

03-3047-02

This is a preliminary manual. The information contained in this manual is subject to change.

© Copyright 1977 by Zilog Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise) without the prior written permission of Zilog Inc.

Zilog assumes no responsibility for the use of any circuitry other than circuitry embodied in a Zilog product. No other circuit patent licenses are implied.

Z8 Microcomputer

Preliminary Technical Manual

PREFACE

This reference manual contains the detailed hardware information you will need to understand and use the Z8 microcomputer. Chapter 1 describes Z8 architecture and organization with special emphasis placed on the addressing spaces and I/O port organization. Chapter 2 explains the function and operation of the various control registers used to configure the Z8. Miscellaneous device control details such as power down operation and test mode operation are described in Chapter 3. Chapter 4 explains the various addressing modes and summarizes the instruction set. Development of Z8-based systems is discussed in Chapter 5. Finally, Chapter 6 provides electrical characteristics and ordering information.

This manual is one of a series describing the Z8. You will need several other manuals to develop, debug and run Z8 programs. The Z8 PLZ/ASM Assembly Language Programming Manual (03-3030-01) describes PLZ/ASM design features, explains the assembly language instruction set and provides information needed to build a source program. The Z8 assembler produces relocatable object modules. Operation of the assembler and object module linkage and relocation are described in the Z8 Assembler User's Guide (03-3048-01).

Z8 programs can be executed in a software environment using a simulator program called Z8SIM, which runs on a Zilog 64K MCZ or ZDS development system. Refer to the Z8 Simulator User Manual (03-3046-01) for detailed information about the features and operation of this program.

Programs are developed on either a Zilog Microcomputer System (MCZ) or the Zilog Development System (ZDS) using the software capabilities of the RIO operating system. The manuals needed to use the operating system are the Z-80 RIO Operating System User's Manual (03-0072-01) and the Z-80 RIO Text Editor User's Manual (03-0074-00).

TABLE OF CONTENTS

PREFACE

1.	ARCHITECTURE.	1-1
1.1	ARCHITECTURAL OVERVIEW	1-1
1.2	ADDRESS SPACES	1-3
1.2.1	Program Memory.	1-4
1.2.2	Data Memory	1-5
1.2.3	External Memory	1-5
1.2.4	Register File	1-6
1.2.5	Stacks.	1-8
1.3	PIN DESCRIPTION.	1-9
1.4	TIMING	1-10
1.4.1	Instruction pipelining.	1-11
1.4.2	Instruction Cycle Timing.	1-12
1.4.3	External Instruction Fetch, I/O and Memory Timing	1-13
1.4.4	Interrupt Timing.	1-16
1.4.5	Reset Timing.	1-17
1.4.6	Alternative Control Signal Uses	1-18
1.5	I/O PORTS.	1-19
1.5.1	Structure	1-19
1.5.2	Port 1.	1-21
1.5.3	Port 0.	1-22
1.5.4	Port 2.	1-23
1.5.5	Port 3.	1-23
1.5.6	Port Handshake.	1-24
1.5.7	Serial I/O.	1-26
1.6	COUNTER/TIMER OPERATION.	1-28
1.7	INTERRUPTS	1-30
1.8	STATUS FLAGS	1-32
2.	CONTROL REGISTERS	2-1
2.1	SERIAL I/O (SIO)	2-1
2.2	TIMER MODE (TMR)	2-1
2.3	T1 COUNTER/TIMER (T1).	2-3
2.4	T1 PRESCALER (PRE1).	2-4
2.5	T0 COUNTER/TIMER (T0).	2-5
2.6	T0 PRESCALER (PRE0).	2-5
2.7	PORT 2 MODE (P2M).	2-5
2.8	PORT 3 MODE (P3M).	2-6
2.9	PORT 0 AND 1 MODE (P01M)	2-8
2.10	INTERRUPT PRIORITY (IPR)	2-9
2.11	INTERRUPT REQUEST (IRQ).	2-10
2.12	INTERRUPT MASK (IMR)	2-11
2.13	FLAG REGISTER (FLAGS).	2-12
2.14	REGISTER POINTER (RP).	2-12
2.15	STACK POINTER (SP)	2-12
2.16	REGISTER RESET VALUES.	2-13

3.	COMPONENT CONTROLS.	3-1
3.1	POWER DOWN OPERATION	3-1
3.2	RESET.	3-2
3.3	CLOCK.	3-2
3.4	TEST MODE OPERATION.	3-3
4.	ADDRESSING MODES AND INSTRUCTION SET SUMMARY.	4-1
4.1	ADDRESSING MODES	4-1
4.1.1	Register Address.	4-1
4.1.2	Indirect-register Address	4-2
4.1.3	Indexed Address	4-3
4.1.4	Direct Address.	4-4
4.1.5	Relative Address.	4-4
4.1.6	Immediate Data.	4-5
4.1.7	A Note on the Register Pointer.	4-5
4.2	INSTRUCTION SUMMARY.	4-6
4.2.1	Functional Summary.	4-6
4.2.2	Flags and Condition Codes	4-9
4.2.3	Notation.	4-11
4.2.4	Instruction Summary	4-12
5.	Z8 DEVELOPMENT SUPPORT.	5-1
5.1	PLZ/ASM ASSEMBLER.	5-1
5.2	Z8 SIMULATOR	5-1
5.3	ZPB-8 PROTOTYPING BOARD.	5-2
5.4	Z8/64 DEVELOPMENT DEVICE	5-2
5.4.1	Z8/64 Pin Description	5-2
6.	ELECTRICAL CHARACTERISTICS.	6-1
6.1	ABSOLUTE MAXIMUM RATINGS	6-1
6.2	STANDARD TEST CONDITIONS	6-1
6.3	DC CHARACTERISTICS	6-1
6.4	EXTERNAL INSTRUCTION FETCH, I/O OR MEMORY READ TIMING.	6-2
6.5	EXTERNAL I/O OR MEMORY WRITE TIMING.	6-3
6.6	MEMORY PORT (Z8/64) TIMING	6-4
6.7	ADDITIONAL TIMING.	6-5
6.8	HANDSHAKE TIMING	6-6
6.9	TEST LOAD CIRCUITS	6-6

LIST OF ILLUSTRATIONS

1-1	Z8 Block Diagram	1-3
1-2	Z8 Address Spaces.	1-3
1-3	Program Memory Map	1-4

1-4	Data Memory Map	1-5
1-5	Register File Organization.	1-7
1-6	Register Pointer Mechanism.	1-8
1-7	Z8 Pin Assignments (40-Pin Production Version).	1-9
1-8	Instruction Pipelining.	1-11
1-9	Instruction Cycle Timing (One-byte Instructions).	1-12
1-10	Instruction Cycle Timing (Two- and Three-byte Instructions)	1-13
1-11a	External Instruction Fetch, I/O or Memory Read Cycle.	1-14
1-11b	External I/O or Memory Write Cycle.	1-14
1-12a	Extended External Instruction Fetch, I/O or Memory Read Cycle	1-15
1-12b	Extended External I/O or Memory Write Cycle	1-15
1-13	Interrupt Cycle Timing.	1-16
1-14	Reset Cycle Timing.	1-18
1-15	Ports 0, 1 and 2 Block Diagram.	1-19
1-16	Port 3 Block Diagram.	1-20
1-17a	Input Handshake Signals	1-25
1-17b	Output Handshake Signals.	1-25
1-18	Serial I/O Block Diagram.	1-27
1-19	Serial Data Formats	1-28
1-20	Counter/Timer Block Diagram	1-29
1-21	Interrupt Cycle Process	1-32
3-1	Recommended Driver Circuit for Power Down Operation.	3-1
3-1	Power Up Reset Circuit.	3-2
4-1	Register Addressing	4-1
4-2	Working-Register Addressing	4-2
4-3	Indirect-Register Address in Register File.	4-3
4-4	Indirect-Register Address in Program or Data Memory	4-3
4-5	Indexed Addressing.	4-4
4-6	Direct Addressing	4-4
4-7	Relative Addressing	4-5
4-8	One-Byte Instruction Formats.	4-13
4-9	Two-Byte Instruction Formats.	4-13
4-10	Three-Byte Instruction Formats.	4-13
5-1	Z8/64 Pin Assignments	5-3
5-2	Z8/64 Package	5-3

LIST OF TABLES

1-1	Port 3 Control Functions.	1-24
1-2	Derivation of Common Bit Rates.	1-27

1-3	Interrupt Types, Sources and Vector locations . . .	1-31
2-1	Tin Functions and Operation	2-3
2-2	Control Register Reset Conditions	2-13
4-1	Z8 Instruction Set: Functional Groups	4-6
4-2	Instruction Summary	4-14

CHAPTER 1 - ARCHITECTURE

1.1 ARCHITECTURAL OVERVIEW

The Z8 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8 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 Z8 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K of internal ROM, a traditional microprocessor that manages up to 124K 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 remains available for I/O.

Real-time control applications, for which the Z8 is particularly suited, require fast instruction execution and fast interrupt response. Operating from an 8 MHz clock source (4 MHz internal clock rate), the Z8 executes most instructions in 1.5 to 2.5 us (6 to 10 machine cycles). Not only is the interrupt response fast, but with six vectored interrupts that are maskable and prioritized, the Z8 offers greater interrupt capabilities than comparable single-chip microcomputers.

The powerful and extensive instruction repertoire of 47 instruction types combined with the efficient internal register addressing scheme not only speeds program execution, but also packs more program into the on-chip ROM than would be possible with comparable microcomputers. This is, of course, extremely important for single-chip microcomputers, where on-chip memory space is limited.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, the Z8 offers an on-chip asynchronous receiver/transmitter (UART), and two counter/timers with a large number of user-selectable modes. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

The internal register organization centers around a 144-byte random-access register file composed of 124 general-purpose registers, 16 control registers and 4 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 9 groups of 16 working registers. A register pointer uses fast, short-format instructions to quickly access any one of the nine groups, resulting in fast and easy task switching.

In addition to these features, the Z8 also offers a power-down mechanism that protects the register file during power failure, and an on-chip oscillator that can accept several types of time-base inputs.

Z8 instructions can operate on several types of data elements including individual bits, 4-bit BCD digits, 8-bit bytes, or 16-bit words. Bits can be set, reset and tested. Nibbles are used in BCD arithmetic operations. Bytes are used for character or small integer values. Words are used for larger integer values and addresses.

The Z8 is a flexible single-chip microcomputer that--under software control--can take many different memory and I/O configurations. The two ends of this spectrum are: the Z8 as an I/O-intensive single-chip microcomputer; and the Z8 as a memory-intensive microcomputer. This capability has been achieved by merging a multiplexed address/data bus with the I/O-oriented ports. One key advantage of this organization is that external memory can be addressed while maintaining many of the I/O lines.

The Z8 has 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or bidirectional. Under software control, the ports can provide timing, status signals, address outputs, and serial or parallel I/O with or without handshake.

To provide for both I/O-intensive and memory-intensive applications, some Z8 address spaces have been separated and others merged to create three basic address spaces: program memory (internal and external), data memory (external) and the register file (internal). As mentioned previously, I/O port space and CPU control registers are mapped into the register file. The results of this organization are byte efficiency, programming ease and a large addressing space.

A 64-pin development version (Z8/64) of the 40-pin mask-programmed Z8 (Z8/40) is also available. The Z8/64 allows the user to prototype the system in hardware, and develop the code that is eventually mask-programmed into the on-chip ROM of the Z8/40.

The remainder of this chapter discusses the details of Z8 architecture. The block diagram for the Z8 is shown in Figure 1-1.

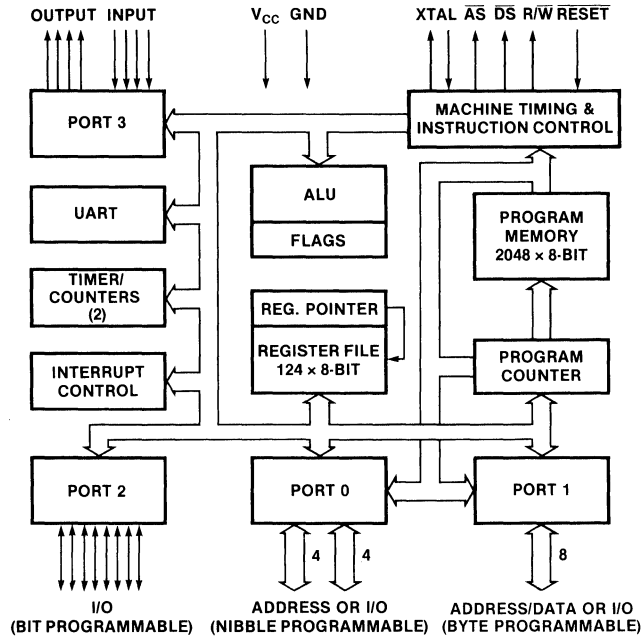


Figure 1-1. Z8 Block Diagram

1.2 ADDRESS SPACES

Figure 1-2 illustrates the arrangement of the various Z8 address spaces. The Z8 can address 64K of memory of program memory and 62K of data memory.

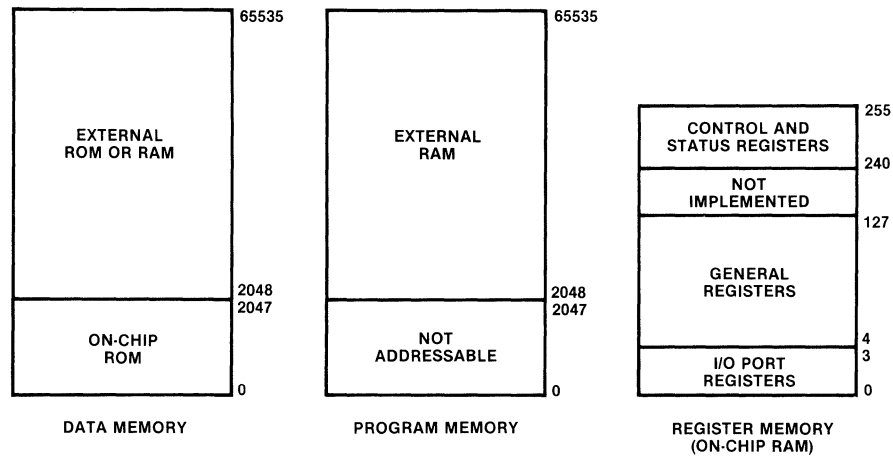


Figure 1-2. Z8 Address Spaces

1.2.1 Program Memory

The 16-bit program counter addresses 65,535 bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 1-3). The first 2048 bytes consist of on-chip mask-programmed ROM. At addresses 2048 and greater, the Z8 executes external program memory fetches, provided that it is appropriately configured.

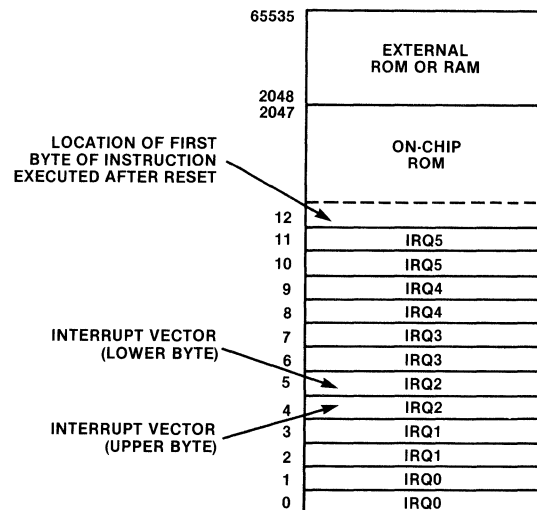


Figure 1-3. Program Memory Map

The first 256 bytes of external program memory (addresses 2048 to 2048 + 255) are addressed by configuring Port 1 as a multiplexed address/data port (AD0-AD7) that provides address bits A0-A7 and data bits D0-D7. Port 0 is configured for an additional four or eight address bits (A8-A12 or A8-A15) for applications that require a 4K or 64K program memory address space.

The first 12 bytes of program memory are reserved for the interrupt vectors. Addresses 0-11 contain six 16-bit vectors that correspond to the six available interrupts. When an interrupt occurs, program control is passed to a service routine pointed to by the address that is stored in the vector location of that particular interrupt. A reset forces the program counter to location 12, the first address available for user program. Refer to sections 1.4.4 and 1.7 for more thorough explanations of interrupt timing and interrupt operation.

1.2.2 Data Memory

A Z8 system can access 62K bytes of external data memory beginning at location 2048 (Figure 1-4). Like external program memory, external data memory addresses are provided by Port 1 for 8-bit addresses, or by Ports 0 and 1 for 12- and 16-bit addresses.

External data memory may be included with or separated from the external program memory addressing space. When data memory is separated from program memory, the Data Memory Select output (\overline{DM}) is used to select between data memory and program memory.

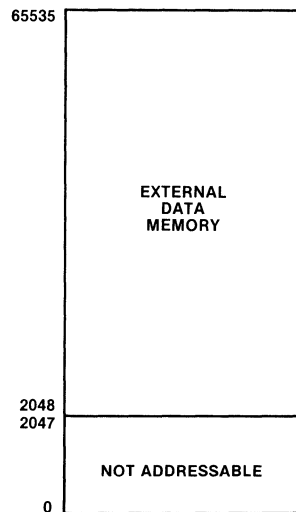


Figure 1-4. Data Memory Map

1.2.3 External Memory

Before external memory references can be made by any instruction, the user must configure Ports 0 and 1 appropriately. Because of instruction pipelining, it is mandatory that, after setting the modes of Ports 0 and 1 for external memory operation, the next two bytes are fetched from internal program memory. Two single-byte instructions such as NOPs can be used to do this.

The two external memory spaces, data and program, can be used as a single memory space of 62K bytes or as two separate spaces of 62K bytes each. If the memory spaces are separated, program memory and data memory are logically selected by the Data Memory Select output (\overline{DM}). \overline{DM} is

available on Port 3, line 4 (P34) by appropriately programming the Port 3 Mode register (Section 7.8). \overline{DM} is active only during the execution of the LDE, LDEI instructions and instructions with an external stack configuration (CALL, PUSH, POP, RET and IRET).

A subtle feature of the Z8 can be used to minimize the number of I/O lines assigned to the role of address outputs for medium-sized memory applications. This feature allows the user to address up to 10K of memory with only 12 address lines (plus the control lines \overline{DM} , \overline{DS} and R/\overline{W}).

Ordinarily, 12 address lines plus \overline{DM} would address only 4K in both the program and data spaces (in reality, a total of 6K because the first 2K of the data memory are not addressable). However, either \overline{DS} or R/\overline{W} can provide--in effect--a 13th address bit. This way, if the application requires between 4K to 6K of program memory (or 2K to 4K of data memory), only Port 1 and the lower nibble of Port 0 need be assigned to the role of address outputs. If this feature is not utilized, the upper nibble of Port 0 must be used to output additional addresses.

The following table illustrates how the 4K to 6K address space can be utilized without a 13th address bit. Address lines A0-All are sufficient to address the internal 0K to 2K space, in which All is always 0 and \overline{DS} and R/\overline{W} are inactive. A0-All are required to address the 2K to 4K space, and notice that \overline{DS} and R/\overline{W} are now active. For 4K to 6K, All is again 0 (as in the 0K to 2K case); however, because \overline{DS} and R/\overline{W} are still active, the 4K to 6K case can be distinguished from the 0K to 2K case where these signals are inactive.

PROGRAM MEMORY ADDRESS (PC)	DATA MEMORY ADDRESS	ADDRESS ON PORTS 0 & 1	All	\overline{DS} and R/\overline{W}
0-2047	-	0-2047	0	Inactive
2048-4095	2048-4095	2048-4095	1	Active
4096-6147	4096-6147	0-2047	0	Active

The 6K to 8K case cannot be distinguished from the 2K to 4K case because--in both cases, \overline{DS} and R/\overline{W} are active and All is 1. Therefore, the upper nibble of Port 0 must be used to address program or data memory spaces greater than 6K.

1.2.4 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). The 144 bytes of

the register file are assigned the address locations shown in Figure 1-5.

LOCATION		IDENTIFIERS
255	STACK POINTER (BITS 7-0)	SPL
254	STACK POINTER (BITS 15-8)	SPH
253	REGISTER POINTER	RP
252	PROGRAM CONTROL FLAGS	FLAGS
251	INTERRUPT MASK REGISTER	IMR
250	INTERRUPT REQUEST REGISTER	IRQ
249	INTERRUPT PRIORITY REGISTER	IPR
248	PORTS 0-1 MODE	P01M
247	PORT 3 MODE	P3M
246	PORT 2 MODE	P2M
245	T0 PRESCALER	PRE0
244	TIMER/COUNTER 0	T0
243	T1 PRESCALER	PRE1
242	TIMER/COUNTER 1	T1
241	TIMER MODE	TMR
240	SERIAL I/O	SIO
	NOT IMPLEMENTED	
127	GENERAL-PURPOSE REGISTERS	
4		
3	PORT 3	P3
2	PORT 2	P2
1	PORT 1	P1
0	PORT 0	P0

Figure 1-5. Register File Organization

The I/O port and control registers are included in the register file to allow any Z8 instruction to process I/O or control information, thereby eliminating special I/O and control instructions. In general 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.

Z8 instructions access registers directly or indirectly with an 8-bit address field. The Z8 also allows 4-bit register addressing using a register pointer mechanism, which saves bytes, reduces program execution time and speeds context switching. (Context switching refers to the saving and restoration of working registers, the program counter, flags and other pertinent information when an interrupt occurs).

In the 4-bit addressing mode, the register file is divided into 9 working register groups, each occupying 16 contiguous locations (Figure 1-6). A register pointer (one of the control registers) addresses the starting location of the active working-register group. Any instruction that can alter the contents of the register file can be used to alter the register pointer. The Z8 instruction set also provides a special Set Register Pointer instruction to initialize or alter the register pointer contents. A specific register within the active working-register group is specified by the 4-bit register designator supplied by the instruction.

