresh address when Address Strobe goes High. This transaction is used to refresh a row of a dynamic memory.

Any memory or I/O transaction can be suppressed (effectively turning it into a null transaction) by keeping Data Strobe High throughout the transaction.

Interrupts

A complete interrupt cycle consists of an interrupt request followed by an interrupt-acknowledge transaction. The request, which consists of $\overline{INT}$ pulled Low by a peripheral, notifies the CPU that an interrupt is pending. The interrupt-acknowledge transaction, which is initiated by the CPU as a result of the request, performs two functions: it selects the peripheral whose interrupt is to be acknowledged, and it obtains a vector that identifies the selected device and cause of interrupt.

A peripheral can have one or more sources of interrupt. Each interrupt source has three bits that control how it generates interrupts. These bits are an Interrupt Pending bit (IP), and Interrupt Enable bit (IE), and an Interrupt Under Service bit (IUS).

A peripheral may also have one or more vectors for identifying the source of an interrupt during an interrupt-acknowledge transaction. Each interrupt source is associated with one interrupt vector and each interrupt vector can have one or more interrupt sources associated with it. Each vector has a Vector Includes Status bit (VIS) controlling its use.

Finally, each peripheral has three bits for

Interrupts (Continued)

controlling interrupt behavior for the whole device. These are a Master Interrupt Enable bit (MIE), a Disable Lower Chain bit (DLC), and a No Vector bit (NV).

Peripherals are connected together via an interrupt daisy chain formed with their IEI and IEO pins (Figure 4). The interrupt sources within a device are similarly connected into this chain with the overall effect being a daisy chain connecting the interrupt sources. The daisy chain has two functions: during an interrupt-acknowledge transaction, it determines which interrupt source is being acknowledged; at all other times it determines which interrupt sources can initiate an interrupt request.

Figure 5 is a state diagram for interrupt processing for an interrupt source (assuming its IE bit is 1). An interrupt source with an interrupt pending (IP = 1) makes an interrupt request (by pulling $\overline{INT}$ Low) if, and only if, it is enabled (IE = 1, MIE= 1), it does not have an interrupt under service (IUS = 0), no higher priority interrupt is being serviced (IEI = High), and no interrupt-acknowledge transaction is in progress (as indicated by $\overline{INTACK}$ at the last rising edge of $\overline{AS}$). IEO is not pulled down by the interrupt source at this time; IEO continues to follow IEI until an interrupt-acknowledge transaction occurs.

Some time after $\overline{INT}$ has been pulled Low, the CPU initiates an interrupt-acknowledge transaction (indicated by $\overline{INTACK}$ Low). Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the IEI/IEO daisy chain settles. Any interrupt source with an interrupt pending (IP = 1, IE = 1, MIE = 1) or under service (IUS = 1) holds its IEO line Low; all other interrupt sources make IEO follow IEI. When $\overline{DS}$ falls, only the highest priority interrupt source with a pending interrupt (IP = 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1, and, if the peripheral's NV bit is 0, identifies itself by placing the vector on AD_0–AD_7. If the NV bit is 1, then the peripheral's $AD_0 - AD_7$ pins remain floating, thus allowing external circuitry to supply the vector. (All interrupts, including the Z8000's non-vectored interrupt, need a vector for identifying the source of an interrupt.) If the vector's VIS bit is 1, the vector will also contain status information further identifying the source of the interrupt. If the VIS bit is 0, the vector held in the peripheral will be output without modification.

While an interrupt source has an interrupt under service (IUS = 1), it prevents all lower priority interrupt sources from requesting interrupts by forcing IEO Low. When interrupt servicing is complete, the CPU must reset the IUS bit and, in most cases, the IP bit (by means of an I/O transaction).

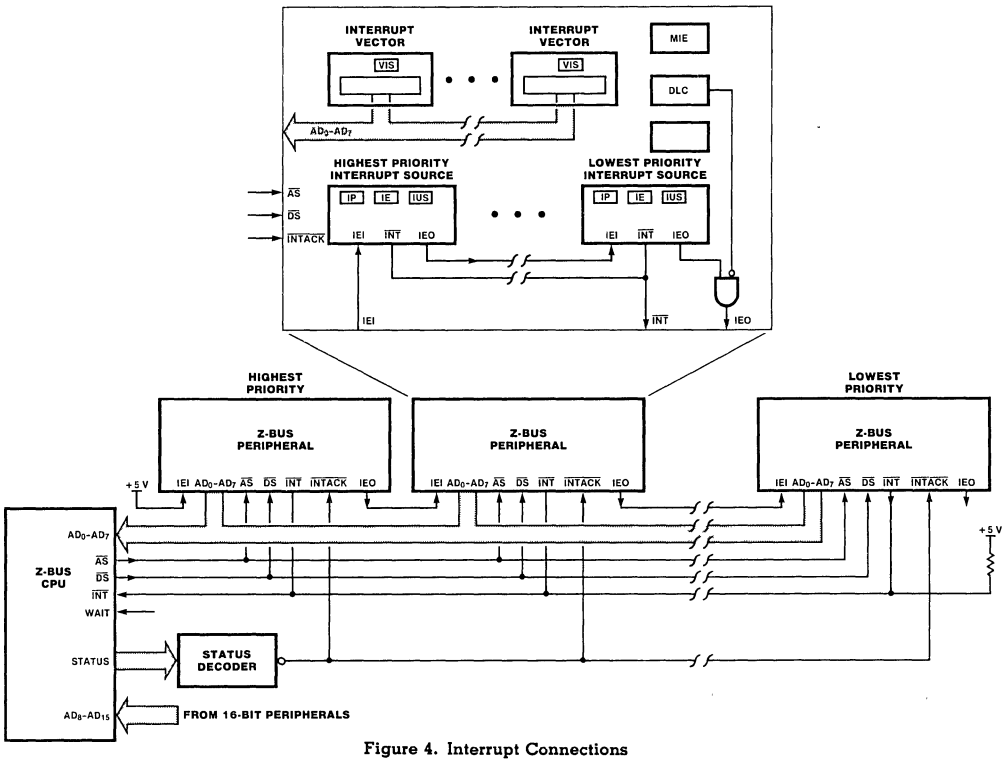


Figure 4. Interrupt Connections

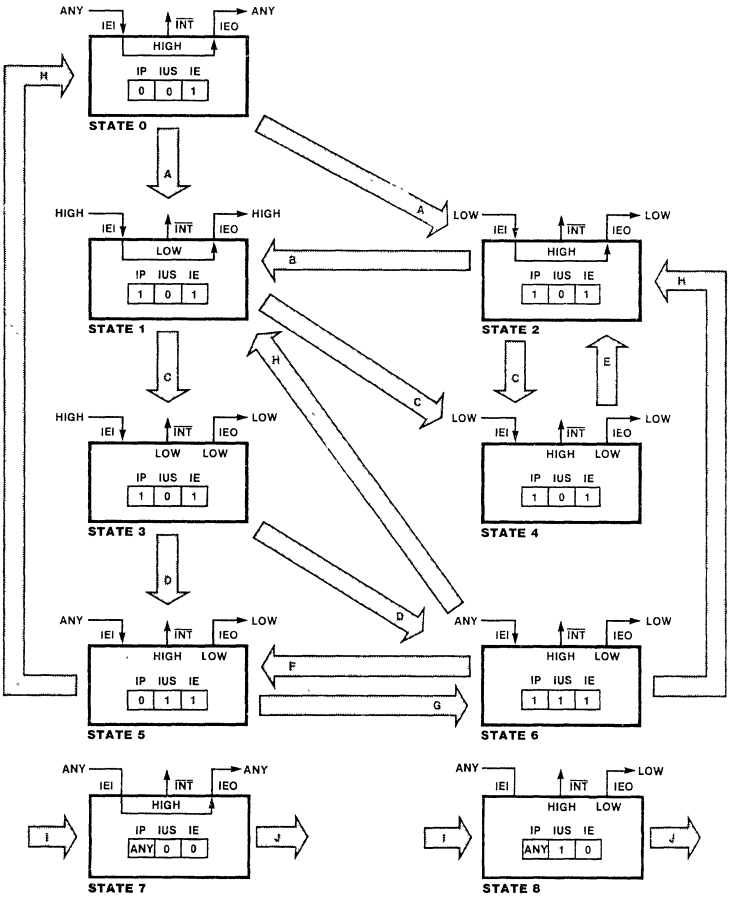


Figure 5. State Diagram for an Interrupt Source

Transition Legend

- A: The peripheral detects an interrupt condition and sets Interrupt Pending.
- B: All higher priority peripherals finish interrupt service, thus allowing IEI to go High.
- C: An interrupt-acknowledge transaction starts, and the IEI/IEO daisy chain settles.
- D: The interrupt-acknowledge transaction terminates with the peripheral selected. Interrupt Under Service (IUS) is set to 1, and Interrupt Pending (IP) may or may not be reset.
- E: The interrupt-acknowledge transaction terminates with a higher priority device having been selected.
- F: The Interrupt Pending bit in the peripheral is reset by an I/O operation.
- G: A new interrupt condition is detected by the peripheral, causing IP to be set again.
- H: Interrupt service is terminated for the peripheral by resetting IUS.
- I[3]: IE is reset to zero, causing interrupts to be disabled.
- J[4]: IE is set to one, re-enabling interrupts

State Legend

- 0: No interrupts are pending or under service for this peripheral.
- 1: An interrupt is pending, and an interrupt request has been made by pulling $\overline{INT}$ Low.
- 2: An interrupt is pending, but no interrupt request has been made because a higher priority peripheral has an interrupt under service, and this has forced IEI Low.
- 3: An interrupt-acknowledge sequence is in progress, and no higher priority peripheral has a pending interrupt.
- 4: An interrupt-acknowledge sequence is in progress, but a higher priority peripheral has a pending interrupt, forcing IEI Low.
- 5: The peripheral has an interrupt under service. Service may be temporarily suspended (indicated by IEI going Low) if a higher priority device generates an interrupt.
- 6: This is the same as State 5 except that an interrupt is also pending in the peripheral.
- 7: Interrupts are disabled from this source because IE = 0.
- 8: Interrupts are disabled from this source and lower priority sources because IE = 0 and IUS = 1.

1. This diagram assumes MIE = 1. The effect of MIE = 0 is the same as that of setting IE = 0.
2. The DLC bit does not affect the states of individual interrupt sources. Its only effect is on the IEO output of a whole peripheral.
3. Transition I to state 6 or 7 can occur from any state except 3 or 4 (which only occur during interrupt acknowledge).
4. Transition J from state 6 or 7 can be to any state except 3 or 4, depending on the value of IEI, IP, and IUS.

Interrupts (Continued)

A peripheral's Master Interrupt Enable bit (MIE) and Disable Lower Chain bit (DLC) can modify the behavior of the peripheral's interrupt sources in the following way: if the MIE bit is 0, the effect is as if every Interrupt Enable bit (IE) in the peripheral were 0; thus all interrupts from the peripheral are disabled. If the DLC bit is 1, the effect is to force the peripheral's IEO output Low, thus disabling all lower priority devices from initiating interrupt requests.

Polling can be done by disabling interrupts (using MIE and DLC) and by reading peripherals to detect pending interrupts. Each Z-BUS peripheral has a single directly addressable register that can be read to determine if there is an interrupt pending in the device and, if so, what interrupt source it is from.

Bus Requests

To generate transactions on the bus, a bus requester must gain control of the bus by making a bus request. This is done by forcing $\overline{\text{BUSREQ}}$ Low (Figure 6). A bus request can be made only if $\overline{\text{BUSREQ}}$ is initially High (and has been for two clock cycles), indicating that the bus is controlled by the CPU and no other device is requesting it.

After $\overline{\text{BUSREQ}}$ is pulled Low, the Z-BUS CPU relinquishes the bus and indicates this condition by making $\overline{\text{BUSACK}}$ Low. The Low on $\overline{\text{BUSACK}}$ is propagated through the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain (Figure 6). $\overline{\text{BAI}}$ follows $\overline{\text{BAO}}$ for components not requesting the bus, and any component requesting the bus holds its $\overline{\text{BAO}}$ High, thereby locking out all lower priority users.

A bus requester gains control of the bus when its $\overline{\text{BAI}}$ input goes Low. When it is ready to relinquish the bus, it stops pulling $\overline{\text{BUSREQ}}$ Low and allows $\overline{\text{BAO}}$ to follow $\overline{\text{BAI}}$. This permits lower priority devices that made simultaneous requests to gain control of the bus. When all simultaneously requesting devices have relinquished the bus, $\overline{\text{BUSREQ}}$ goes High, returning control of the bus to the CPU and allowing other devices to request it.

The protocol to be followed in making a bus request is shown in Figure 7.

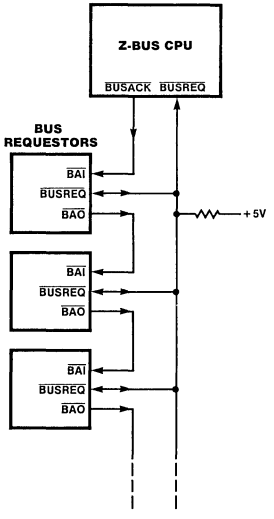


Figure 6. Bus Request Connections

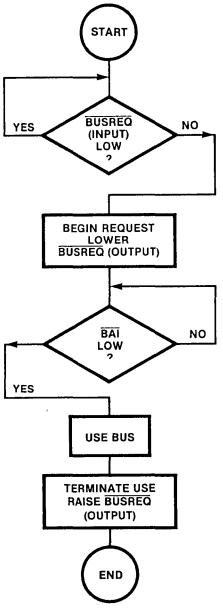


Figure 7. Bus Request Protocol

Resource Requests

Resource requests are used to obtain control of a resource that is shared between several users. The resource can be a common bus, a common memory or any other resource. The requestor can be any component capable of implementing the request protocol.

Unlike the Z-BUS itself, no component has control of a general resource by default; every device must acquire the resource before using it. All devices sharing the general resource drive the $\overline{MMRQ}$ line (Figure 8). When Low, the $\overline{MMRQ}$ line indicates that the resource is being acquired or used by some device. The $\overline{MMST}$ pin allows each device to observe the state of the $\overline{MMRQ}$ line.

When $\overline{MMRQ}$ is High, a device may initiate a resource request by pulling $\overline{MMRQ}$ Low (Figure 9). The resulting Low on $\overline{MMRQ}$ is propagated through the $\overline{MMAI}/\overline{MMAO}$ daisy chain. If a device is not requesting the resource, its $\overline{MMAO}$ output follows its $\overline{MMAI}$ input. Any device making a resource request forces its $\overline{MMAO}$ output High to deny use of the resource to lower priority devices.

A device gains control of the resource if its $\overline{MMAI}$ input is Low (and its $\overline{MMAO}$ output is High) after a sufficient delay to let the daisy chain settle. If the device does not obtain the resource after this short delay, it must stop pulling $\overline{MMRQ}$ Low and make another request at some later time when $\overline{MMRQ}$ is again High. When a device that has gained control of a resource is finished, it releases the resource by allowing $\overline{MMRQ}$ to go High.

The four unidirectional lines of the resource request chain allow the use of line drivers, thus facilitating connection of components separated by some distance. In the case of the Z8000 CPU, the four resource request lines may be mapped into the CPU $\overline{MI}$ and $\overline{MO}$ pins using the logic shown in Figure 10. With this configuration, the Multi-Micro Request Instruction (MREQ) performs a resource request.

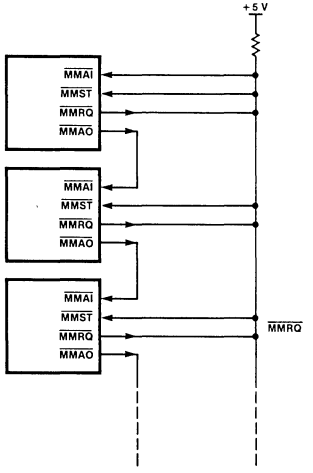


Figure 8. Resource Request Connections

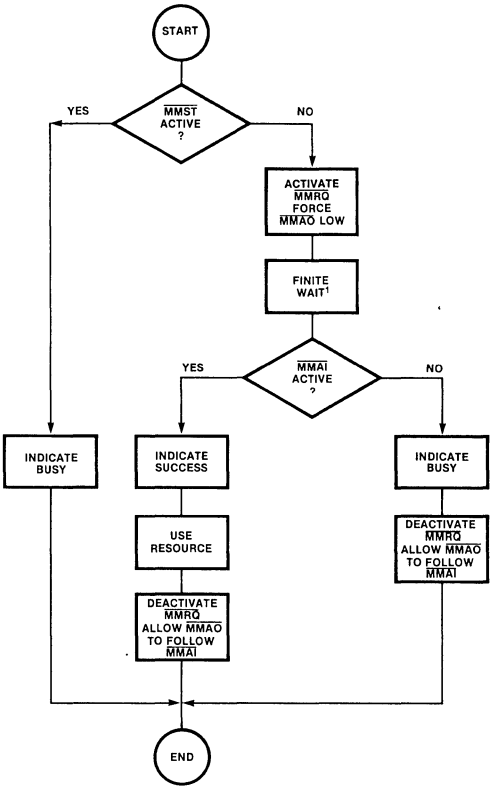


Figure 9. Resource Request Protocol

1 For any resource requested, this wait time must be less than the minimum wait time plus resource usage time of all other requesters

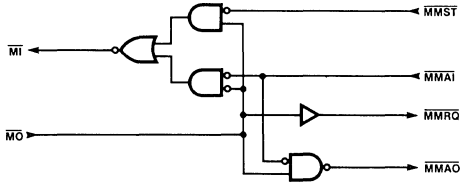
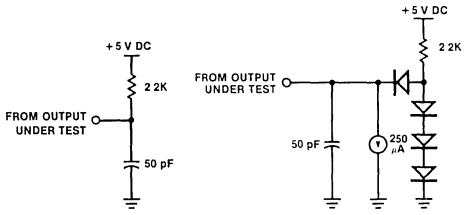


Figure 10. Bus Request Logic for Z8000

Test Conditions

The timing characteristics given in this document reference 2.0 V as High and 0.8 V as Low. The following test load circuit is assumed. The effect of larger capacitive loadings can be calculated by delaying output signal transitions by 10 ns for each additional 50 pF of load up to a maximum 200 pF.



Open-Drain Test Load **Standard Test Load**

DC Characteristics

The following table states the dc characteristics for the input and output pins of Z-BUS components. All voltages are relative to ground.

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
$V_{IHRESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	V_{CC} to 0.3	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250µA
I_{IL}	Input Leakage Current	-10	+10	µA	V_{IN} = 0.4 to 2.4 V
I_{OL}	3-State Output Leakage Current in Float	-10	+10	µA	V_{OUT} = 0.4 to 2.4 V

Capacitance

The following table gives maximum pin capacitance for Z-BUS components. Capacitance is specified at a frequency of 1 MHz over the temperature range of the component. Unused pins are returned to ground.

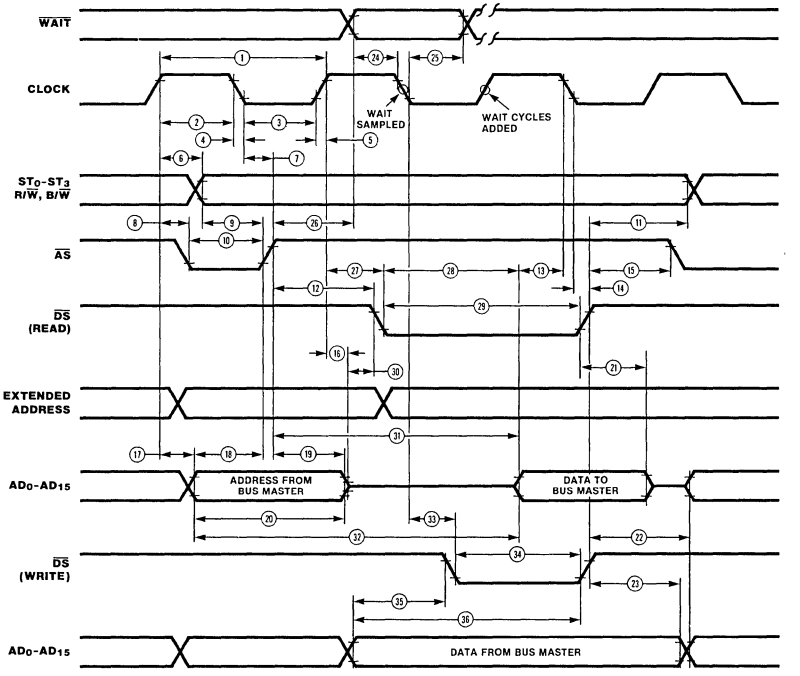
Symbol	Parameter	Max (pF)
C_{IN}	Input Capacitance	10
C_{OUT}	Output Capacitance	15
$C_{I/O}$	Bidirectional Capacitance	15

Timing Diagrams

The following diagrams and tables give the timing for each kind of transaction (except null transactions). Timings are given separately for bus masters and for peripherals and memories and are intended to give the minimum timing requirements which a Z-BUS component must meet. An individual component will have more detailed and sometimes more stringent timing specifications. The differences between bus master timing and peripheral and memory timing allow for buffer and decoding circuit delays and for signal skew. The timing given for memories is a constraint on bus-compatible memories (like the Z6132 Quasi-Static RAM) and is not intended to constrain memory subsystems constructed from conventional components.

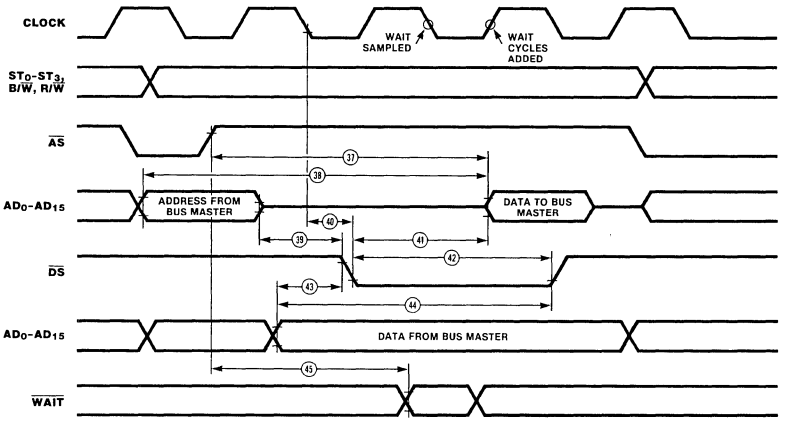
Besides these timings, there is a requirement that at least 128 transactions be initiated in any 2 ms period. This accommodates memories that generate refresh cycles from Address Strobe.

Bus Master Timing

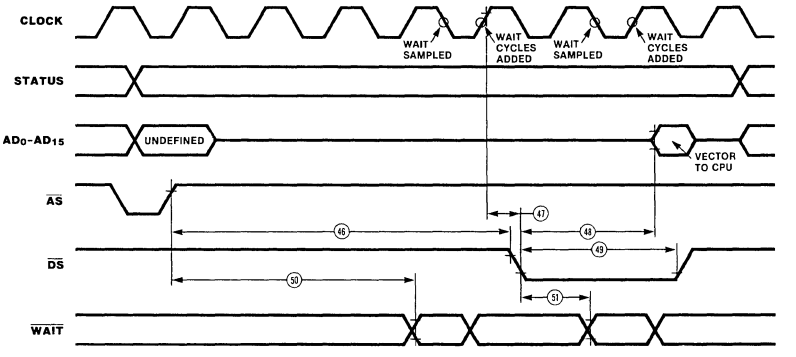


Parameters 1-25 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing

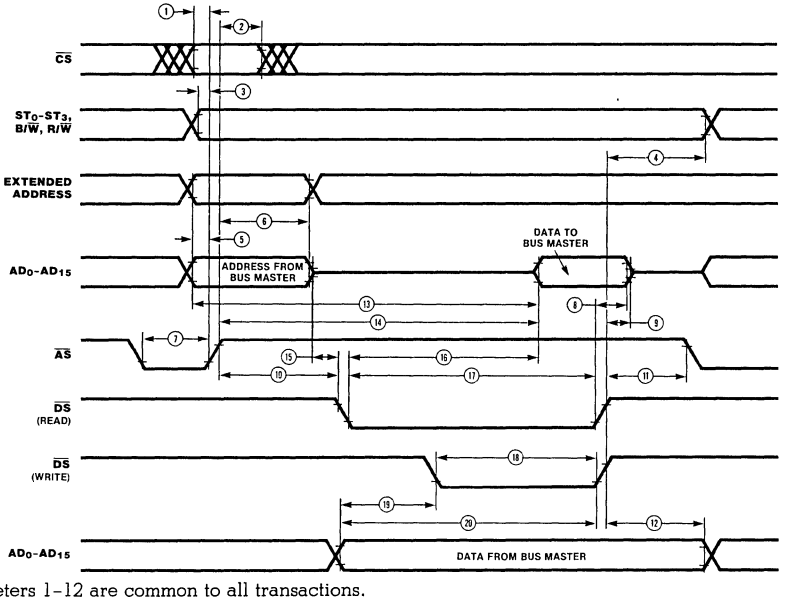


Bus Master Timing Parameters

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes
		All Transactions			
1	TpC	Clock Period	250	2000	
2	TwCh	Clock High Width	105	1895	
3	TwCl	Clock Low Width	105	1895	
4	TfC	Clock Fall Time		20	
5	TrC	Clock Rise Time		20	
6	TdC(S)	Clock ↑ To Status Valid Delay		100	
7	TdC(ASr)	Clock ↓ To $\overline{AS}$ ↑ Delay		90	
8	TdC(ASf)	Clock ↑ To $\overline{AS}$ ↓ Delay		80	
9	TdS(AS)	Status Valid To $\overline{AS}$ ↑ Delay	50		
10	TwAS	$\overline{AS}$ Low Width	80		
11	TdDS(S)	$\overline{DS}$ ↑ To Status Not Valid Delay	80		
12	TdAS(DS)	$\overline{AS}$ ↑ To $\overline{DS}$ ↓ Delay	70	2095	3
13	TsDR(C)	Read Data To Clock ↓ Setup Time	50		
14	TdC(DS)	Clock ↓ To $\overline{DS}$ ↑ Delay		70	
15	TdDS(AS)	$\overline{DS}$ ↑ To $\overline{AS}$ ↓ Delay	70		
16	TdC(Az)	Clock ↑ To Address Float Delay		65	
17	TdC(A)	Clock ↑ To Address Valid Delay		90	
18	TdA(AS)	Address Valid To $\overline{AS}$ ↑ Delay	50		1
19	TdAS(A)	$\overline{AS}$ ↑ To Address Not Valid Delay	60		1
20	TwA	Address Valid Width	150		
21	ThDR(DS)	Read Data To $\overline{DS}$ ↑ Hold Time	0		
22	TdDS(A)	$\overline{DS}$ ↑ To Address Active Delay	80		
23	TdDS(DW)	$\overline{DS}$ ↑ To Write Data Not Valid Delay	80		
24	TsW(C)	$\overline{WAIT}$ To Clock ↓ Setup Time	50		2,5
25	ThW(C)	$\overline{WAIT}$ To Clock ↓ Hold Time	0		2,5
		Memory Transactions			
26	TdAS(W)	$\overline{AS}$ ↑ To $\overline{WAIT}$ Required Valid		90	
27	TdC(DSR)	Clock ↓ To $\overline{DS}$ (Read) ↓ Delay		120	
28	TdDSR(DR)	$\overline{DS}$ (Read) ↓ To Read Data Required Valid		185	
29	TwDSR	$\overline{DS}$ (Read) Low Width		250	
30	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		
31	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Required Valid		320	
32	TdA(DR)	Address Valid To Read Data Required Valid		400	
33	TdC(DSW)	Clock ↓ To $\overline{DS}$ (Write) ↓ Delay		95	
34	TwDSW	$\overline{DS}$ (Write) Low Width	160		
35	TdDW(DSWf)	Write Data Valid To $\overline{DS}$ (Write) ↓ Delay	50		
36	TdDW(DSWr)	Write Data Valid To $\overline{DS}$ (Write) ↑ Delay	230		
		I/O Transactions			
37	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Required Valid		570	
38	TdA(DR)	Address Valid To Read Data Required Valid		650	
39	TdAz(DSI)	Address Float To $\overline{DS}$ (I/O) ↓	0		
40	TdC(DSI)	Clock ↓ To $\overline{DS}$ (I/O) ↓		120	
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ To Read Data Required Valid		320	
42	TwDSI	$\overline{DS}$ (I/O) Low Width	400		
43	TdDW(DSIf)	Write Data To $\overline{DS}$ (I/O) ↓ Delay	50		
44	TdDW(DSIr)	Write Data To $\overline{DS}$ (I/O) ↑ Delay	480		
45	TdAS(W)	$\overline{AS}$ ↑ To $\overline{WAIT}$ Required Valid		340	
		Interrupt-Acknowledge Transactions			
46	TdAS(DSA)	$\overline{AS}$ ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay	960		
47	TdC(DSA)	Clock ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay		120	
48	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ To Read Data Required Valid		420	
49	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	485		
50	TdAS(W)	$\overline{AS}$ ↑ To Wait Required Valid		840	
51	TdDSA(W)	$\overline{DS}$ (Acknowledge) ↓ To Wait Required Valid		130	

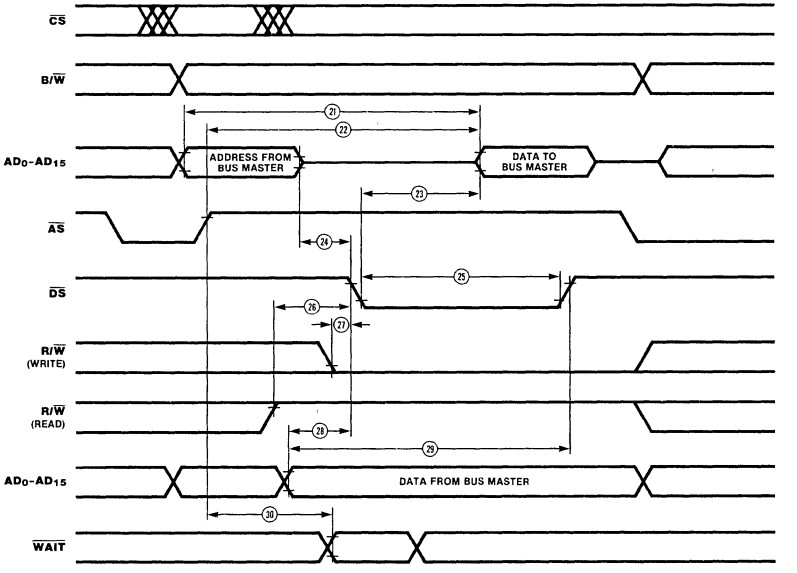
1. Timing for extended addresses is CPU dependent, however, extended addresses must be valid at least as soon as addresses are valid on AD_0-AD_{15} and must remain valid at least as long as addresses are valid on AD_0-AD_{15}.
2. The exact clock cycle that wait is sampled on depends on the type of transaction, however, wait always has the given setup and hold times to the clock.
3. The maximum value for TdAS(DS) does not apply to Interrupt-Acknowledge Transactions.
4. Except where otherwise stated, maximum rise and fall times for inputs are 200 ns.
5. The setup and hold times for $\overline{WAIT}$ to the clock must be met. If $\overline{WAIT}$ is generated asynchronously to the clock, it must be synchronized before input to a bus master

Memory and Peripheral Timing

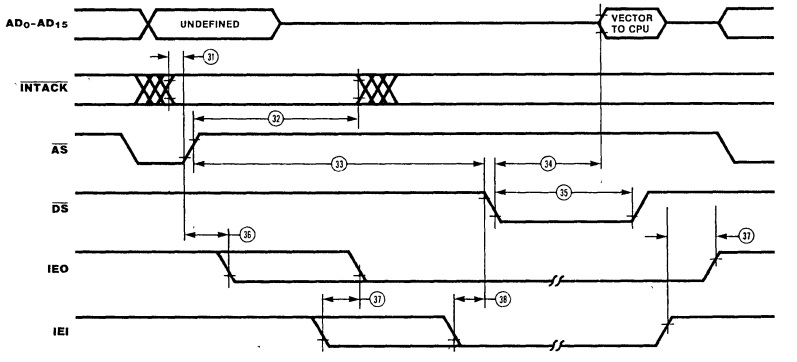


Parameters 1–12 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing



Memory and Peripheral Timing Parameters

Number	Symbol	Parameter	Min (ns)	Max (ns)	Notes
		All Transactions			
1	TsCS(AS)	$\overline{CS}$ To $\overline{AS}$ ↑ Setup Time	0		1
2	ThCS(AS)	$\overline{CS}$ To $\overline{AS}$ ↑ Hold Time	60		1
3	TsS(AS)	Status To $\overline{AS}$ ↑ Setup Time	20		2
4	ThS(DS)	Status To $\overline{DS}$ ↑ Hold Time	60		
5	TsA(AS)	Address To $\overline{AS}$ ↑ Setup Time	10		1
6	ThA(AS)	Address To $\overline{AS}$ ↑ Hold Time	50		1
7	TwAS	$\overline{AS}$ Low Width	70		
8	TdDS(DR)	$\overline{DS}$ ↑ To Read Data Not Valid Delay	0		
9	TdDS(DRz)	$\overline{DS}$ ↑ To Read Data Float Delay		70	
10	TdAS(DS)	$\overline{AS}$ ↑ To $\overline{DS}$ ↓ Delay	60	2095	5
11	TdDS(AS)	$\overline{DS}$↑ To $\overline{AS}$ ↓ Delay	50		
12	ThDW(DS)	Write Data To $\overline{DS}$ ↑ Hold Time	30		1
		Memory Transactions			
13	TdA(DR)	Address Required Valid To Read Data Valid Delay		340	
14	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Valid Delay		230	
15	TdAz(DSR)	Address Float To $\overline{DS}$ (Read) ↓ Delay	0		
16	TdDSR(DR)	$\overline{DS}$ (Read) ↓ To Read Data Valid Delay		95	
17	TwDSR	$\overline{DS}$ (Read) Low Width	240		
18	TwDSW	$\overline{DS}$ (Write) Low Width	150		
19	TsDW(DSWf)	Write Data To $\overline{DS}$ (Write) ↓ Setup Time	30		
20	TsDW(DSWr)	Write Data To $\overline{DS}$ (Write) ↑ Setup Time	210		
		I/O Transactions			
21	TdA(DR)	Address Required Valid To Read Data Valid Delay		590	
22	TdAS(DR)	$\overline{AS}$ ↑ To Read Data Valid Delay		480	
23	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ To Read Data Valid Delay		255	
24	TdAz(DSI)	Address Float To $\overline{DS}$ (I/O) ↓ Delay	0		
25	TwDSI	$\overline{DS}$ (I/O) Low Width	390		
26	TsRWR(DSI)	R/$\overline{W}$ (Read) To $\overline{DS}$ (I/O) ↓ Setup Time	100		
27	TsRWW(DSI)	R/$\overline{W}$ (Write) To $\overline{DS}$ (I/O) ↓ Setup Time	0		
28	TsDW(DSIf)	Write Data To $\overline{DS}$ (I/O) ↓ Setup Time	30		
29	TsDW(DSIr)	Write Data To $\overline{DS}$ (I/O) ↑ Setup Time	460		
30	TdAS(W)	$\overline{AS}$ ↑ To $\overline{WAIT}$ Valid Delay	195		
		Interrupt-Acknowledge Transactions			
31	TsIA(AS)	$\overline{INTACK}$ To $\overline{AS}$ ↑ Setup Time	0		
32	ThIA(AS)	$\overline{INTACK}$ To $\overline{AS}$ ↑ Hold Time	250		
33	TdAS(DSA)	$\overline{AS}$ ↑ To $\overline{DS}$ (Acknowledge) ↓ Delay	940		
34	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ To Read Data Valid Delay	360		
35	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		
36	TdAS(IEO)	$\overline{AS}$ ↓ To IEO ↓ Delay			3, 4
37	TdIEIf(IEO)	IEI To IEO Delay			4
38	TsIEI(DSA)	IEI To $\overline{DS}$ (Acknowledge) ↓ Setup Time			4

1 Parameter does not apply to Interrupt-Acknowledge Transactions

2 Does not cover R/$\overline{W}$ for I/O Transactions

3 Applies only to a peripheral which is pulling $\overline{INT}$ Low at the beginning of the Interrupt-Acknowledge Transaction

4 These parameters are device dependent The parameters for the devices in any particular daisy chain must meet the following constraint. for any two peripherals in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the higher priority peripheral, TsIEI(DSA) for the lower priority peripheral, and TdIEIf(IEO) for each peripheral separating them in the daisy chain

5. The maximum value for TdAS(DS) does not apply to Interrupt-Acknowledge Transactions.

6. Except where stated otherwise, maximum rise and fall times for inputs are 200 ns.

Advanced Architectural Features of the Z8000 CPU

Zilog

Tutorial Information

March 1981

Introduction

The Zilog Z8000 CPU microprocessor is a major advance in microcomputer architecture. It offers many minicomputer and mainframe features for the first time in a microprocessor chip. This tutorial describes the Z8000 CPU with emphasis placed on those features that set it apart from its microprocessor predecessors. For a detailed description of all Z8000 CPU features, consult the Zilog publications listed in the bibliography at the end of this tutorial.

The features to be discussed are grouped into four areas: CPU organization, handling of interrupts and traps, use of memory, and new instructions and data capabilities.

Before discussing these features in more detail, a word about nomenclature is in order. The term Z8000 refers to the concept and architecture of a family of parts. Zilog has adopted the typical conductor industry 4-digit designation for Z8000 Family parts, while also keeping the traditional 3-letter acronym that proved so popular for the Z-80 Family. Thus, the 48-pin version of the Z8000 CPU is called the Z8001 CPU; the 40-pin version is known as the Z8002 CPU.

CPU Organization

The Z8000 CPU is organized around a general-purpose register file (Figure 1). The register file is a group of registers, any one of which can be used as an accumulator, index register, memory pointer, stack pointer, etc. The only exception is Register 0, as explained later.

Flexibility is the major advantage of a general-purpose register organization over an organization that dedicates particular registers to each function. Computation-oriented routines can use general registers as accumulators for intermediate results whereas data manipulation routines can use these registers for memory pointers.

Dedicated registers, however, have a disadvantage: when more registers of a given type are needed than are supplied by the machine, the performance degrades by the extra instructions to swap registers and memory locations. For example, a processor with two index registers suffers when three are needed because a temporary variable in memory (or in another register) must be used for the third index. When the third index is needed, it must be swapped into an index register. In contrast, on a general-register machine three of the registers could be dedicated for index use. In addition, since the need for index registers may vary over the course of a program, a general-register architecture, such as the Z8000, can be adapted to the changing needs of the computation with respect to the number of accumulators, memory pointers and index registers. Thus flexibility results in increased performance and ease of use.

In addition, the registers of the Z8000 are organized to process 8-bit bytes, 16-bit words, 32-bit long words and 64-bit quadruple words. This readily accommodates applications that process data of variable sizes as well as different tasks that require different data sizes.

Although all registers can—in general—be used for any purpose, certain instructions such as Subroutine Call and String Translation make use of specific registers in the general register file, and this must be taken into account when these instructions are used.

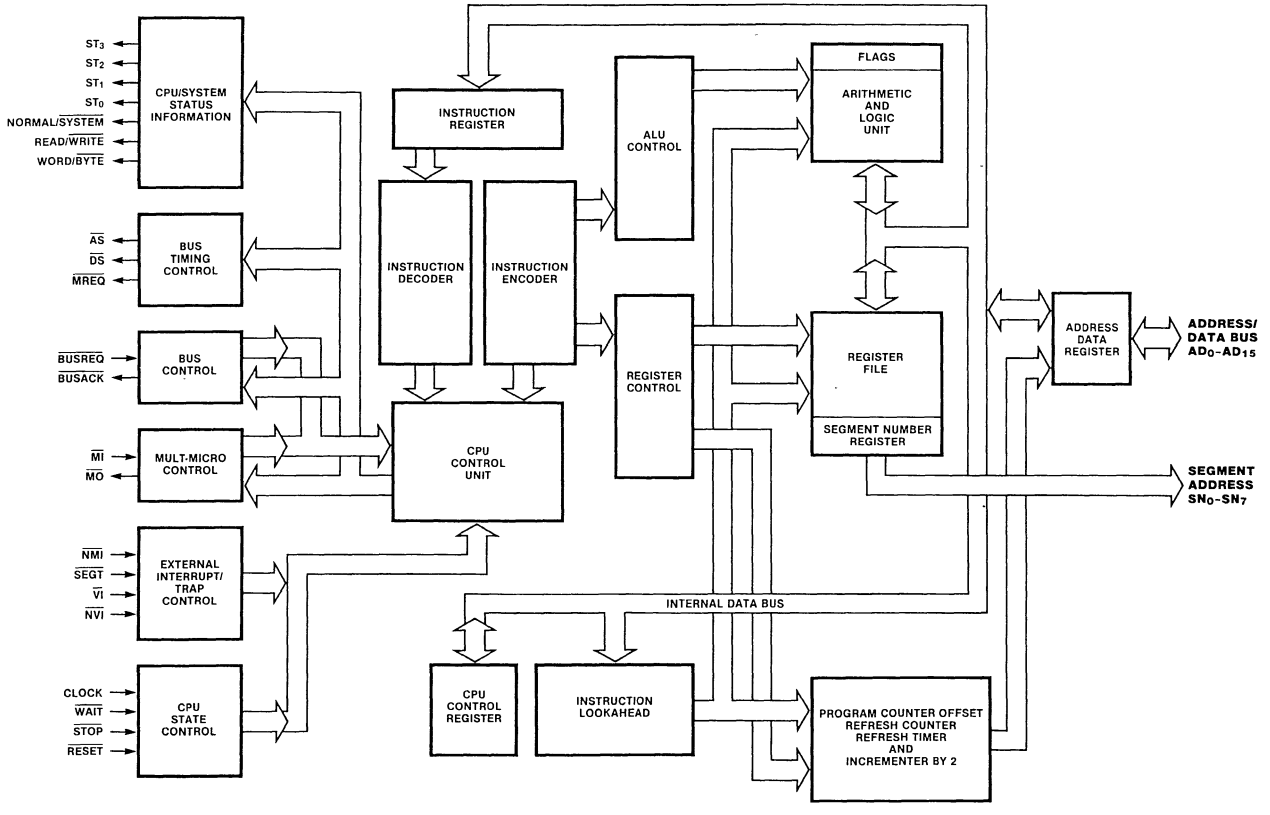


Figure 1. CPU Organization

CPU Organization (Continued)

The Z8000 CPU also contains a number of special-purpose registers in addition to the general-purpose ones. These include the Program Counter, Program Status registers and the Refresh Counter. These registers are accessible through software and provide some of the interesting features of Z8000 CPU architecture.

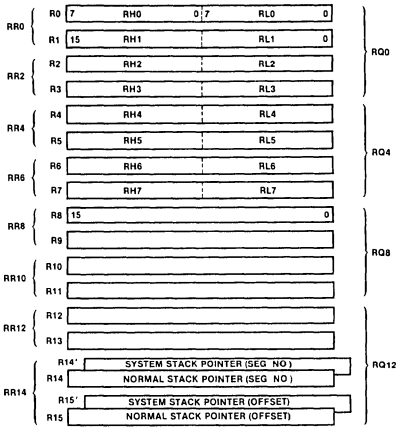


Figure 2. Z8001 General Purpose Registers

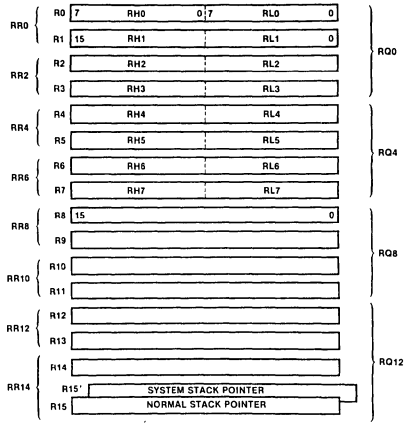


Figure 3. Z8002 General Purpose Registers

Register Organization

All general-purpose registers can be used as accumulators, and all but one as index registers or memory pointers. The one register that cannot be used as an index register is Register 0. Specifying Register 0 is used as an escape mechanism to change the address mode from IR to IM, from X to DA, or—with Load instructions—from BA to RA. This has been done so that the two addressing mode bits in the instruction can specify more than four addressing modes for the same opcode.

The Z8000 CPU register file can be addressed in several groupings: as sixteen byte registers (occupying the upper half of the file only), as sixteen word registers, as eight long-word registers, as four quadruple-word registers, or as a mixture of these. Instructions either explicitly or implicitly specify the type of register. Table 1 illustrates the correspondence between the 4-bit source and destination register fields in the instruction (Figure 4) and the location of the registers in the register file (Figures 2 and 3).

