Z8 MICROCONTROLLER
USER'S MANUAL

UM001600-Z8X0599

ZiLOG

Totally Logical

TABLE OF CONTENTS

Chapter Title and Subsections — Page

Chapter Title and Subsections **Page**

Chapter 3. Clock (Continued)

Chapter 4. Reset—Watch-Dog Timer

Chapter 5. I/O Ports

Chapter 5. I/O Ports

Chapter Title and Subsection **Page**

Chapter 7. Interrupts (Continued)

Chapter Title and Subsections **Page**

Chapter 8. Power-Down Modes

Chapter 9. Serial I/O

Chapter 10. External Interface

Chapter Title and Subsection **Page**

Chapter 11. Addressing Modes

Chapter 12. Instruction Set

Totally Logical

LIST OF FIGURES

Figure Title Page

Figure Title Page

Chapter 4. Reset—Watch-Dog Timer

Chapter 5. I/O Ports

Chapter 5. I/O Ports (Continued)

Figure Title Page

Chapter 6. Counter/Timers

Chapter 6. Counter/Timers (Continued)

Figure Title Page

Chapter 7. Interrupts

Chapter 8. Power-Down Modes

Chapter 9. Serial I/O

Figure Title Page

Chapter 10. External Interface

Chapter 11. Addressing Modes

Figure Title Page

Totally Logical

LIST OF TABLES

Table Title Page

Table Title Page

Chapter 8. Power-Down Modes

Chaper 9. Serial I/O

Totally Logical

CHAPTER 1
Z8 MCU PRODUCT OVERVIEW

1.1 Z8 MCU FAMILY OVERVIEW

The ZiLOG Z8 microcontroller (MCU) product line continues to expand with new product introductions. ZiLOG MCU products are targeted for cost-sensitive, high-volume applications including consumer, automotive, security, and HVAC. It includes ROM-based products geared for high-volume production (where software is stable) and one-time programmable (OTP) equivalents for prototyping as well as volume production where time to market or code flexibility is critical (Table 1-1). A variety of packaging options are available including plastic DIP, SOIC, PLCC, and QFP.

A generalized Z8 MCU® block diagram is shown in Figure 1-1. The same on-chip peripherals are used across the MCU product line with the primary differences being the amount of ROM/RAM, number of I/O lines present, and packaging/temperature ranges available. This allows code written for one MCU device to be easily ported to another family member.

1.1.1 Key Product Line Features

- **General-Purpose Register (GPR) File:** Every RAM register acts like an accumulator, speeding instruction execution and maximizing coding efficiency. Working register groups allow fast context switching.
- **Flexible I/O:** I/O byte, nibble, and/or bit programmable as inputs or outputs. Outputs are software programmable as open-drain or push-pull on a port basis. Inputs are Schmitt-triggered with auto latches to hold unused inputs at a known voltage state.
- **Analog Inputs:** Three input pins are software programmable as digital or analog inputs. When in the analog mode, two comparator inputs are provided with a common reference input. These inputs are ideal for a variety of common functions, including threshold level detection, analog-to-digital conversion, and short circuit detection. Each analog input provides a unique maskable interrupt input.
- **Timer/Counter(T/C):** The T/C consists of a programmable 6-bit prescaler and 8-bit downcounter, with maskable interrupt upon end-of-count. Software controls T/C load/start/stop, countdown read (at any time on the fly), and maskable end-of-count interrupt. Special functions available include T_{IN} (external counter input, external gate input, or external trigger input) and T_{OUT} (external access to timer output or the internal system clock.) These special functions allow accurate hardware input pulse measurement and output waveform generation.
- **Interrupts:** There are six vectored interrupt sources with software-programmable enable and priority for each of the six sources.
- **Watch-Dog Timer (WDT):** An internal WDT circuit is included as a fail-safe mechanism so that if software strays outside the bounds of normal operation, the WDT will timeout and reset the MCU. To maximize circuit robustness and reliability, the default WDT clock source is an internal RC circuit (isolated from the device clock source).
- **Auto Reset/Low-Voltage Protection:** All family devices have internal Power-On Reset. ROM devices add low-voltage protection. Low-voltage protection ensures the MCU is in a known state at all times (in active RUN mode or RESET) without external hardware (or a device reset pin).
- **Low-EMI Operation:** Mode is programmable via software or as a mask option. This new option provides for reduced radiated emission via clock and output drive circuit changes.

1.1 Z8 MCU FAMILY OVERVIEW (Continued)

- **Low-Power:** CMOS with two standby modes; STOP and HALT.
- **Full Z8 Instruction Set:** Forty-eight basic instructions, supported by six addressing modes with the ability to operate on bits, nibbles, bytes, and words.

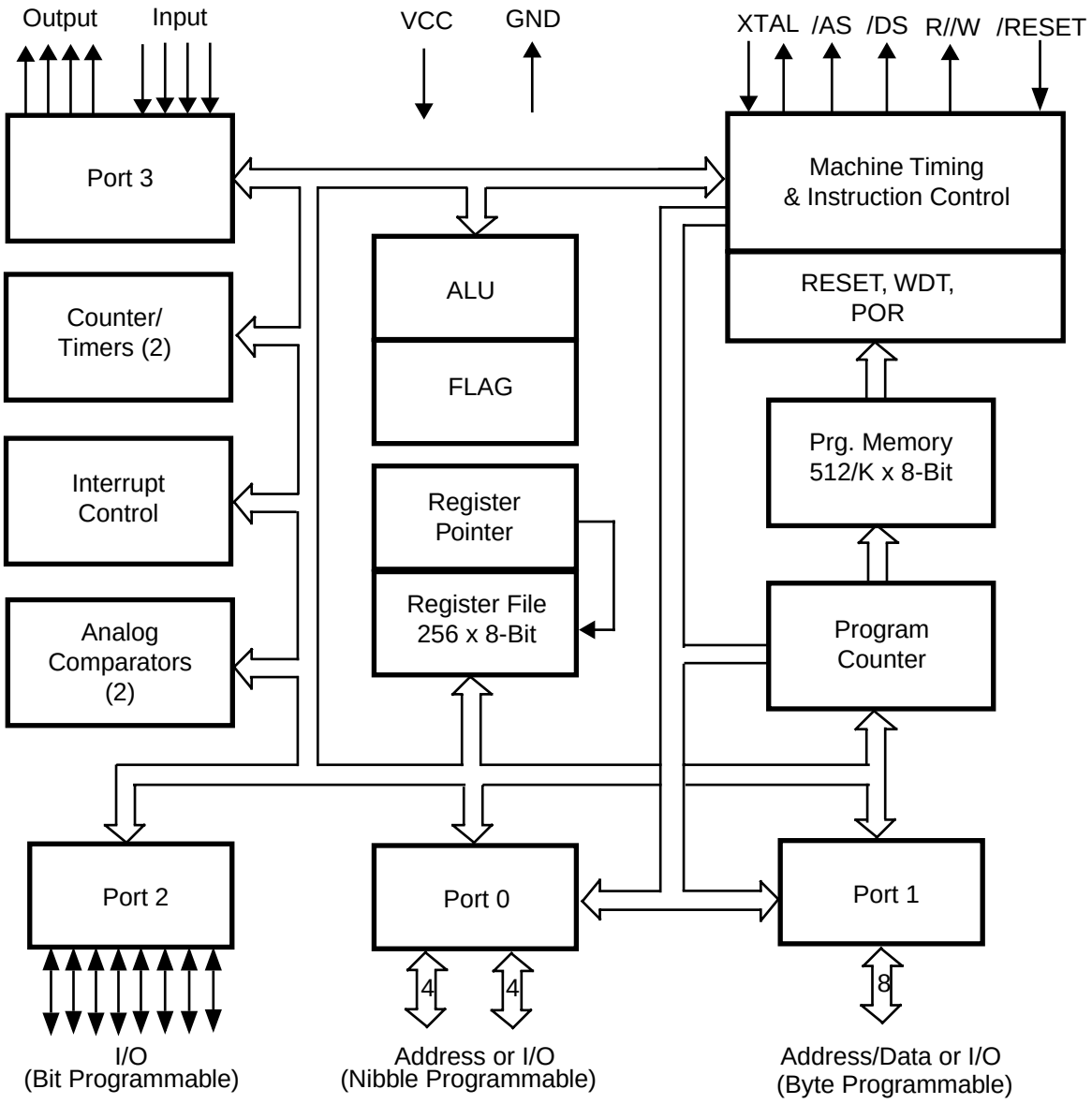


Figure 1-1. Z8 MCU Block Diagram

1.1.2 Product Development Support

The Z8 MCU product line is fully supported with a range of cross assemblers, C compilers, ICEBOX emulators, single and gang OTP/EPROM programmers, and software simulators.

The Z86CCP01ZEM low-cost Z8 CCP™ real-time emulator/programmer kit was designed specifically to support all the products outlined in Table 1-1.

Table 1-1. ZiLOG General-Purpose Microcontroller Product Family

PRODUCT	ROM/RAM	I/0	T/C	AN IN	INT	WDT	POR	V_{BO}	RC	SPEED (MHz)	PIN COUNT
Z86C03	512/60	14	1	2	6	F	Y	Y	Y	8	18
Z86E03	512/60	14	1	2	6	F	Y	N	Y	8	18
Z86C04	1K/124	14	2	2	6	F	Y	Y	Y	8	18
Z86E04	1K/124	14	2	2	6	F	Y	N	Y	8	18
Z86C06	1K/124	14	2	2	6	P	Y	Y	Y	12	18
Z86E06	1K/124	14	2	2	6	P	Y	N	Y	12	18
Z86C08	2K/124	14	2	2	6	F	Y	Y	Y	12	18
Z86E08	2K/124	14	2	2	6	F	Y	N	Y	12	18
Z86C30	4K/236	24	2	2	6	P	Y	Y	Y	12	28
Z86E30	4K/236	24	2	2	6	P	Y	N	Y	12	28
Z86C31	2K/124	24	2	2	6	P	Y	Y	Y	8	28
Z86E31	2K/124	24	2	2	6	P	Y	N	Y	8	28
Z86C40	4K/236	32	2	2	6	P	Y	Y	Y	16	40/44
Z86E40	4K/236	32	2	2	6	P	Y	N	Y	16	40/44

Note: Z86Cxx signify ROM devices; 86xx signify EPROM devices; F = fixed; P = programmable

The Z86CCP01ZEM kit comes with:

- Z8 CCP Evaluation Board
- Z8 CCP Power Cable
- ZiLOG Developer's Studio (ZDS) CD-ROM , Including Windows-Based[1] GUI Host Software
- 1999 ZiLOG Technical Library
- Z8 CCP User's Manual

A Z8 CCP Emulator Accessory Kit (Z8CCP00ZAC) is also available and provides an RS-232 cable and power cable along with the 28- and 40- pin ZIF sockets and 28- and 40-pin target connector cables required to emulate/program 28/40 pin devices.

1. Windows is a trademark of the Microsoft Corporation.

Totally Logical

CHAPTER 2
ADDRESS SPACE

2.1 INTRODUCTION

Four address spaces are available for the Z8 MCU®:

- The Z8 Standard Register File contains addresses for peripheral, control, all general-purpose, and all I/O port registers. This is the default register file specification.
- The Z8 Expanded Register File (ERF) contains addresses for control and data registers for additional peripherals/features.
- Z8 External Program Memory contains addresses for all memory locations having executable code and/or data.
- Z8 External Data Memory contains addresses for all memory locations that hold data only, whether internal or external.

2.2 Z8 MCU STANDARD REGISTER FILE

The Z8 Standard Register File totals up to 256 consecutive bytes (Registers). The register file consists of 4 I/O ports (00H-03H), 236 General-Purpose Registers (04H-EFH), and 16 control registers (F0H-FFH). Table 2-1 shows the layout of the register file, including register names, locations, and identifiers.

Table 2-1. Z8 Standard Register File

Hex Address	Register Description	Register Identifier
FF	Stack Pointer Low Byte	SPL
FE	Stack Pointer High Byte	SPH
FD	Register Pointer	RP
FC	Program Control Flags	FLAGS
FB	Interrupt Mask Register	IMR
FA	Interrupt Request Register	IRQ
F9	Interrupt Priority Register	IPR
F8	Port 0-1 Mode Register	P01M
F7	Port 3 Mode Register	P3M
F6	Port 2 Mode Register	P2M
F5	T0 Prescaler	PRE0
F4	Timer/Counter 0	T0
F3	T1 Prescaler	PRE1
F2	Timer/Counter 1	T1
F1	Timer Mode	TMR

Table 2-1. Z8 Standard Register File

Hex Address	Register Description	Register Identifier
F0	Serial I/O	SIO
EF		R239
.	General-Purpose	.
.	Registers (GPR)	.
.		.
04		R4
03	Port 3	P3
02	Port 2	P2
01	Port 1	P1
00	Port 0	P0

2.2 Z8 MCU STANDARD REGISTER FILE (Continued)

Registers can be accessed as either 8-bit or 16-bit registers using Direct, Indirect, or Indexed Addressing. All 236 general-purpose registers can be referenced or modified by any instruction that accesses an 8-bit register, without the need for special instructions. Registers accessed as 16 bits are treated as even-odd register pairs (there are 118 valid pairs). In this case, the data's Most Significant Byte (MSB) is stored in the even numbered register, while the Least Significant Byte (LSB) goes into the next higher odd numbered register (Figure 2-1).

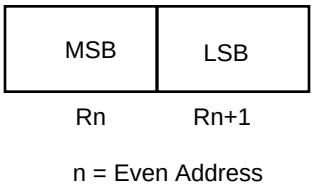


Figure 2-1. 16-Bit Register Addressing

By using a logical instruction and a mask, individual bits within registers can be accessed for bit set, bit clear, bit complement, or bit test operations. For example, the instruction AND R15, MASK performs a bit clear operation. Figure 2-2 shows this example.

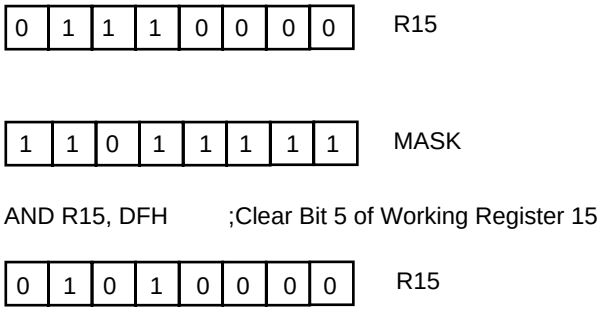


Figure 2-2. Accessing Individual Bits (Example)

When instructions are executed, registers are read when defined as sources and written when defined as destinations. All General-Purpose Registers function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

2.2.1 General-Purpose Registers

General-Purpose Registers (GPR) are undefined after the device is powered up. The registers keep their last value after any reset, as long as the reset occurs in the V_{CC} voltage-specified operating range. It will not keep its last state from a V_{LV} reset if V_{CC} drops below 1.8v.

Note: Registers in Bank E0-EF may only be accessed through the working register and indirect addressing modes. Direct access cannot be used because the 4-bit working register address mode already uses the format [E | dst], where dst represents the working register number from 0H to FH.

2.2.2 RAM Protect

The upper portion of the register file address space 80H to EFH (excluding the control registers) may be protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, the user activates this feature from the internal ROM code to turn off/on the RAM Protect by loading either a 0 or 1 into the IMR register, bit D6. A 1 in D6 enables RAM Protect. Only devices that use registers 80H to EFH offer this feature.

2.2.3 Working Register Groups

Z8 instructions can access 8-bit registers and register pairs (16-bit words) using either 4-bit or 8-bit address fields. 8-bit address fields refer to the actual address of the register. For example, Register 58H is accessed by calling upon its 8-bit binary equivalent, 01011000 (58H).

With 4-bit addressing, the register file is logically divided into 16 Working Register Groups of 16 registers each, as shown in Table 2-2. These 16 registers are known as Working Registers. A Register Pointer (one of the control registers, FDH) contains the base address of the active Working Register Group. The high nibble of the Register Pointer determines the current Working Register Group.

When accessing one of the Working Registers, the 4-bit address of the Working Register is combined within the upper four bits (high nibble) of the Register Pointer, thus forming the 8-bit actual address. Figure 2-3 illustrates this operation. Since working registers are typically specified by short format instructions, there are fewer bytes of code needed, which reduces execution time. In addition, when processing interrupts or changing tasks, the Register Pointer speeds context switching. A special Set Register Pointer (SRP) instruction sets the contents of the Register Pointer.

Table 2-2. Working Register Groups

Register Pointer (FDH) High Nibble	Working Register Group (HEX)	Actual Registers (HEX)
1111(B)	F	F0–FF
1110(B)	E	E0–EF
1101(B)	D	D0–DF
1100(B)	C	C0–CF
1011(B)	B	B0–BF
1010(B)	A	A0–AF
1001(B)	9	90–9F
1000(B)	8	80–8F
0111(B)	7	70–7F
0110(B)	6	60–6F
0101(B)	5	50–5F
0100(B)	4	40–4F
0011(B)	3	30–3F
0010(B)	2	20–2F
0001(B)	1	10–1F
0000(B)	0	00–0F

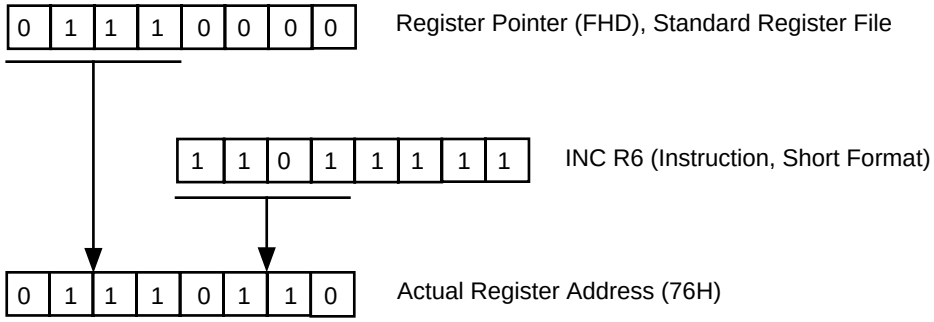


Figure 2-3. Working Register Addressing Examples

2.2 Z8 MCU STANDARD REGISTER FILE (Continued)

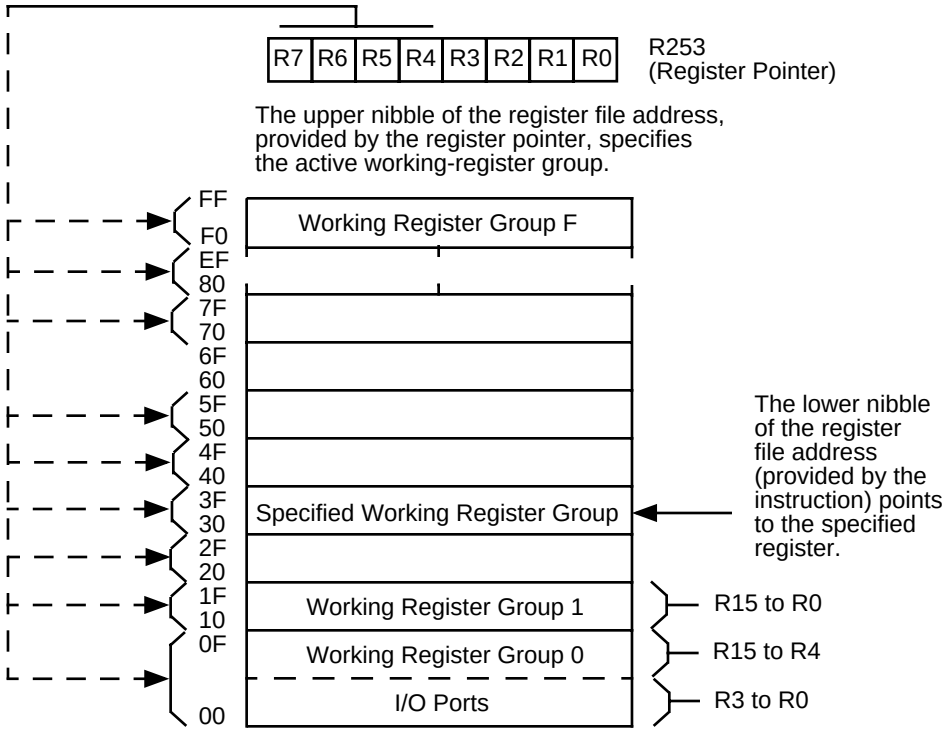


Figure 2-4. Register Pointer

Note: The full register file is shown. Please refer to the selected device product specification for actual file size.

2.2.4 Error Conditions

Registers in the Z8 Standard Register File must be correctly used because certain conditions produce inconsistent results and should be avoided.

- Registers F3H and F5H-F9H are write-only registers. If an attempt is made to read these registers, FFH is returned. Reading any write-only register will return FFH.
- When register FDH (Register Pointer) is read, the least significant four bits (lower nibble) will indicate the current Expanded Register File Bank. (Example: 0000 indicates the Standard Register File, while 1010 indicates Expanded Register File Bank A.)
- When Ports 0 and 1 are defined as address outputs, registers 00H and 01H will return 1s in each address bit location when read.
- Writing to bits that are defined as timer output, serial output, or handshake output will have no effect.
- The Z8 instruction DJNZ uses any general-purpose working register as a counter.
- Logical instructions such as OR and AND require that the current contents of the operand be read. They therefore will not function properly on write-only registers.
- The WDTMR register must be written within the first 60 internal system clocks (SCLK) of operation after a reset.

2.3 Z8 EXPANDED REGISTER FILE

The standard register file of the Z8 has been expanded to form 16 Expanded Register File (ERF) Banks (Figure 2-5). Each ERF Bank consists of up to 256 registers (the same amount as in the Standard Register File) that can then be divided into 16 Working Register Groups. This expansion allows for access to additional feature/peripheral control and data registers.

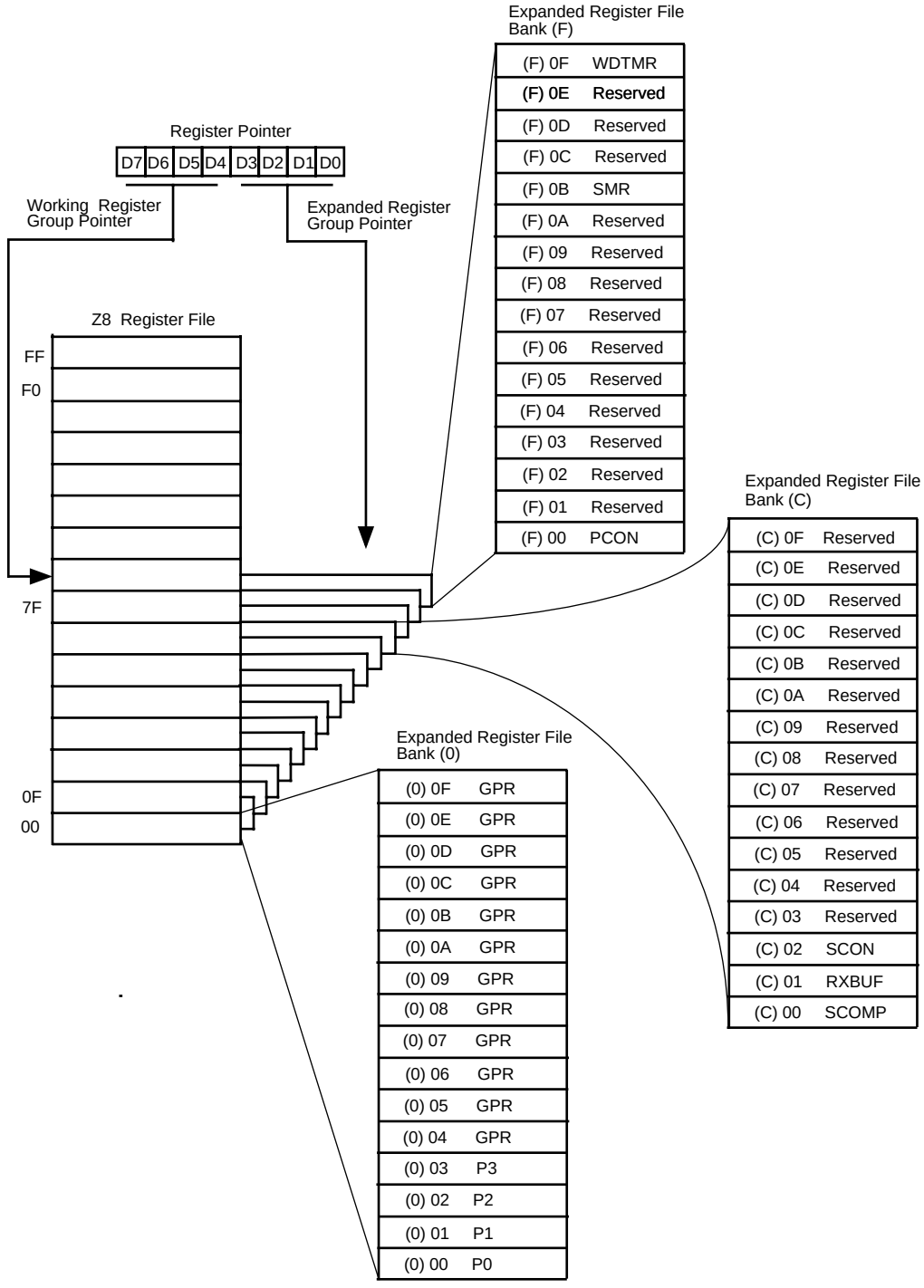


Figure 2-5. Expanded Register File Architecture

Note: The fully implemented register file is shown. Please refer to the specific product specification for actual register file architecture implemented.

2.3 Z8 EXPANDED REGISTER FILE (Continued)

Currently, three out of the possible sixteen Z8 ERF Banks have been implemented. ERF Bank 0, also known as the Z8 Standard Register File, has all 256 bytes defined (Figure 2-1). Only Working Register Group 0 (register addresses 00H to 0FH) have been defined for ERF Bank C and ERF Bank F (Table 2-4). All other working register groups in ERF Banks C and F, as well as the remaining thirteen ERF Banks, are not implemented. All are reserved for future use.

When an ERF Bank is selected, register addresses 00H to 0FH access those sixteen ERF Bank registers – in effect replacing the first sixteen locations of the Z8 Standard Register File.

For example, if ERF Bank C is selected, the Z8 Standard Registers 00H through 0FH are no longer accessible. Registers 00H through 0FH are now the 16 registers from ERF Bank C, Working Register Group 0. No other Z8 Standard Registers are effected since only Working Register Group 0 is implemented in ERF Bank C.

Access to the ERF is accomplished through the Register Pointer (FDH). The lower nibble of the Register Pointer determines the ERF Bank while the upper nibble determines the Working Register Group within the register file (Figure 2-6).

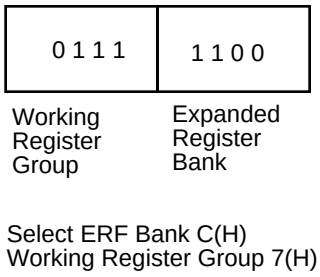


Figure 2-6. Register Pointer (FDH) Example

The value of the lower nibble in the Register Pointer (FDH) corresponds to the ERF Bank identification. Table 2.3 shows the lower nibble value and the register file assigned to it.

Table 2-3. ERF Bank Address

Register Pointer (FDH) Low Nibble	Hex	Register File
0000(B)	0	Z8 Standard Register File *
0001(B)	1	Expanded Register File Bank 1
0010(B)	2	Expanded Register File Bank 2
0011(B)	3	Expanded Register File Bank 3
0100(B)	4	Expanded Register File Bank 4
0101(B)	5	Expanded Register File Bank 5
0110(B)	6	Expanded Register File Bank 6
0111(B)	7	Expanded Register File Bank 7
1000(B)	8	Expanded Register File Bank 8
1001(B)	9	Expanded Register File Bank 9
1010(B)	A	Expanded Register File Bank A
1011(B)	B	Expanded Register File Bank B
1100(B)	C	Expanded Register File Bank C
1101(B)	D	Expanded Register File Bank D
1110(B)	E	Expanded Register File Bank E
1111(B)	F	Expanded Register File Bank F

Note: The Z8 Standard Register File is equivalent to Expanded Register File Bank 0.

The upper nibble of the register pointer selects which group of 16 bytes in the Register File, out of the full 256, will be accessed as working registers.

For example:
(See Figure 2-4)

```
R253 RP = 00H    ;ERF Bank 0, Working Reg. Group 0.
                 R0 = Port 0 = 00H
                 R1 = Port 1 = 01H
                 R2 = Port 2 = 02H
                 R3 = Port 3 = 03H
                 R11 = GPR 0BH
                 R15 = GPR 0FH
If:
R253 RP = 0FH    ;ERF Bank F, Working Reg. Group 0.
                 R0 = PCON = 00H
                 R1 = Reserved = 01H
                 R2 = Reserved = 02H
                 R11 = SMR = 0BH
                 R15 = WDTMR = 0FH
If:
R253 RP = FFH    ;ERF Bank F, Working Reg. Group F.
                 00H = PCON
                 R0 = SI0        01H= Reserved
                 R1 = TMR        02H= Reserved
                                 ...
                 R2 = T1         0BH = SMR
                                 ...
                 R15 = SPL       0FH = WDTMR
```

Since enabling an ERF Bank (C or F) only changes register addresses 00H to 0FH, the working register pointer can be used to access either the selected ERF Bank (Bank C or F, Working Register Group 0) or the Z8 Standard Register File (ERF Bank 0, Working Register Groups 1 through F).

Note: When an ERF Bank other than Bank 0 is enabled, the first 16 bytes of the Z8 Standard Register File (I/O ports 0 to 3, Groups 4 to F) are no longer accessible (the selected ERF Bank, Registers 00H to 0FH are accessed instead). It is important to re-initialize the Register Pointer to enable ERF Bank 0 when these registers are required for use.

The SPI register is mapped into ERF Bank C. Access is easily done using the following example:

```
LD      RP, #0CH    ;Select ERF Bank C working
                    ;register group 0 for access.
LD      R2,#xx      ;access SCON
LD      R1, #xx     ;access RXBUF
LD      RP, #00H    ;Select ERF Bank 0 so I/O ports
                    ;are again accessible.
```

Table 2-4. Z8 Expanded Register File Bank Layout

Expanded Register File Bank	ERF
F(H)	PCON, SMR, WDT, (00H, 0BH, 0FH), Working Register Group 0 only implemented.
E(H)	Not Implemented (Reserved)
D(H)	Not Implemented (Reserved)
C(H)	SPI Registers: SCOMP, RXBUF, SCON (00H, 01H, 02H), Working Register Group 0 only implemented.
B(H)	Not Implemented (Reserved)
A(H)	Not Implemented (Reserved)
9(H)	Not Implemented (Reserved)
8(H)	Not Implemented (Reserved)
7(H)	Not Implemented (Reserved)
6(H)	Not Implemented (Reserved)
5(H)	Not Implemented (Reserved)
4(H)	Not Implemented (Reserved)
3(H)	Not Implemented (Reserved)
2(H)	Not Implemented (Reserved)
1(H)	Not Implemented (Reserved)
0(H)	Z8 Ports 0, 1, 2, 3, and General-Purpose Registers 04H to EFH, and control registers F0H to FFH.

Please refer to the specific product specification to determine the above registers are implemented.

2.4 Z8 CONTROL AND PERIPHERAL REGISTERS

2.4.1 Standard Z8 Registers

The standard Z8 control registers govern the operation of the CPU. Any instruction which references the register file can access these control registers. Available control registers are:

- Interrupt Priority Register (IPR)
- Interrupt Mask Register (IMR)
- Interrupt Request Register (IRQ)
- Program Control Flags (FLAGS)
- Register Pointer (RP)
- Stack Pointer High-Byte (SPH)
- Stack Pointer Low-Byte (SPL)

The Z8 uses a 16-bit Program Counter (PC) to determine the sequence of current program instructions. The PC is not an addressable register.

Peripheral registers are used to transfer data, configure the operating mode, and control the operation of the on-chip peripherals. Any instruction that references the register file can access the peripheral registers. The peripheral registers are:

- Serial I/O (SIO)
- Timer Mode (TMR)
- Timer/Counter 0 (T0)
- T0 Prescaler (PRE0)
- Timer/Counter 1 (T1)
- T1 Prescaler (PRE1)
- Port 0–1 Mode (P01M)
- Port 2 Mode (P2M)
- Port 3 Mode (P3M)

In addition, the four port registers (P0–P3) are considered to be peripheral registers.

2.4.2 Expanded Z8 Registers

The expanded Z8 control registers govern the operation of additional features or peripherals. Any instruction which references the register file can access these registers.

The ERF contains the control registers for WDT, Port Control, Serial Peripheral Interface (SPI), and the SMR functions. Figure 2-4 shows the layout of the Register Banks in the ERF. Register Bank C in the ERF consists of the registers for the SPI. Table 2-5 shows the registers within ERF Bank C, Working Register Group 0.

Table 2-5. Expanded Register File Register Bank C, WR Group 0

Register	Register Function	Working Register
F	Reserved	R15
E	Reserved	R14
D	Reserved	R13
C	Reserved	R12
B	Reserved	R11
A	Reserved	R10
9	Reserved	R9
8	Reserved	R8
7	Reserved	R7
6	Reserved	R6
5	Reserved	R5
4	Reserved	R4
3	Reserved	R3
2	SPI Control (SCON)	R2
1	SPI Tx/Rx Data (Roxburgh)	R1
0	SPI Compare (SCOMP)	R0

Working Register Group 0 in ERF Bank 0 consists of the registers for Z8 General-Purpose Registers and ports. Table 2-6 shows the registers within this group.

Table 2-6. Expanded Register File Bank 0, WR Group 0

Register	Register Function	Working Register
F	General-Purpose Register	R15
E	General-Purpose Register	R14
D	General-Purpose Register	R13
C	General-Purpose Register	R12
B	General-Purpose Register	R11
A	General-Purpose Register	R10
9	General-Purpose Register	R9
8	General-Purpose Register	R8
7	General-Purpose Register	R7
6	General-Purpose Register	R6
5	General-Purpose Register	R5
4	General-Purpose Register	R4
3	Port 3	R3
2	Port 2	R2
1	Port 1	R1
0	Port 0	R0

Working Register Group 0 in ERF Bank F consists of the control registers for STOP mode, WDT, and port control. Table 2-7 shows the registers within this group.

Table 2-7. Expanded Register File Bank F, WR Group 0

Register	Register Function	Working Register
F	WDTMR	R15
E	Reserved	R14
D	Reserved	R13
C	Reserved	R12
B	SMR	R11
A	Reserved	R10
9	Reserved	R9
8	Reserved	R8
7	Reserved	R7
6	Reserved	R6
5	Reserved	R5
4	Reserved	R4
3	Reserved	R3
2	Reserved	R2
1	Reserved	R1
0	PCON	R0

The functions and applications of the control and peripheral registers are described in subsequent sections of this manual.

2.5 PROGRAM MEMORY

The first 12 bytes of Program Memory are reserved for the interrupt vectors (Figure 2-7). These locations contain six 16-bit vectors that correspond to the six available interrupts. Address 12 up to the maximum ROM address consists of on-chip mask-programmable ROM. See the product data sheet for the exact program, data, register memory size, and address range available. At addresses outside the internal ROM, the Z8 executes external program memory fetches through Port 0 and Port 1 in Address/Data mode for devices with Port 0 and Port 1 featured. Otherwise, the program counter will continue to execute NOPs up to address FFFFH, roll over to 0000H, and continue to fetch executable code (Figure 2-7).

The internal program memory is one-time programmable (OTP) or mask programmable dependent on the specific device. *A ROM protect feature prevents dumping of the ROM contents by inhibiting execution of the LDC, LDCI, LDE, and LDEI instructions to Program Memory in all modes. ROM look-up tables cannot be used with this feature.*

The ROM Protect option is mask-programmable, to be selected by the customer when the ROM code is submitted. For the OTP ROM, the ROM Protect option is an OTP programming option.