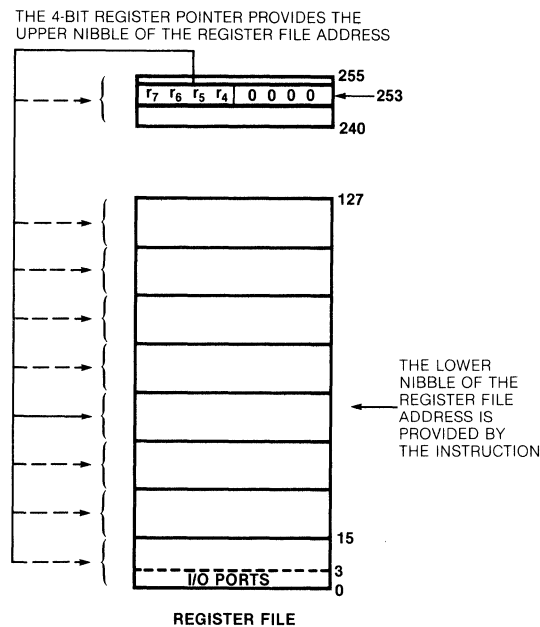


Figure 1-6. Register Pointer Mechanism

1.2.5 Stacks

Either the internal register file or the external data memory can be used for stacks. The selection is made by programming a bit in mode register R248. 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 which resides within the 124 general-purpose registers (R4 to R127).

The program counter during a CALL instruction, or the program counter and the flag registers during an interrupt

cycle are automatically saved on the stack. PUSH and POP instructions can save and restore any register of the register file, with the exception of write-only registers. The RET and IRET instructions pop the saved value of the program counter and of the flag register and program counter respectively.

1.3 PIN DESCRIPTION

P00-P07, P10-P17, P20-P27, P30-P37. *I/O Port Lines* (Inputs/Outputs, TTL compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured in a variety of ways under program control. In addition to their I/O functions, Ports 1 and 2 can be used for external memory interface and 3-stated under program control. Port 2 can be configured for open-drain outputs.

Individual lines of a port are denoted by the second digit of the number. For example, P30 refers to the least significant bit of Port 3.

\overline{AS} . *Address Strobe* (output, active Low). Address Strobe is pulsed once for internal and external program fetches and external data memory transfers. The addresses for all external program or data transfers are valid at the trailing edge of \overline{AS} . Note that \overline{AS} is active at the beginning of each machine cycle. Under program

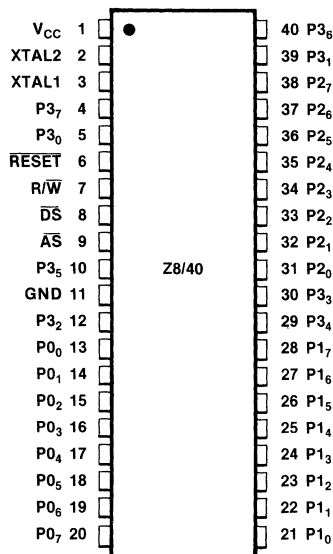


Figure 1-7. Z8 Pin Assignments (40-Pin Production Version)

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. During a write cycle, the Z8 places valid data on Port 1 while \overline{DS} is active. During a read cycle, the data is input through Port 1 while \overline{DS} is active. Under program control, \overline{DS} can be placed in a high-impedance state along with Port 0, Port 1, \overline{AS} and Read/Write.

When the Z8 is not configured for external memory transfer, \overline{DS} acts as an instruction sync signal and is forced Low during the clock period preceding the beginning of the opcode fetch.

R/\overline{W} . *Read/Write* (output, active Low), R/\overline{W} is Low when the Z8 is writing to external program or data memory. R/\overline{W} remains High for all other Z8 cycles. Under program control, R/\overline{W} can be placed in the high-impedance state along with Ports 0 and 1, \overline{DS} and \overline{AS} .

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum), LC network, RC network or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

The Z8 clock must be generated externally and entered via XTAL1 when the power-down option is used (section 3.1). XTAL2 is connected to standby power (V_{mm}), which powers the register file and reset logic during V_{cc} power failure.

\overline{RESET} . *Reset* (input, active Low). \overline{RESET} initializes the Z8. When \overline{RESET} is disasserted, the Z8 begins program execution from internal program location 000C (hex). \overline{RESET} acts as a register file protect during the power-down and power-up sequences. \overline{RESET} is also used to force the Z8 into the test mode. This mode is entered by raising the Reset input to a voltage greater than V_{cc} .

1.4 TIMING

The basic time periods used by the Z8 are machine cycles (M_n), timing states (T_n) and clock periods. All Z8 timing references are made with respect to the output signals \overline{AS} and \overline{DS} . The clock is shown in the following illustrations for clarity only, and does not have a specific timing relationship with other Z8 timing signals.

The important Z8 timing relationships are instruction

pipeline timing, instruction cycle timing, external memory (or I/O) timing, interrupt cycle timing and reset cycle timing. The following sections explain these timing cycles and illustrate each with a timing diagram.

1.4.1 Instruction Pipelining

The high rate of Z8 throughput is due, in part, to the use of instruction pipelining, where the instruction fetch and execution cycles are overlapped. During the execution cycle of an instruction, the opcodes for the next instruction are fetched (Figure 1-8).

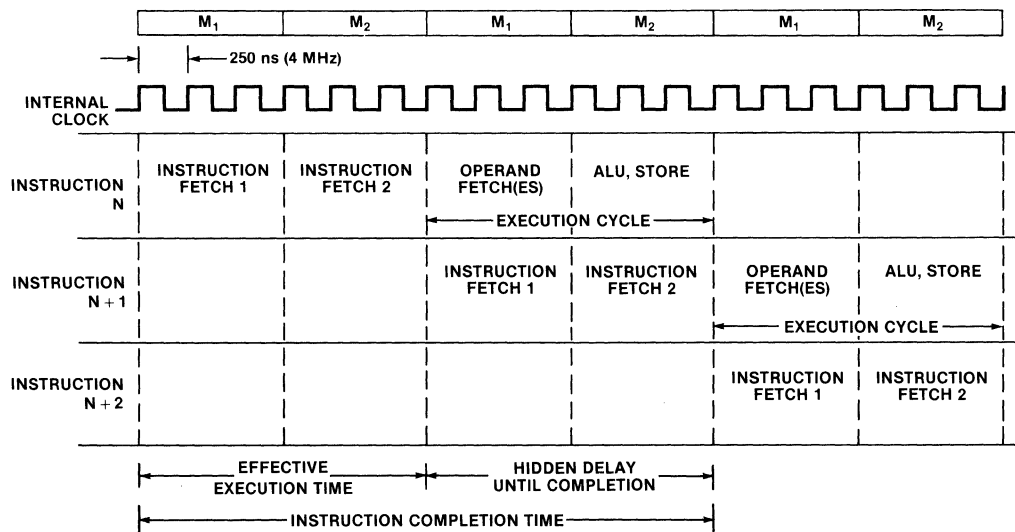


Figure 1-8. Instruction Pipelining

The overlap of instruction fetch and execution makes the "Effective Execution Time" of an instruction shorter than the "Instruction Completion Time" by the amount of overlap. The amount of overlap is called "Hidden Delay Until Completion." This delay is the amount of time incurred by an instruction until its results are valid.

When a program is running, the instruction overlap time is completely hidden in the total program execution time. Consequently, this amount of time can be subtracted from the instruction completion time to calculate the effective execution time of instructions in a program. However, when testing the results of a single instruction, the hidden delay must be taken into account.

The instruction summary in Chapter 4 (Table 4-2) has two

columns pertinent to instruction time calculations: "Execution Cycles" (effective execution time) and "Pipeline Cycles" (hidden delay until completion). Because of the effects of pipelining described above, only the execution time column is used in program throughput calculations. Because the hidden delay does not affect any instruction except the last executed instruction, it should not be used in throughput calculations.

1.4.2 Instruction Cycle Timing

Figures 1-9 and 1-10 show instruction cycle timing for instructions fetched from external memory.

The addresses, Address Strobe (\overline{AS}) and Read Write (R/\overline{W}) are output at the beginning of each machine cycle (M_n). The addresses output via Port 0 (if used) remain stable throughout the machine cycle, whereas addresses output via Port 1 remain valid only during M_nT_1 . The addresses are guaranteed valid at the rising edge of \overline{AS} , which should be used to latch the Port 1 output. Port 1 is placed in an input mode at the end of M_nT_1 . The Data Strobe is output during M_nT_2 , allowing data to be placed on the Port 1 bus. The Z8 accepts the data during M_nT_3 and DS is terminated.

An instruction synchronization pulse \overline{SYNC} is output one clock pulse period prior to the beginning of an opcode fetch machine cycle (M_1). This output is directly available on the 64-pin version of the Z8; whereas, on the 40-pin version, the Data Strobe pin outputs \overline{SYNC} only if external memory is not used.

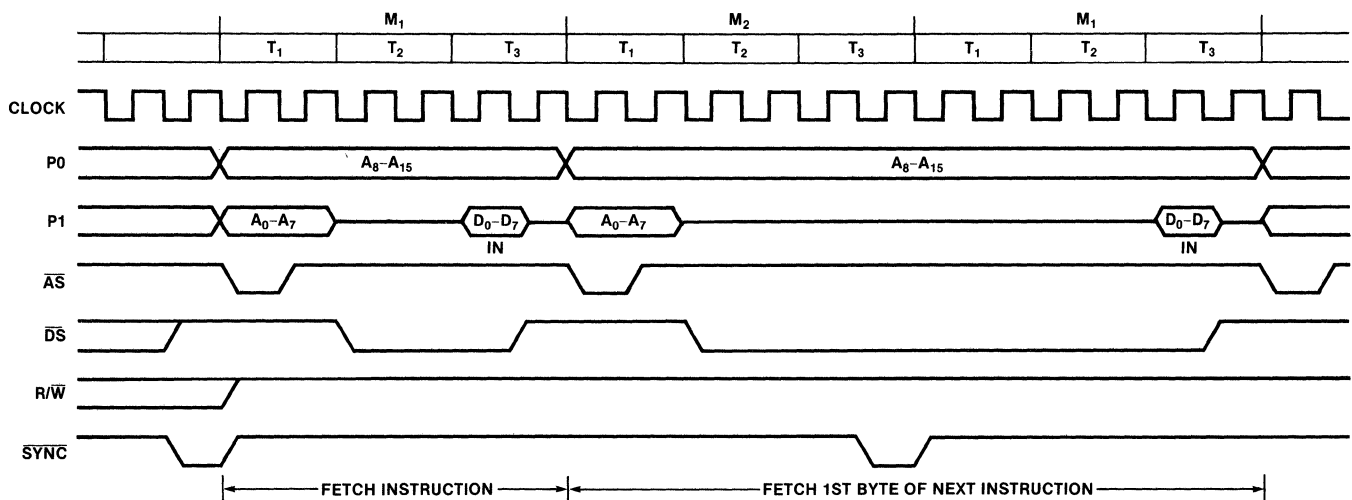


Figure 1-9. Instruction Cycle Timing (One-byte Instructions)

Note that all instruction fetch cycles have the same machine timing regardless of whether the memory is internal or not. If configured for external memory and internal memory is referenced, the addresses are still output via Ports 0 and 1; however \overline{DS} and R/\overline{W} are inactive. If configured for internal memory only, Ports 0 and 1 are used for I/O, \overline{DS} outputs SYNC, and R/\overline{W} is inactive.

The exception to the instruction fetch timing is during the opcode fetch of an instruction following the fetch of a one-byte instruction. One-byte instructions require two machine cycles to execute. The pipeline causes the following opcode fetch to begin one machine cycle early.

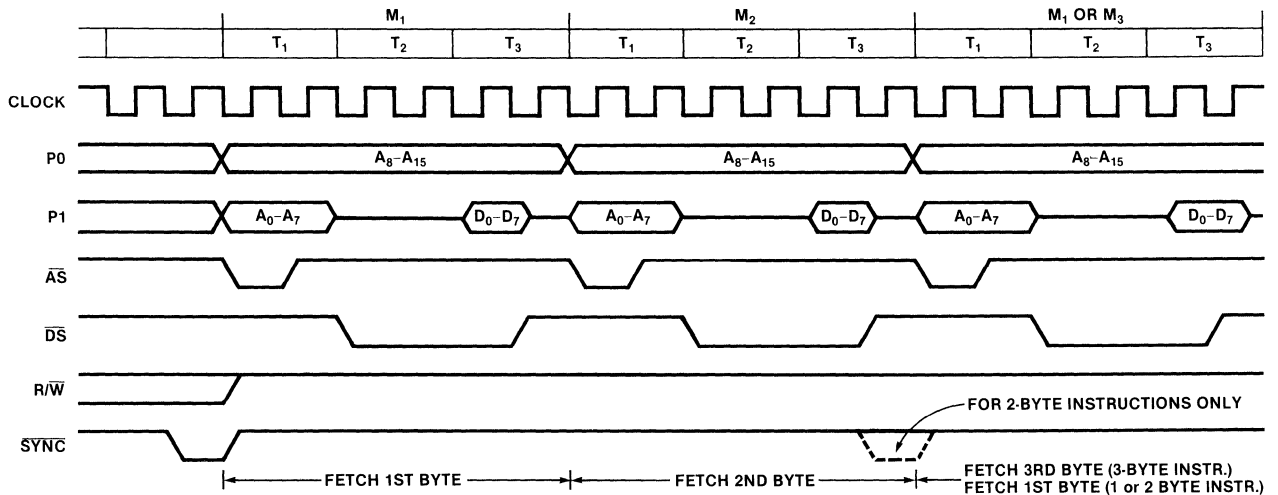


Figure 1-10. Instruction Cycle Timing (Two- and Three-byte Instructions)

1.4.3 External Memory or I/O Timing

When external memory is addressed, Ports 0 and 1 are configured to output the required number of address bits. Port 1 is used as a multiplexed address/data bus for AD₀-AD₇ and Port 0 outputs address bits A₈-A₁₅. The timing relationships for addressing external memory and I/O are illustrated in Figures 1-11 and 1-12. The main difference between these figures is that Figure 1-12 contains an added timing cycle (Tx) that extends external memory timing to allow for slower memory.

Address bits A₀-A₁₅ are valid on Ports 0 and 1 at the trailing edge of \overline{AS} for both the read and write memory cycles. Because Port 0 is not multiplexed, address bits A₈-A₁₅ -- if used -- are present all through the read/write memory cycle.

During the read cycle, the input data must be valid on Port 1 at the trailing edge of the Data Strobe output (\overline{DS}). The Data Memory Select output (\overline{DM}) is used to select external data memory or external program memory. If selected, \overline{DM} is active during the execution of certain instructions.

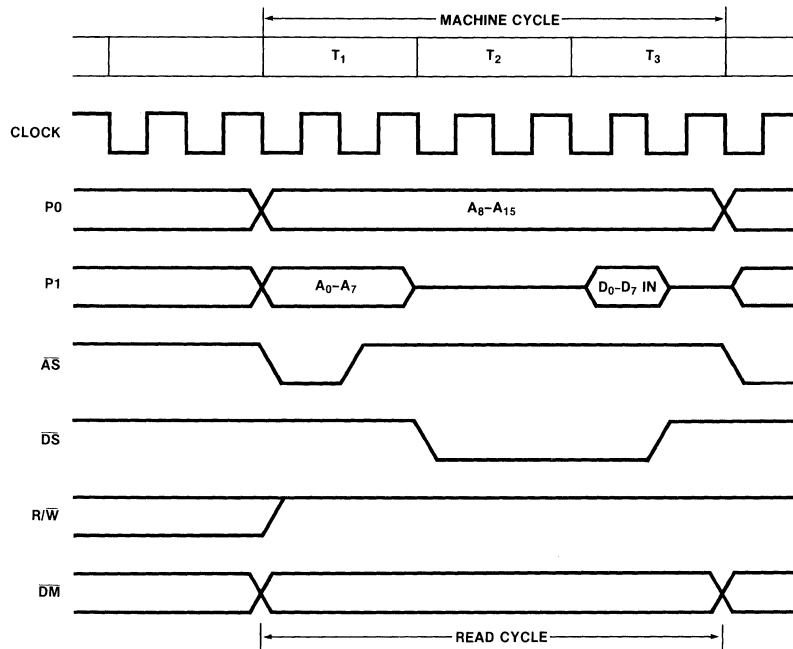


Figure 1-11a. External Instruction Fetch, I/O or Memory Read Cycle

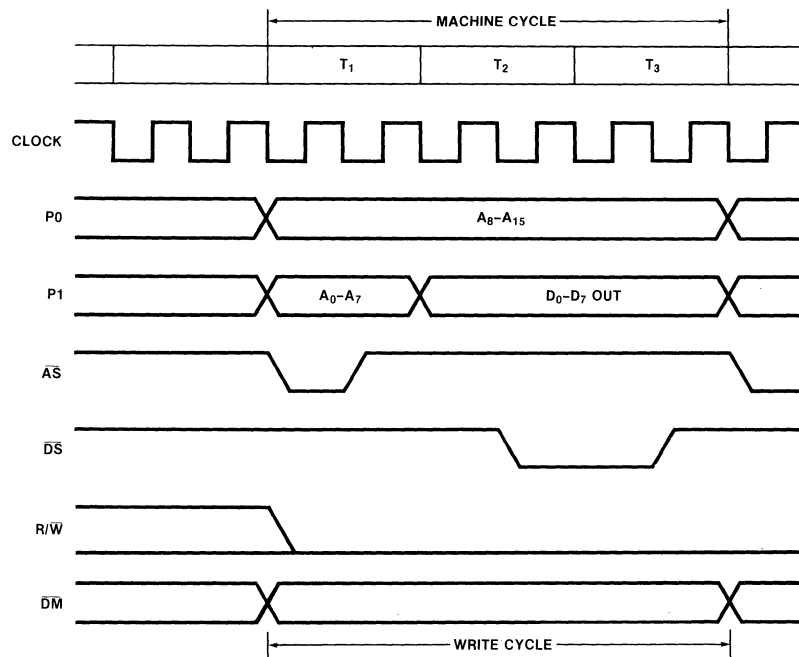


Figure 1-11b. External I/O or Memory Write Cycle

During the write cycle, the address outputs follow the same timing relationships as for the read cycle. However, the output data is valid for the entire period \overline{DS} is active, and R/\overline{W} is active (Low) during the entire write cycle.

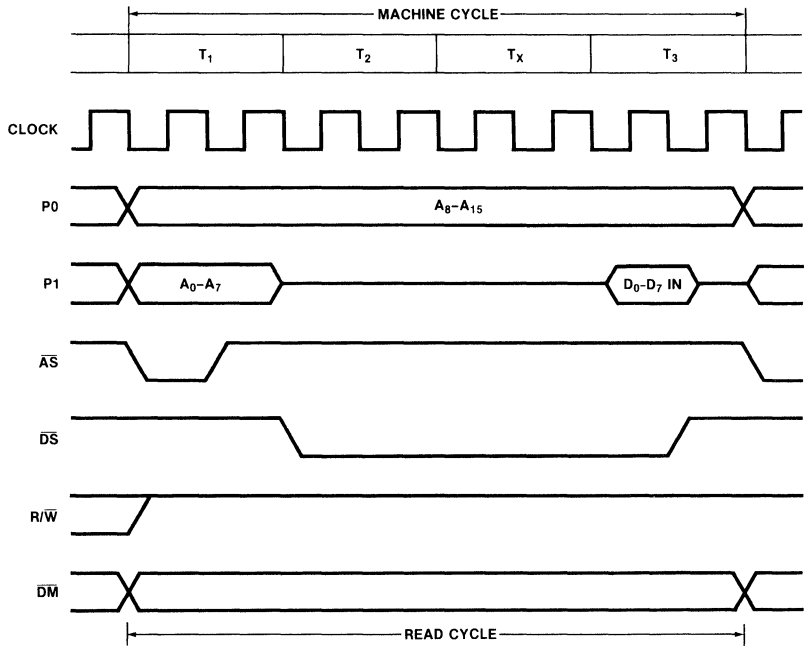


Figure 1-12a. Extended External Instruction Fetch, I/O or Memory Read Cycle

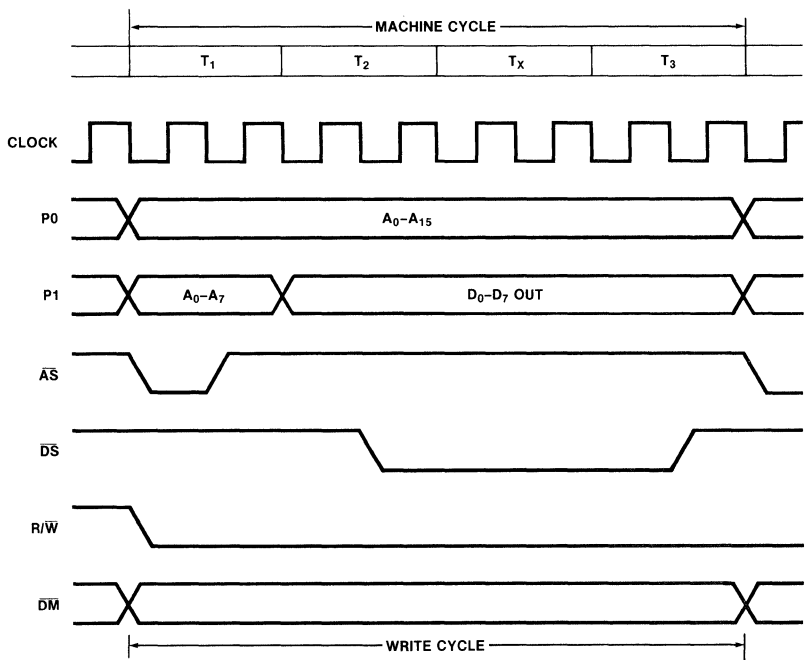


Figure 1-12b. Extended External I/O or Memory Write Cycle

1.4.4 Interrupt Timing

Interrupt requests are sampled before each instruction fetch cycle (Figure 1-13). First, external interrupt requests are sampled four clock periods prior to the active \overline{AS} pulse that corresponds to an instruction fetch cycle. Then, internal interrupt requests are sampled one clock period preceding \overline{AS} .