Register Designator	Byte	Word	Long Word	Quadruple Word
0 0 0 0	RH0	R0	RR0	RQ0
0 0 0 1	RH1	R1		
0 0 1 0	RH2	R2	RR2	
0 0 1 1	RH3	R3		
0 1 0 0	RH4	R4	RR4	RQ4
0 1 0 1	RH5	R5		
0 1 1 0	RH6	R6	RR6	
0 1 1 1	RH7	R7		
1 0 0 0	RL0	R8	RR8	RQ8
1 0 0 1	RL1	R9		
1 0 1 0	RL2	R10	RR10	
1 0 1 1	RL3	R11		
1 1 0 0	RL4	R12	RR12	RQ12
1 1 0 1	RL5	R13		
1 1 1 0	RL6	R14	RR14	
1 1 1 1	RL7	R15		

Table 1

Register Organization (Continued)

Note that the byte register-addressing sequence (most significant bit distinguishes between the two bytes in a word register) is different from the memory addressing sequence (least significant bit distinguishes between the two bytes in a word). Long-word (32-bit) and quadruple-word (64-bit) registers are addressed by the binary number of their starting word registers (most significant word). For example, RR6 is addressed by a binary 6 and occupies word registers 6 and 7.

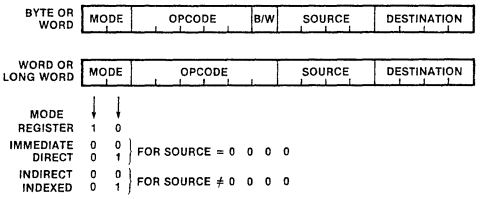


Figure 4. Instruction Format

System/ Normal Mode of Operation

The Z8000 CPU can run in one of two modes: System or Normal. In System Mode, all of the instructions can be executed and all of the CPU registers can be accessed. This mode is intended for use by programs that perform operating system type functions. In Normal Mode, some instructions, such as I/O instructions, are not all allowed, and the control registers of the CPU are inaccessible. In general, this mode of operation is intended for use by application programs. This separation of CPU resources promotes the integrity of the system since programs operating in Normal Mode cannot access those aspects of the CPU which deal with time-dependent or system interface events.

Normal Mode programs that have errors can always reproduce those errors for debugging purposes by simply re-executing the programs with their original data. Programs using facilities available only in System Mode may have errors due to timing considerations (e.g., based on the frequency of disk requests and disk arm position) that are harder to debug because these errors are not easily reproduced. Thus a preferred method of program development would be to partition the task into that portion which can be performed without recourse to resources accessible only in System Mode (which will usually be the bulk of the task) and that portion requiring System Mode resources. The classic example of this partitioning comes from current minicomputer and mainframe systems: the operating system runs in System Mode and the individual users write their programs to run in Normal Mode.

To further support the System/Normal Mode dichotomy, there are two copies of the stack pointer—one for the System Mode and another for Normal. Although the stacks are separated, it is possible to access the normal stack registers while in the System Mode by using the LDCTL instruction.

Status Lines

The Z8000 CPU outputs status information over its four status lines (ST_0-ST_3) and the System/Normal line (S/$\overline{N}$). This information can be used to extend the addressing range or to protect accesses to certain portions of memory. The types of status information and their codes are listed in Table 2.

Status conditions are mutually exclusive and can, therefore, be encoded without penalty. Most status definitions are self-explanatory. One code is reserved for future enhancements of the Z8000 Family.

Extension of the addressing range is accomplished in a Z8000 system by allocating physical memory to specific usage (program vs. data space, for example) and using external circuitry to monitor the status lines and select the appropriate memory space for each address. For example, the direct addressing range of the Z8002 CPU is limited to 64K bytes; however, a system can be configured with 128K bytes if additional logic is used, say, to select the lower 64K bytes for program references and the upper 64K bytes for data references.

ST_3-ST_0	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference
0 1 0 0	Segment trap acknowledge
0 1 0 1	Non-maskable interrupt acknowledge
0 1 1 0	Non-vectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Stack memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer
1 1 1 1	Reserved

Table 2

Status Lines (Continued)

Protection of memory by access types is accomplished similarly. The memory is divided into blocks of locations and associated with each block is a set of legal status signals. For each access to the memory, the external circuit checks whether the CPU status is appropriate for the memory reference. The Z8010 Memory Management Unit is an example of an external memory-protection circuit, and it is discussed later in this tutorial.

The first word in an instruction fetch has its own dedicated status code, namely 1101. This allows the synchronization of external circuits to the CPU. During all subsequent fetch cycles within the same instruction (remember, the longest instruction requires a total of four word fetches), the status is changed from 1101 to 1100. Load Relative and Store Relative also have a status of 1100 with the data reference, so information can be moved from program space to data space.

Refresh

The idea of incorporating the Refresh Counter in the CPU was pioneered by the Z-80 CPU, which performs a refresh access in a normally unused time slot after each opcode fetch. The Z8000 is more straightforward (each refresh has its own memory-access time slot of three clock cycles), and is more versatile (the refresh rate is programmable and capable of being disabled altogether).

The Refresh Register contains a 9-bit Row Counter, a 6-bit Rate Counter and an Enable Bit (Figure 5). The row section is output on AD_0-AD_8 during a refresh cycle. The Z8000 CPU uses word-organized memory, wherein A_0 is only employed to distinguish between the lower and upper bytes within a word during reading or writing bytes. A_0 therefore plays no role in refresh—it is always 0. The Row Counter is—at least conceptually—always incremented by two whenever the rate counter passes through zero. The Row Counter cycles through 256 addresses on lines AD_1-AD_8, which satisfies older and current 64- and 128-row addressing schemes, and can also be used with 256-row refresh schemes for 64K RAMs.

The Rate Counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler (n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed from 1 to 64 μs with a 4 MHz clock. A value of zero in the counter field indicates the maximum time between refreshes; a value of n indicates that refresh is to be performed every 4n clock cycles. Refresh can be disabled by programming the Refresh Enable Bit to be zero.

A memory refresh occurs as soon as possible after the indicated time has elapsed. Generally, this means after the T_3 clock cycle of an instruction if an instruction execution has commenced. When the CPU does not have control of the bus (during the bus-request/bus-acknowledge sequence, for example), it cannot issue refresh commands. Instead, it has internal circuitry to record "missed" refreshes; when the CPU regains control of the bus it immediately issues the "missed" refresh cycles. The Z8001 and Z8002 CPU can record up to two "missed" refresh cycles.

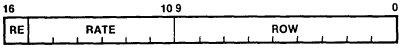


Figure 5. Refresh Counter

Instruction Prefetch (Pipelining)

Most instructions conclude with two or three clock cycles being devoted to internal CPU operations. For such instructions, the subsequent instruction-fetch machine cycle is overlapped with the concluding operations, thereby improving performance by two or three clock cycles per instruction.

Examples of instructions for which the subsequent instruction is fetched while they complete are Arithmetic and Shift instructions. Some instructions for which the overlap is logically impossible are the Jump instructions (because the following instruction location has not been determined until the instruction completes). Some instructions for which overlap is physically impossible are the Memory Load instructions (because the memory is busy with the current instruction and cannot service the fetch of the succeeding instruction).

Extended Instruction Facility

The Z8000 architecture has a mechanism for extending the basic instruction set through the use of external devices. Special opcodes have been set aside to implement this feature. When the CPU encounters instructions with these opcodes in its instruction stream, it will perform any indicated address calculation and data transfer, but otherwise treat the "extended instruction" as being executed by the external device. Fields have been set aside in these extended instructions which can be interpreted by external devices (called Extended Processing Units—EPUs) as opcodes. Thus by using appropriate EPUs, the instruction set of the Z8000 can be extended to include specialized instructions.

In general, an EPU is dedicated to performing complex and time consuming tasks in order to unburden the CPU. Typical tasks suitable for specialized EPUs include floating-point arithmetic, data base search and maintenance operations, network interfaces, graphics support operations—a complete list would include most areas of computing. EPUs are generally designed to perform their tasks on data resident in their internal registers. Moving information into and out of the EPU's internal registers, as well as instructing the EPU as to what operations are to be performed, is the responsibility of the CPU.

For the Z8000 CPU, control of the EPUs takes the following form. The Z8000 CPU fetches instructions, calculates the addresses of operands residing in memory, and controls the movement of data to and from memory. An EPU monitors this activity on the CPU's AD lines. If the instructions fetched by the CPU are extended instructions, all EPUs and the CPU latch the instruction (there may be several different EPUs controlled by one CPU). If the instruction is to be executed by a particular EPU, both the CPU and the indicated EPU will be involved in executing the instruction.

If the extended instruction indicates a transfer of data between the EPU's internal registers and the main memory, the CPU will calculate the memory address and generate the appropriate timing signals ($\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, etc.), but the data transfer itself is between the memory and the EPU (over the AD lines). If a transfer of data between the CPU and EPU is indicated, the sender places the data on the AD lines and the receiver reads the AD lines during the next clock period.

If the extended instruction indicates an internal operation to be performed by the EPU, the EPU begins execution of that task and the CPU is free to continue on to the next instruction. Processing then proceeds simultaneously on both the CPU and the EPU until a second extended instruction is encountered that is destined for the same EPU (if more than one EPU is in the system, all can be operating simultaneously and independently). If an extended instruction specifies an EPU still executing a previous extended instruction, the EPU can suspend instruction fetching by the Z8000 CPU until it is ready to accept the next extended instruction: the mechanism for this is the $\overline{STOP}$ line, which suspends CPU activity during the instruction fetch cycle.

There are four types of extended instructions in the Z8000 CPU instruction repertoire: EPU internal operations; data transfers between memory and EPU; data transfers between EPU and CPU; and data transfer between EPU flag registers and CPU flag and control word. The last type is useful when the program must branch based on conditions determined by the EPU. Six opcodes are dedicated to extended instructions: 0E, 0F, 4E, 4F, 8E and 8F (in hexadecimal). The action taken by the CPU upon encountering these instructions is dependent upon an EPU control bit in the CPU's FCW. When this bit is set, it indicates that the system configuration includes EPUs; therefore, the instruction is executed. If this bit is clear, the CPU traps (extended instruction trap), so that a trap handler in software can emulate the desired operation.

In conclusion, the major features of this capability are, that multiple EPUs can be operating in parallel with the CPU, that the five main CPU addressing modes (Register, Immediate, Indirect Register, Direct Address, Indexed) are available in accessing data for the EPU; that each EPU can have more than 256 different instructions; and that data types manipulated by extended instructions can be up to 16 words long.

Program Status Information

The Program Status Information consists of the Flag And Control Word (FCW) and the Program Counter (PC). The Z8000 CPU uses one byte in FCW to store flags and another byte to store control bits.

Arithmetic Flags. Flags occupy the low byte in the FCW and are loaded, read, set and reset by the special instruction LDCTLB, RESFLG and SETFLG. The flags are:

C Carry

Z Zero

S Sign (1 = negative; two's complement notation is used for all arithmetic on data elements)

P/V Even Parity or Overflow (the same bit is shared)

D Decimal Adjust (differentiates between addition and subtraction)

H Half Carry (from the low-order nibble)

Control Bits. The control bits occupy the upper byte in the FCW. They are loaded and read by the LDCTL instruction, which is privileged in that it can be executed only in the System Mode. The control bits are:

NVIE Non-Vectored Interrupt Enable

VIE Vectored Interrupt Enable

S/N System or Normal Mode

SEG Segmented Mode Enable (Z8001 only)

The SEG bit is always 0 in the Z8002 even if the programmer attempts to set it. In the Z8001, a 1 in this bit indicates segmented operation. A 0 in the Z8001 SEG bit forces non-segmented operation and the CPU interprets all code as non-segmented. Thus, the Z8001 can execute modules of user code developed for the non-segmented Z8002.

Interrupt and Trap Structure

The Z8000 provides a powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processed in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored and non-vectored), three internal traps (system call, unimplemented instruction, privileged instruction) and a segmentation trap. The vectored and non-vectored interrupts are maskable.

The descending order of priority for traps and interrupts is: internal traps, non-maskable interrupts, segmentation trap, vectored interrupts and non-vectored interrupts.

Effects of Interrupts on Program Status

The Flag and Control Word and the Program Counter are collectively called the *Program Status Information*—a useful grouping because both the FCW and PC are affected by interrupts and traps. When an interrupt or trap occurs, the CPU automatically switches to the System Mode and saves the Program Status plus an identifier word on the system stack. The identifier supplies the reason for the interrupt. (The Z8002 pushes three words on the stack; the Z8001 pushes four words.)

After the pre-interrupt or "old" Program Status has been stored, the "new" Program Status is automatically loaded into the FCW and PC. This new Program Status Information is obtained from a specified location in memory, called the Program Status Area.

The Z8000 CPU allows the location of the Program Status Area anywhere in the addressable memory space, although it must be aligned to a 256-byte boundary. Because the Status Line code is 1100 (program reference) when the new Program Status is loaded, the Program Status must be located in program memory space if the memory uses this attribute (for example, when using the Z8010 Memory Management Unit or when separate memory modules are used for program and for data).

The Program Status Area Pointer (PSAP) specifies the beginning of the Program Status Area. In the Z8002, the PSAP is stored in one word, the lower byte of which is zero. The Z8001, however, stores its PSAP in two words. The first contains the segment number and the second contains the offset, the lower byte of which is again zero. The PSAP is loaded and read by the LDCTL instruction.

In the Z8002, the first 14 words (28 bytes) of the Program Status Area contain the Program Status Information for the following interrupt conditions:

Location (In Bytes)	Condition
0-3	Not used (reserved for future use)
4-7	Unimplemented instruction has been fetched, causing a trap
8-11	Privileged instruction has been fetched in Normal Mode, causing a trap
12-15	System Call instruction
16-19	Not used
20-23	Non-maskable interrupt
24-27	Non-vectored interrupt

Effects of Interrupts on Program Status (Continued)

Bytes 28-29 contain the FCW that is common to all vectored interrupts. Subsequent locations contain the vector jump table (new PC for vectored interrupts). These locations are addressed in the following way: the 8-bit vector that the interrupting device has put on the lower byte of the Address/Data bus (AD_0-AD_7) is doubled and added to PSAP + 30. Thus,

Vector 0 addresses PSAP + 30,
Vector 1 addresses PSAP + 32, and
Vector 255 addresses PSAP + 540.

In the segmented Z8001, the first 28 words of the Program Status Area (56 bytes) contain the Program Status Information (reserved word, FCW, segment number, offset), for the following interrupt conditions:

Location (In bytes)	Condition
0-7	Not used (reserved for future use)
8-15	Unimplemented instruction has been fetched causing a trap
16-23	Privileged instruction has been fetched in Normal Mode causing a trap
24-31	System Call instruction
32-39	Segmentation trap (memory violation detected by the Z8010 Memory Management Unit)
40-47	Non-maskable interrupt
48-55	Non-vectored interrupt

Bytes 56-59 contain the reserved word and FCW common to all vectored interrupts. Subsequent locations contain the vector jump table (the new segment number and offset for all vectored interrupts). These locations are addressed in the following way: the 8-bit vector that the interrupting device has put on the lower byte of the Address/Data bus (AD_0-AD_7) is doubled and added to PSAP + 60. Thus,

Vector 0 addresses PSAP + 60,
Vector 2 addresses PSAP + 64, and
Vector 254 addresses PSAP + 568.

Care must be exercised in allocating vector locations to interrupting devices; always use even vectors. Thus there are effectively only 128 entries in the vector jump table. (Figure 6 illustrates the Program Status Area.)

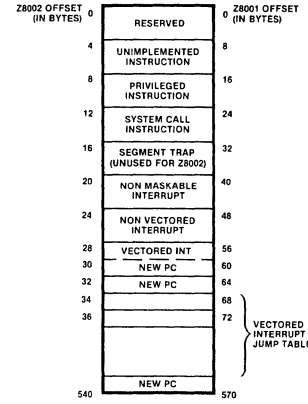


Figure 6. Program Status Area

Z8000 CPU Memory Features

The way a processor addresses and manages its memory is an important aspect in both the evaluation of the processor and the design of a computer system that uses the processor. Z8000 architecture provides a consistent memory address notation in combining bytes into words and words into long words. All three data types are supported for operands in the Z8000 instruction set. I/O data can be either byte- or word-oriented.

The Z8001 CPU provides a segmented addressing space with 23-bit addressing. The Z8010 Memory Management Unit can increase the address range of this processor. To support a memory management system, the Z8001 processor generates Processor Status Information. These signals are also generated by the Z8002 CPU and—as mentioned earlier—can be used to increase the address range of this processor beyond its nominal 64K byte limit. It is not necessary to use a Z8010 Memory Management Unit with a Z8001. The segment number (upper six bits of the address) can be used directly by the memory system as part of the absolute address.

These issues are discussed in more detail in the following sections, along with a description of the method used to encode certain segmented addresses into one word. A brief comment on the use of 16K Dynamic RAMs with the Z8001 concludes this group of sections that deal with Z8000 CPU memory features.

Address Notation

In the Z8000 CPU, memory and I/O addresses are always byte addresses. Words or long words are addressed by the address of their most significant byte (Figure 7). Words always start on even addresses ($A_0 = 0$), so both bytes of a word can be accessed simultaneously. Long words also start on even addresses.

Within a word, the upper (or more significant) byte is addressed by the lower (and always even) address. Similarly, within a long word, the upper (more significant) word is addressed by the lower address. Note that this format differs from the PDP-11 but is identical to the IBM convention.

There is good reason for choosing this format. Because the Z8000 CPU can operate on 32-bit long words and also on byte and word strings, it is important to maintain a continuity of order when words are concatenated into long words and strings. Making ascending addresses proceed from the highest byte of the first word to the lowest byte of the last word maintains this continuity, and allows comparing and sorting of byte and word strings.

Bit labeling within a byte does not follow this order. The least significant bit in a byte, word or long word is called Bit 0 and occurs in the byte with the highest memory address. This is consistent with the convention where bit n corresponds to position 2^n in the conventional binary notation. This ordering of bit numbers is also followed in the registers.

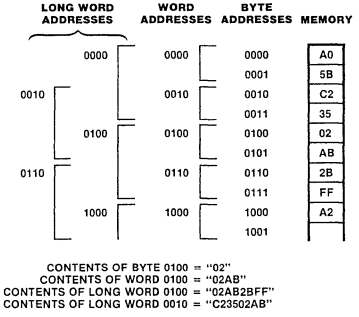


Figure 7. Memory Addressing

Memory and I/O Addressing

Like most 16-bit microprocessors, the Z8000 CPU uses a 16-bit parallel data bus between the CPU and memory or I/O. The CPU is capable of reading or writing a 16-bit word with every access. Words are always addressed with even addresses (A_0 = 0). All instructions are words or multiple words.

The Z8000 CPU can, however, also read and write 8-bit bytes, so memory and I/O addresses are always expressed in bytes. The Byte/Word ($B/\overline{W}$) output indicates whether a byte or word is addressed (High = byte). A_0 distinguishes between the upper and lower byte in memory or I/O. The most significant byte of the word is addressed when A_0 is Low (Figure 8).

For word operations in both the read and write modes, $B/\overline{W}$ = Low, A_0 is simply ignored and A_1–A_{15} address the memory or I/O. For byte operations in the read mode, $B/\overline{W}$ = High, A_0 is again ignored, and a whole word (both bytes) is read, but the CPU internally selects the appropriate byte. For byte operations in the write mode, the CPU outputs identical information on both the Low (AD_0–AD_7) and the High (AD_8–AD_{15}) bytes of the Address/Data bus. External TTL logic must be used to enable writing in one memory byte and disable writing in the other byte, as defined by A_0. The replication of byte information for writes is for the current implementation and may change for subsequent Z8000 CPUs; therefore system designs should not depend upon this feature.

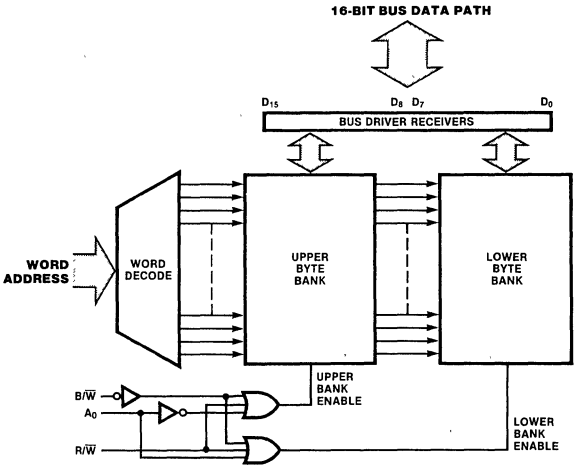


Figure 8. Byte/Word Selection

Segmentation

In organizing memory, segmentation is a powerful and useful technique because it forms a natural way of dividing an address space into different functional areas. A program typically partitions its available memory into disjointed areas for particular uses. Examples of this are storing the procedure instructions, holding its global variables, or serving as a buffer area for processing large, disk-resident data bases. The requirements for these different areas may differ, and the areas themselves may be needed only part of the time.

Segmentation reflects this use of memory by allowing a user to employ a different segment for each different area. A memory management system can then be employed to provide system support, such as swapping segments from disk to primary memory as requested (as in overlays), or in monitoring memory accesses and allowing only certain types of accesses to a particular segment. Thus, dealing with segments is a convenient way of specifying portions of a large address space.

When segmentation is combined with an address translation mechanism to provide relocation capability, the advantages of segmentation are enhanced. Now segments can be of variable user-specifiable sizes and located anywhere in memory.

The Z8001 generates 23-bit logical addresses, consisting of a 7-bit segment number and a 16-bit offset. Thus each of its six memory address spaces consists of 128 segments, and each segment can be up to 64K bytes. Different routines of a program can reside in different segments, and different data sets can reside in different segments. The Z8010 Memory Management Unit translates these logical addresses into physical-memory locations.

Long Offset and Short Offset Addressing

When a segmented address is stored in memory or in a register, it occupies two 16-bit words as previously described for the PC and PSAP. This is a consequence of the large addressing range. When a segmented address is part of an instruction in the Direct Address and Indexed Address Modes, there are two representations: Long and Short Offset addressing.

In the general unrestricted case of Long Offset, the segmented address occupies two words, as described before. The most significant bit in the segment word is a 1 in this case.

The Short Offset Mode squeezes the segment number and offset into one word, saving program size and execution time. Since 23 bits obviously don't fit into a 16-bit word, the 8 most significant bits of the offset are omitted and implied to be zero. The most significant bit of the address word is made 0 to indicate Short Offset Mode. Short Offset addresses are thus limited to the first 256 bytes at the beginning of each segment. This may appear to be a severe restriction, but it is very useful, especially in the Index Mode, where the index register can always supply the full 16-bit range of the offset. Short Offset saves one instruction word and speeds up execution by two clock cycles in Direct Address Mode and three clock cycles in Indexed Mode.

Using the Z8010 Memory Management Unit

The Z8001 CPU can be combined with another 48-pin LSI device—the Z8010 MMU—for sophisticated memory management. The MMU provides address translation from the logical addresses generated by the Z8001 CPU to the physical addresses used by the memory. An address translation table, containing starting addresses and size information for each of the 64 segments, is stored in the MMU. The translation table can be written and read by the CPU using Special I/O instructions. The MMU thus provides address relocation under software control, making software addresses (i.e., logical addresses) independent of the physical memory addresses.

But the MMU provides much more than address relocation; it also monitors and protects memory access. The MMU provides a Trap input to the CPU and—if necessary—an inhibit signal ($\overline{SUP}$) to the memory write logic when specific memory-access violations occur. The MMU provides the following types of memory protection:

- Accesses outside the segment's alloted memory can be prevented.
- Any segment can be declared invalid or non-accessable to the CPU.
- Segments can be declared Read Only.
- By designating a segment as System Only, access can be prohibited during the Normal Mode.
- Declaring a segment Execute Only means it can be accessed only during instruction access cycles. Data or stack use is prohibited.
- Any segment can be excluded from DMA access.
- Segments can have a Direction And Write Warning attribute, which generates a trap when a write access is made in the last 256 bytes of its size. This mechanism can be used to prevent stack overflow.

Multiple MMUs must be used when more than 64 segments are needed. Thus, to support the full complement of 128 segment numbers provided for each Z8001 CPU address space, two MMUs are required. The MMU has been designed for multiple-chip configurations, both to support 128-segment translation tables and to support multiple translation table systems.

Note that the memory management features do not interfere with the ability to directly address the entire memory space. Once programmed, the MMU (or MMUs) translates and monitors any memory address generated by the CPU.

The MMU contains status bits that describe the history of each segment. One bit for each segment indicates whether the segment has been accessed; another bit indicates whether the segment has been written. This is important for certain memory management schemes. For example, the MMU indicates which segments have been updated and, therefore, must be saved on disk before the memory can be used by another program.

When translating logical addresses to physical memory addresses, the MMU must do the following: access its internal 64 × 32-bit RAM, using the segment number as the address, then add the 16 bits of RAM output to the most significant address byte (AD_8–AD_{15}) and finally place the result on its Address outputs. The least significant byte (AD_0–AD_7) bypasses the MMU.

The internal RAM access time is approximately 150 ns. Throughput delay is avoided by making the segment number available early: SN_0–SN_7 are output one clock period earlier than the address information on AD_0–AD_7.

In summary, the Z8000 CPU supports sophisticated memory management through such architectural features as the Status Lines, the R/$\overline{W}$ and S/$\overline{N}$ lines, Segment Trap input line, and early output of segment numbers.

Using 16K Dynamic RAMs with the Z8001

Z8000 systems usually implement most of their memory with 16K × 1-bit dynamic RAMs that have time-multiplexed addresses (Zilog also manufactures this device—the Z6116). In Z8001-based systems with MMUs, CPU Address/Data lines AD_1–AD_7 supply row addresses, MMU address outputs A_8–A_{14} supply column addresses, and MMU outputs A_{15}–A_{23} are decoded to generate Chip Select signals that gate either $\overline{RAS}$ or $\overline{CAS}$ or both.

Gating $\overline{RAS}$ reduces power consumption because all non-selected memories remain in the standby mode. But this technique requires that $\overline{RAS}$ must wait for the availability of the most significant address bits from the MMU. During refresh, the $\overline{RAS}$ decoder must be changed to activate all memories simultaneously.

Gating $\overline{CAS}$ does not achieve lower power consumption; however, this technique allows the use of slower memories because $\overline{RAS}$ can be activated as soon as the CPU address outputs are stable, without waiting for the MMU delay. Also, there is no need to change the $\overline{CAS}$ decoder during refresh.

Data Types and Instructions

The Z8000 architecture directly supports bits, digits, bytes, and 16- or 32-bit integers as primitive operands in its instruction set. In addition, the rich set of addressing modes supports higher-level data constructs such as arrays, lists and records. The Z8000 also introduces a number of powerful instructions that extend the capabilities of microprocessors. The remaining sections of this paper describe Z8000 data types, addressing modes, and a selection of novel instructions.

Data Types

Operands are 1, 4, 8, 16, 32, or 64 bits, as specified by the instruction. In addition, strings of 8- or 16-bit data can be manipulated by single instructions. Of particular interest are the increased precisions of the arithmetic instructions. Add and Subtract instructions can operate on 8-, 16-, or 32-bit operands; Multiply instructions can operate on 16- or 32-bit multiplicands; and Divide instructions can operate on 32- or 64-bit dividends. The Shift instructions can operate on 8-, 16-, and 32-bit registers.

Addressing Modes

The rich variety of addressing modes offered by Z8000 architecture includes: Register, Immediate, Indirect Register, Direct Address, Index, Relative Address, Base Address, and Base Index. Three are of particular interest with respect to high-level data structures: Indirect Register, Base Address, and Base Index. These modes can be used for lists, records, and arrays, respectively.

Indirect Register. In this addressing mode, the contents of the register are used as a memory address. This mode is needed whenever special address arithmetic must be performed to reference data. Essentially, the address is calculated in a register and then used to fetch the data. For example, this mode is useful when manipulating a linked list, where each entry contains a memory pointer to the memory location of the next entry. Essentially, the pointer is loaded into a register and used to access the next item on the list. When the list item is large or has a complex structure, the Base Address or Base Index Modes can be used to access various components of the item.

Base Address. In this addressing mode, the memory address contained in the register (the base) is modified by a displacement in the instruction (known at compile time). This mode is useful, for example, in accessing fields within a record whose format is fixed at compile time.

Base Index. The memory address in this addressing mode is contained in a register (the base) and is modified by the contents of another register (the index). This mode can be useful in accessing the components of an array, because the index of the component is usually calculated during execution time—as a function of the index of a DO-Loop, for example.

Index vs. Base Address. In the Z8002 and in the Z8001 running non-segmented, these two addressing modes are functionally equivalent, because the base address and displacement are both 16-bit values.

When the Z8001 runs segmented, there is a difference: in the Index mode, the base address (including the segment number) is contained in the instruction, in either Short Offset or Long Offset notation. The 16-bit displacement stored in a register is then added to the offset in the base address to calculate the effective address. In the Base Address Mode, on the other hand, the 16-bit displacement is specified in the instruction and is added to the offset of the base address that is stored in a long-word register.

The Instruction Set

The Z8000 offers an abundant instruction set that represents a major advance over its predecessors. The Load and Exchange instructions have been expanded to support operating system functions and conversion of existing microprocessor programs. The usual Arithmetic instructions can now deal with higher-precision operands, and hardware Multiply and Divide instructions have been added. The Bit Manipulation instructions can access a calculated bit position within a byte or word, as well as specify the position statically in the instruction.

The Rotate and Shift instructions are considerably more flexible than those in previous microprocessors. The String instructions are useful in translating between different character codes. Special I/O instructions are included to manage peripheral devices, such as the Memory Management Unit, that do not respond to regular I/O commands. Multiple-processor configurations are supported by special instructions.

The following instructions exemplify the innovative nature of the Z8000 instruction set. A complete list of Z8000 instructions can be found in the reference materials listed at the end of this tutorial.

Load and Exchange Instructions.

Exchange Byte (EX) is practical for converting Z-80, 8080, 6800 and other microprocessor programs into Z8000 code, because the Z8000 uses the opposite assignment of odd/even addresses in 16-bit words.

Load Multiple (LDM) saves n registers and is useful for switching tasks.

Load Relative (LDR) loads fixed values from program space into data space.

Arithmetic Instructions.

Add With Carry and Subtract With Carry (ADC, SBC) are conventionally used in 8-bit microprocessors for multiprecision arithmetic operations. These instructions are rarely used with the Z8000 CPU because it has 16- and 32-bit arithmetic instructions.

Decrement By N and Increment By N (DEC, INC) are intended for address and pointer manipulation, but can also be used for Quick Add/Subtract Immediate with 4-bit nibbles. The flag setting is different from Add/Subtract instructions—as is conventional—in that the Carry and Decimal adjust flags are unaffected by the Increment and Decrement instructions to support multiple precision arithmetic.

Decimal Adjust (DAB) automatically generates the proper 2-digit BCD result after a byte Add or Subtract operation, and eliminates the need for special decimal arithmetic instructions.

Multiply (MULT) provides signed (two's complement) multiplication of two words, generating a long-word result; or of two long-words generating a quadruple word result. No byte multiply exists because it is rarely used and, after sign extension, can be performed by a word multiply.

Divide (DIV) provides signed (two's complement) division of a long word by another word, generating a word quotient and a remainder word; or of one quadruple-word by a long-word, generating a long-word quotient and long-word remainder.

Both Multiply and Divide use a conforming register assignment. That is, a multiply followed by a divide on the same registers is essentially a no-op. The register designation used in the operation description must be even for word operations and must be a multiple of four for long-word operations.

Logical Instructions.

Test Condition Code (TCC) performs the same test as a Jump instruction, but affects the least significant bit of a specified register instead of changing the PC.

Program Control Instructions.

Call Relative (CALR) is a shorter, faster version of Call, but with a limited range.

Decrement And Jump If Non-Zero (DJNZ) is a one-word basic looping instruction.

Jump Relative (JR) is a shorter, faster version of Jump, but with a limited range.

Bit Manipulation Instructions.

Test Bit, Reset Bit, Set Bit (BIT, RES, SET) are available in two forms: static and dynamic. For the static form, any bit (the position is defined in the immediate word of the instruction) located in any byte or word in any register or in memory can be set, reset or tested (inverted and routed into the Z flag).

For the dynamic form, any bit (the position is defined by the content of a register that is, in turn, specified in the instruction) located in any byte or word in any register, but not in memory, can be set, reset or tested.

Test And Set (TSET) is a read/modify/write instruction normally used to create operating system locks. The most significant bit of a byte or word in a register or in memory is routed into the S flag bit and the whole byte or word is then set to all 1s. During this instruction, the processor does not relinquish the bus.

Test Multi-Micro Bit and Multi-Micro Request/Set/Reset (MBIT, MREQ, MSET, MRES) are used to synchronize the access by multiple microprocessors to a shared resource,

The Instruction Set (Continued)

such as a common memory, bus, or I/O device.

Note that the instruction MREQ (Multi-Microprocessor Request) has nothing whatsoever in common with the $\overline{\text{MREQ}}$ (Memory Request) output from the Z8000 CPU.

Rotate and Shift Instructions.

The Z8000 CPU has a complete set of shift instructions that shift any combination of bytes or words, right or left, arithmetically or logically, by any meaningful number of positions as specified either in the instruction (static) or in a register (dynamic).

The CPU also has a smaller repertoire of rotate instructions that rotates bytes or words, either right or left, through carry or not, and by one bit or by two bits.

The instructions Rotate Digit Left and Rotate Digit Right (RLDB, RRDB) rotate 4-bit BCD digits right or left, and are used in BCD arithmetic operations.

Block Transfer and String Manipulation Instructions.

Translate And Decrement/Increment (TRDB, TRIB) is used for code conversion, such as ASCII to EBCDIC. These instructions translate a byte string in memory by substituting one string by its table-lookup equivalent. TRDB and TRIB execute one operation and decrement the contents of the length register; thus they are useful as part of loop performing several actions on each character.

Translate, Decrement/Increment and Repeat (TRDRB, TRIRB) are the same as TRDB and TRIB, except they repeat automatically until the contents of the length register become zero. They are therefore useful in straightforward translation applications.

Translate And Test, Decrement/Increment (TRTDB, TRTIB) tests a character according to the contents of the translation table.

Translate And Test, Decrement/Increment And Repeat (TRTDRB, TRTIRB) scans a string of characters. The first character is tested and, depending on the contents of the translation table, the process stops or skips to the next character. Stopped characters can be used for further processing.

I/O and Special I/O Instructions.

The Z8000 CPU has two complete sets of I/O instructions: Standard I/O and Special I/O. The only difference is the status information on the ST_0-ST_3 outputs. Standard I/O instructions are used to communicate with Z-Bus compatible peripherals. Special I/O instructions are typically used for communicating with the Memory Management Unit.

Both types of instructions transfer 8 or 16 bits and use a type of 16-bit addressing analogous to the Z8002 memory-addressing scheme: For word operations, A_0 is always zero; in byte-input operations, A_0 is used internally by the CPU to select the appropriate byte; in byte-output operations, the byte is duplicated in the high and low bytes of the address/data bus, and external logic uses A_0 to enable the appropriate output device.

Bibliography

Selected Publications on the Z8000 Family

Z8001/Z8002 CPU Product Specification **(00-2045)**

Z8000 CPU Instruction Set **(03-8020-01)**

Z8000 PLZ/ASM Assembly Language Programming Manual **(03-3055-01)**

Z8010 Z-MMU Product Specification **(00-2046)**

An Introduction to the Z8010 MMU Memory Management Unit

Zilog

Tutorial Information

March 1981

Introduction

The declining cost of memory, coupled with the increasing power of microprocessors, has accelerated the trend in microcomputer systems to the use of high-level languages, sophisticated operating systems, complex programs and large data bases. The Z8001 microprocessor supports these advances by offering multiple 8M byte address spaces as well as a rich and powerful instruction set. The Z8010 Memory Management Unit (MMU) supports the Z8001 processor in the efficient and flexible use of its large address space.

Support for managing a large memory can take many forms:

- Providing a logical structure to the memory space that is largely independent of the actual physical location of the data
- Protecting the user from inadvertent mistakes such as attempting to execute data
- Preventing one user from unauthorized access to memory resources or data
- Protecting the operating system from unexpected access by the users.

The Z8010 provides all these features plus additional features that permit a variety of system hardware configurations and system designs.

This paper examines the various uses of memory management in computer systems and how memory management techniques generally meet these requirements. The major features of the Z8010 MMU illustrate how memory management functions can be supported by hardware. A few examples demonstrate how this LSI circuit can be used to configure several different memory management systems.

Motivations for Memory Management

The primary memory of a computer is one of its major resources. As such, the management of this resource becomes a major concern as demands on it increase. These demands can arise from different sources, three of which are of interest in the present context. The first stems from multiple users (or multiple tasks within a dedicated application) contending for a limited amount of physical memory. The second comes from the desire to increase the integrity of the system by limiting access to various portions of the memory. The final source arises from issues surrounding the development of large, complex programs or systems. Each of these three sources involves a multifaceted group of related issues.

When multiple tasks constitute a given system (for example, multiple users of a system or multiple sub-tasks of a dedicated application), the possibility exists that not all tasks may be in primary memory at the same time. (A task is the action of executing a program on its data; a task may be as simple as a single procedure or as complex as a set of related routines.) If the population of memory-resident tasks can vary over time, a useful feature of a system would be the ability for a task to reside anywhere in memory, and perhaps in several different locations during its lifetime. Such tasks are called *relocatable*, and a system in which all tasks are relocatable generally offers greater flexibility in responding to changing system environments than a system in which each task must reside in a fixed location.

A second issue that arises in multi-task environments is that of sharing. Separate tasks may execute the same program on different data, and may therefore share common code. For example, several users compiling FORTRAN programs may wish to share the compiler rather than each user having a separate copy in memory. Alternatively, several tasks may wish to execute different programs using the same data as input, and it may be possible for these tasks to access the same copy of the

Motivations for Memory Management (Continued)

input. For example, a user may wish to print a PASCAL program while it is being compiled; the print process and the compiler process could access the same copy of the text file.

A third issue in multi-task systems is protecting one task from unwanted interactions with another. The classic example of unwanted interaction is one user's unauthorized reading of another user's data. Prohibiting all such interactions conflicts with the goal of sharing and so this issue is usually one of selectively prohibiting certain types of interactions. The issue of protecting memory resources from unauthorized access is usually included in the larger set of issues relating to system integrity.

System integrity takes many forms in addition to protecting a task's data from unwanted access. Another aspect is preventing user tasks from performing operating system functions and thereby interrupting the orderly dispatch of these tasks. For example, most large systems prevent a user task from directly initiating I/O operations because this can disrupt the correct functioning of the system.

Another aspect of separating users from system functions relates to separating system I/O transfers from user tasks, especially with respect to error conditions. For example, an error during a direct memory access, say to a nonexistant memory location, should not cause an error in the program that is currently executing.

A final example of increasing the system integrity is protecting a user task from itself. Obvious errors, such as trying to execute data or overflowing an area set aside for a stack, can be detected while a program is executing and handled appropriately, provided the system is given sufficient information.

The notion of protecting an executing task from performing certain types of actions known to be erroneous introduces a third general motivation for memory management, namely support for the design and correct implementation of large, complex programs and systems. Protecting a task from itself obviously helps in debugging a large program, but there are other system features that can aid in developing complex systems. Modern methodology for developing large systems dictates partitioning a task into a number of small, simple, self-contained sub-tasks with well defined interfaces. Each sub-task generally interacts with only a few other sub-tasks and this communication is carefully controlled. This methodology promotes a systems design that can be readily modified, but it also tends to promote the creation of a large number of nearly independent sub-tasks and many data structures accessible to only one or a few of these sub-tasks. Because modern systems are increasingly driven to support many interacting tasks, possibly written and compiled separately, they must also enforce some communication protocol without sacrificing efficient operation. Modern memory management systems can offer effective tools for implementing large systems designed using this methodology.

In summary, the major goals of memory management systems are to:

- Provide flexible and efficient allocation of memory resources during the execution of tasks
- Support multiple, independent tasks that can share access to common resources
- Provide protection from unauthorized or unintentional access to data or other memory resources
- Detect obviously incorrect use of memory by an executing task
- Separate users from system functions.

Most of today's memory management systems support these functions to some degree. The extent of this support is largely a question of resources to be devoted to these functions and the understood demands of the intended applications for these systems.

The Fundamentals of Memory Management

Memory management has two functions: the *allocation* and the *protection* of memory. Dynamic relocation of tasks during their execution is accomplished by an address translation mechanism. The restriction of memory access is accomplished by memory attribute checking. Both operations occur with each memory request during the execution of a program and both are transparent to the user.

Address translation simply means treating the memory addresses generated by the program as logical addresses to be interpreted or translated into actual physical memory locations before dispatching the memory access requests to the memory unit. Memory attribute checking means that each area of memory has associated with it information as to who can access it and what types of access can be made by each task. Each memory reference is checked to insure that the task has the right to access that location in the given fashion (for example, to read the contents of the location or to write data to that location).

Instead of a linear address space, more elaborate memory management systems have a hierarchical structure in which the memory consists of a collection of memory areas, called segments. Access to this structured memory requires the specification of a segment and an offset within that segment. Thus, instead of specifying memory location 1050 in a linear address space, a task specifices memory location 5 in segment number 23, for example.

The Fundamentals of Memory Management (Continued)

Generally, segments can be of variable size, within limits, and a user can specify the size of each segment to be used. Thus one user may have two segments of two thousand and ten thousand words for his FORTRAN program and data, respectively, while another user might have three segments of three thousand, six thousand and two thousand words for her PASCAL program, data, and run-time stack. If the first user called his data segment number 5, then the first word in his data set would be accessed by the logical address (5,0) indicating segment 5, offset 0. The memory management system translates this symbolic name into the correct physical memory address.

Figure 1 gives a conceptual realization of these two users' logical program spaces. The first user, User A, has his program segment called "Segment 6" and his data segment called "Segment 5." The second user, User B, has her program segment called "Segment 5," her data segment called "Segment 12" and her stack segment called "Segment 2." Notice that both users have named one of their segments "Segment 5," but they refer to different entities. This causes no problem since the system keeps the two memory areas separate. The situation is analogous to both users having an integer variable called "I" in their programs: The system realizes that these are two separate variables stored in different memory locations.

User A's data segment, "Segment 5," is ten thousand words. If he references word 10,050 of Segment 5 he gets an error message from the system indicating that he has exceeded the allocation limit for Segment 5. Note that he does not access word 50 of Segment 6. That is, segments are logically distinct and unordered. A reference to one segment cannot inadvertently result in access to another segment. Thus, in this example, User A is prevented from accidentally (or deliberately) accessing his program as though it were part of his data segment.

Figure 2 illustrates one way that these segments could be arranged in the physical memory. The dotted lines indicate the memory-mapping function from the logical address space of the user to the physical memory locations allocated to him. The figure also indicates the access attributes associated with each user's segments. For example, program segments are "execute only" and data segments are "read/write." Thus a user is prevented from executing a data segment or writing into a code segment.

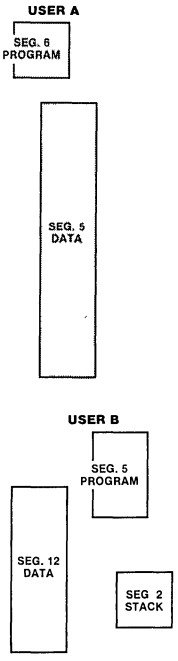


Figure 1. Two User's Logical Address Space

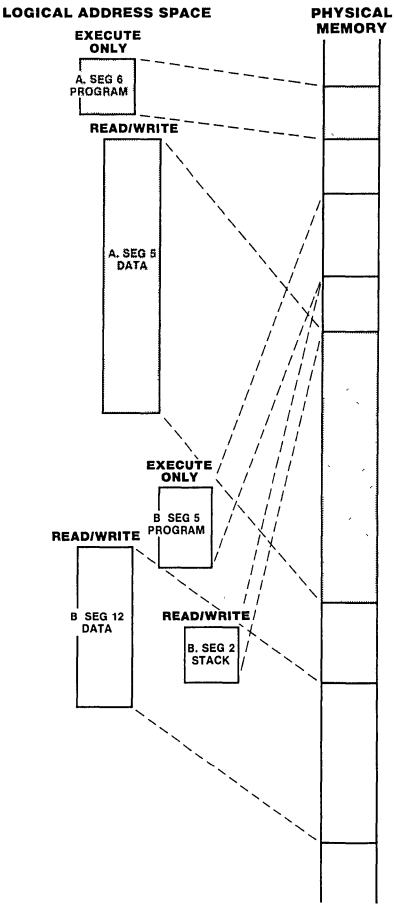


Figure 2. Mapping Logical Segments to Physical Memory

The Fundamentals of Memory Management (Continued)

Figure 3 illustrates what happens when both users have access to the same data set in primary memory, say the results of a questionnaire that both intend to analyze. Each user has a logical name associated with that data set to specify the segment in which the data set is to reside. Note that the two users have chosen to put the data set in different segments of their personal address spaces. The system-mapping function translates these different segment names to the same physical memory locations. Thus User A's access to address (2, 17) references the same physical memory location as User B's access to address (7, 17). In the figure, note that two of B's segments have been moved in physical memory to create a space large enough to hold the questionnaire data.