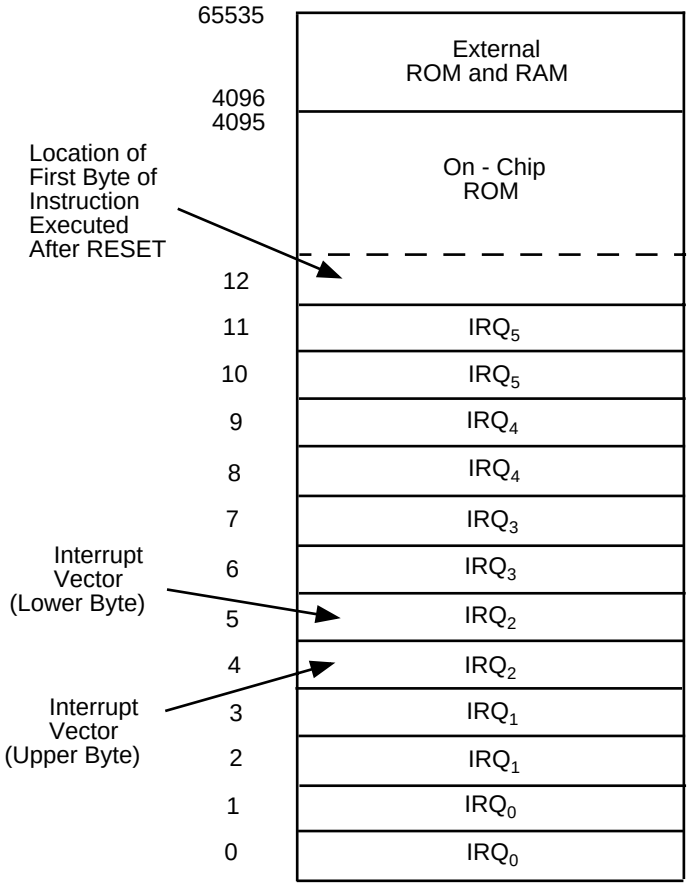


Figure 2-7. Z8 Program Memory Map

2.6 Z8 EXTERNAL MEMORY

The Z8, in some cases, has the capability to access external program memory with the 16-bit Program Counter. To access external program memory the Z8 offers multiplexed address/data lines (AD7-AD0) on Port 1 and address lines (A15-A8) on Port 0. This feature only applies to devices that offer Port 0 and Port 1. The maximum external address is FFFF. This memory interface is supported by the control lines $\overline{AS}$ (Address Strobe), $\overline{DS}$ (Data Strobe), and R/W (Read/Write). The origin of the external program memory starts after the last address of the internal ROM. Figure 2-8 shows an example of external program memory for the Z8.

2.6.1 External Data Memory (/DM)

The Z8, in some cases, can address up to 60 Kbytes of external data memory beginning at location 4096. External Data Memory may be included with, or separated from, the external Program Memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space. The state of the $\overline{DM}$ signal is controlled by the type of instruction being executed. An LDC opcode references Program ($\overline{DM}$ inactive) Memory, and an LDE instruction references Data ($\overline{DM}$ active Low) Memory. The user must configure Port 3 Mode Register (P3M) bits D3 and D4 for this mode.

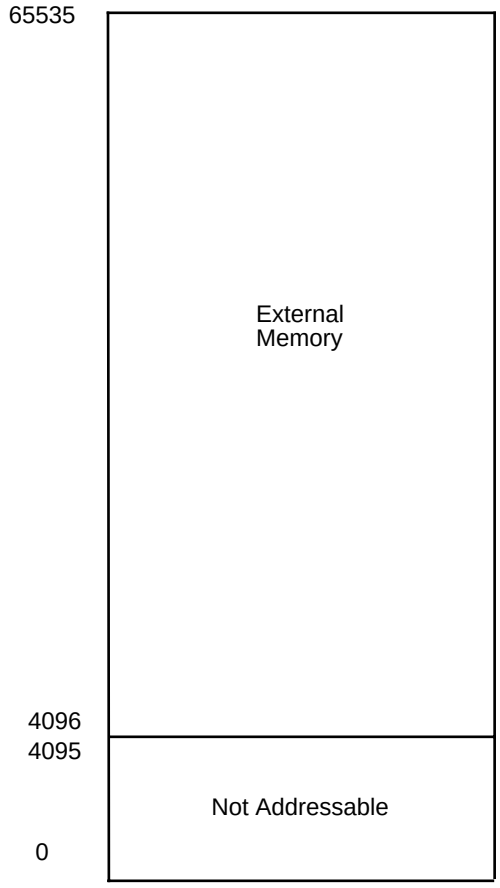


Figure 2-8. External Memory Map

Note: For additional information on using external memory, see Chapter 10 of this manual. For exact memory addressing options available, see the device product specification.

2.7 Z8 STACKS

Stack operations can occur in either the Z8 MCU Standard Register File or external data memory. Under software control, Port 0–1 Mode register (F8H) selects the stack location. Only the General-Purpose Registers can be used for the stack when the internal stack is selected.

The register pair FEH and FFH form the 16-bit Stack Pointer (SP), that is used for all stack operations. The stack address is stored with the MSB in FEH and LSB in FFH (Figure 2-9).

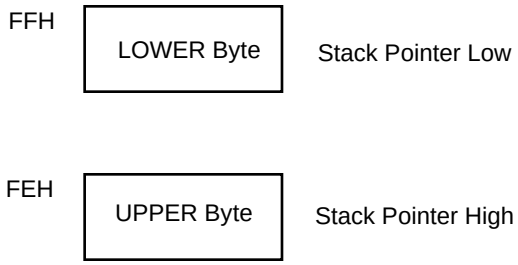


Figure 2-9. Stack Pointer

The stack address is decremented prior to a PUSH operation and incremented after a POP operation. The stack address always points to the data stored on the top of the stack. The Z8 stack is a return stack for CALL instructions and interrupts, as well as a data stack.

During a CALL instruction, the contents of the PC are saved on the stack. The PC is restored during a RETURN instruction. Interrupts cause the contents of the PC and Flag registers to be saved on the stack. The IRET instruction restores them (Figure 2-10).

When the Z8 is configured for an internal stack (using the Z8 Standard Register File), register FFH serves as the Stack Pointer. The value in FEH is ignored. FEH can be used as a general-purpose register in this case only.

An overflow or underflow can occur when the stack address is incremented or decremented during normal stack operations. The programmer must prevent this occurrence or unpredictable operation will result.

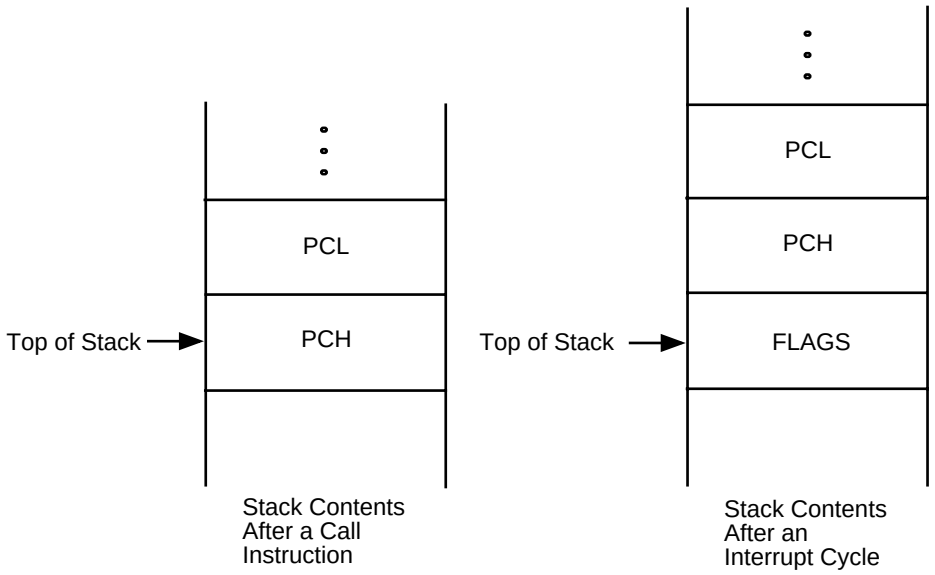


Figure 2-10. Stack Operations

Totally Logical

CHAPTER 3
CLOCK

3.1 CLOCK

The Z8 MCU® derives its timing from on-board clock circuitry connected to pins XTAL1 and XTAL2. The clock circuitry consists of an oscillator, a divide-by-two shaping circuit, and a clock buffer. Figure 3-1 illustrates the clock circuitry. The oscillator's input is XTAL1 and its output is XTAL2. The clock can be driven by a crystal, a ceramic resonator, LC clock, RC, or an external clock source.

3.1.1 Frequency Control

In some cases, the Z8 has an EPROM/OTP option or a Mask ROM option bit to bypass the divide-by-two flip flop in Figure 3-1. This feature is used in conjunction with the low EMI option. When low EMI is selected, the device output drive and oscillator drive is reduced to approximately 25 percent of the standard drive and the divide-by-two flip flop is bypassed such that the XTAL clock frequency is equal to the internal system clock frequency. In this mode, the maximum frequency of the XTAL clock is 4 MHz. Please refer to specific product specification for availability of options and output drive characteristics.

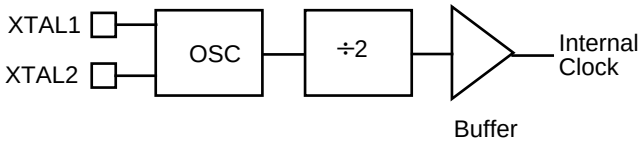
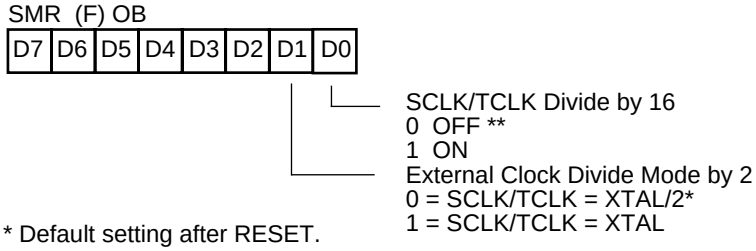


Figure 3-1. Z8 Clock Circuit

3.2 CLOCK CONTROL

In some cases, the Z8 offers software control of the internal system clock via programming register bits. The bits are located in the Stop-Mode Recovery Register in Expanded Register File Bank F, Register 0BH. This register selects the clock divide value and determines the mode of Stop-Mode Recovery (Figure 3-2). Please refer to the specific product specification for availability of this feature/register.



* Default setting after RESET.

**Default setting after RESET and STOP-Mode Recovery.

Figure 3-2. Stop-Mode Recovery Register (Write-Only Except D7, Which is Read-Only)

3.2.1 SCLK/TCLK Divide-By-16 Select (D0)

This bit of the SMR controls a divide-by-16 prescalar of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

3.2.2 External Clock Divide-By-Two (D1)

This bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, SCLK (System Clock) and TCLK (Timer Clock) are equal to the external clock frequency divided by two. The SCLK/TCLK is equal to the external clock frequency when this bit is set (D1 = 1). Using this bit, together with D7 of PCON, further helps lower EMI (D7 (PCON) = 0, D1 (SMR) = 1). The default setting is 0. Maximum frequency is 4 MHz with D1=1 (Figure 3-3).

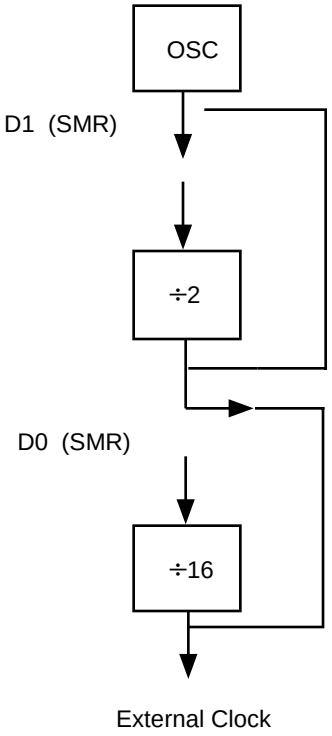


Figure 3-3. External Clock Circuit

3.3 OSCILLATOR CONTROL

In some cases, the Z8 MCU offers software control of the oscillator to select low EMI drive or standard drive. The selection is done by programming bit D7 of the Port Configuration (PCON) register (Figure 3-4). The PCON register is located in Expanded Register File Bank F, Register 00H.

A 1 in bit D7 configures the oscillator with standard drive, while a 0 configures the oscillator with Low EMI drive. This only affects the drive capability of the oscillator and does not affect the relationship of the XTAL clock frequency to the internal system clock (SCLK).

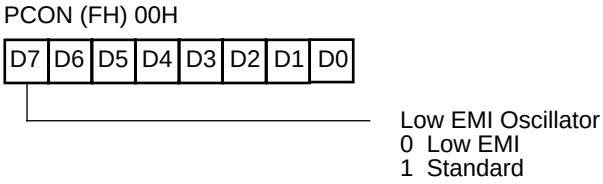


Figure 3-4. Port Configuration Register (PCON) (Write-Only)

3.4 OSCILLATOR OPERATION

The Z8® MCU uses a Pierce oscillator with an internal feedback (Figure 3-5). The advantages of this circuit are low cost, large output signal, low-power level in the crystal, stability with respect to V_{CC} and temperature, and low impedances (not disturbed by stray effects).

One draw back is the need for high gain in the amplifier to compensate for feedback path losses. The oscillator amplifies its own noise at start-up until it settles at the frequency that satisfies the gain/phase requirements A x B = 1, where $A = V_0/V_i$ is the gain of the amplifier and $B = V_i/V_0$ is the gain of the feedback element. The total phase shift around the loop is forced to zero (360 degrees). Since VIN must be in phase with itself, the amplifier/inverter provides 180 degree phase shift and the feedback element is forced to provide the other 180 degrees of phase shift.

R1 is a resistive component placed from output to input of the amplifier. The purpose of this feedback is to bias the amplifier in its linear region and to provide the start-up transition.

Capacitor C_2 combined with the amplifier output resistance provides a small phase shift. It will also provide some attenuation of overtones.

Capacitor C_1 combined with the crystal resistance provides additional phase shift.

C_1 and C_2 can affect the start-up time if they increase dramatically in size. As C_1 and C_2 increase, the start-up time increases until the oscillator reaches a point where it does not start up any more.

It is recommended for fast and reliable oscillator start-up (over the manufacturing process range) that the load capacitors be sized as low as possible without resulting in overtone operation.

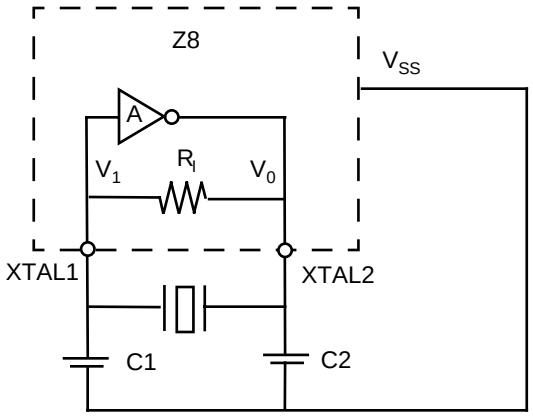


Figure 3-5. Pierce Oscillator with Internal Feedback Circuit

3.4.1 Layout

Traces connecting crystal, caps, and the Z8 oscillator pins should be as short and wide as possible. This reduces parasitic inductance and resistance. The components (caps, crystal, resistors) should be placed as close as possible to the oscillator pins of the Z8.

The traces from the oscillator pins of the IC and the ground side of the lead caps should be guarded from all other traces (clock, V_{CC}, address/data lines, system ground) to reduce cross talk and noise injection. This is usually accomplished by keeping other traces and system ground trace planes away from the oscillator circuit and by placing a Z8 device V_{SS} ground ring around the traces/components. The ground side of the oscillator lead caps should be connected to a single trace to the Z8 V_{SS} (GND) pin. It should not be shared with any other system ground trace or components except at the Z8 device V_{SS} pin. This is to prevent differential system ground noise injection into the oscillator (Figure 3-6).

3.4.2 Indications of an Unreliable Design

There are two major indicators that are used in working designs to determine their reliability over full lot and temperature variations. They are:

Start-up Time. If start -up time is excessive, or varies widely from unit to unit, there is probably a gain problem. C1/C2 needs to be reduced; the amplifier gain is not adequate at frequency, or crystal Rs is too large.

Output Level. The signal at the amplifier output should swing from ground to V_{CC}. This indicates there is adequate gain in the amplifier. As the oscillator starts up, the signal amplitude grows until clipping occurs, at which point the loop gain is effectively reduced to unity and constant oscillation is achieved. A signal of less than 2.5 volts peak-to-peak is an indication that low gain may be a problem. Either C_1 or C_2 should be made smaller or a low-resistance crystal should be used.

3.4.3 Circuit Board Design Rules

The following circuit board design rules are suggested:

- To prevent induced noise the crystal and load capacitors should be physically located as close to the Z8® as possible.
- Signal lines should not run parallel to the clock oscillator inputs. In particular, the crystal input circuitry and the internal system clock output should be separated as much as possible.
- V_{CC} power lines should be separated from the clock oscillator input circuitry.
- Resistivity between XTAL1 or XTAL2 and the other pins should be greater than 10 Mohms.

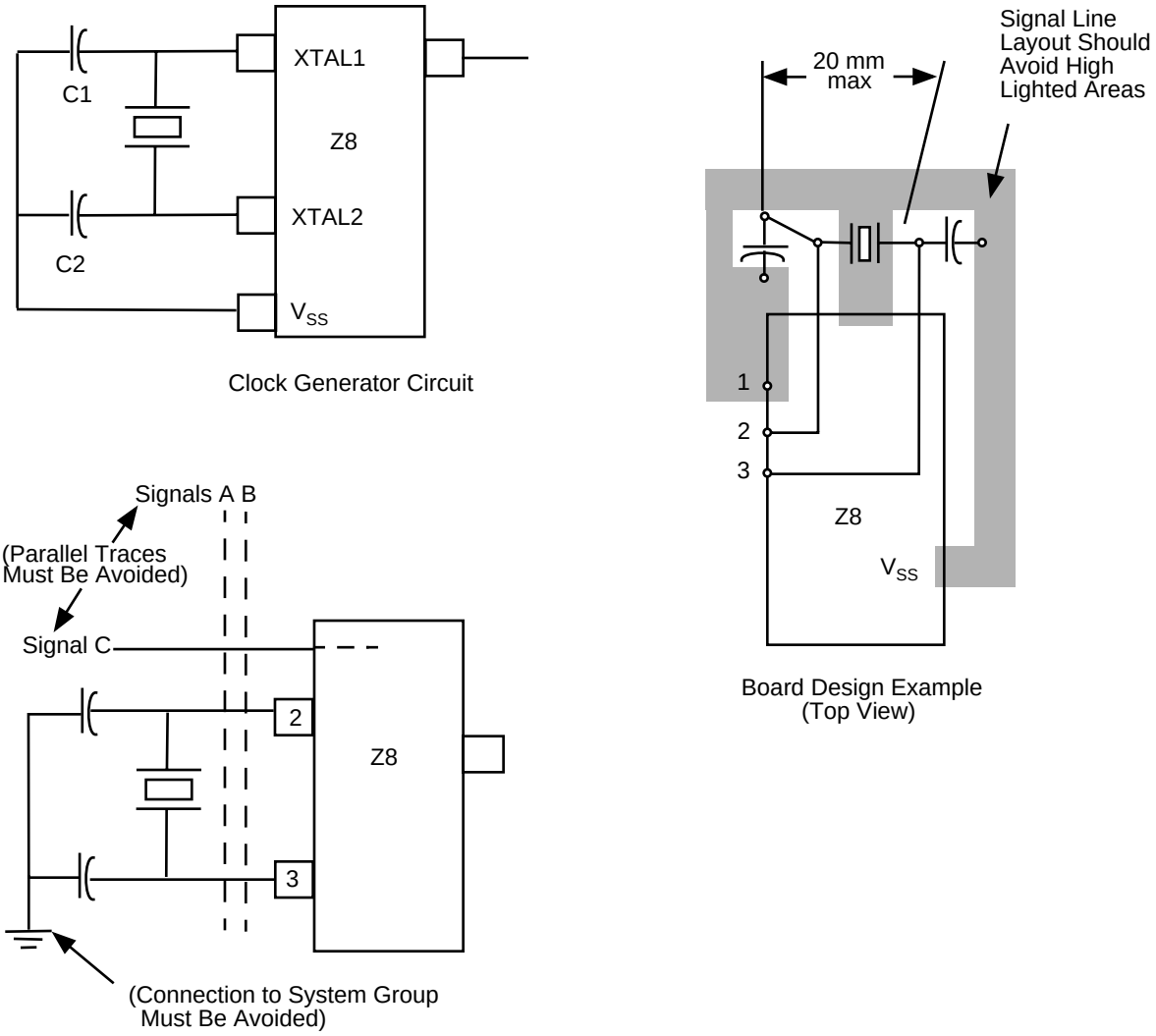


Figure 3-6. Circuit Board Design Rules

3.4.4 Crystals and Resonators

Crystals and ceramic resonators (Figure 3-7) should have the following characteristics to ensure proper oscillator operation:

Crystal Cut	AT (crystal only)
Mode	Parallel, Fundamental Mode
Crystal Capacitance	<7pF
Load Capacitance	10pF < CL < 220 pF, 15 typical
Resistance	100 ohms max

Depending on operation frequency, the oscillator may require the addition of capacitors C1 and C2 (shown in Figures 3-7). The capacitance values are dependent on the manufacturer's crystal specifications.

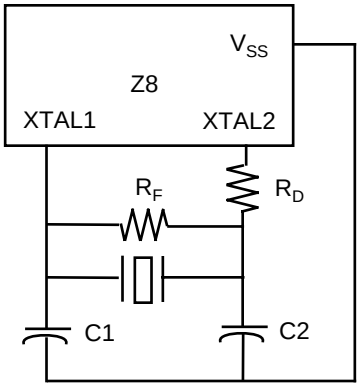


Figure 3-7. Crystal/Ceramic Resonator Oscillator

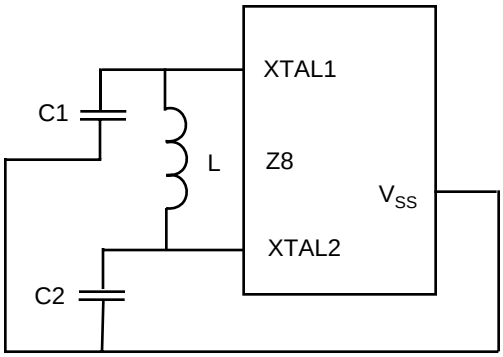


Figure 3-8. LC Clock

In most cases, the R_D is 0 Ohms and R_F is infinite. It is determined and specified by the crystal/ceramic resonator manufacturer. The R_D can be increased to decrease the amount of drive from the oscillator output to the crystal. It can also be used as an adjustment to avoid clipping of the oscillator signal to reduce noise. The R_F can be used to improve the start-up of the crystal/ceramic resonator. The Z8 oscillator already has an internal shunt resistor in parallel to the crystal/ceramic resonator.

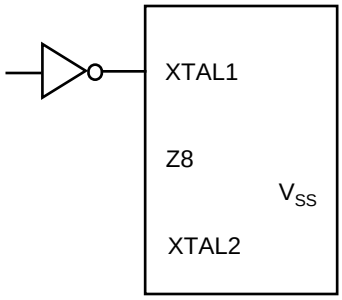


Figure 3-9. External Clock

It is recommended in Figures 3-7, 3-8, and 3-9 to connect the load capacitor ground trace directly to the V_{SS} (GND) pin of the Z8®. This ensures that no system noise is injected into the Z8 clock. This trace should not be shared with any other components except at the V_{SS} pin of the Z8.

In some cases, the Z8 XTAL1 pin also functions as one of the EPROM high-voltage mode programming pins or as a special factory test pin. In this case, applying 2 V above V_{CC} on the XTAL1 pin will cause the device to enter one of these modes. Since this pin accepts high voltages to enter these respective modes, the standard input protection diode to V_{CC} is not on XTAL1. It is recommended that in applications where the Z8 is exposed to much system noise, a diode from XTAL1 to V_{CC} be used to prevent accidental enabling of these modes. This diode will not affect the crystal/ceramic resonator operation.

Please note that a parallel resonant crystal or resonator data sheet will specify a load capacitor value that is the series combination of C_1 and C_2, including all parasitics (PCB and holder).

3.5 LC OSCILLATOR

The Z8 oscillator can use a LC network to generate a XTAL clock (Figure 3-8).

The frequency stays stable over V_{CC} and temperature. The oscillation frequency is determined by the equation:

$$\text{Frequency} = \frac{1}{2\,\pi\,(LCT)1/2}$$

where L is the total inductance including parasitics and C_T is the total series capacitance including the parasitics.

Simple series capacitance is calculated using the following equation:

$$\frac{1}{C_T} = \frac{1}{C_1} + \frac{1}{C_2}$$

$$\text{If } C_1 = C_2$$

$$\frac{1}{C_T} = \frac{2}{C_1}$$

$$C_1 = 2CT$$

Sample calculation of capacitance C_1 and C_2 for 5.83 MHz frequency and inductance value of 27 uH:

$$5.83\ (10\text{^}6) = \frac{1}{2\pi\,[2.7\ (10^{-6})\ CT]\ 1/2}$$

$$CT = 27.6\ pf$$

Thus C_1 = 55.2 pf and C_2 = 55.2 pf.

3.6 RC OSCILLATOR

In some cases, the Z8 has a RC oscillator option. Please refer to the specific product specification for availability. The RC oscillator requires a resistor across XTAL1 and XTAL2. An additional load capacitor is required from the XTAL1 input to V_{SS} pin (Figure 3-10).

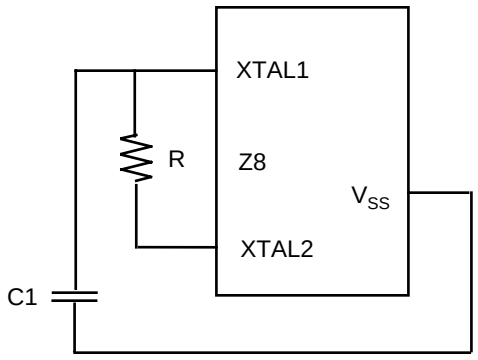


Figure 3-10. RC Clock

Totally Logical

CHAPTER 4
RESET—WATCH-DOG TIMER

4.1 RESET

This section describes the Z8 MCU® reset conditions, reset timing, and register initialization procedures. Reset is generated by Power-On Reset (POR), Reset Pin, Watch-Dog Timer (WDT), and Stop-Mode Recovery.

A system reset overrides all other operating conditions and puts the Z8 into a known state. To initialize the chip's internal logic, the $\overline{\text{RESET}}$ input must be held Low for at least 21 SCP or 5 XTAL clock cycles. The control register and ports are reset to their default conditions after a POR, a reset from the $\overline{\text{RESET}}$ pin, or Watch-Dog Timer timeout while in RUN mode and HALT mode. The control registers and ports are not reset to their default conditions after Stop-Mode Recovery and WDT timeout while in STOP mode.

While $\overline{\text{RESET}}$ pin is Low, $\overline{\text{AS}}$ is output at the internal clock rate, $\overline{\text{DS}}$ is forced Low, and R//W remains High. The program counter is loaded with 000CH. I/O ports and control registers are configured to their default reset state.

Resetting the Z8 does not effect the contents of the general-purpose registers.

4.2 RESET PIN, INTERNAL POR OPERATION

In some cases, the Z8 hardware $\overline{\text{RESET}}$ pin initializes the control and peripheral registers, as shown in Tables 4-1, 4-2, 4-3, and 4-4. Specific reset values are shown by 1 or 0, while bits whose states are unknown are indicated by the letter U. The Tables 4-1, 4-2, 4-3, and 4-4 show the reset conditions for the generic Z8.

Note: The register file reset state is device dependent. Please refer to the selected device product specifications for register availability and reset state.

4.2 RESET PIN, INTERNAL POR OPERATION (Continued)

Table 4-1. Sample Control and Peripheral Register Reset Values (ERF Bank 0)

Register (HEX)	Register Name	Bits 7	6	5	4	3	2	1	0	Comments
F0	Serial I/O	U	U	U	U	U	U	U	U	
F1	Timer Mode	0	0	0	0	0	0	0	0	Counter/Timers Stopped
F2	Counter/Timer1	U	U	U	U	U	U	U	U	
F3	T1 Prescaler	U	U	U	U	U	U	0	0	Single-Pass Count Mode, External Clock Source
F4	Counter/Timer0	U	U	U	U	U	U	U	U	
F5	T0 Prescaler	U	U	U	U	U	U	U	0	Single-Pass Count Mode
F6	Port 2 Mode	1	1	1	1	1	1	1	1	All Inputs
F7	Port 3 Mode	0	0	0	0	0	0	0	0	Port 2 Open-Drain, P33–P30 Input, P37–P34 Output
F8	Port 0–1 Mode	0	1	0	0	1	1	0	1	Internal Stack, Normal Memory Timing
F9	Interrupt Priority	U	U	U	U	U	U	U	U	
FA	Interrupt Request	0	0	0	0	0	0	0	0	All Interrupts Cleared
FB	Interrupt Mask	0	U	U	U	U	U	U	U	Interrupts Disabled
FC	Flags	U	U	U	U	U	U	U	U	
FD	Register Pointer	0	0	0	0	0	0	0	0	
FE	Stack Pointer (High)	U	U	U	U	U	U	U	U	
FF	Stack Pointer (Low)	U	U	U	U	U	U	U	U	

Program execution starts 5 to 10 clock cycles after internal $\overline{\text{RESET}}$ has returned High. The initial instruction fetch is from location 000CH. Figure 4-1 shows reset timing.

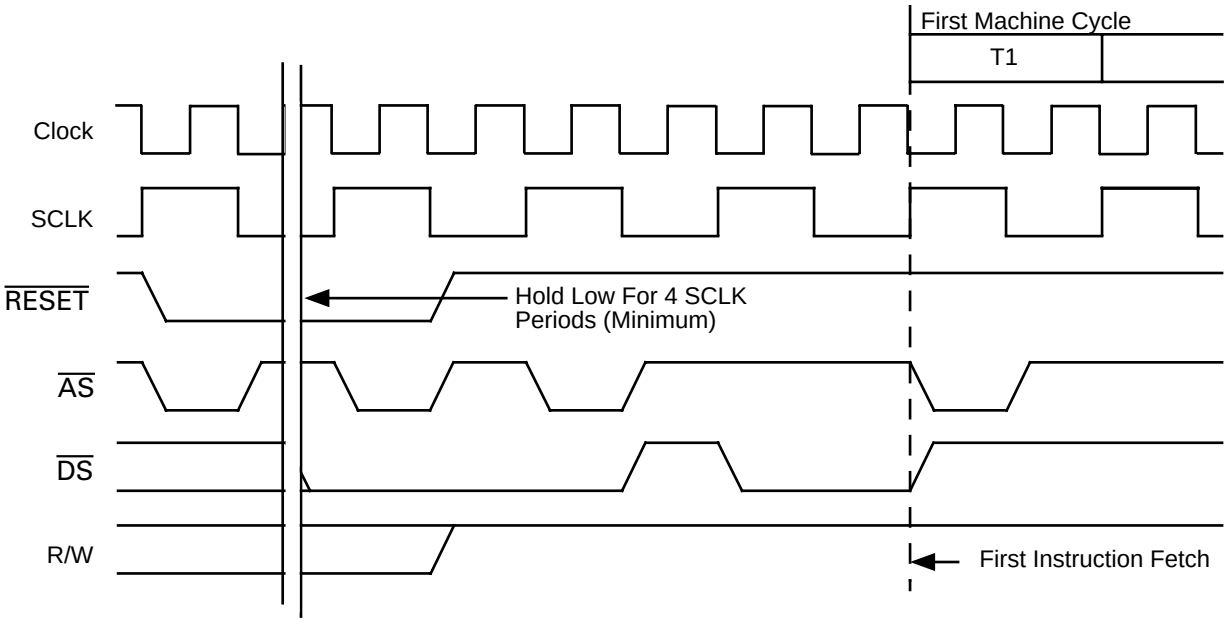


Figure 4-1. Reset Timing

After a reset, the first routine executed should be one that initializes the control registers to the required system configuration.

The $\overline{\text{RESET}}$ pin is the input of a Schmitt-triggered circuit. Resetting the Z8 will initialize port and control registers to their default states. To form the internal reset line, the output of the trigger is synchronized with the internal clock. The clock must therefore be running for $\overline{\text{RESET}}$ to function. It requires 4 internal system clocks after reset is detected for the Z8 to reset the internal circuitry. An internal pull-up, combined with an external capacitor of 1 uf, provides enough time to properly reset the Z8 (Figure 4-2). In some cases, the Z8 has an internal POR timer circuit that holds the Z8 in reset mode for a duration (T_{POR}) before releasing the device out of reset. On these Z8 devices, the internally generated reset drives the reset pin low for the POR time. Any devices driving the reset line must be open-drained in order to avoid damage from possible conflict during reset conditions. This reset time allows the on-board clock oscillator to stabilize.

To avoid asynchronous and noisy reset problems, the Z8 is equipped with a reset filter of four external clocks (4TpC). If the external reset signal is less than 4TpC in duration, no reset occurs. On the fifth clock after the reset is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external reset, whichever is longer. During the reset cycle, $\overline{\text{DS}}$ is held active low while $\overline{\text{AS}}$ cycles at a rate of the internal system clock. Program execution begins at location 000CH, 5-10 TpC cycles after $\overline{\text{RESET}}$ is released. For the internal Power-On Reset, the reset output time is specified as T_{POR}. Please refer to specific product specifications for actual values.