If an interrupt request is set, the Z8 spends seven machine cycles (44 clock periods) resolving interrupt priorities, selecting the proper interrupt vector, and saving the program counter and flags on the stack. Although Figure 1-13 illustrates the timing for an external stack, the same timing is used for an internal stack. The total interrupt response time (including the external interrupt sample time) for an external interrupt is 48 clock periods, at

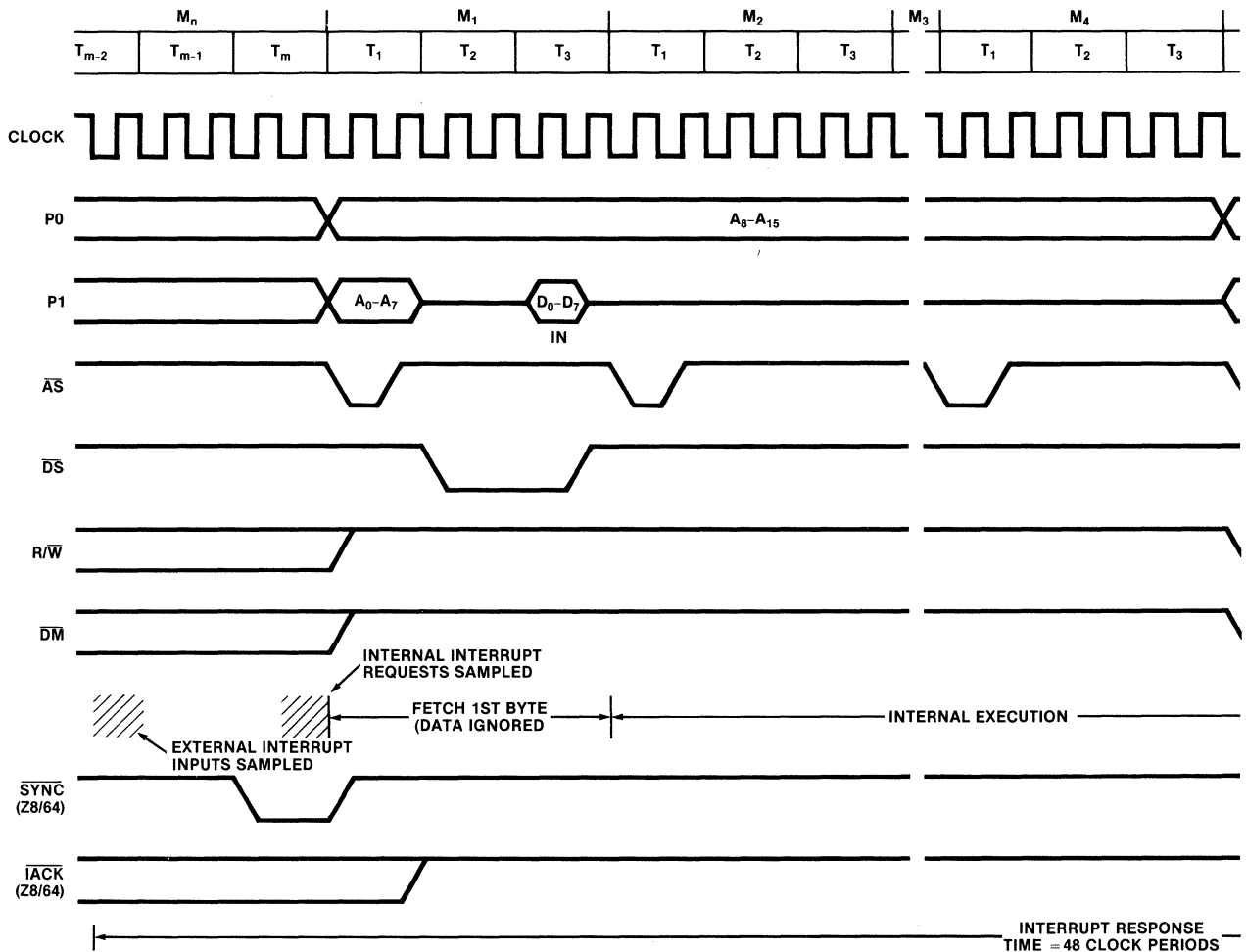


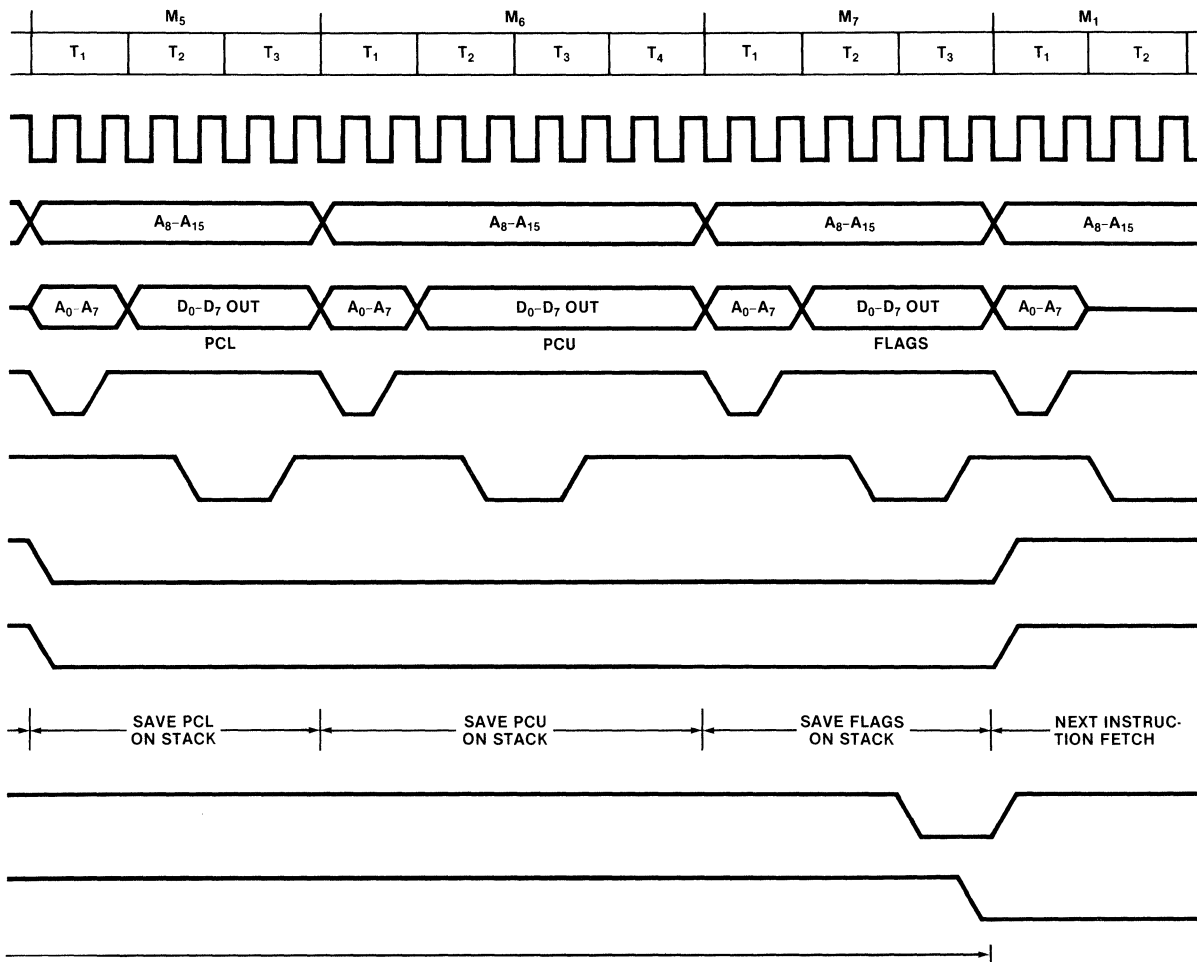
Figure 1-13.

which time the first instruction of the interrupt service routine is fetched.

When an interrupt request is detected in the Z8/64 development device, IACK is activated (High) and remains active until the first instruction of the interrupt service routine is fetched.

1.4.5 Reset Timing

The internal logic is initialized during reset if the Reset input is held Low for at least 18 clock periods (Figure 1-14). During the time RESET is Low, \overline{AS} is output at the internal clock rate, \overline{DS} is forced Low, R/W is inactive and Ports 0, 1 and 2 are placed in an input mode. \overline{AS} and \overline{DS} both Low



Interrupt Cycle Timing

is normally a mutually exclusive condition; therefore, the coincidence of \overline{AS} Low and \overline{DS} Low can be used as a reset condition for other devices. Zilog Z-Bus peripherals take advantage of this reset condition.

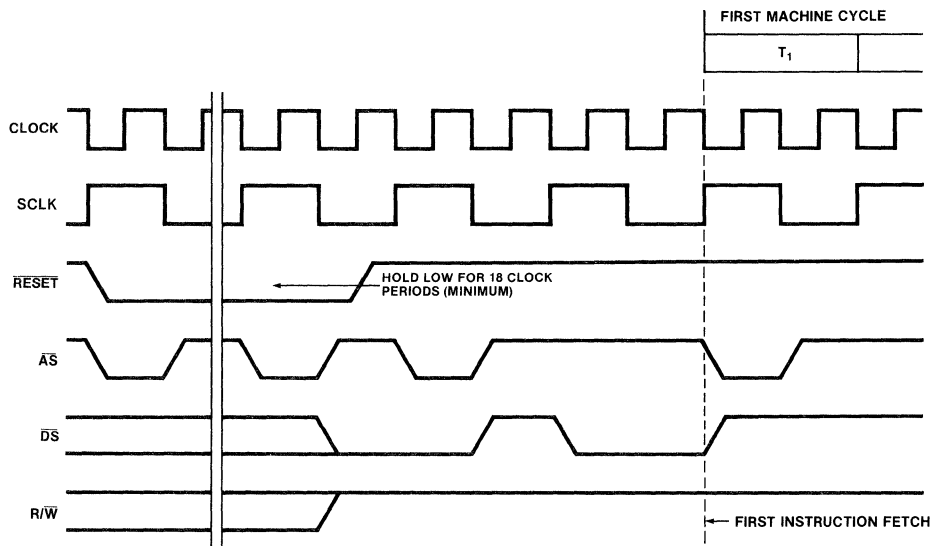


Figure 1-14. Reset Cycle Timing

1.4.6 Alternative Control Signal Uses

In addition to their uses in memory transfers, the control signals \overline{AS} , \overline{DS} and R/\overline{W} can be used in the following interface applications.

\overline{AS} can be modified to provide the \overline{RAS} (Row Address Strobe) signal for dynamic memory interface. \overline{RAS} can be derived from the trailing edge of \overline{DS} to the trailing edge of \overline{AS} .

\overline{DS} has several alternative uses: as a \overline{CAS} (Column Address Strobe) for dynamic memory interface, as a Chip Enable for memory and other interface devices, and as an Enable input for 3-state bus drivers/receivers for memory and interface devices.

R/\overline{W} can be used as a Write input to memory interfaces, and as an Early Status output to switch the direction of 3-state bus drivers/receivers.

1.5 I/O PORTS

This section describes the structure, function and operation of the I/O ports. The control registers used to configure these ports are explained in the next chapter.

The Z8 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, and serial and parallel I/O features with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

1.5.1 General Structure

Each bit of Ports 0, 1 and 2 has an input register, an output register, associated buffer and control logic. A block diagram of these Ports is shown in Figure 1-15.

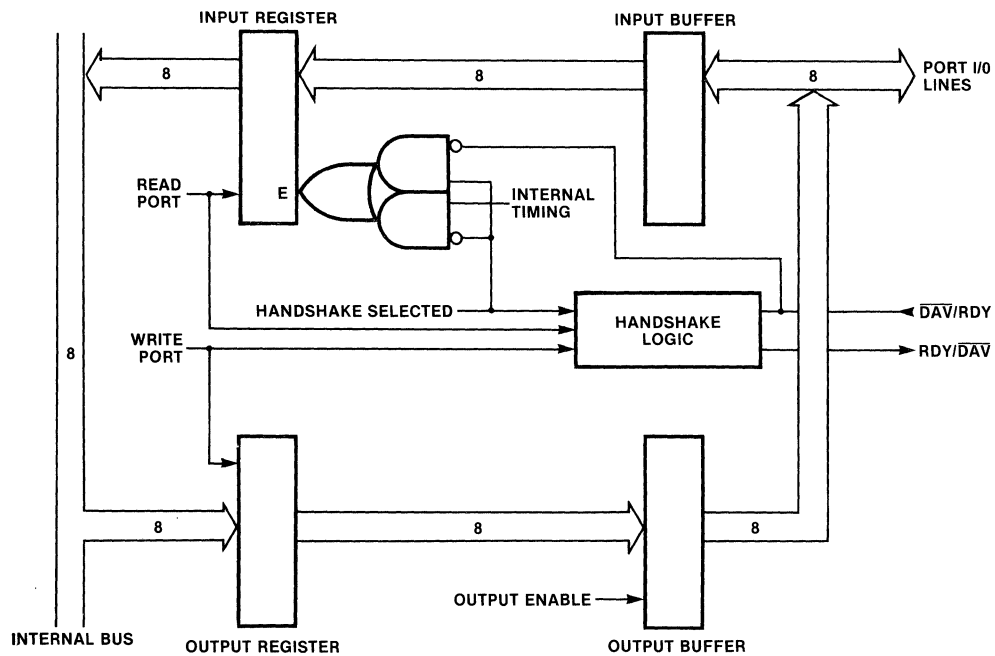


Figure 1-15. Ports 0, 1 and 2 Block Diagram

When a bit of Ports 0, 1 or 2 is configured as an output, writing that bit causes data to be stored in the output register. If an output bit is read, the data present on the external pin is returned. Under normal output loading, this is equivalent to reading the output register. However, if a bit of Port 2 is defined as an open-drain output, the data returned may not be the value contained in the output register; rather, it is the value forced on the input pin by the external system.

When a bit of Ports 0, 1 or 2 is defined as an input, reading that bit causes the data present on the external pin to be returned. The only exception is for input bits that are under handshake control. Reading a handshake input bit returns the data latched into the input register by the input strobe. Input bits can also be written to, but in this case, the data is stored in the output register and cannot be read back. However, if the input bits are re-configured as output bits, the data stored in the output register is reflected on the output pins. This mechanism allows the user to initialize outputs prior to driving their loads.

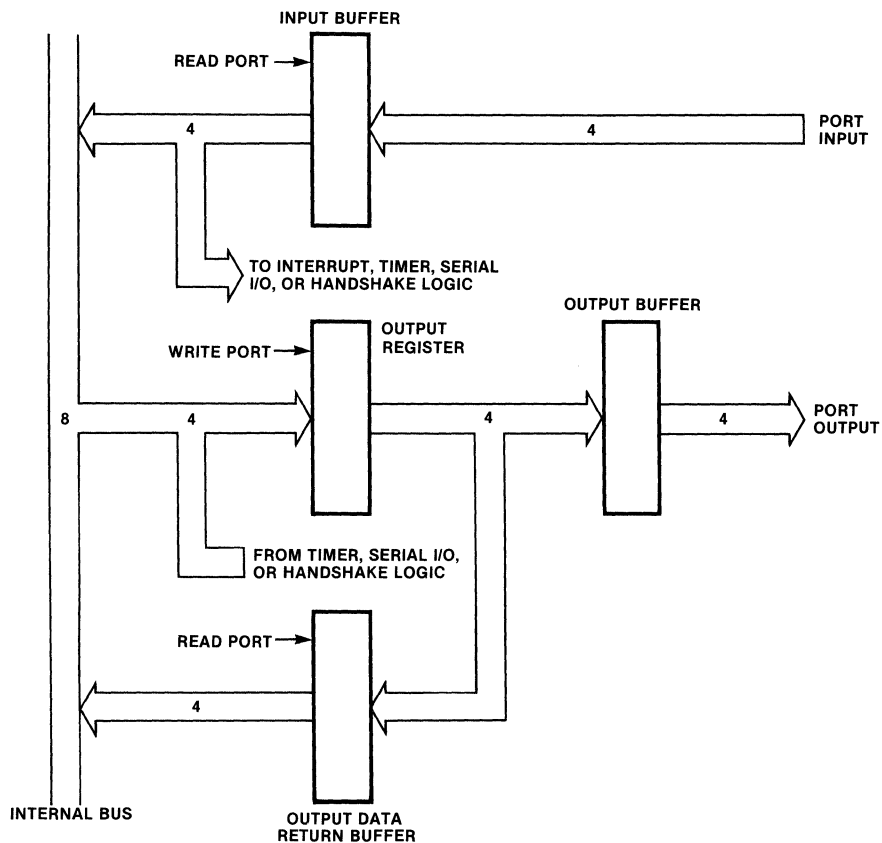


Figure 1-16. Port 3 Block Diagram

Port 3 is structurally different from the other ports in that it has only a single 4-bit output register associated with its four output bits (Figure 1-16). When writing to Port 3, data is stored in the output register. When reading Port 3, the data returned is composed of the data on the input pins and the data stored in the output register, but not the output pins. Port 3 outputs cannot be written if they are used for functions such as serial output, handshake controls or timer output.

1.5.2 Port 1

Port 1 can be programmed as a byte I/O port with or without handshake, or an address/data port for interfacing external memory. The configuration is set using the mode register for Ports 0 and 1 (P01M) R248.

When used in the byte input or byte output mode, the port is accessed as general register R1. The port is written by specifying R1 (of the register file) as the destination register of an instruction. The data is stored in the port output register. The port is read by specifying R1 as the source register of an instruction.

When used as an I/O port, Port 1 may be placed under handshake control by programming the Port 3 Mode register (P3M) R247. In this configuration, Port 3 pins P33 and P34 are used as the handshake control lines $\overline{DAV1}$ and RDY1 for input handshake, or RDY1 and $\overline{DAV1}$ for output handshake.

For external memory references, Port 1 must be programmed for the multiplexed address/data modes (AD0-AD7). In this configuration, the lower eight bits of address (A0-A7) are multiplexed with the data (D0-D7). Associated with Port 1 are the timing and control signals address Strobe (\overline{AS}), Data Strobe (\overline{DS}) and Read/Write (R/ \overline{W}). Figures 1-11 and 1-12 show the timing relationships of these signals during memory read and write operations. In this configuration, two additional control lines--an interrupt request input IRQ1 (P33) and the External Data Memory Select signal \overline{DM} (P34)--may be used under software control.

External memory locations greater than 2048 are referenced through Port 1. If more than 256 external locations are required, Port 0 is used to output the additional address lines.

Note:

When Port 1 is configured as a multiplexed address/data port, it cannot be accessed as a register.

In addition to the I/O and address/data modes, Port 1 can be placed in the high-impedance state (along with control lines \overline{AS} , \overline{DS} and R/\overline{W}), allowing the Z8 to share common resources for multiprocessor and DMA applications. This mode is totally under software control and is programmed using the P01M register (R248). Data transfers can be controlled by logically assigning P33 as a Bus Acknowledge input, BAK (IRQ1), and P34 as a Bus Request output, BRQ.

1.5.3 Port 0

Port 0 can be programmed as a nibble I/O port or as an address output port for addressing external memory. The selection is made by programming the mode register for Ports 0 and 1 (P01M) R248.

When a Port 0 nibble is used in the I/O mode, it is accessed as the corresponding nibble of register R0. The port is written by specifying R0 as the destination register of an instruction, and the data is stored in the port output register. The port is read by specifying R0 as the source register of an instruction.

Note:

A nibble defined as an address output cannot be accessed as a register.

When used as an I/O port, Port 0 may be placed under handshake control by programming the Port 3 Mode register (P3M) R247. In this configuration, Port 3 pins P32 and P35 are used as the handshake control lines $\overline{DAV0}$ and RDY0 for input handshake, or RDY0 and $\overline{DAV0}$ for output handshake. The handshake signal assignment for lines P32 and P35 is dictated by the direction (input or output) assigned to the upper nibble of Port 0.

For external memory references, Port 0 can provide address bits A8-A11 (lower nibble) or A8-A15 (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} by programming P01M (R248).

1.5.4 Port 2

Individual bits of Port 2 can be configured as input or output by programming the Port 2 Mode register (P2M) R246.

The port is accessed as general register R2. As with Ports 0 and 1, the port is written by specifying R2 as the destination register of an instruction, and the data is stored in the output register. The port is read by specifying R2 as the source register of an instruction.

Port 2 may be placed under handshake control by programming the Port 3 Mode register (P3M) R247. In this configuration Port 3 pins P31 and P36 are used as the handshake control lines $\overline{\text{DAV2}}$ and RDY2 for input handshake, or RDY2 and $\overline{\text{DAV2}}$ for output handshake. The handshake signal assignment for lines P31 and P36 is dictated by the direction (input or output) assigned to bit 7 of Port 2.

Port 2 can also be configured to provide open-drain outputs by programming the Port 3 Mode register (P3M).

Because all external memory addresses originate from Ports 0 and 1, Port 2 is always available for I/O operations in both the memory-intensive or I/O-intensive configurations.

1.5.5 Port 3

The Port 3 lines can be configured as I/O or control lines by programming the Port 3 Mode register (P3M) R247. In either case the direction of the eight lines is fixed as four input (P30-P33) and four output (P34-P37).

Port 3 is accessed as general register R3. When reading the port, the data present on the four input pins (P30-P33) and the data stored in the output register (P34-P37) is read. The four bits of the output register can be written only if they are used as data outputs.

For serial I/O, lines P30 and P37 are programmed as serial in and serial out respectively. Refer to Section 1.5.6 for details.

Port 3 control functions (Table 1-1) are defined by programming the P3M register. These control functions can provide the following signals: handshake for Ports 0, 1 and 2 ($\overline{\text{DAV}}$ and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (Tin and Tout) and external Data Memory Select ($\overline{\text{DM}}$).

FUNCTION	LINE	SIGNAL
Handshake	P31	$\overline{\text{DAV2}}/\text{RDY2}$
	P32	$\overline{\text{DAV0}}/\text{RDY0}$
	P33	$\overline{\text{DAV1}}/\text{RDY1}$
	P34	$\text{RDY1}/\overline{\text{DAV1}}$
	P35	$\text{RDY0}/\overline{\text{DAV0}}$
	P36	$\text{RDY2}/\overline{\text{DAV2}}$
Interrupt Request	P30	IRQ3
	P31	IRQ2
	P32	IRQ0
	P33	IRQ1
Counter/ Timer	P31	Tin
	P36	Tout
Status Out	P34	$\overline{\text{DM}}$

Table 1-1. Port 3 Control Functions

Note that the four input lines, regardless of the configuration chosen, can always be used as interrupt request inputs. However, the interrupt is generated only if the Interrupt Mask register (IMR) R251 is appropriately programmed.