Another topic in memory management that is supported by Z8001-Z8010 architecture but requires additional support hardware is demand swapping, or segmented virtual memory, which means that the logical memory area may not actually reside in physical memory until a task actually tries to access it. At the time an access is made to a segment missing from physical memory, the instruction execution is held in abeyance until the logical memory can be brought into the physical memory and then the instruction is allowed to proceed with the memory access. The address translation is performed, access protection is checked and the instruction proceeds as if the logical memory area had been in the physical memory at the beginning of the instruction. The instructions in the Z8001 must run to completion before the CPU can perform any action, such as responding to a missing segment trap. But with the conjunction of hardware and software to simulate the above functions, a segmented virtual memory scheme can be implemented.

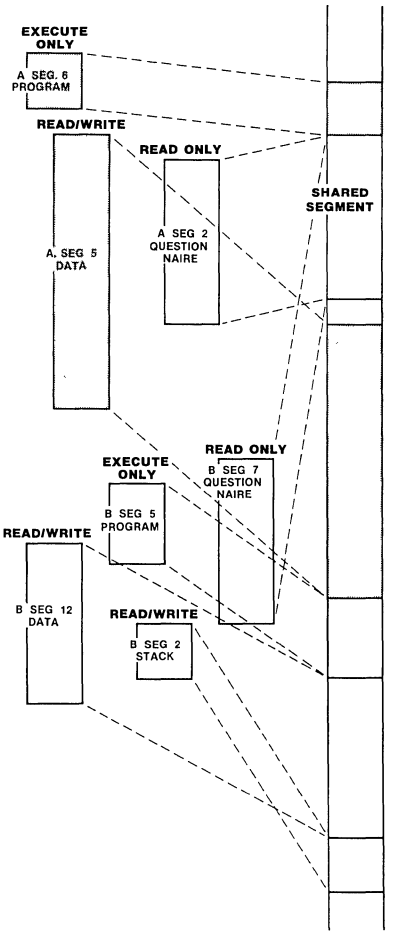


Figure 3. Two Users Sharing a Common Segment

A final topic in memory management is paging, which is another method for partitioning a user address space and mapping it onto the physical memory. Paging is most effective when demand swapping can be supported. Essentially, paging divides the logical memory into fixed-size blocks, called pages. Like segments, the individual pages can be located anywhere in the physical memory and a translation mechanism maps logical addresses to physical memory locations. There are two differences between paging and segmenting a logical memory. First, pages are of fixed size whereas segments are of various sizes. Second, under paging, the logical memory is still linear, that is, a task accesses memory using a single number, rather than a pair as in segmentation. The major advantage of paging is in treating memory as blocks of fixed sizes, which simplifies allocating memory to users and deciding where to place the logical pages in physical memory. The major disadvantage of paging is in assigning different protection attributes to different areas in a user address space because a paged memory appears homogeneous to the user and the operating system. Paging can be combined with segmentation to produce a memory management system with the advantages of both paging and segmentation. The implementation of paging for the Z8001 requires additional support hardware and may be implemented independent of the Z8010.

Before proceeding to the mechanism of memory management, it is instructive to review how a segmented address translation mechanism with protection attributes achieves the five major goals of memory management outlined in the previous section. The first goal permits dynamic allocation of memory during the execution of tasks; that is, a task could be located anywhere in memory and even moved about when its execution is suspended. The address translation mechanism provides this flexibility because the task deals exclusively

The Fundamentals of Memory Management (Continued)

with logical addresses and hence is independent of the addresses of the physical memory locations it accesses. Moving the task to different physical memory locations requires that the address mapping function be changed to reflect the change in memory location, but the task's code need not be modified. Of course, this flexibility does incur the price of managing the various system tables required to implement memory management.

The second goal supports sharing of common memory areas by different tasks. This is accomplished by mapping different logical areas in different tasks to the same physical memory locations.

The third provides protection against certain types of memory accesses. This is accomplished by associating accessing attributes with each logical segment and checking the type of access to see if each access is permitted.

The fourth goal detects obvious execution errors related to memory accessing. This can be accomplished by checking each access to a segment to see whether the address falls within the allocated physical memory for that segment. It could also include affixing a read/write attribute to data to prevent a task from trying to execute a data segment, and affixing an execute-only attribute to code segments to prevent a task from trying to read or write data to this segment. Additionally, if a segment is used for a stack, the system could issue a warning to a task when the stack approaches the allocated limit of the segment. The task could then request more memory for the stack before the stack overflows and creates a fatal error.

The final goal listed for memory management systems separates user functions from system functions. For processors that distinguish between System mode and User mode of operation, this goal can be accomplished by associating a system-only attribute with system segments so users cannot directly access system tables and tasks.

As a final point, it should be noted how segmentation can be used to support the development and execution of large, complex programs and systems. The concept of segmentation corresponds to the concept of partitioning a large system into procedures and data structures where each procedure and data structure can be associated with a separate segment. A task can then invoke a procedure or sub-task or access a data structure by referring to its logical segment name. Access to these objects can be individually restricted by using the protection-checking mechanism of the memory management system.

As a specific example of how segmentation could be used in the design of a large system, consider a multi-user interactive BASIC system with a large data base shared by all users. Such a system could be designed with segments 0 through 15 reserved for system use, segments 16 through 31 reserved for the BASIC interpreter and its internal tables, segments 32 through 63 allocated to user tasks and segments 64 through 127 reserved for portions of the data base when they are in primary memory being accessed by users. For this system, segments 0 through 31 would probably always be in memory; the other segments would be assigned as needed and the memory they require allocated dynamically.

The Mechanics of Memory Management

Essentially there are four issues in implementing a memory management system: how addresses are specified, how these addresses are translated, what attributes are checked for each access, and how the protection mechanism is implemented. Some of the major alternatives in each of these issues are briefly discussed here, primarily from the point of view of a segmented memory.

Two approaches have traditionally been taken for specifying addresses in a segmented memory. For simplicity, only addresses in instructions are discussed. The first way puts all the addressing information in the instruction itself. That is, each memory address in an instruction contains both the segment name and the offset within the segment. The alternative sets aside special registers that contain some of this information, for example the segment name or the address in physical memory where the segment resides.

The advantage of the latter approach lies in the fact that fewer bits are needed in an instruction to specify addresses. Thus programs may be shorter. Also, because there is reduced traffic between the memory and the processor for fetching shorter instructions, a program may execute faster.

On the other hand, these special registers must be manipulated to access more segments than there are registers, and this manipulation adds to the number of instructions, the program size and the execution time. In practice, these can destroy the advantages described above. If the special registers contain physical memory locations, then these must be protected from user access to maintain the integrity of the system, and changing segments requires system calls which can be time consuming if too few registers are supplied. The Z8001 architecture specifies the complete logical address in the instruction.

Address translation is performed by adding the logical segment offset to the memory location where the segment begins. Thus, when an address of the form (a, b) is presented to the translation mechanism, the segment name "a" is used to determine where segment "a" resides in memory. Assume that it resides in locations 10000 to 25000. Then the actual

The Mechanics of Memory Management (Continued)

memory location of (a, b) is memory location 10000 + b. The major option in implementing this type of address translation is in determining the segment location in physical memory. When special registers have been set aside to contain the starting location of the segment instead of putting all address information in the instruction, the addressing mechanism is similar to using the segment register as an index register or a base register.

When logical addresses are either completely specified in the instruction or when the special register contains the symbolic segment name, a table must be used to translate the logical segment name into a physical memory location. The table may have an associative capability, that is, the segment name is presented to the table and the device returns the physical memory location where the segment begins. Alternatively, the table could have one entry for every possible segment name. The Z8010 implementation of the address translation table sets aside a specific table entry for each logical segment name.

A number of attributes can be associated with a segment and checked during each access. One of these is the allocated length of the segment, and each access is checked to see if it falls within the bounds of the segment. The Z8010 provides limit checking.

Another type of attribute deals with ownership or class of ownership: tasks are grouped into classes and only those in certain classes are permitted access. The simplest example is the system versus user classification, where tasks are either one or the other and this determines whether or not any type of access can be made to the segment. The Z8010 has this feature—users are prevented from accessing system segments.

Other types of attributes that can be associated with a segment involve modes of accessing, for example read only, read/write or execute only. For these attributes, the processor must indicate the type of access to be made, be it code fetch, read from memory, write to memory, etc. The Z8001 indicates when it is fetching code, reading or writing data, or performing stack operations, and thus the Z8010 can offer protection for these operations. The other issue with respect to attributes is whether they are permissive or prohibitive. That is, whether the attribute is in the form of "write to this segment is permitted" or of the form "write to this segment is prohibited." The Z8010 adopts the approach of specifying attributes that prohibit certain types of accessing.

The final issue in the mechanics of memory management systems is the implementation of the protection attributes. These may be associated either with the logical address space or with the physical memory itself. The IBM 360 series, for example, places the memory protection information with the physical memory itself. Thus the processor generates a memory address and the memory module checks to see if the access is permitted. The main difficulty with this approach is in the lack of flexibility, because protection is associated with fixed memory partitions. Also, sharing memory is cumbersome because each user is given a protection key to match the memory key; thus both users must have the same access key or a universal access key. Associating access attributes with the logical segment permits a versatile memory management scheme because different users can access the same segment and have different access attributes associated with their accessing. The Z8010 implements access attributes using the segment mapping information.

Other information associated with each segment does not pertain to the protection mechanism but can be of use to the memory management system. This information generally relates to the history of the segment; for example, whether a segment has been modified while resident in primary memory. If it has not been modified and the system requires the memory for another segment, the memory can be freed immediately; otherwise, the updated version of the segment must be stored in secondary memory and the primary memory is not available until the segment has been saved. Although not strictly necessary, such information can improve the performance of the memory management system. The Z8010 collects information on segment usage, and this information can be used to enhance performance of systems that use this device.

The Z8010 Memory Management Unit

The Z8001 CPU generates segmented addresses consisting of a 7-bit segment number and a 16-bit segment offset address. In addition, the CPU generates status signals indicating its current mode of operation (such as Instruction Fetch, Data Memory Reference, Stack Memory Reference, and Internal Operation), whether it is performing a Read or a Write Memory Reference and whether it is in Normal (User) or System Mode. The Z8010 Memory Management Unit uses this information to perform its memory management functions. This section describes the Z8010 MMU in some detail, beginning with the translation procedure and continuing with a description of the internal registers of the chip. The section concludes with a description of the system commands that alter the contents of these registers.

The Z8010 MMU has three functional states. The first is the memory management state: when a logical address is presented to the unit, the MMU checks the access to insure its validity and translates the logical address to a physical memory location. The second state is a command state: when a special I/O instruc-

The Z8010 Memory Management Unit (Continued)

tion is issued to the MMU, such as reading or writing one of its internal registers, the MMU responds to the command as appropriate. The third state is a quiescent state: when the CPU issues an I/O instruction or a refresh cycle, the MMU address lines remain 3-stated.

The inputs to the MMU are the Address/Data lines (A/D lines), Segment Number lines, Bus Status and Timing Lines, and special control lines for chip selection and DMA. The outputs from the MMU are Address lines, a Segment Trap line and a Suppress line (Figure 4). During address translation and access protection, logical addresses are presented to the MMU on the Segment Number and Address/Data lines; the MMU puts the translated physical memory location on its Address lines and, if appropriate, activates the Segment Trap and/or Suppress lines.

Segment Trap is a special type of synchronous interrupt for the Z8001 CPU; Suppress aborts the memory access. In the command state, the MMU receives commands on the A/D lines; data to be read from or written into the MMU is also placed on the A/D lines.

The MMU selects which of the three states it will be in according to the status information on the Bus Status lines during the initial clock cycle of an instruction or DMA cycle. The MMU performs address translation during a memory reference for either a regular instruction or a DMA request. Only I/O instructions (either regular or special), memory refresh and reserved bus status states cause the MMU to cease performing memory address translations and enter another state.

The MMU uses the segment number to access an internal table of segment descriptor registers, each register containing the starting memory location of the segment (called the base address), the segment's limit (used to determine the range of legal address offsets) and the types of accesses permitted to that segment.

Physical memory for segments is allocated in blocks of 256 bytes. The eight least significant bits of the base address are all zero and are not stored in the Segment Descriptor Register. Also, since the eight low-order bits of the segment base are always zero, the eight low-order bits of the segment offset need not participate

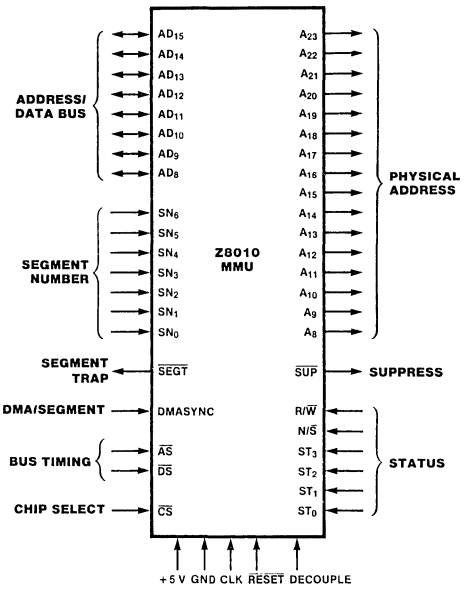


Figure 4. Z8010 MMU Pin Functions

in the addition of the base address to the offset. Rather, they can be juxtaposed to the result of adding the high-order byte of the offset to the most significant 16 bits of the base address.

This process is illustrated in Figure 5. Note that the low-order eight bits of the offset are not used by the MMU. Figure 6 goes through an example of mapping the logical address (5, 1528) to a physical memory location when segment 5 begins at location 231100.

Figure 6a illustrates the full addition to be performed during address translation. The segment number 5 selects Segment Descriptor Register 5 in the MMU. The base address field in this register contains 2311 which corresponds to a base address of 231100. The offset, 1528, is then added to 231100 to produce the physical memory location 232628. Figure 6b represents the same logical procedure, but illustrates the actual operation of the MMU. Again segment number 5 is used to select the base address. However, only the high-order byte of the offset is added to the contents of the

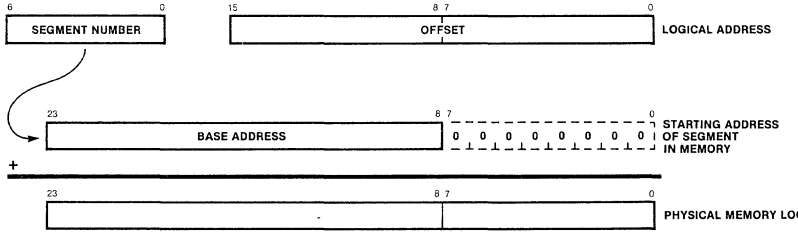


Figure 5. Generation of the Physical Memory Location from a Logical Address

The Z8010 Memory Management Unit (Continued)

MMU base-address field: 15 is added to 2311 to produce the most significant 16 bits of the physical memory location. The low-order byte of the physical location is the same as the low-order byte of the offset.

The results of the two processes illustrated in figures 6a and 6b are the same, but in 6a a 24-bit addition is implied whereas in 6b only a 16-bit addition is needed. Also, the low-order eight bits of the offset are not needed by the MMU and this reduces the number of pins required by the MMU package.

The MMU checks memory references for two types of trap conditions. The first type is an access violation. This occurs when a memory reference is performed in a mode that is not allowed by the read-only, execute-only, CPU-inhibit or system-only attribute of a segment. A memory reference outside the allocated memory for the segment also constitutes an access violation.

The second type is a write warning. This occurs when a write is made to the last 256 bytes of a special type of segment (indicated by a special attribute flag called the Direction And Warning Flag). These segments are typically used for stacks and are therefore logically organized so that successive writes (or stack pushes) access lower-numbered memory locations. By generating a segment trap request when a write is performed into the lowest-numbered 256 bytes of the memory allocated for these segments, the MMU is signaling that a stack is in danger of overflowing. The operating system in servicing this trap can increase the memory allocated for the segment and avoid a fatal stack overflow condition.

The MMU generates two control signals that can be used by the system to perform memory management functions. Segment Trap Request is generated upon the first detected occurrance of a violation or write warning. Once asserted, this signal remains set until a trap acknowledge signal is received. Only when the Fatal Flag, a special MMU control flag, is set will a detected violation not cause a segment trap request. This flag is set only when a second violation is detected while a previous trap is being processed and thus indicates that the system software is in error.

The other control signal generated by the MMU is Suppress. Once a violation has been detected, this signal is asserted on that and every succeeding memory reference for the remainder of the instruction. In particular, I/O and Special I/O instructions are checked for memory access violations, and once a memory access violation is detected, subsequent memory accesses cause Suppress signals to be generated. I/O addresses, of course, bypass the MMU and are neither translated nor checked. Intervening DMA cycles and memory refresh cycles are exceptions to this rule. During such cycles Suppress is not asserted unless a violation is detected during that cycle. Only DMA can generate a violation; refresh can never cause a violation. Suppress can be used by the memory system to inhibit writes, thus protecting the memory from illegal alterations.

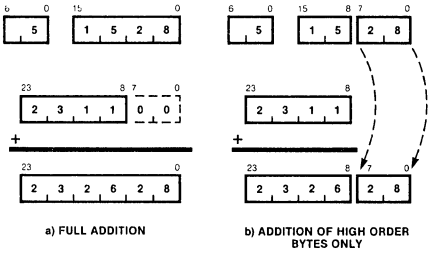


Figure 6. Two Methods of Address Translation

MMU Internal Registers

There are three groups of registers in the MMU: Segment Descriptor Registers, Control Registers and Status Registers. The Segment Descriptor Registers contain all the information relating to the address translation and access protection of a particular segment. The Control Registers contain information used to control the various functions of the MMU, including how to interpret various signals generated by the CPU. The Status Registers contain all the information the MMU generates when it detects an access violation.

Segment Descriptor Registers

Because there are 64 Segment Descriptor Registers in the MMU, two MMUs are required to handle all 128 segments that the Z8001 can manipulate directly. An MMU is programmed to handle either segments 0 through 63 or segments 64 through 127; the particular set of 64 segments in an MMU can be changed using special operating system commands. Each Segment Descriptor contains three fields, a 16-bit Base Field, an 8-bit Limit Field and an 8-bit Attribute Field (Figure 7). The segment number of a logical address determines which segment descriptors are used in address translation.

The Base Field specifies the starting location in memory of the segment.

The Limit Field specifies the segment size in blocks of 256 bytes. The address offset is compared against the segment limit and a size violation occurs if the offset falls outside the segment boundaries. A write warning occurs if the destination is in the last block of a segment being used as a stack.

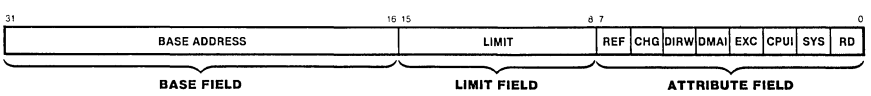


Figure 7. A Segment Descriptor

The Attribute Field contains eight flags. Five flags protect the segment against certain types of access, one indicates a special orientation of the segment, and two indicate the types of accesses that have been made to the segment. The following brief description explains how these flags are used.

The Read-Only Flag (RD) indicates that the only accesses to this segment are reads. Writes are prohibited when this flag is set. Thus this flag is a write-inhibit flag; in particular, code can be executed from a read-only segment. This flag is useful in protecting data from being written by unauthorized users. For example, if one user wants to give another access to a document that he has created, but does not want this user to be able to modify it, the system can set the Read-Only Flag when it copies the file into the user's address space. If the data is already in memory (in a read-only mode), then this same memory area can be made accessible to that user without another copy of the document being required.

The System-Only Flag (SYS) indicates that only accesses made in System Mode are to be permitted. When this flag is set, accesses in the Normal Mode are prohibited. This attribute is useful in protecting system tables and tasks from being accessed by users. For example, system I/O routines can be left in the memory with this flag set and a user is unable to call them directly. This feature is useful if a system is designed so that users are given certain segment names and other segment names are reserved for system use. This flag prevents users from accessing system segments, even though they can generate the logical addresses.

The CPU-Inhibit Flag (CPUI) indicates that the segment is not to be referenced by the CPU. When this flag is set, CPU access to this segment is prohibited, but DMA channels can access the segment. This flag is useful in preventing a program from accessing a segment whose data resides on secondary storage and has not been brought into primary memory. For example, a user may request the operating system to read a file from disk into segment number 19; if the operating system returns control to the user before the file has been read, this flag should be set in Segment Descriptor Register 19.

The Execute-Only Flag (EXC) indicates that the segment is to be referenced only during the instruction fetch cycle of the processor. When this flag is set, access to the segment during any other cycle of an instruction, for example during the memory request cycle, is prohibited. This flag is useful in preventing a program from making a copy of a proprietary program. For example, if this flag is set for a segment containing code that a user can access, that code is protected from being read and hence from being copied.

The DMA-Inhibit Flag (DMAI) indicates that the segment is not to be referenced by a DMA Channel. When this flag is set, only the CPU has access to the segment. This flag is useful in preventing a DMA device from modifying a segment being used by an executing task. For example, segments with valid data should have this flag set to protect them from modification by a DMA device.

The Direction And Warning Flag (DIRW) indicates that memory accesses are to be monitored and certain accesses are to be signaled, although allowed to proceed. When this flag is set, any write to the lowest 256 bytes of the segment generates a write warning. This flag is useful for segments that are used as stacks since the Z8001 has special stack instructions to manipulate stacks that grow toward lower memory locations. Thus a write warning for a stack indicates that the stack may soon overflow its allotted memory space and that more physical memory should be obtained. For example, if a segment serves as a run-time stack for a block-structured programming language such as PASCAL, memory can be allocated to this segment only as a program requires during its execution. The alternative in a fixed allocation environment is to allocate as much memory for the stack as the system expects the program to need, whether or not it is actually used by the program.

The Changed Flag (CHG) indicates that a write has occurred to this segment. This flag is set automatically whenever a program or DMA device writes into the segment. This flag is useful in indicating which segments have been modified in the case where the segment must be written to a secondary storage device. Segments that have not been updated need not be copied back to disk if a copy already exists. For example, when a user task is suspended in a multiple-user environment and his task is to be swapped out of memory temporarily to make room for another task, only those segments that have been changed need to be updated on the disk.

The Referenced Flag (REF) indicates that a memory access has been made to a segment. This flag is set automaticaly whenever a program or DMA device accesses the segment. This flag is useful in indicating which segments are active in the case that a segment must be

Segment Descriptor Registers (Continued)

selected to be swapped out of primary memory to make room for another task. For example, seldom-used operating-system tasks that usually reside in primary memory may be swapped out to make room for users with large memory requirements. This flag is a way of ascertaining which segments contain seldom used tasks.

Control Registers

Three user-accessible 8-bit registers in the MMU control the functioning of the MMU (Figure 8). The Mode Register provides a sophisticated method for selectively enabling MMUs in a multiple-MMU configuration. The Segment Address Register (SAR) selects a particular segment descriptor to be accessed by a system routine when it is changing the organization of primary memory. The Descriptor Selection Counter Register selects the particular byte in the Segment Descriptor Register that is accessed.

Two flags in the Mode Register govern the functioning of the MMU. The Master Enable Flag (MSEN) indicates whether the device will perform address translation. When this flag is set, addresses translated by the MMU are placed on its Address lines; when this flag is clear, the Address lines are 3-stated. Thus, once this flag is reset, no memory request can pass through the MMU. In a single-MMU configuration, MSEN set to zero requires that the CPU must have access to a special memory, since it will not be able to fetch an instruction from the primary memory. This flag can be set during hardware reset (this is discussed later).

The second flag in the mode register that governs the functioning of the MMU is the Translate Flag (TRNS). This flag indicates whether the MMU is to translate the addresses presented to it. When the flag is set, the MMU translates logical addresses to physical memory locations and checks to see if a violation will occur on that access. When the flag is clear, addresses presented to the MMU are passed to the output Address lines without change, and no protection checking is done.

When multiple-MMUs are used in a memory-management system, some mechanism must be present to select those devices that are to be active during the memory translation process. More specifically, if two MMUs are employed so that all 128 segments can be used at random by an executing process, then some way must exist for each of the MMUs to know which 64 Segment Descriptors are located in its Segment Descriptor Registers. The Upper Range Select Flag (URS) indicates which set of 64 descriptors is stored in the MMU. When the flag is set, the MMU contains descriptors 64 through 127; when the flag is reset, the MMU contains descriptors 0 through 63.

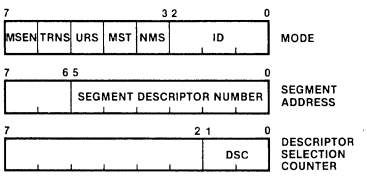


Figure 8. MMU Control Registers

When multiple-MMU devices keep separate tables for system descriptors and user descriptors, the Multiple Segment Table Flag (MST) and the Normal Mode Select Flag (NMS) in the Mode Register distinguish which MMUs contain system descriptors and which contain user descriptors. When the MST flag is set, multiple tables are present in the configuration, and each MMU is dedicated to one of the tables. In this case the MMU translates addresses only when the $N/\overline{S}$ signal matches the NMS flag. Thus, if there are two tables in the memory management system (one for the system and one for users), the NMS flag is set in those MMUs containing the users' segment descriptors, and is not set in the remaining MMUs. All MMUs in the system have the MST flag set to indicate more than one table in the system.

The final piece of control information in the Mode Register is a 3-bit Identification Field (ID) that indicates a logical name for the MMU. When a segment trap is acknowledged by the CPU, the MMU uses this field to select one of the A/D lines; each enabled MMU should select a different line. If an MMU requested a segment trap, it outputs a 1 on its assigned A/D line; otherwise it outputs a 0. Since the ID field is three bits, up to eight MMUs can be uniquely identified. One instruction might result in multiple violations in different MMUs, so that the segment trap software might have to deal with several MMUs to process the trap.

The other two control registers in the MMU are the Segment Address Register (SAR), which points to one of the 64 segment descriptors, and the Descriptor Selection Counter Register. Commands to read or write a segment descriptor use the SAR pointer to select which descriptor is to be accessed. This register has an auto-incrementing capability for accessing consecutive descriptors in succession without having to reload the SAR. Thus if descriptors 0 through 4 are to be modified, the SAR is initialized to 0 and then auto-incremented to point to descriptors, 1, 2, 3 and 4 in succession.

The Segment Descriptor Number is a 6-bit field that contains the address of the descriptor within the MMU. If the MMU holds segments 64 through 127 (that is, if the URS flag is set), the segment named 64 is accessed when the SAR number field is 0. This is a result of the 6-bit limit of the descriptor number field. The field indicates the 6 least-significant bits of the logical segment descriptor number.

Control Registers (Continued)

Segment Descriptors consist of four bytes; the Descriptor Selection Counter indicates which byte is being accessed during a command (commands to the MMU can read or write only one byte at a time). A counter value of 0 indicates the high-order byte of the base address is being accessed, 1 indicates the low-order byte of the base address, 2 indicates the limit field, and 3 indicates the attribute field. This counter is used by MMU commands that access multiple bytes within a descriptor. In general, the counter is handled automatically by the MMU commands. Only when a command could be interrupted—and intervening MMU commands issued—should this register be saved and later restored by the interrupting program.

Status Registers

Six 8-bit registers contain information useful in recovering from memory trap conditions (Figure 9). The Violation Type Register describes the conditions that generated the segment trap. The Violation Segment Number and Offset Registers contain the segment number and upper byte of the segment address offset for the logical address that caused the segment trap. The Instruction Segment Number and Offset Registers contain the segment number and uper byte of the segment address offset for the last instruction before the segment trap was issued. The Bus Cycle Status Register records the status of the bus at the time the trap condition was detected.

Only violations caused by CPU access have trap information stored in the status registers; DMA violations cause Suppress to be asserted, but the Status Registers are not altered. Thus if a DMA violation occurs between a CPU violation and entry to the trap service routine, the service routine still has the CPU trap information available to process the trap. It is the responsibility of the DMA device to save enough information in the event of a violation so that a software DMA violation service routine can process the violation correctly.

Eight flags in the Violation Type Register describe the cause of the segment trap. Four flags correspond to access protection modes in the segment descriptor attribute mode. A read-only violation sets the RDV flag, a system-only violation sets the SYSV flag, a CPU access to a CPU-Inhibit segment sets the CPUIV flag, an execute-only violation sets the EXCV flag.

Three flags correspond to addressing violation or warnings. The Segment Length Violation Flag (SLV) is set whenever the offset of the logical address falls outside the memory space allocated to the segment. The Primary Write Warning Flag (PWW) is set whenever a write occurs in the last 256 bytes of a segment whose Direction And Warning Flag is set (that is, for segments being used as stacks where the top of the stack is within 256 bytes of the allocated memory space of the segment). The Secondary Write Warning Flag (SWW) is similar to the PWW flag, only it is set when the CPU is in system mode, a stack push is being performed to a segment with a Direction And Warning Flag set, and some other addressing violation or warning has occurred (the EXCV, CPUIV, SLV, SYSV, RDV or PWW flags have been set). When the SWW flag is set it indicates that the system stack is in danger of overflowing its allotted memory. Once the SWW flag is set, further write warnings are suppressed. This prevents the system from repeatedly being interrupted for the same warning while it is in the process of eliminating the cause of the warning.

The final violation-type register flag to be discussed is the Fatal Condition Flag (FATL). This flag is set when any other flag in the violation type register is set and either a violation is detected or a write-warning condition occurs in normal mode. This flag is not set during a stack push in system mode that results in a warning condition. This flag indicates that a memory access error has occurred in the trap processing routine. Once this flag has been set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

The Bus Cycle Status Register contains information pertaining to the status of the bus when a trap condition is detected. This includes CPU Status (ST_0-ST_3), plus flags indicating whether a read or a write was being performed and whether or not the N/$\overline{S}$ line was asserted.

The Violation Segment Number and Offset Registers record the first logical address to cause a trap. Only the high-order byte of the offset is saved, however, so that external support circuitry is needed to save the low-order eight bits of the logical address offset. If the trap occurred during the instruction fetch cycle, this information is the logical address of the instruction; otherwise it indicates the

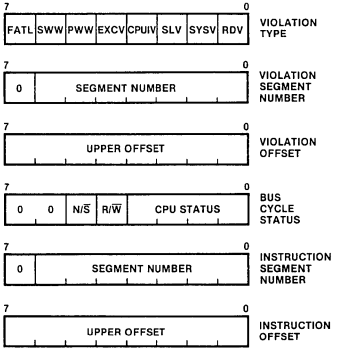


Figure 9. MMU Violation Information Registers

Status Registers (Continued)

logical address of a data item which was to be accessed.

The Instruction Segment Number and Offset Registers record the logical address of the last instruction fetch that occurred before the trap. Only the high-order byte of the offset is saved, however, so external support circuitry is needed to save the low-order eight bits of the offset.

If an instruction fetch caused the trap, these registers indicate the logical address of the previous instruction. Such information is useful if the preceding instruction was a branch instruction to an invalid address since—in this case—these registers indicate which branch instruction led to the erroneous situation. If a data reference caused the segment trap, then these registers indicate the logical address of the instruction that specified the illegal access.

Stack Segments

Segments are specified by a base address and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access outside the segment is attempted, a Trap Request and a Suppress signal are generated.

Normally the legal range of offsets within a segment is from 0 to 256N + 255 bytes, where $0 \leq N \leq 255$. (N is the value in the limit field of the segment descriptor.) However, a segment may be specified so that legal offsets range from 256N to 65,535 bytes, where $0 \leq N \leq 255$. The latter type of segment is useful for stacks because the Z8001 stack-manipulation instructions cause stacks to grow toward lower memory locations. Thus, when a stack grows to the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

The DIRW flag indicates that a segment is to be treated in this special way by the MMU. When the DIRW flag is set, the range of allowed offsets is from 256N to 65,535 bytes and writes into the range 256N to 256N + 255 generate Segment Trap but not Suppress, indicating a write warning.

Segment Trap and Acknowledge

The Z8010 MMU generates a Segment Trap whenever it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress. Suppress can be used to inhibit memory writes and to request that special data be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU memory references until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only; no segment trap requests are ever generated during DMA cycles. This is because the CPU would not be able to respond to these traps until the conclusion of the DMA cycle.

Segment traps to the Z8001 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU enters a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is aborted. The MMU has been designed so that this dummy instruction fetch cycle is ignored. During the acknowledge cycle, all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap request outputs a 1 on the A/D line associated with the number in its ID field. An MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a segment trap acknowledge cycle, an MMU uses A/D line 8 + i if the content of its ID field is i.

Following the acknowledge cycle, the CPU automatically pushes the program status words and program counter onto the system stack, and loads a new program status word and program counter from the program status area. The Segment Trap line is reset during the segment trap acknowledge cycle, and no Suppress signal is generated during the stack push. If the store creates a write warning condition, a segment trap request is generated and is serviced at the end of the context swap; the SWW flag is also set. Servicing this second Segment Trap request also creates a write warning condition, but—because the SWW flag is set—no Segment Trap request is generated. If a violation rather than a write warning condition occurs during the context swap, the FATL flag is set rather than the SWW flag. In this case, subsequent violations cause the Suppress to be asserted but not Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to pro-

Segment Trap and Acknowledge (Continued)

cess the violations they create.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the violation type register reset.

Commands to the MMU

When a memory management system must read or change information in the MMU to respond to a segment trap or to re-organize the physical memory, it can issue control commands to the MMU. These commands fall into two generic categories: reset commands and read/write commands. Reset commands are simply orders to the MMU to set or clear specified fields. For these commands, the Z8001 Special I/O output command can be used with the destination field set to be the MMU command code corresponding to the desired action.

Read and write commands are slightly more complicated because they consist of both commands and data. Such commands to the MMU are issued using the Z8001 Special I/O instructions. These instructions have a source and a destination field. For an input instruction, the source field contains an MMU command code and the destination field indicates where in primary memory the data is placed. For an output instruction, the destination field contains an MMU command and the source field indicates where the data to be written into the MMU resides in memory.

The high-order byte of the command contains the opcode for that command; the low-order byte of the command can be used to specify the particular MMU to be accessed. The MMU does not receive information on AD_0-AD_7, so external circuitry must decode information on these lines during the Special I/O commands and then select a particular MMU. The encoding of the low-order byte is dependent upon the system implementation. This paper always uses the convention that bit i specifies MMU number i.

The reset commands to the MMU are: Reset Violation Type Register, Reset SWW Flag In Violation Type Register, and Reset Fatal Flag In Violation Type Register. Resetting the Violation Type Register is similar to a hardware reset in that it clears this register and returns the internal control of the MMU to an initial state (as if no violation had occurred since system initialization). Resetting the SWW flag or the FATL flag in the Violation Type Register clears these flags.

Two other commands are similar to reset commands in that they have no data associated with them. These are Set All CPU-Inhibit Flags in the segment attribute fields and Set All DMA-Inhibit Flags in the segment attribute fields, both of which cause all segment descriptors in the MMU to have the CPUI or DMAI flags set, respectively. These two set commands can be useful in initializing address translation tables or when swapping between tasks. For example, when swapping between tasks the Set All CPUI Flags command automatically makes the previous task's segments inaccessible to the next task, unless the system explicitly initializes the segment attribute field in these segments.

As an example of using the Special Output instruction SOUT to control an MMU, consider resetting the fatal flag of MMU #1. The MMU command opcode for this is "%14" (% denotes hexadecimal). The assembler syntax for the SOUT instruction is "SOUT destination field, source field" so that the instruction to reset the fatal flag of MMU #1 is "SOUT %1402, R0." Specifying register 0 in this instruction is an arbitrary choice—the content of this register is placed on the A/D lines during the data phase of the SOUT instruction, but it is ignored by the MMU. The low-order byte of the command (the destination field of the instruction) encodes which MMU is to reset its fatal flag. The convention followed in this paper is that MMU i is specified by setting bit i in the low order byte of the command. (Bit 1 set is hex "%02.")

The rest of the MMU commands consist of both operation and data. The following internal registers can be read or written: the Mode Register, the Segment Address Register, the Descriptor Registers and the Descriptor Selection Counter Register. A Descriptor Register can be read or written as a whole, or selected subfields can be accessed. In addition, by using the auto-increment feature of the Segment Address Register, successive Descriptor Registers can be accessed, or a selected field within successive Descriptor Registers can be accessed. For example, one Special I/O command in block mode could read a number of segment attribute fields. This is useful in determing which segments have been modified.

As an example of using the Special Output instruction SOUT to write data into an MMU, consider writing the contents of Register 6 into the Mode Register of MMU #2. The opcode for this command is "%00" and so the command is "SOUT %0004, R6." Here the high-order byte of the destination field contains the opcode and the low-order byte has bit 2 set (hexadecimal 4 if 0100 in binary) indicating MMU #2.

Commands to the MMU (Continued)

Certain MMU internal registers can only be read—there is no corresponding write instruction. This is because these registers contain information relating to a detected violation and thus it is not necessary to be able to write into these registers. These registers are the Violation Type Register, the Violation Segment Number Register, the Violation Offset Register, the Instruction Segment Number Register, the Instruction Offset Register and the Violation Bus Status Register. Although the Violation Type Register cannot be written, it should be noted that it can be cleared and that two of its flags can be individually cleared: the SWW flag and the FATL flag.

Direct Memory Access

DMA operations may occur between Z8001 machine cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal Mode of operation. For each memory access, segment attributes are checked and—if a violation is detected—a Suppress signal is generated. Unlike a CPU violation, which automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per-memory-access basis. The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA (hence warning conditions are not signaled). There are no trap requests because the CPU would not acknowledge the request until the end of the DMA cycle.

At the start of a DMA cycle, the DMASYNC line must go Low, indicating to the MMU the beginning of a DMA cycle. A Low DMASYNC inhibits the MMU from using an indeterminate segment number on lines SN_0-SN_6. When the DMA logical memory address is valid, DMASYNC must be High on one rising edge of Clock and the MMU then performs its address-translation and access-protection functions. Upon the release of the bus at the termination of the DMA cycle, DMASYNC must again be High. After two clock cycles of DMASYNC High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two clock cycles after the CPU regains bus control. During CPU cycles, DMASYNC should always be High.

Hardware and Software Reset

The MMU can be reset by either hardware or software mechanisms but note that they have different effects. A hardware reset occurs on the falling edge of the Reset input; a software reset is performed by an MMU command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low while Reset is Low the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the A/D and A lines are 3-stated. The $\overline{SUP}$ and $\overline{SEGT}$ open-drain outputs are not driven. If the Master Enable Flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with Chip Select.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state as if no violations or warnings had occurred.

Multiple-MMU Configurations

Z8010 MMU architecture supports system configurations that use more than one MMU. Multiple MMU devices can be used either to manage 128 CPU segments rather than the 64 supported by one MMU, or to manage multiple translation tables.

The Z8001 CPU generates logical addresses that can specify up to 128 different segment names. Because the MMU contains only 64 Segment Descriptor Registers, two MMUs are needed to perform address translation for 128 logical segments. Systems designed with only one MMU device still have the power and flexibility offered by memory management, although tasks in such a system are restricted to manipulating only 64 logical segment names. These names must either be 0 through 63 or 64 through 127. If the MMU in a single-MMU configuration is set to translate segment names in one range and the CPU generates a logical segment name in the other range, the MMU does not perform address translation and no physical memory location is output. In this case, no request is made to memory. Therefore, a single-MMU configuration should have additional external logic to detect erroneous segment names and generate a Segment Trap and Suppress signal.

The Upper Range Select flag (URS) is used in multiple MMU configurations to indicate which group of logical segment names

Multiple-MMU Configurations (Continued)

are to be translated by an MMU. When this flag is set, the Segment Descriptor Registers in the MMU are used in translating logical addresses in the range 64 through 127. When the flag is clear, the range is 0 through 63. Thus the URS flag corresponds to the most significant bit (bit 6) in the logical segment names that the MMU translates. Because this flag is under program control, the range of logical segment names can be changed during execution in System Mode.

MMU architecture also supports multiple segment translation tables. This feature is useful when separate tables are maintained for different tasks. Each task has its own table and switching between tasks requires enabling the appropriate MMU devices. In contrast, systems with only one translation table must either restrict the logical segment names that an individual task can use, or change the Descriptor Register entries whenever tasks are swapped. Two flags in the Mode Register, together with the N/$\overline{S}$ signal, are used in multiple table configurations.

The Multiple Segment Table (MST) flag indicates whether the configuration is being used to support multiple tables. When this flag is set, the MMU will compare the N/$\overline{S}$ line against the Normal Mode Select Flag (NMS) before generating a physical memory location on its Address lines. When the line and the flag match (both asserted or both de-asserted), the MMU is enabled and an address translation is performed (assuming the URS flag matches the most significant bit in the logical segment name). If the N/$\overline{S}$ line fails to match the state of the NMS flag, no translated address is generated by the MMU. The MST flag and the NMS flag are under program control and can be changed in System Mode.

The simplest multiple translation table configuration has one table for Normal Mode access and one for System Mode access. In such a configuration, the Multiple Table Flag is set in all MMUs and the N/$\overline{S}$ line of each MMU receives its input from the N/$\overline{S}$ output of the Z8001 CPU. MMUs containing descriptors of system segments have the NMS flag clear, and those containing descriptors to be used in Normal Mode have the flag set. When the Z8001 is in System Mode, the N/$\overline{S}$ line is Low and it matches the NMS flag in those MMUs whose Descriptor Registers contain system segment information. Therefore, these MMUs are used in address translation for system references.

When the Z8001 is in Normal Mode, the N/$\overline{S}$ line is High and it matches the NMS flag in those MMUs whose Descriptor Registers contain user segment information. Consequently, these MMUs are used in address translation for user segments. In this configuration, system segments are separated from user segments. When the Z8001 changes from Normal to System Mode of operation, the appropriate translation table is automatically selected. A more elaborate example of a configuration with multiple translation tables is given in the next section.

Examples

This section describes two Z8001-Z8010 configurations: one contains two MMUs and one address translation table; the other contains seven MMUs and four address translation tables. These examples are given in sufficient detail to illustrate some of the major ideas in constructing memory-management systems around the Z8010 MMU. High-level block diagrams illustrate some of the major features of typical hardware configurations and short programs illustrate software techniques for using the MMU.

The first example system is the two-MMU configuration illustrated in Figure 10. The two MMUs are called MMU #1 and #2, and they are selected during a command cycle by AD_1 and AD_2 being Low, respectively. Since a Special I/O instruction is being used bit 0 must always be zero. Thus, when a low-order byte of a command is "%02," MMU #1 responds; when it is "%04," MMU #2 responds; and when it is "%06," both MMUs respond. (Note that AD_1 is inverted before attachment to the $\overline{CS}$ pin.)

The A/D_1 line, which controls MMU #1 through the Chip Select input, is first combined with the Reset line. This allows the Master Enable Flag to be set upon system initialization, so the logical addresses generated by the CPU are passed to the physical memory. This is done because—upon reset—the mode register is otherwise cleared, the Translate Flag is clear and addresses pass through the MMUs untranslated. The bootstrap program can therefore reside in absolute memory locations in the physical memory. If the Reset line is not an input to the Chip Select line, the Master Enable Flag would not be set during system initialization and the CPU would not be able to address memory through the MMUs.

Note that there is a direct path from the CPU and DMA to the system bus. This path is used during I/O and memory refresh because the MMUs are quiescent during these cycles. It is also used for data on memory reads and writes. Also, note that the Suppress line goes both to the memory, where it can be used to protect the memory from erroneous

Examples (Continued)

writes, and back to the DMA device to save information upon the event of a DMA access error.

Of further interest in the example, address latches are used to buffer addresses between the Z8001 and a demultiplexed bus. This is required to demultiplex the address and data onto the bus. The address latch for AD_8–AD_{15} may not be needed if the I/O device does not use separate address and data lines.

A detailed example indicates how such a system could be used. First, consider setting Segment Descriptor Register 65 to point to a read-only segment of 768 bytes starting at memory location %115200. The segment is to be accessed in Normal Mode. The Descriptor Register should be %115202 01. The first two bytes, %1152, indicate the starting location of the segment (note that the low-order byte of the memory address is all zeros and is not stored in the Descriptor Register). The third byte, %02, indicates that three blocks of 256 bytes have been allocated to this segment. The fourth byte, %01, indicates that only the read-only segment flag has been set.

To write this descriptor into the MMU, a copy of the descriptor should be created in primary memory and a Special I/O block transfer instruction used. The SOTIRB instruction can be used for this.

This instruction has the assembler syntax "SOTIRB destination, source, count register" where both the destination and source are registers. The destination register contains the command to the MMU, the memory location pointed to by the source register contains the first byte of the data to be transferred, and the Count Register contains the number of bytes to be transferred.

The opcode to load the Descriptor Register is "%0B". Segment Descriptor Register 65 is Segment Descriptor Register 1 of MMU #2, so the MMU command is "%0B04".