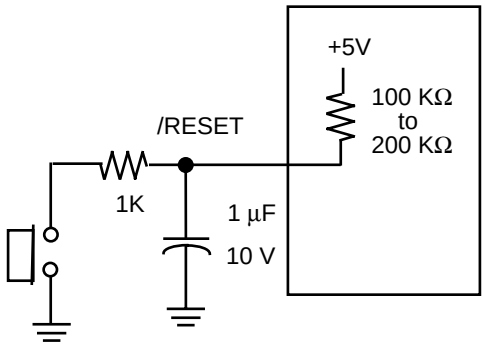


Figure 4-2. Example of External Power-On Reset Circuit

Table 4-2. Expanded Register File Bank 0 Reset Values at RESET

Register (HEX)	Register Name	Bits 7	6	5	4	3	2	1	0	Comments
00	Port 0	U	U	U	U	U	U	U	U	Input mode, output set to push-pull
01	Port 1	U	U	U	U	U	U	U	U	Input mode, output set to push-pull
02	Port 2	U	U	U	U	U	U	U	U	Input mode, output set to open drain
03	Port 3	1	1	1	1	U	U	U	U	Standard Digital input and output Z86L7X Family Device Port P34-P37 = 0 (Except Z86L70/71/75) All other Z8 = 1
04–EF	General-Purpose Registers 04-EF	U	U	U	U	U	U	U	U	Undefined

4.2 RESET PIN, INTERNAL POR OPERATION (Continued)

Table 4-3. Sample Expanded Register File Bank C Reset Values

Register (HEX)	Register Name	Bits 7	6	5	4	3	2	1	0	Comments
00	SPI Compare (SCOMP)	0	0	0	0	0	0	0	0	
01	Receive Buffer (RxBUF)	U	U	U	U	U	U	U	U	
02	SPI Control (SCON)	U	U	U	U	0	0	0	0	

Table 4-4. Sample Expanded Register File Bank F Reset Values

Register (HEX)	Register Name	Bits 7	6	5	4	3	2	1	0	Comments
00	Port Configuration (PCON)	1	1	1	1	1	1	1	0	Comparator outputs disabled on Port 3 Port 0 and 1 output is push-pull Port 0, 1, 2, 3, and oscillator with standard output drive
0B	STOP-Mode Recovery (SMR)	0	0	1	0	0	0	0	0	Clock divide by 16 off XTAL divide by 2 POR and / OR External Reset Stop delay on Stop recovery level is low, STOP flag is POR
0F	Watch-Dog Timer Mode (WDTMR)	U	U	U	0	1	1	0	1	512 TPC for WDT time out, WDT runs during STOP

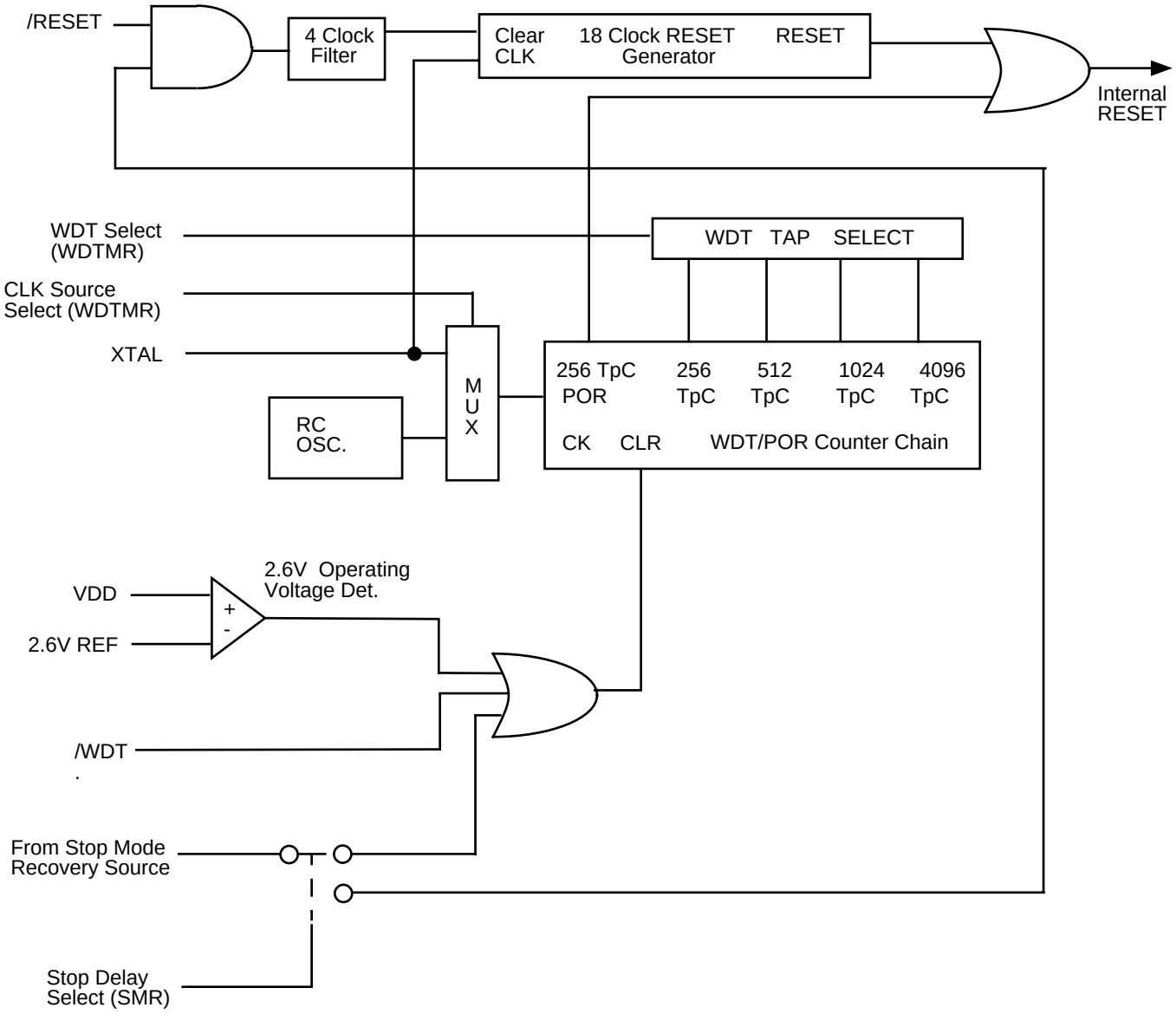


Figure 4-3. Example of Z8 Reset with $\overline{\text{RESET}}$ Pin, WDT, SMR, and POR

4.2 RESET PIN, INTERNAL POR OPERATION (Continued)

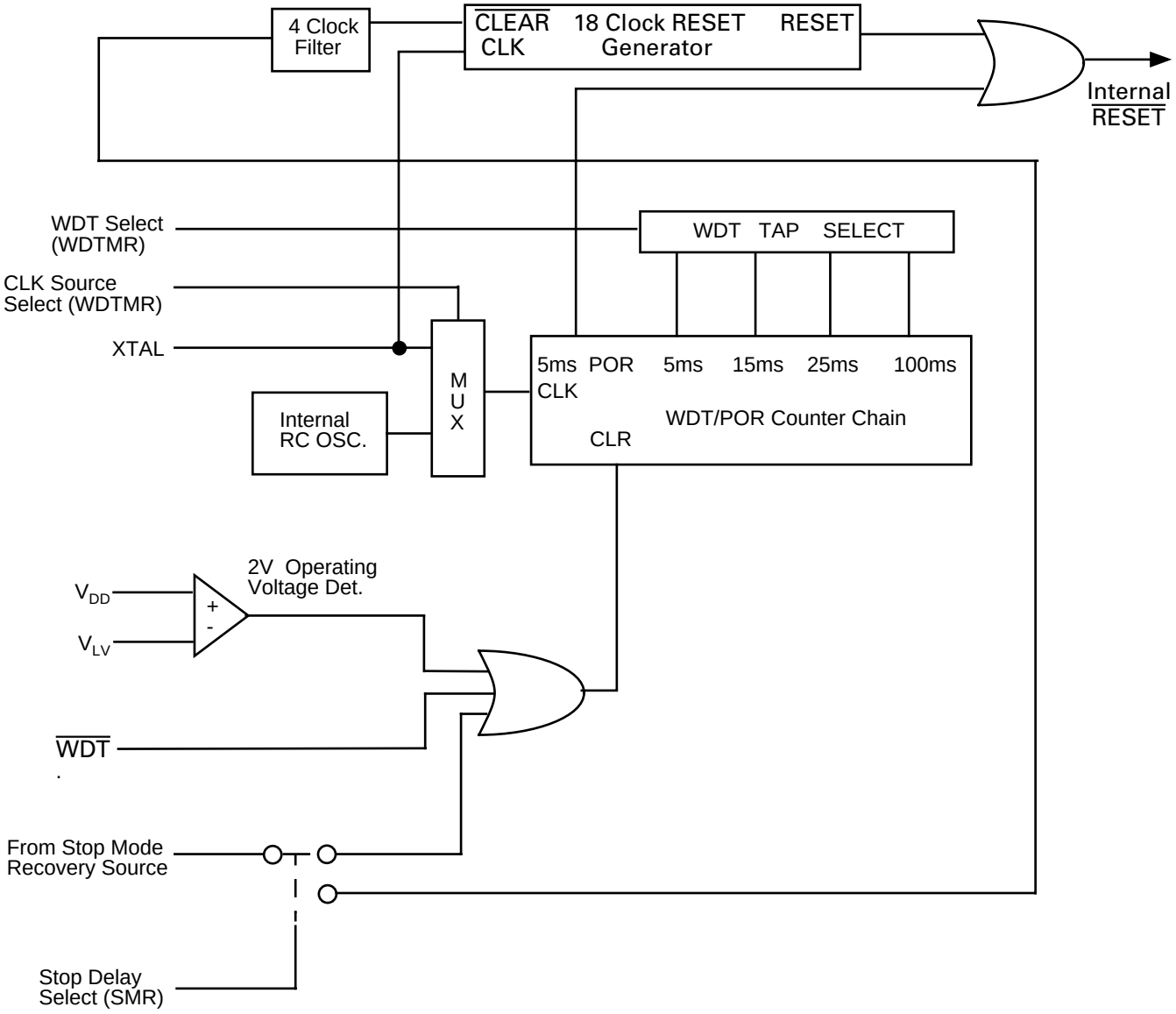


Figure 4-4. Example of Z8 Reset with WDT, SMR, and POR

4.3 WATCH-DOG TIMER (WDT)

The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. When operating in the RUN or HALT modes, a WDT reset is functionally equivalent to a hardware $\overline{\text{POR}}$ reset. The WDT is initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT cannot be disabled after it has been initially enabled. Permanently enabled WDTs are always enabled and the WDT instruction is used to refresh it. The WDT circuit is driven by an on-board RC oscillator or external oscillator from the XTAL1 pin. The POR clock source is selected with bit 4 of the Watch-Dog Timer Mode register (WDTMR). In some cases, a Z8 that offers the WDT but does not have a WDTMR register, has a fixed WDT timeout and uses the on board RC oscillator as the only clock source. Please refer to specific product specifications for selectability of timeout, WDT during HALT and STOP modes, source of WDT clock, and availability of the permanently-on WDT option.

Note: Execution of the WDT instruction affects the Z (zero), S (sign), and V (overflow) flags.

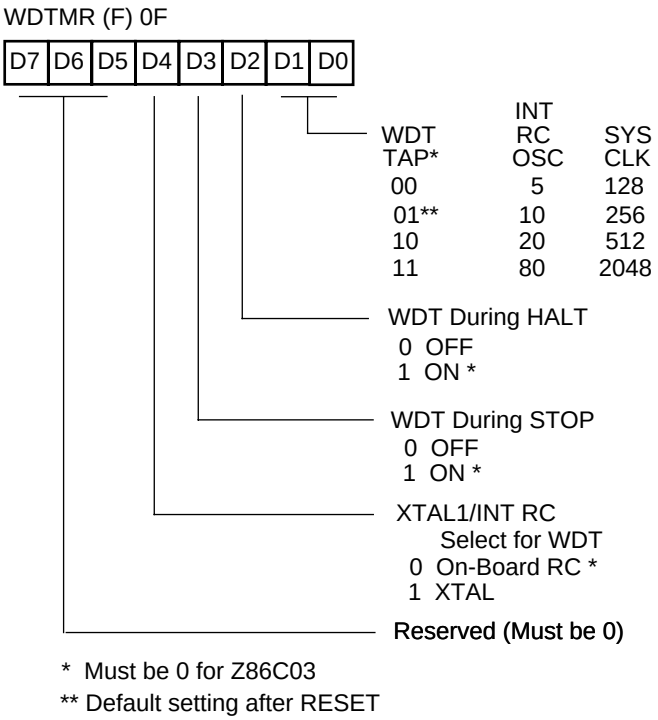


Figure 4-5. Example of Z8 Watch-Dog Timer Mode Register (Write-Only)

Note: The WDTMR register is accessible only during the first 60 processor cycles from the execution of the first instruction after Power-On Reset, Watch-Dog Reset or a Stop-Mode Recovery. After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR is a write-only register.

The WDTMR is located in Expanded Register File Bank F, register 0FH. The control bits are described as follows:

WDT Time Select (D1, D0). Bits 0 and 1 control a tap circuit that determines the time-out period. Table 4-5 shows the different values that can be obtained. The default value of D1 and D0 are 0 and 1, respectively.

Table 4-5. Time-Out Period of the WDT

Time-Out of D1	D0	Typical Time-Out of Internal RC OSC	SYS Clock
0	0	5 ms min	256TpC
0	1	15 ms min	512TpC
1	0	25 ms min	1024TpC
1	1	100 ms min	4096TpC

Notes:
TpC = XTAL clock cycle
The default on reset is, D0 = 1 and D1 = 0.
The values given are for VCC = 5.0V.
See the device product specification for exact WDTMR time out select options available.

WDT During HALT (D2). This bit determines whether or not the WDT is active during HALT mode. A 1 indicates active during HALT. The default is 1. A WDT time out during HALT mode will reset control register ports to their default reset conditions.

WDT During STOP (D3). This bit determines whether or not the WDT is active during STOP mode. Since XTAL clock is stopped during STOP Mode, unless as specified below, the on-board RC must be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1. If bits D3 and D4 are both set to 1, the WDT only, is driven by the external clock during STOP mode. This feature makes it possible to wake up from STOP mode from an internal source. Please refer to specific product specifications for conditions of control and port registers when the Z8 comes out of STOP mode. A WDT time out during STOP mode will not reset all control registers. The reset conditions of the ports from STOP mode due to WDT time out is the same as if recovered using any of the other STOP mode sources.

Clock Source for WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the internal RC oscillator.

Bits 5, 6 and 7. These bits are reserved.

V_{CC} Voltage Comparator. An on-board voltage comparator checks that V_{CC} is at the required level to insure correct operation of the device. Reset is globally driven if V_{CC} is below the specified voltage. This feature is available in select ROM Z8 devices. See the device product specification for feature availability and operating range.

4.4 POWER-ON-RESET (POR)

A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer (T_{POR}) function. The POR time allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power fail to Power OK status (cold start).
2. STOP-Mode Recovery (if bit 5 of SMR=1).
3. WDT timeout.

The POR time is specified as TPOR. On Z8 devices that feature a Stop-Mode Recovery register (SMR), bit 5 selects whether the POR timer is used after Stop-Mode Recovery or by-passed. If bit D5 = 1 then the POR timer is used. If bit 5 = 0 then the POR timer is by-passed. In this case, the Stop-Mode Recovery source must be held in the recovery state for 5 T_PC or 5 crystal clocks to pass the reset signal internally. This option is used when the clock is provided with an RC/LC clock. See the device product specification for timing details.

POR (cold start) will always reset the Z8 control and port registers to their default condition. If a Z8 has a SMR register, the warm start bit will be reset to a 0 to indicate POR.

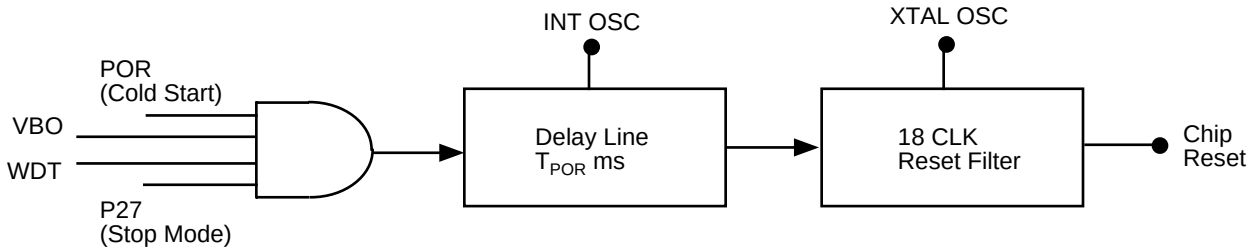


Figure 4-6. Example of Z8 with Simple SMR and POR

Totally Logical

CHAPTER 5
I/O PORTS

5.1 I/O PORTS

The Z8 has up to 32 lines dedicated to input and output. These lines are grouped into four 8-bit ports known as Port 0, Port 1, Port 2, and Port 3. Port 0 is nibble programmable as input, output, or address. Port 1 is byte configurable as input, output, or address/data. Port 2 is bit programmable as either inputs or outputs, with or without handshake and SPI. Port 3 can be programmed to provide timing, serial and parallel input/output, or comparator input/output.

All ports have push-pull CMOS outputs. In addition, the push-pull outputs of Port 2 can be turned off for open-drain operation.

5.1.1 Mode Registers

Each port has an associated Mode Register that determines the port's functions and allows dynamic change in port functions during program execution. Port and Mode Registers are mapped into the Standard Register File as shown in Figure 5-1.

Register	HEX	Identifier
Port 0-1 Mode	F8H	P01M
Port 3 Mode	F7H	P3M
Port 2 Mode	F6H	P2M
Port 3	03H	P3
Port 2	02H	P2
Port 1	01H	P1
Port 0	00H	P0

Figure 5-1. I/O Ports and Mode Registers

Because of their close association, Port and Mode Registers are treated like any other general-purpose register. There are no special instructions for port manipulation. Any instruction which addresses a register can address the ports. Data can be directly accessed in the Port Register, with no extra moves.

5.1.2 Input and Output Registers

Each bit of Ports 0, 1, and 2, have an input register, an output register, associated buffer, and control logic. Since there are separate input and output registers associated with each port, writing to bits defined as inputs stores the data in the output register. This data cannot be read as long as the bits are defined as inputs. However, if the bits are reconfigured as outputs, the data stored in the output register is reflected on the output pins and can then be read. This mechanism allows the user to initialize the outputs prior to driving their loads (Figure 5-2).

Since port inputs are asynchronous to the Z8 internal clock, a READ operation could occur during an input transition. In this case, the logic level might be uncertain (somewhere between a logic 1 and 0). To eliminate this meta-stable condition, the Z8 latches the input data two clock periods prior to the execution of the current instruction. The input register uses these two clock periods to stabilize to a legitimate logic level before the instruction reads the data.

Note: The following sections describe the generic function of the Z8 ports. Any additional features of the ports such as SPI, C/T, and Stop-Mode Recovery are covered in their own section.

5.2 PORT 0

This section deals with only the I/O operation of Port 0. The port's external memory interface operation is covered later in this manual. Figure 5-2 shows a block diagram of Port 0. This diagram also applies to Ports 1 and 2.

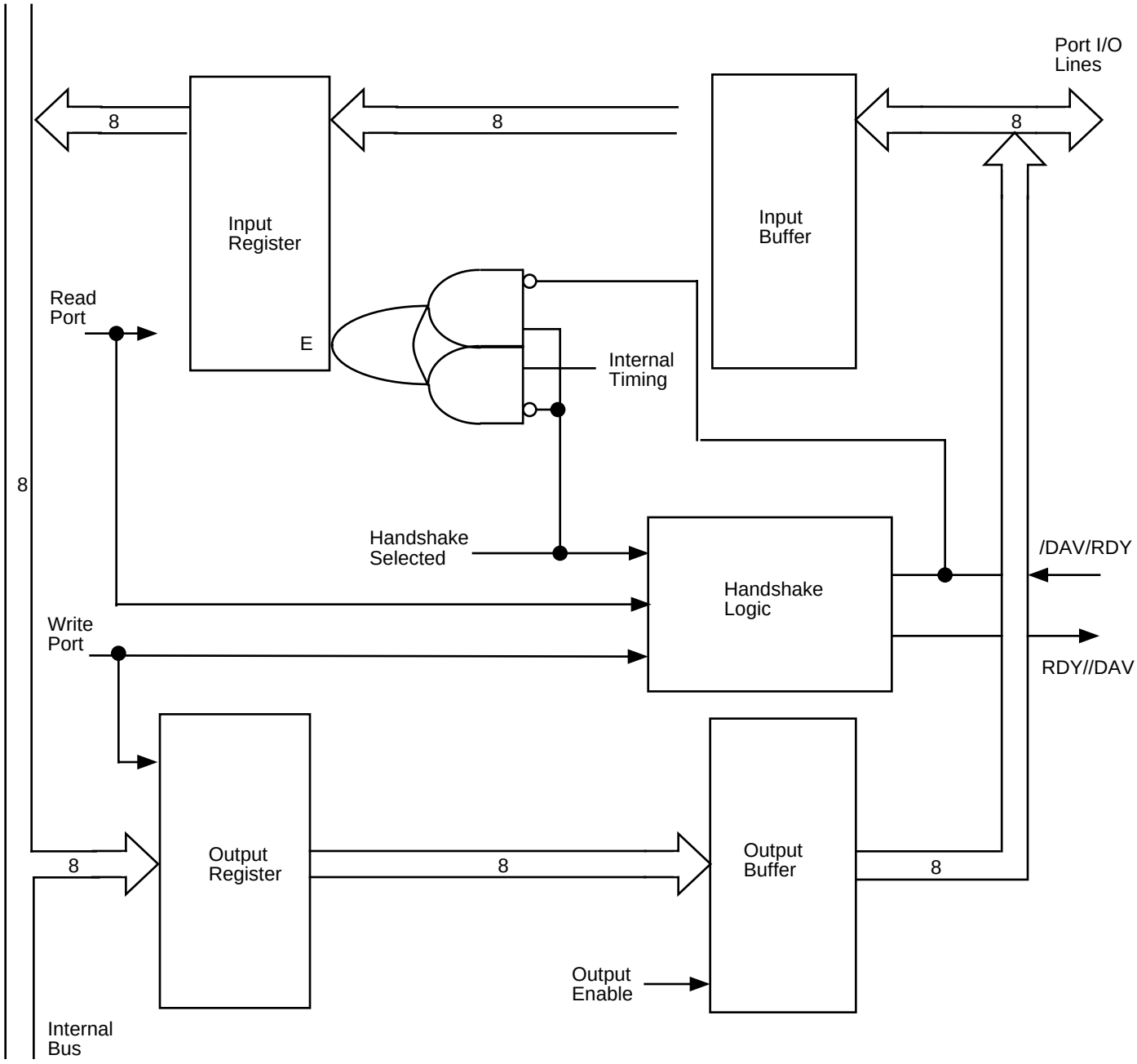


Figure 5-2. Ports 0, 1, 2 Generic Block Diagram

5.2.1 General I/O Mode

Port 0 can be an 8-bit, bidirectional, CMOS or TTL compatible I/O port. These eight I/O lines can be configured under software control as a nibble I/O port (P03-P00 input/output and P07-P04 input/output), or as an address port for interfacing external memory. The input buffers can be Schmitt-triggered, level shifted, or a single-trip point buffer and can be nibble programmed. Either nibble output can be globally programmed as push-pull or open-drain. Low EMI output buffers in some cases can be globally programmed by the software, as an OTP program option, or as a ROM mask option. In some, the Z8 has Auto Latches hardwired to the inputs. Please refer to specific product specifications for exact input/output buffer type features that are available (Figures 5-3 and 5-4).

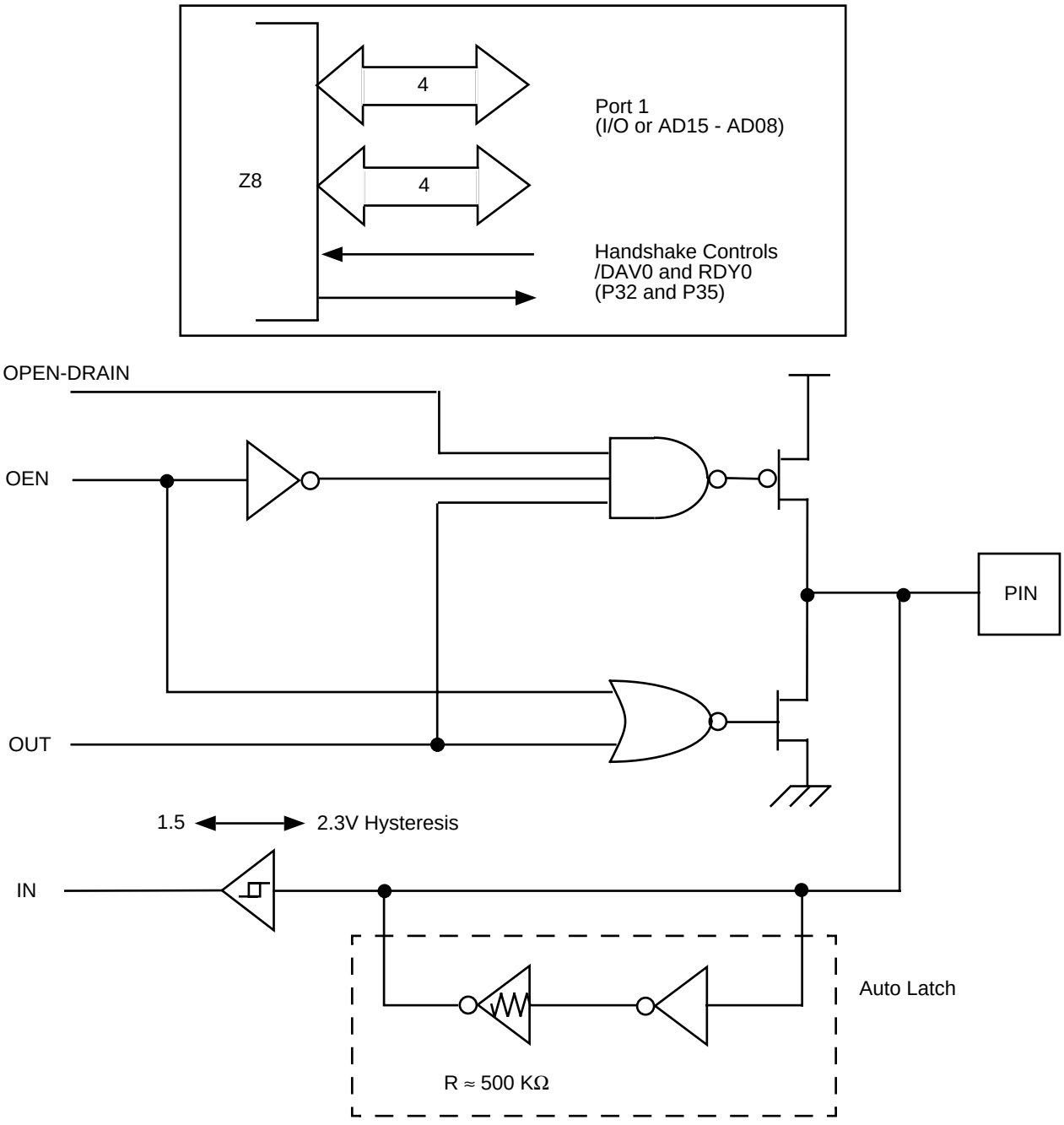


Figure 5-3. Port 0 Configuration with Open-Drain Capability, Auto Latch, and Schmitt-Trigger

5.2 PORT 0 (Continued)

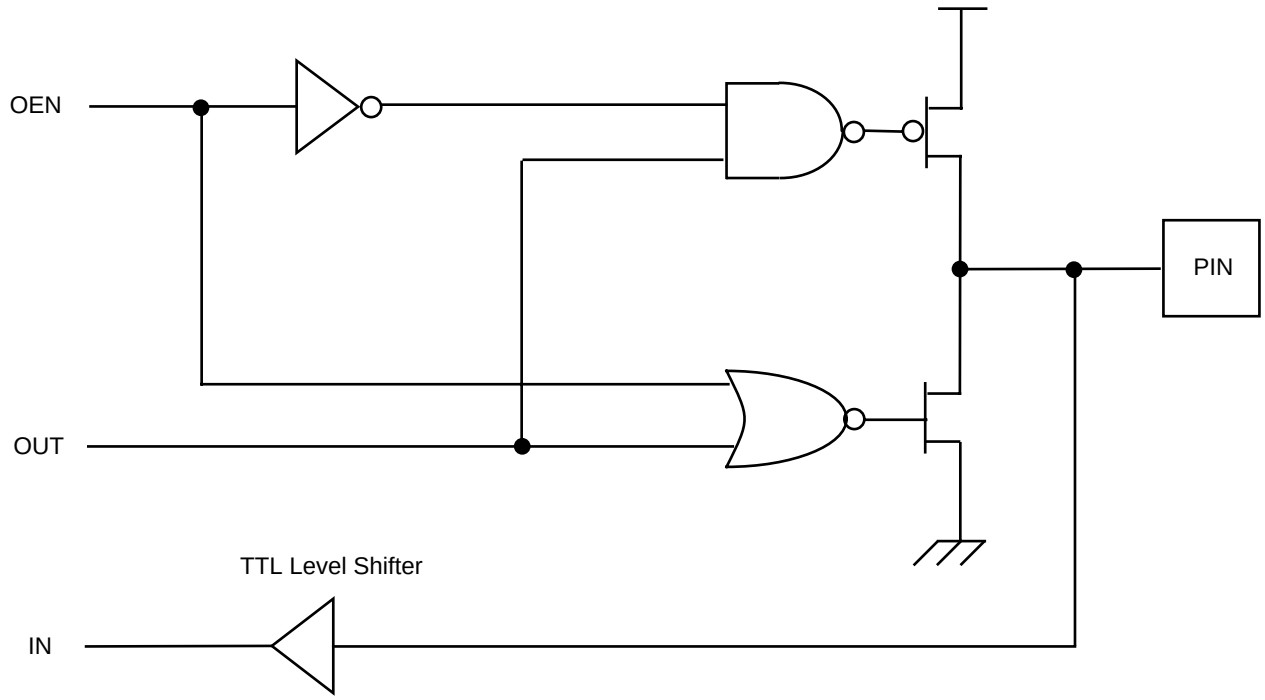


Figure 5-4. Port 0 Configuration with TTL Level Shifter

5.2.2 Read/Write Operations

In the nibble I/0 Mode, Port 0 is accessed as general-purpose register P0 (00H) with ERF Bank set to 0. The port is written by specifying P0 as an instruction's destination register. Writing to the port causes data to be stored in the port's output register.

The port is read by specifying P0 as the source register of an instruction. When an output nibble is read, data on the external pins is returned. Under normal loading conditions this is equivalent to reading the output register. However, for Port 0 outputs defined as open–drain, the data returned is the value forced on the output by the external system. This may not be the same as the data in the output register. Reading a nibble defined as input also returns data on the external pins. However, input bits under handshake control return data latched into the input register via the input strobe.

The Port 0–1 Mode resister bits D_1D_0 and D_7D_6 are used to configure Port 0 nibbles. The lower nibble ($P0_0$–$P0_3$) can be defined as inputs by setting bits D_1 to 0 and D_0 to 1, or as outputs by setting both D_1 and D_0 to 0. Likewise, the upper nibble ($P0_4$–$P0_7$) can be defined as inputs by setting bits D_7 to 0 and D_6 to 1, or as outputs by setting both D_6 and D_7 to 0 (Figure 5-5).

5.2.3 Handshake Operation

When used as an I/0 port, Port 0 can be placed under handshake control by programming the Port 3 Mode register bit D_2 to 1. In this configuration, handshake control lines are DAV_0 ($P3_2$) and RDY_0 ($P3_5$) when Port 0 is an input port, or RDY_0 ($P3_2$) and DAV_0 ($P3_5$) when Port 0 is an output port. (See Figure 5-6)

Handshake direction is determined by the configuration (input or output) assigned to the Port 0 upper nibble, $P0_4$–$P0_7$. The lower nibble must have the same I/0 configuration as the upper nibble to be under handshake control. Figure 5-3 illustrates the Port 0 upper and lower nibbles and the associated handshake lines of Port 3.

5.3 PORT 1

This section deals only with the I/0 operation. The port's external memory interface operation is discussed later in this manual. Figure 5-2 shows a block diagram of Port 1.

5.3.1 General I/O Mode

Port 1 can be an 8-bit, bidirectional, CMOS or TTL compatible port with multiplexed Address (A7–A0) and Data (D7–D0) ports. These eight I/O lines can be byte programmed as inputs or outputs or can be configured under software control as an Address/Data port for interfacing to external memory. The input buffers can be Schmitt-triggered, level- shifted, or a single-point buffer. In some cases, the output buffers can be globally programmed as either push-pull or open-drain. Low-EMI output buffers can be globally programmed by software, as an OTP program option, or as a ROM Mask Option. In some cases, the Z8can have auto latches hardwired to the inputs. Please refer to specific product specifications for exact input/output buffer-type features available (Figures 5-7 and 5-8).

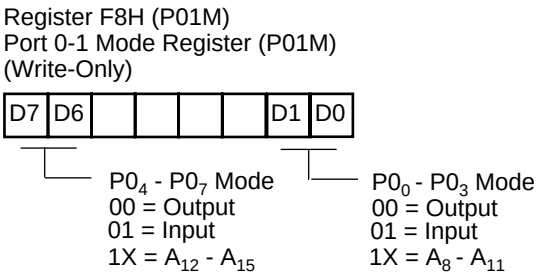


Figure 5-5. Port 0 I/O Operation

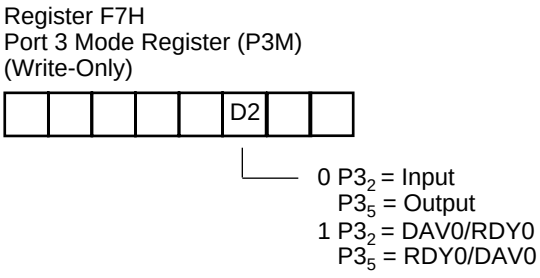


Figure 5-6. Port 0 Handshake Operation

5.3 PORT 1 (Continued)

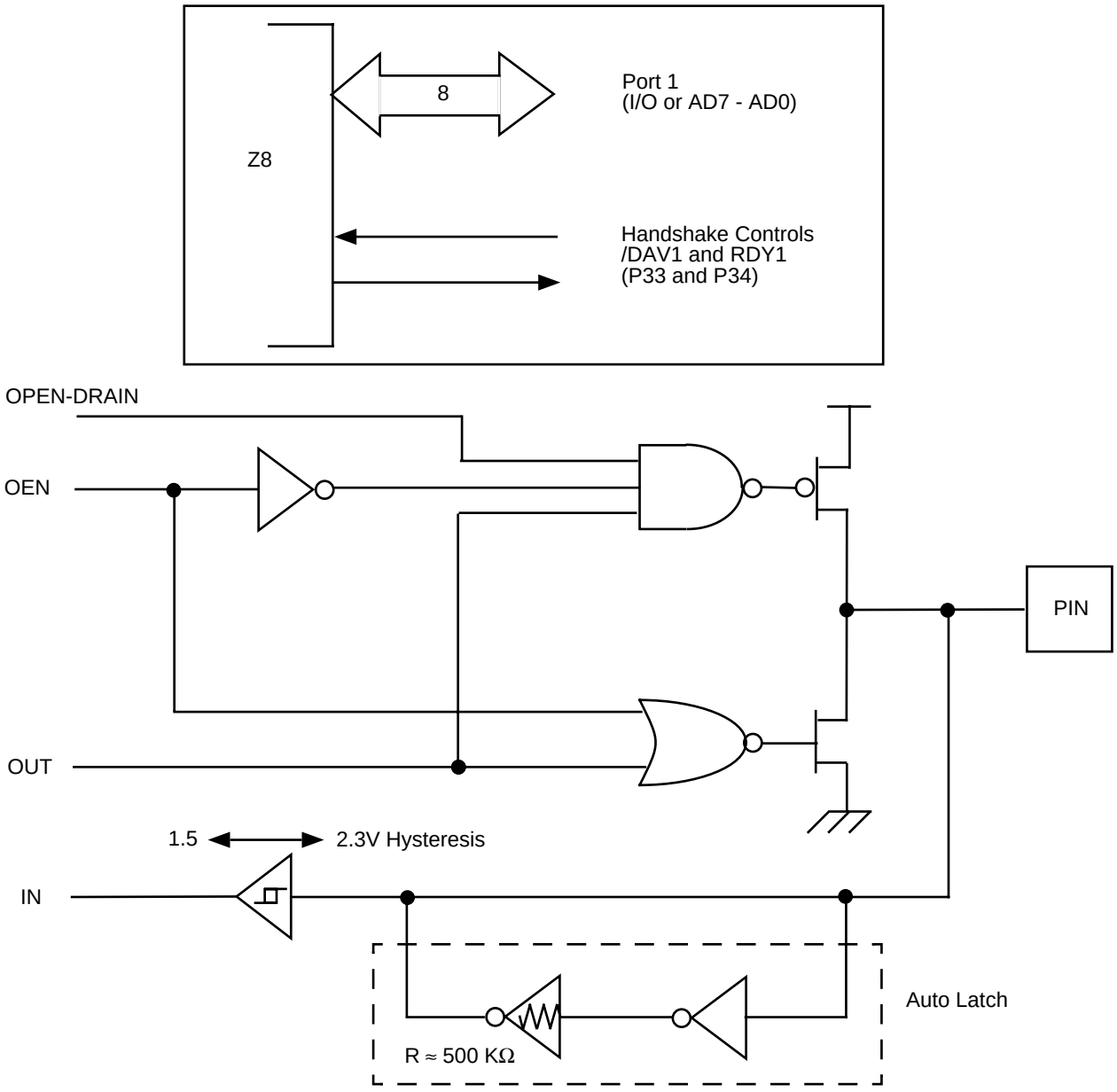


Figure 5-7. Port 1 Configuration with Open-Drain Capability, Auto Latch, and Schmitt-Trigger

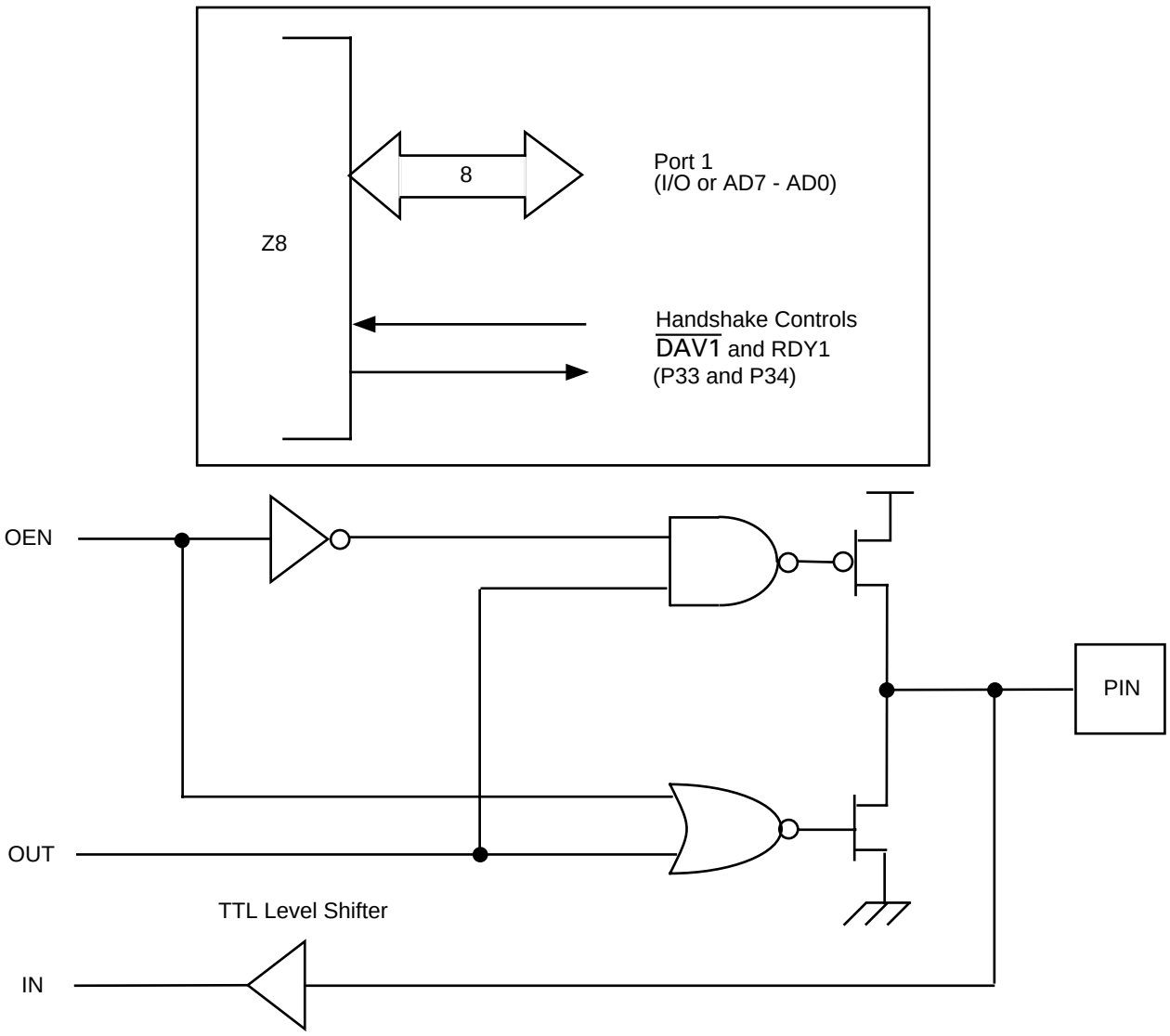


Figure 5-8. Port 1 Configuration with TTL Level Shifter

5.3.2 Read/Write Operations

In byte input or byte output mode, the port is accessed as General-Purpose Register P1 (01H). The port is written by specifying P1 as an instruction's destination register. Writing to the port causes data to be stored in the port's output register.