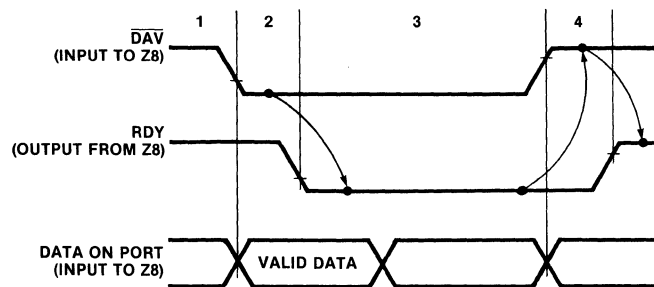
1.5.6 Port Handshake

Ports 0, 1 and 2 can transfer data using the interlocked handshake signals Ready (RDY) and Data Available ($\overline{\text{DAV}}$). Use of this feature is optional. These handshake signals are interlocked so the data transfer can be an asynchronous operation. A pair of Port 3 lines (one handshake output and one handshake input) is required for each of the other ports used in the handshake. The pair of handshake signals function as Ready (output) and Data Available (input) when the port is in the input mode, or as Data Available (output) and Ready (input) when the port is in the output mode.

Note that the user cannot write the output handshake lines of Port 3, but can always read the output and input handshake lines. It is recommended that, while enabling the handshake sequence for the first time, the user:

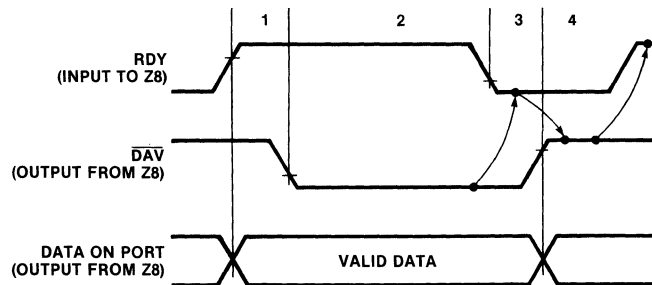
- o Set the port for input/output.
- o Set the output handshake bit of Port 3 to logic 1.
- o Select the handshake mode for the port.

Figure 1-17 describes the detailed operation for the various phases of the handshake sequence. Once the data transfer begins, the direction of the handshake signals must not be changed until the handshake sequence has been completed. In the input mode, data is latched into the input register by the first Data Available signal, and is protected from being overwritten when additional pulses on the Data Available input occur, until the data is read. This is not the case for the output mode. In the output mode, the data



- State 1.** Port 3 Ready output is High, indicating that the Z8 is ready to accept data.
- State 2.** The I/O device places data on the port and then activates the Data Available input (\overline{DAV}). This generates an interrupt request.
- State 3.** The Z8 forces the Ready output (RDY) Low, acknowledging to the I/O device that the data has been latched. The I/O device can then force \overline{DAV} High. The Z8 must respond to the interrupt request and read the contents of the Port so the handshake can be completed.
- State 4.** RDY goes High if and only if the port has been read and \overline{DAV} is High.

Figure 17a. Input Handshake Signals



- State 1.** The Z8 writes to the port register to initiate a data transfer. Writing the port forces \overline{DAV} Low, if and only if RDY is High, and outputs valid data.
- State 2.** The I/O device forces RDY Low after acceptance of data. RDY Low causes an interrupt request to be generated in the Z8.
- State 3.** A Low on RDY allows the Z8 to drive \overline{DAV} High.
- State 4.** After \overline{DAV} goes High, the I/O device is free to raise RDY High, signifying that it is ready for the next data.

Figure 17b. Output Handshake Signals

written to a port can be overwritten by the Z8 during the handshake sequence, thus requiring that the user provide software protection for data.

In applications requiring a strobed input or strobed output for data transfer instead of the interlocked handshake sequence, the user can satisfy the Z8 handshake signal requirements as follows:

- o In the strobed input mode, the user can ignore the Ready output and load data using the Data Available signal, at a rate that allows enough time for the Z8 to accept the data before the next data character is loaded.
- o In the strobed output mode, the user can tie the Ready input to the Data Available output.

1.5.7 Serial Input/Output

Port 3 lines P30 and P37 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 (T0), and has a maximum data rate of 62.5 kilobits per second ($f_{XTAL}/128$ with an 8 MHz clock). The data to be transmitted is loaded into register R240 and shifted out via P37 (Figure 1-18). The serial data is received through P30, assembled into an 8-bit character, and transferred to the receive buffer. Register R240 is actually two registers: when written into, it is the transmitter; when read from, it is the receiver buffer.

The T0 counter/timer runs at 16 times the bit rate to synchronize the incoming data stream. For easy derivation of commonly used asynchronous data communications bit rates, a 7.3728 MHz crystal can be used for the Z8 clock input. Table 1-2 lists the different bit rates and their required T0 initial values.

In the transmit mode, the Z8 automatically adds a start bit and two stop bits to the transmitted data. The Z8 also provides odd parity if control register R247 (P3M) is programmed accordingly. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. Between characters, the P37 output is held High to maintain the mark condition.

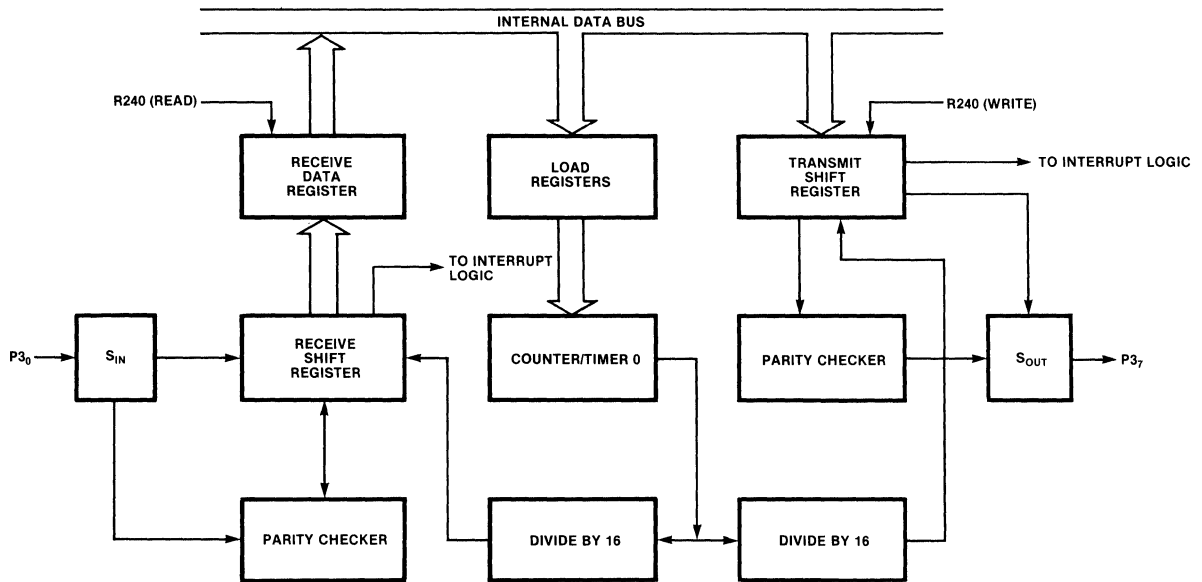


Figure 1-18. Serial I/O Block Diagram

BIT RATES	TO PRESET VALUE
19200	1
9600	2
4800	4
2400	8
1200	16
600	32
300	64
150	128
110	175 (error 0.3%)

Notes: 7.3728 MHz crystal; T0 prescaler=3

Table 1-2. Derivation of Common Bit Rates

In the receive mode, the data format must have a start bit, eight data bits, and at least one stop bit (Figure 1-19). If parity is on, bit 7 of the data received (parity bit) is replaced by a parity error flag. An error sets the flag to a logic 1.

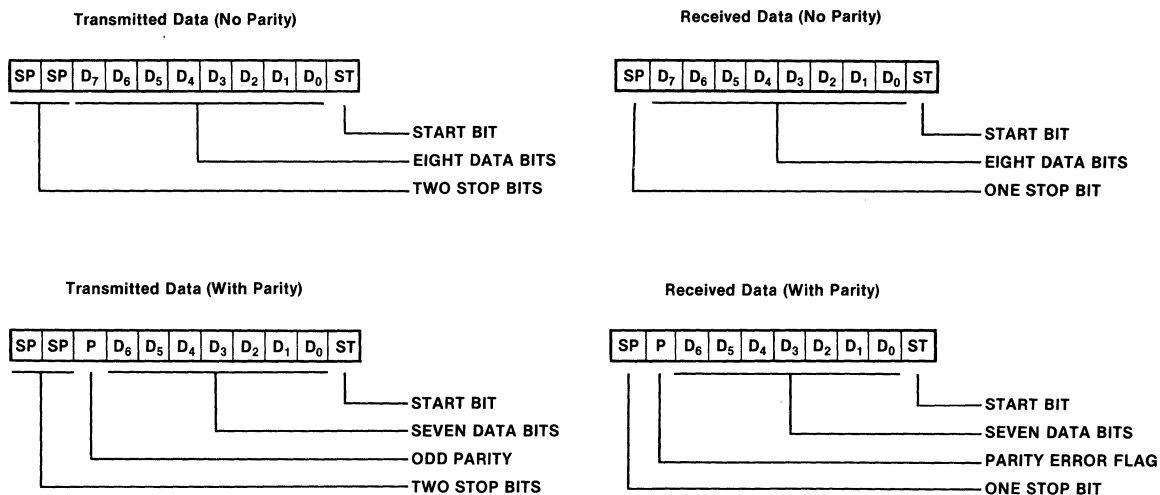


Figure 1-19. Serial Data Formats

An interrupt request (IRQ3) is generated whenever a character is transferred into the receive buffer. Although the receiver is double-buffered, it is not protected from being overwritten. A transmitted character also generates an interrupt request (IRQ4) and, like the receive buffer, the transmit buffer can also be overwritten.

1.6 COUNTER/TIMERS

The Z8 contains two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler (Figure 1-20). The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only. Both counter/timers can operate independently from the processor instruction sequence, thereby unburdening the program from time-critical operations such as event counting or elapsed-time calculations.

Registers R243 and R245 program the 6-bit prescalers to divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter (R244 for T0; R242 for T1), 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--IRQ4 (T0) or IRQ5 (T1)--is generated. The value N should be loaded for a count of N.

The counters can be started, stopped, restarted to continue, or restarted from the initial value by

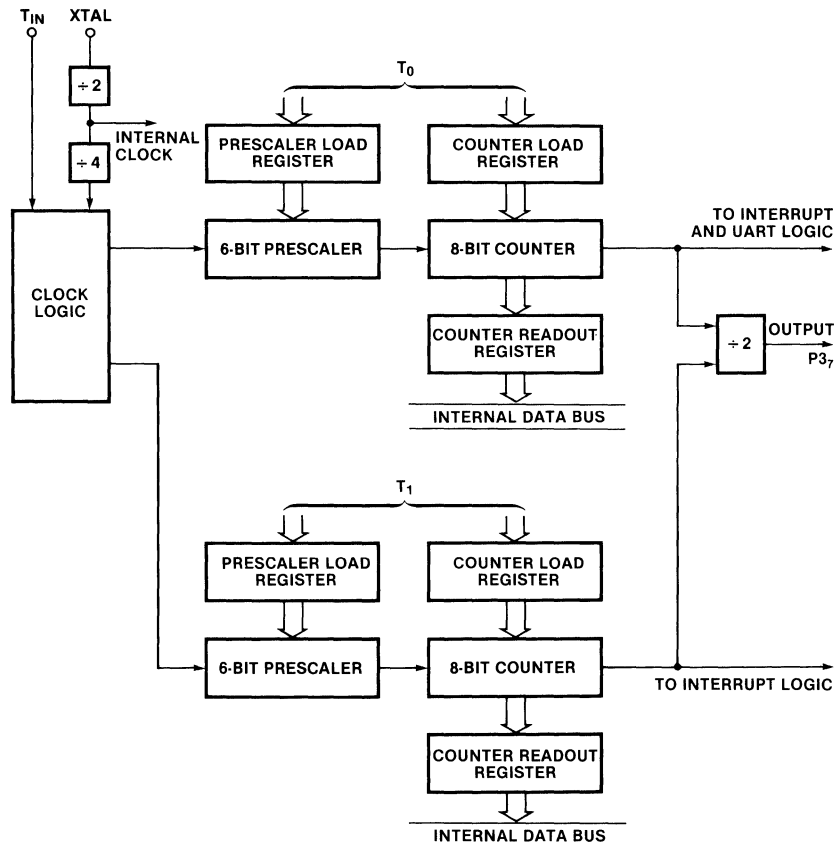


Figure 1-20. Counter/Timer Block Diagram

programming the Timer Mode register (TMR) R241. 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) by programming the Prescaler 0 (PRE0) and Prescaler 1 (PRE1) control registers. The counter, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T1 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-triggerable, or as a gate input for the internal clock. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

Port 3 line P36 also serves as a timer output (T_{out}) through which T0, T1 or the internal clock can be output. The timer output toggles at the end of count. If the timer is programmed in a continuous count mode, P36 generates a

50% duty cycle output. Tout resets whenever new initial values are loaded in T0 or T1 from the load registers either by a software load command or by an external trigger input (T1 only).

In the modulo-n count mode, new values for both counters can be written into the load register without affecting the ongoing countdown operation. When end-of-count is reached, the new initial values are loaded for subsequent counting operations.

1.7 INTERRUPTS

The Z8 allows six different interrupts from eight sources: the four Port 3 lines P30-P33, serial in, serial out, and the two counter/timers. These interrupts can be masked and prioritized using the Interrupt Mask Register (IMR) R251 and the Interrupt Priority Register (IPR) R249. All six interrupts can be globally disabled by resetting the master interrupt enable bit in the Interrupt Mask Register (IMR) R251.

All Z8 interrupts are vectored. When an interrupt occurs, control passes to the service routine pointed to by the specific location in program memory reserved for that interrupt. This program memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Table 1-3 lists the available interrupts, their sources, types and vector locations. Since T0 supplies the clock for serial I/O operation, the interrupts from the T0 counter and Serial Out are mutually exclusive and both use the same interrupt request line IRQ4. Similarly, the Serial In interrupt is combined with IRQ3 since serial data is input through P30 (IRQ3). The four inputs of Port 3 (P30-P33) always are the interrupt request inputs IRQ0-IRQ3. A High-to-Low transition on their inputs always generates an interrupt request.

Six bits in the Interrupt Mask register R251 can individually enable or disable the six interrupt requests IRQ0-IRQ5. Bit 7 globally disables all interrupts. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register R249. The output of the priority encoder points to the program memory vector location associated with the interrupt request being serviced. Before the contents of the Interrupt Mask Register (IMR) or the Interrupt Priority Register (IPR) are changed, the enable interrupt bit of the IMR must be reset

NAME	SOURCE	VECTOR LOCATION	COMMENTS
IRQ0	$\overline{\text{DAV0}}$, IRQ0	0,1	External (P3 ₂), † Edge Triggered
IRQ1	$\overline{\text{DAV1}}$, IRQ1	2,3	External (P3 ₃), † Edge Triggered
IRQ2	$\overline{\text{DAV2}}$, IRQ2, T _{IN}	4,5	External (P3 ₁), † Edge Triggered
IRQ3	IRQ3	6,7	External (P3 ₀), † Edge Triggered
	Serial In	6,7	Internal
IRQ4	T ₀	8,9	Internal
	Serial Out	8,9	Internal
IRQ5	T ₁	10,11	Internal

Table 1-3. Interrupts Types, Sources and Vector Locations

by the Disable Interrupt (DI) instruction. Use of the DI instruction is mandatory for correct interrupt handling.

When an interrupt request is granted, the Z8 enters an interrupt machine cycle that globally disables all subsequent interrupts, saves the program counter and status flags, and branches to the address contained within the vector location for the interrupt. Only at this point does control pass to the interrupt service routine. Figure 1-21 illustrates the interrupt cycle process when an interrupt request occurs. Figure 1-12 illustrates the actual interrupt timing sequence.

Interrupts can be re-enabled by the interrupt handling routine (EI instruction) to allow interrupt nesting. Interrupts can also be re-enabled automatically by issuing an Interrupt Return (IRET) instruction as the last instruction of the interrupt handling routine. IRET also restores the program counter and status flags.

The Z8 supports both polled and interrupt-driven systems. To accommodate a polled structure, any or all of the IRQ inputs can be masked and the Interrupt Request register polled to determine which of the normal interrupt request needs service.

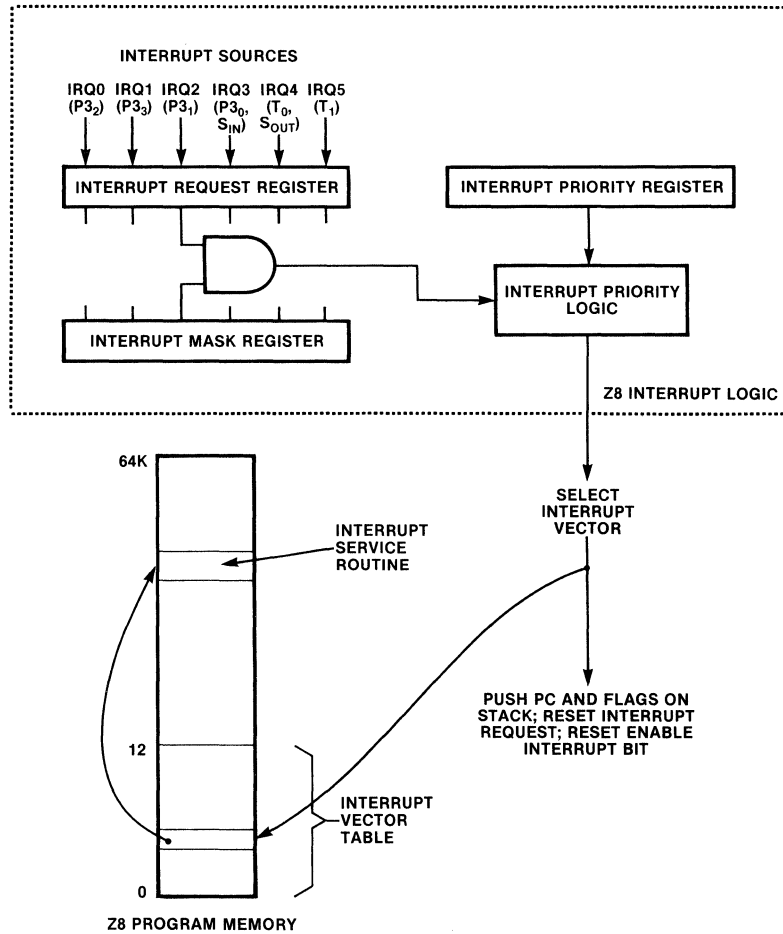


Figure 1-21. Interrupt Cycle Process

1.8 Status Flags

Flag register R252 contains eight flags:

C Carry	S Sign	D Decimal Adjust	F1 User Flag 1
Z Zero	V Overflow	H Half Carry	F2 User Flag 2

User flags F1 and F2 are available to the programmer for general use. The half-carry and decimal-adjust flags are specialized flags that are used only by specific instructions. The remaining flags can be used by the programmer with Jump and Jump Relative instructions to provide a repertoire of 19 conditional tests. Chapter 4 shows how the flags are affected by the Z8 instruction set.

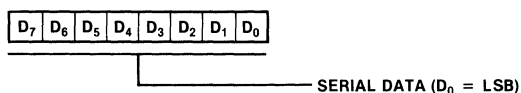
The flags can be set or reset by instructions; however, only those instructions that do not affect the flags as an outcome of the execution should be used (e.g., Load Immediate). In addition, the carry flag can be set to 1, by the Set Carry Flag (SCF) instruction, cleared to 0 by the Reset Carry Flag (RCF) instruction, or complemented by the Complement Carry Flag (CCF) instruction.

CHAPTER 2. CONTROL REGISTERS

Previous sections of this manual have provided a functional overview of the Z8 and have mentioned that it is configured by programming control registers R240 through R255. The following sections describe these registers in detail.

2.1 R240 SERIAL I/O REGISTER (SIO)

R240 (F0_H) Serial I/O Register (SIO; Read/Write)

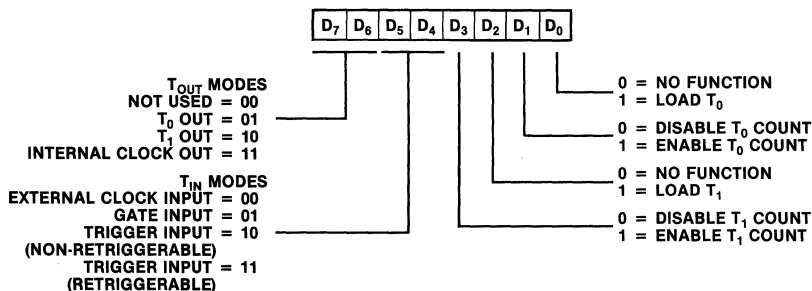


Read R240 = Receive Data
Write R240 = Transmit Data

Register R240 is used as the serial I/O data register when R247, D6 is set to 1 to configure P30 and P37 as serial input and output lines. If parity is not selected, all eight bits of R240 transmit data. If parity is selected R240, D7 contains odd parity during serial transmission and a parity error flag during reception. Parity is selected by setting R247, D7. When R240 is read, the character in the receive buffer is read; when written, a character is loaded into the transmitter.

2.2 R241 TIMER MODE REGISTER (TMR)

R241 (F1_H) Timer Mode Register (TMR; Read/Write)



This register selects the counter/timer clock modes and controls the operation of T0 and T1.

Load T0 (D0). The contents of the T0 Load register and the T0 Prescaler Load register are transferred to the T0 counter and the prescaler one clock period after this bit is set. This operation alone does not start the counter. D0 is automatically reset following the load, or after a master reset.