To specify which Segment Descriptor Register to write, it is necessary to load the Segment Address Register of MMU #2 with 1. The MMU opcode to do this is "%01" and so the command is "%0104." The segment number (in this case 65) is a parameter to the example routine, passed in register 0. The

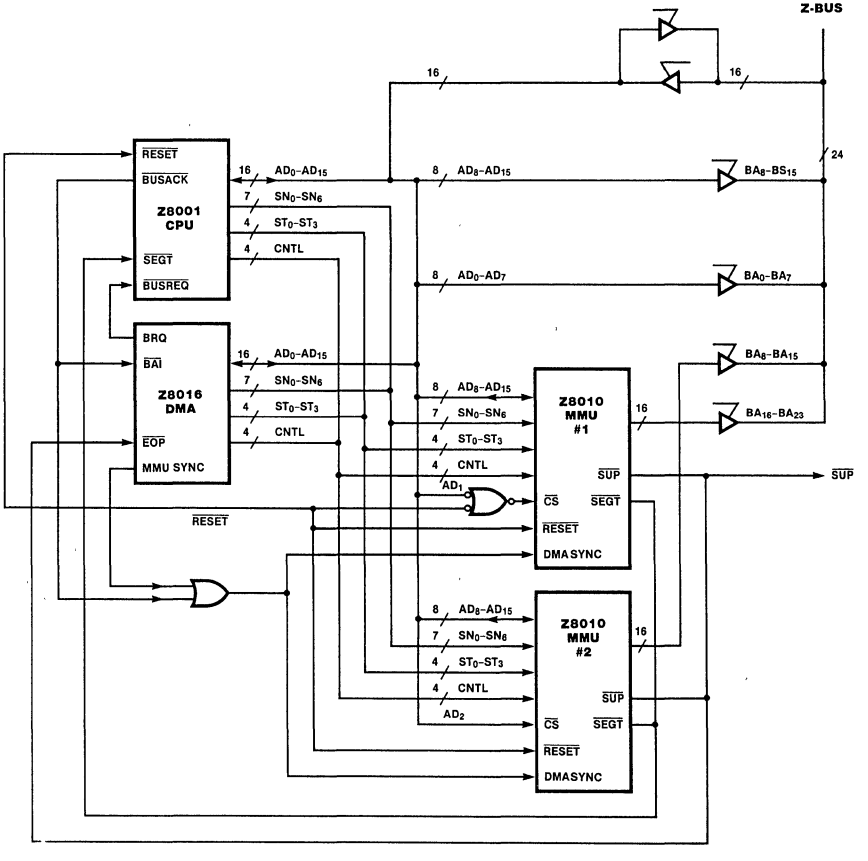


Figure 10. A Dual-MMU Configuration

Examples
(Continued)

```
        BIT     R0, #6          !Test to see if Descriptor Register is in MMU #1!
        JR      Z, OVER         !or MMU #2!
        SOUTB   %0104, RH0      !Set SAR in MMU #2!
        LD      R1, #%0B04      !Prepare to write descriptor!
        JR      NEXT

OVER:   SOUTB   %0102, RH0      !Set SAR in MMU #1!
        LD      R1, #%0B02      !Prepare to write descriptor!

NEXT:   LD      R0, #4          !Load count field—4 bytes!
        SOTIRB  @R1, @RR2, R0   !Write descriptor!
```

descriptor to be written is another parameter to this routine: RR2 contains the address in memory where this information resides. The SOUTB instruction has a similar syntax to the SOTIRB instruction explained previously except that it writes one byte instead of a series of bytes, and the destination I/O address is in the instruction itself instead of in a register specified by the instruction.

The routine on this page initializes the Segment Descriptor. Its parameters are found in Register R0, which contains the segment number to be written, and in Register RR2, which points to the descriptor information in primary memory. Registers R0 through R3 are used by this routine.

Now suppose that the user tries to write into location <<65>>%9328. This causes a segment trap both because of the write to a read-only segment and because the access exceeds the segment limit. At the end of the instruction that has the illegal memory access, the CPU acknowledges the trap. During the trap acknowledge cycle, MMU #2 asserts AD_{10} (assuming its ID field is "010") and this information is placed on the system stack for the trap-handling routine.

The trap-handling routine reads the violation information registers from the MMU. The violation type register contains "%05" indicating both a length violation and a read-only violation. The Violation Bus Status Normal Register contains "%28". The first nibble indicates a write in Normal Mode was in progress and the second nibble indicates a memory data access cycle was in progress. The violation segment register contains "%41" indicating segment 1 of MMU #2 caused the violation (which is segment number 65), and the violation offset register contains "%93" indicating the high-order byte of the logical address offset. The operating system can then issue an error message to the user indicating a read-only violation to segment 65. Using the program counter that was stacked when the segment trap was acknowledged, the system can also indicate the next instruction that was to be executed. Note that in this system the low-order byte of the violation offset is lost. This condition is corrected in the next example system.

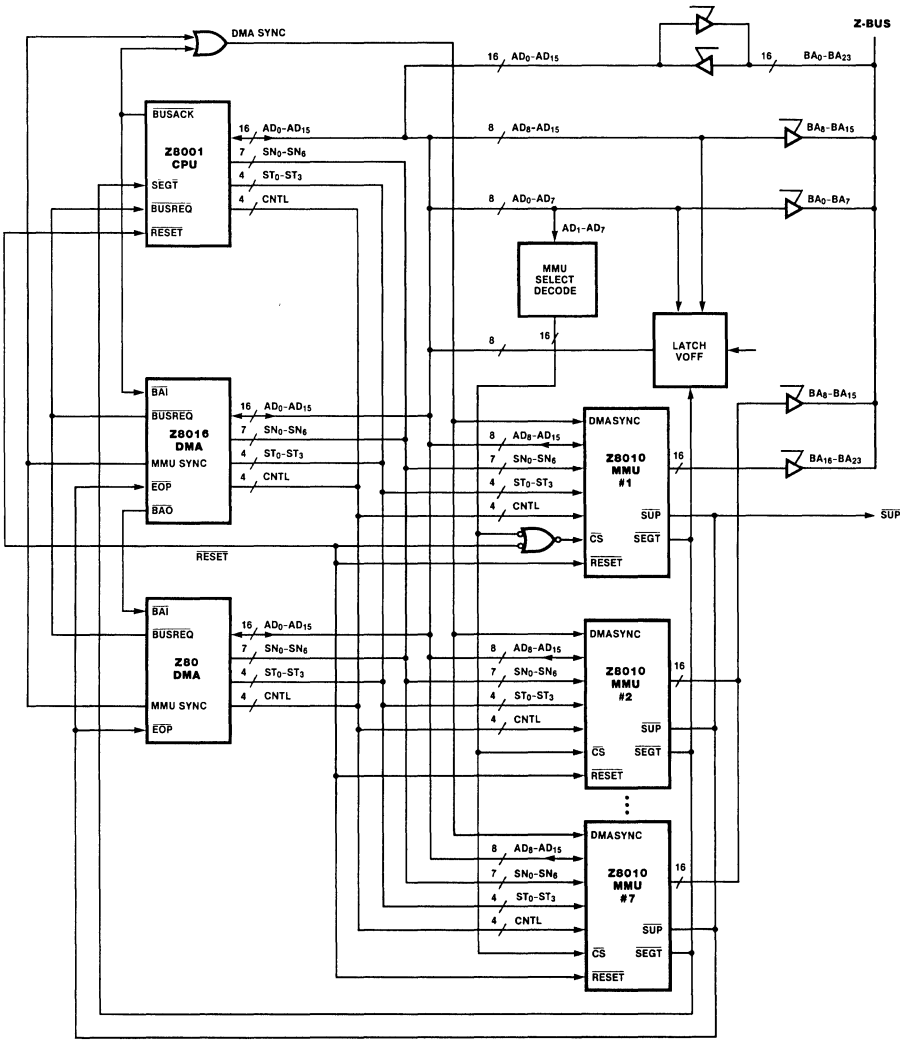


Figure 11. 16-MMU Configuration

Figure 11 gives a high-level diagram of the second system to be discussed. This configuration contains 16 MMUs, and the A/D lines select the appropriate MMU when in Command mode. The major innovation in this example, aside from the additional MMUs, is the latch that retains the least significant byte of an address offset when a violation is detected. This latch is enabled when a segment trap is generated by an MMU and holds the low-order byte of the address that generates an access violation.

In addition, external decoding logic for selecting one MMU Chip Select line is indicated. Seven MMUs is the limit in one configuration without additional decoding logic for selecting one MMU Chip Select line. (The reason why AD_0 cannot be used to control an eighth MMU is due to the Special I/O input convention of the CPU. When the CPU inputs a byte of information and AD_0 is asserted, the data is taken from AD_0–AD_7, which are not driven by the MMU.)

Switching Tables in a 16-MMU System. The 16-MMU configuration can support a memory management system designed with two MMUs permanently allocated to the operating system and the others allocated in pairs to different user tasks. Thus, seven user tasks can have translation tables resident in the 14-user MMUs, and switching between active tasks requires the appropriate MMUs to be enabled and disabled. This selection process can be effected by manipulating the Master Enable (MSEN) flags in the mode registers of the appropriate MMUs.

Examples (Continued)

The routine performs the selective enabling of MMUs required by a task swap. This routine disables all user MMUs (thus disabling the currently enabled user MMUs), then enables the appropriate pair. (The system pair is always enabled.) The code selecting the new task is passed in register R1; it contains %n, if task n is to be dispatched.

Two peculiarities of this example are worth noting. First, each user ID number corresponds to seven MMUs (for example, all upper-range user MMUs). The Segment Trap processing routine has to take this into account. Second, the Chip Select code is assumed to be as follows:

	AD_0-AD_7	MMU Selected
System:	02	#1 ID=0, URS=0
	04	#2 ID=1, URS=1
User 0:	08	#3, ID=2, URS=0
	10	#4, ID=3, URS=1
User 1:	18	#5, ID=2, URS=0
	20	#6, ID=3, URS=1
User 2:	28	#7, ID=2, URS=0
	30	#8, ID=3, URS=1
	.	.
		.
User 6:	68	#15, ID=2, URS=0
	70	#16, ID=3, URS=1

It is also assumed that %F8 will select all user MMUs.

```
CLR     R0              !Clear R0!
SOUT    %00F8,R0        !Disable all user MMUs by clearing their mode registers!
SLA     R1,#1           !Multiply R1 by 2—the number of bytes in a memory word!
LD      R1,TABLE(R1)    !Get the command word (opcode always %00) for user n,
                         URS=0!
LDA     RR2,DATA        !Get the new mode register bit pattern (%DA)!
SOUTIB  @R1,@RR2,R0     !Send %DA to lower-range MMU and increment RR2 to
                         DATA +1!
INC     R1, #8          !Command word for URS=1!
SOUTIB  @R1,@RR2,R0     !Send %FB to upper range MMU!
END:
DATA:   BYTES(%DA,%FB)  !Mode register bit patterns!
TABLE:  WORDS (%8,%18,%28,%38,%48,%58,%68)
```

Program to Switch Tables

MMU Command Summary

Opcode	Operation	Opcode	Operation
00	Read/Write Mode Register	0C	Read/Write Base Field And Increment SAR
01	Read/Write Segment Address Register	0D	Read/Write Limit Field And Increment SAR
02	Read Violation Type Register	0E	Read/Write Attribute Field And Increment SAR
'03	Read Violation Segment Number	0F	Read/Write Descriptor And Increment SAR
04	Read Violation Offset (high byte)	10	Reserved
05	Read Bus Cycle Status Register	11	Reset Violation Type Register
06	Read Instruction Segment Number	12	Reserved
07	Read Instruction Offset (high byte)	13	Reset SWW Flag In VTR
08	Read/Write Base Field In Descriptor	14	Reset FATL Flag In VTR
09	Read/Write Limit Field In Descriptor	15	Set All CPU-Inhibit Flags
0A	Read/Write Attribute Field In Descriptor	16	Set All DMA-Inhibit Flags
0B	Read/Write Descriptor (all fields)	17-1F	Reserved
		20	Read/Write Descriptor Selector Counter Register
		21-3F	Reserved

Zilog

Military Specification Standards

March 1981

General Description

Zilog offers high-reliability versions of the entire family of Z80 and Z8000 logic circuits, processed in accordance with the requirements of MIL-STD-833 level B (Test Methods and Procedures for Microelectronics). In addition, the Z80 CPU and the Z80A CPU are included as part of MIL-M-38510 (General Specification for Microcircuits) in 1980, with the remaining devices scheduled for inclusion in 1981. Each of the Zilog devices will become military-qualified as soon as the detailed specifications are released.

General Considerations. Zilog high-reliability microcircuits are designed to meet the full military temperature range of -55°C to +125°C and are packaged in hermetic dual-in-line packages. These packages can reliably withstand the thermal shock requirements of MIL-STD-833, method 1011, Condition C (-65°C to +150°C). For industrial users, Zilog offers an extended operating temperature range of -40°C to +85°C. All of Zilog's high-reliability microcircuits receive 1005 processing in accordance with the requirements of MIL-STD-833 level B or C (as specified). Table 1 lists the screening tests performed on the two levels. An X indicates that the test is performed 100% of the time, an S indicates that testing is done on a sample basis, and a Z indicates that the test can be done upon request. Table 2 lists the Zilog products available with the 100% testing process shown with X's in Table 1.

Test	Condition	MIL-STD-883 Method	MIL-STD-883 Condition	Screening Level B	Screening Level C
SEM Inspection	—	2018	—	Z	Z
Precap Visual	—	2010	B	X	X
Seal and Lot I.D.	—	—	—	X	X
Stabilization Bake	48 hrs. @ 150°C	1008	C	X	X
Temperature Cycling	10 cycles	1010	C	X	X
Centrifuge	Y_1 Plane	2001	E	X	X
Fine Leak	—	1014	A	X	X
Gross Leak	—	1014	C	X	X
Electrical Test	Per Zilog Data Sheets	—	—	X	X
Burn-In	168 hr.	1015	Done	X	—
	240 hr.	1015	Done	Z	—
Final Electrical	25°C, -55°C,	—	—	X	X
	and +125°C	—	—	X	S
Radiographic Inspection	1 view	2012	—	Z	Z
External Visual	—	2009	—	X	X

NOTES: S = Sample testing only, X = 100% testing, Z = Optional (tested if requested).

Table 1. Total Lot Screening

General Description (Continued)

Product	Speed	Mil Temp Range	Extended Temp Range	Planned JAN
Z80 CPU	2.5 MHz	Yes	Yes	Early 1981
Z80A CPU	4.0 MHz	Yes	Yes	Early 1981
Z80 PIO	2.5 MHz	Yes	Yes	Mid 1981
Z80A PIO	4.0 MHz	Yes	Yes	Mid 1981
Z80 SIO	2.5 MHz	Yes	Yes	Mid 1981
Z80A SIO	4.0 MHz	Yes	Yes	Mid 1981
Z80 DMA	2.5 MHz	Yes	Yes	Late 1981
Z80A DMA	4.0 MHz	Yes	Yes	Late 1981
Z80 CTC	2.5 MHz	Yes	Yes	Late 1981
Z80A CTC	4.0 MHz	Yes	Yes	Late 1981
Z8001 CPU	4.0 MHz	Yes	Yes	Late 1981
Z8002 CPU	4.0 MHz	Yes	Yes	Late 1981

NOTE: See Ordering Information for package and temperature designators.

Table 2. High-Reliability Products Available

Manufacturing and Process Controls

Zilog high-reliability microcircuits are processed and assembled in accordance with the Zilog Product Assurance Program Plan, which conforms to the requirements of Appendix A of MIL-M-38510. The following are some of the items contained in the plan:

- A clear, concise procedure for converting a customer specification to a Zilog internal specification, assuring the customer that parts received meet or exceed specified requirements.
- A formalized training and testing program for all operator and inspection personnel to ensure that each operation is performed correctly.
- An inspection system that includes a complete Incoming Inspection Laboratory, a Chemical Analysis Laboratory, and a Failure Analysis Laboratory to assure that all materials, utilities, and work-in-progress meet Zilog requirements.
- Rigid requirements for the cleanliness of work areas and the maintenance of a Class 100 environment at all stations where critical operations are performed.
- A document control system to control changes in design, materials, and processes.
- A system for maintaining documents and records in active files for three years and in archive files for ten years.
- An instrument maintenance and calibration system complying to the requirements of MIL-C-45662 (Calibration System Requirements).
- A quality audit system in accordance with MIL-Q-9858 (Quality Program Requirements).

Package Dimensions

Zilog

Zilog

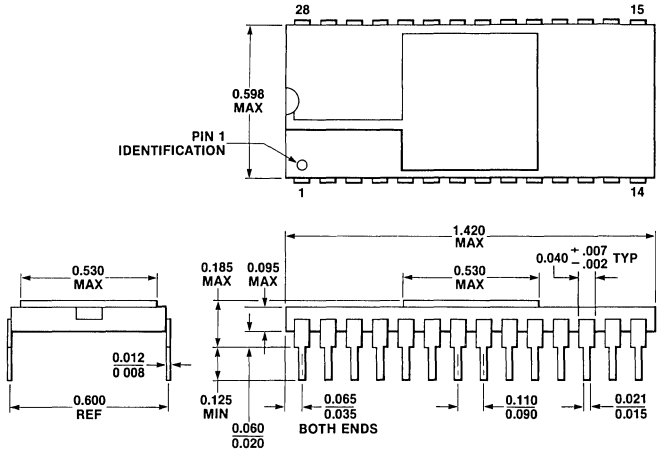
March 1981

Package Summary

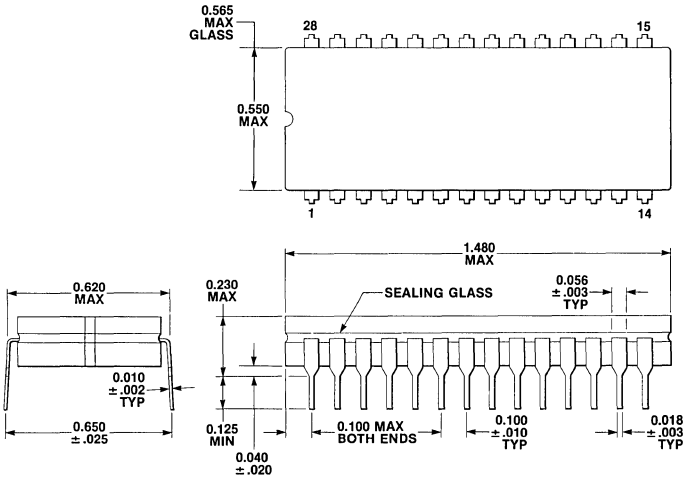
This table summarizes the microprocessor components available from Zilog by number of pins and package type. Following the table are detailed drawings for each package type. For further information on specific components, see the Ordering Information section of each product specification.

Pins	Package	Component
28	Ceramic, Cerdip, Plastic	Z6132 Quasi-Static RAM Z8430 Z80 CTC
40	Ceramic, Cerdip, Plastic	Z8002 Z8000 CPU Z8030 Z8000 Z-SCC Z8036 Z8000 Z-CIO Z8038 Z8000 Z-FIO Z8090 Z8000 Z-UPC Z8400 Z80 CPU Z8410 Z80 DMA Z8420 Z80 PIO Z8440 Z80 SIO/0 Z8441 Z80 SIO/1 Z8442 Z80 SIO/2 Z8449 Z80 SIO/9 Z8470 Z80 DART Z8530 SCC Z8536 CIO Z8538 FIO Z8590 UPC
40	Ceramic, Cerdip, Plastic (Continued)	Z8601 Z8 MCU Z8611 Z8 MCU Z8671 Z8 MCU Z8681 Z8 MCU
40	Protopack	Z8093 Z8000 Z-UPC Z8094 Z8000 Z-UPC Z8593 UPC Z8594 UPC Z8603 Z8 MCU Z8613 Z8 MCU
48	Ceramic, Plastic	Z8001 Z8000 CPU Z8010 Z8000 Z-MMU
64	Quip	Z8091 Z8000 Z-UPC Z8092 Z8000 Z-UPC Z8591 UPC Z8592 UPC Z8602 Z8 MCU Z8612 Z8 MCU

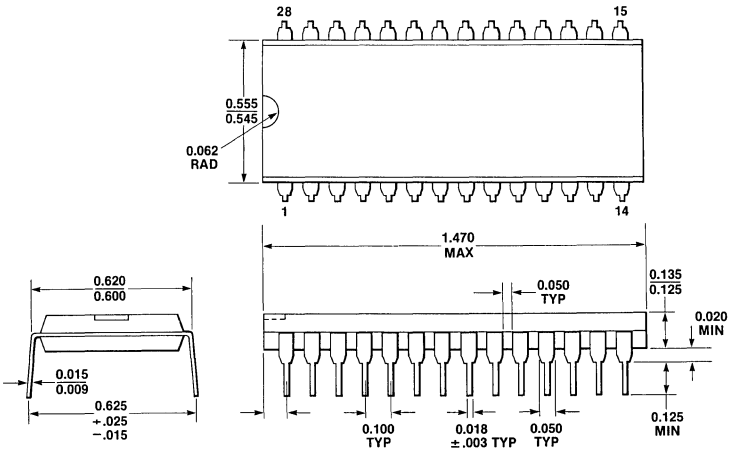
Package Dimensions



28-Pin Ceramic Package



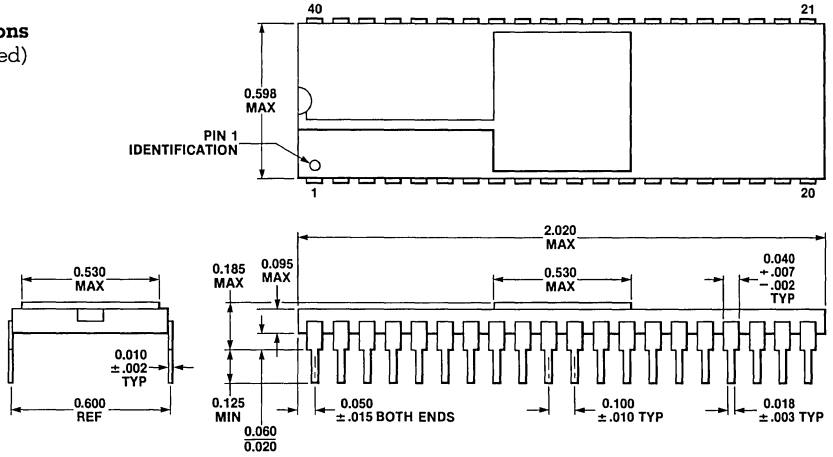
28-Pin Cerdip Package



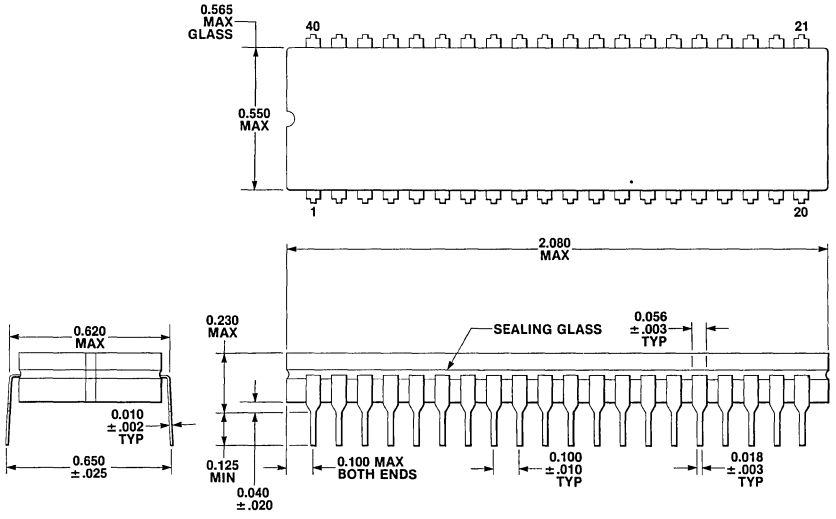
28-Pin Plastic Package

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

Package Dimensions (Continued)

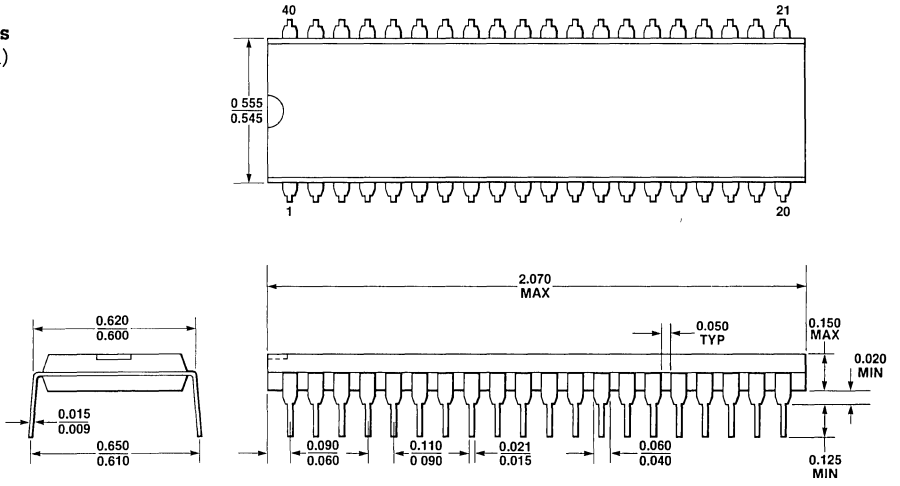


40-Pin Ceramic Package

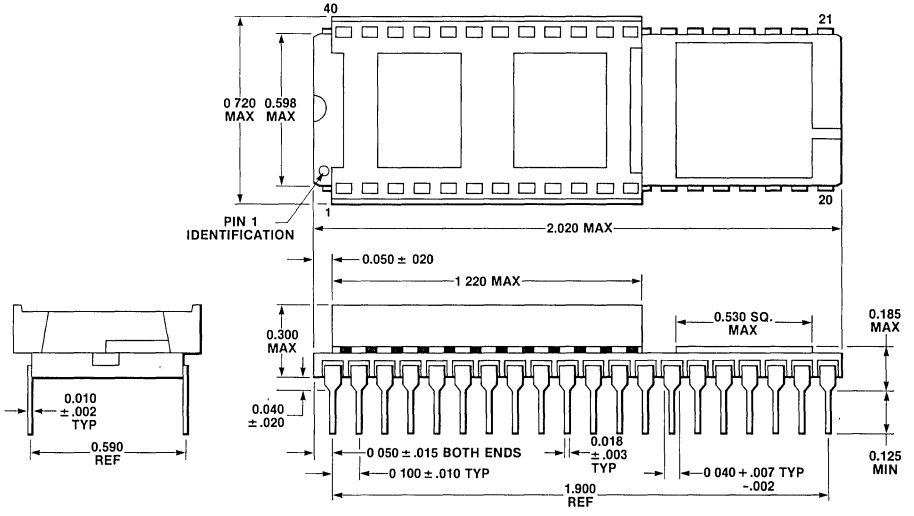


40-Pin Cerdip Package

Package Dimensions (Continued)

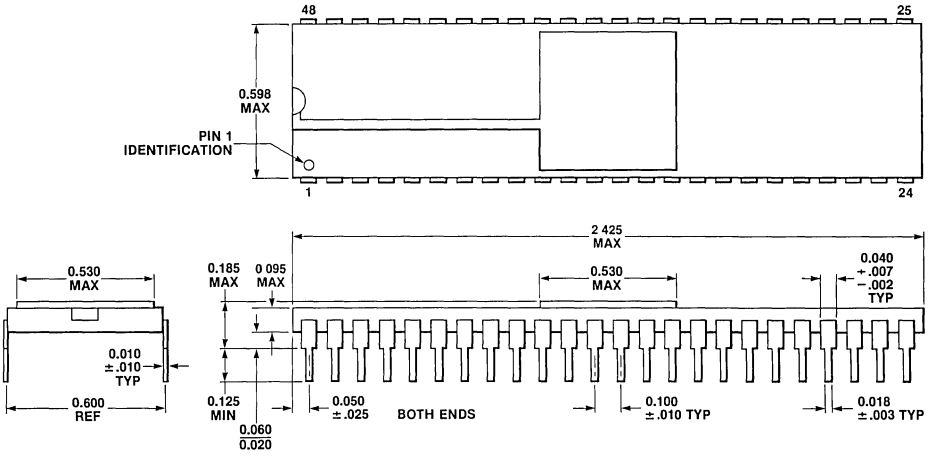


40-Pin Plastic Package

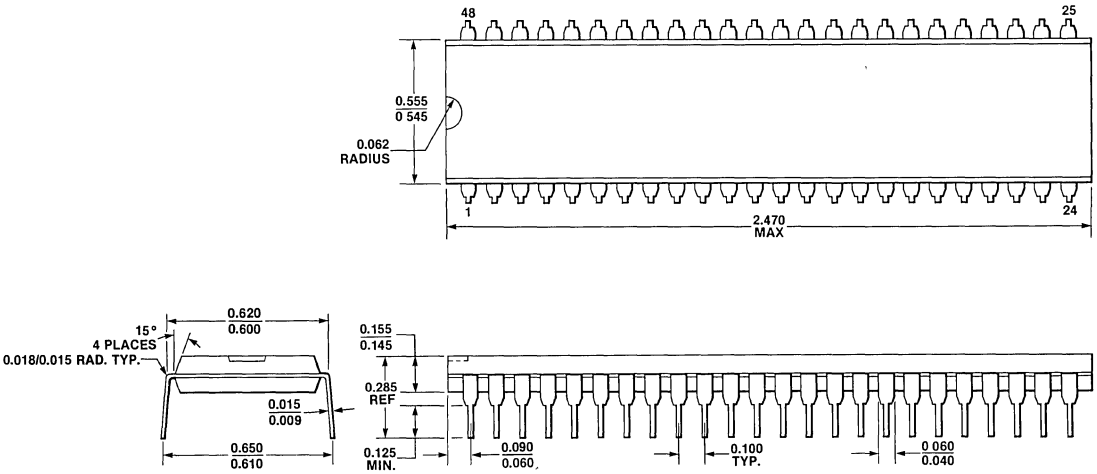


40-Pin Protopack Package

Package Dimensions (Continued)

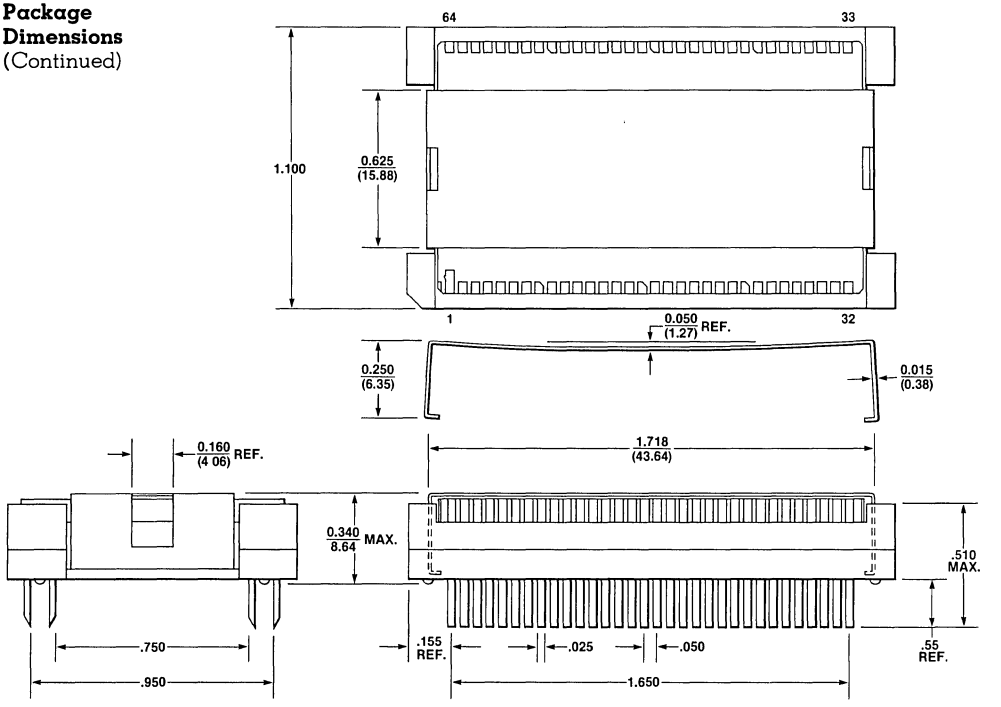


48-Pin Ceramic Package



48-Pin Plastic Package

Package Dimensions (Continued)



64-Pin Quip Package

Z80
Microcomputer Boards

Zilog

The Problem Solvers for Microcomputer Systems

The Z80 MCB family of bus compatible microcomputer boards features powerful performance and application flexibility at a low total systems cost. For every application, from a single-board solution to a high-performance board set, the MCB family provides the right combination to easily solve most microcomputer system problems.

Performance. The powerful architecture of the Z80 Central Processing Unit (CPU) is at the heart of the MCB family. The dual-register set of the Z80 CPU allows high-speed interrupt processing, context switching and other forms of foreground/background programming. Each register set includes an 8-bit storage register which can also be used as three 16-bit memory address or general-purpose registers. Two index registers provide greater memory addressing capability. A 16-bit external stack pointer permits unlimited subroutine nesting and temporary data storage. In addition, the CPU features vectored interrupts and supports dynamic memories requiring periodic refresh.

Economy. Because each Z80 microcomputer board provides a large number of functions within a convenient and compact size, implementing an MCB family solution requires fewer boards and less space than comparable alternatives. Fewer boards mean lower power consumption, lower-cost power supply, less heat generation and, therefore, lower cooling costs and greater economy in connector and other mechanical costs. Feature for feature, the MCB family adds up — a superior solution with unbeatable economy.

The Competitive Edge. The time it takes from product conception to market introduction may mean the difference between success or failure. Success is assured with the Z80 MCB family. The boards are compatible, can be integrated into a system quickly, are easy to learn and use, allow the convenient addition of last minute features, and are available off-the-shelf.

Proven Design. The MCB family has been used in hundreds of applications throughout the world, demonstrating reliability and performance day after day. All Zilog microcomputer boards undergo extensive burn-in with both pre and post burn-in testing to ensure constant performance and reliability.

Family Members. The Z80 microcomputer board family includes powerful CPU and memory boards as well as a variety of versatile, high-performance I/O expansion boards. The Z80 Microcomputer Board (MCB) is a complete single-board microcomputer with its own self-contained memory plus serial and parallel I/O ports. The Z80 Memory and Disk Controller (MDC) adds up to 48K bytes of system memory and interface for up to eight floppy disk drives. The Z80 Serial Interface Board (SIB) provides four high-performance serial interface channels to solve a variety of data communications problems. Analog interface is simplified with the Z80 Analog Input Board (AIB) or the Analog Input/Output (AIO) board—each provides up to 32 input channels and 12-bit resolution. Flexible, parallel I/O is provided by the Z80 Input/Output Board (IOB) with 64 I/O lines and a liberal amount of "wire-wrap" area to give the user a head start on special interface solutions. Memory expansion is easily handled by the Z80 RAM Memory Board (RMB). It contains both RAM (up to 64K bytes) and fixed memory socket area, while the Z80 PROM Memory Board (PMB) allows up to 32K bytes of non-volatile memory.

Make vs Buy. The make vs buy decision impacts both strategic and economic issues including new product introduction schedules, product reliability, test fixture design, resource allocation, spare parts inventory, field maintenance and many others. These issues all involve hidden costs and potential product development delays. When all costs are considered, it is often more economical to purchase, rather than manufacture, microcomputer boards.

Purchasing microcomputer boards for initial production quantities and later switching to in-house manufacture of these boards provides an effective compromise solution. Zilog supports this approach by licensing the manufacture of its microcomputer boards. The high front-end manufacturing costs can thereby be postponed until the success of the product is confirmed by market acceptance.

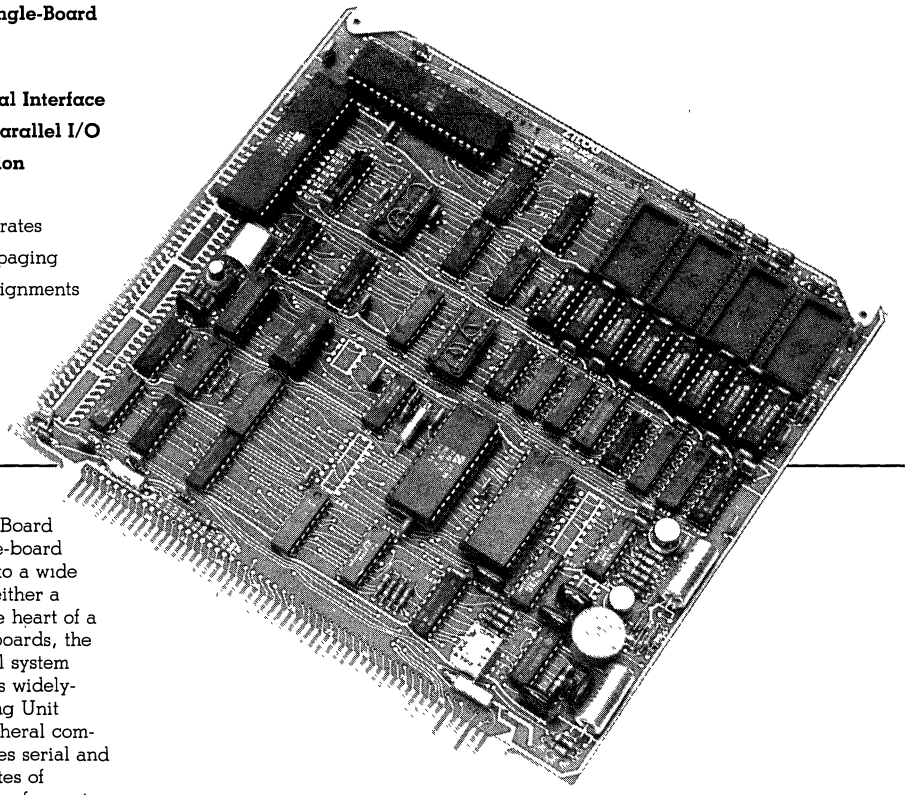
Z80® MCB
Z80 Microcomputer Board

Zilog

Product Description

March 1981

- **Complete, Powerful Single-Board Solution**
- **16K or 4K Bytes RAM**
- **Industry Standard Serial Interface**
- **Convenient, Flexible Parallel I/O**
- **Low-Power 5 V Operation**
- **Many User Options**
  - Programmable baud rates
  - Relocatable address paging
  - Variable I/O port assignments



OVERVIEW

The Z80 Microcomputer Board (MCB) is a complete single-board microcomputer adaptable to a wide range of applications. As either a stand-alone board or as the heart of a system of bus-compatible boards, the MCB provides the essential system functions. Built from Zilog's widely-used Z80 Central Processing Unit (CPU) and other Z80 peripheral components, this board provides serial and parallel I/O, 4K or 16K bytes of dynamic RAM and provision for up to 4K bytes of E/P/ROM all on a compact 7.7 × 7.5 in. circuit board.

All address, data and control lines are fully buffered to standard TTL levels for easy expansion with other boards in the Z80 MCB family. The MCB employs an on-board dc-dc converter to allow operation from a single +5 V power supply; the converter circuit generates the +12 V and −5 V necessary for the dynamic RAM array and −10 V for serial communication interface.

FUNCTIONAL DESCRIPTION

Central Processing Unit. The MCB is controlled by the Z80 CPU with 158 instructions including 16-bit arithmetic, block moves and block I/O, bit manipulation and versatile addressing modes. This powerful set of instructions provides programming ease and, for convenient portability, contains all 8080 instructions as a proper subset. The CPU has an operating frequency of 2.457 MHz derived from a 19.6608 MHz system clock and is able to execute instructions as fast as 1.6 μs.

The CPU has a powerful and versatile vectored interrupt capability which allows identification of up to 128 unique interrupt service subroutines without additional hardware. See the *Z80 CPU Product Specification* for additional information.

Memory—RAM Array. The MCB includes a dynamic Ramdom Access Memory (RAM) array of either 4K or 16K bytes. A unique refresh register in the CPU sends a new refresh address to the memory array after each op code fetch; therefore, automatic refresh is transparent and no wait states are imposed. This manner of memory refresh removes all the disadvantages of dynamic memory while still retaining economy and speed performance.

The addressable memory space may be located at any 4K byte boundary by changing the position of two jumpers on the board. Systems requiring additional fixed memory, such as the Z80® PROM Memory Board (PMB) can thereby obtain a large block of continuous address space starting at zero. This same memory paging scheme generates a RAM SELECT signal routed to the array by a pair of connectors. Thus, external hardware may be used to disable the memory for bank selection. *Figure 1* shows the memory addressing for the MCB/4 and MCB/16.

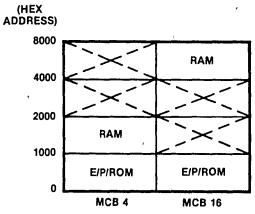


Figure 1. Memory Addressing for MCB/4 and MCB/16

Memory—E/P/ROM Array. The MCB includes four 24-pin sockets that can accommodate up to 4K bytes of non-volatile memory. The type of memory device to be used—Erasable Programmable Read Only Memory (EPROM), Programmable Read Only Memory (PROM) or Read Only Memory (ROM)—can be selected by changing the jumper wires. Although the MCB dc-dc converter generates the voltages required by P/ROM arrays, it cannot deliver sufficient current from these outputs to drive EPROM devices. When 2708 or 2704 EPROMs are used, external supplies must provide the required voltages. This option is easily implemented by selecting the appropriate jumpers on the board. *Table 1* lists devices that can be used in these sockets. The standard board configuration is for the 2708.

Non-Volatile Memory	Device Number	
MOS E/PROM	2704	8704
	2708	8708
	2716	2316
Bipolar P/ROM	6341	
	6381	
	82S181	
	82S191	

Table 1. Non-Volatile Memory Devices

As with the RAM array, addressing is designed to allow the user to relocate the E/P/ROM array to any 4K byte boundary within the address range of the CPU. A ROM SELECT output signal and corresponding input contacts on the edge connector allow the user to implement shadow E/P/ROM or select an alternate PROM set.

Counter-Timer. The Z80 CTC contains four independent 8-bit counter channels which can be programmed by system software for a broad range of counting and timing applications. One of the four channels is used as a baud-rate generator for serial interface; the additional channels can be used to satisfy other system requirements.

Each of the four channels may be decremented either from an external input in the counter mode or from a prescaled version of the system clock. Upon reaching zero, a pulse is available from three of the channels and interrupts may be generated by all four channels if they are programmed to do so. The device will supply an interrupt vector indicating which channel is causing the interrupt. The four independent input lines are each available on a separate position of the edge connector. The input signal may serve as a positive or negative trigger for the timer mode or as the actual event to be counted. Each output may be used as the input or trigger to a subsequent channel in order to achieve long time delays.

If an external device must cause an interrupt to indicate a status change, one channel of the CTC can be used as a vectored interrupt generator by programming in a time constant of 1 and driving the input trigger with a transition signal from the external device. Thus, when no other parallel data need to be transferred, interrupts can occur without using the PIO strobe line.

The output of channel 1 serves as the transmit and receive clock for the USART, providing a convenient way to implement software programmable baud rates. This signal is routed to the edge connector of the board and is returned on a separate contact. Consequently, channel 1 of the CTC may be used as either the USART clock or in the user's application, depending on edge connector wiring. See the *Z80 CTC Product Specification* for details.

I/O Capability. The MCB provides both parallel and serial I/O via a Counter-Timer Circuit (CTC), Parallel Input/Output (PIO) device and a Universal Synchronous/Asynchronous Receiver/Transmitter (USART). These devices occupy eleven locations of port-assigned I/O space as shown in *Table 2.* Jumper options allow relocation of the I/O devices within the port-assigned address space.

MCB I/O PORT ASSIGNMENTS

FUNCTION	PORT
CTC Channel 0	D4
CTC Channel 1	D5
CTC Channel 2	D6
CTC Channel 3	D7
PIO Port A Data	D8
PIO Port B Data	D9
PIO Port A Control	DA
PIO Port B Control	DB
Switch Register	DD
USART Data	DE
USART Status/Control	DF

Table 2. MCB Port Assignments

Serial I/O. A serial data communication channel provides support for either asynchronous or synchronous data transfer with either half- or full-duplex signaling. Driver and receiver devices are included to provide RS-232C compatible interface to passive 20 mA equipment simply by relocating two jumpers and attaching the serial line to the appropriate locations on the edge connector.