The port is read by specifying P1 as the source register of an instruction. When an output is read, data on the external pins is returned. Under normal loading conditions, this is equivalent to reading the output register. However, if Port 1 outputs are defined as open-drain, the data returned is the value forced on the output by the external system. This may not be the same as the data in the output register. When Port 1 is defined as an input, reading also returns data on the external pins. However, inputs under handshake control return data latched into the input register via the input strobe.

Using the Port 0-1 Mode Register, Port 1 is configured as an output port by setting bits D_4 and D_3 to 0, or as an input port by setting D_4 to 0 and D_3 to 1 (Figure 5-8).

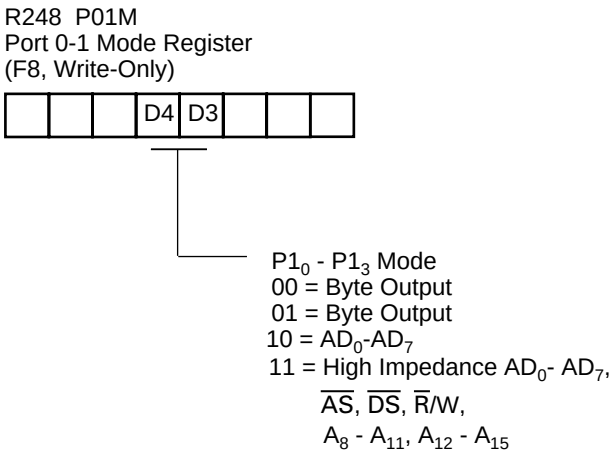


Figure 5-9. Port 1 I/O Operation

5.3.3 Handshake Operations

When used as an I/O port, Port 1 can be placed under handshake control by programming the Port 3 Mode register bits D_4 and D_3 both to 1. In this configuration, handshake control lines are DAV_1 ($P3_3$) and RDY1 ($P3_4$) when Port 1 is an input port, or RDY*1* ($P3_3$) and DAV1 ($P3_4$) when Port 1 is an output port. See Figures 5-8 and 5-10.

Handshake direction is determined by the configuration (input and output) assigned to Port 1. For example, if Port 1 is an output port then handshake is defined as output.

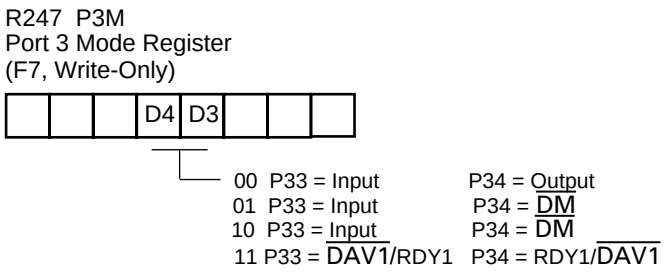


Figure 5-10. Handshake Operation

5.4 PORT 2

Port 2 is a general-purpose port. Figure 5-2 shows a block diagram of Port 2. Each of its lines can be independently programmed as input or output via the Port 2 Mode Register (F6H) as seen in Figure 5-11. A bit set to a 1 in P2M configures the corresponding bit in Port 2 as an input, while a bit set to 0 configures an output line.

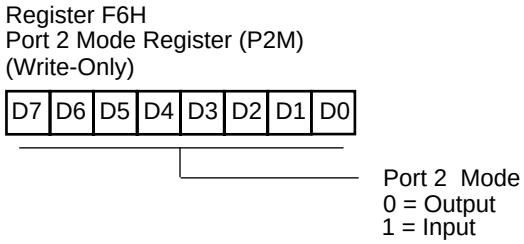


Figure 5-11. Port 2 I/O Mode Configuration

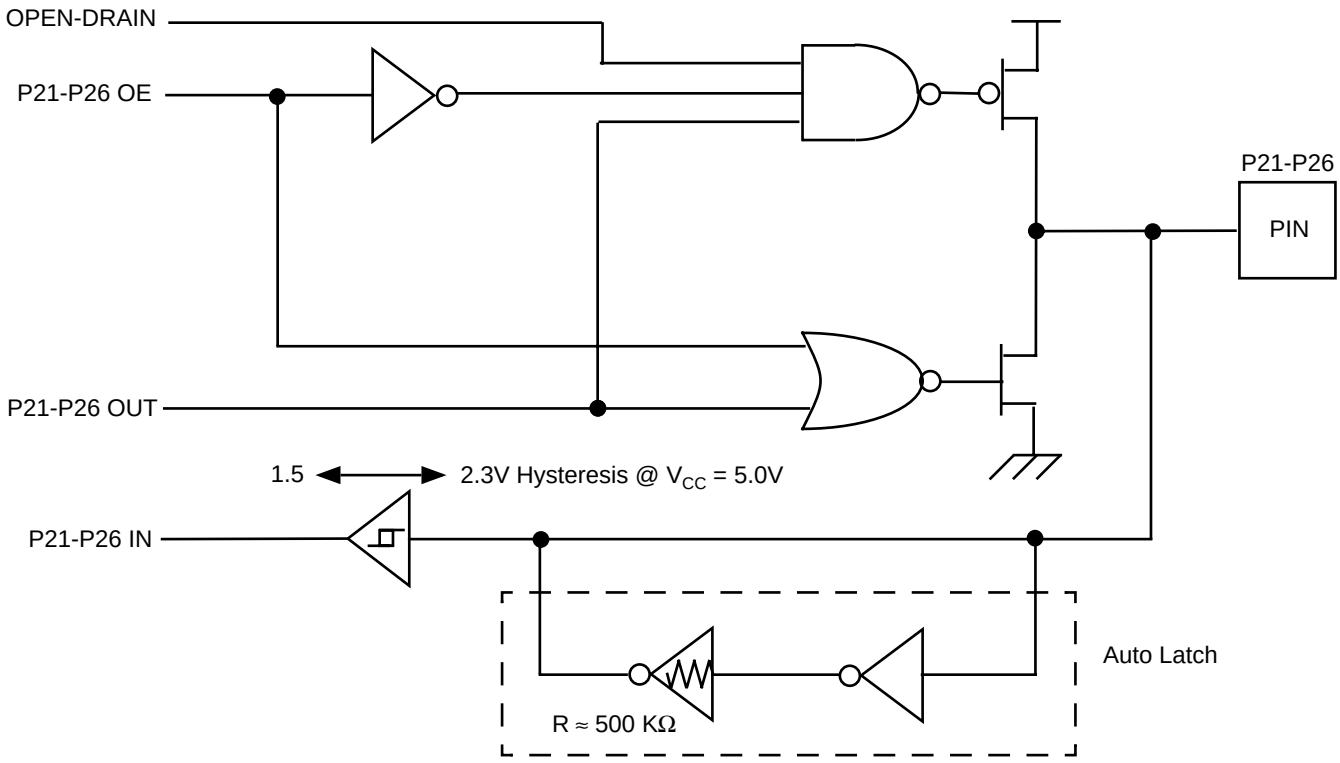


Figure 5-12. Port 2 Configuration with Open-Drain Capability, Auto Latch, and Schmitt-Trigger

5.4 PORT 2 (Continued)

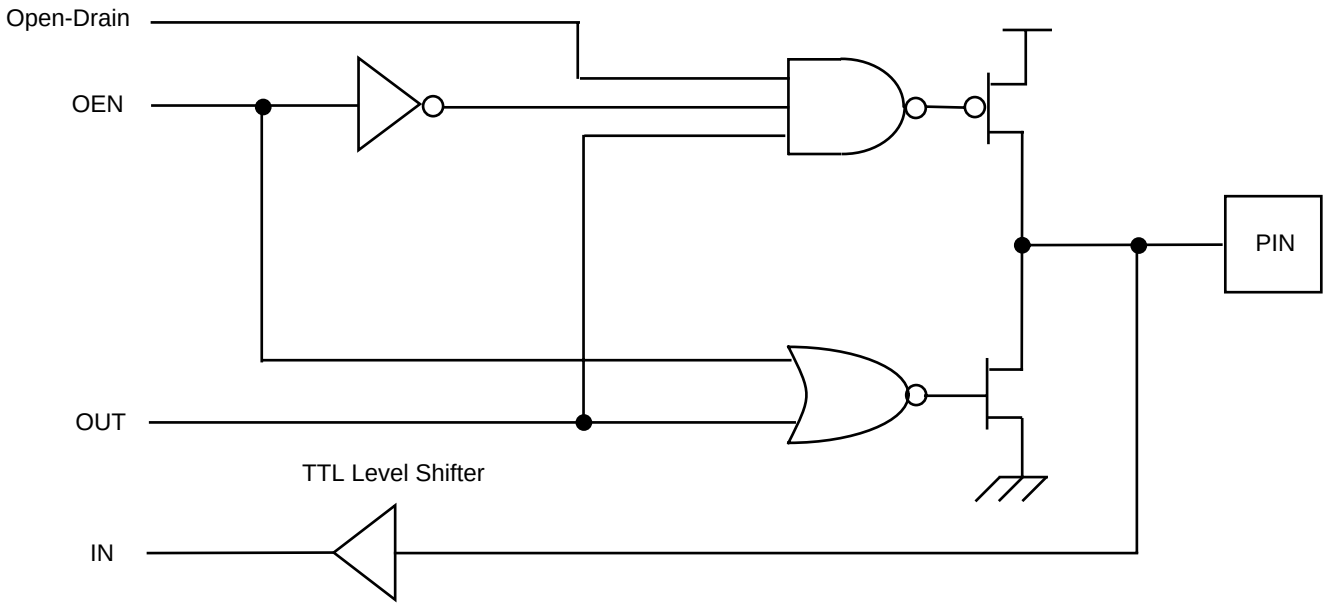


Figure 5-13. Port 2 Configuration with TTL Level Shifter

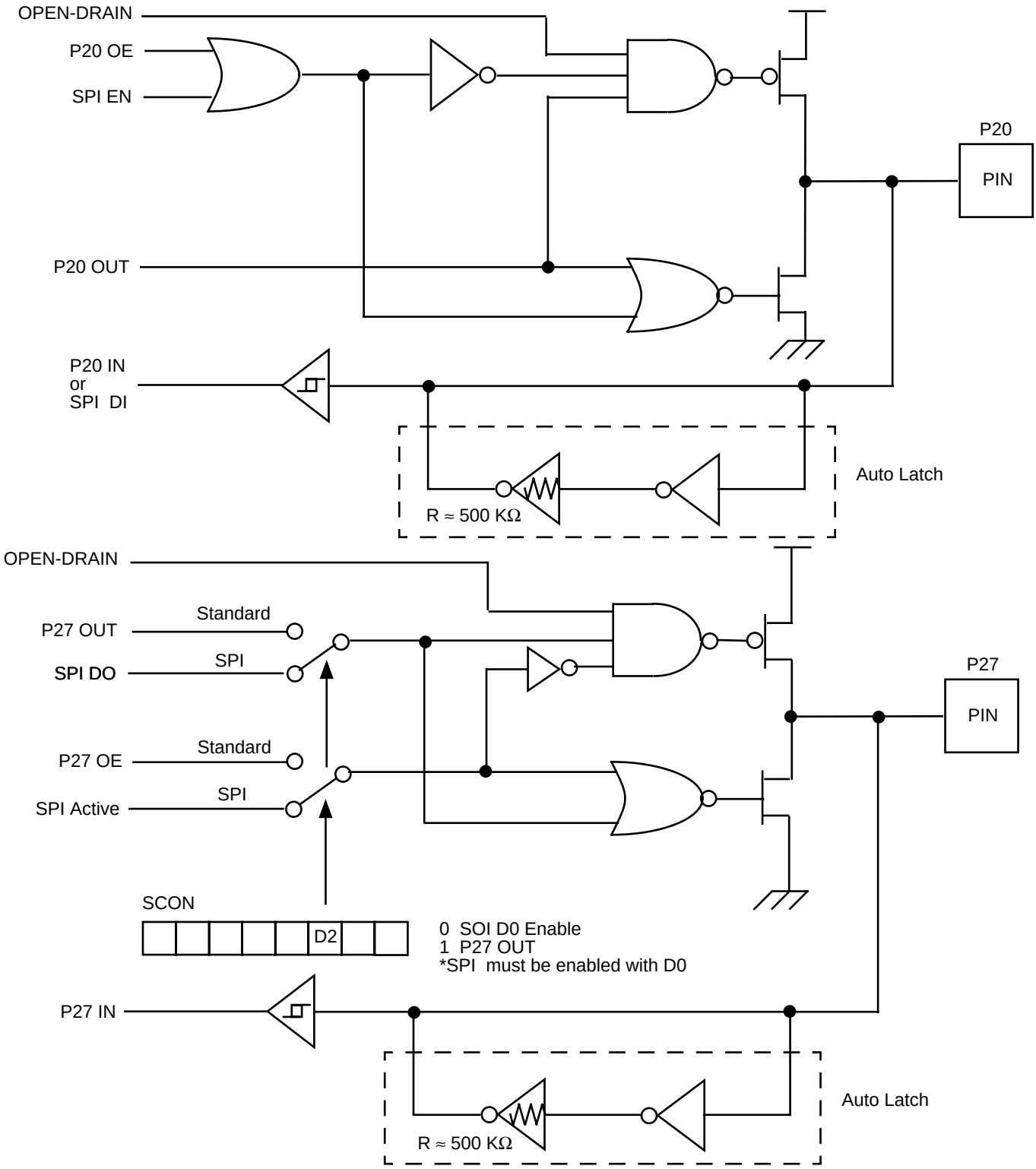


Figure 5-14. Port 2 Configuration with Open-Drain Capability, Auto Latch, Schmitt-Trigger and SPI

5.4.2 Read/Write Operations

Port 2 is accessed as General-Purpose Register P2 (02H). Port 2 is written by specifying P2 as an instruction's destination register. Writing to Port 2 causes data to be stored in the output register of Port 2, and reflected externally on any bit configured as an output.

Port 2 is read by specifying P2 as the source register of an instruction. When an output bit is read, data on the external pin is returned. Under normal loading conditions, 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 is the value forced on the output pin by the external system. This may not be the same as the data in the output register. Reading input bits of Port 2 also returns data on the external pins. However, inputs under handshake control return data latched into the input register via the input strobe.

5.4.3 Handshake Operation

Port 2 can be placed under handshake control by programming bit 6 in the Port 3 Mode Register (Figure 5-15). In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines $\overline{DAV2}$ and RDY2 for input handshake, or RDY2 and $\overline{DAV2}$ for output handshake.

Handshake direction is determined by the configuration (input or output) assigned to bit 7 of Port 2. Only those bits with the same configuration as P27 will be under handshake control. Figure 5-16 illustrates bit lines of Port 2 and the associated handshake lines of Port 3.

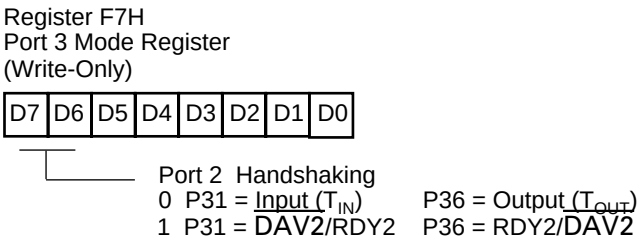


Figure 5-15. Port 2 Handshake Configuration

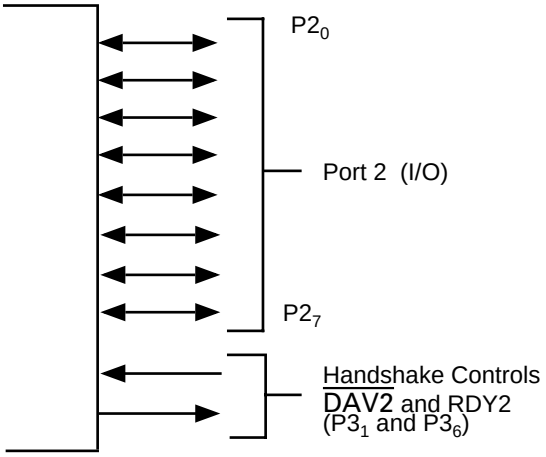


Figure 5-16. Port 2 Handshaking

5.5 PORT 3

5.5.1 General Port I/O

Port 3 differs structurally from Port 0, 1, and 2. Port 3 lines are fixed as four inputs (P33–P30) and four outputs (P37–P34) Port 3 does not have an input and output register for each bit. Instead, all the input lines have one input register, and all the output lines have an output register. Port 3 can be a CMOS- or TTL- compatible I/O port. Under software control, the lines can be configured as special control lines for handshake, comparator inputs, SPI control, external memory status, or I/O lines for the on-board serial and timer facilities. Figure 5-17 is a generic block diagram of Port 3.

The inputs can be Schmitt-triggered, level-shifted, or single-trip point buffered. In some cases, the Z8 may have auto latches hardwired on certain Port 3 inputs and Low-EMI capabilities on the outputs. Please refer to specific product specifications for exact input/output buffer type features. Please refer to the section on counter/timers, Stop-Mode Recovery, serial I/O, comparators, and interrupts for more information on the relationships of Port 3 to that feature.

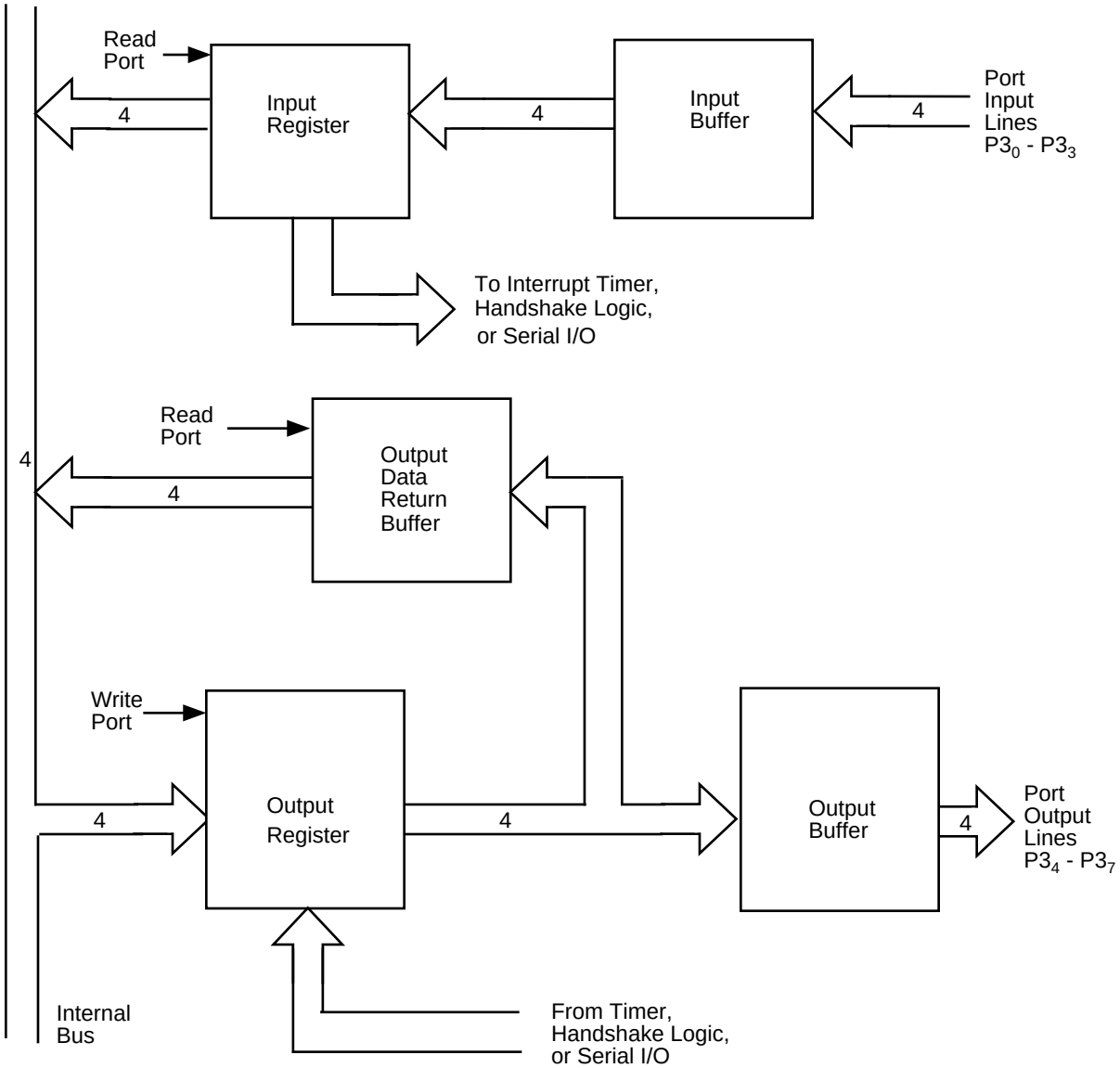


Figure 5-17. Port 3 Block Diagram

5.5 PORT 3 (Continued)

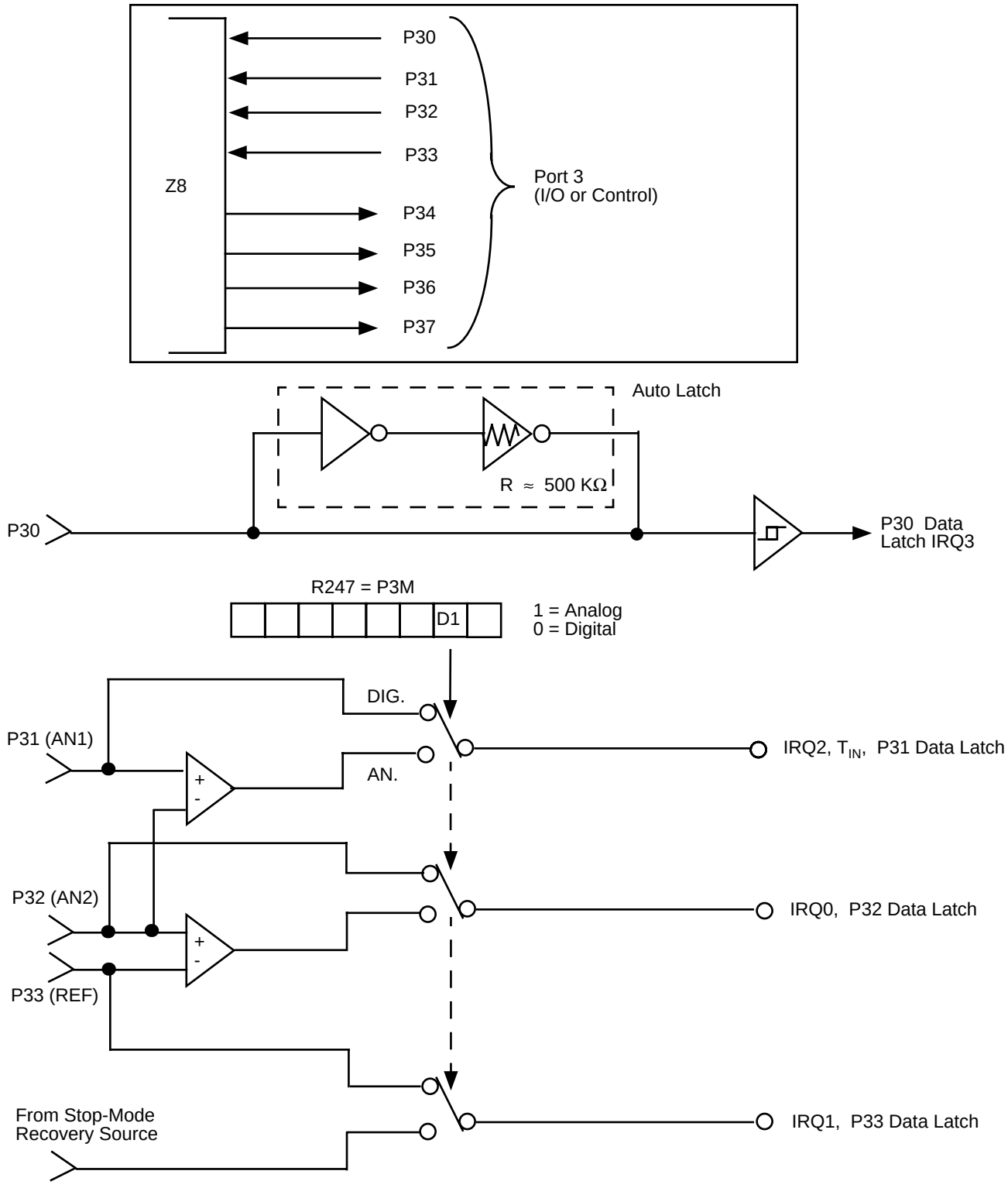


Figure 5-18. Port 3 Configuration with Comparator, Auto Latch, and Schmitt-Trigger

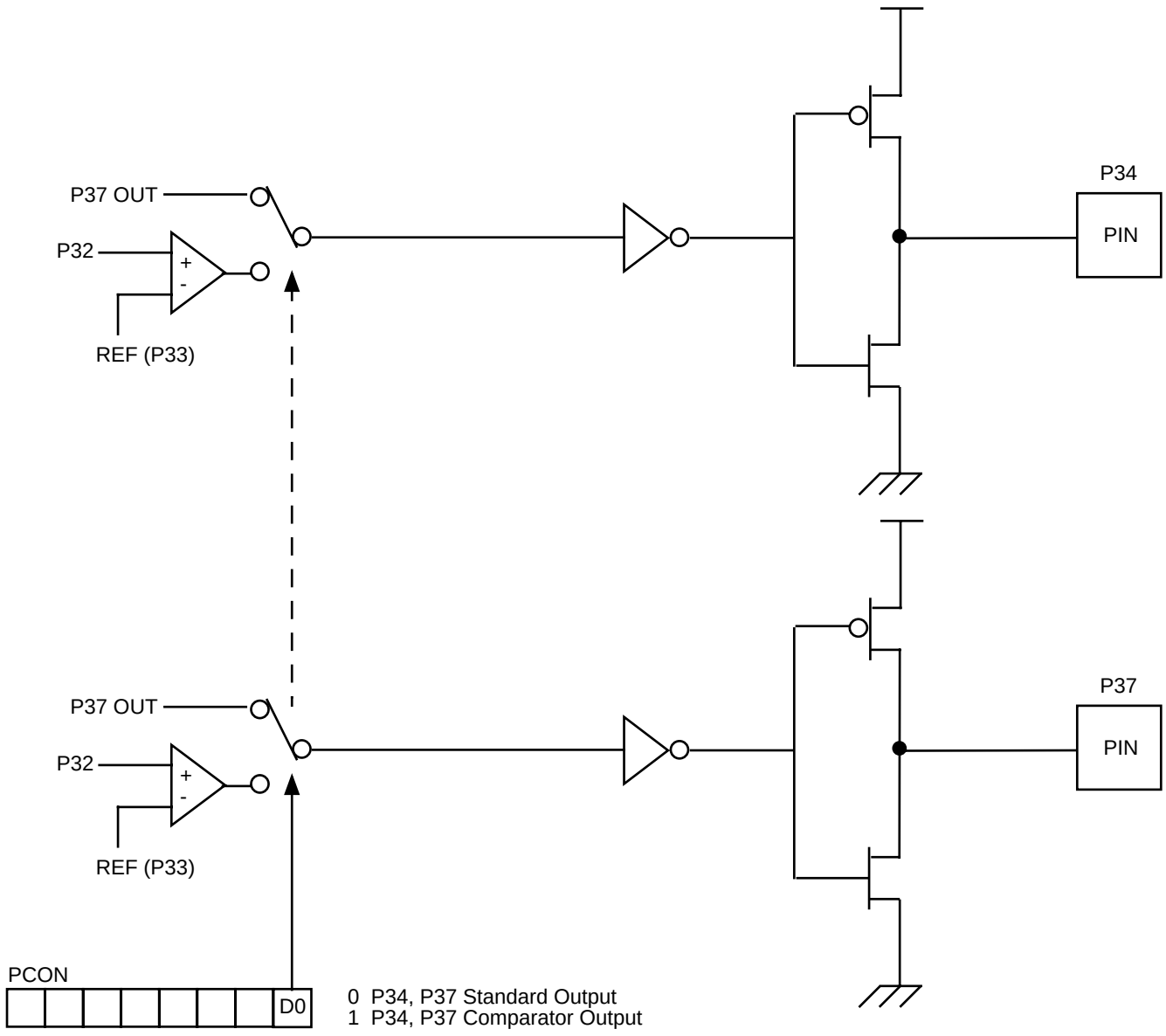


Figure 5-19. Port 3 Configuration with Comparator

5.5 PORT 3 (Continued)

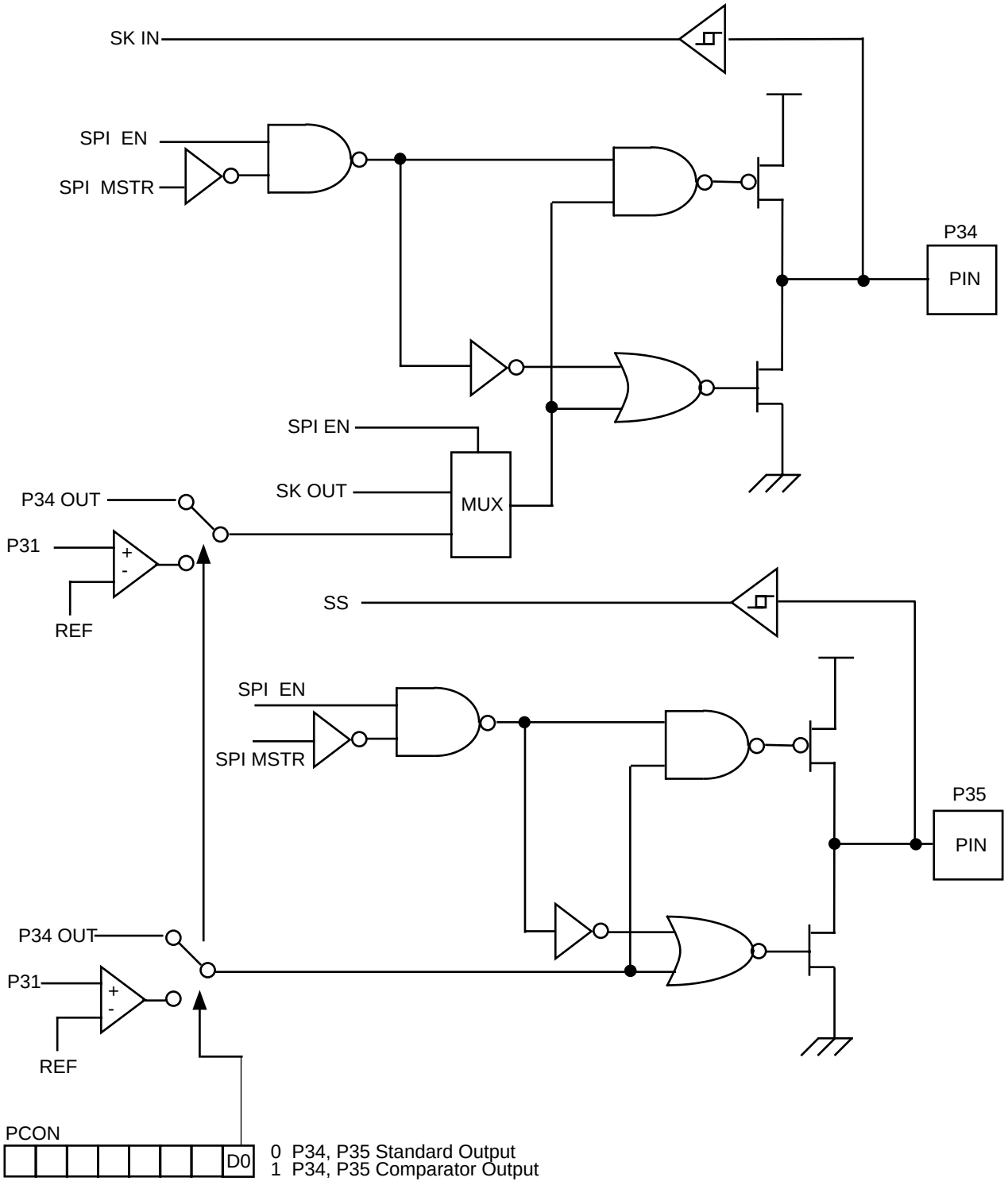


Figure 5-20. Port 3 Configuration with SPI and Comparator Outputs Using P34 and P35

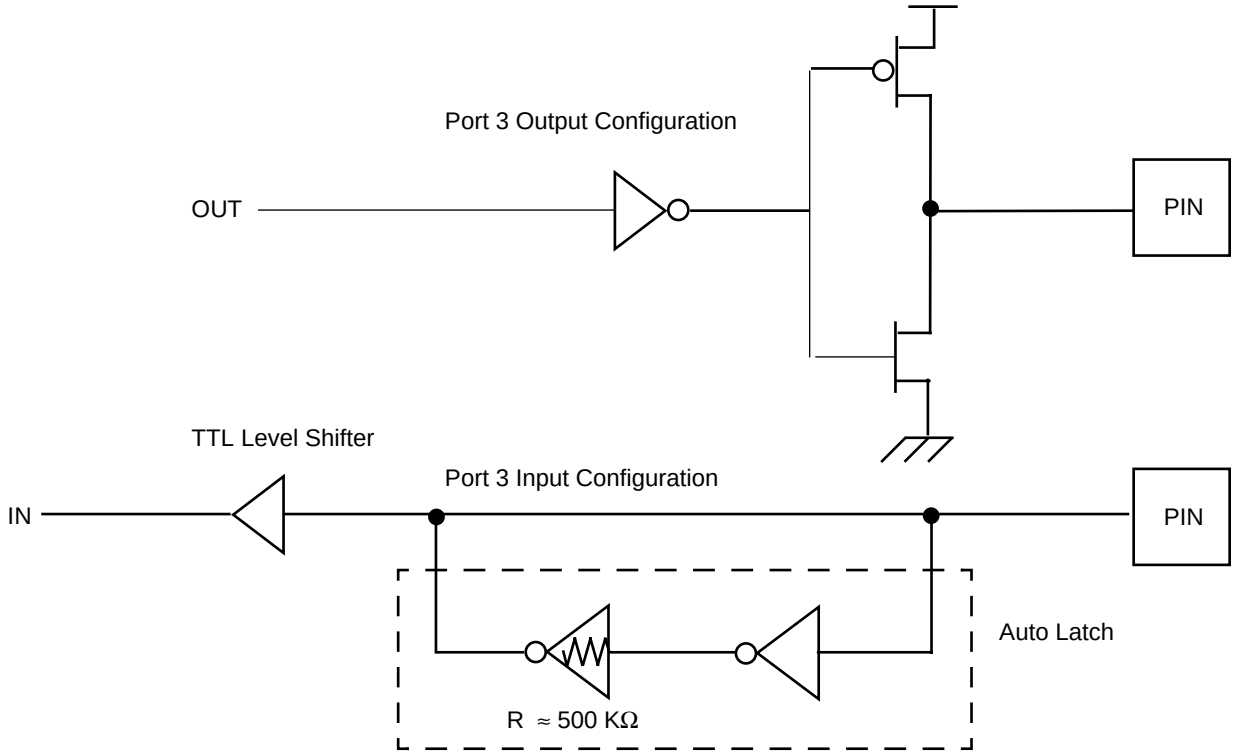


Figure 5-21. Port 3 Configuration with TTL Level Shifter and Auto Latch

5.5.2 Read/Write Operations

Port 3 is accessed as a General-Purpose Register P3 (03H). Port 3 is written by specifying P3 as an instruction's destination register. However, Port 3 outputs cannot be written to if they are used for special functions. When writing to Port 3, data is stored in the output register.