T0 Enable Count (D1). D1 enables or disables T0 counting operations. When set, the values in T0 and its prescaler are counted down using the internal clock. When reset, the countdown operation is discontinued. This bit is reset following a master reset. It is permissible to set both the load bit D0 and the enable count bit D1 simultaneously.

Load T1 (D2). This bit has the same function for T1 as D0 has for T0.

T1 Enable Count (D3). This bit has the same functions for T1 as D1 has for T0.

External Timer Input Modes (D4, D5). D4 and D5 are encoded to define four modes of the External Timer input (Tin) for counter/timer (Table 2-1). P34 must first be defined as an External Timer input (R243, D1)

Counter/Timer Output Modes (D6, D7). These two bits are encoded to define four output modes for the Timer Out signal Tout. Port 3 pin P36 is implicitly defined to operate as Tout when R241, D6 or D7 is set. When used with T0 or T1, Tout is toggled by the end-of-count from T0 or T1. When D6 and D7 are both set to 0 (the unused function), P36 defaults to a data output bit and is controlled by R247, D5.

D7	D6	Tout Function
0	0	
0	1	T0 Output
1	0	T1 Output
1	1	System clock (Xtal Freq. divided by 2)

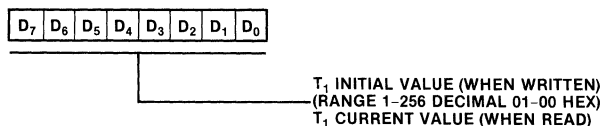
D5	D4	Tin USED AS	COMMENT
0	0	External Clock input	Tin is used as an external clock for T1. In this mode, the external clock bypasses the divide-by-four counter and directly drives the prescaler.
0	1	Gate input for internal clock (active High)	T1 is clocked by the internal clock (XTAL frequency divided by eight) gated by this input. If enabled, an interrupt is generated when the gate input switches from High to Low.
1	0	Non-retriggerable Trigger input	T1 is loaded and clocked with the internal clock only after a High-to-Low transition occurs on this input. Further transitions do not affect T1 operation until after the end-of-count.
1	1	Retriggerable Trigger input	T1 is loaded and clocked with the internal clock only after a High-to-Low transition occurs on Tin. When additional trigger pulses occur on Tin, the initial value of T1 is reloaded and counting is restarted.

Table 2-1. Tin Functions and Operation

2.3 R242 COUNTER/TIMER REGISTER (T1)

This address serves two registers: as a write register, it stores the T1 initial value; as a read register, it contains the current count value of T1. R242 can be loaded with values that range from 1 (01H) to 256 (00H). This value is transferred into T1 by setting the Load T1 bit (R241, D2) and is subsequently counted down. An interrupt request is generated when the end-of-count is reached.

R242 (F2H) Counter/Timer 1 Register (T1; Read/Write)

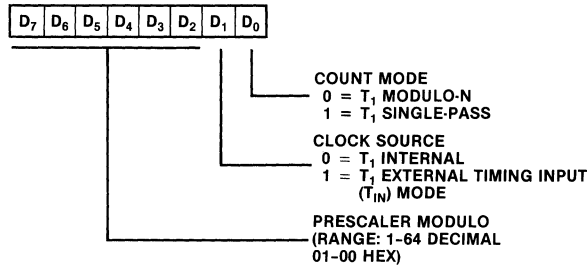


Write R242 = Initial Value
 Read R242 = Current Count Value

2.4 R243 T1 PRESCALER LOAD REGISTER (PRE1)

This register holds the T1 prescaler initial value, and defines the T1 clock source and count mode.

R243 (F3_H) Prescaler 1 Register (PRE1; Write Only)



Mode Selection (D0). When this bit is set, T1 operates in a single-pass count mode, in which the value in T1 is counted down to zero once for each Load T1 command (R241, D2). An interrupt request is generated when the counter reaches end-of-count.

When this bit is reset, T1 operates in the modulo-n (continuous) count mode. Upon receiving the Load T1 command, T1 initial values are loaded and counted down until end-of-count is reached. The initial values are reloaded and counted down as long as the T1 enable count bit (R241, D3) is set. New values can be loaded into the load register without affecting the counting operation. When end-of-count is reached, the new initial values are loaded into the counter for subsequent count cycles.

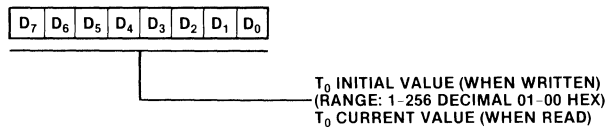
T1 Clock Source Selection (D1). When D1 is set, T_{in} supplies the T1 clock. When reset, the internal clock (system clock divided by 4) supplies the T1 clock. When T_{in} is used for the clock, the appropriate modes must be encoded in the TMR register (R241).

T1 Prescaler Value (D2-D7). The remainder of the bits in R243 contain a 6-bit binary value that determines the modulus of the prescaler. D2 is the least significant bit. The value should be loaded for a prescale of n.

2.5 R244 COUNTER/TIMER 0 REGISTER (T0)

This register provides the same functions for T0 as R242 does for T1.

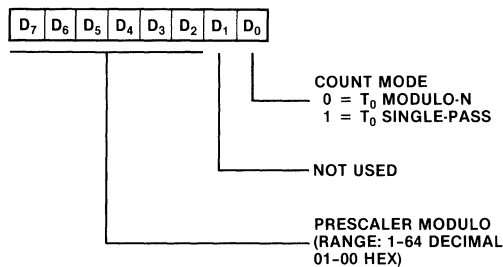
R244 (F4_H) Counter/Timer 0 Register (T0; Read/Write)



2.6 R245 T0 PRESCALER REGISTER (PRE0)

This register has the same functions as its T1 counterpart (R243), except that D1 is not used.

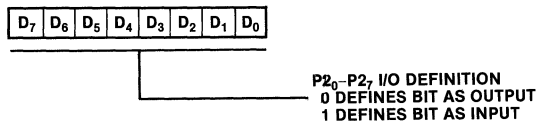
R245 (F5_H) Prescaler 0 Register (PRE0; Write Only)



2.7 R246 PORT 2 MODE (P2M)

The Port 2 Mode register programs each bit of Port 2 as an input or output line. When a bit of R246 is set, the corresponding line of Port 2 is defined as an input. When reset, the corresponding line is defined as an output. Following a master reset, R246 contains FF_H and all Port 2

R246 (F6_H) Port 2 Mode Register (P2M; Write Only)

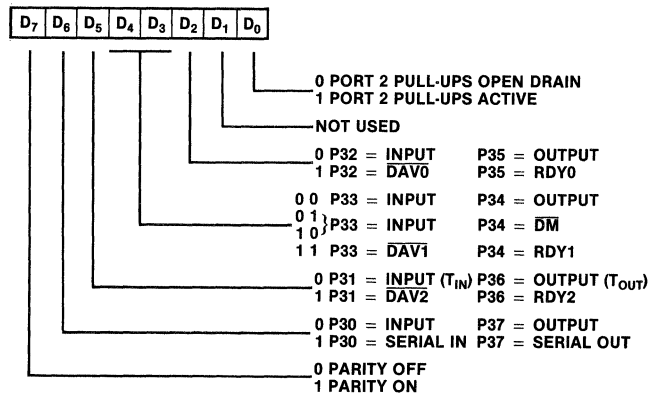


lines are defined as inputs. The mode of Port 2 is further specified by R247, D0.

2.8 R247 PORT 3 MODE (P3M)

The Port 3 Mode Register specifies which lines of Port 3 are used for parallel I/O, serial I/O, interrupt request, timer I/O, and handshake or status outputs.

R247 (F7_H) Port 3 Mode Register (P3M; Write Only)



Port 2 Pullups (D0). Resetting this bit provides open-drain outputs for Port 2 by inhibiting the active pullups. Open-drain outputs allow Port 2 lines to be OR-tied with other signals.

NOT USED (D1). Bit 1 is not used.

P32 and P35 Mode (D2). Bit D2 specifies whether P32 and P35 are used for Port 3 I/O, or as handshake lines that support Port 0.

D2	FUNCTION	
0	P32=Input	P35=Output
1	P32=DAV0/RDY0	P35=RDY0/DAV0

P33 and P34 Mode (D3, D4). Bits D3 and D4 specify whether P33 and P34 are used for I/O, or support Port 1 functions.

D4 and D3		FUNCTION	
0	0	P33 = Input	P34 = Output
0	1	P33 = Input	P34 = \overline{DM}
1	0		
1	1	P33 = $\overline{DAV1}/RDY1$	P34 = $\overline{RDY1}/\overline{DAV1}$

P31 and P36 Mode (D5). Bit D5 determines whether Port 3 lines P31 and P36 are configured as I/O (D5=0), or as handshake controls (D5=1) used to support Port 2. When programmed for I/O, P31 and P36 normally reflect register R3 bits 1 and 6; however, other data sources can be selected as follows.

If bit D1 of T1 Prescaler register R243 is set, P31 becomes the timer input (T_{in}) control and behaves in accordance with the truth table describing the Timer Mode register R241, bits D4 and D5. R243, bit D1 must be reset for P31 to act as a data input line for R3.

P36 output becomes T_{out} (for either T0 or T1) or SCLK depending on the values of the Timer Mode register R241, bits D6 and D7. For P36 to act as a data output for R3, bits D6 and D7 of R241 must both be reset to zeroes.

If P31 and P36 are selected as handshake controls, R241 (D4-D7) and R245 (D1) have no effect.

D5	FUNCTION	
0	P31=Input (T_{in})	P36=Output (T_{out})
1	P31= $\overline{DAV2}/RDY2$	P36= $\overline{RDY2}/\overline{DAV2}$

P30 and P37 Mode (D6). D6 determines whether Port 3 lines P30 and P37 are used for I/O or the serial receiver/transmitter interface.

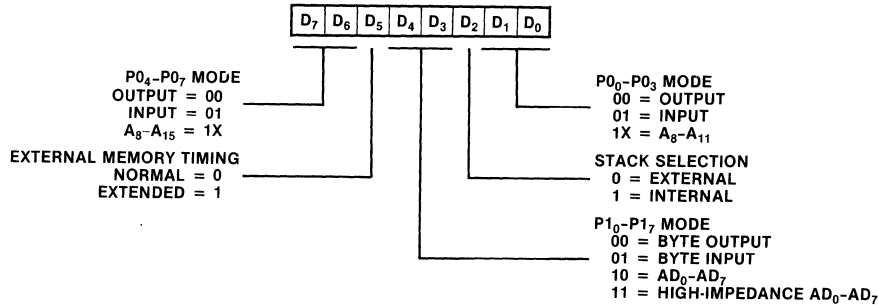
D6	FUNCTION	
0	P30=Input	P37=Output
1	P30=Serial In	P37=Serial Out

Parity (D7). If serial I/O is selected by D6, setting D7 provides odd parity for transmitted data and odd parity checking for received data.

2.9 R248 PORTS 0 AND 1 MODES (P01M)

This register configures Ports 0 and 1, selects an internal or external stack, and selects normal or extended memory timing.

R248 (F8_H) Ports 0 and 1 Mode Register (P01M; Write Only)



Port 0 Mode (D0, D1). These two bits are encoded to set the mode of Port 0 lines P00-P03. Following a reset, the Port 0 lower nibble is configured in the input mode. Note that--when R248, D7 is set--P00-P03 become A8-A11 regardless of the configuration set by D0 and D1.

D7	D1	D0	P00-P03 CONFIGURATION
0	0	0	4-bit output
0	0	1	4-bit input
0	1	x	4-bit address (A8-A11)
1	x	x	4-bit address (A8-A11)

Stack Location (D2). When D2 is set, the stack is internal in the register file and is pointed to by the stack pointer SPL (R255). When reset, the stack is located in external data memory and is pointed to by the stack pointer SP (R254 and R255). When the internal stack is selected R254 can be used as a data register; however, care should be taken to properly handle the overflow/underflow from R255.

Port 1 Mode (D3, D4). Bits D3 and D4 are encoded to define the mode of the Port 1 lines P10-P17. Following a reset, Port 1 is configured as an 8-bit input port.

D4	D3	P10-P17 CONFIGURATION
0	0	8-bit output
0	1	8-bit input
1	0	8-bit multiplexed address/data (AD0-AD7)
1	1	High-impedance state

The high-impedance state and the address modes selected by R248 are interdependent as shown in the following table.

D7	D4	D3	D1	3-STATED LINES
0	1	1	0	Port 1, \overline{AS} , \overline{DS} , R/ \overline{W}
0	1	1	1	Port 1, \overline{AS} , \overline{DS} , R/ \overline{W} and P00-P03
1	1	1	x	Port 1, \overline{AS} , \overline{DS} , R/ \overline{W} and P00-P07

Extended Memory Timing (D5). When set, this bit extends the memory cycle by one clock period to allow interfacing slower memory. When reset, the normal external memory timing is selected.

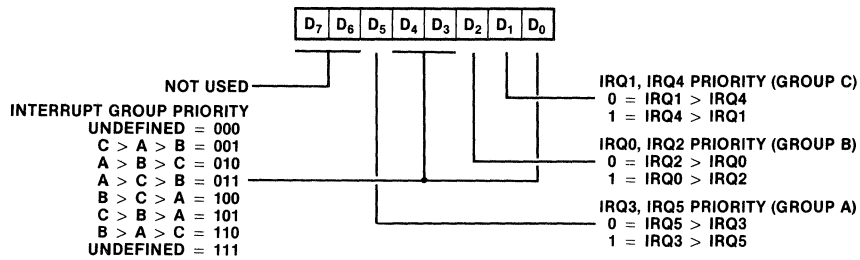
Port 0 Mode (D6 and D7). Bits D6 and D7 control the mode of Port 0 lines P04-P07. Following a reset, the Port 0 upper nibble is configured for the input mode. When D7 is set, P00-P03 are A8-A11 regardless of any other configuration set by D0 and D1.

D7	D6	P04-P07 CONFIGURATION
0	0	4-bit output
0	1	4-bit input
1	x	4-bit address (A12-A15)

2.10 R249 INTERRUPT PRIORITY REGISTER (IPR) Write only

This register prioritizes the six levels of vectored interrupts. The interrupts can be prioritized in 48 different orders to resolve simultaneous interrupt requests. The six interrupt levels IRQ0-IRQ5 are divided into three groups--arbitrarily named A, B and C for this discussion--each containing two interrupt requests. Priority group A contains IRQ3 (SI/P30) and IRQ5 (T1), group B contains IRQ0 (P32) and IRQ2 (P31), and group C IRQ1 (P33) and IRQ4 (S0/T0).

R249 (F9_H) Interrupt Priority Register (IPR; Write Only)



Bits D1, D2 and D5 define the priority of the individual members within the groups.

GROUP	BIT	PRIORITY	
		HIGHEST	LOWEST
C	D1=0	IRQ1	IRQ4
	1	IRQ4	IRQ1
B	D2=0	IRQ2	IRQ0
	1	IRQ0	IRQ2
A	D5=0	IRQ5	IRQ3
	1	IRQ3	IRQ5

Priorities can be set between groups A, B and C as well as within a group. Bits D0, D3 and D4 of R249 are encoded to define six priority orders for groups A, B and C.

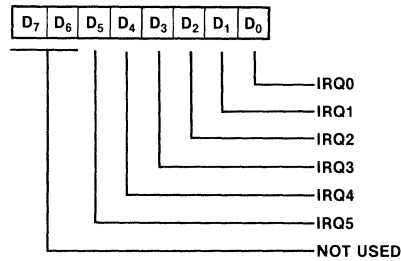
BIT PATTERN			GROUP PRIORITY
D4	D3	D0	HIGHEST --> LOWEST
0	0	0	NOT USED
0	0	1	C A B
0	1	0	A B C
0	1	1	A C B
1	0	0	B C A
1	0	1	C B A
1	1	0	B A C
1	1	1	NOT USED

Bits D6 and D7 are not used.

2.11 R250 INTERRUPT REQUEST REGISTER (IRQ)

This register stores the interrupt requests. Since it is a read/write register, it can also be used for polling. When an interrupt is made on any of the six levels, a 1 is written into the corresponding bit position of the interrupt request register. Bits D0-D5 are assigned to interrupt requests IRQ0-IRQ5 respectively. D6 and D7 are not used.

R250 (FA_H) Interrupt Request Register (IRQ; Read/Write)

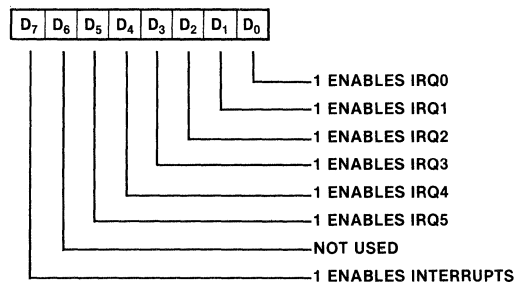


2.12 R251 INTERRUPT MASK REGISTER (IMR)

This register individually or globally enables or disables the six interrupt requests. When bits D0-D5 are set, the corresponding interrupt requests are enabled. D7 is the master enable and must be set before any of the individual interrupt requests can be recognized. Resetting D7 globally disables all of the interrupt requests. D7 can be set and reset by the Enable Interrupt (EI) and Disable Interrupt instructions. It is automatically reset during an interrupt machine cycle and set following the execution of the Interrupt Return instruction (IRET).

D7 MUST BE RESET BY THE DI INSTRUCTION BEFORE THE CONTENTS OF THE INTERRUPT MASK REGISTER OR THE INTERRUPT PRIORITY REGISTER ARE CHANGED.

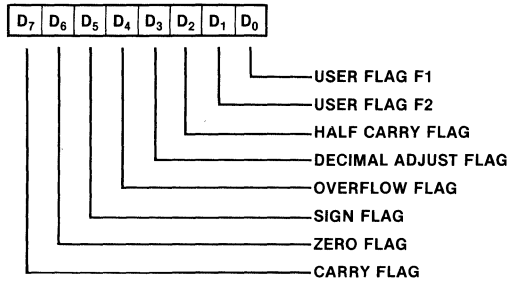
R251 (FB_H) Interrupt Mask Register (IMR; Read/Write)



2.13 R252 FLAG REGISTER (FLAGS)

Bits D2-D7 contain the status flags as shown. D0 and D1 are user-definable.

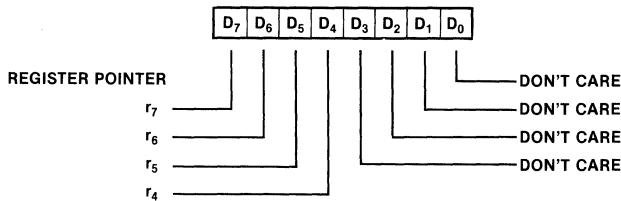
R252 (FC_H) Flag Register (Flags; Read/Write)



2.14 R253 REGISTER POINTER (RP)

The four upper bits of this register form a register pointer that points to the origin of the current working register group. The lower four bits (register designator) that specify the individual register within the working register group are supplied by the instruction. In this register, the four lower-order bits (D0-D3) are always zero when read and don't cares when written.

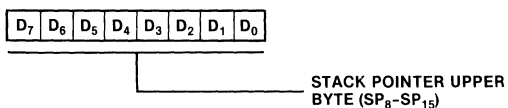
R253 (FD_H) Register Pointer (RP; Read/Write)



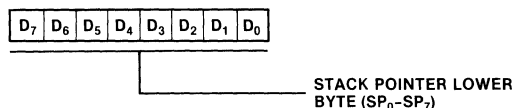
2.15 R254, 255 STACK POINTER (SPL, SPH)

The stack pointer is composed of two 8-bit registers: SPL (R255) contains the lower address byte; SPH (R254) contains the upper byte. As mentioned previously bit D2 of R248 specifies whether the stack is located in the register file or in external memory. If the stack is internal, R254 is not used as a stack pointer and is available as a data register as long as the overflow/underflow from R255 is properly handled.

R254 (FE_H) Stack Pointer (SPH; Read/Write)



R255 (FF_H) Stack Pointer (SPL; Read/Write)



2.16 R240 TO R255 REGISTER RESET VALUES

The contents of the control registers following a reset are defined in Table 2-2. The symbol X means "don't care."

CONTROL REGISTER	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	COMMENT
R240 Serial I/O Register	Not Defined								
R241 Timer Mode	0	0	0	0	0	0	0	0	Stops T ₀ and T ₁ .
R242 T ₁ Register	Not Defined								
R243 T ₁ Prescaler	X	X	X	X	X	X	0	0	Modulo-n count mode
R244 T ₀ Register	Not Defined								
R245 T ₀ Prescaler	X	X	X	X	X	X	X	0	Modulo-n count mode, internal clock source.
R246 Port 2 Mode	1	1	1	1	1	1	1	1	Port 2 lines defined as inputs.
R247 Port 3 Mode	0	0	0	1	0	0	X	0	Port 2 pull-ups open drain; P3 ₀ -P3 ₃ defined as inputs and P3 ₄ -P3 ₇ defined as outputs.
R248 Ports 0 and 1 Mode	0	1	0	0	1	1	0	1	Port 0, Port 1 defined as inputs, normal memory cycle, internal stack.

Table 2-2. Control Register Reset Conditions

R249		Not Defined
Interrupt Priority		
R250	X X 0 0 0 0 0 0	Reset interrupt requests
Interrupt Request		
R251	0 X X X X X X X	Interrupts disabled
Interrupt Mask		
R252		Not Defined
Flag Register		
R253		Not Defined
Register Pointer		
R254		Not Defined
Stack Pointer, Lower		
Stack Pointer, Upper		Not Defined

Table 2-2 Cont. Control Register Reset Conditions

CHAPTER 3. COMPONENT CONTROLS

3.1 POWER DOWN

The power down option allows power to be removed from the Z8 without losing the contents of the general-purpose registers. The XTAL2 output is replaced by a V_{MM} power supply input to power the 124 general-purpose registers R4-R127 and the reset logic. Since XTAL2 is replaced by V_{MM}, an external clock generator must be used to input the Z8 clock via the XTAL1 input.

The following sequence must occur to preserve data:

Power failure must be detected externally, early enough for a software routine to store the required data into the register file.