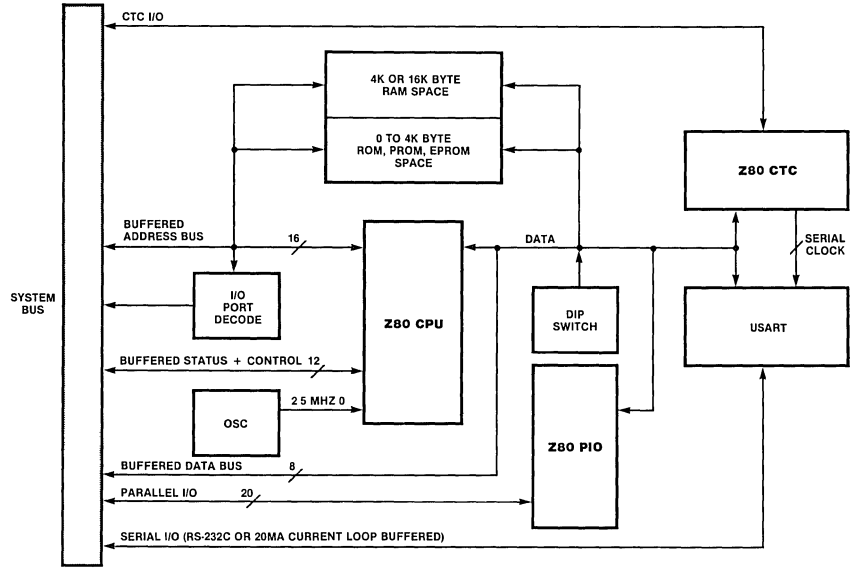
Although the 8251 USART is designed for polled operations, it is possible to utilize the mode 2 interrupt structure of the CPU by coupling the transmitter ready and receiver ready lines from the USART to the input lines of the parallel I/O device. The baud-rate clock is derived from the 19.6608 MHz crystal oscillator and channel 1 of the CTC device. This allows baud-rate selection under program control as shown in *Table 3.*

BAUD RATE	TIME CONSTANT	
50	96	
75	64	
110	44	
150	32	
200	24	
300	16	
600	8	
1200	4	
2400	2	
4800	1	
9600	4	Counter Mode
19200	2	Counter Mode
38400	1	Counter Mode

Table 3. Programmable Baud Rates for Serial I/O

As an alternative to the on-board clock, user-selected jumpers allow independent transmit and receive clocks from external sources to be applied directly to the USART. A single external clock operating at twice the desired frequency may be applied to the on-board wave-shaping flip-flop, thus providing a clean, reliable clock signal.

Parallel I/O. The Z80 PIO contains two independent 8-bit parallel I/O ports. It can be configured by the CPU to operate in any of four major modes—input, output, bidirectional or control. Data direction characteristics can be programmed individually or in byte configuration. Each byte has two independent handshake lines for completely asynchronous data transfers with any general-purpose interface. To allow maximum flexibility for the user, the 16 PIO data lines and four handshake lines are totally uncommitted. Also, four 16-pin IC sockets may be wired to accept any necessary logic device or terminator package. See the *Z80 PIO Product Specification* for details.



Z80® MCB Block Diagram

SPECIFICATIONS

Processor
Zilog Z80 CPU

Operating Frequency
2.5 MHz

RAM Array
MCB/4 4K × 1 RAMs, t_{AC} = 250 ns
MCB/16 16K × 1 RAMs, t_{AC} = 250 ns

E/P/ROM Sockets
Four 24-Pin Sockets

E/P/ROM Types
E/PROM 2704, 2708 or Equivalent
P/ROM 6341,6381,82S181,82S191 or Equivalent

Serial I/O Channels
1 Channel — RS232C or 20 mA Current Loop

Serial Modes
Synchronous or Asynchronous

Data Rates
50 to 38.4K Baud

Parallel I/O Lines
16 Lines with 4 Handshake Lines

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @ 2 A (max)
(with 3 PROMs)

Environmental

Temperature	0 to 50°C
Humidity	0 to 90% noncondensing

Physical

Height	7.5" (191 mm)
Width	7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6009-01	MCB/4 Z80 Microcomputer Board with 4K bytes RAM
05-6009-02	MCB/16 Z80 Microcomputer Board with 16K bytes RAM
05-6009-19	MCB/16 Z80 Microcomputer Board with 16K bytes RAM for use with RIOTM operating system software

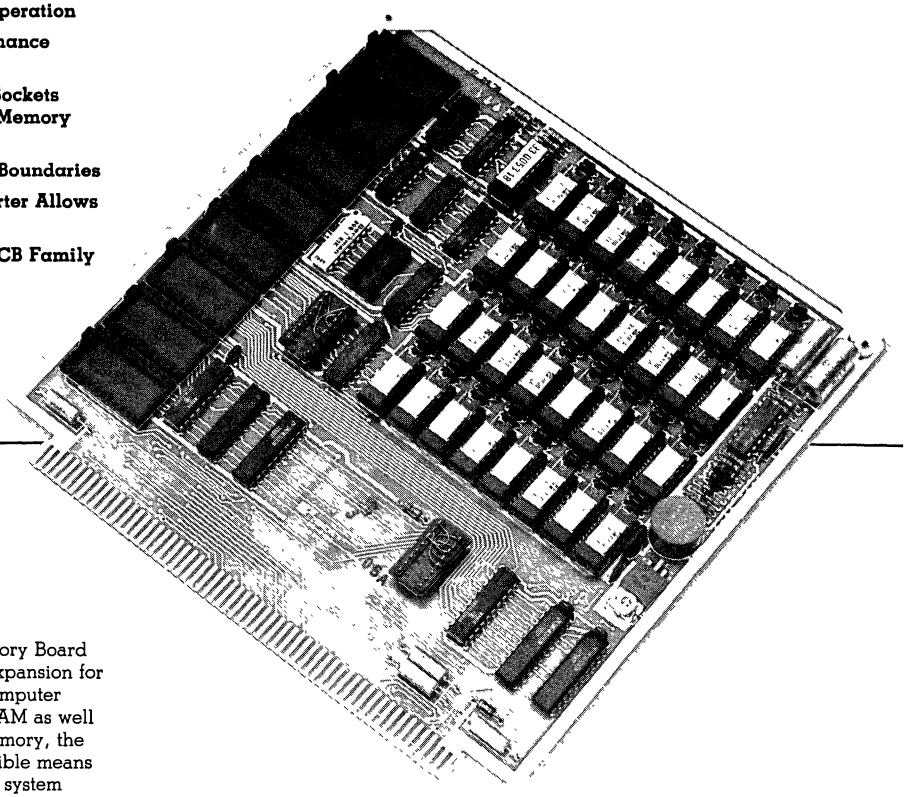
Z80® RMB
Z80 RAM Memory Board

Zilog

Product Description

March 1981

- **Automatic Refresh by CPU for Simple, Fast System Operation**
- **Low-Cost, High-Performance Dynamic Memory**
- **8K Bytes of E/P/ROM Sockets Available for Flexible Memory Arrangement**
- **User-Selected Address Boundaries**
- **On-Board dc-dc Converter Allows Low-Power Operation**
- **Compatible with All MCB Family Microcomputer Boards**



OVERVIEW

The Z80 RMB RAM Memory Board provides system memory expansion for the MCB family of microcomputer boards. Containing both RAM as well as sockets for E/P/ROM memory, the RMB board provides a flexible means of implementing additional system memory. Each board contains a dc-dc converter that generates +12 V and −5 V bias voltages, thereby allowing operation from a single +5 V system power supply.

FUNCTIONAL DESCRIPTION

Address Map. The RMB memory address selection is completely compatible with the MCB microcomputer board. *Figure 1* shows the memory map for the RMB/16 and RMB/48.

Location of the memory array may be altered by the user. The RAM chip-select logic allows each 4K segment to have a starting address at any of 16 boundries within the 64K of addressable memory space. Chip selection is accomplished by using a PROM decoder to select the Row Address Strobe (RAS) signal to the appropriate bank of devices. This method of bank selection minimizes overall system power since only the selected bank dissipates active power. The address select PROM is socketed so that it may be easily replaced by the user for address reassignment.

PROM Sockets. The RMB contains eight 24-pin sockets that may be used for a variety of E/P/ROM devices. Through selection of appropriate jumpers the socket area can be configured to accept the device types shown in *Table 1.*

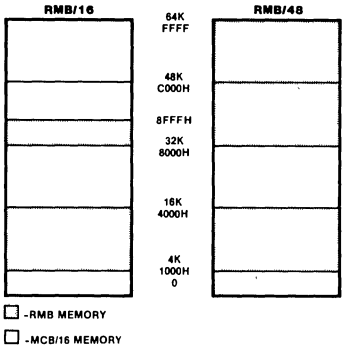


Figure 1. RMB Memory Map

Non-Volatile Memory	Device Number	
MOS E/PROM	2704	8704
	2708	8708
	2716	2316
Bipolar P/ROM	6341	
	6381	
	82S181	
	82S191	

Table 1. Non-Volatile Memory Devices

Chip selection is accomplished by means of a PROM decoder, supplied socketed and unprogrammed so that the user has complete flexibility in its application. When using EPROM devices the −5 V and +12 V requirements must be supplied from a source external to the board.

Refresh. Although dynamic RAMs are used, the RMB does not require any additional circuitry for refresh. Unique characteristics of the MCB CPU allow memory to be refreshed automatically and in a transparent mode. Following each op-code fetch, a new refresh address is available on the system address bus while the op-code is being decoded within the CPU. The CPU does not require wait states; therefore, there is no degradation of system performance (See *Figure 2*).

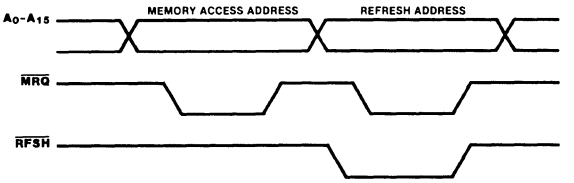
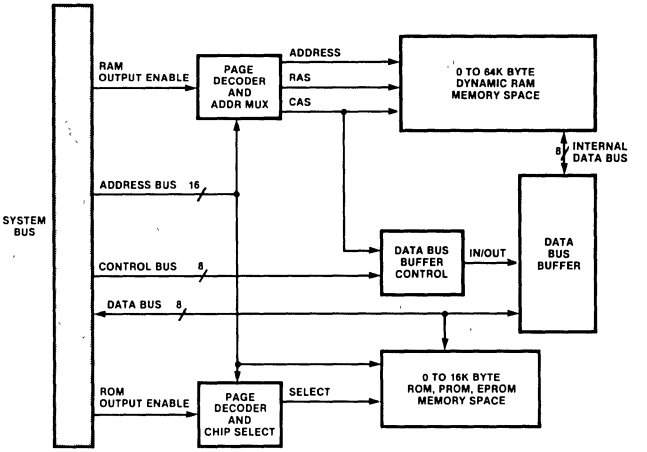


Figure 2. Automatic Refresh Generation



Z80® RMB Block Diagram

SPECIFICATIONS

Memory Capacity
Dynamic RAM 64K
E/P/ROM 16K

Memory Size
Standard Configurations
16K or 48K RAM

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @ 1.6 A (max)

DC-DC Converter Output
+12 V @ 320 mA (max)
−5 V @ 50 mA (max)

Environmental
Temperature 0 to 50°C
Humidity 0 to 90% noncondensing

Physical
Height 7.5" (191 mm)
Width 7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6003-02	Z80 RMB/16 16K RAM Memory Board
05-6003-03	Z80 RMB/32 32K RAM Memory Board
05-6003004	Z80 RMB/48 48K RAM Memory Board
05-6003-05	Z80 RMB/64 64K RAM Memory Board

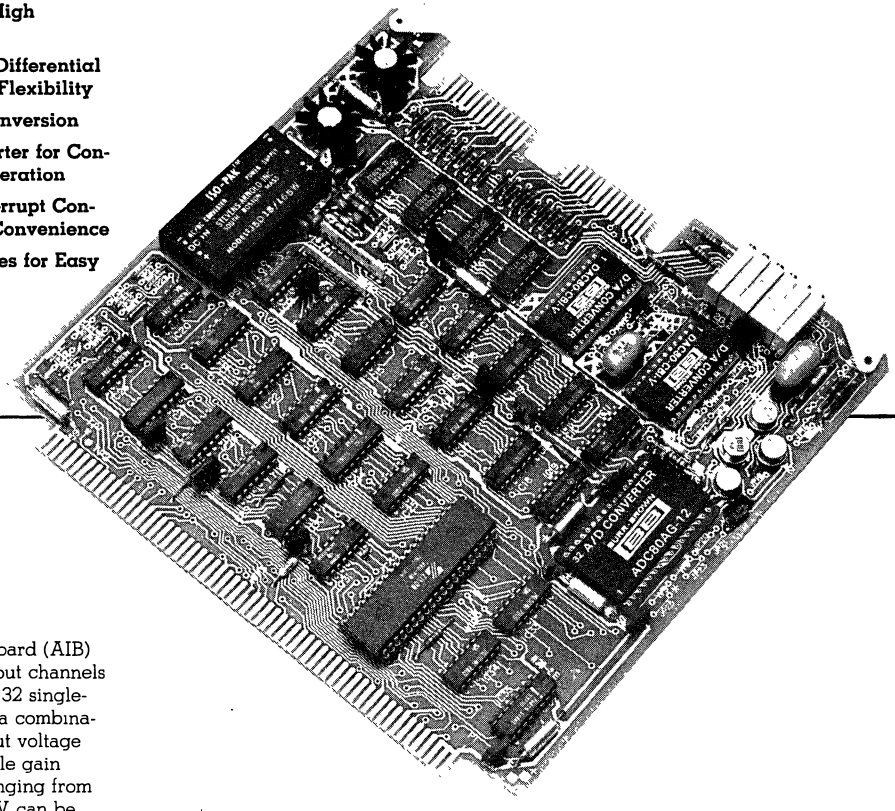
Z80® AIO/AIB
Z80 Analog Input/Output and Analog Input Boards

Zilog

Product Description

March 1981

- **12-Bit Resolution and High Accuracy**
- **16 Single-Ended or 32 Differential Inputs for Application Flexibility**
- **Fast 45 ms Channel Conversion**
- **On-Board dc-dc Converter for Convenient Low-Power Operation**
- **Polled or Vectored Interrupt Control for Programming Convenience**
- **Multiple Voltage Ranges for Easy Interface**



OVERVIEW

The Z80 Analog Input Board (AIB) provides 16 differential input channels that may be configured as 32 single-ended channels. Through a combination of user-selectable input voltage ranges and a programmable gain amplifier, input signals ranging from millivolts to as high as 10 V can be converted to a 12-bit word. In order to ensure accuracy and compatibility with the other MCB family boards, a 5 V dc-dc converter is included as a standard feature.

The Z80 Analog Input/Output (AIO) Board has input features identical to the AIB except that there are also two 12-bit D/A output channels, each with a wide range of user-selectable output voltages.

FUNCTIONAL DESCRIPTION

Input Ranges. The AIB and AIO contain an input multiplexer, an amplifier whose gain may be altered from 1 to 1000, and an analog-to-digital converter module. Five basic input ranges are shown in *Table 1*. The bipolar inputs are converted into a 12-bit value in twos complement format; the unipolar inputs are converted into a 12-bit straight binary value.

0.0	to	+4.9988 V
0.0	to	+9.9975 V
-2.500	to	+2.4988 V
-5.000	to	+4.9975 V
-10.000	to	+9.9951 V

Table 1. Input and Output Voltage Ranges

Amplifier Gain. Amplifier gain is set to 1 but can be changed by a resistor substitution according to the following formula:

$$R = \frac{20\ k\Omega}{Gain - 1}$$

Increasing the gain of the amplifier effectively allows the input voltage range to be scaled by the reciprocal of the gain factor. For example, by increasing the amplifier gain to 1000, an input voltage range of ±2.5 V becomes ±2.5 mV. As the gain is increased the settling time of the amplifier will also increase.

Because the AIO and AIB use a fixed timing sequence between channel selection and the start of data conversion, the system delay time must be lengthened, via a resistor change, to allow for the greater settling time of the amplifier at higher gain (see *Table 2*).

Amplifier Gain	Delay Time μs	Resistance kΩ
1	20	13.3
10	30	14.3
100	40	19.0
1000	100	47.5

Table 2. Recommended System Delay Time vs Amplifier Gain

Input Modes. The standard 16-channel differential input configuration is recommended in areas of common-mode noise and for low-level inputs. For input signals of 1.0 V or more, a 32-channel single-ended configuration can be jumper selected.

Equivalent Input Circuit. Source output impedance has an effect on the settling time of the multiplexer. The formula for the time constant and the equivalent single-ended input circuit is shown in *Figure 1*. The multiplexer must be allowed to settle to ±.01% (approximately nine time constants) to insure accuracy. For high source impedance, it may be necessary to increase the system delay time beyond that shown in *Table 2*. For the differential input configuration, the multiplexer time constant is one half of that in *Figure 1*.

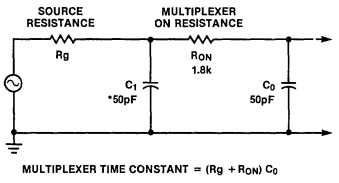


Figure 1. Input Equivalent Circuit

System Interface. The AIO and AIB occupy 10 locations within the MCB CPU's I/O address space as shown in *Table 3*. Input status, control and data are interfaced through a PIO while the data for the two output channels is written to a set of 12-bit output registers.

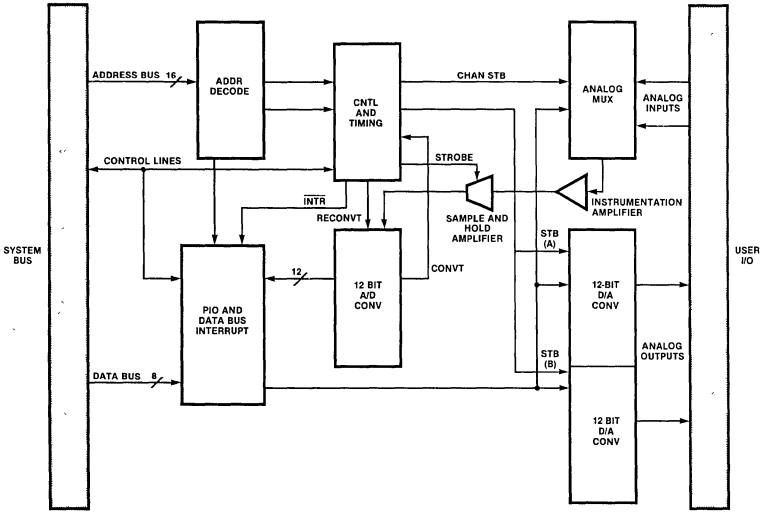
The location of the port assignments may be moved anywhere within valid I/O space of the CPU, with the restriction that both the PIO and output registers must reside within the same 20H block of I/O addresses. These address changes are jumper-selectable.

Data may be obtained in either a polled or fully vectored interrupt mode. The mode is selected entirely by software control.

Output Ranges. The AIO board is configured with two independent 12-bit digital-to-analog convertor output channels. Output voltage range is selectable by the appropriate jumper configuration. The available full scale output ranges are shown in *Table 1*. Output quantities are represented as twos complement numbers for bipolar ranges and as straight binary numbers for the unipolar configuration.

Function	Port	
P10 Port A Data	80	
P10 Port B Data	81	
P10 Port A Control	82	
P10 Port B Control	83	
Address Register (Channel Select)	88	
Status Register	89	
DAC1 Output (Lo Byte)	8C	
DAC1 Output (Hi Byte)	8D	AIO
DAC2 Output (Lo Byte)	8E	Only
DAC2 Output (Hi Byte)	8F	

Table 3. AIO/AIB Port Assignments



Z80 AIO/AIB Block Diagram

SPECIFICATIONS

Input Characteristics
- Number of Channels
  - 32 Single-ended/ 16 Differential
- ADC Gain Ranges
  - 0–5 V, 0–10 V, ±2.5V, ±5 V, ±10 V
- Amplifier Gain Ranges
  - 1 to 1000
- Max Input Voltage
  - ±26 V
- Input Impedance
  - 100 MΩ, 10pF OFF Channel
  - 100 MΩ ON Channel
- Bias Current
  - 20 nA
- Differential Bias Current
  - 10 nA
- Resolution
  - 12 Bits
- Throughput Time
  - Gain = 1 — 45 μs Channel
  - Gain = 100 — 100 μs Channel
- Accuracy
  - Gain = 1 — ±0.025% FSR
  - Gain = 1000 — ±0.100% FSR
- Linearity
  - ±1/2 LSB
- Differential Linearity
  - ±1/2 LSB
- Quantizing Error
  - ±1/2 LSB
- Temperature Stability
  - Gain = 1 — ±30ppm of FSR/°C
  - Gain = 1000 — ±80ppm of FSR/°C
- Dynamic Accuracy
  - Sample and Hold Aperature
    - 30 ms
  - Aperature Time Variation
    - ±5 ms
  - Differential Amplifier CMR
    - 74 db (dc to 1 kHz)
  - Crosstalk
    - 80 db down @ 1 kHz for OFF and ON Channel

Output Characteristics
- Number of Channels
  - 2
- Output Voltage Ranges
  - 0–5 V, 0–10 V, ±2.5V, ±5 V, ±10 V
- Output Current
  - 5 mA
- Output Impedance
  - 1
- Resolution
  - 12 bits
- Output Settling Time
  - 10 μs (max)
- Accuracy
  - Output Accuracy
    - ±0.0125% FSR
  - Temperature Coefficient
    - ±30ppm of FSR/°C

Connectors
- 122-Pin Edge (100 mil spacing)

Power
- +5V ±5% @ 1.6 A (max)

Environmental
- Temperature — 0 to 50°C
- Humidity — 0 to 90% noncondensing

Physical
- Height — 7.5" (191 mm)
- Width — 7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6075-01	Z80 AIO Analog Input/Output Board
05-6075-02	Z80 AIB Analog Input Board

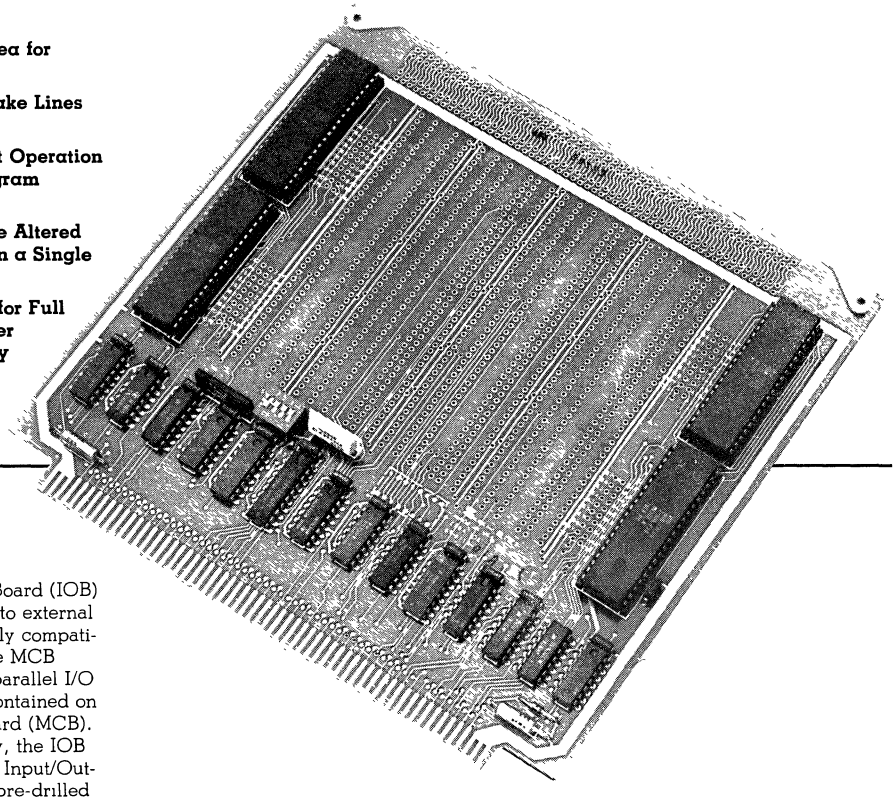
Z80® IOB
Z80 Input/Output Board

Zilog

Product Description

March 1981

- **Large User Interface Area for Application Flexibility**
- **64 Data and 16 Handshake Lines for Easy Interface**
- **Fully Vectored Interrupt Operation Allows Convenient Program Design**
- **Port Assignment May Be Altered to Allow Several IOBs in a Single System**
- **Uses Z80A PIO Devices for Full Compatibility with Other Members of MCB Family**



OVERVIEW

The Z80A Input/Output Board (IOB) provides system expansion to external digital I/O devices. It is fully compatible with other boards in the MCB family and provides eight parallel I/O ports to augment the two contained on the Z80 Microcomputer Board (MCB). Designed for user flexibility, the IOB contains four Z80A Parallel Input/Output (PIO) devices, a large pre-drilled user interface area, daisy-chain interrupt priority logic and user-selectable port address assignment.

FUNCTIONAL DESCRIPTION

The IOB contains four PIO controllers which provide 64 programmable I/O lines. These lines may be configured either as individual data lines with independent data direction or as groups of eight lines for byte-oriented data transfer. The IOB gives the user a headstart on special interface requirements by providing a large pre-drilled, pre-etched interface area. The hole array is spaced on .3" and .6" centers in a flexible arrangement that accommodates 16-pin, 24-pin or 40-pin ICs.

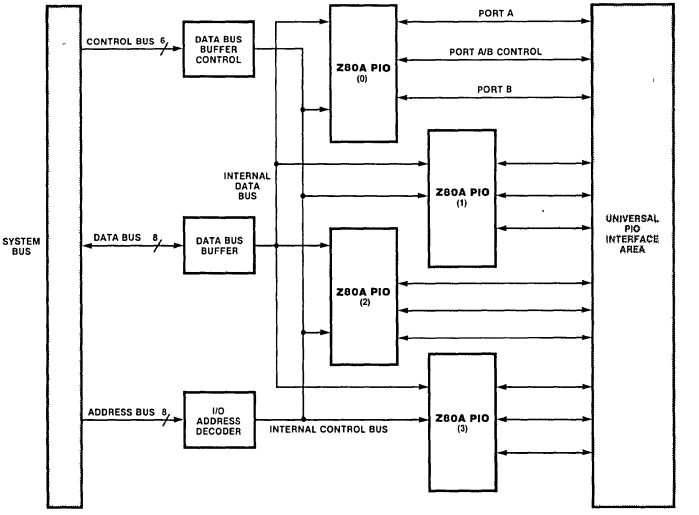
Parallel Input/Output. Each Z80A PIO device is a programmable, dual-port circuit that provides a TTL-compatible interface between peripheral devices and the Z80 CPU.

The PIO interfaces to peripherals via two independent general-purpose I/O ports designated Port A and Port B. Each port has eight data bits and two handshake signals, READY and STROBE, which control data transfer. The READY output indicates to the peripheral that the port is ready for a data transfer; STROBE is an input from the peripheral that indicates that the data transfer has occurred. In addition, the eight output lines from Port B can drive Darlington transistors (1.5 mA at 1.5 V).

Operating Modes. Each group of eight lines is capable of being programmed in one of four modes of operation—byte output, byte input, byte input/output and bit input/output.

Input Operation. The PIO device allows fully vectored interrupt operation with a unique vector for each port. The interrupt ability of each port may be enabled or disabled independently of the other ports. Interrupt priority is established by a hardware daisy-chain arrangement. Each group of lines has a fixed position within the priority structure; individual lines within each port are assigned equal priority. (See the *Z80 PIO Product Specification* for details.)

Port Assignments. By jumper placement, the four PIOs can be placed in any of eight 32-byte address ranges allowing the system to be easily configured and expanded.



Z80A® IOB Block Diagram

SPECIFICATIONS

I/O Lines
64 Programmable

Operational Modes
Input, Output, Bidirectional, Bit Control

Handshake
8 Ready and 8 Strobe Lines

Interrupt Vectors
8

I/O Port Locations
16 User-selectable within 1-of-8 Blocks

Output Voltage

HIGH	2.4 V (min) @ 250 mA Output Current
LOW	0.4 V (max) @ 2.0 mA Sink Current

Darlington Drive Current
Port B of Each PIO
3.8 mA (max) @ 1.5 V

Input Voltage

HIGH	2.0 V (min)
LOW	0.8 V (max)

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @ 0.5 A (max)
(without user ICs)

Environmental

Temperature	0 to 50°C
Humidity	0 to 90% noncondensing

Physical

Height	7.5" (191 mm)
Width	7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6006-03	Z80 IOB Input/Output Board

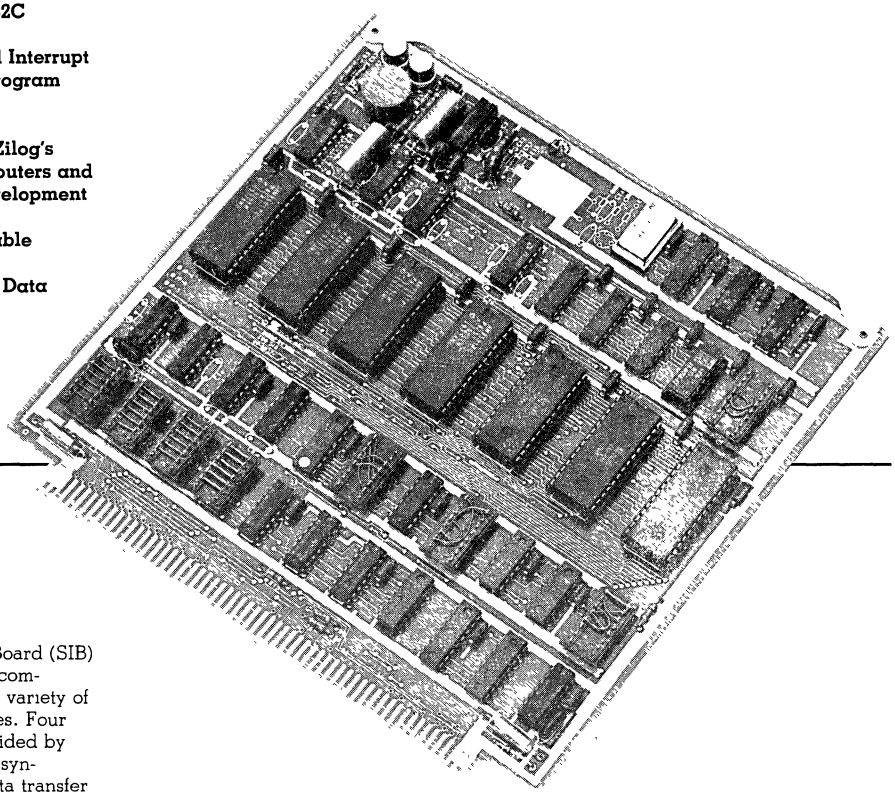
Z80® SIB
Z80 Serial Interface Board

Zilog

Product Description

March 1981

- **Industry Standard RS-232C Interface**
- **Polled or Fully Vectored Interrupt Control for Maximum Program Flexibility**
- **Single +5 V Operation**
- **Fully Compatible with Zilog's MCZ-1 Series Microcomputers and PDS 8000™ Product Development System**
- **Error Detection for Reliable Message Handling**
- **Four Powerful, Flexible Data Channels**



OVERVIEW

The Z80 Serial Interface Board (SIB) is a multiple channel serial communications interface with a variety of powerful, convenient features. Four independent channels, provided by 8251 USART devices, allow synchronous or asychronous data transfer with either half or full duplex signal handling. All four channels have drivers and receivers for RS-232C system interface and one will also accommodate a 20 mA current-loop interface. A dc-dc converter generates all necessary voltages from a single +5 V supply. An on-board crystal oscillator provides communication timing independent of the system clock.

FUNCTIONAL DESCRIPTION

The four SIB channels are capable of independent operation in either asynchronous or synchronous protocols. The system program may initiate and control either mode by selecting the appropriate command words. Both the transmitter and receiver sections are double-buffered for maximum performance and convenience. All data transfer status signals, such as TxRDY and RxRDY, are available in a readable status register or as external signals so that either polled operation

or full interrupt control may be selected by the user under software control. In addition to the normal data transmission, each channel can generate break signals and be individually reset under software control.

Asynchronous Mode. In the asynchronous mode, the system program controls the number of data bits (5, 6, 7, or 8), the number of stop bits (1, 1½, or 2,) and the sense of parity protection (even or odd) if enabled. Each channel has a programmable baud rate factor of 1, 16, or 64 controlling the relationship between the transmitted or received data rates and the frequency of the baud rate reference clock. See *Figure 1* for a description of the asynchronous mode control word. Error detection signals are available for each channel and may be read from the channel status register; these signals include parity error (PE), framing error (FE), and receiver overrun error (OE). *Figure 2* describes the channel status register.

Synchronous Mode. In the synchronous receive mode, character synchronization may be obtained from an external device or internally from the received data stream. The nature of the SYNC connection for each channel is programmed as either an input when the channel is expecting an external sync signal or as an output to identify that sync has been achieved. In addition, each channel may be programmed to operate with either single or double synchronizing characters.

Timing. The transmitter and receiver clocks for each USART channel can be derived from either the on-board crystal oscillator, thereby enabling operations to be independent of the main system clock frequency, or provided externally by the appropriate jumper selection.

For internal clock signal generation, input signals to two Counter/Timer Circuits (CTC) can be jumper-selected to be either 1/2 or 1/32 of the crystal frequency. The outputs of the CTCs are further divided by flip-flops to provide a 50% duty cycle to the USARTs. By programming each channel of the third on-board CTC with the proper time constant, baud rates of 50 to 38.4K are possible. *Table 1* shows time constants for various data rates when the USART has been programmed for a baud rate faster than 16.

Baud Rate	Time Constant Decimal	Hex
50	96	60
75	64	40
110	44	2C
134.5	36	24
150	32	24
200	24	18
300	16	10
600	8	8
1200	4	4
2400	2	2
4800	1	1
9600	4*	4*
19200	2*	2*
38400	1*	1*

*CTC in counter mode

Table 1. Baud Rate vs Time Constant for 16× Baud Rate Factor

Interrupt Control. Each channel may be selected to operate in either a polled mode or a fully vectored interrupt mode. The interrupt capability for each channel may be enabled or disabled by the programmer to allow mixing both modes. Each channel may be programmed to have a unique interrupt vector for the receiver ready and the transmitter ready signals, allowing independent interrupt service subroutines for each direction of data transfer. Interrupt priorities are assigned by the hardware on a daisy-chain basis. The four receiver ready signals are given priority over the four transmitter ready signals. The channel priority for each group ranges from channel 0 having highest priority to channel 3 the lowest.

Hardware Interface. Each of the four channels has drivers and receivers to allow full industry standard RS-232C interface parameters to external equipment. All voltages necessary for this

ADDRESS RANGE	J4 JUMPERS
00 to 1F	5-16, 1-7, 3-6
20 3F	5-15, 1-7, 3-6
40 5F	5-16, 2-7, 3-6
60 7F	5-15, 2-7, 3-6
80 9F	5-16, 1-7, 4-6
A0 BF	5-15, 1-7, 4-6
C0 DF	5-16, 2-7, 4-6
E0 FF	5-15, 2-7, 4-6

Table 2. Port Address Range

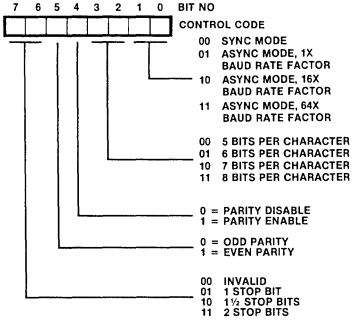


Figure 1. Channel Mode Control Word

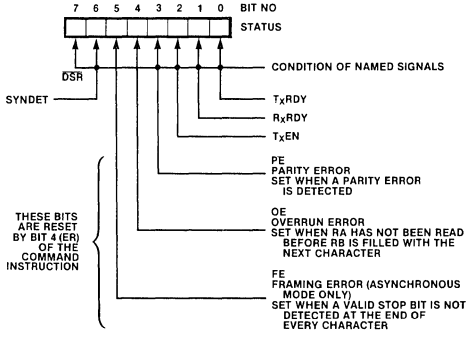
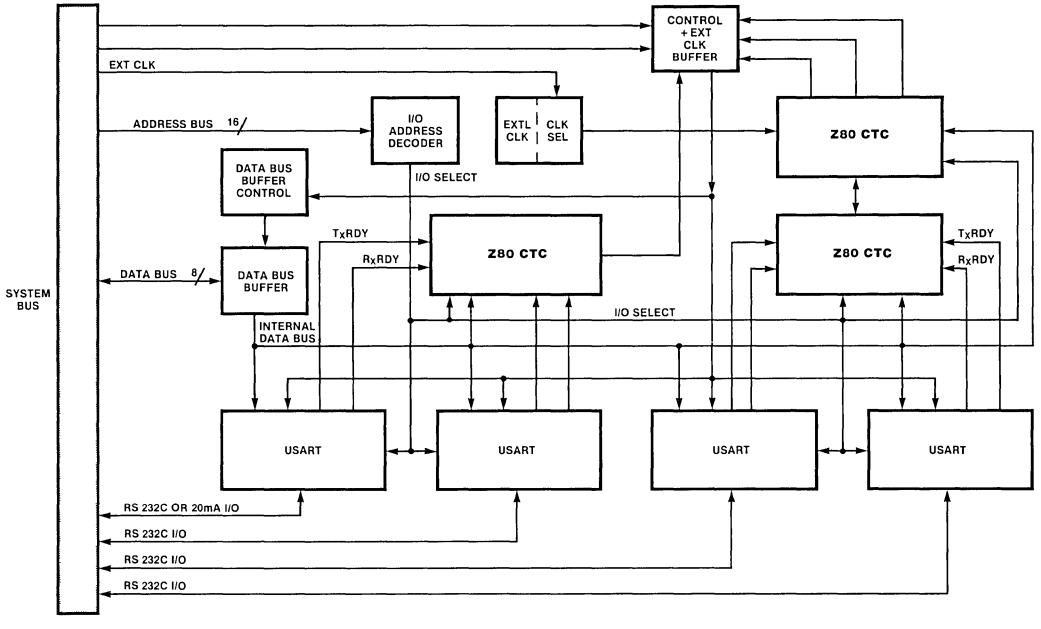


Figure 2. Channel Status Register

interface are provided by the dc-dc converter operating from a single +5 V input. Channel 3 is supplied with an active 20 mA current loop interface which the user may disable in favor of the RS-232C interface by selecting the appropriate jumper. In addition to the separate transmit and receive data signals, standard modem control signals such as DSR (data set ready), DTR (data terminal ready), CTS (clear to send), and RTS (request to send) are provided for each channel. The sense of each channel's interface is jumper-selectable so that the board may behave as either a terminal device or a modem device.

Port Selection. The SIB utilizes port assigned I/O and occupies locations within the I/O port assignment space. By selection of appropriate jumpers the user may place the SIB into any one of eight port address ranges, each offering 32 available port addresses. *Table 2* shows the possible address ranges for the SIB. Each of the four USARTs and the three CTCs may be placed at a unique location within the selected range. The user selects the appropriate jumper configuration for the location.



Z80 SIB Block Diagram

SPECIFICATIONS

Number of Channels
4

Mode
Full or Half Duplex

Baud Rates
50 to 38.4K Baud

Baud Rate Reference Clock
19.6608 MHz

Synchronization Method
External or Internal Character Match

Interface
Channels 0-3
RS232C
Channel 3
Current Loop Available

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @ 1.5 A (max)

Environmental

Temperature	0 to 50°C
Humidity	0 to 90% noncondensing

Physical

Height	7.5" (191 mm)
Width	7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6007-01	Z80 SIB Serial Interface Board

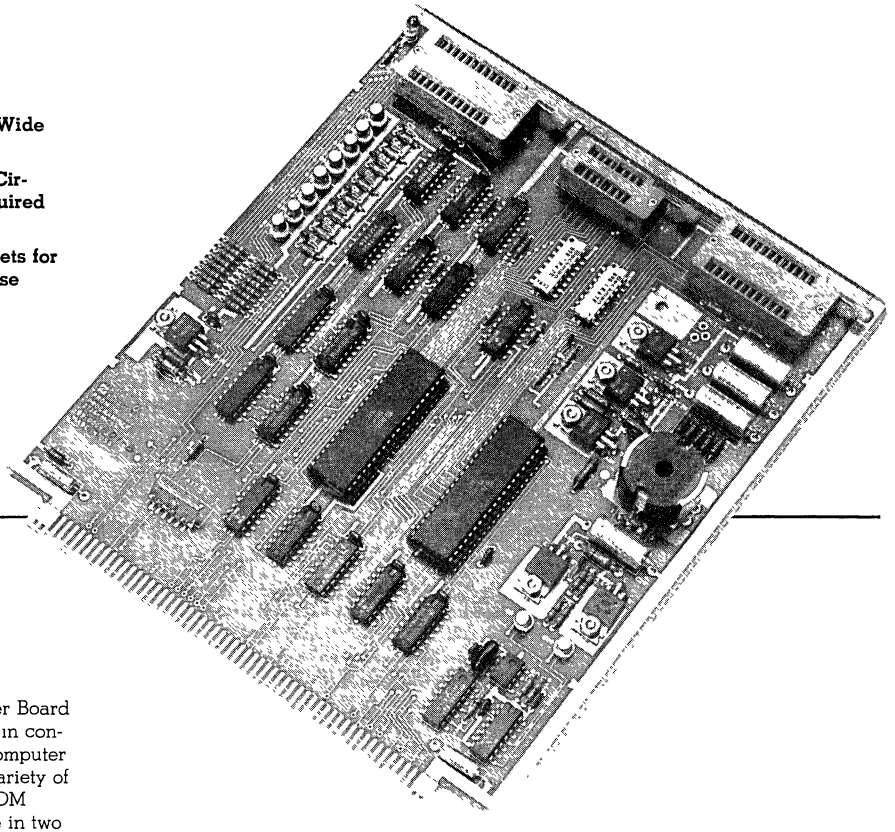
Z80® PPB
PROM Programmer Board

Zilog

Product Description

March 1981

- **Flexibility to Program a Wide Range of E/PROMs**
- **Complete Programming Circuitry Generates All Required Programming Voltages**
- **Zero Force Insertion Sockets for Reliability and Ease of Use**



OVERVIEW

The Z80 PROM Programmer Board (PPB) is designed to be used in conjunction with the Z80 Microcomputer Board (MCB) to program a variety of MOS E/PROM or bipolar PROM devices. The PPB is available in two configurations, PPB and PPB/16, each capable of programming a specific type of E/PROM. All necessary programming voltages are generated on the boards making them completely compatible with the MCB family, MCZ™ microcomputers or ZDS development systems.

FUNCTIONAL DESCRIPTION

The PPB uses Z80 PIO devices to interface between the E/PROM sockets and the system microprocessor. Single-byte data transfers in both directions permit either reading or programming of the selected E/PROM socket. Additional parallel I/O lines control the mode of operation and provide chip select to the desired socket.

Zero force insertion sockets are used in the programming locations to provide convenience, reliability and long life. The programmer board extends beyond the card cage for easy access to programming sockets mounted near the board edge. Each board contains one 16-pin and two 24-pin sockets.

Device	Organization
MOS E/PROMs	
2704	512 × 8
2708	1024 × 8
Bipolar PROMs	
7610	256 × 4
7611	256 × 4
7620	512 × 8
7621	512 × 8
7640	1024 × 8
7641	1024 × 8

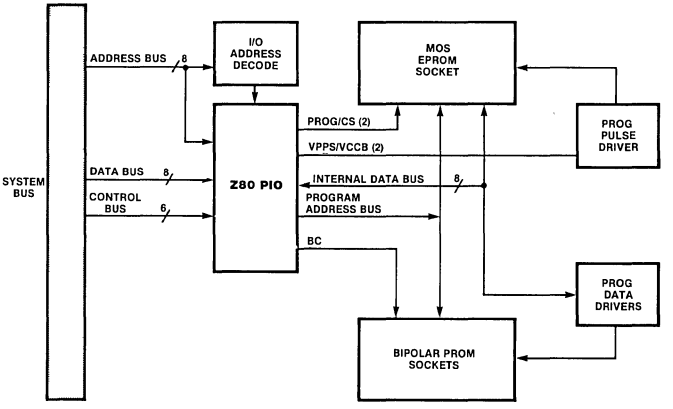
Table 1. PPB E/PROM Devices

PROM Types. The PPB is designed to program 2704 and 2708 E/PROM devices and Harris-type bipolar devices. (See *Table 1* for device selection.) The PPB/16 allows programming of 5 V 2716-type E/PROM devices and Signetic-type bipolar devices (see *Table 2*).

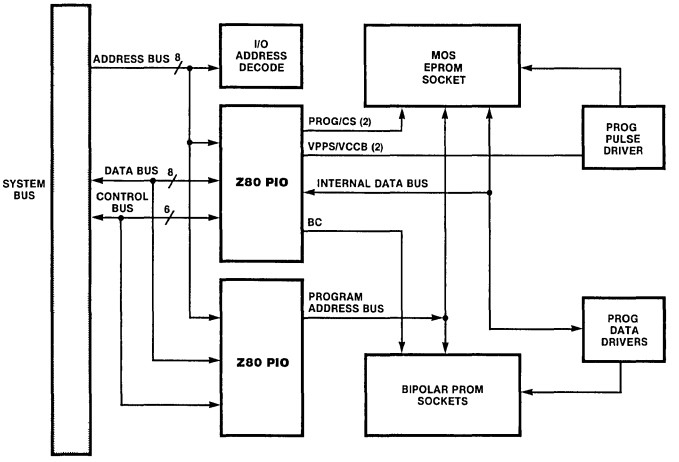
Device	Organization
MOS E/PROMs	
2716	2048 × 8
Bipolar PROMs	
82S126	256 × 4
82S129	256 × 4
82S130	512 × 4
82S131	512 × 4
82S140	512 × 8
82S141	512 × 8
82S180	1024 × 8
82S181	1024 × 8
82S2708	1024 × 8

Table 2. PPB/16 E/PROM Devices

Software. Both programmer boards are supported by the Z-PROG utility which is part of Zilog's RIO™ operating system. Z-PROG is an easy to use interactive program that allows E/PROMs to be read, programmed from disk file and duplicated, and allows the user to select the appropriate socket by specifying the E/PROM type and the word length. Z-PROG also provides address boundary selection for partial E/PROM programming.