Port 3 is read by specifying P3 as the source register of an instruction. When reading from Port 3, the data returned is both the data on the input pins and in the output register.

5.5.3 Special Functions

Special functions for Port 3 are defined by programming the Port 3 Mode Register. By writing 0s in bit 6 through bit 1, lines P37–P30 are configured as input/output pairs (Figure 5-22). Table 5-1 shows available functions for Port 3. The special functions indicated in the figure are discussed in detail in their corresponding sections in this manual.

Port 3 input lines P33–P30 always function as interrupt requests regardless of the configuration specified in the Port 3 Mode Register.

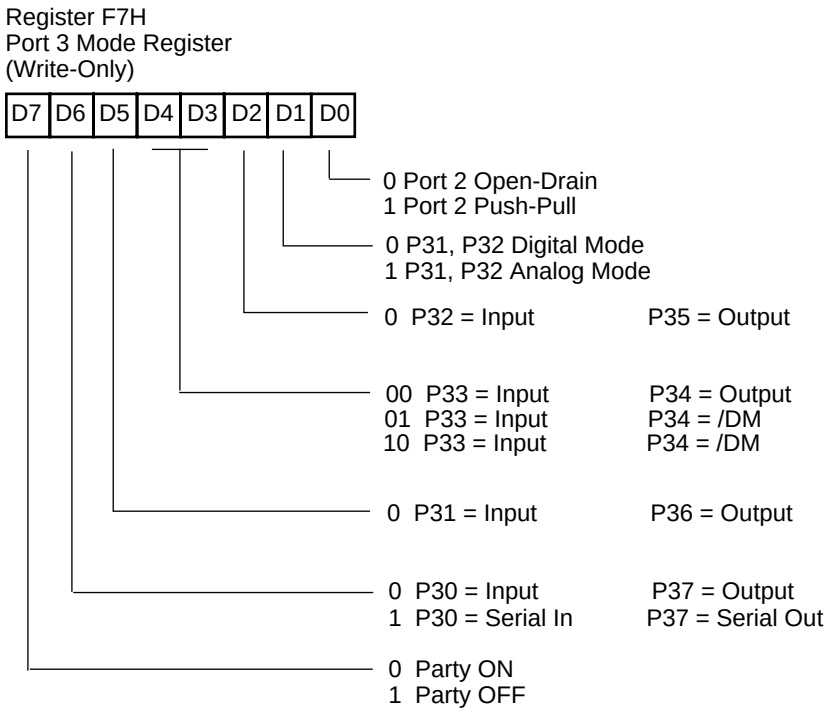


Figure 5-22. Port 3 Mode Register Configuration

Table 5-1. Port 3 Line Functions

Function	Line	Signal
Inputs	P30	Input
	P31	Input
	P32	Input
	P33	Input
Outputs	P34	Output
	P35	Output
	P36	Output
	P37	Output
Port 0 Handshake Input	P32	$\overline{DAV0}$/RDY0
Port 1 Handshake Input	P33	$\overline{DAV1}$/RDY1
Port 2 Handshake Input	P31	$\overline{DAV2}$/RDY2
Port 0 Handshake Output	P35	RDY0/$\overline{DAV0}$
Port 1 Handshake Output	P34	RDY1/$\overline{DAV1}$
Port 2 Handshake Output	P36	RDY2/$\overline{DAV2}$
Analog Comparator Input	P31	AN1
	P32	AN2
	P33	REF
Analog Comparator Output	P34	AN1-OUT
	P35	AN2-OUT
	P37	AN2-OUT

Table 5-1. Port 3 Line Functions

Function	Line	Signal
Interrupt Requests	P30	IRQ3
	P31	IRQ2
	P32	IRQ0
	P33	IRQ1
Serial Input (UART)	P30	DI
Serial Output (UART)	P37	DO
SPI Slave Select	P35	SS
SPI Clock	P34	SK
Counter/Timer	P31	T_{IN}
	P36	T_{OUT}
External Memory Status	P34	$\overline{DM}$

5.6 PORT HANDSHAKE

When Ports 0, 1, and 2 are configured for handshake operation, a pair of lines from Port 3 are used for handshake controls. The handshake controls are interlocked to properly time asynchronous data transfers between the Z8® and a peripheral. One control line (/DAV) functions as a strobe from the sender to indicate to the receiver that data is available. The second control line (RDY) acknowledges receipt of the sender's data, and indicates when the receiver is ready to accept another data transfer.

In the input mode, data is latched into the Port's input register by the first /DAV signal, and is protected from being overwritten if additional pulses occur on the /DAV line. This overwrite protection is maintained until the port data is read. In the output mode, data written to the port is not protected and can be overwritten by the Z8 during the handshake sequence. To avoid losing data, the software must not overwrite the port until the corresponding interrupt request indicates that the external device has latched the data.

The software can always read Port 3 output and input handshake lines, but cannot write to the output handshake line.

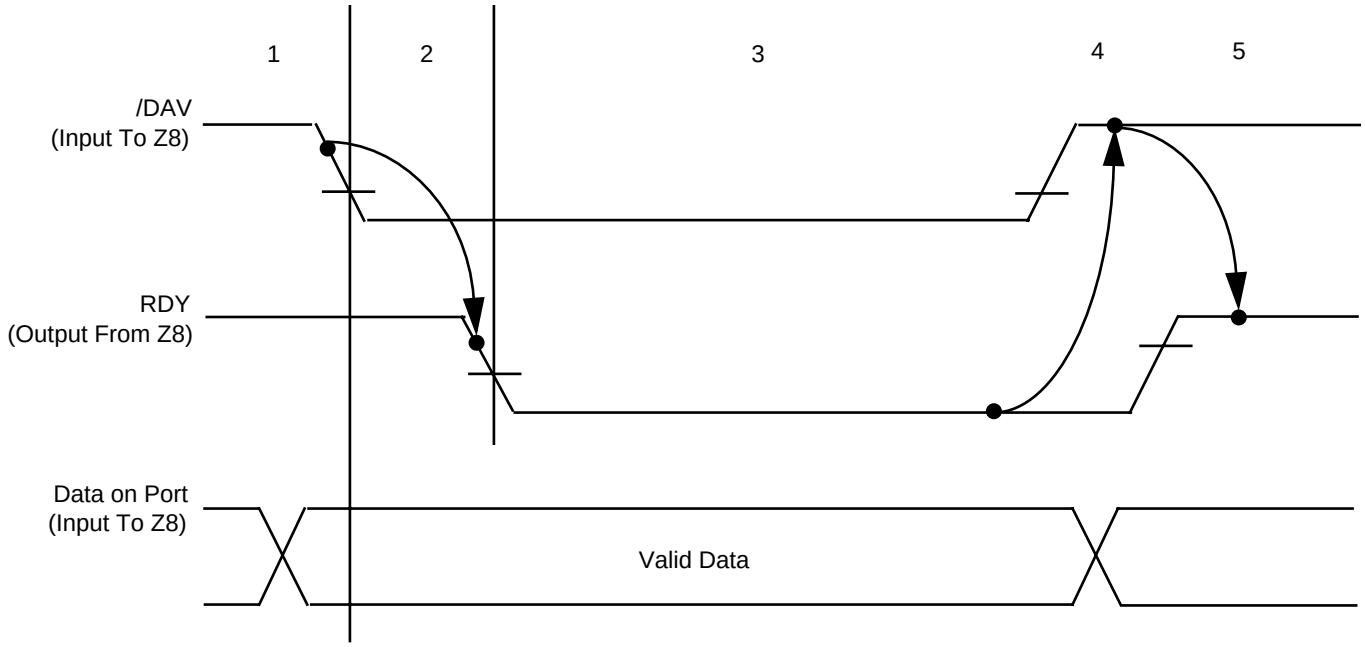
The following is the recommended setup sequence when configuring a Port for handshake operation for the first time after a reset:

- Load P01M or P2M to configure the port for input/output.
- Load P3 to set the Output Handshake bit to a logic 1.
- Load P3M to select the Handshake Mode for the port.

Once a data transfer begins, the configuration of the handshake lines should not be changed until the handshake is completed.

Figures 5-23 and 5-24 show detailed operation for the handshake sequence.

5.6 PORT HANDSHAKE (Continued)



State 1. Port 3 output is High, indicating that the I/O device is ready to accept data.

State 2. The I/O device puts data on the port and then activates the $\overline{DAV}$ input. This causes the data to be latched into the port input register and generates an interrupt request.

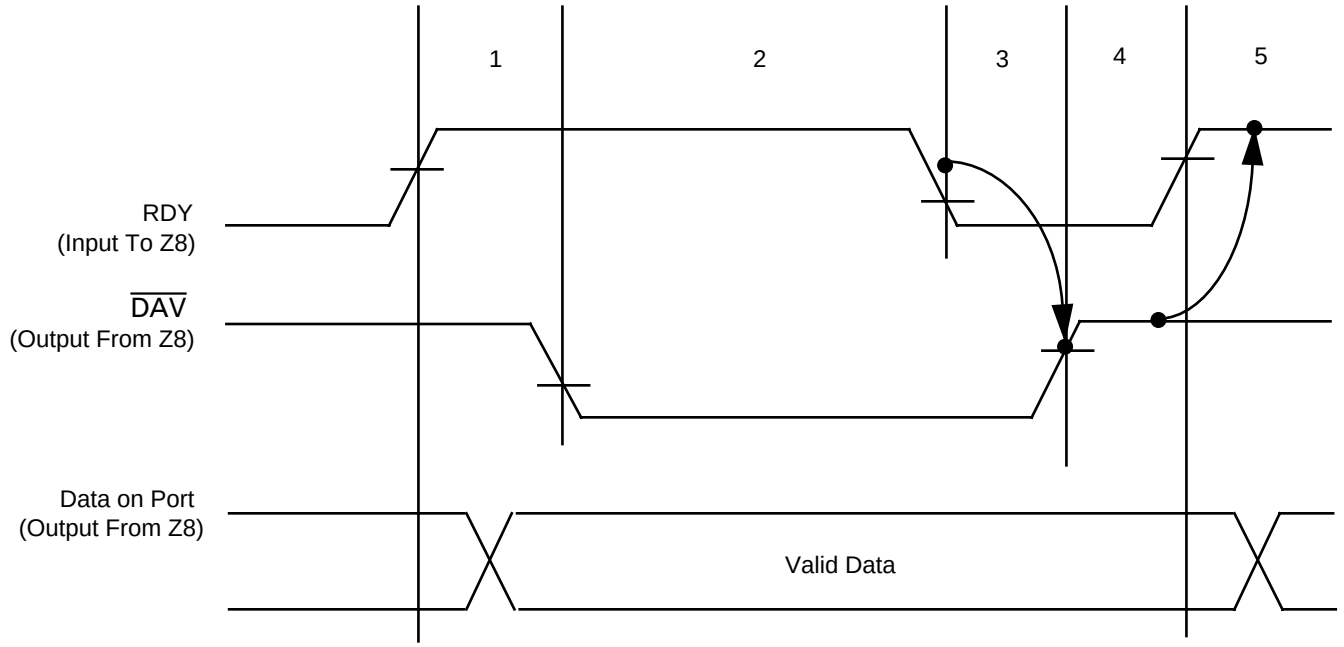
State 3. The Z8 forces the Ready (RDY) output Low, signaling to the I/O device that the data has been latched.

State 4. The I/O device returns the $\overline{DAV}$ line High in response to RDY going Low.

State 5. The Z8 RR software must respond to the interrupt request and read the contents of the port in order for the handshake sequence to be completed. The RDY line goes High if and only if the port has been read and $\overline{DAV}$ is High. This returns the interface to its initial state.

Figure 5-23. Z8 Input Handshake

5.6 PORT HANDSHAKE (Continued)



State 1. RDY input is High indicating that the I/O device is ready to accept data.

State 2. The Z8 Writes to the port register to initiate a data transfer. Writing to the port outputs new data and forces $\overline{DAV}$ Low if and only if RDY is High.

State 3. The I/O device forces RDY Low after latching the data. RDY Low causes an interrupt request to be generated. The Z8 can write new data responses to RDY going Low; however, the data is not output until State 5.

State 4. The $\overline{DAV}$ output from the Z8 is driven High in response to RDY going Low.

State 5. The $\overline{DAV}$ goes High, the I/O device is free to raise RDY High thus returning the interface to its initial state.

Figure 5-24. Z8 Output Handshake

In applications requiring a strobed signal instead of the interlocked handshake, the Z8 MCU can satisfy this requirement as follows:

- In the Strobed Input mode, data can be latched in the Port input register using the $\overline{DAV}$ input. The data transfer rate must allow enough time for the software to read the Port before strobing in the next character. The RDY output is ignored.
- In the Strobed Output Mode, the RDY input should be tied to the $\overline{DAV}$ output.

Figures 5-25 and 5-26 illustrate the strobed handshake connections.

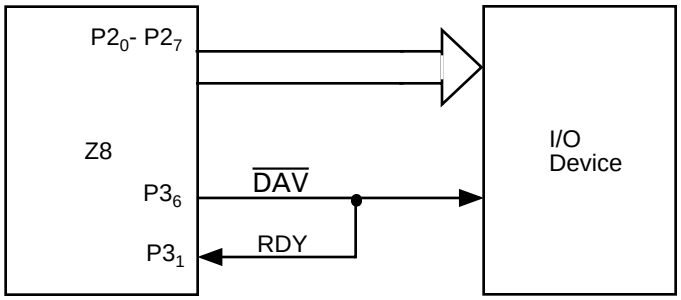


Figure 5-25. Output Strobed Handshake on Port 2

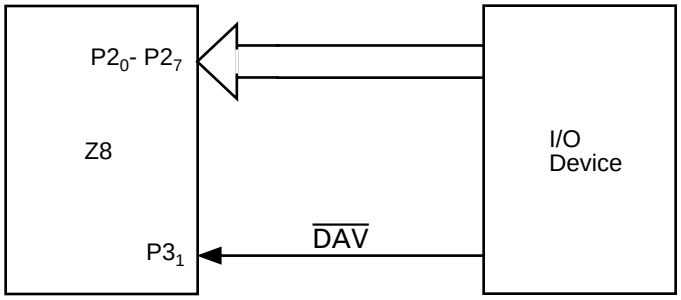


Figure 5-26. Input Strobed Handshake on Port 2

5.7 I/O PORT RESET CONDITIONS

5.7.1 Full Reset

After a hardware reset, Watch-Dog Timer (WDT) reset, or a Power-On Reset (POR), Port Mode Registers P01M, P2M, and P3M are set as shown in Figures 5-27 through 5-22. Port 2 is configured for input operation on all bits and is set for open-drain (Figure 5-29). If push-pull outputs are desired for Port 2 outputs, remember to configure them using P3M. Please note that a WDT time-out from Stop-Mode Recovery does not do a full reset. Certain registers that are not reset after Stop-Mode Recovery will not be reset.

For the condition of the Ports after Stop-Mode Recovery, please refer to specific device product specifications. In some cases, the Z8 has the P01M, P2M, and P3M control register set back to the default condition after reset while others do not.

All special I/O functions of Port 3 are inactive, with P33–P30 set as inputs and P37–P34 set as outputs (Figure 5-29).

Note: Because the types and amounts of I/O vary greatly among the Z8 family devices, the user is advised to review the selected device's product specifications for the register default state after reset.

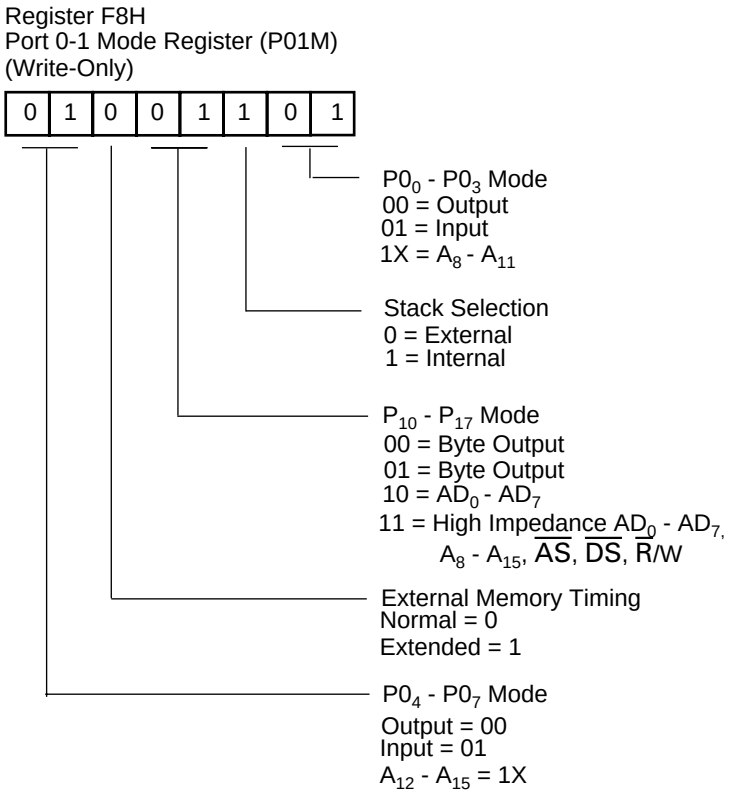


Figure 5-27. Port 0/1 Reset

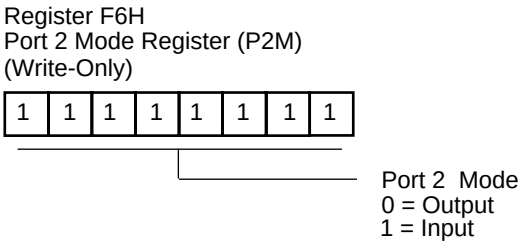


Figure 5-28. Port 2 Reset

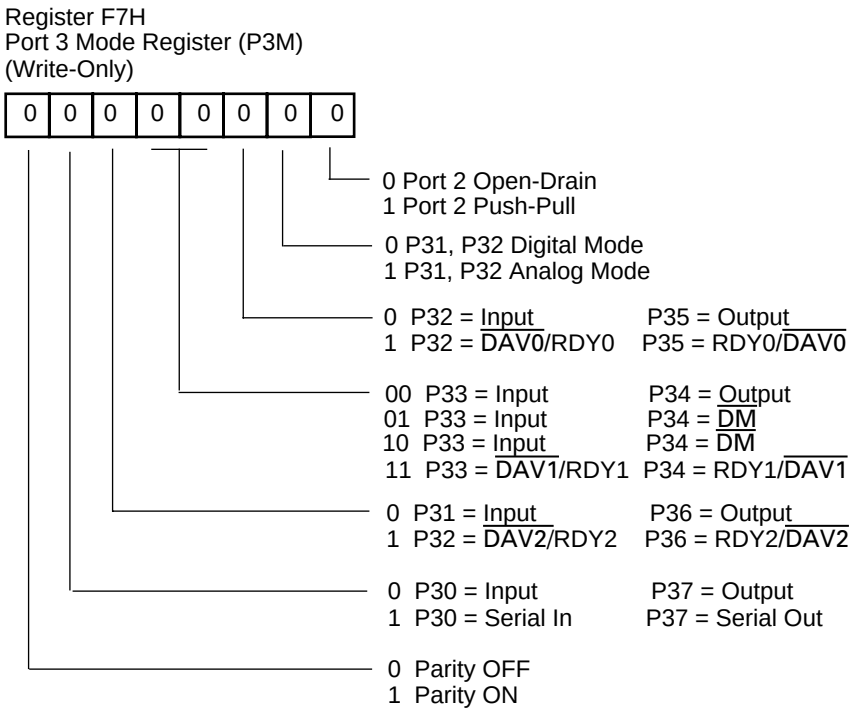


Figure 5-29. Port 3 Mode Reset

5.8 ANALOG COMPARATORS (Continued)

5.8 ANALOG COMPARATORS

Select Z8 devices include two independent on-chip analog comparators. See the device product specification for feature availability and use. Port 3, Pins P31 and P32 each have a comparator front end. The comparator reference voltage, pin P33, is common to both comparators. In Analog Mode, the P31 and P32 are the positive inputs to the comparators and P33 is the reference voltage supplied to both comparators. In Digital Mode, pin P33 can be used as a P33 register input or IRQ1 source. P34, P35, or P37 may output the comparator outputs by software-programming the PCON Register bit D0 to 1.

5.8.1 Comparator Description

Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming the Port 3 Mode Register (P3M bit 1). For interrupt functions during analog mode, P31 and P32 can be programmable as rising, falling, or both edge triggered interrupts (IRQ register bits 6 and bit 7).

Note: P33 cannot generate an external interrupt while in this mode. P33 can only generate interrupts in the Digital Mode.

Note: Port 3 inputs must be in digital mode if Port 3 is a Stop-Mode Recovery source. The analog comparator is disabled in STOP mode.

P31 can be used as T_{IN} in Analog or Digital Modes, but it must be referenced to P33, when in Analog Mode.

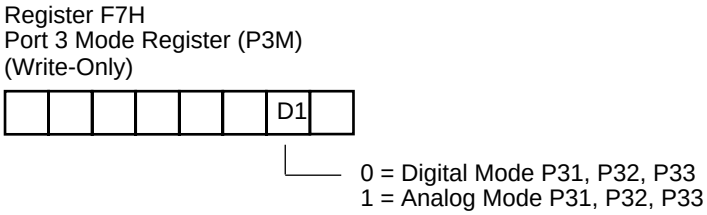


Figure 5-30. Port 3 Input Analog Selection

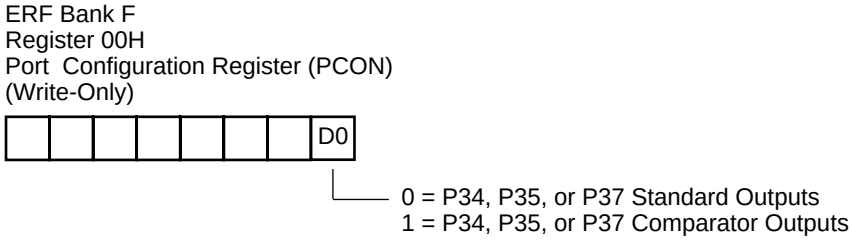


Figure 5-31. Port 3 Comparator Output Selection

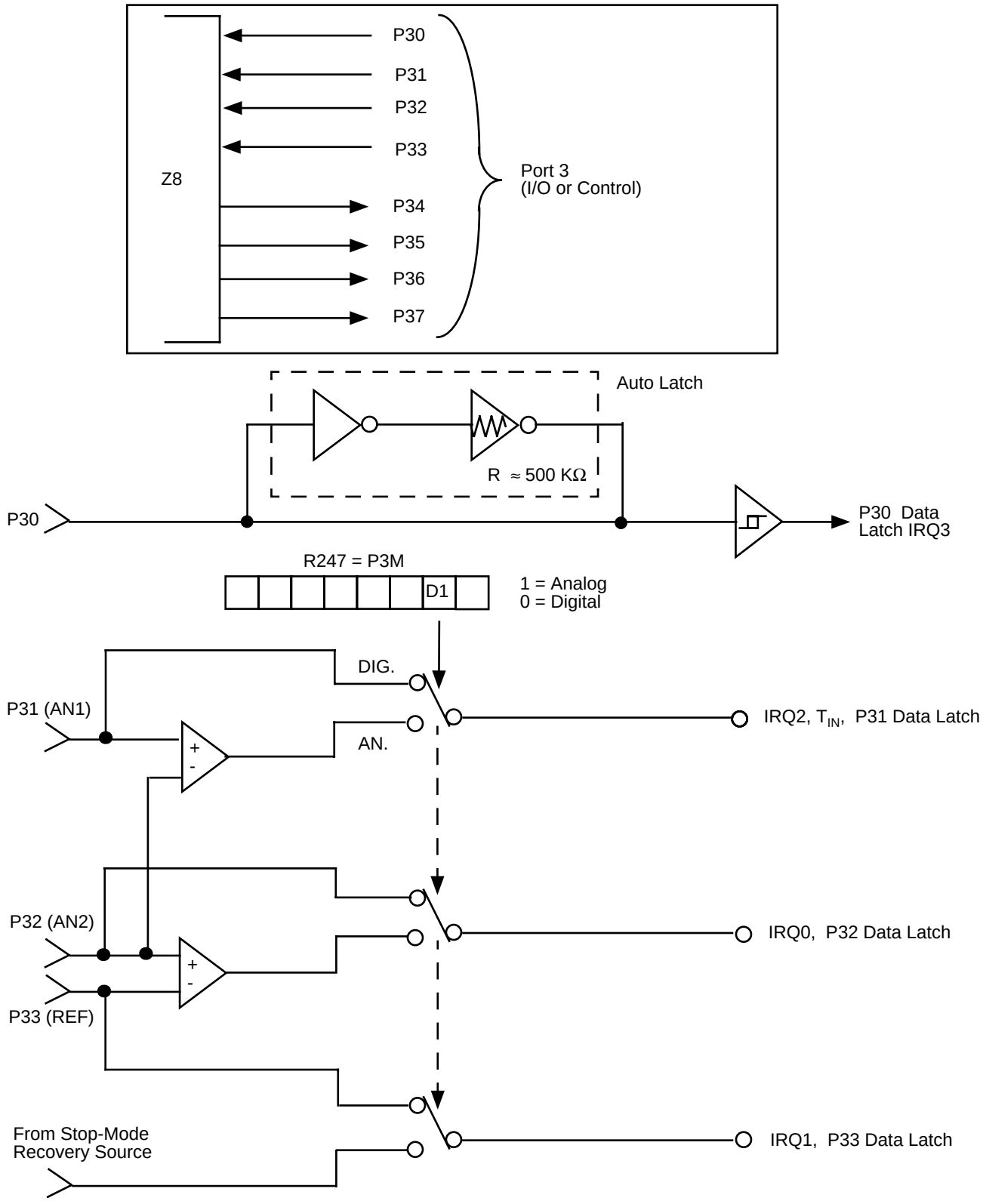


Figure 5-32. Port Configuration of Comparator Inputs on P31, P32, and P33

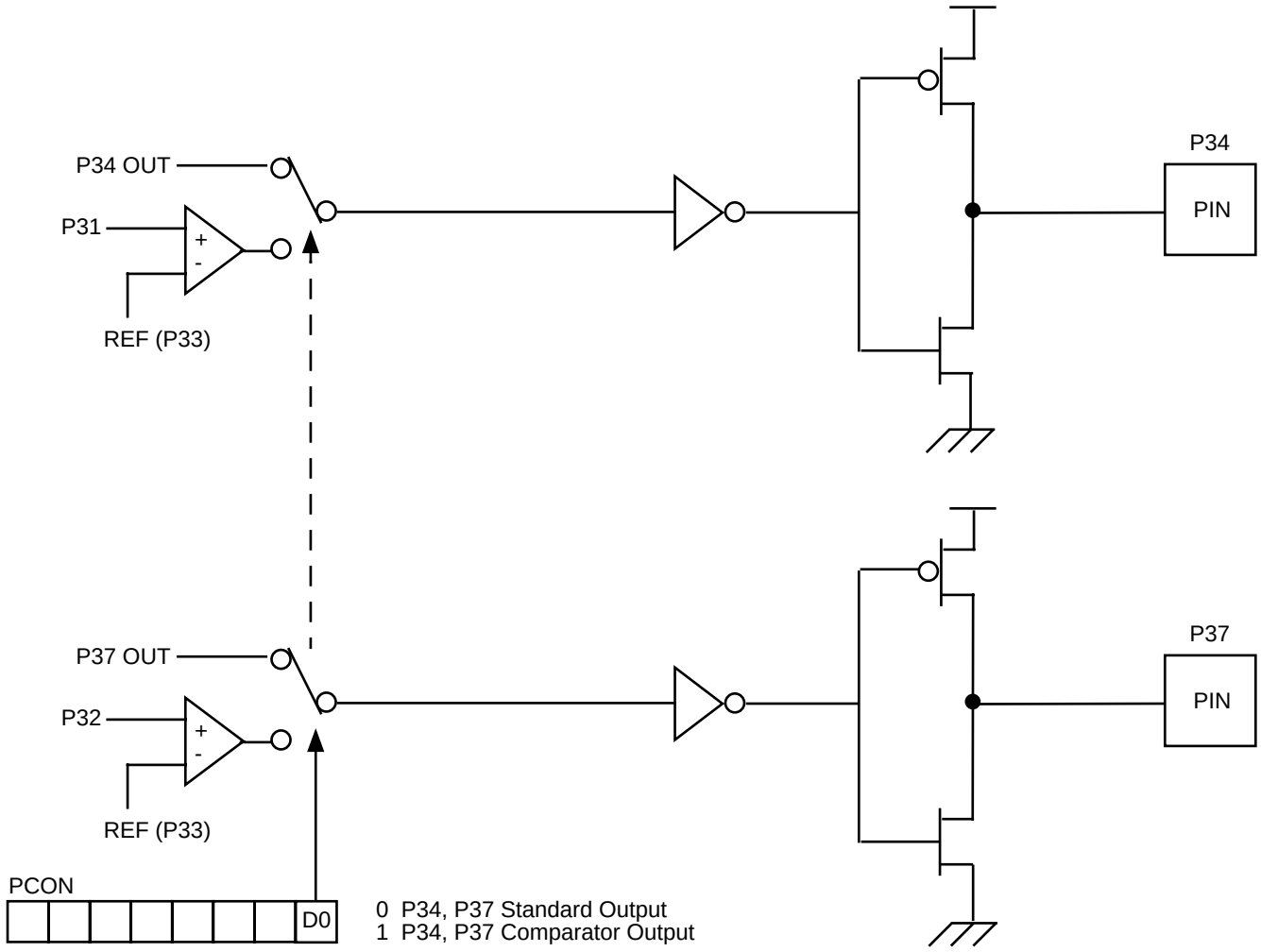


Figure 5-33. Port 3 Configuration

5.8.2 Comparator Programming

Example of enabling analog comparator mode.

```
LD P3M, #XXXX XX1XB
```

Note: X = Any Binary Number

Example of enabling analog comparator output.

```
LD RP, #%0FH        ;Sets register pointer to
                    ;working register group 0
                    ;and Expanded Register
                    ;File Bank F.
LD R0, #XXXX XXX1B  ;Enables comparator
                    ;outputs using PCON

                    ;Register programming.
```

5.8.3 COMPARATOR OPERATION

After enabling the Analog Comparator mode, P33 becomes a common reference input for both comparators. The P33 (Ref) is hard wired to the reference inputs to both comparators and cannot be separated. P31 and P32 are always connected to the positive inputs to the comparators. P31 is the positive input to comparator AN1 while P32 is the positive input to comparator AN2. The outputs to comparators AN1 and AN2 are AN1-out and AN2-out, respectively.

The comparator output reflects the relationship between the positive input to the reference input.

Example: If the voltage on AN1 is higher than the voltage on Ref then AN1-out will be at a high state. If voltage on AN2 is lower than the voltage on Ref then AN2-out will be at a Low state. In this example, when the Port 3 register is read, Bits D1 = 1 and D2 = 0. If the comparator outputs are enabled to come out on P34 and P37, then P34 = 1 and P37 = 0. Please note that the previous data stored in P34 and P37 is not disturbed. Once the comparator outputs are de-selected the stored values in the P34 and P37 register bits will be reflected on these pins again.

5.8.4 Interrupts

In the example from Section 5.8.3, P32 (AN2) will generate an interrupt based on the result of the comparison being low and the Interrupt Request Register (IRQ FAH) having bits D7=0 and D6=0. If IRQ D7=1 and D6=0 then both P31 and P32 would generate interrupts.

5.8.5 Comparator Definitions

5.8.5.1 V_{ICR}

The usable voltage range for both positive inputs and the reference input is called the common mode voltage range (V_{ICR}). The comparator is not guaranteed to work if the inputs are outside of the V_{ICR} range.

5.8.5.2 V_{OFFSET}

The absolute value of the voltage between the positive input and the reference input required to make the comparator output voltage switch is the input offset voltage (V_{OFFSET}). If AN1 is 3.000V and Ref is 3.001V when the comparator output switches states then the Voffset = 1mV.

5.8.5.3 I_{IO}

For CMOS voltage comparator inputs, the input offset current (I_{IO}) is the leakage current of the CMOS input gate.

5.8.6 RUN Mode

P33 is not available as an interrupt input during Analog Mode. P31 and P32 are valid interrupt inputs in conjunction with P33 (Ref) when in the Analog Mode.

P31 can still be used as T_{IN} when the analog mode is selected. If comparator outputs are desired to be outputted on the Port 3 outputs, please refer to specific products specification for priority of mixing when other special features are sharing those same Port 3 pins.

5.8.7 HALT Mode

The analog comparators are functional during HALT Mode if the Analog Mode has been enabled. P31 and P32, in conjunction with P33 (Ref) will be able to generate interrupts. Only P33 cannot generate an interrupt since the P33 input goes directly to the Ref input of the comparators and is disconnected from the interrupt sensing circuits.

5.8.8 STOP Mode

The analog comparators are disabled during STOP Mode so it does not use any current at that time. If P31, P32, or P33 are used as a source for Stop-Mode Recovery, the Port 3 Digital Mode must be selected by setting bit D1=0 in the Port 3 Mode Register. Otherwise in STOP Mode, the P31, P32, and P33 cannot be sensed. If the Analog Mode was selected when entering STOP Mode, it will still be enabled after a valid SMR triggered reset.

5.9 OPEN-DRAIN CONFIGURATION

All Z8s can configure Port 2 to provide open-drain outputs by programming the Port 3 Mode Register (P3M) bit D0=0.

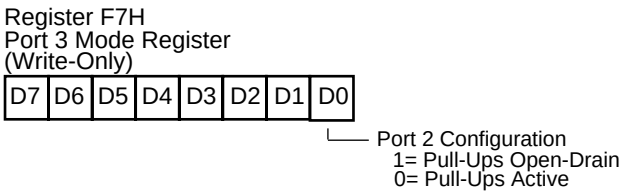


Figure 5-34. Port 2 Configuration

Other Z8s that have a Port Configuration Register (PCON) that can configure Port 0 and Port 1 to provide open-drain outputs. The PCON Register is located in Expanded Register File (ERF) Bank F, Register 00H. See Figure 5-35.