$\overline{\text{RESET}}$ must be held Low after data is saved and during removal of power.

$\overline{\text{RESET}}$ must be held Low during power-up to protect data.

Figure 3-1 shows the recommended circuit for a battery back-up supply system.

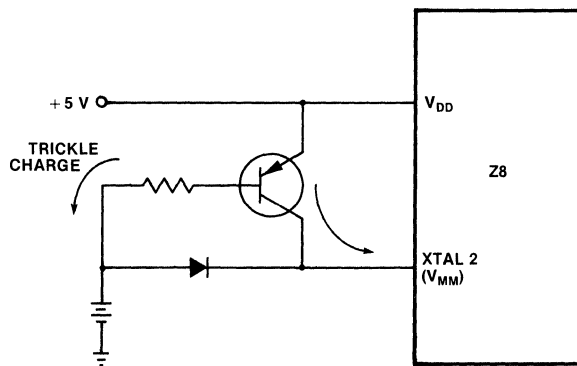


Figure 3-1. Recommended Driver Circuit for Power Down Operation

3.2 RESET

To initialize the Z8, the Reset input must be held Low for at least 50 ms after power is applied and is within the supply tolerance, or 18 clock cycles after the power supply is in tolerance and the clock oscillator has stabilized. During a power-up cycle, an internal pullup device combined with an external capacitor of 1 μ F provides enough time to properly reset the Z8 (Figure 3-2).

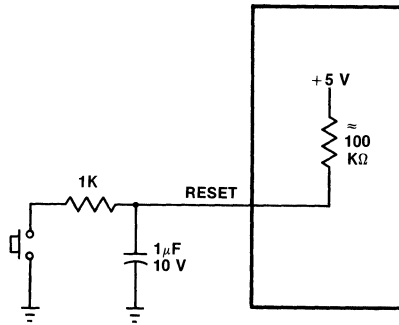


Figure 3-2. Power Up Reset Circuit

A reset configures Port 0, 1 and 2 as input ports, sets the program counter to location 12, and disables interrupts. In addition, the control registers are initialized as shown in Table 2-2.

After $\overline{\text{RESET}}$ is removed, execution begins at program memory location 12. It is recommended that the first software routine be used to configure the Z8.

$\overline{\text{RESET}}$ is also used in conjunction with the power-down option and the test mode operation.

3.3 CLOCK

The on-chip oscillator can be driven by a crystal (series resonant, AT cut), series RC, LC or an external clock source. The on-chip oscillator has a high-gain series-resonant amplifier, with XTAL1 as the input and XTAL2 the output.

When a crystal is used (8 MHz maximum), it is connected across the XTAL1 and XTAL2 pins. The crystal frequency is internally divided by two to generate the system clock (4 MHz maximum).

3.4 Test Mode

An additional 64 bytes of on-chip program memory has been provided to facilitate testing the Z8. To enter the test modes, the Reset input must be raised to a level greater than Vcc (see Electrical Characteristics, Chapter 6). This forces the program memory locations 0 to 63 to reference the test ROM rather than the user ROM. Program memory locations greater than 63 are the normal user space.

Zilog uses test ROM to configure the Z8 for fetching instructions from external program memory (i.e., the test system) via Port 1. These instructions are used to functionally test the Z8, and to verify the user's program by dumping the contents of the program ROM.

A normal reset forces the Z8 to exit the test mode.

CHAPTER 4. ADDRESSING MODES AND INSTRUCTION SET SUMMARY

This chapter explains the Z8 addressing modes and summarizes the instruction set. The detailed instruction set description is found in the Z8 PLZ/ASM Assembly Language Programming Manual.

4.1 ADDRESSING MODES

With the exception of immediate data and condition codes, all operands are expressed as register file addresses, program memory addresses or data memory addresses. The various addressing modes provided by the Z8 are:

Register	Direct
Indirect Register	Relative
Indexed	Immediate

The Summary of Instructions (Table 4-2) specifies the applicable addressing modes for each Z8 instruction.

4.1.1 Register Address

In the register addressing mode, the operand value is the contents of the specified register. The register can be addressed in one of two ways: by an 8-bit address in the range of 0-127, 240-255 (Figure 4-1); or by a 4-bit working-register address in the range of 0-15.

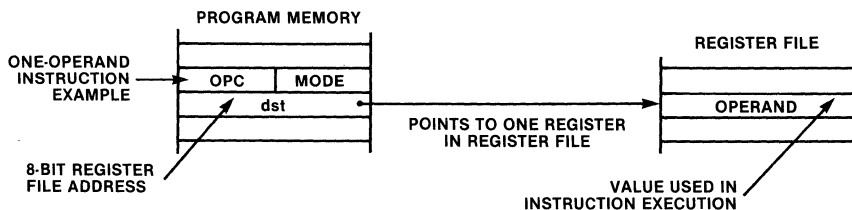


Figure 4-1. Register Addressing

Working-register Address. Designating a register by a 4-bit working-register address--rather than an 8-bit register-file address--reduces the length of an instruction and results in a shorter execution time. In this case, the full register-file address is formed by concatenating the 4-bit field (address range 0-15) with the upper four bits of the register pointer (Figure 4-2). Thus, the working-register set can be varied dynamically by changing the value of the register pointer (R253).

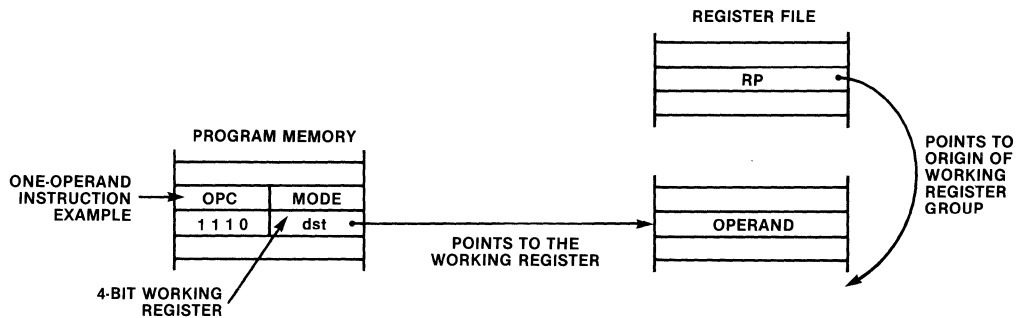


Figure 4-2. Working-Register Addressing

Register Pair Address. Registers can be used in pairs to designate 16-bit values or memory addresses. A register pair must be specified as an even number in the range 0, 2, 4, ..., 126 or 240, 242, 244, ..., 254.

Working-register Pair Address. Working-register pairs can also be used to designate 16-bit values or memory addresses. The allowable register pairs begin with an even number and are in the range 0, 2, 4, ..., 14.

4.1.2 Indirect-Register Address

In indirect addressing, the value of the operand is not the contents of a register. Instead, the register (register pair, working register or working-register pair) contains the address of the location whose contents are to be used as the operand value (Figures 4-3 and 4-4).

Depending on the instruction selected, the address may point to a register, program memory, or external data memory location.

Register pairs or working-register pairs are used to hold the 16-bit addresses when accessing program or external

data memory. Pairs are indicated by an even number (see 4.4.1).

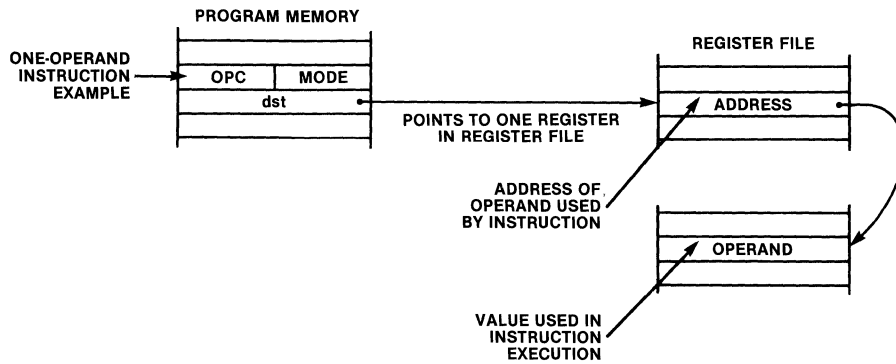


Figure 4-3. Indirect-Register Addressing in the Register File

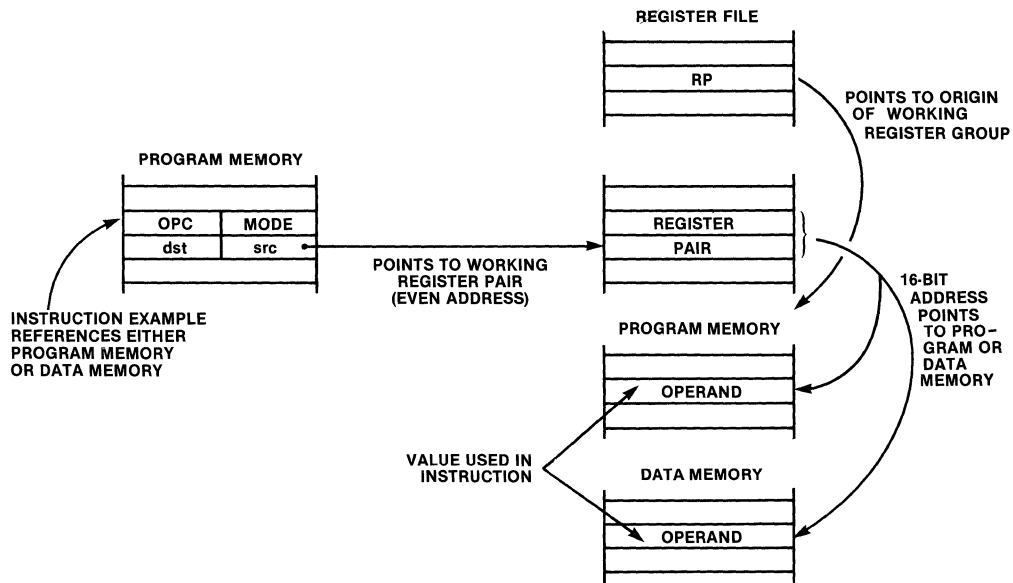


Figure 4-4. Indirect-Register Address with Program or Data Memory

4.1.3 Indexed Address

An indexed address consists of a register address offset by the contents of a designated working register (the index). This offset is added to the register address and the resulting address points to the location whose value is

used by the instruction (Figure 4-5). This address mode is used only by the Load (LD) instruction to address the register file.

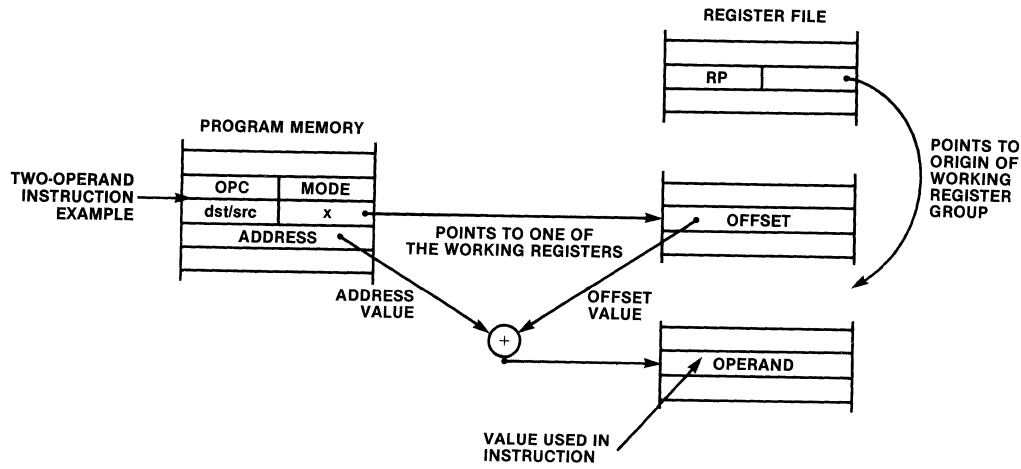


Figure 4-5. Indexed Addressing

4.1.4 Direct Address

Direct addressing is used only by Conditional Jump (JP) and Call (CALL) instruction to specify the destination where program control is to be transferred (Figure 4-6).

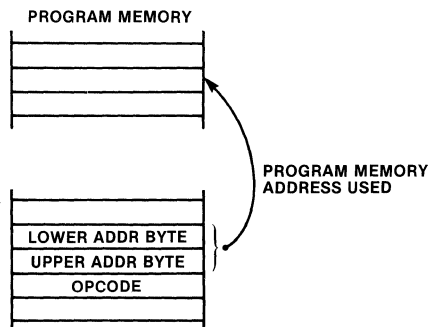


Figure 4-6. Direct Addressing

4.1.5 Relative Address

The relative-address mode is implied by its instruction. It is used only by the Jump Relative (JR) and the Decrement And Jump (DJNZ) instructions and is the only mode available to these instructions. In this case, the operand contains

a two's-complement offset that is added to the contents of the program counter to form the destination address (the program address where control is to be transferred). The content of the program counter is the address of the instruction following the JR or DJNZ instruction. The offset value is an 8-bit signed value in the range of -128 to +127 (Figure 4-7).

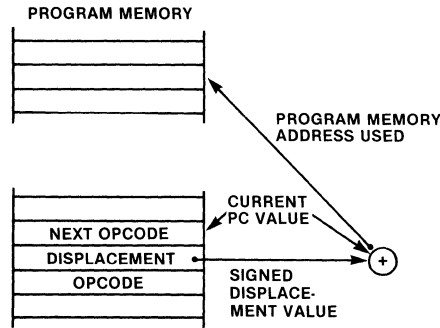


Figure 4-7. Relative Addressing

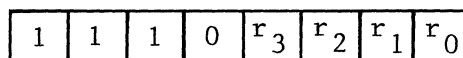
4.1.6 Immediate Data

Immediate data is considered an "address mode" for the purposes of this discussion. The operand value used by the instruction in this case is the value supplied in the operand field itself.

4.1.7 A Note on the Register Pointer

When a full 8-bit register designator is required by an instruction (PUSH R, for example), a mechanism is available that allows the lower four bits of the 8-bit register designator to be used in conjunction with the register pointer. Thus, all register instructions can specify a working register or a working register pair.

This mechanism is invoked by specifying register addresses using the following format.



Whenever the upper nibble of the register address is coded as a hexadecimal "E" (1110), the Z8 automatically uses the lower nibble as a 4-bit address in conjunction with the register pointer to form the effective 8-bit address.

Combinations of 4- and 8-bit address designators are allowed. These formats are shown in Figures 4-9 and 4-10.

4.2 INSTRUCTION SUMMARY

4.2.1 Functional Summary

Z8 instructions are functionally divided into eight groups:

Load	Bit Manipulation (Test)
Arithmetic	Block Transfer
Logical	Rotate and Shift
Program Control (Branch)	CPU Control

The following summary shows the instructions belonging to each group and the number of operands required for each, where "src" is the source operand, "dst" is the destination operand, and "cc" is a condition code.

Load Instructions		
Instruction	Operand (s)	Name of Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant
LDE	dst, src	Load External Data
POP	dst	Pop
PUSH	src	Push

Table 4-1. Z8 Instruction Set: Functional Groups

Arithmetic Instructions

Instruction	Operand (s)	Name of Instruction
ADC	dst, src	Add With Carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
SBC	dst, src	Subtract With Carry
SUB	dst, src	Subtract

Logical Instructions

Instruction	Operand (s)	Name of Instruction
AND	dst, src	Logical And
COM	dst	Complement
OR	dst, src	Logical Or
XOR	dst, src	Logical Exclusive Or

Program-Control Instructions

Instruction	Operand (s)	Name of Instruction
CALL	dst	Call
DJNZ	r, dst	Decrement and Jump If Nonzero
IRET		Interrupt Return
JP	cc, dst	Jump
JR	cc, dst	Jump Relative
RET		Return

Table 4-1. Z8 Instruction Set: Functional Groups

Bit-Manipulation Instructions

Instruction	Operands	Name of Instruction
TCM	dst, src	Test Complement Under Mask
TM	dst, src	Test Under Mask
AND	dst, src	Logical And
OR	dst, src	Logical Or
XOR	dst, src	Logical Exclusive Or

Block-Transfer Instructions

Instruction	Operands	Name of Instruction
LDCI	dst, src	Load Constant Autoincrement
LDEI	dst, src	Load External Data Auto-increment

Rotate and Shift Instructions

Instruction	Operand	Name of Instruction
RL	dst	Rotate Left
RLC	dst	Rotate Left Through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right Through Carry
SRA	dst	Shift Right Arithmetic
SWAP	dst	Swap Nibbles

Table 4-1. Z8 Instruction Set: Functional Groups

CPU Control Instructions

Instruction	Operand	Name of Instruction
CCF		Complement Carry Flag
DI		Disable Interrupts
EI		Enable Interrupts
NOP		No Operation
RCF		Reset Carry Flag
SCF		Set Carry Flag
SRP	src	Set Register Pointer

Table 4-1. Z8 Instruction Set: Functional Groups

4.2.2 Flags and Condition Codes

The Z8 provides six flags for program control. The Flags are effected by most instructions, and can be used by the programmer for conditional testing with the Jump and Jump Relative instructions.

Control register R252 contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Carry Flag. The carry flag is affected by the following instructions: Addition (ADD, ADC), Subtraction (SUB, SBC), Compare (CP), Decimal Adjust (DA), Rotate (RL, RLC, RR, RRC), Swap (SWAP), and Interrupt Return (IRET). When set, the carry flag generally indicates a carry from the bit 7 position of a register being used as an accumulator.

Zero Flag. The zero flag is affected by the same instructions as the carry flag plus the Logical (AND, OR, XOR, COM), Increment and Decrement (INC, INCW, DEC, DECW), and Test (TCM, TM) instructions. In general, the zero flag is set when the result of an operation is zero.

Sign Flag. The sign flag is affected by the same instructions as the zero flag. The sign flag is set when bit 7 of the result of an operation is a "1" (minus).

Overflow Flag. The overflow flag is affected by the same instructions as the zero and sign flags. When set, the overflow flag indicates that a two's-complement number of a result is in error because it has exceeded the range (-128 to +127) that can be expressed in two's-complement notation.

Decimal Adjust Flag. The decimal adjust flag is used for BCD arithmetic, and to adjust 8-bit binary numbers to form two 4-bit BCD digits. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag specifies the type of instruction last executed.

The decimal adjust flag is set to 1 whenever Subtract (SUB, SBC) instructions are executed and is reset to 0 whenever Add (ADD, ADC) instructions are executed.

Half-Carry Flag. The half-carry flag indicates a carry from the bit 3 position of a register being used as an accumulator. The half carry flag is affected as the result of the Subtract (SUB, SBC) or Add (ADD, ADC) instructions being executed.

Condition Codes

Jump and Jump Relative instructions use a 4-bit condition code field (CC) to specify the type of conditional test. These condition codes are summarized as follows:

Code	Meaning	Flags Set
(blank)-1000	Always true	---
C	Carry	C=1
NC	No carry	C=0
Z	Zero	Z=1
NZ	Not zero	Z=0
PL	Plus	S=0
MI	Minus	S=1
OV	Overflow	V=1
NOV	No overflow	V=0
EQ	Equal	Z=1
NE	Not equal	Z=0
GE	Greater than or equal	(S XOR V)=0
LT	Less than	(S XOR V)=1
GT	Greater than	(Z OR (S XOR V))=0
LE	Less than or equal	(Z OR (S XOR V))=1
UGE	Unsigned greater than or equal	C=0
ULT	Unsigned less than	C=1
UGT	Unsigned greater than	(C=0 AND Z=0)=1
ULE	Unsigned less than or equal	(C OR Z)=1
-0000	Never true	---

4.2.3 Notation

The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary (Table 4-2).

Addressing Modes

Symbol	Meaning
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

Symbol	Meaning
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)

The flags affected by each instruction are indicated by:

```

0: cleared to zero
1: set to one
*: set or cleared according to operation
-: unaffected
X: undefined

```

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.

4.2.4 Instruction Summary

The instruction summary (Table 4-2) defines the operation and applicable addressing modes for Z8 instructions. In addition, the opcodes, number of bytes, internal clock cycles and flags affected are listed for each instruction's addressing mode.

The number of internal clock cycles listed in the Execution Cycles column represents the values to be used to compute the execution time of a program. The time elapsed between the beginning of an instruction and when data changes is given by the sum of the Execution Cycles and the Pipeline Cycles. The beginning of an instruction is defined as the first internal clock following an instruction Sync.