Z80® PPB/16 Block Diagram



Z80® PPB Block Diagram

SPECIFICATIONS

E/PROM Sockets
One 16-Pin Zero Force Insertion
Two 24-Pin Zero Force Insertion

E/PROM Types
24-Pin MOS
2704 (512 × 8) PPB
2708 (1024 × 8) PPB
2716 (2048 × 8) PPB/16
24-Pin Bipolar
7640 (1024 × 8) PPB
7641 (1024 × 8) PPB
82S140 (512 × 8) PPB/16
82S141 (512 × 8) PPB/16
82S180 (1024 × 8) PPB/16
82S181 (1024 × 8) PPB/16
82S2708 (1024 × 8) PPB/16
16-Pin Bipolar
7610 (256 × 4) PPB
7611 (256 × 4) PPB
7620 (512 × 8) PPB
7621 (512 × 8) PPB
82S126 (256 × 4) PPB/16
82S129 (256 × 4) PPB/16
82S130 (512 × 4) PPB/16
82S131 (512 × 4) PPB/16

Control Interface
TTL Interface with MCZ Series Data, Address and Control Signals

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @
2.5 A during Programming
1.5 A during Read

Environmental

Temperature	0 to 50°C
Humidity	0 to 90% noncondensing

Physical:

Height	9.0 in. (229 mm)
Width	7.7 in. (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6005-01	Z80 PPB PROM Programming Board
05-6079	Z80 PPB/16 PROM Programming Board

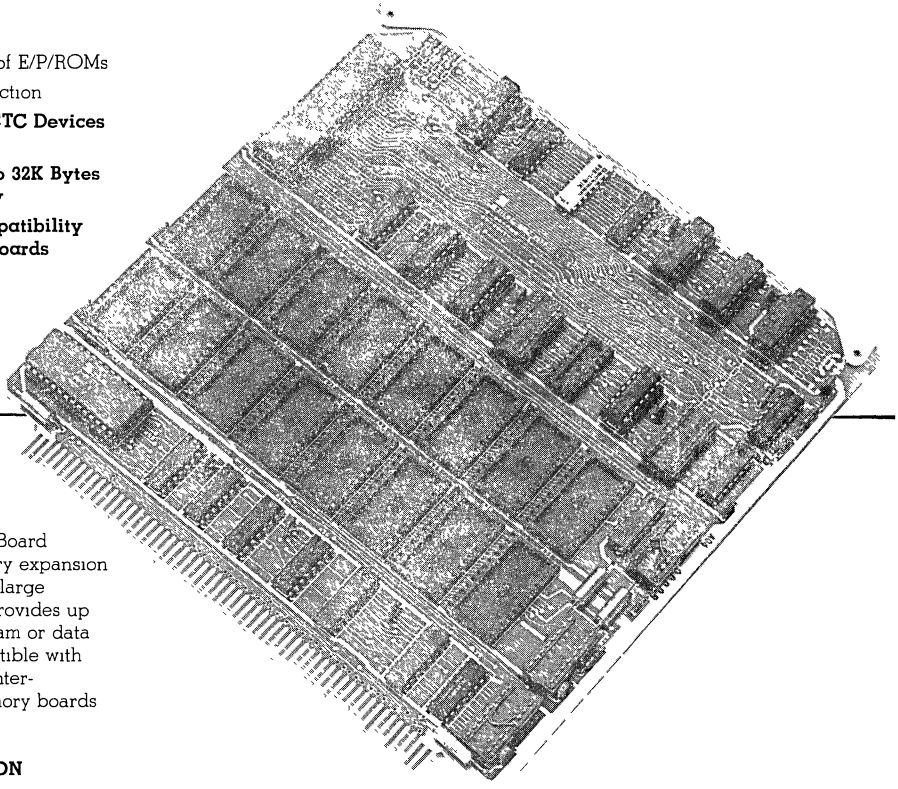
Z80® PMB
Z80 PROM Memory Board

Zilog

Product Description

March 1981

- **Flexible Application**
  - Allows several types of E/P/ROMs
  - Variable address selection
- **Includes Z80 PIO and CTC Devices for I/O Expansion**
- **Allows Expansion Up to 32K Bytes of Non-volatile Memory**
- **Fully Buffered for Compatibility with All MCB Family Boards**



OVERVIEW

The Z80 PROM Memory Board (PMB), designed for memory expansion in systems which require a large amount of fixed memory, provides up to 32K bytes of fixed program or data storage. Completely compatible with the Z80 MCB, the PMB is interchangeable with other memory boards within the MCB family.

FUNCTIONAL DESCRIPTION

Memory Array. The PMB contains 16 24-pin sockets to accommodate a variety of E/P/ROM devices as shown in *Table 1.* Flexibility in the selection of the device type is provided in the form of jumpers that may be installed on a 16-pin component carrier. Chip selection logic allows each socket within the array to be configured to have a unique address starting on 1K byte boundaries. In addition, each socket may be programmed to have either a 1K byte or 2K byte granularity depending upon the memory device chosen. Chip selection is accomplished by a pair of socketed 32 × 8 PROMs.

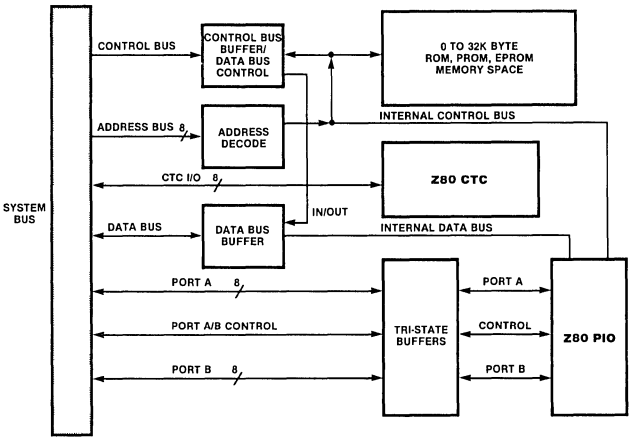
Parallel I/O. An on-board Z80 PIO device provides additional system I/O via 16 status or data lines which may be configured individually or in two groups of eight. (See *Z80 PIO Product Specification* for additional details.) There are two sets of Ready-Strobe handshake lines for each group of I/O lines. Drivers for both ports are provided for use in the output mode; termination resistor sockets are available for use in the input mode.

Non-Volatile Memory	Device Number	
MOS E/PROM	2704	8704
	2708	8708
	2716	2316
Bipolar P/ROM	6341	
	6381	
	82S181	
	82S191	

Table 1. Non-Volatile Memory Devices

Counter/Timer. An on-board Z80 Counter/Timer Circuit provides expanded timing capability. The Z80 CTC includes four independent 8-bit counter/timers and can be programmed by system software for event counting, interrupt and interval timing, and general clock rate generation (See *Z80 CTC Product Specification* for specific details.)

Port Assignments. The chip select logic allows each of the two I/O devices (CTC and PIO) to be located within any one of eight port assignment blocks each containing 20H bytes for I/O locations. Each device must occupy four consecutive locations within the chosen block. The configuration desired by the user is easily achieved by selecting appropriate jumpers that reside on component carriers.



Z80® PMB Block Diagram

SPECIFICATIONS

Memory Capacity
32K (Populated with 2K Devices)

E/P/ROM Socket Array
Number 16 (24-pin)

E/P/ROM Device Types
2708, 2716, 6381

Parallel I/O
Number of Lines—16 (Programmable)
Operating Modes—Input, Output, Bidirectional, Bit Control
Handshake Lines—Ready, Strobe
Interrupt Vectors—2 (User Programmable)

Counter, Timer
Channels
4 (8 Bits Each)
Interrupt Vectors
4 (User Programmable)

Connector
122-Pin Edge (100 mil spacing)

Power
+5 V ±5%
@ 0.60 A (max) without Memory
@ 2.28 A (max) 2716
@ 3.40 A (max) 6381
@ 0.84 A (max) 2708
−5 V ±5%
@ 0.96 A (max) 2708
+12 V ±5%
@ 1.28 A (max) 2708

DC-DC Converter Output
+12 V @ 320 mA (max)
−5 V @ 50 mA (max)

Environmental
Temperature 0 to 50°C
Humidity 0 to 90% noncondensing

Physical
Height 7.5" (191 mm)
Width 7.7" (196 mm)

ORDERING INFORMATION

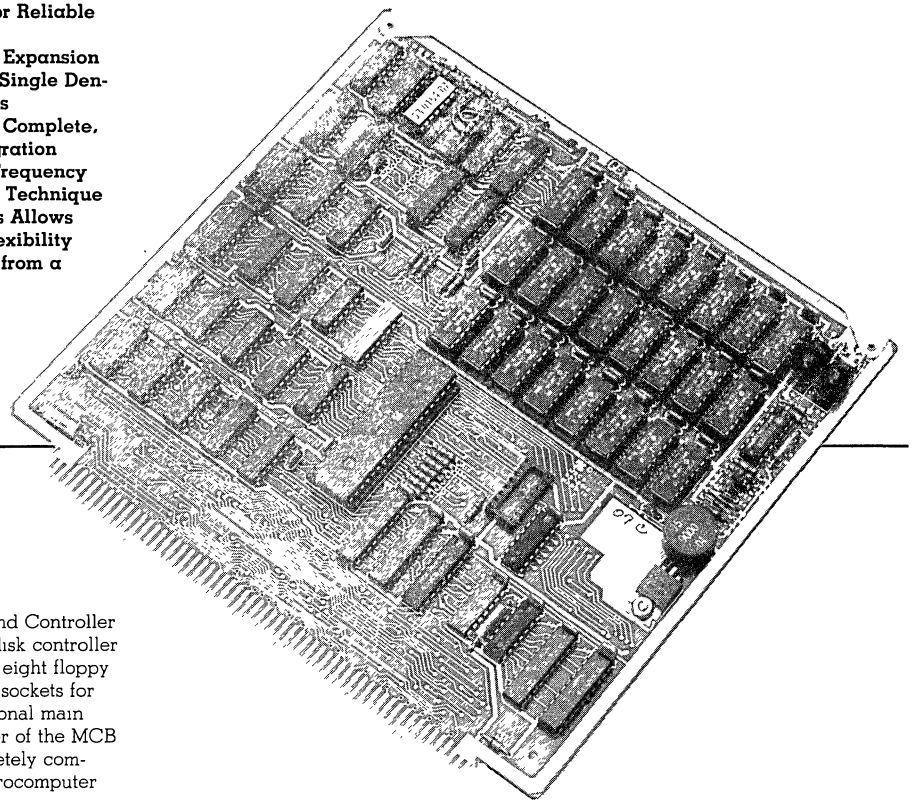
Part No.	Description
05-6023-01	Z80 PMB PROM Memory Board

Z80® MDC
Z80 Memory and Disk Controller Board

Product Description

March 1981

- **CRC Error Checking for Reliable Data Transfer**
- **Control Signals Allow Expansion Up to Eight Full Size, Single Density Floppy Disk Drives**
- **Memory Array Allows Complete, Compact System Integration**
- **Reliable and Proven Frequency Modulation Recording Technique**
- **CPU-Controlled Access Allows Complete Software Flexibility**
- **Low-Power Operation from a Single +5 V Supply**



OVERVIEW

The Z80 Memory Disk and Controller (MDC) board is a floppy disk controller capable of handling up to eight floppy disk drives and providing sockets for 16K to 48K bytes of additional main system memory. A member of the MCB family, the MDC is completely compatible with the other microcomputer boards in the series.

The MDC is most effectively used with the MCB/16 Microcomputer Board. Together these two boards comprise a complete microcomputer system that includes 64K bytes of RAM, 4K bytes of PROM, parallel interface, serial interface, and control of up to eight floppy disk drives—on a 115 sq. in. circuit board which operates from a single +5 V power supply.

FUNCTIONAL DESCRIPTION

Memory Array. The memory array is implemented using 16K × 1-bit dynamic RAM devices to provide 16K bytes to 48K bytes of main system memory. Although dynamic RAMs are used in the memory array, additional refresh circuitry is not required due to the unique memory refresh characteristic of the MCB CPU. Following each op-code fetch, a new refresh address is available on the system address bus while the op-code is being decoded within the processor.

An on-board dc-dc converter generates the −5 and +12 V signals for the dynamic memory devices, enabling the MDC board to be operated from a single +5 V power supply.

Memory address selection is completely compatible with the MCB/16. This two-board combination provides 64K bytes of continuous memory within the address space of the MCB CPU. For maximum flexibility, the RAM chip select logic is designed to allow the memory to be addressed in 4K byte blocks that may be located anywhere within the address range of the CPU. Chip selection is accomplished using a PROM decoder to select the Row Address Strobe (RAS) signal to the appropriate bank of devices. This address select PROM is socketed so that it may easily be replaced by the user for address reassignment.

Disk Control. The disk control signals, formatting information and data transfer are provided by the CPU under program control. A PIO device is used as the interface element to transfer disk control and status information between the CPU and the control circuitry on the disk drive units. Disk status signals include READY, TRACK 0, SECTOR MARKER, WRITE PROTECT, and CRC ERROR. The control signals are DIRECTION, STEP, four DISK SELECT lines, READ, WRITE, and ENABLE CRC.

The MDC includes a CRC used during read and write operations. This circuit generates a 16-bit word which is appended to the end of the data stream during write operations. During read operations a 16-bit word is again computed and then compared with the value previously written on the disk. A CRC error condition causes an error flag to be read into the CPU through the PIO interface.

Data is recorded onto the floppy diskette in a serial format. Parallel-to-serial and serial-to-parallel data conversion is performed by on-board circuitry. During the frequency modulation recording mode, each data bit recorded on the diskette has an associated clock bit recorded.

Formatting of serial data into the disk is accomplished under program control by the MCB CPU. Optional PROM-based firmware to control up to two Shugart 801R Floppy Disk Drives is available from Zilog. This firmwave assumes that 32 data sectors (records) are utilized per track and 77 tracks are utilized per disk. The firmwave provides all control functions for the disk and performs all data transfer. The sector data format is illustrated in *Figure 1.*

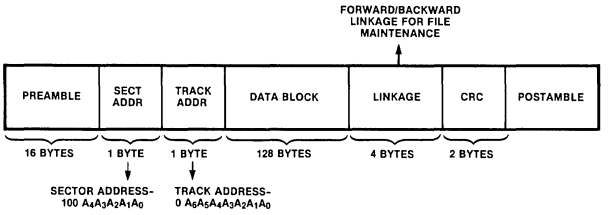
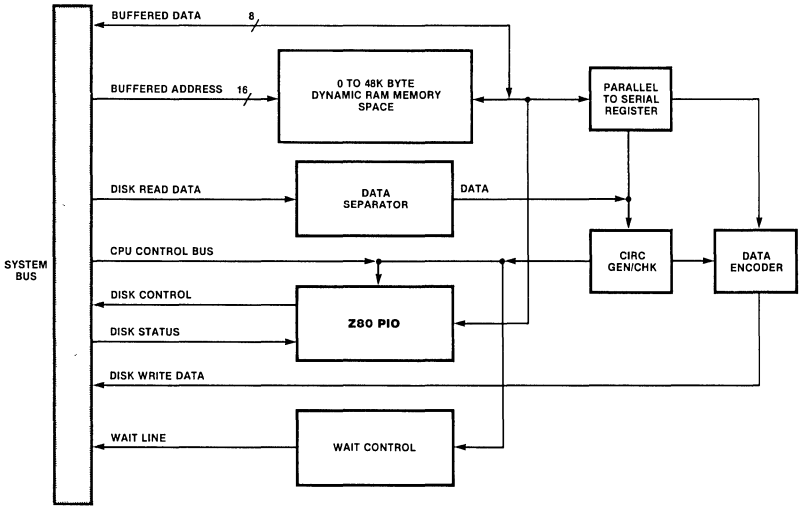


Figure 1. Sector Data Format



Z80 MDC Block Diagram

SPECIFICATIONS

Disk Drive Capability
8 Single-Sided Drives

Disk Drive Characteristics

Sector Type	Hard
Recording	Single Density
Sectors per Track	32
Tracks per Disk	77
Capacity	308K Bytes Data

Data Transfer Mode
Programmed I/O

Memory Capacity
48K Bytes

Memory Configurations
16K, 32K, or 48K Bytes Dynamic RAM. Each 4K page may have its starting address assigned to any of 16 possible values.

Connectors
122-Pin Edge (100 mil spacing)

Power
+5 V ±5% @ 1.6 A max.

Environmental

Temperature	0 to 50°C
Humidity	0 to 90% noncondensing

Physical

Height	7.5" (191 mm)
Width	7.7" (196 mm)

ORDERING INFORMATION

Part No.	Description
05-6011-04	Z80 MDC/16 16K Memory and Disk Controller
05-6011-03	Z80 MDC/32 32K Memory and Disk Controller
05-6011-02	Z80 MDC/48 48K Memory and Disk Controller

Zilog
Development Systems

Zilog

Comprehensive Development Environments for All Zilog Microprocessors

Innovative Design. Zilog's development system products feature ideal environments for software development for the Z8, Z80, and Z8000 microprocessors. The modularized design approach of the Zilog development systems allows the user a choice of hardware and software modules to meet current needs, while providing the necessary upgradability for future requirements.

Proven Components. The PDS 8000 Family and ZDS-1 Family of development systems provide development support for the Z8, Z80, and Z8000 microprocessors. The PDS 8000 systems are software development stations, while the ZDS-1 systems contain integrated Z80 emulators, which permit full hardware and software debugging of the Z80 target system. Each of these systems offers variable configuration choices and extra card slots for additional peripherals. Ample provisions have been made for the expansion of memory, disk storage, PROM programming, and external interface. And each system is supplied with Zilog's field-proven RIO operating system and the necessary utilities.

The Z-LAB 8000, Zilog's newest development system, offers multi-user capabilities with ZEUS, a UNIX*-based operating system with the 6MHz Z8010A microcomputer. The Z-LAB 8000 incorporates 2.4M bytes of reliable Winchester disk storage, a cartridge tape backup facility, and 256K bytes of error-correcting memory to represent the programmer's ideal development system.

In addition to the standard development system configuration, several optional modules are also available for system enhancement.

Hardware. The Z-SCAN 8000 Emulator provides in-circuit emulation for the Z8001 or Z8002 16-bit microprocessors and may be used as a stand-alone unit or as a peripheral to a host CPU such as the Z-LAB 8000 and/or a CRT.

The Z8 and Z8000 Development Modules are complete single-board microcomputers that permit the development of code for the Z8 or Z8000. They facilitate prototyping with large wire-wrap areas and are totally transparent to the CRTs and host CPU systems.

Software. To facilitate program development, Zilog offers the complementary PLZ application languages, PLZ/SYS and PLZ/ASM. Similar constructs within the PLZ languages permit the user to combine high-level, machine-independent modules together with machine-dependent modules.

PLZ/SYS is a procedure-oriented language with a style that blends elements of other well known languages such as Pascal, ALGOL, PL/I and C.

PLZ/ASM is a structured assembly language that provides all the capabilities needed to manage the microprocessor resources such as registers, memory accesses, and I/O operations.

This modular programming technique enables the programmer to concentrate on program design rather than on development system software.

The Z8000 Cross-Software Package, running on UNIX*, enables multi-user access for enhanced software development. The package consists of a complete set of software tools for developing Z8000 programs on DEC's PDP-11/44, 11/45, and 11/70 systems. The C language, including compiler and code optimizer, protects the user's software investment by permitting program transportability.

* UNIX is the trademark of Bell Laboratories.

ZDS-1/25 Development System

Zilog

Product Description

March 1981

- **Full Development Support for the Z80® Microprocessor.**
- **EPROM-based Monitor/Debug Software.**
- **In-Circuit Emulation up to 2.5 MHz.**
- **Memory Mapping Allows Borrowing of System Resources before Prototype Memory Is Available.**

OVERVIEW

The ZDS-1/25 Development System is a cost-effective development aid providing total design and prototyping capability for Z80 CPU-based systems. This capability enables the user to develop Z80 code before prototype hardware is built. Once prototype hardware is available, the ZDS-1/25 system can then be used as an in-circuit emulator to allow integration of software and hardware.

FUNCTIONAL DESCRIPTION

The ZDS-1/25 Development System consists of two functional parts: a software development system and an in-circuit emulation subsystem.

Software Development. As a software development system, the ZDS-1/25 includes a Z80-based microcomputer, 60K bytes of dynamic RAM, an EPROM monitor, floppy disk controller, serial RS-232C console interface and dual single-sided, single-density floppy disk drives. Zilog's field-proven RIO™ operating system is used to edit, assemble and modify Z80 code.

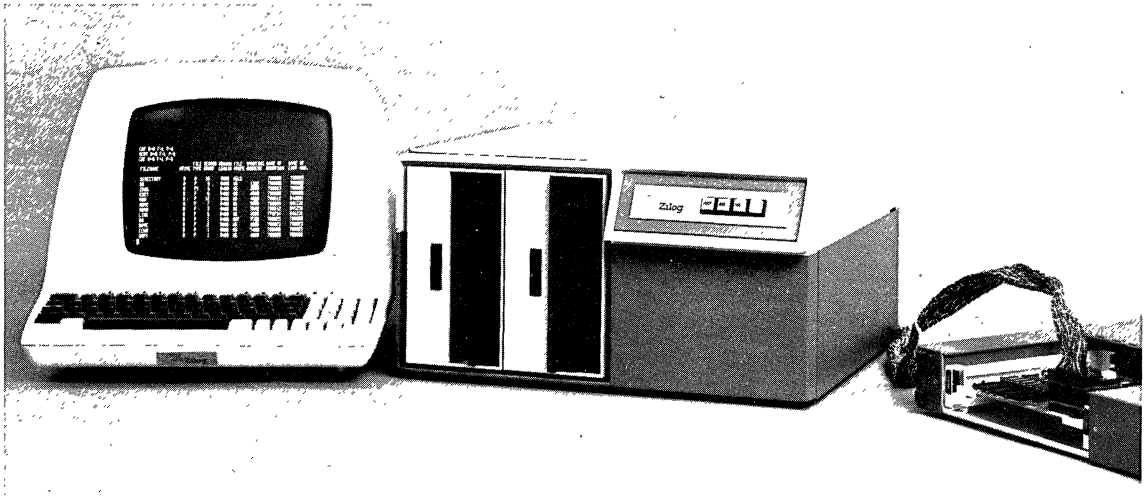
With relocatable modules, I/O management and general-purpose computing power, the RIO operating system facilitates both the development process and the expansion of system features to meet individual user needs. RIO's main features include an OS executive, relocating macro assembler, linker, text editor and ZDOS II file manager.

OS Executive. The OS executive handles I/O requests, dynamically allocates system storage areas to active programs on an "as needed" basis and invokes programs in response to operator commands.

Relocating Macro Assembler. The relocating macro assembler permits external symbol references, global symbol definitions and conditional assembly.

Linker. The linker resolves external references and assigns absolute addresses to program modules, thereby creating executable code. The linker also permits overlays and produces a memory map and a global address table.

Text Editor. A line-oriented text editor can handle files or programs larger than the available memory space. All operations within a file are based on character string matching to allow quick and easy search and modification of text. The capability to access other files during an edit session saves the repetitive entry of commonly used routines and enables the user to build libraries of commonly used code. Automatic backup of an existing file prevents accidental destruction of valuable data.



ZDOS II File Manager. The file manager organizes, stores and retrieves data from the floppy disk units. A directory provides a data index which is accessed using a "hierarchical linked list." All space on the disk is dynamically allocated on an as needed basis to prevent gaps in the storage space. Logical record lengths from 128 to 4096 bytes per record may be used. Also, all files may be assigned one or more attributes for protection and privacy.

In-Circuit Emulation. Once the software development task is complete, the ZDS-1/25 may be switched to the User mode to function as an in-circuit emulator. In this mode of operation, the system CPU becomes the emulator CPU and, in conjunction with the user interface module, monitor module and real-time trace module, provides full real-time emulation up to 2.5 MHz.

In the Emulator mode, 60K bytes of system memory, system peripheral devices and I/O ports (with the exception of E0 Hex through EF Hex) may be wholly or partially used by the user target system. The development system is connected to the user's target system via a three-foot emulator cable and 40-pin DIP plug.

As a standard part of the emulation package, the ZDS-1/25 system includes a mapper utility and EPROM-based monitor/debug firmware allowing real-time execution, trace and debug.

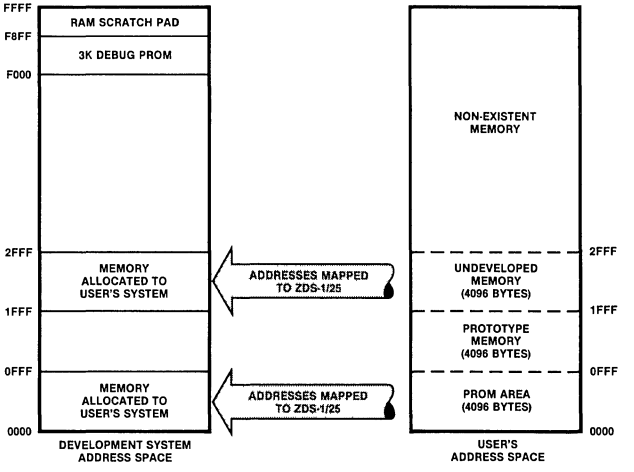


Figure 1. ZDS-1/25 Memory Mapping

Mapper Utility. The mapper utility allows the nature of memory used during emulation to be described. All or part of the memory used may be mapped to the ZDS-1/25 system memory or target system memory. Mapping occurs using blocks containing 256 bytes each. Mapped memory internal to the system may be write-protected and programmed to cause a break in emulation in the event of illegal access. The memory in the ZDS-1/25 must exist for the associated block being mapped in the user's prototype system.

EPROM Monitor/Debugger. The EPROM-based monitor provides the capability to control, analyze and debug software which may reside in either internal system memory, external target system memory or a combination of the two. The monitor command package consists of the following functional groups: Execution, such as GO, GET, SAVE and JUMP; Register/Memory Manipulation, such as DISPLAY, REGISTER and COMPARE; Input/Output; and Debug, such as BREAK, TRACE and HISTORY.

SPECIFICATIONS

CPU
Z80 CPU

Memory
60K Bytes User Memory Space (Dynamic RAM)

Word Size
8 Bits (1 Byte)

Clock Rate
2.5 MHz Crystal Controlled

Interrupts
Three modes including vectored, nonvectored and nonmaskable

Option Card Slots
Two (2)

Floppy Disk Storage

Capacity	300,000 bytes/drive
Type	Single-sided, single-density, hard-sectored

Floppy Disk Storage (Continued)

Maximum Capacity	600,000 bytes (dual drives)
Transfer Rate	250K bits/s
Average Latency	83 ms
Track-to-Track Seek	10 ms
Average Access Time	260 ms
Physical Sectors	32 sectors/track, 77 tracks

IN-CIRCUIT EMULATOR

Clock Rate	2.5 MHz
Hardware Trigger	Break on Address
Real-Time Trace Module	256 x 36 wide, high-speed static RAM
Emulator Cable	3 ft. (91 cm)

Physical

Height	10.0 in. (25.4 cm)
Width	19.0 in. (48.3 cm)
Depth	16.0 in. (40.6 cm)
Weight	65.0 lbs. (29.5 kg)

Power

110 V/50 Hz @ 1.5 A
220 V/50 Hz @ 0.9 A
110 V/60 Hz @ 1.8 A

Environmental

Operating Temperature	0° to 40°C
Storage Temperature	0° to 85°C
Humidity	20 to 80% noncondensing

ORDERING INFORMATION

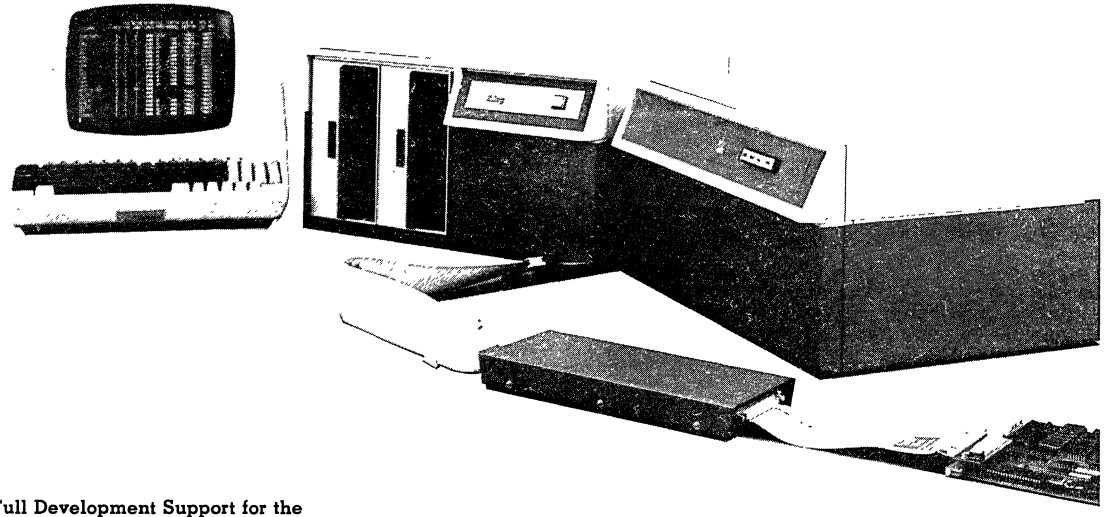
Part No.	Description
05-6030-05	ZDS-1/25 Development System (60 Hz)
05-6030-06	ZDS-1/25 Development System (50 Hz)

ZDS-1/40
Development System

Zilog

Product Description

March 1981



- **Full Development Support for the Z80® and Z80A Microprocessors**
- **64K Bytes of Memory to Support Large Programs**
- **600K Bytes Floppy Disk Storage**
- **In-Circuit Emulation up to 4 MHz**
- **Memory Mapping Allows Borrowing of System Memory Before Prototype Memory Is Built**
- **ZAP Package Provides Interactive, Symbolic Debugging With Disassembly**

OVERVIEW

The ZDS-1/40 Development System provides total development support for Z80 and Z80A CPU-based system designs. This support begins with a complete Z80-based microcomputer system that includes 64K bytes of RAM, dual single-sided, single density floppy disk drives and system software to assist in every phase of software development.

Included with the powerful microcomputer is an in-circuit emulation subsystem which connects to the user's prototype to monitor the execution of the software, control the behavior of the microprocessor in the prototype, and minimize the problems encountered in integrating software with hardware. Interactive debug software—the ZAP package, provided with the emulation system—allows debugging of the prototype, full disassembly of memory data and trace information, the use of symbolic references, and the capability of placing all debug commands on disk for execution.

FUNCTIONAL DESCRIPTION

The ZDS-1/40 Development System in effect consists of two functional parts: a software development host and an in-circuit emulation subsystem.

Software Development. The software development host is a Z80-based general-purpose microcomputer with 3K bytes of EPROM, 64K bytes of dynamic RAM, a floppy disk controller, serial RS-232C console interface, and two single-sided, single density floppy disk drives.

Included with the microcomputer system is Zilog's RIO™ Operating System and System Utilities. This set of tools provides the user with the full capability of carrying out the various development tasks from the inputting and assembly of source code to the printing of listings and the creation of EPROMs. The RIO operating system is designed to provide the user with the capability of tailoring commands and initialization routines to suit the needs of the specific application. The main features of RIO include a PROM-based monitor, OS executive, ZDOS II file manager, text editor, Z80 relocating macro assembler and linker.

PROM-Based Monitor. 3K bytes of nonvolatile storage provide system primitives for communication with floppy disk and console devices, and contain the bootstrap routine for the system.

OS Executive. The executive is the focus of system activity and thus handles I/O requests, dynamically allocates system storage areas to active programs on an "as needed" basis and invokes programs in response to operator commands.

ZDOS II File Manager. The file manager organizes, stores and retrieves data from the floppy disk units. A directory provides an index for the data, which is accessed using a "hierarchical linked list." All space on the disk is dynamically allocated on an "as needed" basis to prevent gaps in the storage space. Logical record lengths from 128 to 4096 bytes per record may be used. Also, all files may be assigned one or more attributes for protection and privacy.

Text Editor. A line-oriented text editor can handle files or programs larger than the available memory space. All operations within a file are based on character string matching to allow quick and easy search and modification of text. The capability to access other files during an edit session saves the repetitive entry of commonly used routines and enables the user to build libraries of commonly used code. Automatic backup of an existing file prevents accidental destruction of valuable data.

Z80 Relocating Macro Assembler. The relocating macro assembler provides a quick way to create Z80 code in a modular fashion. Its design supports absolute or relocatable object code formats, global definitions, external references, macros and conditional assembly. Optionally, a cross-reference and/or symbol table is limited only by available storage on the disk. All diagnostic messages are routed to the system console with pertinent line number, error and the statement itself so that there is no waiting for a listing to locate erroneous statements.

Z80 Linker. The Z80 Linker provides a means to link various program modules together and resolve communication between global modules, described by external references. The result is the generation of a single, executable program with absolute addresses. The use of the linker allows individual modules to be built and debugged, then merged with others without performing a complete assembly.

System Utilities. All of the software used to drive or control the various accessory boards available is included with the system. There is no need to write software to communicate with printers or PROM programmers because it is already completed. The source code for the utilities is included so that the user can supplement or custom-tailor the software.

In-Circuit Emulation. The in-circuit emulation subsystem enables the software developed on the microcomputer to be debugged before the hardware prototype is completed and even while the prototype is nonexistent. Resource-lending capabilities enable the software to be tested in the prototype hardware before it is completed. After the hardware is complete, the emulation subsystem allows total integration and testing to occur in a real-time environment. The subsystem consists of a trigger or breakpoint module, a monitor module, a user pod controller, a user pod, and a Z80A emulator CPU.

Hardware trigger capability enables searching for a specific condition while the software is executing in real time, and executing breaks when detected. The detection can also be used to generate a sync pulse to trigger other instruments, such as oscilloscopes or logic analyzers used in the debug process.

Monitoring Functions. The emulation subsystem provides a means of monitoring the interaction of the microprocessor with the target design. A special high-speed trace memory records the microprocessor's bus activity, while running the software in real time. The contents of the memory may then be dumped on the console after emulation has been halted for subsequent debug. The output of the trace memory can be displayed in three available formats. The user may qualify the inputs to the trace memory to select the specific type of bus cycle to be recorded, such as a memory write or an I/O operation.

Resource Sharing Functions. The ZDS-1/40 system allows the user to borrow memory resources so that testing can begin even before the hardware is complete.

The system provides a memory mapping mechanism, whereby the user can describe the addressable memory space of the microprocessor. This memory space is divided into blocks, each containing 1024 bytes of contiguous memory addresses. These blocks may be described to exist in the user's prototype, in the development system memory, or not to exist at all. All commands executed to examine or modify memory are qualified by the mapping mechanism.

The mapping mechanism also allows hardware write protection of any block. Any attempted write to a write-protected block will be reported as a write violation and will terminate program execution without causing overwrites to the block. The nonexistent memory feature enables the user to declare blocks of memory nonexistent. Any attempt to access these blocks will immediately terminate program execution with a nonexistent memory violation message.

Emulation occurs by removing the Z80 or Z80A microprocessor from the prototype and replacing it with the Z80A Emulator CPU of the development system. This emulator is connected to and controlled by the emulation subsystem. Monitoring and resource lending capabilities provided by the emulation subsystem also simplify the development process.

Emulation Functions. The emulation subsystem provides several functions extremely useful to software and hardware designers: 1) control of the microprocessor in the hardware prototype; 2) the ability to monitor the bus signals of the microprocessor and record them; and 3) the ability to lend development system resources to the user's hardware prototype.

Control Function. The cable connection between the user's prototype and the development system allows start/stop control of the Z80A CPU Emulator. This feature enables the user to execute the software in a normal run mode, single-step the software, or execute multiple instructions. When the emulator is idling or not running the user's software, it generates the necessary refresh timing signals to keep dynamic memory in the prototype alive. Control of the microprocessor also allows the user to examine or modify CPU registers, memory or I/O devices.

SPECIFICATIONS

SW HOST

CPU
Z80 CPU and Z80A Emulator CPU

Memory
64K bytes (3K EPROM, 1K static RAM, 60K dynamic RAM)

Word Size
8 bits (1 byte)

Clock Rate
2.5 MHz crystal-controlled

Interrupts
Three modes including vectored, nonvectored and nonmaskable

Option Card Slots
Five (5)

Floppy Disk Storage

Capacity	300,000 bytes/drive
Type	Single-sided, single density, hard-sectored
Maximum Capacity	600,000 bytes (dual drives)
Transfer Rate	260K b/s
Average Latency	83 ms
Track-to-Track Seek	10 ms
Average Access Time	250 ms
Physical Sectors	32 sectors/track, 77 tracks

IN-CIRCUIT EMULATOR

Clock Rate	4 MHz
Trigger	Break on address
Real-time trace module	256 x 36 bits wide, high-speed static RAM
Emulation cable (including pod)	6 ft. (1.82 m)

PHYSICAL

System

Height	Width	Depth	Weight
10.0 in. (25.4 cm)	19.0 in. (48.3 cm)	16.0 in. (40.6 cm)	35.0 lbs. (15.9 kg)

Disk Unit

Height	Width	Depth	Weight
10.0 in. (25.4 cm)	19.0 in. (48.3 cm)	16.0 in. (40.6 cm)	35.0 lbs. (15.9 kg)

POWER

System

Frequency	Voltage	Current
50 Hz	110 Vac	1.5 A
50 Hz	220 Vac	0.7 A
60 Hz	110 Vac	1.5 A

Disk Unit

Frequency	Voltage	Current
50 Hz	110 Vac	1 5 A
50 Hz	220 Vac	0.7 A
60 Hz	110 Vac	1.5 A

ENVIRONMENTAL

Operating Temperature	Storage Temperature	Relative Humidity
0° to 40°C	0° to 85°C	20 to 80% noncondensing

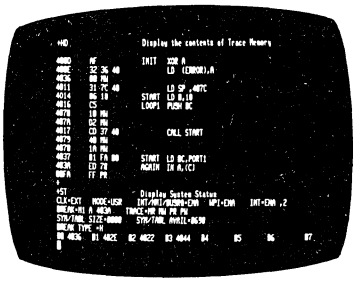
ORDERING INFORMATION

Part No.	Description
05-6013-05	ZDS-1/40 Development System (60 Hz).
05-6013-06	ZDS-1/40 Development System (50 Hz).

ZILOG ANALYZER PACKAGE (ZAP)

EMULATION SOFTWARE

Product Overview. The Zilog Analyzer Program is a sophisticated software module used to operate the emulation hardware of the ZDS-1/40 Development System. ZAP provides full control over the emulation hardware for quick and easy debugging of Z80- and Z80A-based designs. This module allows the user to inspect the microprocessor and to interact with the prototype system. The registers, user or system memory, and I/O ports may be interrogated and controlled. Control of program execution, interrupt, and Direct Memory Access (DMA) activity is also provided. These features, combined with full symbolic debugging and disaaembly capability, provide powerful debugging tools for the Z80 microprocessor.

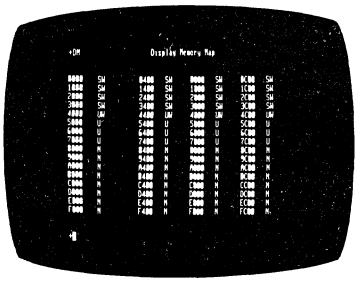


Functional Description. The Zilog Analyzer Program is a disk-resident program used to control the Z80/Z80A emulation hardware of the ZDS-1/40 Development System. It provides an interface between a command source and the emulation hardware. Commands may therefore be supplied from the disk file system (command file), system console device or a control program. This provides flexibility in hardware debugging as well as testing applications.

ZAP provides a complete spectrum of commands and data formats to enable prototype hardware and software to be quickly integrated and debugged. A simple command syntax, using abbreviated command words, provides visibility into the Z80's registers, user memory, system memory, user I/O ports, and CPU status. Microprocessor registers may be displayed and altered individually, or the complete register set may be displayed. Commands for accessing user or system memory include FILL, SET, DISPLAY, and ALTER. A block compare command is also provided for comparison of a given string with memory. All memory data may be disassembled to reflect the actual source code mnemonics and symbolic references if the symbol table for the code is available.

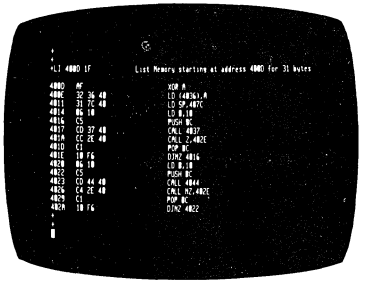
Emulator Start/Stop Control. The use of ZAP and the ZDS-1/40 emulation hardware allows the user to control the start up and shut down of the Z80A Emulator CPU. Emulation is initiated by the GO command and continues until one of the following conditions occurs:

- One of eight different software breakpoints is encountered
- Hardware breakpoint compare
- Operator intervention (manual break)
- Bad clock detection (in target system)
- Non-existent memory access
- Write-protect violation



The user may select single step or multi-step execution of the program under test. This enables the registers to be examined after each step operation. In multi-step mode a group of instructions may be executed in real time. Any group of up to 255 instructions may be multi-stepped before stopping the emulator.

Memory Mapping. The memory mapping capabilities of the ZDS-1/40 Development System are easily manipulated by ZAP. Blocks of memory, each containing 1024 contiguous bytes, may be assigned to exist in the user's system, in the development system at the normal address, in the development system with a translated address, or not to exist at all. In addition, these blocks of memory may be hardware write-protected to assist in the debugging task and prevent accidental destruction of data.



Disassembly Capability. User memory, system memory, and the trace memory of the development system may be displayed in the hexadecimal or disassembled format. In disassembled format, the instructions are displayed in both hexadecimal machine code and assembly language mnemonics.

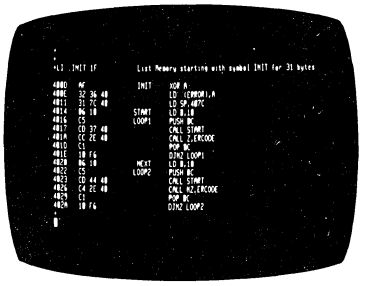
Symbolic Debugging Capability. Symbol tables for each program module may be loaded individually, or the entire symbol table for the program may be loaded. The user may define local symbols to assist in the debugging process.

Symbols are loaded into development system memory and are automatically hardware write-protected to prevent accidental destruction. A maximum of 29K bytes of system memory may be used for the symbol table: this equals approximately 3000 symbols. Since development system memory may be shared with the user's prototype system, the maximum symbol table size is a function of the number of blocks allocated for use in the user's prototype hardware.

The use of symbols in place of numeric values in the ZAP command syntax, teamed with disassembly, enables the user to have an electronic listing of the program under test. This allows the user to concentrate on the debugging task instead of having to struggle with the development system software.

Command File Capability. The Zilog Analyzer Program is structured to accept command input from several sources: console, file, or program. This capability is important in the debugging process, the training process, and even the manufacturing test process.

In the debugging process, commonly used commands for establishing the memory map, enabling interrupts, and



loading program modules may be placed in a disk file and executed. This allows a series of necessary operations to be performed with a minimum number of keystrokes. It also insures that the system will be initialized the same way, no matter how many individuals are using the system.

The same technique may be used for training new users of the ZAP command structures. Tutorial files can be created to execute the various system commands and illustrate the results. An example of this is ZAP TUTOR, a software training package included with ZAP to acquaint the user with the commands and their use.

In a manufacturing test operation, the user may create software which formats command parameters for ZAP and pass these using a CALL to ZAP. The ZAP software will perform the requested operation and return the results to the calling program. This enables the user to diagnose designs using the emulation hardware controlled by applications software.

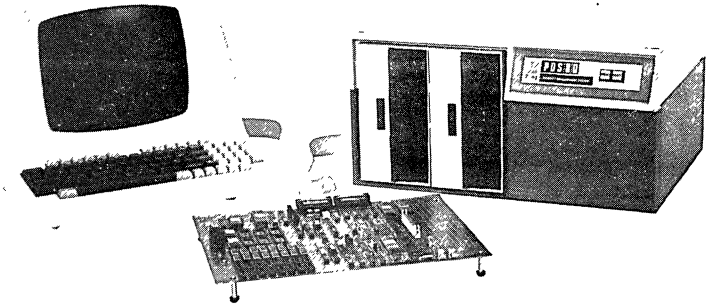
PDS 8000™ Development Systems

Zilog

Product Description

March 1981

- **SUPPORTS entire family of Zilog microprocessors—Z8™, Z80®, and Z8000™.**
- **SPEEDS program development with high-level, structured assembler.**
- **EXTENDS system capability with an intelligent CRT console.**
- **EXPANDS to allow full emulation and debugging.**
- **PROVIDES either floppy or cartridge hard disk back-up data storage.**



FUNCTIONAL DESCRIPTION

System Hardware. The PDS 8000 Series consists of several models of single-user systems for the design, development, and debug of Zilog microprocessor-based systems. The PDS 8000 is a Z80-based microcomputer system with 64K bytes of RAM, a disk controller, parallel printer interface, and an intelligent CRT console.