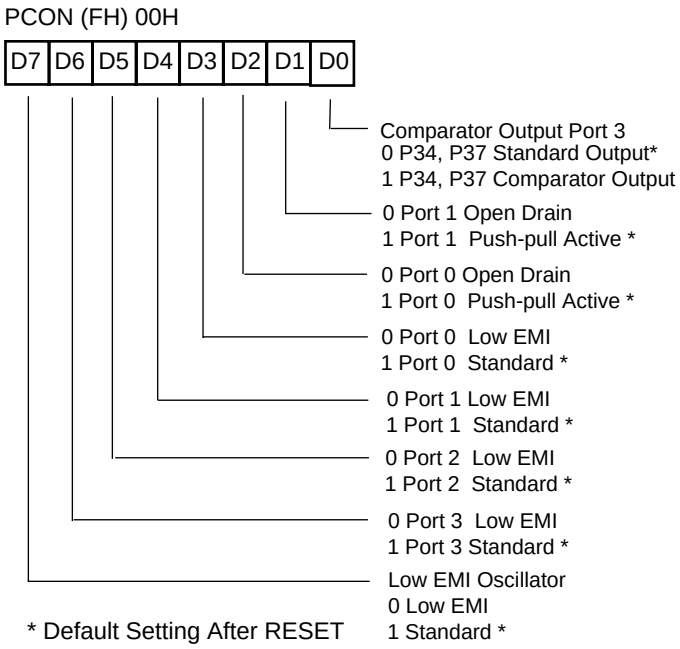


Figure 5-35. Port Configuration Register (PCON) (Write-Only)

Port 1 Open-Drain (D1). Port 1 can be configured as open-drain by resetting this bit (D1=0) or configured as push-pull active by setting this bit (D1=1). The default value is 1.

Port 0 Open Drain (D2). Port 0 can be configured as open-drain by resetting this bit (D2=0) or configured as push-pull active by setting this bit (D2=1). The default value is 1.

5.10 LOW EMI EMISSION

Some Z8s can be programmed to operate in a Low EMI Emission Mode using the Port configuration register (PCON). The PCON register allows the oscillator and all I/O ports to be programmed in the Low-EMI Mode independently. Other Z8s may offer a ROM Mask or OTP programming option to configure the Z8 Ports and oscillator globally to a Low-EMI mode (where the XTAL frequency is set equal to the internal system clock frequency.

Use of the Low EMI feature results in:

- The output pre-drivers slew rate reduced to 10 ns (typical).
- Low EMI output drivers have resistance of 200 Ohms (typical).
- Low EMI Oscillator.
- All output drivers are approximately 25 percent of the standard drive.
- Internal SCLK/TCLK = XTAL operation limited to a maximum of 4 MHz - 250 ns cycle time, when Low EMI Oscillator is selected and system clock (SCLK=XTAL, SMR Reg. Bit D1=1).

For Z8s having the PCON register feature, the following bits control the Low EMI options:

- **Low EMI Port 0 (D3).** Port 0 can be configured as a Low EMI Port by resetting this bit (D3=0) or configured as a Standard Port by setting this bit (D3=1). The default value is 1.
- **Low EMI Port 1 (D4)**. Port 1 can be configured as a Low EMI Port by resetting this bit (D4=0) or configured as a Standard Port by setting this bit (D4=1). The default value is 1.
- **Low EMI Port 2 (D5).** Port 2 can be configured as a Low EMI Port by resetting this bit (D5=0) or configured as a Standard Port by setting this bit (D5=1). The default value is 1.
- **Low EMI Port 3 (D6).** Port 3 can be configured as a Low EMI Port by resetting this bit (D6=0) or configured as a Standard Port by setting this bit (D6=1). The default value is 1.
- **Low EMI OSC (D7).** This bit of the PCON Register controls the Low EMI oscillator. A 1 in this location configures the oscillator with standard drive, while a 0 configures the oscillator with low noise drive. The Low-EMI mode will reduce the drive of the oscillator (OSC). The default value is 1. XTAL/2 mode is not effected by this bit.

Note: The maximum external clock frequency is 4 MHz when running in the Low EMI oscillator mode.

Please refer to the selected device product specification for availability of the Low EMI feature and programming options.

5.11 INPUT PROTECTION

All CMOS ROM Z8s have I/O pins with diode input protection. There is a diode from the I/O pad to V_{CC} and to V_{SS}. See Figure 5-36.

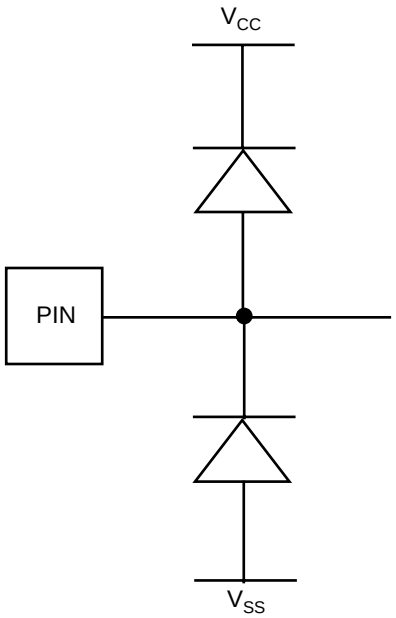


Figure 5-36. Diode Input Protection

On CMOS OTP EPROM Z8s, the Port 3 inputs P31, P32, P33 and the XTAL 1 pin have only the input protection diode from pad to V_{SS}. See Figure 5-37.

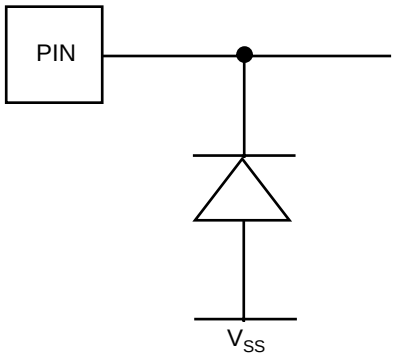


Figure 5-37. OTP Diode Input Protection

The high-side input protection diodes were removed on these pins to allow the application of +12.5V during the various OTP programming modes.

For better noise immunity in applications that are exposed to system EMI, a clamping diode to V_{CC} from these pins may be required to prevent entering the OTP programming mode or to prevent high voltage from damaging these pins.

5.12 Z8 CMOS AUTO LATCHES

I/O port bits that are configurable as inputs are protected against open circuit conditions using Auto Latches. An Auto Latch is a circuit which, in the event of an open circuit condition, latches the input at a valid CMOS level. This inhibits the tendency of the input transistors to self-bias in the forward active region, thus drawing excessive supply current. A simplified schematic of the CMOS Z8 I/O circuit is shown in Figure 5-38.

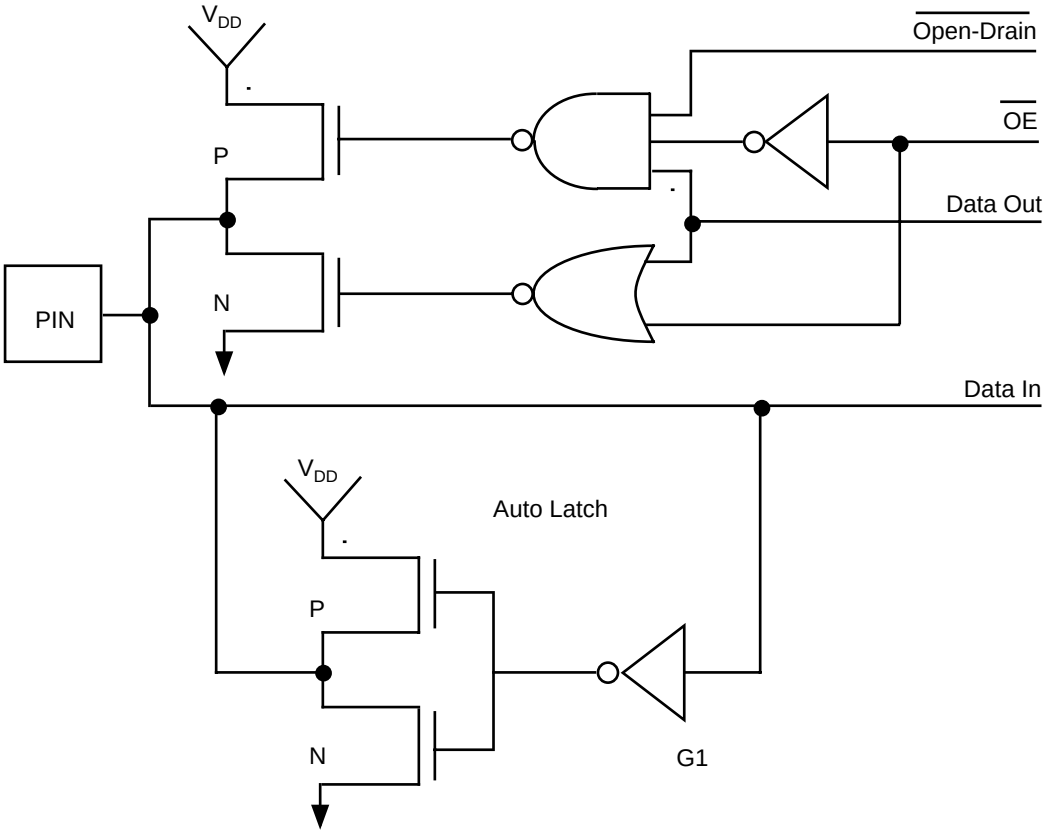


Figure 5-38. Simplified CMOS Z8 I/O Circuit

The operation of the Auto Latch circuit is straight-forward. Assume the input pad is latched at +5V (logic 1). The inverter G1 inverts the bit, turning the P-channel FET ON and the N-channel FET OFF. The output of the circuit is effectively shorted to V_{DD}, returning +5V to the input. If the pad is then disconnected from the +5V source, the Auto Latch will hold the input at the previous state. If the device is powered up with the input floating, the state of the Auto Latch will be at either supply, but which state is unpredictable.

There are four operating conditions which will activate the Auto Latches. The first, which occurs when the input pin is physically disconnected from any source, is the most obvious. The second occurs when the input is connected to the output of a device with tri-state capability.

The Auto Latch will also activate when the input voltage at the pin is not within 200 microvolts or so of either supply rail. In this case, the circuit will draw current, which is not significant compared to the Icc operating current of the device, but will increase I_{CC2} STOP Mode current of the device dramatically.

The fourth condition occurs when the I/O bit is configured as an output. Referring to the output section of Figure 5-38, there are two ways of tri-stating the port pin. The first is by configuring the port as an input, which disables the $\overline{OE}$ signal turning both transistors off. The second can be achieved in output mode by writing a “1” to the output port, then activating the open drain mode. Both transistors are again off, and the port bit is in a high impedance state. The Auto Latches then pull the input section toward V_{DD}.

Auto Latch Model:

The Auto Latch's equivalent circuit is shown in Figure 5-39. When the input is high, the circuit consists of a resistance Rp from V_{DD} (the P-channel transistor in its ON state) and a much greater resistance Rh to G_{ND}. Current Iao flows from V_{DD} to the output. When the input is low, the circuit may be modeled as a resistance Rp from G_{ND} (the N-channel transistor in the ON state) and a much greater resistance Rh to V_{DD}. Current Iao now flows from the input to ground. The Auto Latch is characterized with respect to Iao, so the equivalent resistance Rp is calculated according to $R_P = (V_{DD}-VIN)/I_{AO}$. The worst case equivalent resistance Rp (min) may be calculated at the worst case input voltage, $V_I = V_{IH}$(min).

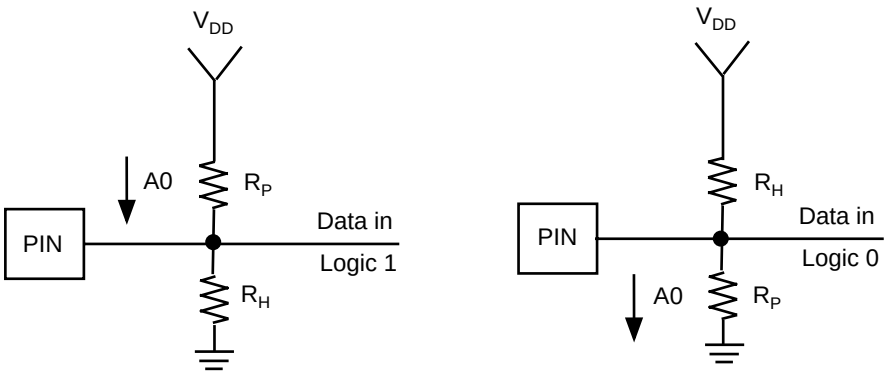


Figure 5-39. Auto Latch Equivalent Circuit

Design Considerations:

For circuits in which the Auto Latch is active, consideration should be given to the loading constraints of the Auto Latches. For example, with weak values of V_{IN}, close to Vih (min) or Vil (max), pullup or pull-down resistances must be calculated using Ref = R/Rp. For best case STOP mode operation, the inputs should be within 200 mV of the supply rails.

In output mode, if a port bit is forced into a tri-state condition, the Auto Latches will force the pad to V_{DD}. If there is an external pulldown resistor on the pin, the voltage at the pin may not switch to GND due to the Auto Latch. As shown in Figure 5-40, the equivalent resistance of the Auto Latch and the external pulldown form a voltage divider, and if the external resistor is large, the voltage developed across it will exceed Vil(max). For worst case:

$V_{IL}(max > V_{DD}\ [Rext/(Rext+Rp)]$
$Rext(max) = [(Vil(max)/V_{DD})Rp]/[1-(Vil(max)/V_{DD})]$

For V_{DD} = 5.0V and Iao = 5 uA we have Vih(max) =0.8V:
R_{EXT}(max) = (0.16/1M)/(1–0.16) = 190 K ohms.

Rp increases rapidly with V_{DD}, so increased V_{DD} will relax the requirement on Rext.

In summary, the CMOS Z8 Auto Latch inhibits excessive current drain in Z8 devices by latching an open input to either V_{DD} or GND. The effect of the Auto Latch on the I/O characteristics of the device may be modeled by a current Iao and a resistor Rp, whose value is V_{DD}/Iao.

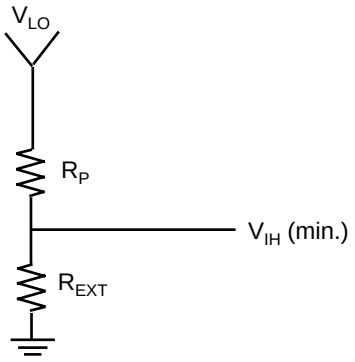


Figure 5-40. Effect of Pulldown Resistors on Auto Latches

Totally Logical

CHAPTER 6
COUNTER/TIMERS

6.1 INTRODUCTION

The Z8 MCU® provides up to two 8-bit counter/timers, T0 and T1, each driven by its own 6-bit prescaler, PRE0 and PRE1 (Figure 6-1). Both counter/timers are independent of the processor instruction sequence, that relieves software from time-critical operations such as interval timing or event counting. Some MCUs offer clock scaling using the SMR register. See the device product specification for clock available options. The following description is typical.

Each counter/timer operates in either Single-Pass or Continuous mode. At the end-of-count, counting either stops or the initial value is reloaded and counting continues. Under software control, new values are loaded immediately or when the end-of-count is reached. Software also controls the counting mode, how a counter/timer is started or stopped, and its use of I/O lines. Both the counter and prescaler registers can be altered while the counter/timer is running.

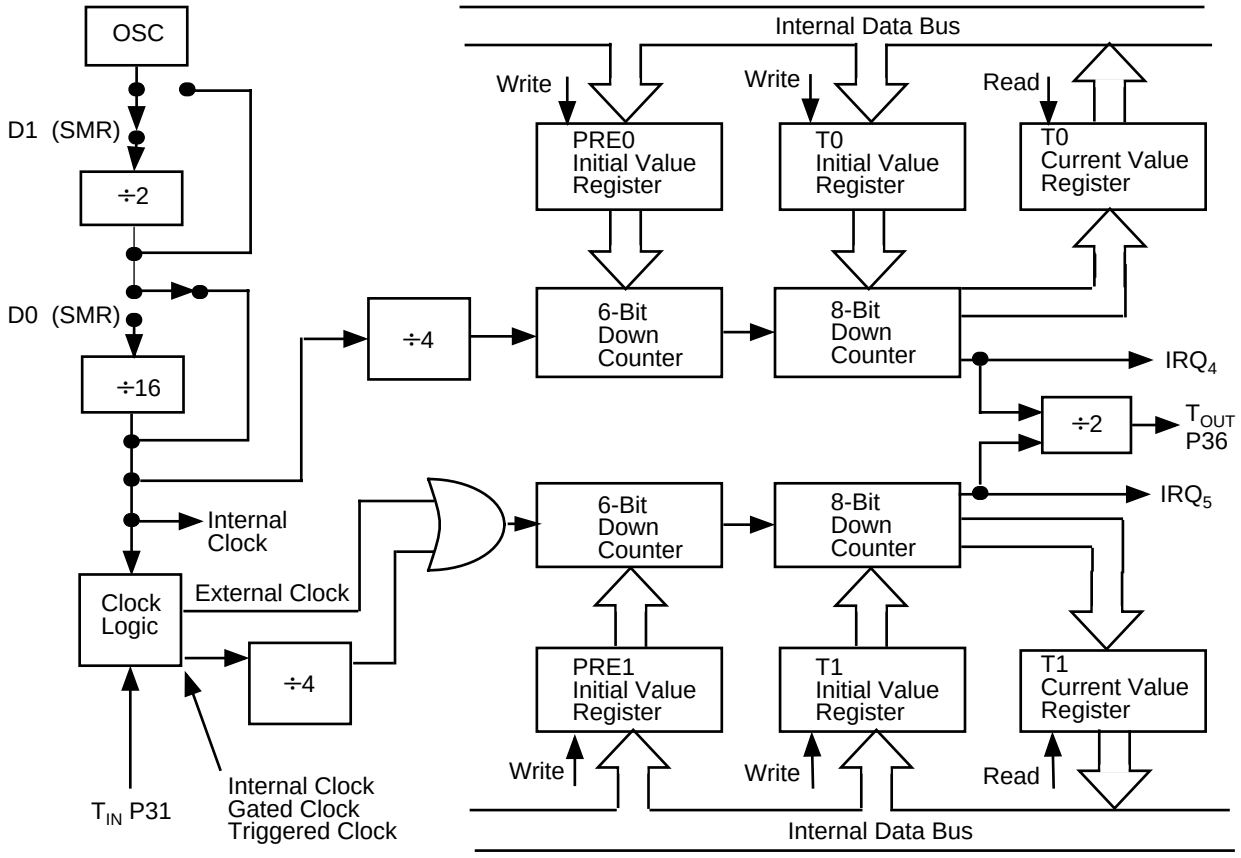


Figure 6-1. Counter/Timer Block Diagram

Counter/timers 0 and 1 are driven by a timer clock generated by dividing the internal clock by four. The divide-by-four stage, the 6-bit prescaler, and the 8-bit counter/timer form a synchronous 16-bit divide chain. Counter/timer 1 can also be driven by a external input (T_{IN}) using P31. Port 3 line P36 can serve as a timer output (T_{OUT}) through which T0, T1, or the internal clock can be output. The timer output will toggle at the end-of-count.

The counter/timer, prescaler, and associated mode registers are mapped into the register file as shown in Figure 6-2. This allows the software to treat the counter/timers as general-purpose registers, and eliminates the need for special instructions.

6.2 PRESCALERS AND COUNTER/TIMERS

The prescalers, PRE0 (F5H) and PRE1 (F3H), each consist of an 8-bit register and a 6-bit down-counter as shown in Figure 6-1. The prescaler registers are write-only registers. Reading the prescalers returns the value FFH. Figures 6-3 and 6-4 show the prescaler registers.

The six most significant bits (D2-D7) of PRE0 or PRE1 hold the prescalers count modulo, a value from 1 to 64 decimal. The prescaler registers also contain control bits that specify T0 and T1 counting modes. These bits also indicate whether the clock source for T_1 is internal or external. These control bits will be discussed in detail throughout this chapter.

The counter/timer registers, T0 (F4H) and T1 (F2H), each consist of an 8-bit down-counter, a write-only register that holds the initial count value, and a read-only register that holds the current count value (Figure 6-1). The initial value can range from 1 to 256 decimal (01H,02H,..,00H). Figure 6-5 illustrates the counter/timer registers.

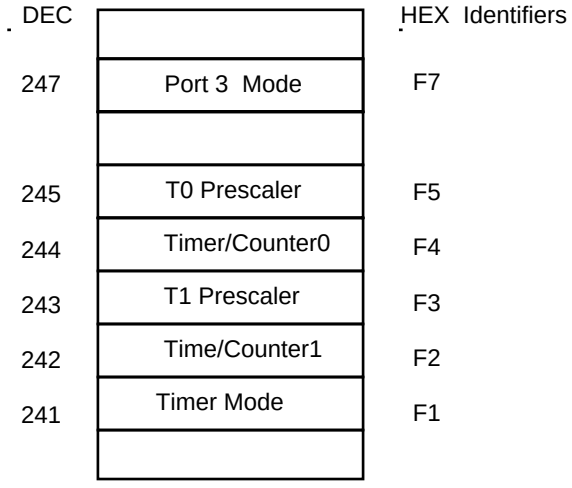


Figure 6-2. Counter/Timer Register Map

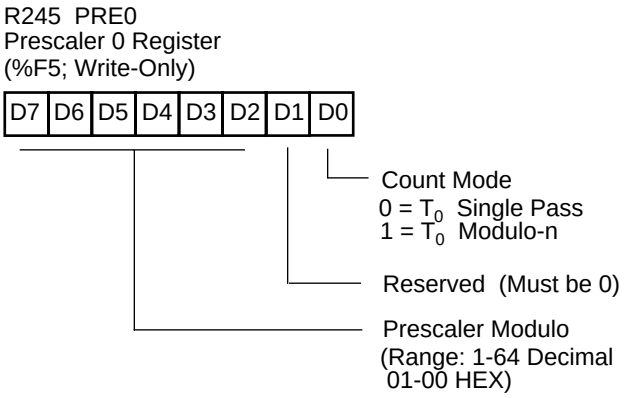


Figure 6-3. Prescaler 0 Register

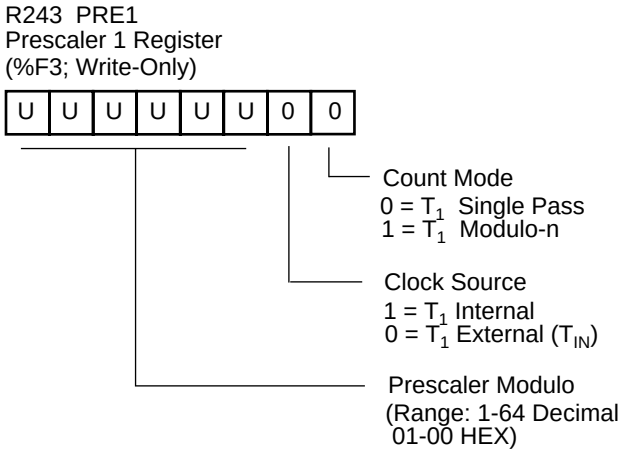


Figure 6-4. Prescaler 1 Register

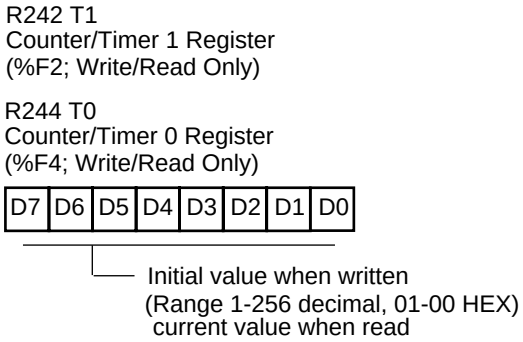


Figure 6-5. Counter / Timer 0 and 1 Registers

6.3 COUNTER/TIMER OPERATION

Under software control, counter/timers are started and stopped via the Timer Mode Register (TMR,F1H) bits D_0-D_3 (Figure 6-6). Each counter/timer is associated with a Load bit and an Enable Count bit.

6.3.1 Load and Enable Count Bits

Setting the Load bit (D_0 for T0 and D_2 for T1) transfers the initial value in the prescaler and the counter/timer registers into their respective down-counters. The next internal clock resets bits D_0 and D_2 to 0, readying the Load bit for the next load operation. New values may be loaded into the down-counters at any time. If the counter/timer is running, it continues to do so and starts the count over with the new value. Therefore, the Load bit actually functions as a software re-trigger.

The counter timers remain at rest as long as the Enable Count bits are 0. To enable counting, the Enable Count bit (D_1 for T0 and D_3 for T1) must be set to 1. Counting actually starts when the Enable Count bit is written by an instruction. The first decrement occurs four internal clock periods after the Enable Count bit has been set. If T1 is configured to use an external clock, the first decrement begins on the next clock period. The Load and Enable Count bits can be set at the same time. For example, using the instruction:

OR TMR,#03H

sets both D0 and D1 of the TMR. This loads the initial values of PRE0 and T0 into their respective counters and starts the count after the M2T2 machine state after the operand is fetched (Figure 6-7).

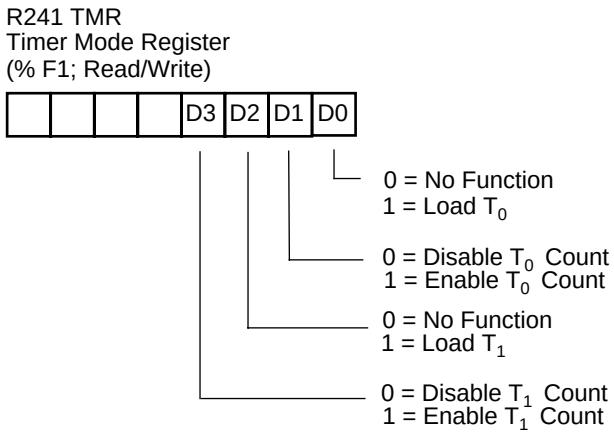


Figure 6-6. Timer Mode Register

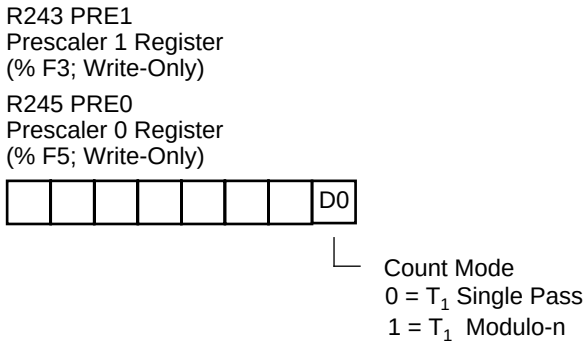


Figure 6-7. Starting The Count

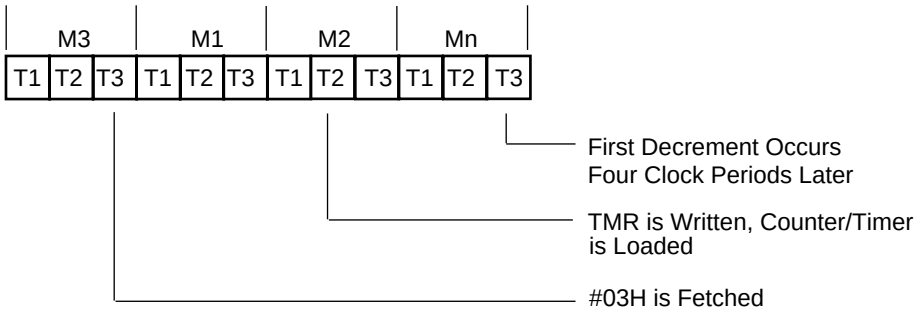


Figure 6-8. Counting Modes

6.3.2 Prescaler Operations

During counting, the programmed clock source drives the 6-bit Prescaler Counter. The counter is counted down from the value specified by bits of the corresponding Prescaler Register, PRE0 (bit 7 to bit 2) or PRE1 (bit 7 to bit 2). (Figures 6-3, 6-4). When the Prescaler Counter reaches its end-of-count, the initial value is reloaded and counting continues. The prescaler never actually reaches 0. For example, if the prescaler is set to divide-by-three, the count sequence is:

3–2–1–3–2–1–3–2–1–3...

Each time the prescaler reaches its end of count a carry is generated, that allows the Counter/Timer to decrement by one on the next timer clock input. When the Counter/Timer and the prescaler both reach the end-of-count, an interrupt request is generated (IRQ4 for T0, IRQ5 for T1). Depending on the counting mode selected, the Counter/Timer will either come to rest with its value at 00H (Single-Pass Mode) or the initial value will be automatically reloaded and counting will continue (Continuous Mode). The counting modes are controlled by bit 0 of PRE0 and bit 0 of PRE1. (Figure 6-8). A 0, written to this bit configures the counter for Single-pass counting mode, while a 1 written to this bit configures the counter for Continuous mode.

The Counter/Timer can be stopped at any time by setting the Enable Count bit to 0, and restarted by setting it back to 1. The Counter/Timer will continue its count value at the time it was stopped. The current value in the Counter/Timer can be read at any time without affecting the counting operation.

Note: The prescaler registers are write-only and cannot be read.

New initial values can be written to the prescaler or the Counter/Timer registers at any time. These values will be transferred to their respective down counters on the next load operation. If the Counter/Timer mode is continuous, the next load occurs on the timer clock following an end-of-count. New initial values should be written before the desired load operation, since the prescalers always effectively operate in Continuous count mode.

The time interval (i) until end-of-count, is given by the equation:

$i = t \times p \times v$

in which:

t = four times the internal clock period.

The internal clock frequency defaults to the external clock source (XTAL, ceramic resonator, and others) divided by 2. Some Z8 microcontrollers allow this divisor to be changed via the Stop-Mode Recovery register. See the product data sheet for available clock divisor options.

Note that t is equal to eight divided-by-XTAL frequency of the external clock source for T1 (external clock mode only).

p = the prescaler value (1 – 63) for T_0 and T_1.

The minimum prescaler count of 1 is achieved by loading 000001xx. The maximum prescaler count of 63 is achieved by loading 111111xx.

v = the Counter/Timer value (1-256)

Minimum duration is achieved by loading 01H (1 prescaler output count), maximum duration is achieved by loading 00H (256 prescaler outputs counts).

The prescaler and counter/timer are true divide-by-n counters.

6.4 T_{OUT} MODES

The Timer Mode Register TMR (F1H) (Figure 6-9), is used in conjunction with the Port 3 Mode Register P3M (F7H) (Figure 6-10) to configure P36 for T_{OUT} operation for T0 and T1. In order for T_{OUT} to function, P36 must be defined as an output line by setting P3M bit 5 to 0. Output is controlled by one of the counter/timers (T0 or T1) or the internal clock.

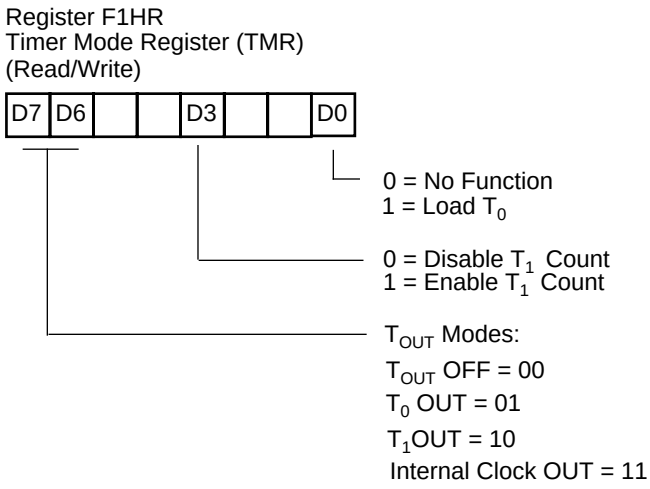


Figure 6-9. Timer Mode Register (T_{OUT} Operation)

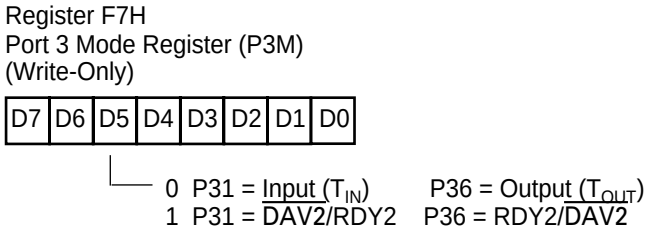


Figure 6-10. Port 3 Mode Register (T_{OUT} Operation)

6.4 T_{OUT} MODES (Continued)

The counter/timer to be output is selected by TMR bit 7 and bit 6. T0 is selected to drive the T_{OUT} line by setting bit 7 to 0 and bit 6 to 1. Likewise, T1 is selected by setting bit 7 and bit 6 to 1 and 0, respectively. The counter/timer T_{OUT} mode is turned off by setting TMR bit and bit 6 both to 0, freeing P36 to be a data output line.

T_{OUT} is initialized to a logic 1 whenever the TMR Load bit (bit 0 for T0 or bit 1 for T2) is set to 1. The T_{OUT} configuration timer load, and Timer Enable Count bits for the counter/timer driving the T_{OUT} pin can be set at the same time. For example, using the instruction:

OR TMR,#43H

- Configures T0 to drive the T_{OUT} pin (P36).
- Sets the P36 T_{OUT} pin to a logic 1 level.
- Loads the initial PRE0 and T0 levels into their respective counters and starts the counter after the M2T2 machine state after the operand is fetched.

At end-of-count, the interrupt request line (IRQ4 or IRQ5), clocks a toggle flip-flop. The output of this flip-flop drives the T_{OUT} line, P36. In all cases, when the selected counter/timer reaches its end-of-count, T_{OUT} toggles to its opposite state (Figure 6-11). If, for example, the counter/timer is in Continuous Counting Mode, Tout will have a 50 percent duty cycle output. This duty cycle can easily be controlled by varying the initial values after each end-of-count.

The internal clock can be selected as output instead of T0 or T1 by setting TMR bit 7 and bit 6 both to 1. The internal clock (XTAL frequency/2) is then directly output on P36 (Figure 6-12).

While programmed as T_{OUT}, P36 cannot be modified by a write to port register P3. However, the Z8® software can examine the P36 current output by reading the port register.

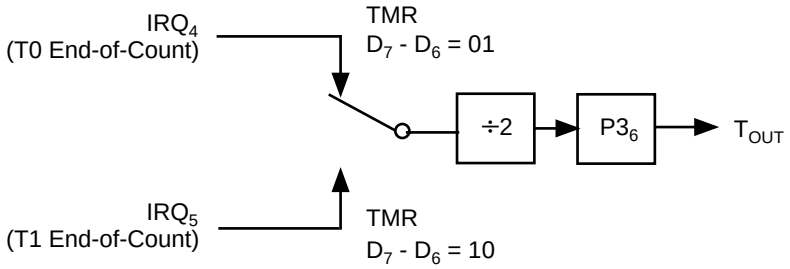


Figure 6-11. T0 and T1 Output Through T_{OUT}

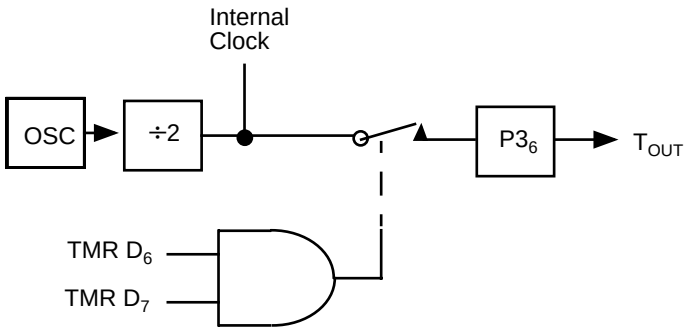


Figure 6-12. Internal Clock Output Through T_{OUT}

6.5 T_{IN} MODES

The Timer Mode Register TMR (F1H) (Figure 6-13) is used in conjunction with the Prescaler Register PRE1 (F3H) (Figure 6-14) to configure P31 as T_{IN}. T_{IN} is used in conjunction with T1 in one of four modes:

- External Clock Input
- Gated Internal Clock
- Triggered Internal Clock
- Retriggerable Internal Clock

Note: The T_{IN} mode is restricted for use with timer 1 only. To enable the T_{IN} mode selected (via TMR bits 4- 5), bit 1 of PRE1 must be set to 0.

The counter/timer clock source must be configured for external by setting the PRE1 Register bit 2 to 1. The Timer Mode Register bit 5 and bit 4 can then be used to select the desired T_{IN} operation.

For T1 to start counting as a result of a T_{IN} input, the Enable Count bit (bit 3 in TMR) must be set to 1. When using T_{IN} as an external clock or a gate input, the initial values must be loaded into the down counters by setting the Load bit (bit 2 in TMR) to a 1 before counting begins. In the descriptions of T_{IN} that follow, it is assumed the programmer has performed these operations. Initial values are automatically loaded in Trigger and Retrigger modes so software loading is unnecessary.