Figure 28. One-Byte Instruction Formats

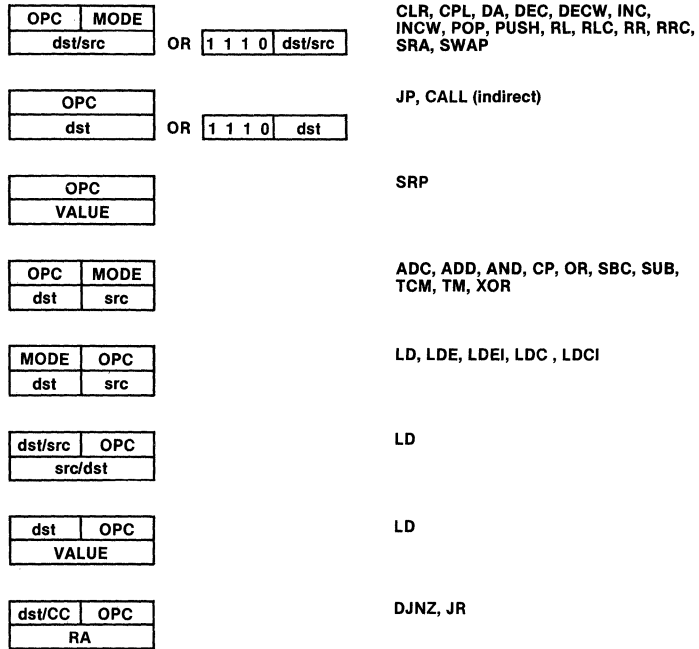


Figure 29. Two-Byte Instruction Formats

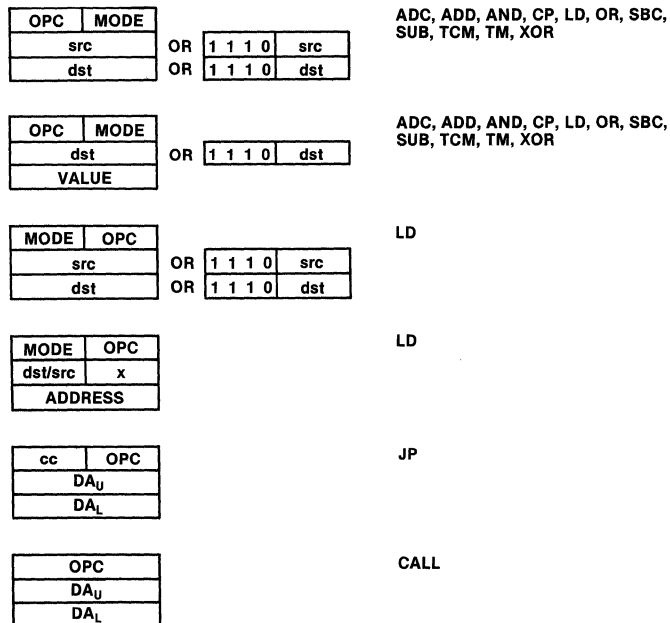


Figure 30. Three-Byte Instruction Formats

Instruction and Operation	Addr Modes		Hex Opcode	Execution		Pipeline Cycles	Flags Affected'					
	dst	src		Bytes	Cycles		C	Z	S	V	D	H
ADC dst,src	r	r	12	2	6	5	*	*	*	*	0	*
dst<-dst+src+C	r	Ir	13	2	6							
	R	R	14	3	10							
	R	IR	15	3	10							
	R	IM	16	3	10							
	IR	IM	17	3	10							
ADD dst,src	r	r	02	2	6	5	*	*	*	*	0	*
dst<-dst+src	r	Ir	03	2	6							
	R	R	04	3	10							
	R	IR	05	3	10							
	R	IM	06	3	10							
	IR	IM	07	3	10							
AND dst,src	r	r	52	2	6	5	-	*	*	0	-	-
dst<-dst AND												
src	r	Ir	53	2	6							
	R	R	54	3	10							
	R	IR	55	3	10							
	R	IM	56	3	10							
	IR	IM	57	3	10							
CALL dst	DA		D6	3	20	0	-	-	-	-	-	-
SP<-SP-2	IRR		D4	2	20	0						
@SP<-PC												
PC<-dst												
CCF			EF	1	6	5	*	-	-	-	-	-
C<-NOT C												
CLR dst	R		B0	2	6	5	-	-	-	-	-	-
dst<-0	IR		B1	2	6							
COM dst	R		60	2	6	5	-	*	*	0	-	-
dst<-NOT dst	IR		61	2	6							
CP dst,src	r	r	A2	2	6	5	*	*	*	*	-	-
dst-src	r	IR	A3	2	6							
	R	R	A4	3	10							
	R	IR	A5	3	10							
	R	IM	A6	3	10							
	IR	IM	A7	3	10							
DA dst	R		40	2	8	5	*	*	*	X	-	-
dst<-DA dst	IR		41	2	8							

Table 4-2. Instruction Summary

Instruction and Operation	Addr Modes		Hex Opcode	Execution		Pipeline Cycles	Flags Affected						
	dst	src		Bytes	Cycles		C	Z	S	V	D	H	
DEC dst dst<-dst-1	R IR		00 01	2 2	6 6	5	-	*	*	*	*	-	-
DECW dst dst<-dst-1	RR IR		80 81	2 2	10 10	5	-	*	*	*	*	-	-
DI IMR(7)<-0			8F	1	6	1	-	-	-	-	-	-	-
DJNZ r,dst r<-r-1 if r≠0 PC<-PC+dst Range: +127,-128	RA		rA r=0-F	2	12/10 (taken/ not taken)	3/5	-	-	-	-	-	-	-
EI IMR(7)<-1			9F	1	6	1	-	-	-	-	-	-	-
INC dst dst<-dst+1	r R IR		rE r=0-F 20 21	1 2 2	6 6 6	5	-	*	*	*	*	-	-
INCW dst dst<-dst+1	RR IR		A0 A1	2 2	10 10	5	-	*	*	*	*	-	-
IRET FLAGS<-@SP SP<-SP+1 PC<-@SP SP<-SP+2 IMR(7)<-1			BF	1	16	0	*	*	*	*	*	*	*
JP cc,dst if cc is true, PC<-dst	DA IRR		cD c=0-F 30	3 2	12/10 (taken/ not taken) 8	0	-	-	-	-	-	-	-
JR cc,dst if cc is true, PC<-PC+dst Range: +127,-128	RA		cB c=0-F	2	12/10 (taken/ not taken)	3	-	-	-	-	-	-	-
LD dst,src dst<-src	r r	IM R	rC r8	2 2	6 6	5	-	-	-	-	-	-	-

Table 4-2. Instruction Summary

Instruction and Operation	Addr dst	Modes src	Hex Opcode	Execution		Pipeline Cycles	Flags Affected								
				Bytes	Cycles		C	Z	S	V	D	H			
	R	r	r9 r=0-F	2	6										
	r	X	C7	3	10										
	X	r	D7	3	10										
	r	Ir	E3	2	6										
	Ir	r	F3	2	6										
	R	R	E4	3	10										
	R	IR	E5	3	10										
	R	IM	E6	3	10										
	IR	IM	E7	3	10										
	IR	R	F5	3	10										

LDC dst,src	r	Irr	C2	2	12	0	-	-	-	-	-	-	-	-	-
dst<-src	Irr	r	D2	2	12										

LDCI dst,src	Ir	Irr	C3	2	18	0	-	-	-	-	-	-	-	-	-
dst<-src	Irr	Ir	D3	2	18										
r<-r+1															
rr<-rr+1															

LDE dst,src	r	Irr	82	2	12	0	-	-	-	-	-	-	-	-	-
dst<-src	Irr	r	92	2	12										

LDEI dst,src	Ir	Irr	83	2	18	0	-	-	-	-	-	-	-	-	-
dst<-src	Irr	Ir	93	2	18										
r<-r+1															
rr<-rr+1															

NOP			FF	1	6	0	-	-	-	-	-	-	-	-	-

OR dst,src	r	r	42	2	6	5	-	*	*	0	-	-	-	-	-
dst<-dst OR src	r	Ir	43	2	6										
	R	R	44	3	10										
	R	IR	45	3	10										
	R	IM	46	3	10										
	IR	IM	47	3	10										

POP dst	R		50	2	10	5	-	-	-	-	-	-	-	-	-
dst<-@SP	IR		51	2	10										
SP<-SP+1															

PUSH src		R	70	2	10/12	1	-	-	-	-	-	-	-	-	-
SP<-SP-1					(int/ext stack)										
@SP<-src		IR	71	2	12/14										
					(int/ext stack)										

Table 4-2. Instruction Summary

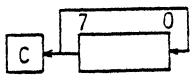
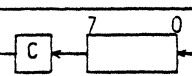
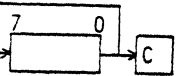
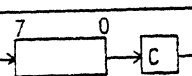
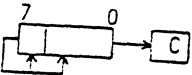
Instruction and Operation	Addr Modes		Hex Opcode	Execution Pipeline		Flags Affected						
	dst	src		Bytes	Cycles	Cycles	C	Z	S	V	D	H
RCF C←0			CF	1	6	5	0	-	-	-	-	-
RET PC←-@SP SP←-SP+2			AF	1	14	0	-	-	-	-	-	-
RL dst	R		90	2	6	5	*	*	*	*	-	-
	IR		91	2	6							
												
RLC dst	R		10	2	6	5	*	*	*	*	-	-
	IR		11	2	6							
												
RR dst	R		E0	2	6	5	*	*	*	*	-	-
	IR		E1	2	6							
												
RRC dst	R		C0	2	6	5	*	*	*	*	-	-
	IR		C1	2	6							
												
SBC dst,src	r	r	32	2	6	5	*	*	*	*	1	*
dst←-dst-src-C	r	Ir	33	2	6							
	R	R	34	3	10							
	R	IR	35	3	10							
	R	IM	36	3	10							
	IR	IM	37	3	10							
SCF C←1			DF	1	6	5	1	-	-	-	-	-
SRA dst	R		D0	2	6	5	*	*	*	0	-	-
	IR		D1	2	6							
												

Table 4-2. Instruction Summary

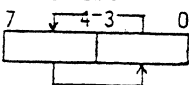
Instruction and Operation	Addr dst	Modes src	Hex Opcode	Bytes	Execution Cycles	Pipeline Cycles	Flags Affected C Z S V D H
SRP src RP←src		IM	31	2	6	1	- - - - -
SUB dst,src dst←dst-src	r r	r Ir	22 23	2 2	6 6	5	* * * * 1 *
	R	R	24	3	10		
	R	IR	25	3	10		
	R	IM	26	3	10		
	IR	IM	27	3	10		
SWAP dst	R		F0	2	8	5	X * * X - -
	IR		F1	2	8		
TCM dst,src (NOT dst)AND src	r r	r Ir	62 63	2 2	6 6	5	- * * 0 - -
	R	R	64	3	10		
	R	IR	65	3	10		
	R	IM	66	3	10		
	IR	IM	67	3	10		
TM dst,src dst AND src	r r	r Ir	72 73	2 2	6 6	5	- * * 0 - -
	R	R	74	3	10		
	R	IR	75	3	10		
	R	IM	76	3	10		
	IR	IM	77	3	10		
XOR dst,src dst←dst XOR src	r r	r Ir	B2 B3	2 2	6 6	5	- * * 0 - -
	R	R	B4	3	10		
	R	IR	B5	3	10		
	R	IM	B6	3	10		
	IR	IM	B7	3	10		

Table 4-2. Instruction Summary

CHAPTER 5 - Z8 DEVELOPMENT SUPPORT

Zilog offers the following for development of Z8-based systems:

- o PLZ/ASM Assembler
- o Z8 Simulator
- o Z8/64 Development Device
- o ZPB-8 Prototyping Board

5.1 PLZ/ASM ASSEMBLER

The PLZ/ASM assembler generates relocatable and absolute object code. High-level control and data structures in the language enable the user to balance structured programming practices with machine-dependent operations. Special features of PLZ/ASM includes IF...THEN...ELSE conditional statements; CASE statements to execute one of several groups of instructions depending on the value in a selector register; a DO...OD looping construct; and procedures. Data structures include bytes, words, arrays and records. In addition, data and instructions can be mapped into any of the three Z8 address spaces: register, data and program.

The Z8 assembler runs on an MCZ or ZDS system containing the operating system RIO and stream I/O packages. For more detailed information on the PLZ/ASM assembler, consult the Z8 PLZ/ASM Assembly Language Programming Manual (03-3030-01) and the Z8 Assembler User's Guide (03-3048-01).

5.2 Z8 SIMULATOR

Z8SIM allows Z8 program development in a software environment. Except for time-dependent code, the program can be completely debugged using the simulator before the code is masked programmed into the Z8 microcomputer. Z8SIM allows systematic testing of Z8 machine code as well as detection of the more difficult types of program bugs. Z8SIM features include:

- o A DO-file and macro-command facility that allows fast setup of test situations. A logging facility records the test session on diskette file.
- o Every byte of simulated Z8 code can be set for a breakpoint.
- o Abnormal conditions, such as references to undefined memory, cause breakpoints.
- o Code execution can be memory mapped with a reference/change history that maintains a map of all memory locations referenced or changed.

Z8SIM runs on a Zilog 64K MCZ or ZDS development system with the RIO operating system. Refer to the Z8 Simulator User Manual (03-3046-01) for detailed information about the features and operation of this program.

Both the assembler and simulator are available in a Z8 Software development package that includes a diskette with software and the following documentation: A Z8 Technical Manual, a Z8 PLZ/ASM Assembly Language Programming Manual, a Z8 Assembler User's Guide, a Z8 Simulator Manual, sample Z8 programs, and sample simulator sessions.

5.3 ZPB-8 PROTOTYPING BOARD

The ZPB-8 allows the user to build a system prototype designed for a 40-pin Z8 (Z8/40) where the Z8/40 is replaced by the ZPB-8 during the debug phase. When the program is finalized, the user runs the prototype system with a mask-programmed Z8/40.

The ZPB-8 is an inexpensive circuit board that contains a 64-pin Z8 development device (Z8/64), a 2716 EPROM socket, crystal oscillator parts and a flat cable assembly terminated in a 40-pin connector that connects the ZPB-8 to the user system.

5.4 Z8/64 DEVELOPMENT DEVICE

The 64-pin development version of the 40-pin mask-programmed Z8 allows the user to prototype the system in hardware with an actual Z8 device, and develop the code that is eventually mask-programmed into the on-chip ROM of the Z8/40.

The Z8/64 is identical to the Z8/40 with the following exceptions:

- o The internal ROM has been removed.
- o The ROM address lines and data lines are buffered and brought out to external pins.
- o Control lines for the new memory have been added.

5.4.1 Z8/64 Pin Description

The functions of the Z8/64 I/O lines, \overline{AS} , \overline{DS} , R/\overline{W} , XTAL1, XTAL2 and \overline{RESET} are identical to those of their Z8/40 counterparts. The functions of the remaining 24 pins is as follows:

A00-A11. *Program Memory Address* (outputs). A00-A11 access the first 2K bytes of program memory. All is a reserved pin.

D0-D7. *Program Data (inputs)*. Program data from the first 2K bytes of program memory is input through pins D0-D7.

$\overline{\text{MDS}}$. *Program Memory Data Strobe (output, active Low)*. $\overline{\text{MDS}}$ is Low during an instruction fetch cycle when the first 2K bytes of program memory are being accessed.

$\overline{\text{SYNC}}$. *Instruction Sync (output, active Low)*. $\overline{\text{SYNC}}$ is a strobe output that is forced Low during the clock period preceding the beginning of an opcode fetch.

SCLK. *System Clock (output)*. SCLK is the internal clock output through a buffer. The clock rate is equal to one-half the crystal frequency.

IACK. *Interrupt Acknowledge (output, active High)*. IACK is driven High in response to an interrupt during the interrupt machine cycle.

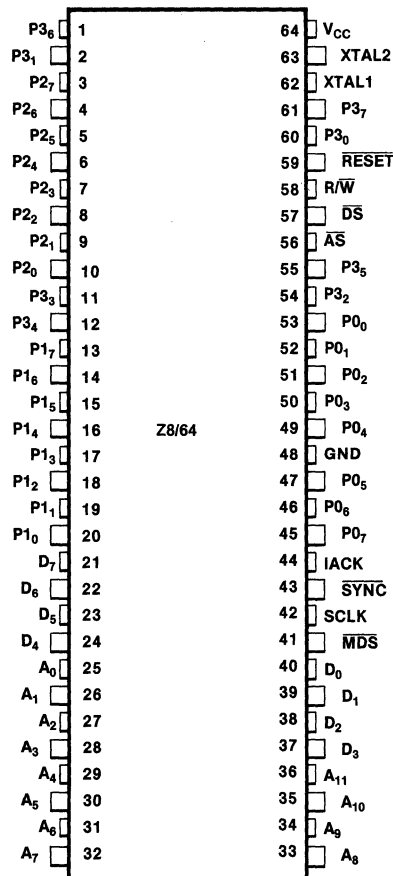


Figure 5-1. Z8/64 Pin Configuration

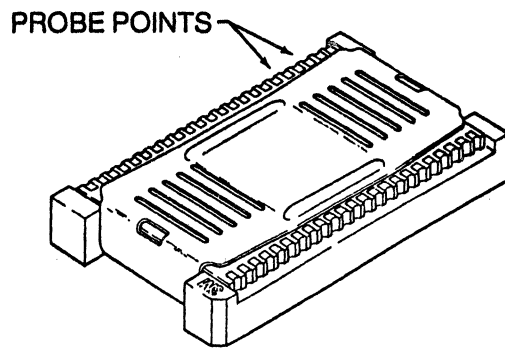
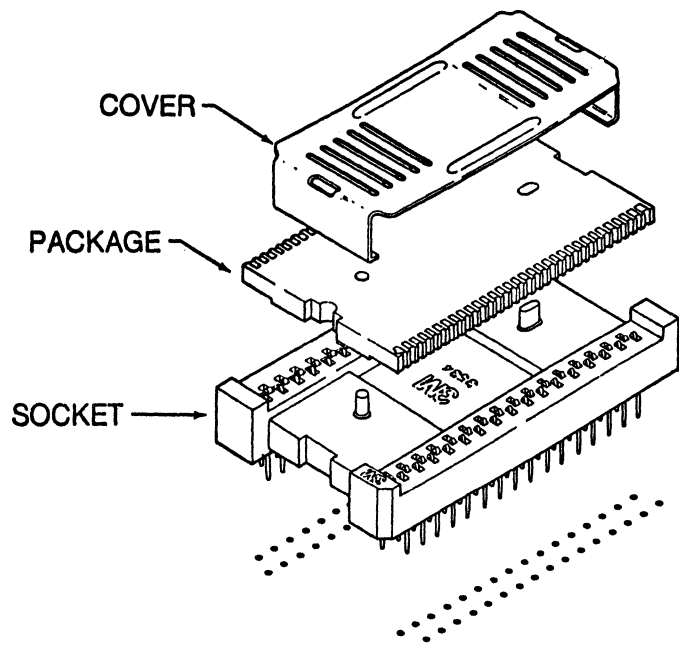


Figure 5-2. Z8/64 Package

Chapter 6

Electrical Parameters

6.1 Absolute Maximum Ratings

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

6.2 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: $+4.75V \leq V_{CC} \leq +5.25V$, $GND = 0V$, $0^\circ C \leq T_A \leq +70^\circ C$.

6.3 DC Char- acteristics

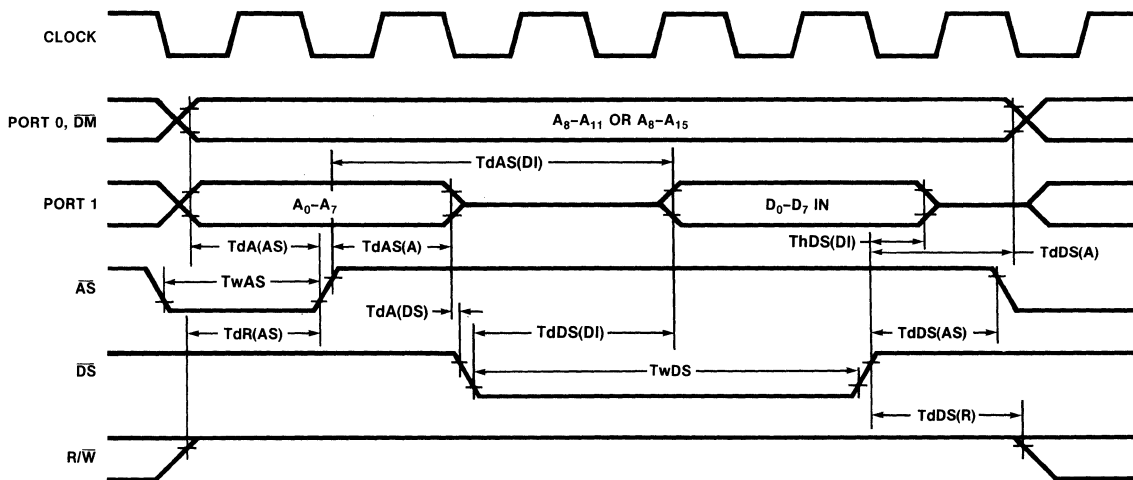
Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage			V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage			V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage			V		
V_{RL}	Reset Input Low Voltage			V		
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250 \mu A$	1
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = +2.0 \text{ mA}$	1
I_{IL}	Input Leakage			μA	$0 \leq V_{IN} \leq +5.25V$	
I_{OL}	Output Leakage			μA	$0 \leq V_{IN} \leq +5.25V$	
I_{IR}	Reset Input Current			μA	$V_{RL} = 0V$, $V_{CC} = +5.25V$	
I_{DD}	V_{DD} Supply Current			mA		
I_{MM}	V_{MM} Supply Current			mA		

1. For A_0 - A_{11} , \overline{MDS} , \overline{SYNC} , SCLK and IACK on the Z8/64 pin version, $I_{OH} = -100 \mu A$ and $I_{OL} = 1.0 \text{ mA}$.

6.4 External Instruction Fetch, I/O or Memory Read Timing

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Float Delay Time	60		ns	Test Load 1	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 1	3
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdA(DS)	Address Float to Data Strobe Delay Time	0		ns	Test Load 1	
TwDS	Data Strobe Width	230		ns	Test Load 1	2
TdDS(DI)	Data Strobe to Data In Valid Delay Time		160	ns	Test Load 1	3
ThDS(DI)	Data In Hold Time	0		ns		
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		ns	Test Load 1	1
TdR(AS)	Read Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(R)	Data Strobe to Read Change Delay	60		ns	Test Load 1	1

1. 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.
2. 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. Refer to Figures 1-9 and 1-10.
3. 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).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



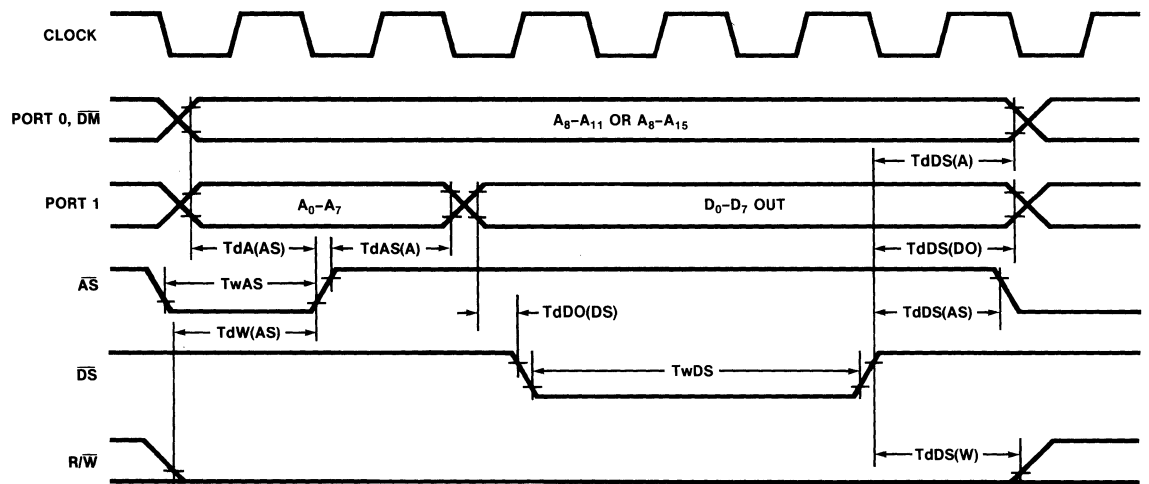
**6.5
External I/O
or Memory
Write Timing**

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdDO(DS)	Data Out Valid to Data Strobe Delay Time	30		ns	Test Load 1	1
TwDS	Data Strobe Width	150		ns	Test Load 1	2
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(DO)	Data Strobe to Data Out Change Delay Time	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		ns	Test Load 1	1
TdW(AS)	Write Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(W)	Data Strobe to Write Change Delay Time	60		ns	Test Load 1	1

1. 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.
2. 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. Refer to Figures 1-9 and 1-10.