A variety of configurations allows the selection of unbundled hardware options or a complete package of hardware, development software, and a development module. A choice of one of the following types of disk drives is available in the standard PDS 8000 configuration.

- A dual, floppy-disk drive unit with 600K bytes of hard-sectored storage capacity.
- A cartridge disk drive with 10M bytes storage capacity (5M bytes removable).

Dual Floppy Disk Drive. The floppy disk drive is interfaced to the microcomputer via the Z80 Memory Disk Controller (MDC), which provides the Z80 microcomputer with all the data formatting required for reading and writing onto the floppy disks from RAM storage. Disk read/write accuracy is ensured by 16-bit CRC-code circuitry. Wait control logic synchronizes the speed of the CPU to the disk read or write speed by setting the CPU wait line to an active state until a complete byte of data is assembled. The MDC employs a parallel interface to control the disk drives and to provide status information to the CPU. The MDC also provides 48K bytes of dynamic RAM memory for programs or data storage.

Cartridge Disk Drives. The cartridge disk drive is interfaced to the microcomputer via the Hard Disk Interface. The Hard Disk Interface is an intelligent disk control unit that provides control for up to four cartridge disk drives. It consists of a Z80 Direct Memory Access (DMA) peripheral device, software-defined file data buffer, and read/write control logic on four PC boards. Primary control of hard disk operations is exercised by a resident processing system, which includes a Z80 microprocessor CPU, 16K bytes of high-speed static RAM, and CPU support circuitry.

The Hard Disk Interface to the host is comprised of a pair of Z80 Serial Input/Output (SIO) chips and a Z80 Counter/Timer Circuit (CTC). One SIO device connects to the hard disk interface bus and the other SIO device connects to the host system bus.

The Hard Disk Interface executes high-speed transfers of disk files between the disk drive and system memory with all the necessary DMA control intelligence provided by the interface itself. The basic responsibility of the microcomputer host is to download the file system software to the Hard Disk Interface memory, and to output disk I/O requests (also performed under DMA control), thus minimizing the host microcomputer's interaction with the Hard Disk Interface.

An automatic bootstrap feature allows file system software to be loaded automatically. The hard disk processing subsystem serves as the central control for all hard disk control functions, including:

- Processing messages received serially from the host CPU via the SIO.
- Issuing commands such as SEEK, READ, WRITE, etc., to the selected drives.
- Reading the status of a selected drive.
- Initializing DMA circuits in preparation for disk file transfers.
- Servicing interrupts from hard disk controller elements.
- Performing various housekeeping tasks required by software.

System Software. The PDS 8000 System provides all the necessary software to handle software development tasks, from inputting source code to printing listings and creating EPROM's.

RIO Operating System. The PDS 8000 utilizes Zilog's field-proven RIO Operating System for the creation, editing, assembly, and debugging of software. RIO, Version 2, with relocatable modules and I/O management, is a general-purpose computing system with architecture designed to facilitate the development process. RIO provides straightforward linking to various system routines and enables expansion of system features to meet the particular needs of individual user. RIO is composed of the following elements which aid in the development process:

Operating System Executive. The RIO Executive maps requests of operations on logical units to specific device-handling programs. Commands may be issued to the operating system from the system console or by an executing program. Any number of user-defined commands may be added to the system. Command sequences may be recorded in files and executed as a group. The Executive manages the allocation of memory blocks.

Relocating MACRO Assembler. The Relocating MACRO Assembler offers relocatable or absolute object code format with external symbol references and global symbol definitions, MACRO's and conditional assembly. The Assembler pages the symbol table, permitting assembly of arbitrarily large programs in standard memory. It also includes a directive permitting additional files to be merged with the source at assembly time.

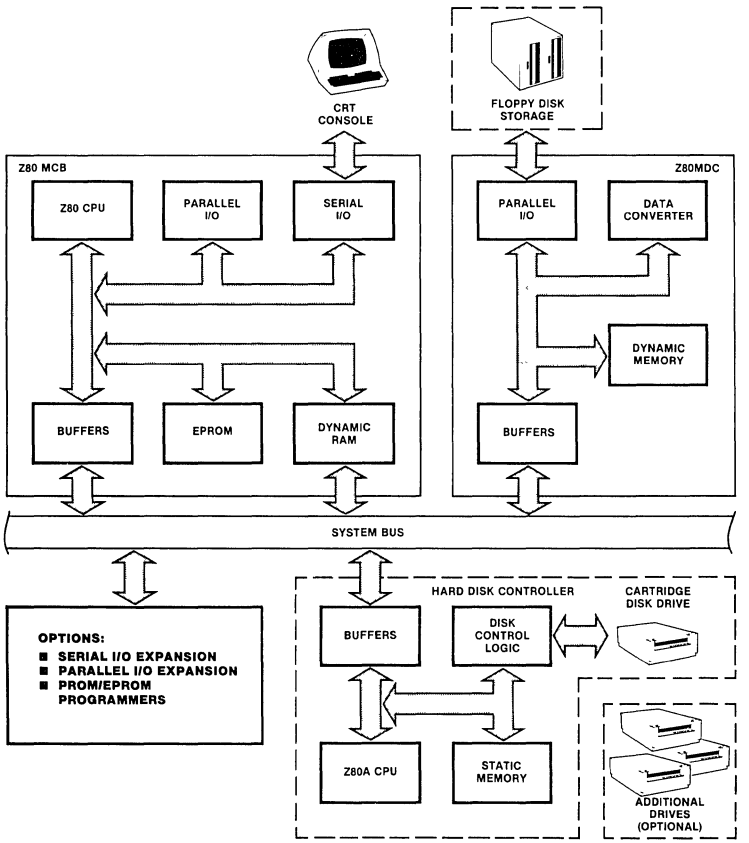


Figure 1. PDS 8000 Development System

Linker. The Linker assigns absolute addresses to program modules, resolves external references, permits overlays, and produces a load memory map with a global address table.

Text Editor. A line-oriented Text Editor pages work space so that files of any size can be edited and also provides automatic file backup and access to other disk files during editing. String matching allows for locating and modifying lines within a file.

PROM Monitor. The PROM Monitor bootstrap loads for easy system entry, supports a full machine-language debug package, and includes low-level device handlers for system console and disk.

ZDOS II File Manager for Floppy Disk Configurations. The ZDOS II File Manager allocates all disk space automatically on an "as needed" basis and supports sequential access to disk files or direct access to any specific disk address. A directory provides an index for data, accessed by calling a "hierarchical linked list." Logical record lengths range from 128 to 4096 bytes. The File Manager also assigns attributes for protection and privacy.

DFS File Manager for Cartridge Disk Configurations. The DFS File Manager provides features similar to the ZDOS II File Manager. It resides in the Hard Disk Controller to save memory working space for user software.

Utilities. In addition to the RIO Operating System, the PDS 8000 System includes a utilities package that contains programs such as peripheral drivers, Z-PROG for programming 2708 and 2716 EPROM's for the Z80, and a memory test routine. All these programs are modular and relocatable.

Processor-Oriented Support. To enhance the development capability of the PDS 8000 Series of systems, Zilog also provides specific software packages and development tools to aid the microprocessor system designer.

Z8000 Support. For Z8000-based system designs, the Z8000 Software Development Package (SDP) provides the necessary tools to aid in software development. Utilizing PLZ, Zilog's high-level language, the Z8000 SDP includes a Cross Assembler, Linker, and PROM programming utility.

For a tried and tested environment to run Z8000 code, the Z8000 Development Module is available. Providing support for either the Z8001 or Z8002, the Development Module is a single-board computer with RAM, I/O and monitor/debug firmware. The Z8000 Development Module is a convenient tool to evaluate Z8000 CPU performance, as well as a first-level software debug tool for use early in the design process.

For real-time emulation of either the Z8001 or Z8002, the Z-SCAN 8000 Emulator is available. Operable both stand-alone and with a host system, Z-SCAN 8000 makes possible software and hardware integration with real-time breakpoint, monitor/debug software, mappable memory, and an interactive user interface.

Z80 Support. The Z80 Software Development Package (SDP) affords the designer a high-level PLZ Compiler, a Cross Assembler, Linker, and Code Generator.

Z8 Support. The Z8 Software Development Package (SDP) includes a PLZ Cross Assembler, Linker, and PROM programming utility for Z8-based designs. The Z8 Development Module is a single-board computer that provides a ready environment to evaluate and debug Z8 code. With 2K bytes of static RAM, breakpoint, and PROM-based monitor, the Z8 Development Module can operate stand-alone with a CRT or with a host CPU.

SPECIFICATIONS

MICROCOMPUTER

Microprocessor
Z80 CPU

Interrupts
Three modes—maskable vectored, maskable non-vectored and nonmaskable

Memory
60K bytes of dynamic RAM
3K bytes of EPROM

Option Card Slots Available
Floppy disk configuration—7
Hard disk configuration—9

Word Size
8 bits (1 byte)

I/O Channels
One serial I/O port with EIA RS-232C interface
Two parallel I/O ports for printer interface

CRT CONSOLE

Screen Format
24 lines by 80 characters

Attributes
Reversed, blinking and blanked fields

Screen Size
12-inch diagonal, non-glare screen

Keyboard
Standard typewriter keyboard
15 numeric keys
5 separate cursor control keys

Character Set
128 ASCII code, 32 control codes

Interface
Standard—RS-232C/CCITT.V24
Auxiliary—EIA RS-232C

FLOPPY DISK DRIVE

Capacity
300K bytes/drive

Data Transfer Rate
250K bits/second

Physical Sectors
32 sectors/track, 77 tracks

Type
Single-sided, single density, hard sectored

Average Latency
83 milliseconds

Maximum Capacity
600K bytes (dual drives)

Track-to-Track Seek Time
10 milliseconds

CARTRIDGE DISK DRIVE

Capacity
10M bytes/drive (5M bytes formatted storage, 5M bytes removable)

Data Transfer Rate
2.5M bits/second

Type
Top-loading cartridge

Average Latency
12.5 milliseconds

Access Time
60 milliseconds (maximum)

Maximum Capacity
40M bytes (four drives)

Track-to-Track Access Time
7.5 milliseconds

Physical Sectors
12 sectors/track, 402 tracks

SPECIFICATIONS (Continued)

Power

	AC	Single-Phase
Microcomputer	110,220VAC (50 Hz)	550VA (60 Hz)
Floppy Disk Drive	15,230VAC (50 Hz)	52VA (60 Hz)
Cartridge Disk Drive	110,230VAC (50 Hz)	110VA (60 Hz) start/stop, 400VA typical

Physical

	Height	Width	Depth
Microcomputer	30.00" 76.2cm	15.60" 55.9cm	20.20" 76.2cm

Physical (Continued)

	Height	Width	Depth
Floppy Disk Drive	9.50" 24.1cm	4.60" 11.8cm	14.25" 36.2cm
Cartridge Disk Drive	8.75" 22.2cm	19" 48.3cm	29.25" 74.3cm

Environmental

	Operating Temperature	Storage Temperature	Relative Humidity
Microcomputer	0°-40°C	0°-85°C	20%-85%
Floppy Disk Drive	5°-50°C	0°-85°C	20%-80%
Cartridge Disk Drive	15°-30°C	10°-65°C	5%-85%

ORDERING INFORMATION

Floppy Disk-Based

Part No.	Description
05-6102-01	PDS 8000/05 Development System (60 Hz). Includes Z80 Microcomputer, 64K bytes dynamic RAM, 3K monitor, printer interface, editing-type video terminal w/line-drawing capability, dual floppy disk and RIO Operating System. (115 VAC)
06-6102-02	PDS 8000/05 Development System (50 Hz). Same as 05-6102-01 except 230 VAC.
05-6102-04	PDS 8000/05-1 Development System (60 Hz). Includes Z80 Microcomputer, 64K bytes dynamic RAM, 3K monitor, printer interface, serial interface, dual floppy disk and RIO Operating System. (115 VAC)

Floppy Disk-Based (Continued)

Part No.	Description
05-6102-03	PDS 8000/05-1 Development System (50 Hz). Same as 05-6102-04 except 230 VAC.
05-6104-01	PDS 8000/15 Development System (60 Hz). Includes Z80 Microcomputer, 64K bytes dynamic RAM, 3K monitor, printer interface, editing-type video terminal w/line-drawing capability, dual floppy disk, Z8000 Development Module, Z8000 SDP Software Development Package, and RIO Operating System. (115 VAC)
05-6104-02	PDS 8000/15 Development System (50 Hz). Same as 05-6104-01 except 230 VAC.

Cartridge Disk-Based

Part No.	Description
05-6105-01	PDS 8000/20 Development System. Includes Z80 Microcomputer, 64K bytes dynamic RAM, 4K monitor, printer interface, editing-type video terminal w/line drawing capability, 10M byte disk subsystem, 30-inch high enclosure, RIO Operating System. (115 VAC)
05-6105-02	PDS 8000/20 Development System. Same as 05-6105-01 except 230 VAC.
05-6105-04	PDS 8000/20-1 Development System (60 Hz). Includes Z80 Microcomputer, 64K bytes dynamic RAM, 4K monitor, printer interface, 10M byte disk subsystem, 30-inch high enclosure, RIO Operating System. (115 VAC)
05-6105-03	PDS 8000/20-1 Development System (50 Hz). Same as 05-6105-04 except 230 VAC.

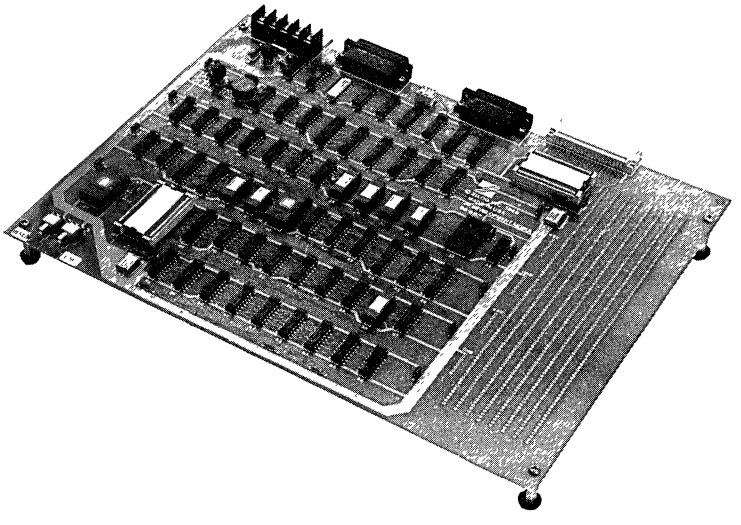
Z8™ Development Module

Zilog

Product Description

March 1981

- **Two Z8-02 Devices Offer Complete Configuration Choice for Any Application.**
- **2048 Bytes Static RAM for Convenient Execution and Debug of User Code.**
- **On-board 2716 Socket to Test User Code in EPROM Without Additional Hardware.**
- **As Many as 2048 Hardware Breakpoints on Address Compare Cover the Entire Internal ROM Space.**
- **Versatile Monitor Software for Debugging, Register and Memory Manipulation, and File Upload and Download.**
- **'Transparent' Operation Allows Software Development Without Disconnecting from CRT and Host. Industry-Standard Interface Compatible with Most CRT Terminals and Development Hosts.**
- **Wire-Wrap Area for Prototyping.**



OVERVIEW

The Z8 Development Module is a single-board microcomputer system specifically designed to assist in the development and evaluation of hardware and software designs based on the Z8 microcomputer. It allows system prototyping in hardware with the Z8-02 prototyping device, thereby developing code that will eventually be mask programmed into the Z8 on-chip ROM.

Two Z8-02 devices on the Z8 Development Module provide flexibility: one serves as a controller while the other is totally user-definable. All user ports on the second Z8-02 are unconfigured and available to suit any application.

To simulate the final mask-programmed version on which user code resides, 2048 bytes of high-speed static RAM are available for executing and debugging code. An on-board EPROM socket allows the user to substitute EPROM for static RAM. This enables the user to test PROM after software development and debug without building special hardware.

The EPROM-resident monitor software offers debugging features, register and memory manipulation, as well as a convenient means to upload and download software between the host and user RAM space.

The Development Module connects to the CRT terminal and host system via two on-board standard RS-232C serial ports and is physically located between the CRT and host. A simple command makes the Development Module transparent in the serial path to allow software development without disconnecting from the CRT and host.

The Development Module can operate stand-alone for simple debugging operations or it can interface directly to a host development system such as the Zilog ZDS-1 or PDS 8000™ Series for software development and file storage.

Twenty square inches of wire-wrap area with conveniently located 5 V and ground points are provided near the user Z8-02 for prototyping.

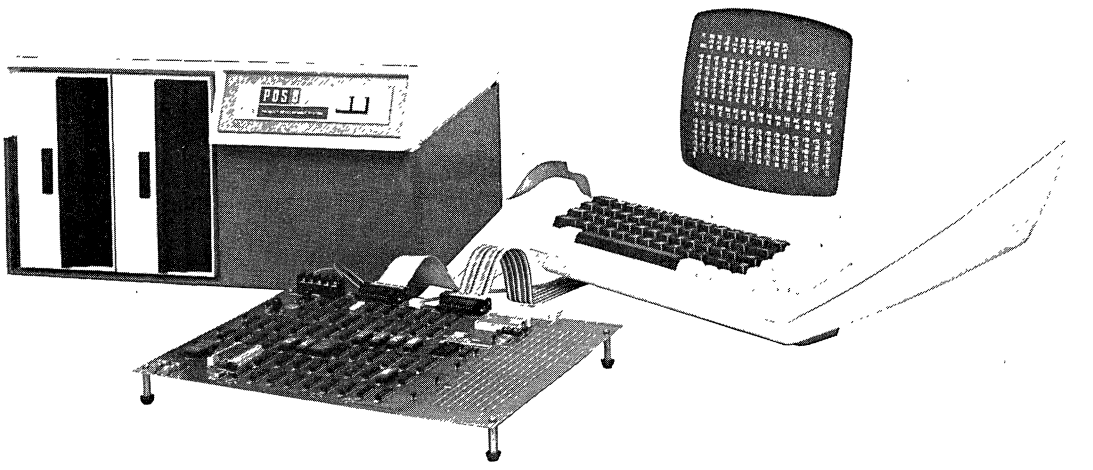
FUNCTIONAL DESCRIPTION

Hardware. Two Z8 microcomputer units designated the Monitor MCU and User MCU are at the heart of the Z8 Development Module. The Monitor MCU controls operation of the User MCU and the monitor/debug software. The monitor/debugger resides in 4K bytes of EPROM. Hardware breakpoint logic provides a maximum of 2048 breakpoints. Single stepping and software trace capabilities are also available.

The User MCU is a Z8-02 controlled by the Monitor MCU via internal address/data and control lines brought out to external pins. This effectively leaves all ports on the User MCU unconfigured and available to the user. The 2K bytes of static RAM on the 'internal' bus are for user code that may be executed by the User MCU. Execution is in real time at full processor speed. Both MCUs utilize 7.4 MHz crystal oscillators, the outputs of which are divided internally to provide 3.7 MHz clocks.

In addition to wire-wrap area, a 40-pin header (3M type 3495-1002) for the User Z8 can connect to a ribbon cable with a 40-pin plug that may plug into a target system. Bus driver logic may be added on the wire-wrap area for basic emulation capability. Two switches, 'Mode' and 'Reset', provide a means to re-enter the Monitor and reinitialize the system, respectively. Baud rate from 110 to 19200 may be selected with an on-board 4-element DIP switch.

Software. The monitor/debug program, residing in 4096 bytes of EPROM, includes debug, input/output, control and host interface commands. The commands are grouped into four major functional blocks: monitor, debug, manipulation and file commands.



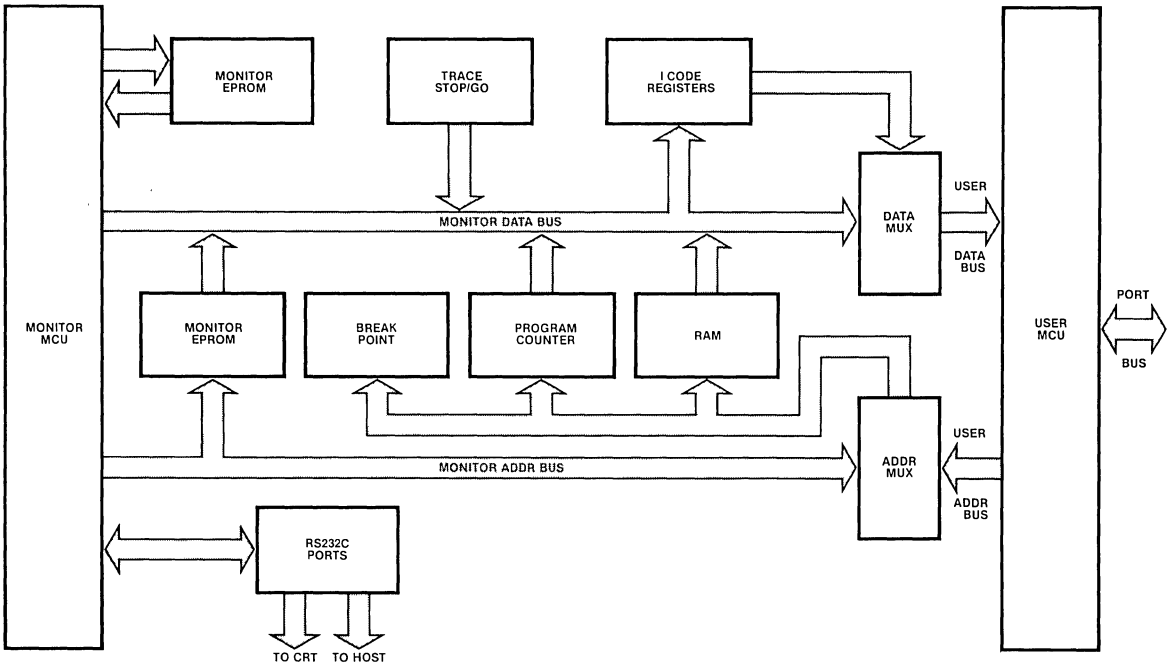
Z8 Development Module conveniently connects to both the CRT and PDS 8000 Development System.

Monitor Commands. This group of commands controls execution of the User MCU, monitors user interrupts and transfers controls from the monitor to the host system.

GO <ADDRESS>	Causes User MCU to execute program disallowing further debug until a BREAK or HALT command is encountered.
HALT	Halts program execution of the User MCU.
QUIT	Returns control to the host system and enters the 'transparent' mode.
INTERRUPTS [E/D]	Enables or disables all user generated interrupts. Note: All user interrupts are automatically disabled when a breakpoint is encountered. It is necessary to reenable such interrupts by this command.

Debug Commands. This group of commands allows the user to debug code by tracing through code and setting breakpoints and jumps to specified locations within the 'internal' ROM space.

BREAK <ADDRESS>	Sets a breakpoint at the specified address.
KILL [<ADDRESS>]	Clears the breakpoint at the specified address.
JUMP <ADDRESS>	Allows the User MCU to jump to a specified address anywhere within the internal ROM space, by changing the value of the program counter.
NEXT [<n>]	Causes execution of n instructions of the User MCU and then halts the User MCU.
TRACE	Causes single step execution of the User MCU. Every instruction executed is output to the console.



Z8 Development Module Block Diagram

Manipulation Commands. The manipulation commands display and alter registers and memory. This group may be subdivided into two categories: register manipulation and memory manipulation.

Register Manipulation

REGISTER [<REG NUMBER>] [<NEW REG VALUE>]]	Allows examination and modification of the User MCU registers.
WORKING REGISTERS	Displays contents of the 16 working registers of the User MCU.
PHILL <STARTING REGISTER> <NUMBER OF REGISTERS> [<DATA BYTES>]	Stores the sequence of DATA BYTES into User MCU registers beginning at the STARTING REGISTER and is copied as many times as necessary for the NUMBER OF REGISTERS specified.

Memory Manipulation

DISPLAY [<STARTING ADDRESS>[<n>]]	Allows display and modification of user memory contents for n number of bytes.
SET <ADDRESS> <LENGTH> [<DATA BYTES>]	Allows a sequence of data bytes beginning at the ADDRESS specified to be written into user memory.
FILL <STARTING ADDRESS> <LENGTH> [<DATA BYTES>]	Stores the sequence of DATA BYTES into user memory beginning at the starting ADDRESS and is copied as many times as necessary for the LENGTH specified.
MOVE <SOURCE ADDRESS> <DESTINATION ADDRESS>[<n>]	Moves contents of a user memory block from a source address to a destination address for a length of n bytes.
COMPARE <ADDRESS 1> <ADDRESS 2>[<n>]	Compares two blocks of user memory data, one beginning at ADDRESS 1 and the other at ADDRESS 2 for n bytes.

File Commands. The File group enables the user to upload and download programs to and from the host system.

LOAD <FILE NAME>	Downloads a file to user memory starting at the low address of the file and continuing until the entire file is transferred.
UPLOAD <FILE NAME> <ADDRESS 1> <NUMBER OF BYTES> [<ENTRY ADDRESS>]	Creates a RIO file image of user memory, beginning at ADDRESS 1, creating default length records, and imaging memory for the specified number of bytes.

Note: The following notation is used in the command description.

- < > Enclose descriptive names for the quantities to be entered, and are not actually entered as part of the command.
- [] Denote optional entries in the command syntax.
- | Denotes "or."

SPECIFICATIONS

Central Processor
Monitor MCU: Z8-02 (64-pin package)
User MCU: Z8-02 (64-pin package)
Clock Rate: 3.7 MHz

Memory
Monitor: 4K bytes of EPROM
User: 2K bytes of static RAM
User: Wired socket for EPROM to substitute for static RAM

Input/Output
Two RS-232C ports to CRT terminal and host system

Baud Rate
Switch selectable from 110 to 19200 baud

Breakpoint
2048 max., valid for Address Compare, applicable to user 'internal' memory only

Control
Mode and Reset switches

Power
+5 V, 1.4 A

Physical

Wire Wrap Area	20 sq. in. 0.036" dia. plated-through holes on 3/32 in. centers
Height	1.75 in. (4.76 cm), including standoffs
Width	14.5 in. (35.6 cm)
Depth	11.0 in. (29.9 cm)

ORDERING INFORMATION

Part No.	Description
05-6158-01	Z8 Development Module. Includes one serial interface ribbon cable and reference manual.

Systems recommended for use with above:

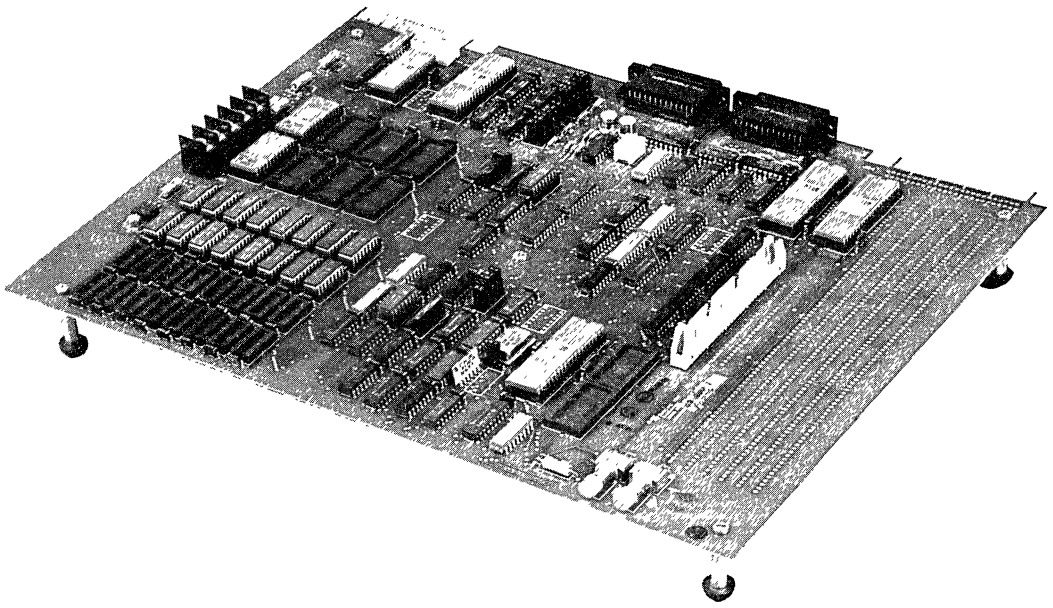
Description	Prerequisites
ZDS-1 Series Development Systems	Z8 Software Development Package
PDS-8000 Series Development Systems	Z8 Software Development Package

Z8000™ Development Module

Zilog

Product Description

March 1981



- **Z8001/Z8002 CPU Evaluation and Debug Support**
- **16K Words Dynamic RAM (Expandable to 32K for User Code Execution and Debug**
- **32 Programmable I/O Lines**
- **EPROM Monitor and Debugger**
- **Transparent Operation Allows Software Development without Disconnection from CRT and Host System**
- **RS-232C Standard Serial Interfaces Compatible with Most CRT Terminals and Development Hosts**
- **Wire-wrap Area for Prototyping**

OVERVIEW

The Z8000 Development Module is a complete, single-board microcomputer that is used as a tool for the evaluation and debug of Z8000-based microprocessor systems. The Development Module is used in the first stages of the design and development process, not only as a tool for evaluating Z8000 microprocessor capabilities, but also as an environment in which code can be executed and debugged.

Evaluation. The Development Module provides a ready-made environment in which the user can execute software unique to his Z8000-based application, evaluate the CPU's performance, and then reach a realistic decision about its suitability for a specific application.

Software Debug. In addition to use as an evaluation tool, the Z8000 Development Module can be used to debug and modify user code. For the software designer, the Development Module is a real Z8000 environment in which he can execute code and carry out fairly extensive debugging. For the hardware designer, the Development Module is an example of Z8000 hardware design which provides special hooks and wire-wrap facilities to strap on additional logic.

FUNCTIONAL DESCRIPTION

Z8000 code developed on a software host may be downloaded serially to the Development Module RAM area via a serial port, and executed and debugged under EPROM monitor control. Once the system is connected, no further disconnection is necessary as the module has two serial ports (one connected to a host and the other connected to a CRT terminal). A simple software command makes the development process transparent in the serial path, thereby allowing direct communication between the host and terminal. The serial RS-232C interfaces allow virtually any software development host and CRT terminal to be used. For PROM-based code testing, the development module is self-contained and can operate stand-alone with a CRT terminal, since the host is only required for storage of user code on disk.

A variety of jumper areas and switches permit the selection of clock rates ranging from 2.5 to 3.9 MHz; the use of 2708, 2716, or 2732 EPROMs; the use of 4K or 16K RAMS; serial interface to modem, terminal, or teletype; I/O port addressing; and baud-rate selection from 110 to 19200 baud.

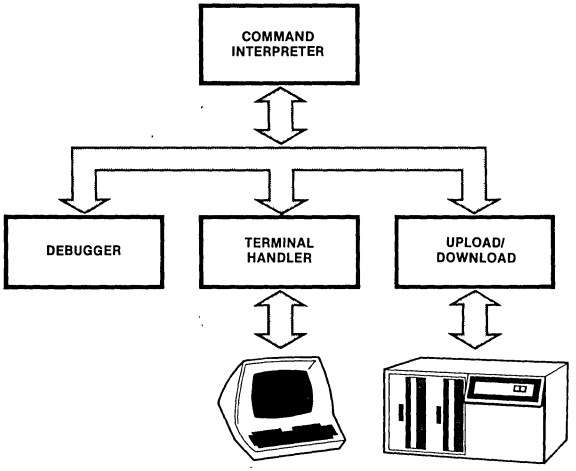


Figure 1. Monitor Block Diagram

Hardware. The Z8000 Development Module is available in two versions: one supports the segmented Z8001 microprocessor; the other supports the non-segmented Z8002 microprocessor.

Z8001 Development Module. The Z8001 Development Module consists of a Z8001 CPU, 16K words of dynamic RAM (expandable to 32K words), 4K words of EPROM monitor (user-expandable to 8K words), a Z80A SIO providing dual serial ports, a Z80A CTC peripheral chip providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire-wrap area for prototyping hardware.

Z8002 Development Module. The Z8002 Development Module consists of a Z8002 CPU, 16K words of dynamic RAM (expandable to 24K words), 2K words of EPROM monitor (user-expandable to 8K words), a Z80A SIO device providing dual serial ports, a Z80A CTC peripheral device providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire-wrap area for prototyping.

Software. The monitor software (*Figure 1*) contained in EPROM (4K words for the Z8001 and 2K words for the Z8002) provides debugging commands, I/O control and host interface. It consists of a terminal handler, command interpreter, debugger and upload/download handler.

Terminal Handler. A Terminal Handler provides interface to the console device to facilitate output to a display or printing mechanism and input from a standard ASCII keyboard.

Debugger. The Debugger provides a basic set of debug commands to allow the user to start and stop program execution, display and alter CPU registers, flags or memory, and trap instruction sequences.

Command Interpreter. The Command Interpreter scans console inputs, ensures command validity and passes to other software modules in the monitor.

Upload/Download Handler. The Upload/Download Handler provides an interface between the serial connection and the host computer, the command interpreter and the memory resources of the Z8002 Development Module. It formats and interprets asynchronous data streams to and from the host and provides error checking and recovery for the serial interface (see *Figure 2*).

Memory Organization. *Tables 1 and 2* show the memory maps for the two versions of the Development Module. The organization of ROM and RAM in both the segmented and nonsegmented modes is indicated.

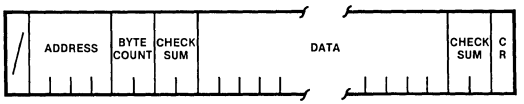


Figure 2. Serial Data Format

Address (Hex)	Memory
0000 0FFF	Monitor EPROM
1000 3FFF	User EPROM (User Installed)
4000 BFFF	Standard RAM
C000 FFFF	Expansion RAM (User Installed)

Table 1. Z8002 Development Module Memory Map

Segment 0		Segment 1	
Address (Hex)	**Memory**	**Address (Hex)**	**Memory**
0000 1FFF	Monitor EPROM	0000 3FFF	Expansion RAM (User Installed)
2000 3FFFF	User EPROM (User Installed)	4000 FFFF	Unused
4000 49FF	Monitor RAM (Scratchpad Area)		
4A00 BFFF	Standard RAM		
C000 FFFF	Expansion RAM (User Installed)		

Table 2. Z8001 Development Module Memory Map

MONITOR COMMAND SUMMARY

The following notation is used in the command description:

< > Enclose descriptive names for the quantities to be entered, and are not actually entered as part of the command.

[] Denote optional entries in the command syntax.

| Denotes "OR", eg. W|B denotes that either W or B may be used but not simultaneously.

< Prompt sign for the nonsegmented Z8002 monitor.

[Prompt sign for the segmented Z8001 monitor.

The following commands apply when the Z8001 monitor is used. All commands listed remain the same except those that permit reference to segmented addresses as follows:

<address> =
[<segment number>] <offset address>

<segment number> =
"<"<hex number in 7-bit range>">"

BREAK<address> [<n>] — Sets and clears a breakpoint at a given memory address. The option <n> allows specification of the number of occurrences, where n is from 1 to 128. The default is one.

COMPARE <address 1> <address 2><n> — Compares two blocks of memory data beginning with the addresses specified for <n> bytes, where n is from 1 to 128. Errors are reported on the console device.

DISPLAY<address> <n>[L|W|B] — Displays and modifies memory for <n> number of words or bytes. The optional entry allows data to be handled as bytes, words, or long words. The default is words.

FILL<address 1> <address 2><word> — Stores the <word> from memory address 1 to and including address 2.

GO — Begins program execution at the address contained in the current PC; execution is resumed where it was last interrupted. All registers are restored prior to execution.

IOPORT<address> [W|B] — Allows direct communications from the console to a selected I/O port. A word (W) or a byte (B) may be read from the selected port and a word or byte may be sent to the selected port; default is byte.

JUMP<address> — Unconditional branch to the specified address. All registers are restored prior to execution.

MOVE<address 1> <address 2><n> — Moves contents of a memory block from source address <address 1> to destination address <address 2> for <n> bytes.

NEXT[<n>] — Executes the next <n> machine instructions. <n> may be from 1 to 128. If n is omitted, 1 is assumed.

PUNCH<address 1> <address 2> — Punches a copy of memory from address 1 to address 2 on paper tape on the console device. Automatically turns on punch and a null leader is created. Upload/Download section describes the tape format used.

QUIT — Places serial channels into transparent mode. The Z8000 Development Module must be connected to both the Zilog host and the console device, and the Development Module acts as a message switcher.

REGISTER [<register name>] — Allows examination and modification of Z8000 registers. 8-bit, 16-bit or 32-bit quantities may be selected by the appropriate register-naming conventions.

TAPE — Loads memory from paper tape via the console device. The Upload/Download section describes the tape format used.

SPECIFICATIONS

Microprocessor
Z8001 or Z8002 CPU
Clock Rate: 2.5 MHZ or 3.9 MHz

Memory
ROM: 2K or 4K Words (Expandable to 8K Words)
RAM: 16K Words (Expandable to 32K Words)

Input/Output
Parallel: 32 Lines (Two Z80A-PIOs)
Serial: Dual RS-232C or RS-232C and Current Loop (Z80A-SIO)

Note
The user has access to all bus signals to allow custom system expansion into the wire-wrap area off-board.

Interrupts
Maskable Vectored (256), Maskable Non-vectored, Non-maskable, Segmentation Trap

Power
+5 V, 3 A
+12 V, 1 A
-12 V, 0.2 A

Physical

Height	1.75 in. (4.5 cm) Inclusive of Standoffs
Width	14.0 in. (35.6 cm)
Depth	11.0 in. (27.9 cm)
Weight	Approx. 30 oz. (850 gm)

ORDERING INFORMATION

Part No.	Description
05-6168-01	Z8001 Development Module
05-6101-01	Z8002 Development Module
05-6171-01	Z8001 Conversion Kit (converts Z8002 Development Module into Z8001 Development Module)

Systems recommended for use with the above:

Description	Prerequisite
ZDS-1 Series Development Systems	Z8000 Software Development Package
PDS 8000 Series Development Systems	Z8000 Software Development Package

Z8000™ Emulator
Z-SCAN 8000

Zilog

Product Description

March 1981

- **Provides Real Time Emulation up to 4 MHz of the Z8001 and Z8002 CPUs.**
- **Two RS-232C Serial Ports Make It a Peripheral Usable with Most Standard CRTs and Software Hosts.**
- **Transparent Operation Permits Direct Communication Between CRT and Host without Physical Disconnect.**
- **Highly Interactive, Screen-Oriented User Interface Makes Z-SCAN Easy To Use.**
- **Shadow Monitor Removes All Restrictions on Target System Memory Space, Making It Fully Available To the User.**
- **High-Speed Mappable Memory (no wait states) Is Available to Simulate Target System RAM/ROM.**

OVERVIEW

The Z-SCAN 8000 Emulator is an in-circuit emulator that has been designed as a peripheral unit for Zilog's Z8001 and Z8002 16-bit microprocessors. Interfacing via two RS-232C Serial ports to host and CRT terminal, Z-SCAN 8000 can work with Zilog's family of development hosts.

Because it employs a standard serial interface, Z-SCAN 8000 can also be used with virtually any software host system that runs a cross assembler or cross compiler capable of generating Z8000 code. Communication between the host system and Z-SCAN 8000 is with a standard serial format requiring only a simple upload and download utility to operate. For PROM-based target systems, Z-SCAN can operate stand-alone with a CRT terminal because the monitor and debug software is EPROM-resident.

In keeping with Zilog's design philosophy of separating a development system into two identifiable units (the software host and an emulation peripheral), Z-SCAN 8000 fits into three scenarios, making it a highly versatile unit:

- As a peripheral to Zilog's PDS 8000 and ZDS-1 Series of development systems, Z-SCAN 8000 completes the development support package for the Z8001 and Z8002 microprocessors available from Zilog.
- As a peripheral to any development host with the capability of compiling or assembling Z8000 code, Z-SCAN 8000 allows a low-cost emulation capability which precludes substantial reinvestment in a software host system.
- As a stand-alone in-circuit emulator that can operate with most CRT terminals, Z-SCAN 8000 provides simple testing and debugging capability for PROM-based target systems.

SYSTEM FEATURES

User Interface. Z-SCAN 8000 incorporates the use of a two-dimensional screen-oriented user interface which makes it easy to use. Because it is general-purpose in nature, the user interface does not require a customized CRT terminal to operate. The only requirements are that the CRT terminal have screen erase, line erase, and cursor addressing capability.

The objective of the user interface is to provide a screen format with a menu-like approach, which directs the user through the operation of the emulator. The user is aware at all times of where he/she is in the debug process because Z-SCAN 8000 provides the CRT information about system parameters, system resources, current execution, and error messages. When the system is turned on, a bootstrap routine produces a display informing the user of the unit's configuration and requesting the user to define set-up parameters. A menu of display choices shows the user the different capabilities of the system:

- The Memory/I/O command display shows the various memory and I/O manipulation commands which access the target system.
- The Resources display presents the user with the full complement of arguments applicable to emulation of the target system.
- The Execution display shows all the commands and parameters necessary to cause emulation to take place.

At all times, execution of specific Monitor commands is possible, and information on other relevant system parameters and resources is always displayed. This highly interactive user interface makes it possible to use Z-SCAN 8000 without frequent reference to the operating manual.

Shadow Memory. Z-SCAN 8000 is a single, CPU-based system that can be configured to emulate either the Z8001 or Z8002 by simply exchanging the CPU, monitor EPROM, and the emulator cable.

Although the system uses a single CPU for both monitor and emulation functions, no restrictions are placed on the target system memory size. This is because the entire monitor resides in shadow memory and, therefore, does not appear in the target system memory space. This feature also provides the benefit of making future system expansion possible without any hardware redesign.

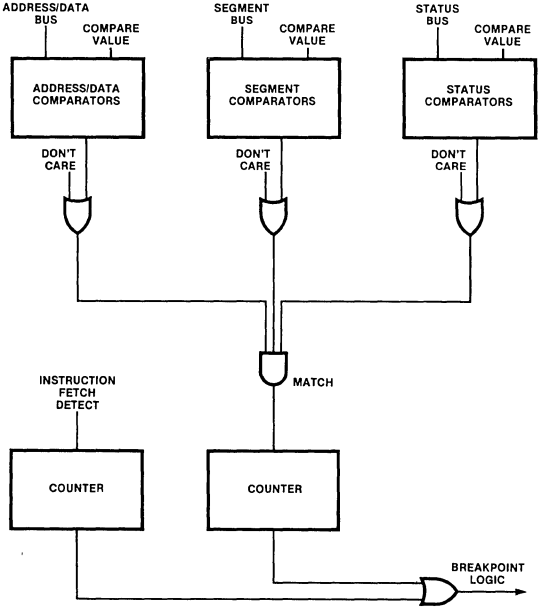


Figure 1. Hardware Trigger Implementation

Hardware Trigger. Z-SCAN 8000 offers the capability of setting breakpoints in three different fields or in a combination of these fields. These are the Address/Data Field, the Segment Field, and the Control/Status Field. A Pass Counter can be set up to a maximum of 255 counts to allow multiple pass triggering. In addition, Z-SCAN 8000 may also be set to break on instruction fetches only (single-step execution), or, by using a Pass Counter, may be set up to a maximum of 247 counts to allow triggering on multiple instruction fetches (multi-step execution).

With these two capabilities, a breakpoint argument can be set up which is on ORed condition allowing for either a break-on-field (or combination of fields) argument or for "n" instruction fetches, whichever occurs first. This ORed situation is convenient when tracing through a program in search of a specific occurrence. A pulse output, providing a trigger pulse on breakpoint match condition is available on the rear panel to trigger auxiliary test instrumentation.

Mappable Memory. Z-SCAN 8000 offers a 4K work block of high-speed static RAM. This block is available to the user to simulate a target system memory block which would typically be ROM. No Wait states are required at 4 MHz. This block is mappable anywhere in the Z8001 and Z8002 address space and can be specified to be Normal Code, Normal Data, Normal Stack, System Code, System Data, System Stack, or Space Independent. Mapping must be done on 4K word boundaries only, and the entire block can be write protected against illegal writes to cause system emulation either to break on such occurrences or continue emulation. An error message appears on the CRT display informing the user of an illegal write.

Software Trace. Z-SCAN 8000 offers a software trace feature which provides insight into target system activity and CPU resources. In the Trace Mode, the system displays the address of the instruction being executed and the contents of the CPU registers (both general-purpose and control) consecutively, covering one full screen format.