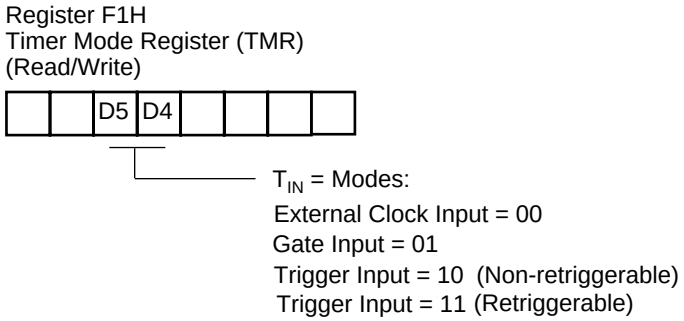


Figure 6-13. Timer Mode Register (T_{IN} Operation)

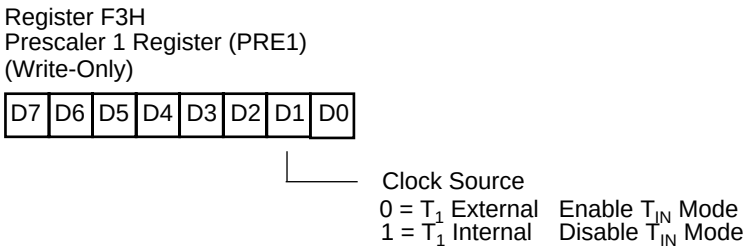


Figure 6-14. Prescaler 1 Register (T_{IN} Operation)

It is suggested that P31 be configured as an input line by setting P3M Register bit 5 to 0, although T_{IN} is still functional if P31 is configured as a handshake input.

Each High-to-Low transition on T_{IN} generates an interrupt request IRQ2, regardless of the selected T_{IN} mode or the enabled/disabled state of T1. IRQ2 must therefore be masked or enabled according to the needs of the application.

6.5.1 External Clock Input Mode

The T_{IN} External Clock Input Mode (TMR bit 5 and bit 4 both set to 0) supports counting of external events, where an event is considered to be a High-to-Low transition on T_{IN} (Figure 6-15).

Note: See the product data sheet for the minimum allowed T_{IN} external clock input period (T_P T_{IN}).

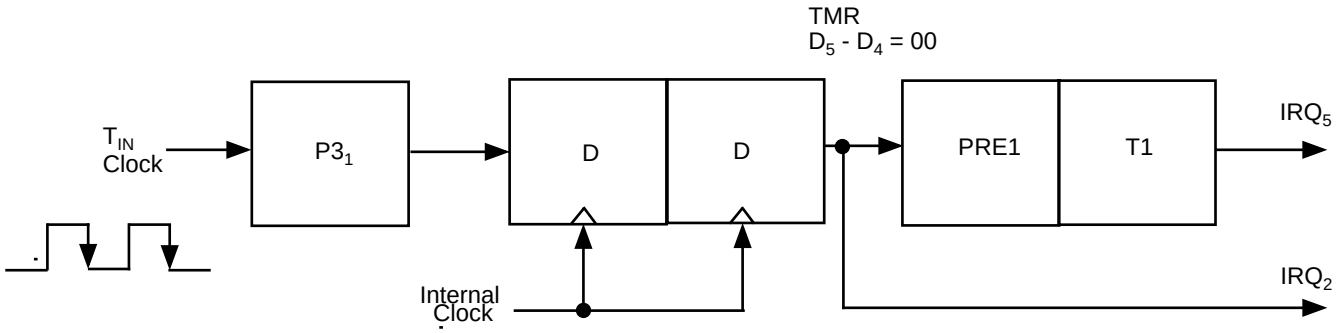


Figure 6-15. External Clock Input Mode

6.5.2 Gated Internal Clock Mode

The T_{IN} Gated Internal Clock Mode (TMR bit 5 and bit 4 set to 0 and 1 respectively) measures the duration of an external event. In this mode, the T1 prescaler is driven by the internal timer clock, gated by a High level on T_{IN} (Figure 6-16). T1 counts while T_{IN} is High and stops counting while T_{IN} is Low. Interrupt request IRQ2 is generated on the High-to-Low transition of T_{IN} signalling the end of the gate input. Interrupt request IRQ5 is generated if T1 reaches its end-of-count.

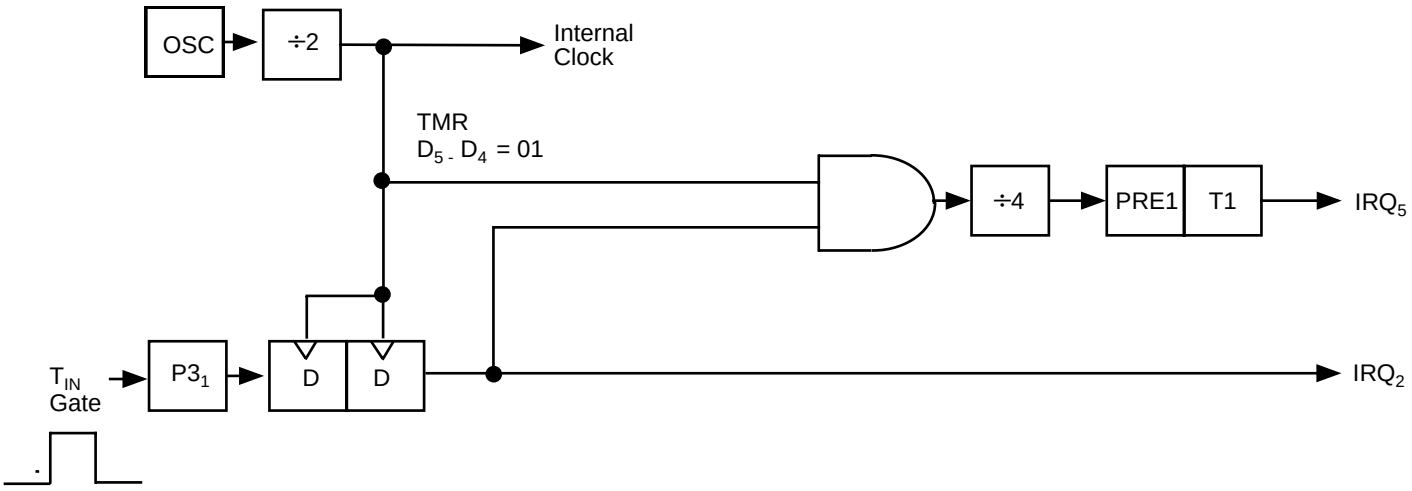


Figure 6-16. Gated Clock Input Mode

6.5.3 Triggered Input Mode

The T_{IN} Triggered Input Mode (TMR bits 5 and 4 are set to 1 and 0, respectively) causes T1 to start counting as the result of an external event (Figure 6-17). T1 is then loaded and clocked by the internal timer clock following the first High-to-Low transition on the T_{IN} input. Subsequent T_{IN} transitions do not affect T1. In the Single-Pass Mode, the Enable bit is reset whenever T1 reaches its end-of-count. Further T_{IN} transitions will have no effect on T1 until software sets the Enable Count bit again. In Continuous mode, once T1 is triggered counting continues until software resets the Enable Count bit. Interrupt request IRQ5 is generated when T1 reaches its end-of-count.

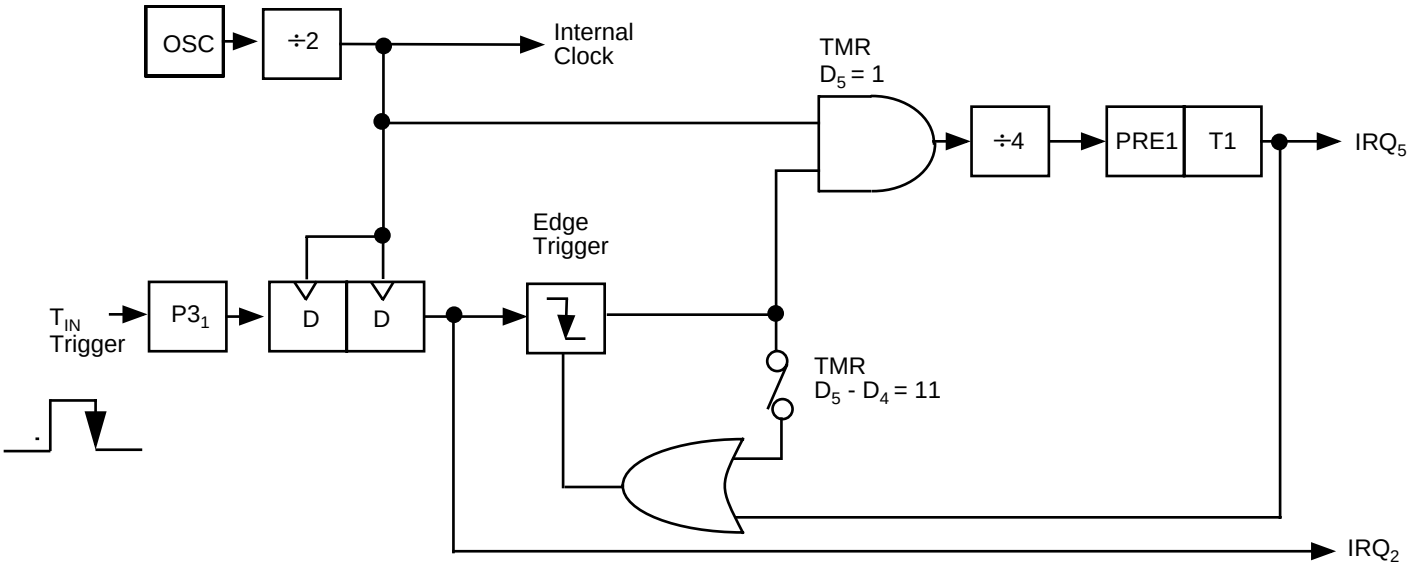


Figure 6-17. Triggered Clock Mode

6.5.4 Retriggerable Input Mode

The T_{IN} Retriggerable Input Mode (TMR bits 5 and 4 are set to 1) causes T1 to load and start counting on every occurrence of a High-to-Low transition on T_{IN} (Figure 6-17). Interrupt request IRQ5 will be generated if the programmed time interval (determined by T1 prescaler and counter/timer register initial values) has elapsed since the last High-to-Low transition on T_{IN}. In Single-Pass Mode, the end-of-count resets the Enable Count bit. Subsequent T_{IN} transitions will not cause T1 to load and start counting until software sets the Enable Count bit again. In Continuous Mode, counting continues once T1 is triggered until software resets the Enable Count bit. When enabled, each High-to-Low T_{IN} transition causes T1 to reload and restart counting. Interrupt request IRQ5 is generated on every end-of-count.

6.6 CASCADING COUNTER/TIMERS

For some applications, it may be necessary to measure a time interval greater than a single counter/timer can measure. In this case, T_{IN} and T_{OUT} can be used to cascade T0 and T1 as a single unit (Figure 6-18). T0 should be configured to operate in Continuous mode and to drive T_{OUT}. T_{IN} should be configured as an external clock input to T1 and wired back to T_{OUT}. On every other T0 end-of-count, T_{OUT} undergoes a High-to-Low transition that causes T1 to count.

T1 can operate in either Single-Pass or Continuous mode. When the T1 end-of-count is reached, interrupt request IRQ5 is generated. Interrupt requests IRQ2 (T_{IN} High-to-Low transitions) and IRQ4 (T0 end-of-count) are also generated but are most likely of no importance in this configuration and should be disabled.

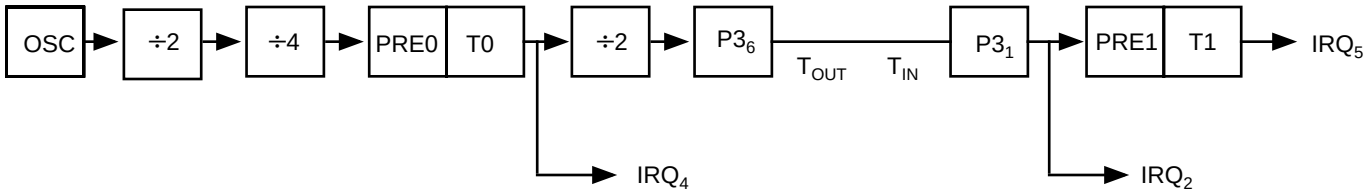


Figure 6-18. Cascaded Counter / Timers

6.7 RESET CONDITIONS

After a hardware reset, the counter/timers are disabled and the contents of the counter/timer and prescaler registers are undefined. However, the counting modes are configured for Single-Pass and the T1 clock source is set for external.

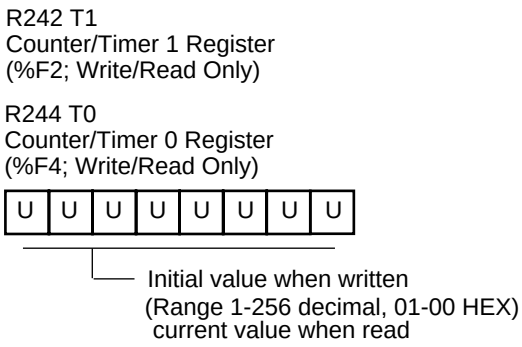


Figure 6-19. Counter / Timer Reset

T_{IN} is set for External Clock mode, and the T_{OUT} mode is off. Figures 6-19 through 6-22 show the binary reset values of the Prescaler, Counter/Timer, and Timer Mode registers.

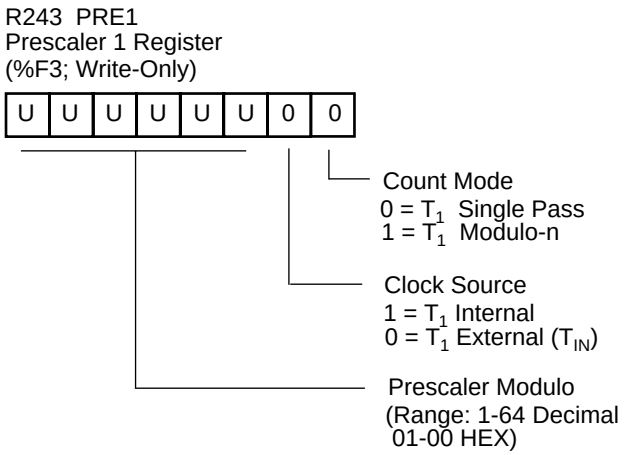


Figure 6-20. Prescaler 1 Register Reset

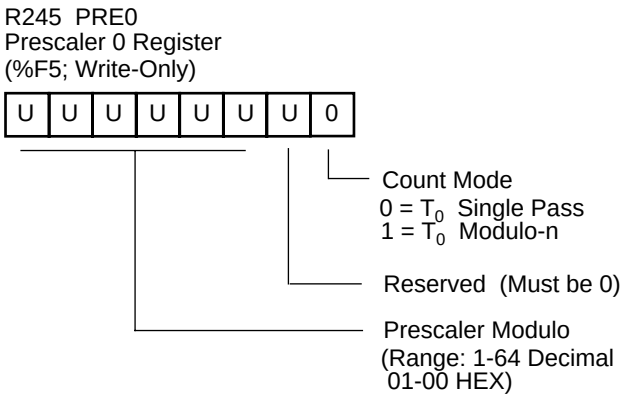


Figure 6-21. Prescaler 0 Reset

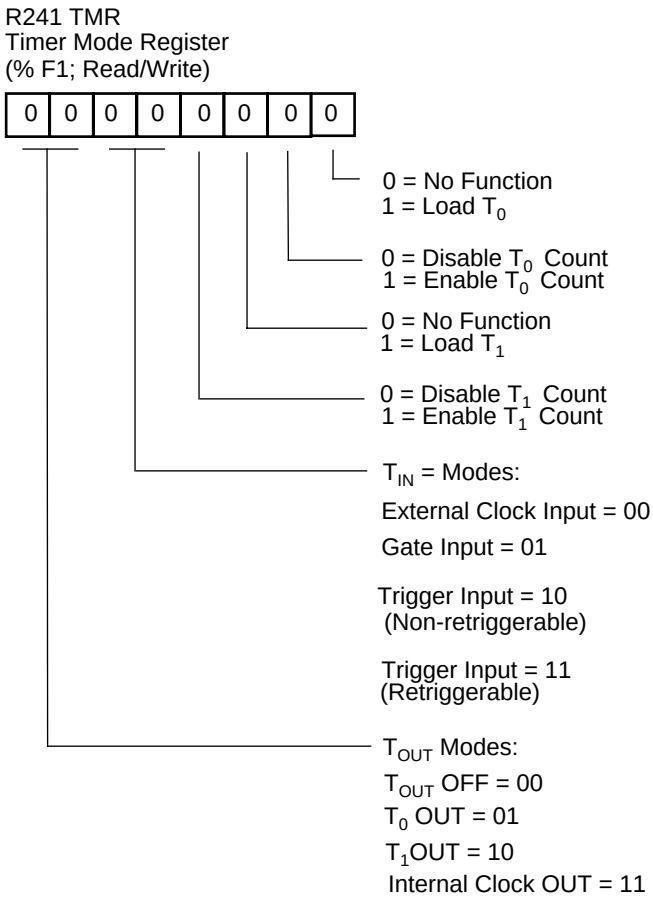


Figure 6-22. Timer Mode Register Reset

Totally Logical

CHAPTER 7
INTERRUPTS

7.1 INTRODUCTION

The Z8 MCU® allows 6 different interrupts from a variety of sources; up to four external inputs, the on-chip Counter/Timer(s), software, and serial I/O peripherals. These interrupts can be masked and their priorities set by using the Interrupt Mask and the Interrupt Priority Registers. All six interrupts can be globally disabled by resetting the master Interrupt Enable, bit 7 in the Interrupt Mask Register, with a Disable Interrupt (DI) instruction. Interrupts are globally enabled by setting bit 7 with an Enable Interrupt (EI) instruction.

There are three interrupt control registers: the Interrupt Request Register (IRQ), the Interrupt Mask register (IMR), and the Interrupt Priority Register (IPR). Figure 7-1 shows addresses and identifiers for the interrupt control registers. Figure 7-2 is a block diagram showing the Interrupt Mask and Interrupt Priority logic.

The Z8 MCU family supports both vectored and polled interrupt handling. Details on vectored and polled interrupts can be found later in this chapter.

Register	HEX	Identifier
Interrupt Mask	FBH	IMR
Interrupt Request	FAH	IRQ
Interrupt Priority	F9H	IPR

Figure 7-1. Interrupt Control Registers

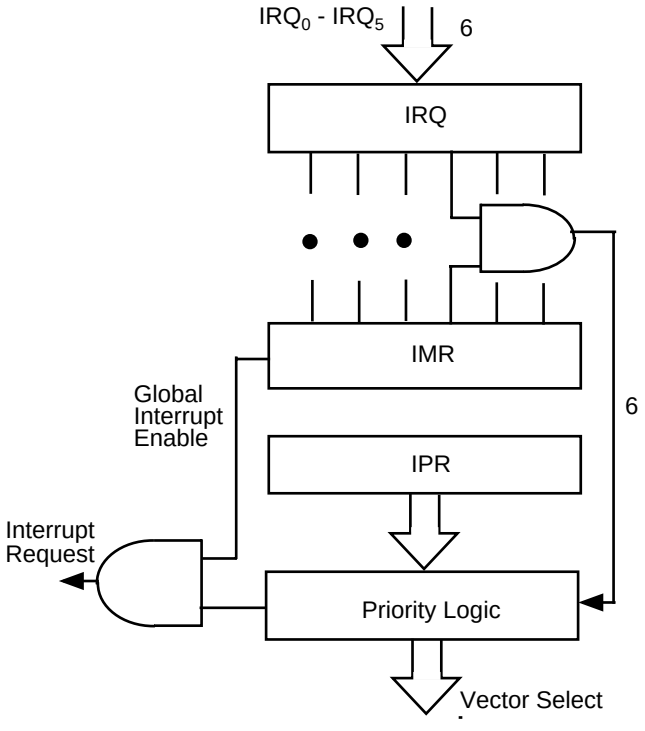


Figure 7-2. Interrupt Block Diagram

Note: See the selected Z8 MCU's product specification for the exact interrupt sources supported.

7.2 INTERRUPT SOURCES

Table 7-1 presents the interrupt types, sources, and vectors available in the Z8 family of processors.

Table 7-1. Interrupt Types, Sources, and Vectors *

Name	Sources	Vector Location	Comments
IRQ_0	DAV_0, IRQ_0, Comparator	0,1	External ($P3_2$), Edge Triggered; Internal
IRQ_1	DAV_1, IRQ_1	2,3	External ($P3_3$), Edge Triggered; Internal
IRQ_2	DAV_2, IRQ_2, TIN, Comparator	4,5	External ($P3_1$), Edge Triggered; Internal
	IRQ_3	6,7	External ($P3_0$) or ($P3_2$), Edge Triggered; Internal
	Serial In	6,7	Internal
	T_0	8,9	Internal
	Serial Out	8,9	Internal
IRQ_5	T_1	10,11	Internal

7.2.1 External Interrupt Sources

External sources involve interrupt request lines IRQ0-IRQ3. IRQ0, IRQ1, and IRQ2 can be generated by a transition on the corresponding Port 3 pin (P32, P33, and P31 correspond to IRQ0, IRQ1, and IRQ2, respectively).

Figure 7-3 is a block diagram for interrupt sources IRQ0, IRQ1, and IRQ2.

Note: The interrupt sources and trigger conditions are device dependent. See the device product specification to determine available sources (internal and external), triggering edge options, and exact programming details.

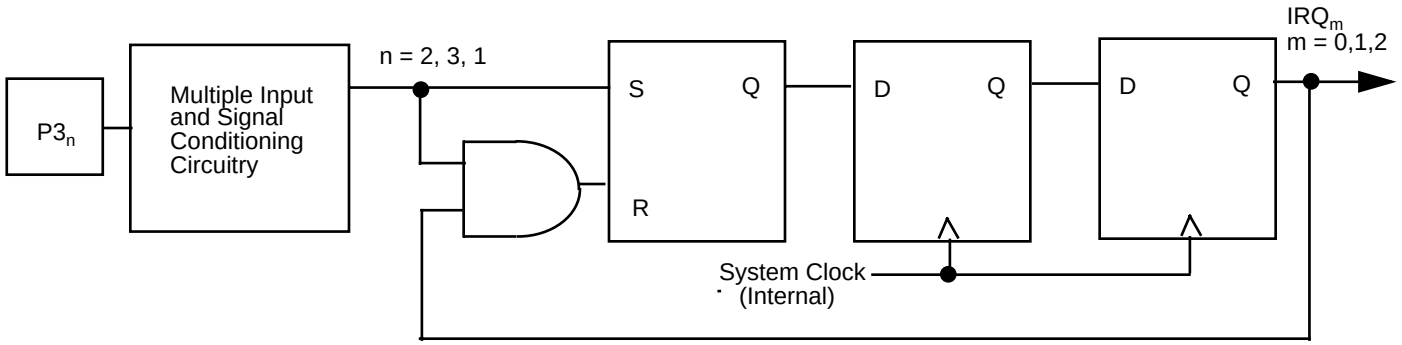


Figure 7-3. Interrupt Sources IRQ0-IRQ2 Block Diagram

When the Port 3 pin (P31, P32, or P33) transitions, the first flip-flop is set. The next two flip-flops synchronize the request to the internal clock and delay it by two internal clock periods. The output of the last flip-flop (IRQ0, IRQ1, or IRQ2) goes to the corresponding Interrupt Request Register.

IRQ3 can be generated from an external source only if Serial In is not enabled. Otherwise, its source is internal. The external request is generated by a Low edge signal on P30 as shown in Figure 7-4. Again, the external request is synchronized and delayed before reaching IRQ3. Some Z8 products replace P30 with P32 as the external source for IRQ3. In this case, IRQ3 interrupt generation follows the logic as illustrated in Figure 7-3.

Note: Although interrupts are edge triggered, minimum interrupt request Low and High times must be observed for proper operation. See the device product specification for exact timing requirements on external interrupt requests (T_WIL, T_WIH).

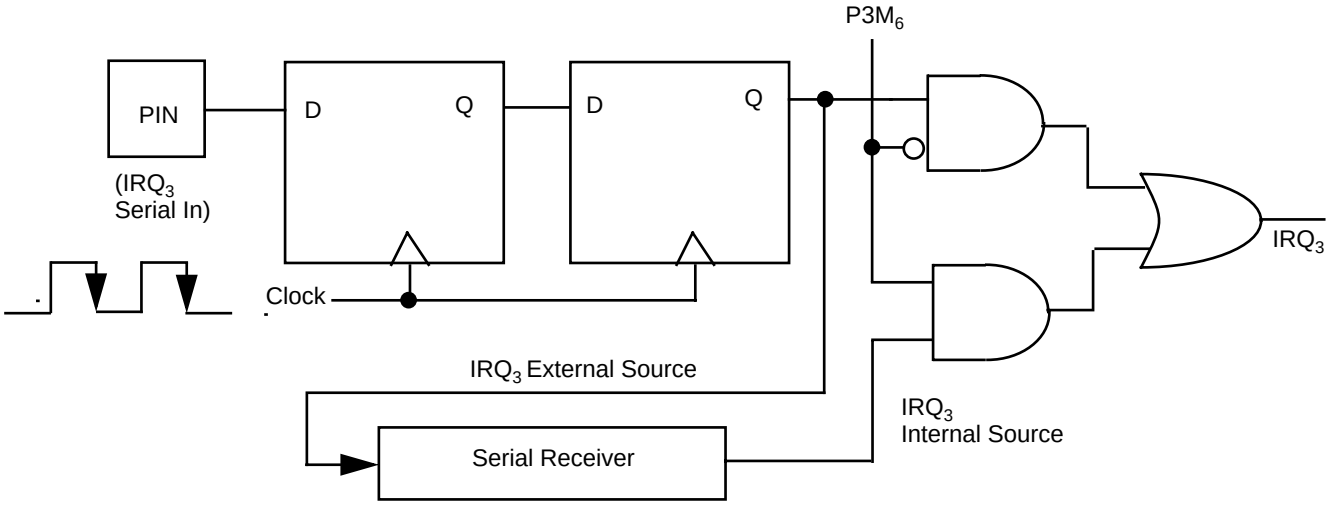


Figure 7-4. Interrupt Source IRQ3 Block Diagram

7.2.2 Internal Interrupt Sources

Internal sources involve interrupt requests IRQ0, IRQ2, IRQ3, IRQ4, and IRQ5. Internal sources are ORed with the external sources, so either an internal or external source can trigger the interrupt. Internal interrupt sources and trigger conditions are device dependent.

See the device product specification to determine available sources, triggering edge options, and exact programming details. For more details on the internal interrupt sources, refer to the chapters describing the Counter/Timer, I/O ports, and Serial I/O.

7.3 INTERRUPT REQUEST REGISTER LOGIC AND TIMING

Figure 7-5 shows the logic diagram for the Interrupt Request (IRQ) Register. The leading edge of the request will set the first flip-flop, that will remain set until interrupt requests are sampled.

Requests are sampled internally during the last clock cycle before an opcode fetch (Figure 7-6). External requests are sampled two internal clocks earlier, due to the synchronizing flip-flops shown in Figures 7-3 and 7-4.

At sample time the request is transferred to the second flip-flop in Figure 7-5, that drives the interrupt mask and priority logic. When an interrupt cycle occurs, this flip-flop will be reset only for the highest priority level that is enabled.

The user has direct access to the second flip-flop by reading and writing the IRQ Register. IRQ is read by specifying it as the source register of an instruction and written by specifying it as the destination register.

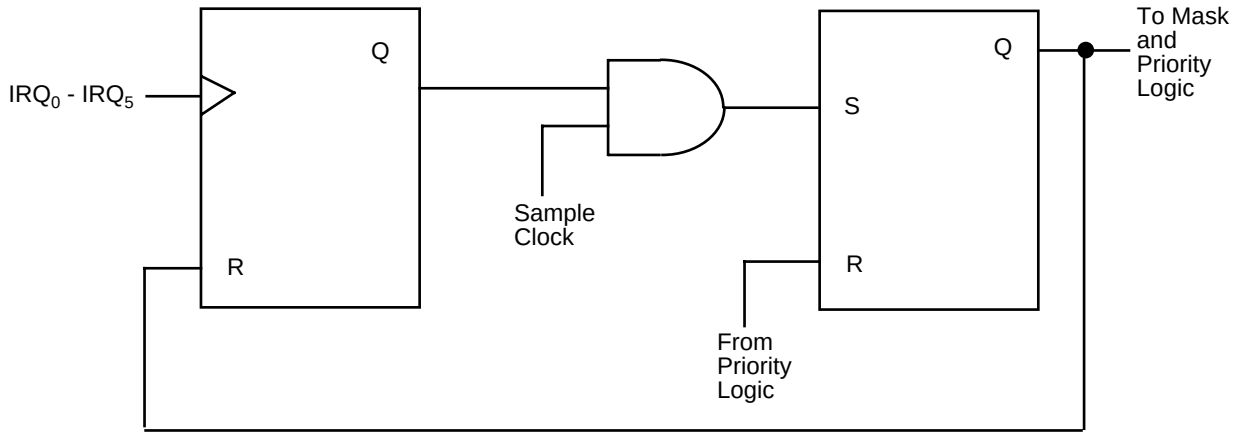


Figure 7-5. IRQ Register Logic

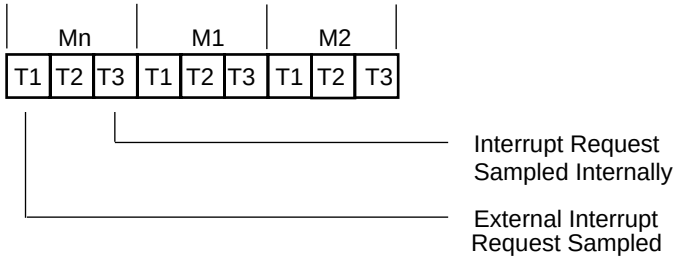


Figure 7-6. Interrupt Request Timing

7.4 INTERRUPT INITIALIZATION

After reset, all interrupts are disabled and must be initialized before vectored or polled interrupt processing can begin. The Interrupt Priority Register (IPR), Interrupt Mask Register (IMR), and Interrupt Request Register (IRQ) must be initialized, in that order, to start the interrupt process.

7.4.1 Interrupt Priority Register (IPR) Initialization

IPR (Figure 7-7) is a write-only register that sets priorities for the vectored interrupts in order to resolve simultaneous interrupt requests. (There are 48 sequence possibilities for interrupts.) The six interrupt levels IRQ0-IRQ5 are divided into three groups of two interrupt requests each. One group contains IRQ3 and IRQ5. The second group contains IRQ0 and IRQ2, while the third group contains IRQ1 and IRQ4.

Priorities can be set both within and between groups as shown in Tables 7-2 and 7-3. Bits 1, 2, and 5 define the priority of the individual members within the three groups. Bits 0, 3, and 4 are encoded to define six priority orders between the three groups. Bits 6 and 7 are reserved.

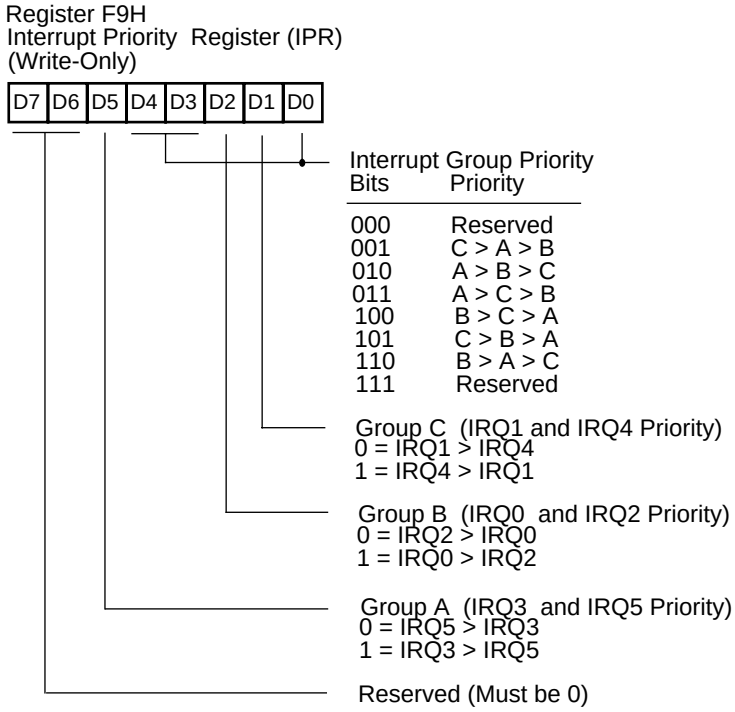


Figure 7-7. Interrupt Priority Register

Table 7-2. Interrupt Priority

Group	Bit	Value	Priority Highest	Priority Lowest
C	Bit 1	0	IRQ1	IRQ4
		1	IRQ4	IRQ1
B	Bit 2	0	IRQ2	IRQ0
		1	IRQ0	IRQ2
A	Bit 5	0	IRQ5	IRQ3
		1	IRQ3	IRQ5

Table 7-3. Interrupt Group Priority

Bit Pattern			Group Priority		
Bit 4	Bit 3	Bit 0	High	Medium	Low
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		

7.4 INTERRUPT INITIALIZATION (Continued)

7.4.2 Interrupt Mask Register (IMR) Initialization

IMR individually or globally enables or disables the six interrupt requests (Figure 7-8). When bit 0 to bit 5 are set to 1, the corresponding interrupt requests are enabled. Bit 7 is the master enable and must be set before any of the individual interrupt requests can be recognized. Resetting bit 7 globally disables all the interrupt requests. Bit 7 is set and reset by the EI and DI instructions. It is automatically reset during an interrupt service routine and set following the execution of an Interrupt Return (IRET) instruction.

Note: Bit 7 must be reset by the DI instruction before the contents of the Interrupt Mask Register or the Interrupt Priority Register are changed except:

- Immediately after a hardware reset.
- Immediately after executing an interrupt service routine and before IMR bit 7 has been set by any instruction.

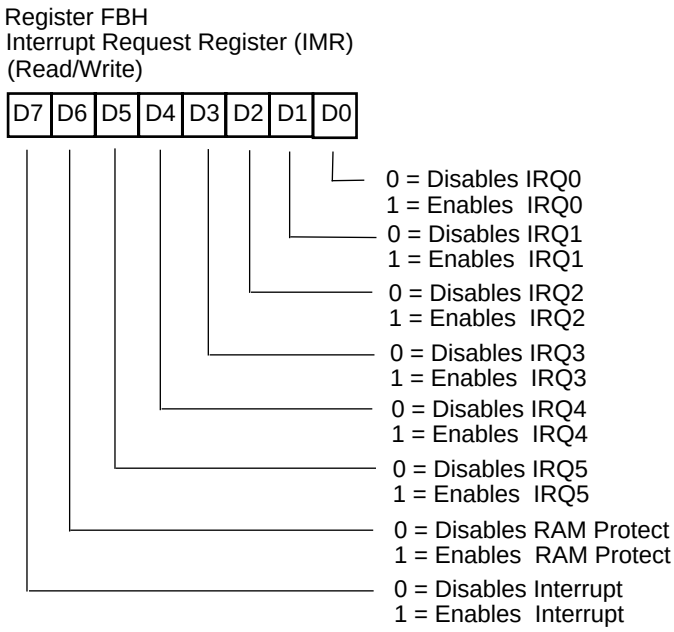


Figure 7-8. Interrupt Mask Register

Note: The RAM Protect option is selected at ROM mask submission time or at EPROM program time. If not selected or not an available option, this bit is reserved and must be 0.

7.4.3 Interrupt Request (IRQ) Register Initialization

IRQ (Figure 7-9) is a read/write register that stores the interrupt requests for both vectored and polled interrupts. When an interrupt is made on any of the six, the corresponding bit position in the register is set to 1. Bit 0 to bit 5 are assigned to interrupt requests IRQ0 to IRQ5, respectively.

Whenever Power-On Reset (POR) is executed, the IRQ resister is reset to 00H and disabled. Before the IRQ register will accept requests, it must be enabled by executing an ENABLE INTERRUPTS (EI) instruction.

Note: Setting the Global Interrupt Enable bit in the Interrupt Mask Register (IMR, bit 7) will not enable the IRQ. Execution of the EI instruction is required (Figure 7-10).