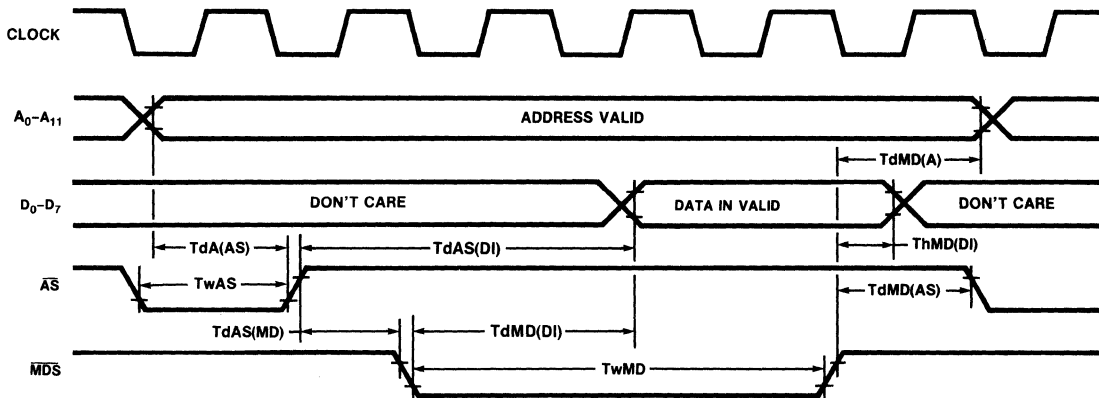
3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



**6.6
Memory Port
(Z8/64)
Timing**

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 2	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 2	3
TwAS	Address Strobe Width	60		ns	Test Load 2	1
TdAS(MD)	Address Strobe to Memory Data Strobe Delay Time	60		ns	Test Load 2	1
TwMD	Memory Data Strobe Width	230		ns	Test Load 2	2
TdMD(DI)	Memory Data Strobe to Data In Valid Delay Time		160	ns	Test Load 2	1
ThMD(DI)	Data In Hold Time	0		ns		
TdMD(A)	Memory Data Strobe to Address Change Delay Time	60		ns	Test Load 2	1
TdMD(AS)	Memory Data Strobe to Address Strobe Delay Time	50		ns	Test Load 2	1

1. 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.
2. Memory 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 Memory Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-9 and 1-10.
3. Address Strobe and Memory Data Strobe to Data In Valid delay times represent memory system access times and are given at 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 TdMS(DI).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



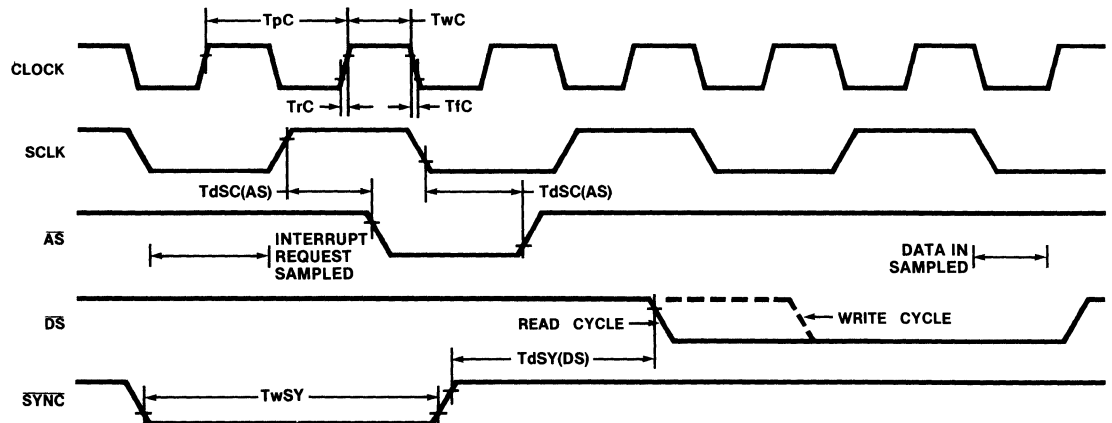
**6.7
Additional
Timing**

Symbol	Parameter	Min	Max	Unit	Condition	Notes
T_{pC}	Input Clock Period	125		ns		
T_{rC}, T_{fC}	Input Clock Rise and Fall Times			ns	From External Clock Generator	
T_{wC}	Input Clock Width			ns	From External Clock Generator	
$T_{dSC}(AS)$	System Clock Out to Address Strobe Delay Time			ns		1
$T_{dSY}(DS)$	Instruction Sync Out to Data Strobe Delay Time			ns		1, 2
T_{wSY}	Instruction Sync Out Width			ns		1, 2

1. Test Conditions use Test Load 1 for SCLK and \overline{SYNC} when output through their respective Port 3 pins and Test Load 2 on the SCLK and \overline{SYNC} direct outputs on the 64 pin version.
2. Times given assume an 8 MHz crystal input frequency.

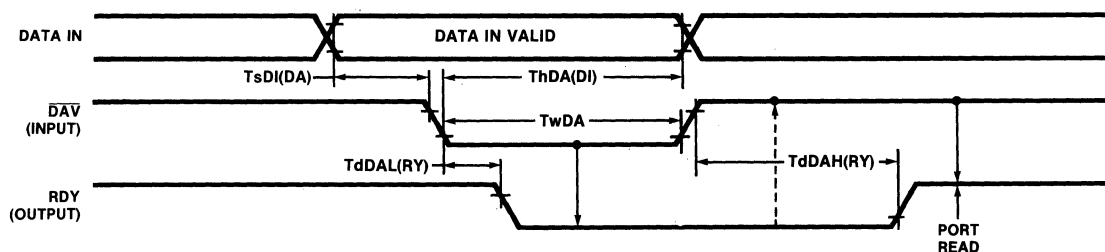
For lower frequencies, the change in two clock periods must be added.

3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."

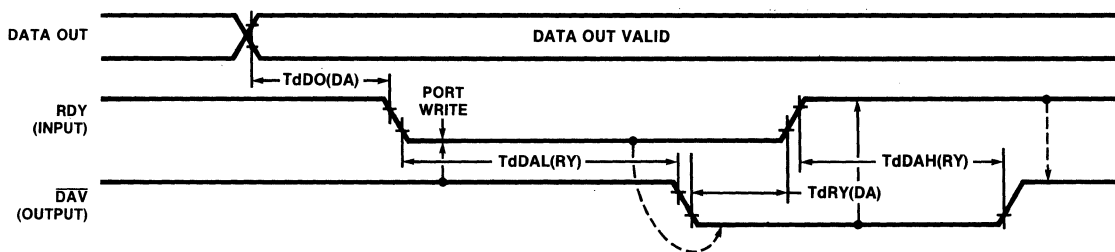


6.8 Handshake Timing

Symbol	Parameter	Min	Max	Unit	Condition
$T_{sDI}(DA)$	Data In Setup Time	0		ns	
$T_{hDA}(DI)$	Data In Hold Time	190		ns	
T_{wDA}	Data Available Width			ns	Input Handshake Test Load 1
$T_{dDAL}(RY)$	Data Available Low to Ready Delay Time			ns	Input Handshake Test Load 1
		0		ns	Output Handshake Test Load 1
$T_{dDAH}(RY)$	Data Available High to Ready Delay Time			ns	Input Handshake Test Load 1
		0		ns	Output Handshake Test Load 1
$T_{dDO}(DA)$	Data Out to Data Available Delay Time			ns	Test Load 1
$T_{dRY}(DA)$	Ready to Data Available Delay Time			ns	Test Load 1

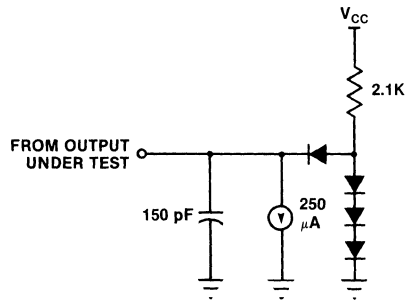


Input Handshake

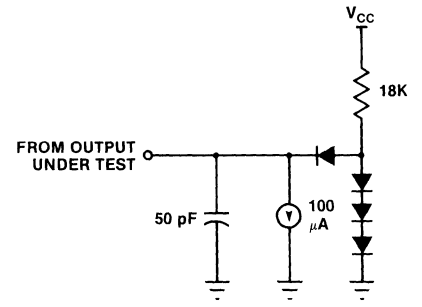


Output Handshake

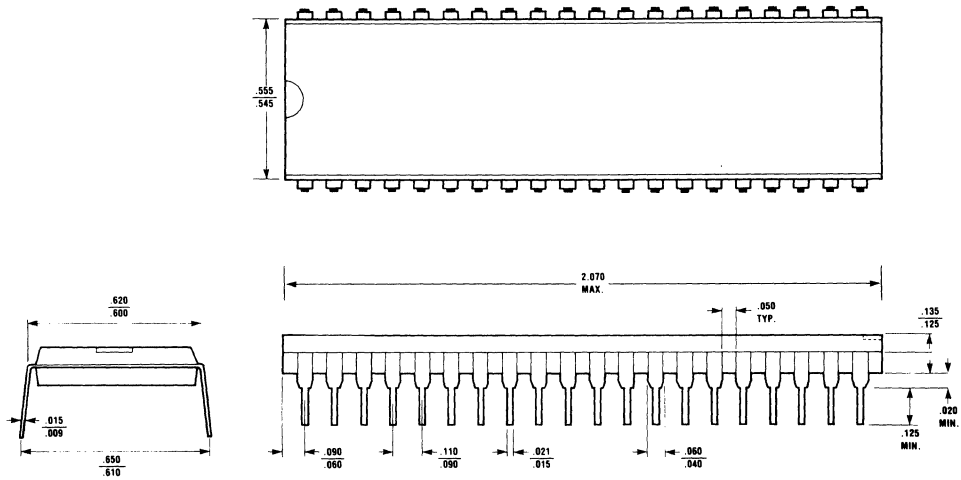
6.9 Test Load Circuits



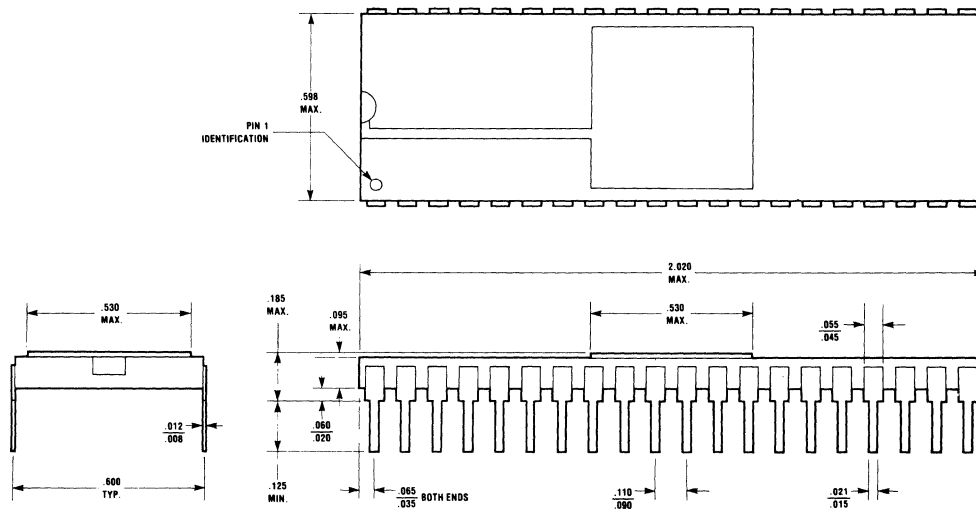
Test Load 1



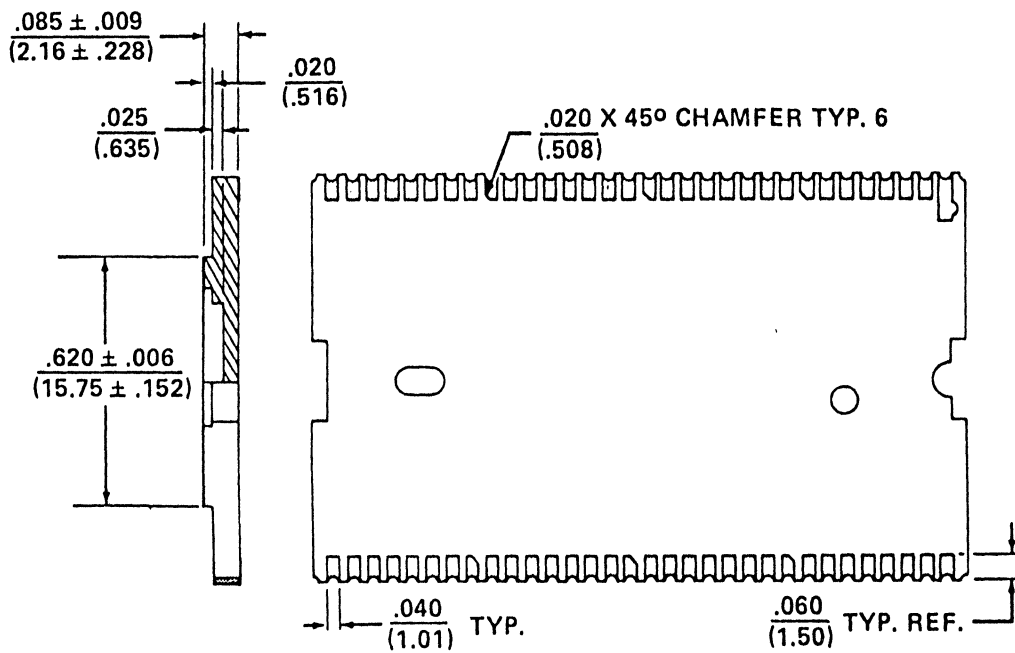
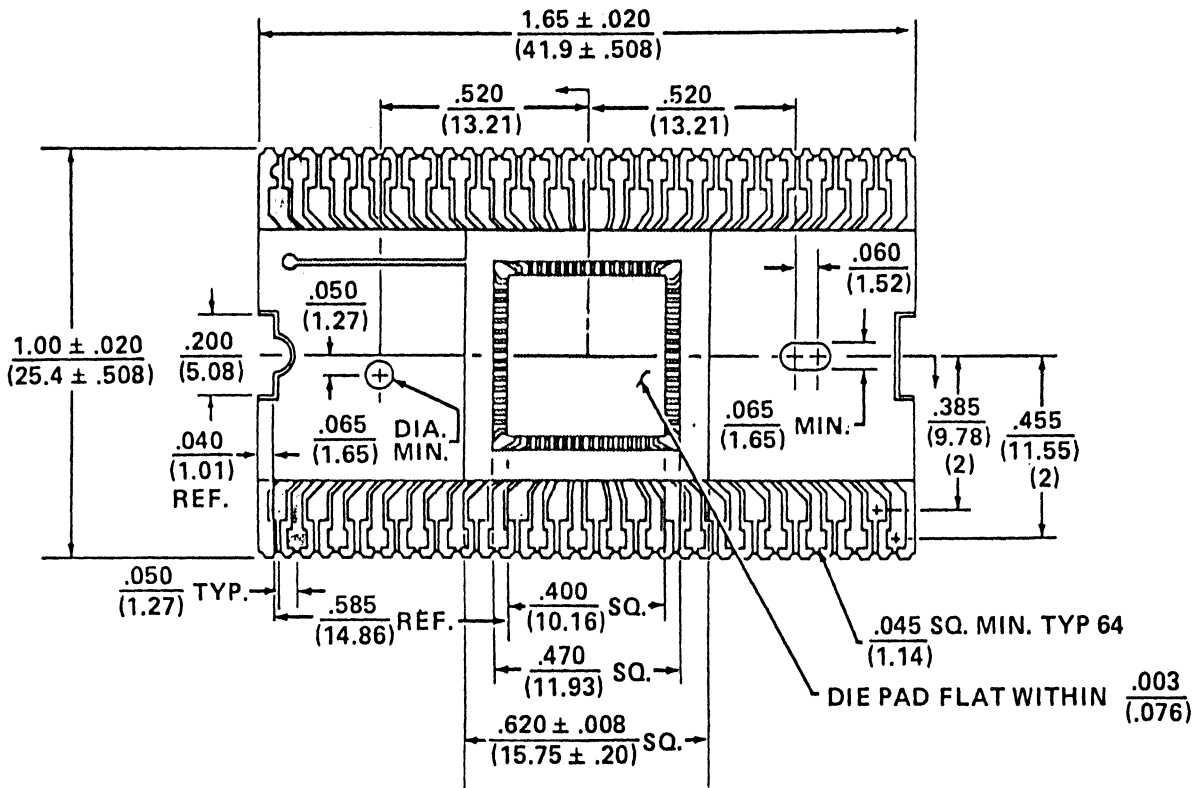
Test Load 2



40-Pin Plastic Package Dimensions

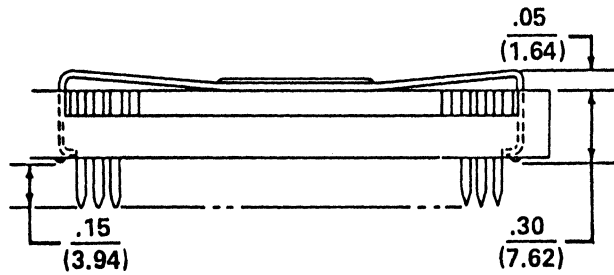
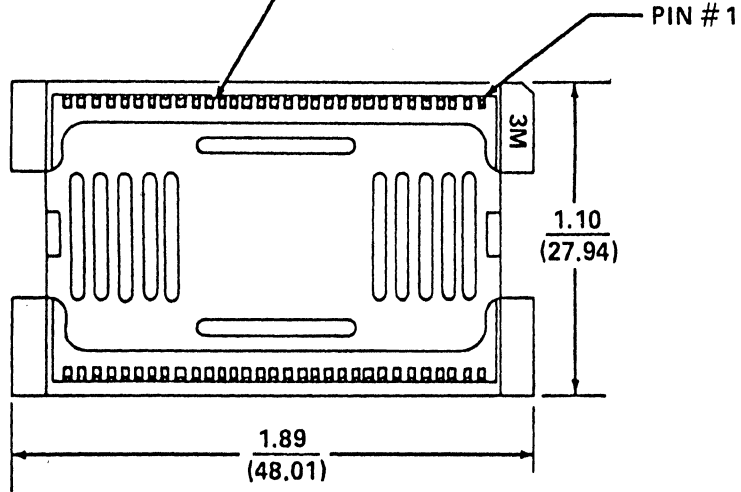


40-Pin Ceramic Package Dimensions

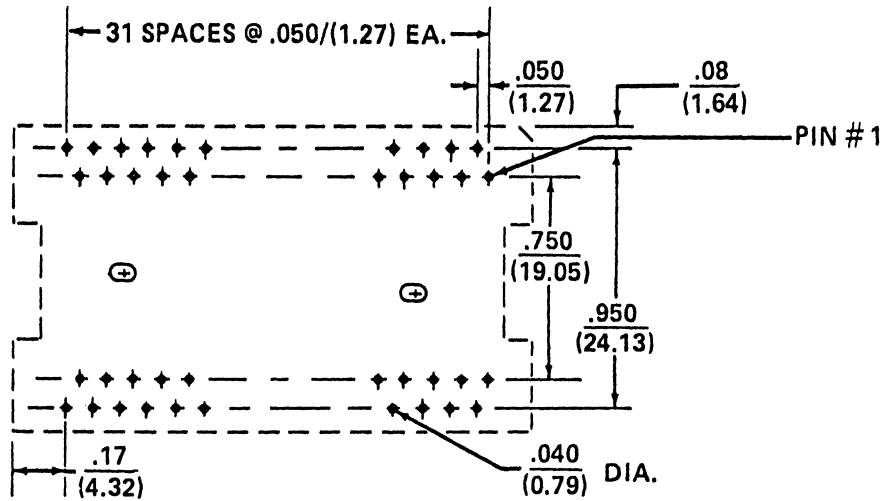


64-Pin Carrier Dimensions

LEADLESS QUAD IN-LINE
CERAMIC PACKAGE



CONNECTOR DETAIL



HOLE PATTERN DETAIL

64-Pin Socket Dimensions

READER'S COMMENTS

Your feedback about this document is important to us: only in this way can we 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 the future users of Zilog systems. Thank you.

Your Name: _____

Company Name: _____

Address: _____

Title of this document: _____

What software products do you have? _____

What is your hardware configuration (including memory size)? _____

Does this publication meet your needs? Yes No

If not, why not? _____

How do you use this publication? (Check all that apply)

As an introduction to the subject?

As a reference manual?

As an instructor or student?

How do you find the material?

	Excellent	Good	Poor
Technicality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

What would have improved the material? _____

Other comments, suggestions or corrections: _____

If you found any mistakes in this document, please let us know what and where they were:



First Class

Permit No. 475
Cupertino
California
95014

Business Reply Mail

No Postage Necessary if Mailed in the United States

Postage Will Be Paid By



Zilog
Semiconductor Division
10341 Bubb Road
Cupertino, California 95014

Attention: Publications Department