For example, displaying the CPU registers associated with every instruction executed just prior to executing a Break is tremendously useful to the user during debug of target system activity.

SPECIFICATIONS

CPU
Z8001 or Z8002 per configuration

Clock Rate
500 kHz-4.0 MHz (external)

I/O
Two RS-232C Serial Ports for CRT and host

Baud Rate
Automatically selected from 50 to 19.2K

Breakpoint
Address, Data, Segment and Address, Control, Address and Control, Data and Control, Segment and Address and Control, Instruction Fetch, OR combination of Instruction Fetch and any Field argument

Mappable Memory
4096 × 16 Static RAM (no Wait states at 4 MHz while operating off User clock)

Inputs
One standard LS-TTL load plus 30 pF maximum

Outputs
Capable of driving one standard LS-TTL load plus 30 pF preload

Rear Panel Output
BNC connector for pulse output, standard LS-TTL

Front Panel
Target/Monitor, Reset, and NMI toggle switches

Power
110/220 Vac, 50/60 Hz switch selectable, 60 VA maximum

Dimensions
4 in. (10.2 cm) (H) × 14½ in. (36.8 cm) (W) × 18 in. (45.7 cm) (D)

Emulator Cable
12 inches

AC CHARACTERISTICS

Number	Symbol	Parameter	Z8001/2 Min (ns)	Z8001/2 Max (ns)	Z-SCAN Min (ns)	Z-SCAN Max (ns)
1	TcC	Clock Cycle Time	250	2000	250	2000
2	TwCh	Clock Width (High)	105	2000	105	2000
3	TwCl	Clock Width (Low)	105	2000	105	2000
4	TfC	Clock Fall Time		20		20
5	TrC	Clock Rise Time		20		20
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		175
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		35	
8	TdC(Bz)	Clock ↑ to Bus Float		65		165
9	TdC(A)	Clock ↑ to Address Valid		100		163
10	TdC(Az)	Clock ↑ to Address Float		65		154
11	TdA(DI)	Address Valid to Data In Required Valid	455		383	
12	TsDI(C)	Data In to Clock ↓ Setup Time	50		76	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80		-4	
14	TdC(DO)	Clock ↑ to Data Out Valid		100		163
15	ThDI(DS)	Data In to $\overline{DS}$ ↑ Hold Time	0		-20	
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	295		269	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		29	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80		143
19a	TwMRh	$\overline{MREQ}$ Width (High)	210		193	
19b	TwMRh	$\overline{MREQ}$ Width (High) During Monitor Operation			184	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70		53	
21	TdDO(DSW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55		59	
22	TdMR(DI)	$\overline{MREQ}$ ↓ to Data In Required Valid	350		287	
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		134
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		134
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55		29	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		144
27	TdAS(DI)	$\overline{AS}$ ↑ to Data In Required Valid	340		277	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		53	
29	TwAS	$\overline{AS}$ Width (Low)	70		53	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	60		43	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		-41	4

CONTINUED ON NEXT PAGE

AC CHARACTERISTICS

Number	Symbol	Parameter	Z8001/2 Min (ns)	Z8001/2 Max (ns)	Z-SCAN Min (ns)	Z-SCAN Max (ns)
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	70		53	
33	TdDSR(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid	185		122	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65
35	TdDS(DO)	$\overline{DS}$ ↑ to Data Out and STATUS Not Valid	75		58	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180		154	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		174
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275		258	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		149
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185		168	
41	TdDSI(DI)	$\overline{DS}$ (Input) ↓ to Data In Required Valid	320		266	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		174
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410		393	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065		1048	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		174
46	TdDSA(DI)	$\overline{DS}$ (Acknowledge) ↓ to Data In Required Delay	435		381	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		162
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	60		45	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		208	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		15	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		116	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		154	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		118	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	0		22	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		78	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		22	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		188	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		22	
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		120		165
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		148	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		22	
62	TsWT(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		78	
63	ThWT(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		25	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		98	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		32	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		145
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		145

ORDERING INFORMATION

Part No.	Description
05-0100-00	Z-SCAN 8000/1 Emulator (Supports Z8001 Emulation and Control)
05-0100-01	Z-SCAN 8000/2 Emulator (Supports Z8002 Emulation and Control)
05-0101-00	Z8001 Field Support Kit (Converts Z-SCAN 8000/2 into Z-SCAN 8000/1)
05-0102-00	Z8002 Field Support Kit (Converts Z-SCAN 8000/1 into Z-SCAN 8000/2)

Systems recommended for use with above:

Description	Prerequisites
ZDS-1 Series Development Systems	Z8000 SDP
PDS 8000 Series Development Systems	Z8000 SDP

Z80 PLZ™

Zilog

Product Description

March 1981

- **High-Level Procedure-Oriented Language Permits Efficient Writing of Machine-Independent Modules and Programs.**
- **Structured Format for Fast and Easy-to-Compile Programs.**
- **Produces Efficient Code for Economical Memory Usage and Processing Time.**
- **Simplifies Software Production and Maintenance.**
- **Allows Direct or Interpretive Execution of Program Modules.**

OVERVIEW

Z80 PLZ is a family of different programming languages designed to satisfy a wide range of microcomputer software development requirements. The two members of the PLZ family, PLZ/SYS and PLZ/ASM, produce object code-compatible modules and share common control structures and data definition facilities. Thus, selective portions of programs may be written in the most appropriate language for the specific application and still maintain a consistent structure between modules.

PLZ/SYS is a high-level, procedure-oriented language that is syntactically similar to PASCAL. It provides a medium for writing structured, machine-independent programs with a minimum of programming effort.

PLZ/ASM, on the other hand, is a structured assembly language that permits access to the low-level capabilities of the processor by mixing assembly language and high-level control structures.

FEATURES

Compiler. The Z80 PLZ/SYS Compiler translates source code modules into an intermediate stage called Z-code. The Z-code modules may then be executed interpretively or processed by the code generator to produce a machine-code object module.

Code Generator. The Z80 PLZCG Code Generator accepts a file of intermediate Z-code generated by PLZ/SYS and produces the corresponding Z80 machine code as a relocatable object module. This file may be linked with other modules to form the complete executable load module.

Interpreter. The intermediate Z-code modules produced by the Z80 PLZ/SYS Compiler can be executed interpretively by ZINTERP. Linking ZINTERP with the other modules generated by the compiler produces an executable load module.

PLZ/ASM Translator. The PLZ FILTER translates a PLZ/ASM source module into a file of the corresponding Z80 Assembler source. This gives the Assembler the benefit of logical data structure, program flow control, and modular program design, in addition to its existing features.

PLZ Linker. The PLZ Linker, PLINK, links Z-code, ZINTERP and/or machine code modules into a single relocatable load module, allowing the user to control the overall size and speed of the program.

Although interpretive Z-code runs more slowly than machine code, the space savings over machine code is usually substantial for larger programs where the 3K bytes of ZINTERP is a small percentage of the entire program. By balancing the number of Z-code and machine code modules, the user can maximize the efficiency of a particular program.

PLINK resolves any external references between separately assembled modules, so that the load module produced is relocatable. It also allows the reordering and combining of named sections between modules and supports incremental linking.

ORDERING INFORMATION

Part No.	Description
07-3301-01	Z80 PLZ Object Diskette for use with PDS 8000/05 and PDS 8000/15
07-3302-01	Z80 PLZ Object Diskette for use with ZDS-1 Series
07-3303-01	Z80 PLZ Object Cartridge Disk for use with PDS 8000/20 and PDS 8000/30
07-3303-02	Z80 PLZ Object Diskette for Hard Disk Systems with Optional Floppy Drives
07-3303-04	Z80 PLZ Object Cartridge Disk for use with PDS 8000/20A

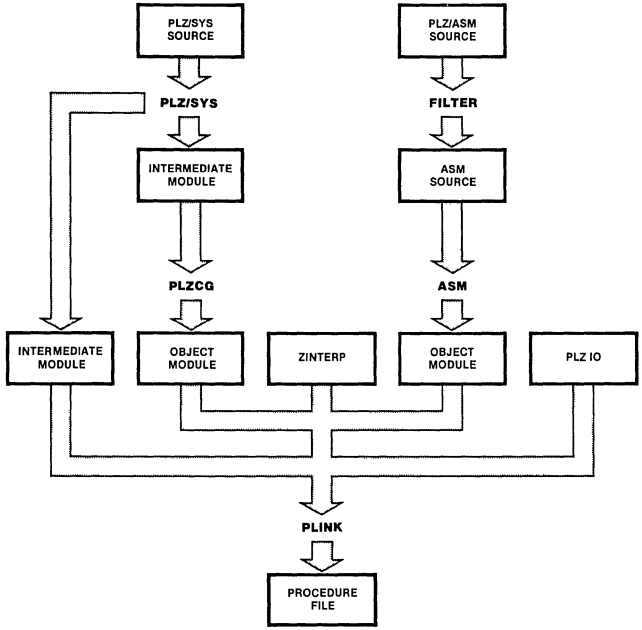


Figure 1. Z80 PLZ Language Modules.

Z8 Software Development Package

Product Description

March 1981

- **Structured Assembly Language with High-Level Constructs.**
- **Relocatable and Absolute Object Code Format.**
- **Free Format Statements Allow Indentation and Spacing for Readability.**
- **External Symbol References.**
- **Global Symbol Definitions.**

OVERVIEW

The Z8 Software Development Package consists of five utility programs which aid and simplify software development for Z8-based systems. Z8 PLZ/ASM, part of Zilog's PLZ family, brings all the advantages of modular programming to Z8 software development. The programming task can be broken into easily managed modules, giving more work assignment options to the engineering manager and a clear-cut structure to the individual programmer. The Z8 linker completes the task by combining the modules and resolving any external references.

FEATURES

Assembler. The Z8 PLZ/ASM Assembler translates easy-to-read, free-format PLZ/ASM source programs to object code. Because the user may specify that either absolute or relocatable object code be produced, he may choose a memory location for the program or leave that responsibility to the Linker. The Z8 PLZ/ASM Assembler produces a listing file containing both the source and assembled code.

Z8 PLZ/ASM allows an efficient mix of powerful assembly language mnemonics with high-level control structures such as IF . . . THEN . . . ELSE . . . FI and DO . . . OD loops. The PLZ/ASM programmer may map instructions and information into the Z8's register, program and data memory spaces, and organize the data space with such data declarations as RECORDS and ARRAYS. The PLZ/ASM Assembler supports external symbol references and global symbol definitions and is fully supported by the RIO™ operating system.

Z-LINK. Z-LINK links assembled modules into a single relocatable module and resolves any external references among separately assembled modules. It can also reorder and combine named sections found in the input assembly language modules. Z-LINK accepts a symbolic specification of the program entry point in the command line and, on request, produces a detailed link map which gives the locations of global references and relocated modules and sections. Errors in the linking process are reported in the optional link map and at the system console.

IMAGER. IMAGER accepts multiple linked-object files from the linker and translates them into absolute code. IMAGER can then either store the absolute code in a disk file or leave it in system memory. Named sections found in the input object modules may be reordered and loaded anywhere in system memory.

LOAD/SEND. LOAD/SEND downloads an absolute program file into the Z8 Development Module for debugging, then sends it back to the disk for back-up and storage.

Z-PROG. Z-PROG stores the perfected load module in PROM.

ORDERING INFORMATION

Part No.	Description
07-0086-01	Z8 Software Development Package Object Cartridge Disk for Use with PDS 8000/20A
07-3361-01	Z8 Software Development Package Object Diskette for Use with PDS 8000/5 and PDS 8000/15
07-3362-01	Z8 Software Development Package Object Diskette for Use with ZDS-1 Series
07-3363-01	Z8 Software Development Package Object Cartridge Disk for Use with PDS 8000/20 and PDS 8000/30
07-3363-02	Z8 Software Development Package Object Diskette for Hardware Disk Systems with Optional Floppy Drives

Z8000™ Software Development Package

Zilog

Product Description

March 1981

- **Structured assembly language with high-level constructs.**
- **Relocatable and absolute object code format.**
- **Free format statements allow indentation and spacing for readability.**
- **External symbol references.**
- **Global symbol definitions.**

OVERVIEW

The Z8000 Software Development Package consists of six utility programs which aid and simplify the development of Z8000 programs on the Z8000 Development Module. Z8000 PLZ/SYS and PLZ/ASM from Zilog's PLZ family bring all the advantages of modular programming to the Z8000 software developer and ensure transportability to future processors. The Z8000 LINKER, IMAGER, LOAD/SEND and Z-PROG simplify the testing and production stages of new software. Each program facilitates a single step towards completing a segmented or nonsegmented program; together they guarantee a smooth, logical, and manageable software development process.

FEATURES

Assembler. The Z8000 PLZ/ASM Assembler assembles easy-to-read, free-format PLZ/ASM source programs directly to machine code. PLZ/ASM allows an efficient mix of powerful assembly language mnemonics with high-level control structures, such as IF . . . THEN . . . ELSE . . . FI and DO . . . OD loops. The PLZ/ASM programmer may map instructions and information into the Z8000's program and data memory space, and organize the data space with such data declarations as RECORDS and ARRAYS. The PLZ/ASM Assembler supports both segmented and nonsegmented programs and is fully supported by the RIO™ operating system.

LINKER. Z-LINK links assembled modules into a single-load module. Z-INTERP, Z-code, and machine code modules may be combined in a single program to facilitate execution speed and memory conservation, or the most efficient balance of the two. Z-LINK resolves any external references between separately assembled modules, so that the load module produced is relocatable. It also allows the reordering and combining of named sections between modules. Z-LINK permits a symbolic specification of the program entry point in the command line and, on request, produces a detailed link map for program documentation.

IMAGER. The PLZ IMAGER can accept multiple linked object files from Z-LINK and translate them into absolute code. IMAGER can then either store the absolute code in a disk file or leave it in system memory. IMAGER may load Z-INTERP and Z-code object files and supports segmented code. Named sections found in the input object modules may be reordered and loaded anywhere in system memory.

LOAD/SEND. LOAD/SEND downloads an absolute program file into the Z8000 Development Module for debugging, then sends it back to the disk for back-up and storage.

Z-PROG. Z-PROG stores the perfected load module in PROM.

ORDERING INFORMATION

Part No.	Description
07-0085-01	Z8000 Software Development Package Object Cartridge Disk for Use with PDS 8000/20A
07-3306-01	Z8000 Software Development Package Object Cartridge Disk for Use with PDS 8000/20
07-3306-02	Z8000 Software Development Package Object Diskette for Hard Disk Systems with Optional Floppy Drives
07-3309-01	Z8000 Software Development System Object Diskette for Use with PDS 8000/5
07-3310-01	Z8000 Software Development System Object Diskette for Use with ZDS-1 Series

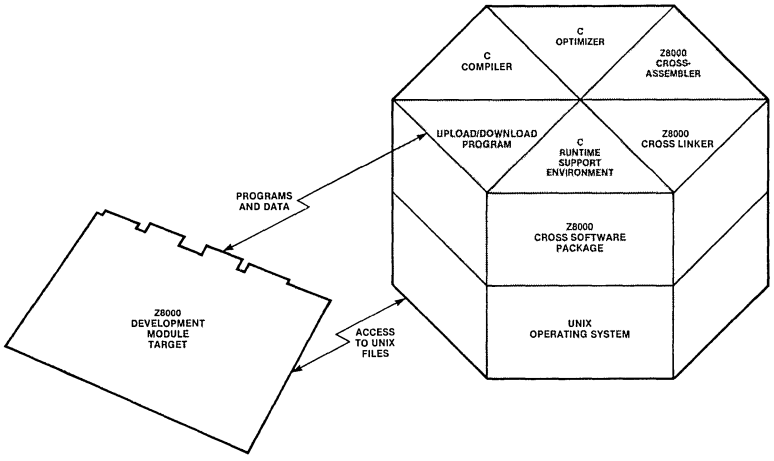
Z8000™ Cross-Software Package

Zilog

Product Description

March 1981

- **Runs on the UNIX* Operating System.**
  This enables multi-user access for more efficient software development and provides tools to aid documentation production.
- **High-Level, Machine-Independent, Systems Implementation Language, C, Generates Efficient Z8000 Code.**
  C improves programmer productivity, shortens product time-to-market, and protects software investment.
- **Includes C Run-Time Support Environment for the Z8000 Development Module.**
  This keeps product development on schedule by reducing dependency on prototype hardware.
- **C Compiler Produces Z8000 Cross-Assembler Source Code.**
  Assembly language listing of C programs simplifies debugging in any target environment.



OVERVIEW

In today's complex microprocessor-based products, software development costs typically exceed those of hardware development. The Z8000 Cross-Software Package, running on the UNIX operating system, reduces software development costs by improving programmer productivity and enabling software to be developed before prototype hardware is ready. This allows time for thorough product testing while still meeting development schedules. The result is a higher quality product delivered on schedule.

The Z8000 Cross-Software Package (CSP) is a complete set of software tools for developing Z8000 programs. The package works on Digital Equipment Corporation's PDP-11/44, 11/45, and 11/70 systems with the Seventh Edition of the UNIX operating system. Programmers and related support personnel at a UNIX installation can easily transfer their knowledge of the UNIX environment to the Z8000 development project. The result is that programmers become productive more quickly. And, there is a greater likelihood of the project finishing on schedule.

The C language, like other high-level, machine-independent, systems implementation languages, improves programmer productivity and protects the software investment made in a product by assuring program transportability. In addition, C produces Z8000 code which is efficient both in terms of execution time and memory space used. The result is a lower cost, higher performance product.

The development environment supported by the Z8000 CSP allows for multiple user software development on various Z8000 target systems (see figure below). The pass-through mode of the Z8000 Development Module enables any terminal connected to the host system to be a hardware and software evaluation station. In this mode, the terminal and the host system communicate directly as if the Z8000 Development Module were not present. Thus, each terminal on a host system can text edit and compile programs and then download them into a development module for testing.

The pass-through mode of the development module offers a more effective means of debugging than software emulation because programs can be debugged in real-time on actual hardware, without requiring any host system resources. New Zilog emulation products, such as Z-SCAN 8000, will continue to use the pass-through mode to communicate to the host system. Thus, a single host system with Zilog's development modules, emulation products, and the Z8000 CSP can support total product development.

*UNIX is a trademark of Bell Laboratories.

PRODUCT DESCRIPTION

The major pieces of software in the Z8000 CSP are the C compiler, C optimizer, Z8000 cross-assembler, Z8000 cross-linker, upload/download program for the Z8000 Development Module, and C run-time support environment for the Z8000 Development Module.

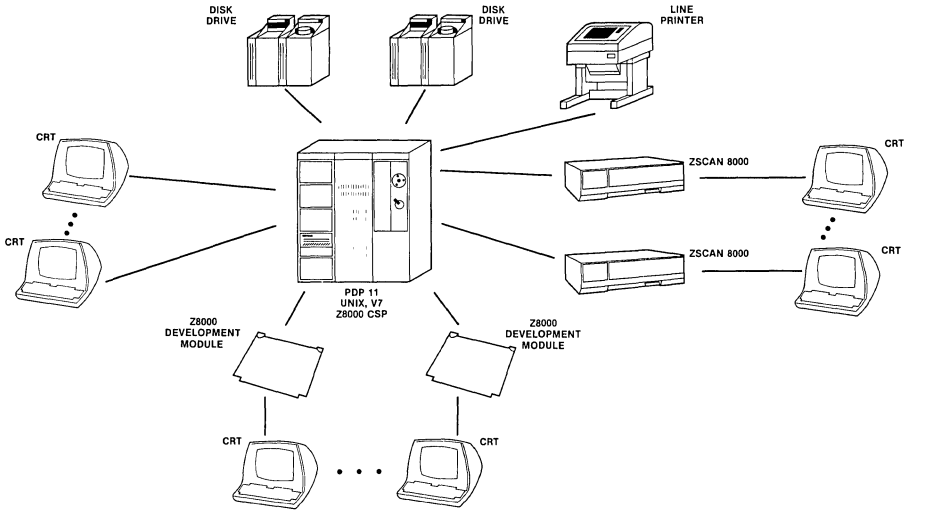
The Z8000 C compiler is the portable PDP-11 C compiler from the Seventh Edition of the UNIX system modified to generate Z8000 code. This means that existing PDP-11 C programs can be compiled by the Z8000 C compiler and, if the programs are machine-independent, they will run on a Z8000 target system. The C compiler presently generates non-segmented Z8000 code; segmented code will be supported later.

The C optimizer speed optimizes the code produced by the compiler and outputs Z8000 cross-assembler source code. This process yields an assembly language listing of the optimized code.

The Z8000 cross-assembler accepts Zilog's standard mnemonics and uses the pseudo-operations familiar to UNIX assembly language programmers. It supports programs with combined or separate code and data spaces. The Z8000 cross-linker links cross-assembler and C program modules together.

The upload/download program transfers programs and data between the Z8000 target system and the UNIX host using Tektronix hex format. The C run-time support environment provides the necessary facilities to run sophisticated C programs on the Z8000 Development Module. Because it includes routines for terminal and UNIX file access, significant software development can take place using C and the Z8000 Development Module.

The Z8000 Cross-Software Package combines with the UNIX operating system to provide a complete development environment for Z8000 software.



Typical Z8000 Cross-Software Package Installation

ORDERING INFORMATION

Part Number
07-3341-01

Description
Z8000 Cross-Software Package, includes:

Software

- Z8000 C compiler
- Z8000 C code optimizer
- Z8000 cross-assembler
- Z8000 cross-linker
- Upload/download program
- C run-time support environment

Software (Continued)

All software is distributed on one reel of magnetic tape recorded at 800 BPI.

Documentation

- Z8000 Cross-Software Package User's Guide
- The C Programming Language Manual
- Z8000 CPU Technical Manual

PREREQUISITES

- License for the Seventh Edition of the UNIX operating system.
- One of the following computers from Digital Equipment Corporation:
  PDP 11/44
  PDP 11/45
  PDP 11/70

License Requirement

- A special license is required for Z8000 Cross-Software Package

Training

Zilog

Zilog Technical Training

Courses for All Zilog Components and Systems

March 1981

Time and money: precious commodities in the 1980's. There never seems to be enough to get the job done. At Zilog, our wide range of innovative components and systems helps you get the edge on your competitors by reducing both system design costs and time. Now Zilog's Training and Education Department can help you turn the clock back even further by saving on those costly hours spent getting up to speed.

Zilog offers sophisticated microcomputer products in every form—from microprocessor components and development systems to OEM boards and general-purpose microcomputer systems. And to give you the knowledge necessary to take full advantage of these products, we also offer thorough training programs geared for the design engineer.

Zilog courses offer an informal hands-on, interactive approach that takes you where you need to be: up to speed in the quickest, most efficient way. Each course enhances your ability to use individual Zilog products effectively. You get all the information you want and need.

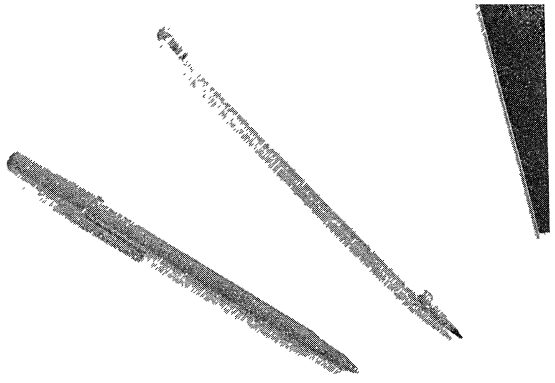
The Zilog Training and Education Department offers an exceptionally wide range of courses. This section describes them in detail.

Because enrollment is limited, register at least five weeks before class start date. Classes cancelled less than 15 days before class start date are subject to a $100 cancellation fee.

Zilog offers discounts to companies with three or more students attending any given course.

On request, Zilog also offers on-site courses at a customer's plant.

For schedules and enrollment information, call the Zilog Training and Education Department at (408) 446-4666.



Microprocessors: A General Introduction

This course introduces the world of microprocessors. In it you will learn basic microprocessor fundamentals and capabilities as well as the basics of microcomputer-based design. Some of the topics covered include:

- *What is a microprocessor?*
- *Some fundamental concepts about microprocessors*
- *Microprocessor organization*
- *Instruction execution*
- *Memories, central processing units, support chips*
- *Microprocessor interfacing*
- *Programming a microprocessor*

A background in digital electronics is helpful but not necessary.

Length: Three days
Tuition: $450
Order Number:
05-0008-00

Z80 Component Family

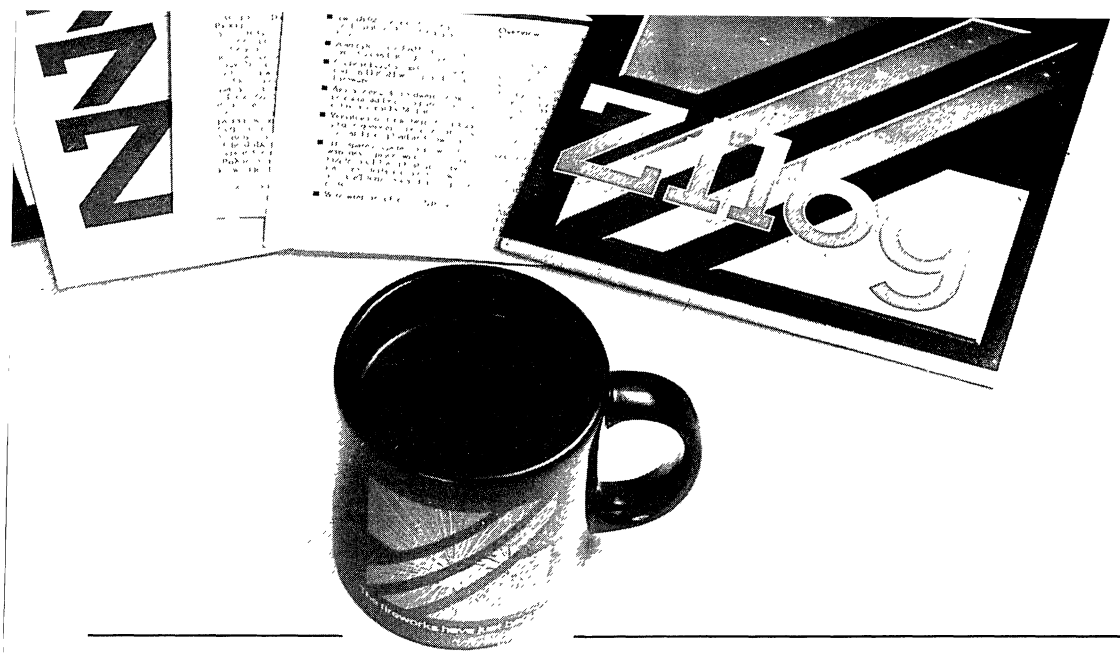
This basic course on Z80 components is designed for hardware and software development personnel with a modest background in microprocessors and assembly language programming. This course should be taken by anyone interested in effectively using the Z80 family of products. Some topics covered are:

- *Z80 architecture and timing*
- *Z80 assembly language programming*
- *Z80 interrupt processing (interfacing non-Zilog peripherals)*
- *Z80 PIO Parallel I/O Controller*
- *Z80 CTC Counter/Timer Controller*
- *Z80 DMA Direct Memory Access Controller*

This course offers a "hands-on" approach to learning by doing. As each chip is covered, students measure their progress by programming a single-board computer in the laboratory.

A general microcomputer course or equivalent experience is suggested as a course prerequisite.

Length: Four days
Tuition: $595
Order Number:
05-1001-08



Z8000 Component Family

Zilog's basic course on the Z8000 family of components is for hardware and software development personnel who are familiar with microprocessor system design. Anyone interested in effectively using the Z8000 family of products should take this course. Some of the topics covered include:

- *Z8000 architecture and timing*
- *Z8000 assembly language programming*
- *Z8010 MMU Memory Management Unit*
- *Z-BUS peripheral interfacing*
- *Z8000 peripheral devices (CIO, FIO, SCC, and UPC)*
- *Z8000 software development tools*
- *Z8000 Development Module and other support products*

A general microcomputer course or equivalent experience is suggested as a course prerequisite.

Length: Four days
Tuition: $595
Order Number:
05-1001-09

Z8 Component Family

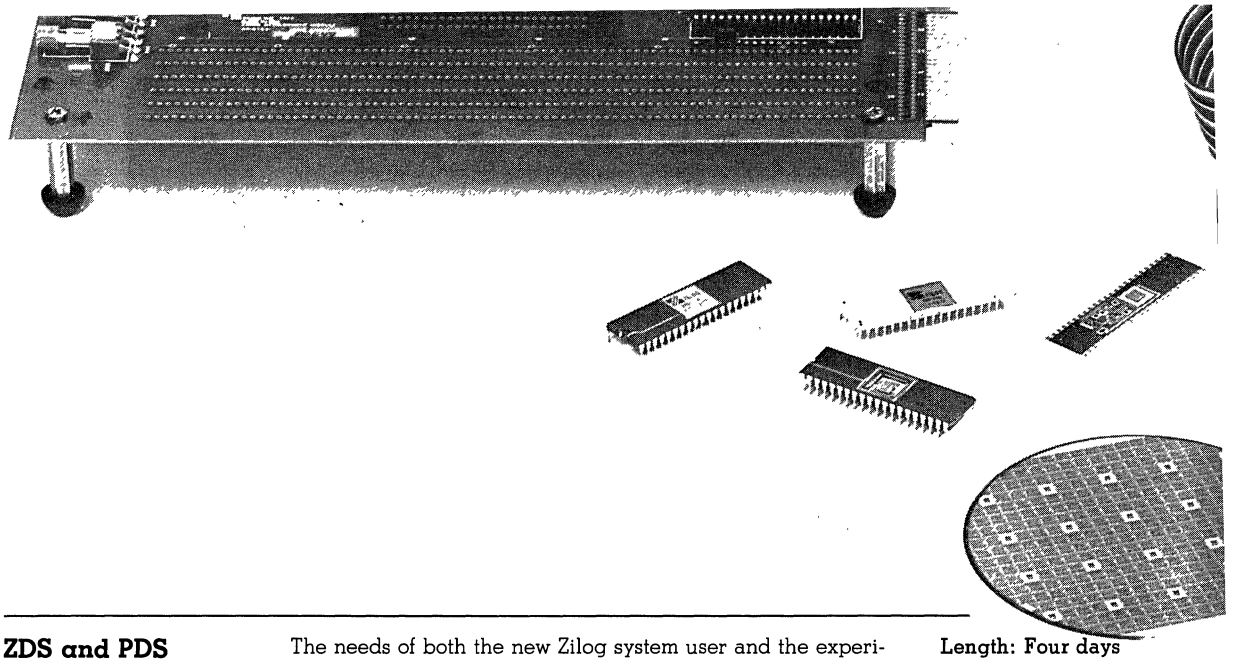
The Z8 is Zilog's powerful single-chip, 8-bit microcomputer. This seminar is designed for hardware and software development personnel who are familiar with microcomputer system design and who are interested in learning Z8 architecture, capabilities, and supporting systems. Some of the topics covered are:

- *Z8 architecture and timing*
- *Z8 assembly language programming*
- *Interfacing memory and peripheral devices*
- *Z8 software development tools*
- *Z8 Development Module and other supporting products*

Designers interested in using the Z8034 UPC Universal Peripheral Controller should also attend this seminar, since the architecture of the UPC is very similar to that of the Z8.

A general microcomputer course or equivalent experience is suggested as a course prerequisite.

Length: Three days
Tuition: $450
Order Number:
05-1001-06



ZDS and PDS Operating Systems

The needs of both the new Zilog system user and the experienced designer are met in this Z80-based systems course. The full range of ZDS and PDS microcomputer systems is described. Emphasis is placed on RIO, the Zilog operating system. Some topics covered are:

- *ZDS and PDS hardware*
- *Z80 assembler, linker, debugger, editor*
- *Advanced debugging techniques (symbolic debugging ZBUG and NBUG)*
- *Elements of RIO—the ZDS and PDS operating system*
- *RIO structure—making system calls*
- *RIO floppy-disk driver—ZDOS*
- *Device drivers—printers, consoles*

This course provides a "hands-on" approach to learning by doing. As each portion of the operating system is covered in lecture, students can measure their progress by writing their own programs in class.

A Z80 component class or equivalent Z80 assembly language experience is suggested as a course prerequisite.

Length: Four days
Tuition: $595
Order Number:
05-1001-02

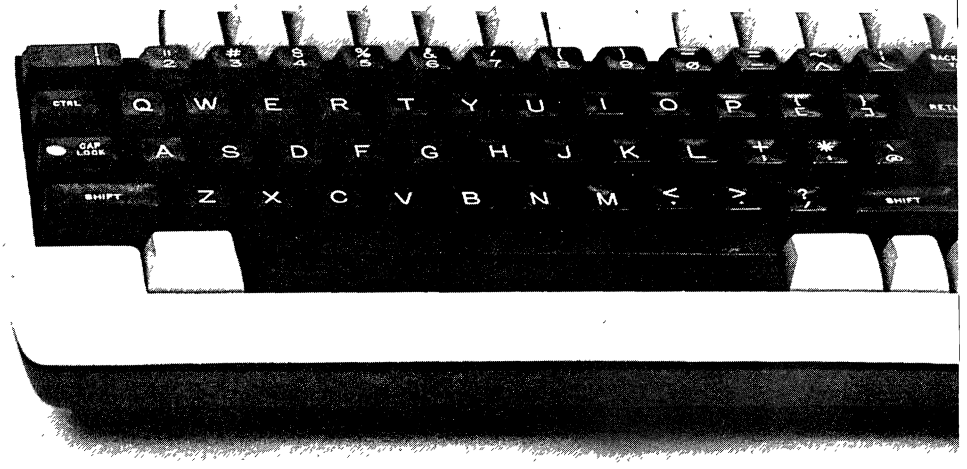
MCZ-2 Systems

This Z80A-based systems course introduces the systems user to MCZ-2 local network microcomputer systems architecture and operation. The full range of MCZ-2 systems is described, with emphasis placed on RIO/CP, Zilog's multitasking operating system. Some topics covered are:

- *RIO/CP (Concurrent Processing) multitasking operating system*
- *MCZ-2 System Kernel—dispatcher for multitasking environment*
- *Logical I/O mapping*
- *RIO/CP floppy disk driver—FFS*
- *COBOL calls to assembler*
- *Z-NET philosphy and local networking concepts*

A Z80 component class or equivalent Z80 assembly language experience is suggested as a course prerequisite.

Length: Four days
Tuition: $595
Order Number:
05-0009-00



PLZ/SYS Programming

The PLZ programming seminar is for programmers who need language tools that permit methodical and well-organized programs. PLZ, Zilog's Pascal-like language, includes the PLZ/SYS (high level, user-oriented) and PLZ/ASM (a structured assembly language) elements. Some topics covered in this seminar are:

- *Program structure*
- *Data types—simple and structured*
- *Recursive programming*
- *Pointers and linked lists*
- *System I/O calls*
- *Comparison of programming languages*
- *Protocols for communicating with other languages*
- *The PLZ symbolic Debugging Tool (PDT)*

Length: Four days
Tuition: $595
Order Number:
05-1001-04

ZDS-1/40 Development System

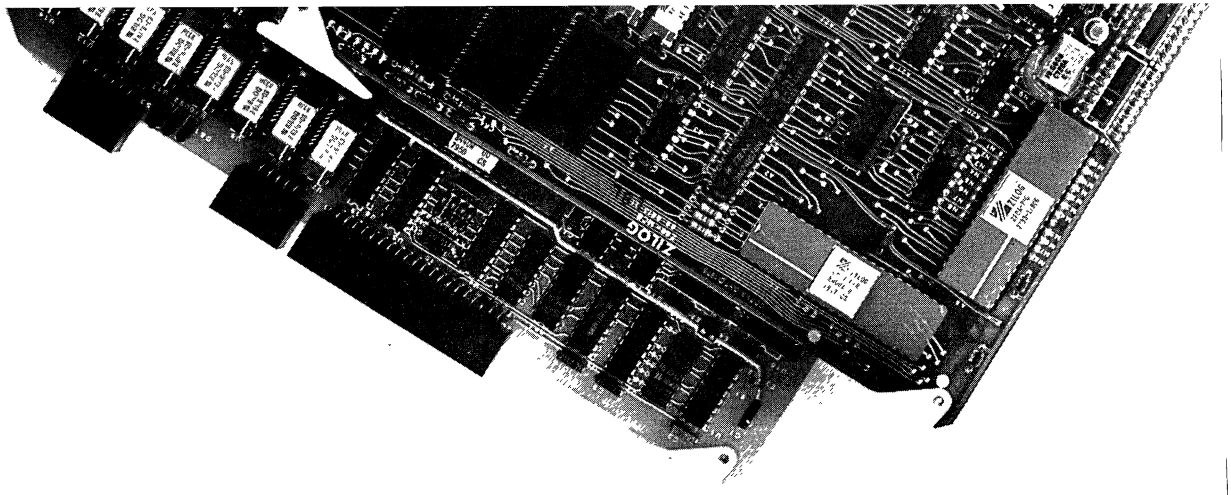
This seminar describes Z80 emulation using the ZDS-1/40 development system. Description of the ZDS-1/40 emphasizes those aspects of the development system that affect the emulation process. Some of the topics covered include:

- *ZDS-1/40 hardware design*
- *Z80 system design hints to aid the emulation process*
- *The Zilog Analyzer Program (ZAP)*
- *The RIO Hardware Emulation Driver (RHED)*

This course is recommended for designers of Z80 systems where the emulation process is used as a development tool, as well as for engineers who are directly involved in Z80 emulation.

A Z80 component class or equivalent Z80 assembly language experience is suggested as a course prerequisite.

Length: One day
Tuition: $150
Order Number:
05-1001-03



Data Communications Concepts

This course provides the engineer with a thorough background in the terminology and operating concepts of data communications. Some of the topics covered are:

- *Line protocols, telecommunications codes (ASCII, EBCDIC, BCD)*
- *Asynchronous and synchronous transmission*
- *Error Detection, CRC and parity codes*
- *Half and full duplex concepts*
- *Modems; RS232C and RS449 specifications*
- *Introduction to protocols—Bisync, SDLC, X.25, DDCMP*
- *Introduction to networking*

Zilog offers a variety of solutions to data communications support problems. This course also offers an overview of Zilog's SIO and SCC data communication controllers, and a description of Zilog's ASYNC and 2780/3780 emulator.

Length: Four days
Tuition: $595
Order Number:
05-0010-00

Z80 OEM Board Family Component Course

This component-level course is offered for the OEM or large volume end user who needs to know component level theory, maintenance, and repair techniques for the Zilog MCB series boards. Some of the topics covered are:

- *MCB (MicroComputer Board) theory and repair*
- *MDC (Memory Disc Controller) theory and repair*
- *SIB (Serial Interface Board) theory and repair*
- *HDC (Hard Disk Controller Boards) theory and repair*

The Zilog Training and Education Department can offer a variety of special hardware classes, each tailored to your needs. Call Zilog's Training and Education Department at (408) 446-4666 for further information.

Length: Four days
Tuition: $895
(offered on request)

Z8000 Architecture Correspondence Course

Length: 5 Lessons
Tuition: $39
Order Number:
05-0007-00

Zilog's generation-ahead, 16-bit Z8000 is changing the way systems manufacturers and designers think about microprocessing. Now there's an easy way for you to learn everything you need to know to stay on top of this powerful new technology. Enroll today in Zilog's five-part, home-study seminar on Z8000 architecture for the advanced engineer. Learn the details of the Z8000's 16-bit architecture, techniques of memory management, methods of interfacing memory and peripherals, proper handling of interrupts and traps, and use of the Z8000's powerful instruction set.

You study at your own pace at home. Each lesson includes a test consisting of ten questions which is individually graded and critiqued. The total cost for all course materials and tests is $39. (On completion of this course, every registrant gets a colorful Captain Zilog T-shirt!) Become your company's expert on the microprocessor technology of the future. Allow up to six weeks for receipt of your first lesson.

Lesson 1

Introduces the Z8000 architecture, starting with a description of the function of each signal pin on the Z8001 and Z8002 processors. The Z8000 register structure is examined in detail, followed by a discussion of the Z8000's eight basic operand addressing modes. Some basic system considerations, such as initialization requirements, are also introduced.

Lesson 2

Concentrates on memory and peripheral interfacing. The memory organization for a typical Z8000 system is described, leading to an examination of the memory control circuitry necessary to support the Z8000's powerful 16-bit architecture. Hardware design considerations, such as address/data bus buffering, are included in this lesson. Basic instruction timing is analyzed, with emphasis on memory and peripheral access timing.

Lesson 3

Provides a detailed discussion of interrupts, traps, and other context switching within a Z8000 system. The concept of program status is introduced, and the methods of changing program status through context switching are analyzed. Both the hardware and software considerations necessary for proper handling of interrupts and traps are investigated.

Lesson 4

Studies the concepts of memory and peripheral management. Memory segmentation in Z8000 systems is examined, including an analysis of the reasons for using memory segmentation as well as a discussion of possible implementations. Peripheral management and resource sharing in Z8000 systems are also explored.

Lesson 5

Offers an overview of the Z8000's powerful instruction set. After reviewing the operand addressing modes, the instruction set is divided into functional categories, and the instructions in each category are described in detail.

Zilog
Training and Education
10460 Bubb Road, Cupertino, CA 95014

Z8000 Architecture Correspondence Course Registration Card

Enroll me today in the 1981 Zilog Z8000 Architecture Course (Order Number: 05-0007-00).
☐ Enclosed is my check or money order for $39 (no cash or purchase orders please). Make check payable to Zilog, Inc.

Please charge my: ☐ Mastercharge or ☐ Visa account: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration date _____(month) _____(year) Signature _____

Business Address

Name _____
Company Name _____
Address _____
City/State/Zip _____
Telephone _____

Billing Address (as shown on charge card)

Name _____
Company Name _____
Address _____
City/State/Zip _____

Mail lessons to: ☐Business Address ☐Billing Address

My T-Shirt size is: S☐ M☐ L☐ XL☐

Please allow 4-6 weeks for delivery.

Zilog
Training and Education
10460 Bubb Road, Cupertino, CA 95014

Z8000 Architecture Correspondence Course Registration Card

Enroll me today in the 1981 Zilog Z8000 Architecture Course (Order Number: 05-0007-00).
☐ Enclosed is my check or money order for $39 (no cash or purchase orders please). Make check payable to Zilog, Inc.

Please charge my ☐ Mastercharge or ☐ Visa account: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration date _____(month) _____(year) Signature _____

Business Address

Name _____
Company Name _____
Address _____
City/State/Zip _____
Telephone _____

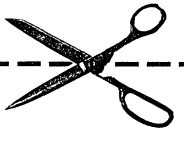
Billing Address (as shown on charge card)

Name _____
Company Name _____
Address _____
City/State/Zip _____

Mail lessons to: ☐Business Address ☐Billing Address

My T-Shirt size is: S☐ M☐ L☐ XL☐

Please allow 4-6 weeks for delivery.



Zilog
Training and Education
10460 Bubb Road, Cupertino, CA 95014

Z8000 Architecture Correspondence Course Registration Card

Enroll me today in the 1981 Zilog Z8000 Architecture Course (Order Number: 05-0007-00).
☐ Enclosed is my check or money order for $39 (no cash or purchase orders please). Make check payable to Zilog, Inc.

Please charge my: ☐ Mastercharge or ☐ Visa account: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration date _____(month) _____(year) Signature _____

Business Address

Name _____
Company Name _____
Address _____
City/State/Zip _____
Telephone _____

Billing Address (as shown on charge card)

Name _____
Company Name _____
Address _____
City/State/Zip _____

Mail lessons to: ☐Business Address ☐Billing Address

My T-Shirt size is: S☐ M☐ L☐ XL☐

Please allow 4-6 weeks for delivery.

Reader's Comments

Your feedback about this document helps us ascertain your needs and fulfill them in the future. Please take the time to fill out this questionnaire and return it to us. This information will be helpful to us and, in time, to future users of Zilog products.

Your Name: _____

Company Name: _____

Address: _____

Title of this document: _____

Briefly describe application: _____

_____

_____

Does this publication meet your needs? ☐ Yes ☐ No If no, why? _____

_____

_____

How are you using this publication?

☐ As an introduction to the subject?

☐ As a reference?

☐ As an instructor or student?

How do you find the material?

	Excellent	Good	Poor
Technically	☐	☐	☐
Organization	☐	☐	☐
Completeness	☐	☐	☐

Rated on a scale of 1 to 10, this data book is a _____.

What would have improved the material? _____

_____

_____

Other comments and suggestions: _____

_____

_____

_____

_____

If you found any mistakes in this document, please let us know what and where they are: _____

_____

_____

_____

_____

After you have filled out this page, please clip it and return to Zilog, Inc., Corporate Communications, Building 2, 10340 Bubb Road, Cupertino, California 95014.

Zilog, Inc.
10340 Bubb Road
Cupertino, California 95014
Telephone (408) 446-4666
TWX 910-338-7621

00-2034-01
Printed in U.S.A.