For polled processing, IRQ must still be initialized by an EI instruction.

To properly initialize the IRQ register, the following code is provided:

```
CLR     IMR     //make sure disabled vectored interrupts
EI              //enable IRQ register otherwise read
                only.
                //not needed if interrupts were
                previously enabled.
DI              //disable interrupt heading.
```

Note: IRQ is always cleared to 00Hex and is read only until the 1st EI instruction which enables the IRQ to be read/write.

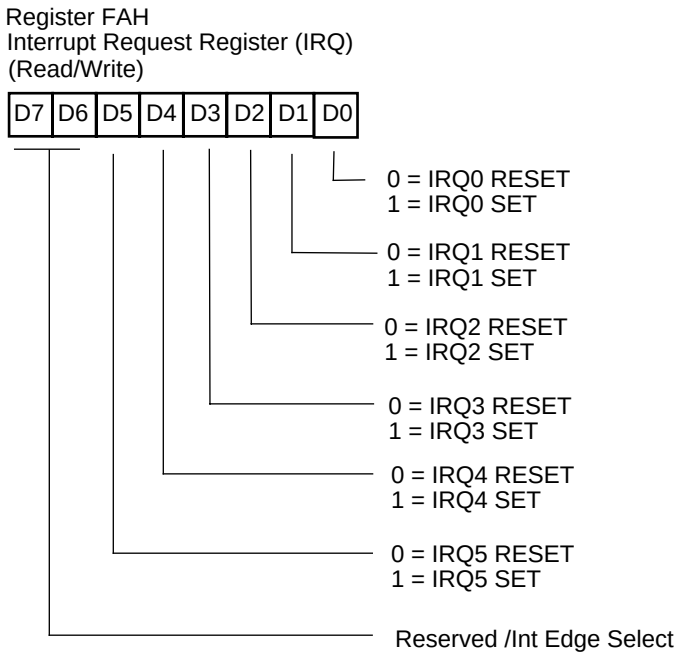


Figure 7-9. Interrupt Request Register

7.4 INTERRUPT INITIALIZATION (Continued)

IMR is cleared before the IRQ enabling sequence to insure no unexpected interrupts occur when EI is executed. This code sequence should be executed prior to programming the application required values for IPR and IMR.

Note: IRQ bits 6 and 7 are device dependent. When reserved, the bits are not used and will return a 0 when read. When used as the Interrupt Edge select bits, the configuration options are as show in Table 7-4.

Table 7-4. IRQ Register Configuration

IRQ		Interrupt Edge	
D7	**D6**	**P31**	**P32**
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F = Falling Edge
R = Rising Edge

The proper sequence for programming the interrupt edge select bits is (assumes IPR and IMR have been previously initialized):

```
DI                      ;Inhibit all interrupts
                        until input edges are
                        configured
OR      IRQ,#XX 000000B ;Configure interrupt
                        do not disturb
                        edges as needed -
                        IRQ 0-5.
EI                      ;Re-enable interrupts.
```

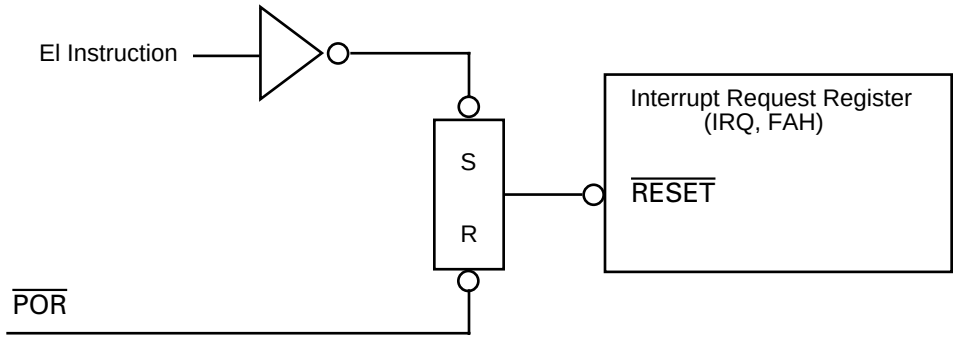


Figure 7-10. IRQ Reset Functional Logic Diagram

7.5 IRQ SOFTWARE INTERRUPT GENERATION

IRQ can be used to generate software interrupts by specifying IRQ as the destination of any instruction referencing the Z8 Standard Register File. These Software Interrupts (SWI) are controlled in the same manner as hardware generated requests (in other words, the IPR and the IMR control the priority and enabling of each SWI level).

To generate a SWI, the desired request bit in the IRQ is set as follows:

```
ORIRQ,          #NUMBER
```

where the immediate data, NUMBER, has a 1 in the bit position corresponding to the level of the SWI desired. For example, if an SWI is desired on IRQ5, NUMBER would have a 1 in bit 5:

```
OR              IRQ, #00100000B
```

With this instruction, if the interrupt system is globally enabled, IRQ5 is enabled, and there are no higher priority pending requests, control is transferred to the service routine pointed to by the IRQ5 vector.

7.6 VECTORED PROCESSING

Each Z8 interrupt level has its own vector. When an interrupt occurs, control passes to the service routine pointed to by the interrupt's vector location in program memory. The sequence of events for vectored interrupts is as follows:

- PUSH PC Low Byte on Stack
- PUSH PC High Byte on Stack
- PUSH FLAGS on Stack
- Fetch High Byte of Vector
- Fetch Low Byte of Vector
- Branch to Service Routine specified by Vector

Figures 7-11 and 7-12 show the vectored interrupt operation.

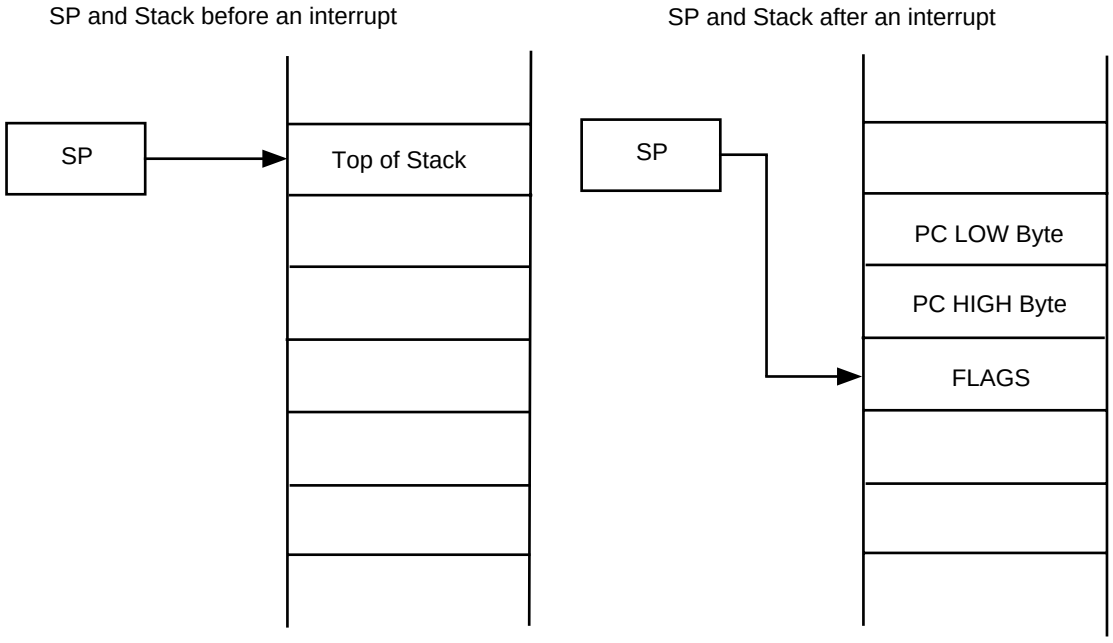


Figure 7-11. Effects of an Interrupt on the STACK

7.6 VECTORED PROCESSING (Continued)

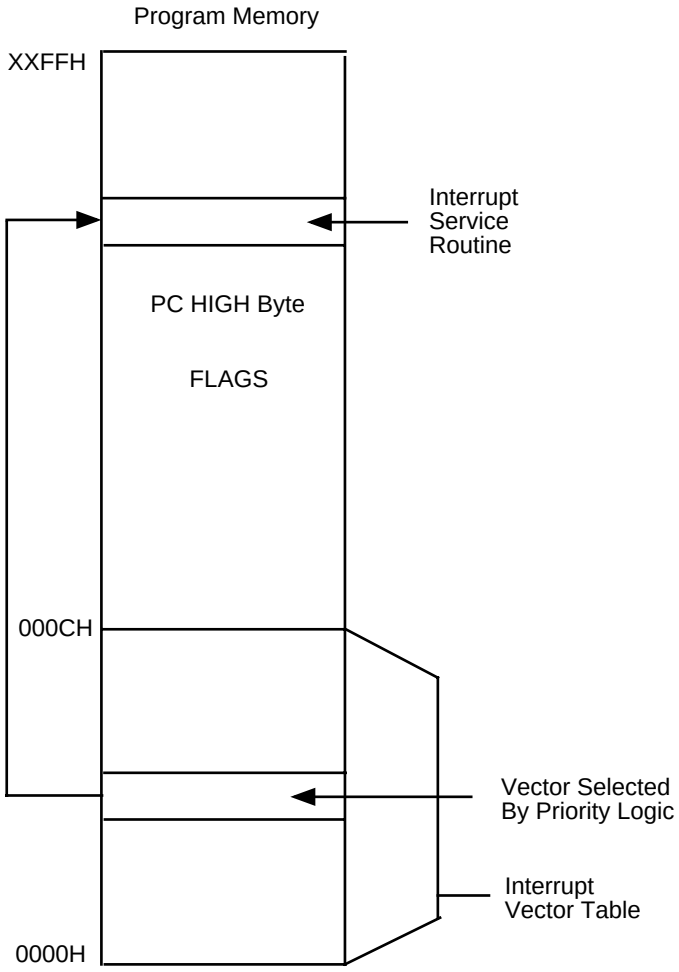


Figure 7-12. Interrupt Vectoring

7.6.1 Vectored Interrupt Cycle Timing

The interrupt acknowledge cycle time is 24 internal clock cycles and is shown in Figure 7-13. In addition, two internal clock cycles are required for the synchronizing flip-flops. The maximum interrupt recognition time is equal to the number of clock cycles required for the longest executing instruction present in the user program (assumes worst case condition of interrupt sampling, Figure 7-6, just prior to the interrupt occurrence). To calculate the worst case interrupt latency (maximum time required from interrupt generation to fetch of the first instruction of the interrupt service routine), sum these components:

Worst Case Interrupt Latency ≈ 24 INT CLK (interrupt acknowledge time) + # T_PC of longest instruction present in the user's application program + $2T_PC$ (internal synchronization time).

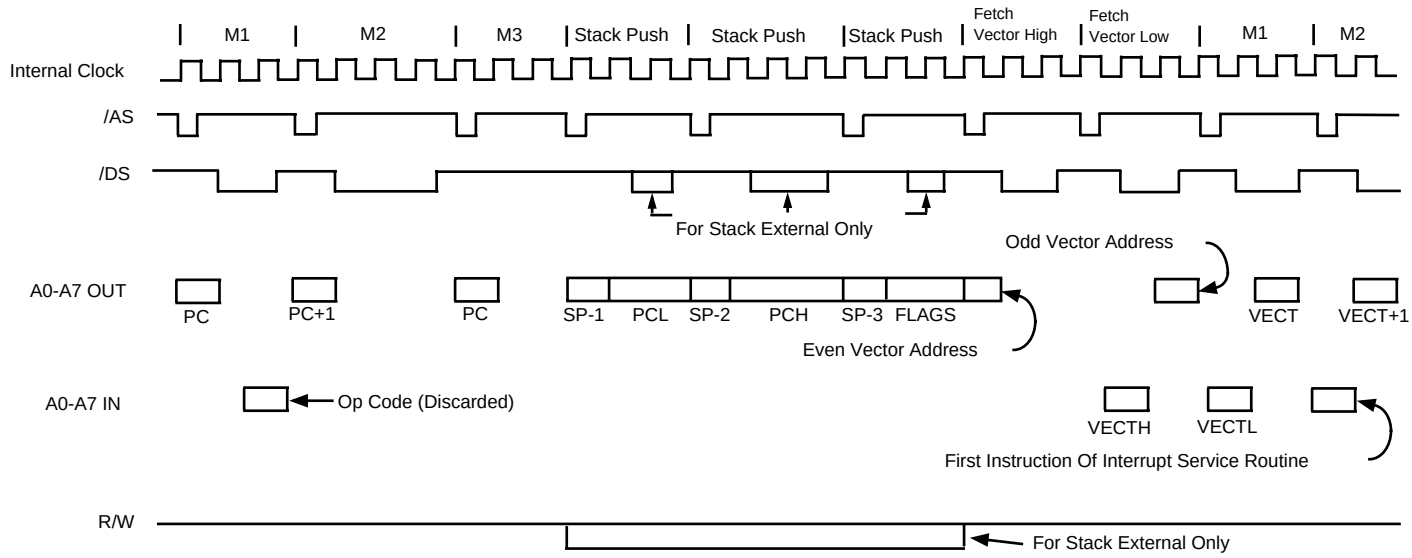


Figure 7-13. Z8 Interrupt Acknowledge Timing

7.6.2 Nesting of Vectored Interrupts

Nesting of vectored interrupts allows higher priority requests to interrupt a lower priority request. To initiate vectored interrupt nesting, do the following during the interrupt service routine:

- Push the old IMR on the stack.
- Load IMR with a new mask to disable lower priority interrupts.
- Execute EI instruction.
- Proceed with interrupt processing.
- After processing is complete, execute DI instruction.
- Restore the IMR to its original value by returning the previous mask from the stack.
- Execute IRET.

Depending on the application, some simplification of the above procedure may be possible.

7.7 POLLED PROCESSING

Polled interrupt processing is supported by masking off the IRQ to be polled. This is accomplished by clearing the corresponding bits in the IMR.

To enable any interrupt, first the interrupt mechanism must be engaged with an EI instruction. If only polled interrupts are to be serviced, execute:

```
EI      ;Enable interrupt mechanism
DI      ;Disable vectored interrupts.
```

To initiate polled processing, check the bits of interest in the IRQ using the Test Under Mask (TM) instruction. If the bit is set, call or branch to the service routine. The service routine services the request, resets its Request Bit in the IRQ, and branches or returns back to the main program. An example of a polling routine is as follows:

```
        TM      IRQ, #MASKA     ;Test for request
        JR      Z, NEXT         ;If no request go to
                                NEXT
        CALL SERVICE            ;If request is there, then
                                ;service it
NEXT:
        .
        .
        .
SERVICE:                        ;Process Request
        .
        .
        .
        AND IRQ, #MASKB         ;Clear Request Bit
        RET                     ;Return to next
```

In this example, if IRQ2 is being polled, MASKA will be 00000100B and MASKB will be 11111011B.

7.8 RESET CONDITIONS

Upon reset, all bits in IPR are undefined.

In IMR, bit 7 is 0 and bits 0-6 are undefined. The IRQ register is reset and held in that state until an enable interrupt (EI) instruction is executed.

Totally Logical

CHAPTER 8
POWER-DOWN MODES

8.1 INTRODUCTION

In addition to the standard RUN mode, the Z8 MCU® supports two Power-Down modes to minimize device current consumption. The two modes supported are HALT and STOP.

8.2 HALT MODE OPERATION

The HALT mode suspends instruction execution and turns off the internal CPU clock. The on-chip oscillator circuit remains active so the internal clock continues to run and is applied to the Counter/Timer(s) and interrupt logic.

To enter the HALT mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the application program must execute a NOP instruction (opcode = FFH) immediately before the HALT instruction (opcode 7FH), that is,

```
FF    NOP     ;clear the instruction pipeline
7F    HALT    ;enter HALT mode
```

The HALT mode is exited by interrupts, either externally or internally generated. Upon completion of the interrupt service routine, the user program continues from the instruction after HALT.

The HALT mode may also be exited via a POR/RESET activation or a Watch-Dog Timer (WDT) timeout. (See the product data sheet for WDT availability). In this case, program execution will restart at the reset restart address 000CH.

To further reduce power consumption in the HALT mode, some Z8 family devices allow dynamic internal clock scaling. Clock scaling may be accomplished on the fly by reprogramming bit 0 and/or bit1 of the STOP-Mode Recovery register (SMR). See Figure 8-1.

Note: Internal clock scaling directly effects Counter/Timer operation — adjustment of the prescaler and downcounter values may be required. To determine the actual HALT mode current (I_{CC1}) value for the various optional modes available, see the related Z8 device's product specification.

8.3 STOP MODE OPERATION

The STOP mode provides the lowest possible device standby current. This instruction turns off the on-chip oscillator and internal system clock.

To enter the STOP mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the application program must execute a NOP instruction (opcode=FFH) immediately before the STOP instruction (opcode=6FH), that is,

```
FF    NOP     ;clear the instruction pipeline
6F    STOP    ;enter STOP mode
```

The STOP mode is exited by any one of the following resets: Power-On Reset activation, WDT time out (if available), or a STOP-Mode Recovery source. Upon reset generation, the processor will always restart the application program at address 000CH.

POR/RESET activation is present on all Z8 devices and is implemented as a reset pin and/or an on-chip power on reset circuit.

Some Z8 devices allow for the on-chip WDT to run in the STOP mode. If so activated, the WDT timeout will generate a reset some fixed time period after entering the STOP mode.

Note: STOP-Mode Recovery by the WDT will increase the STOP mode standby current (I_{CC2}). This is due to the WDT clock and divider circuitry that is now enabled and running to support this recovery mode. See the product data sheet for actual I_{CC2} values.

All Z8 devices provide some form of dedicated STOP-Mode Recovery (SMR) circuitry. Two SMR methods are implemented — a single fixed input pin or a flexible, programmable set of inputs. The selected Z8 device product specification should be reviewed to determine the SMR options available for use.

Note: For devices that support SPI, the slave mode compare feature also serves as a SMR source.

In the simple case, a low level applied to input pin P27 will trigger a SMR. To use this mode, pin P27 (I/O Port 2, bit 7) must be configured as an input before the STOP mode is entered. The low level on P27 must meet a minimum pulse width T_{WSM}. (See the product data sheet) to trigger the device reset mode). Some Z8 devices provide multiple SMR input sources. The desired SMR source is selected via the SMR Register.

Note: Use of specialized SMR modes (P2.7 input or SMR register based) or the WDT timeout (only when in the STOP mode) provide a unique reset operation. Some control registers are initialized differently for a SMR/WDT triggered POR than a standard reset operation. See the product specification (register file map) for exact details.

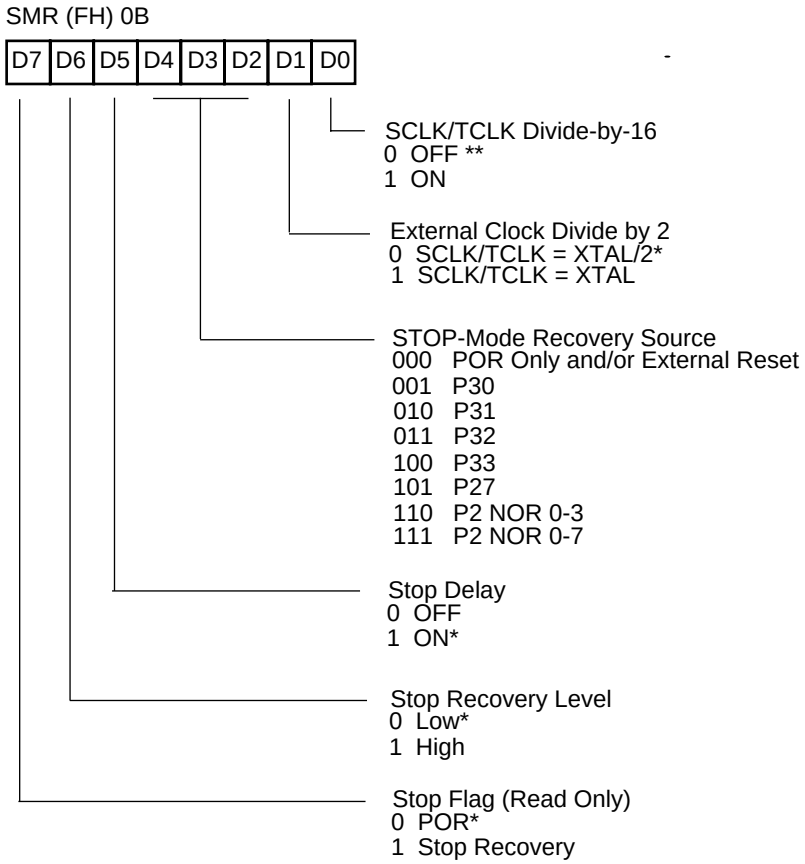
To determine the actual STOP mode current (I_{CC2}) value for the optional SMR modes available, see the selected Z8 device's product data sheet.

Note: The STOP mode current (I_{CC2}) will be minimized when:

- V_{CC} is at the low end of the devices operating range.
- WDT is off in the STOP mode.
- Output current sourcing is minimized.
- All inputs (digital and analog) are at the low or high rail voltages.

8.4 STOP-MODE RECOVERY REGISTER

This register selects the clock divide value and determines the mode of STOP-Mode Recovery (Figure 8-1). All bits are Write-Only, except bit 7, that is Read-Only. Bit 7 is a flag bit that is hardware set on the condition of STOP recovery and reset by a power-on cycle. Bit 6 controls whether a low level or a high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4, of the SMR register, specify the source of the STOP-Mode Recovery signal. Bits 0 and 1 control internal clock divider circuitry. The SMR is located in Bank F of the Expanded Register File at address 0BH.



* Default setting after RESET.

** Default setting after RESET and STOP-Mode Recovery.

Figure 8-1. STOP-Mode Recovery Register (Write-Only Except Bit D7, Which Is Read-Only)

Note: The SMR register is available in select Z8 MCU products. Refer to the device product specification to determine SMR options available.

SCLK/TCLK Divide-by-16 Select (DO). This bit of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

External Clock Divide-by-Two (D1). This bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, the System Clock (SCLK) and Timer Clock (TCLK) are equal to the external clock frequency divided by two. The SCLK/TCLK is equal to the external clock frequency when this bit is set (D1=1). Using this bit together with D7 of PCON helps further lower EMI (D7 (PCON) =0, D1 (SMR) =1). The default setting is zero.

8.4 STOP-MODE RECOVERY REGISTER (Continued)

STOP-Mode Recovery Source (D2, D3, and D4). These three bits of the SMR specify the wake-up source of the STOP recovery and (Table 8-1 and Figures 8-2).

Table 8-1. STOP-Mode Recovery Source

SMR: 432 D4	D3	D2	Operation Description of Action
0	0	0	POR and/or external reset recovery
0	0	1	P30 transition
0	1	0	P31 transition (not in Analog Mode)
0	1	1	P32 transition (not in Analog Mode)
1	0	0	P33 transition (not in Analog Mode)
1	0	1	P27 transition
1	1	0	Logical NOR of P20 through P23
1	1	1	Logical NOR of P20 through P27

STOP-Mode Recovery Delay Select (D5). This bit, if High, enables the T_{POR} /RESET delay after Stop-Mode Recovery. The default configuration of this bit is 1. If the "fast" wake up is selected, the Stop-Mode Recovery source is kept active for at least 5 TpC.

STOP-Mode Recovery Edge Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the Z8 from STOP mode. A 0 indicates low-level recovery. The default is 0 on POR (Figure 8-2).

Cold or Warm Start (D7). This bit is set by the device upon entering STOP mode. A 0 in this bit (cold) indicates that the device reset by POR/WDT RESET. A 1 in this bit (warm) indicates that the device awakens by a SMR source.

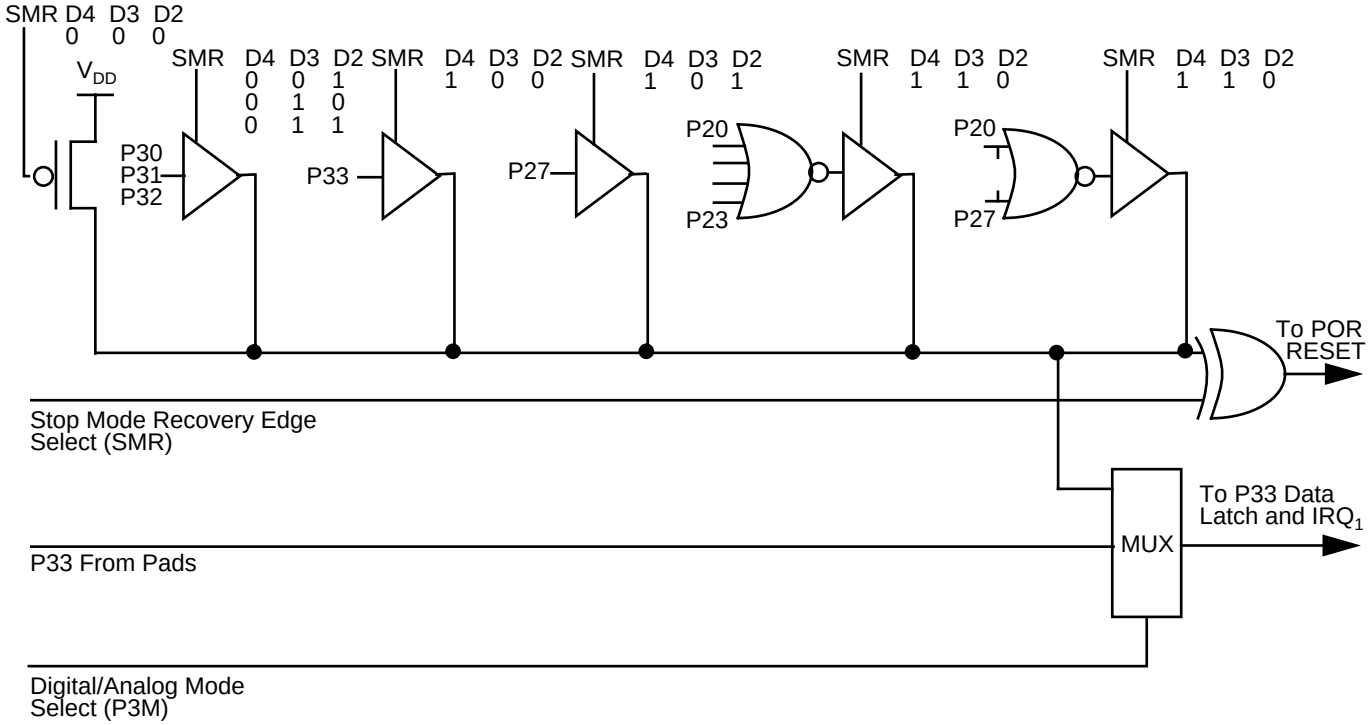


Figure 8-2. STOP-Mode Recovery Source

Note: If P31, P32, or P33 are to be used for a SMR source, the digital mode of operation must be selected prior to entering the STOP Mode.

Totally Logical

CHAPTER 9
SERIAL I/O

9.1 UART INTRODUCTION

Select Z8 MCU® microcontrollers contain an on-board full-duplex Universal Asynchronous Receiver/Transmitter (UART) for data communications. The UART consists of a Serial I/O Register (SIO) located at address F0H, and its associated control logic (Figure 9-1). The SIO is actually two registers, the receiver buffer and the transmitter buffer, which are used in conjunction with Counter/Timer T0 and Port 3 I/O lines P30 (input) and P37 (output). Counter/Timer T0 provides the clock input for control of the data rates.

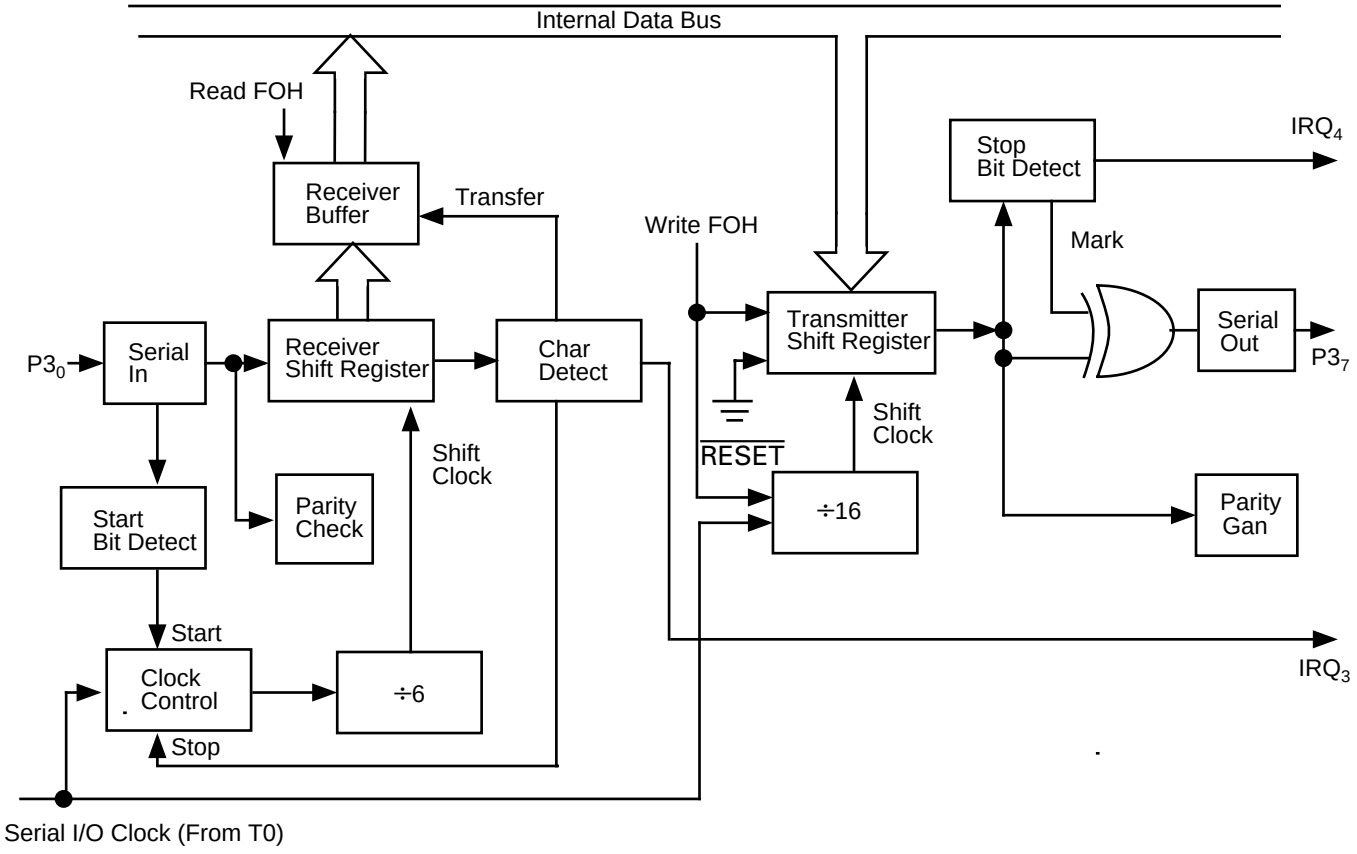


Figure 9-1. UART Block Diagram

Configuration of the UART is controlled by the Port 3 Mode Register (P3M) located at address F7H. The Z8 always transmits eight bits between the start and stop bits (eight Data Bits or seven Data Bits and one Parity Bit). Odd parity generation and detection is supported.

The SIO Register and its associated Mode Control Registers are mapped into the Standard Z8 Register File as shown in Table 9-1. The organization allows the software to access the UART as general-purpose registers, eliminating the need for special instructions.

Table 9-1. UART Register Map

Register Name	Identifier	Hex Address
Port 3 Mode	P3M	F7
T0 Prescaler	PRE0	F5
Timer/Counter0	T0	F4
Timer Mode	TMR	F1
UART	SIO	F0

9.2 UART BIT-RATE GENERATION

When Port 3 Mode Register bit 6 is set to 1, the UART is enabled and T0 automatically becomes the bit rate generator (Figure 9-2). The end-of-count signal of T0 no longer generates Interrupt Request IRQ4. Instead, the signal is used as the input to the divide-by-16 counters (one each for the receiver and the transmitter) that clock the data stream.

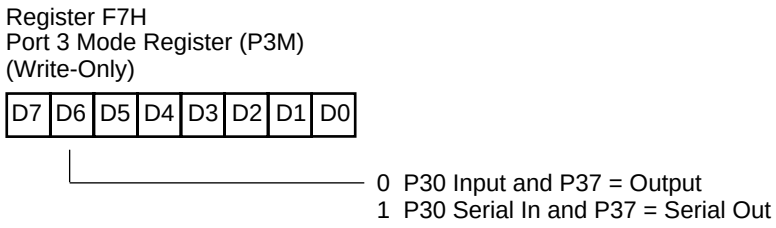


Figure 9-2. Port 3 Mode Register (P3M) and Bit-Rate Generation

The divide chain that generates the bit rate is shown in Figure 9-3. The bit rate is given by the following equation:

Bit Rate = XTAL Frequency/(2 x 4 x p x t x 16)

where p and t are the initial values in Prescaler0 and Counter/Timer0, respectively. The final divide-by-16 is required since T0 runs at 16 times the bit rate in order to synchronize on the incoming data.

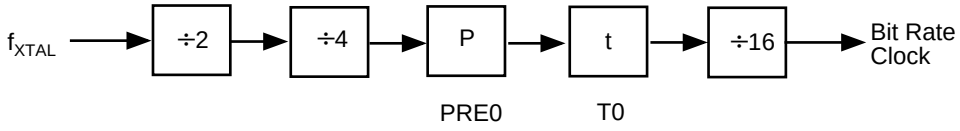


Figure 9-3. Bit Rate Divide Chain

To configure the Z8 for a specific bit rate, appropriate values as determined by the above equation must be loaded into registers PRE0 (F5H) and T0 (F4H). PRE0 also controls the counting mode for T0 and should therefore be set to the Continuous Mode (D0 = 1).

For example, given an input clock frequency (XTAL) of 11.9808 MHz and a selected bit rate of 1200 bits per second, the equation is satisfied by p = 39 and t = 2. Counter/Timer T0 should be set to 02H. With T0 in Continuous Mode, the value of PRE0 becomes 9DH (Figure 9-4).

Table 9-2 lists several commonly used bit rates and the values of XTAL, p, and t required to derive them. This list is presented for convenience and is not intended to be exhaustive.

Table 9-2. Bit Rates

Bit Rate	7,3728		7,9872		9,8304		11,0592		11,6736		11,9808		12,2880	
	p	t	p	t	p	t	p	t	p	t	p	t	p	t
19200	3	1	–	–	4	1	–	–	–	–	–	–	5	1
9600	3	2	–	–	4	2	9	1	–	–	–	–	5	2
4800	3	4	13	1	4	4	9	2	19	1	–	–	5	4
2400	3	8	13	2	4	8	9	4	19	2	39	1	5	8
1200	3	16	13	4	4	16	9	8	19	4	39	2	5	16
600	3	32	13	8	4	32	9	16	19	8	39	4	5	32
300	3	64	13	16	4	64	9	32	19	16	39	8	5	64
150	3	128	13	32	4	128	9	64	19	32	39	16	5	128
110	3	175	3	189	4	175	5	157	4	207	17	50	8	109

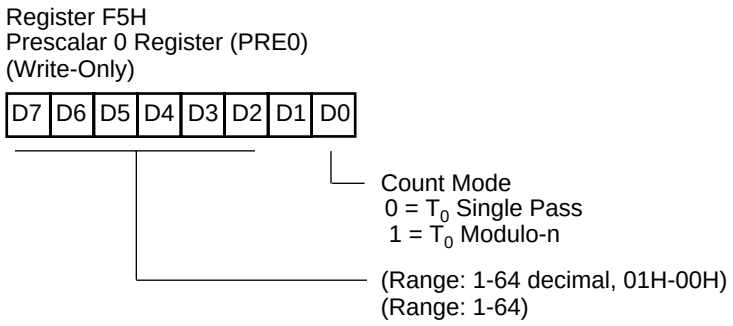


Figure 9-4. Prescaler 0 Register (PRE0) Bit-Rate Generation

The bit rate generator is started by setting the Timer Mode Register (TMR) (F1H) bit 1 and bit 0 both to 1 (Figure 9-5). This transfers the contents of the Prescaler 0 Register and Counter/Timer0 Register to their corresponding down counters. In addition, counting is enabled so that UART operations begin.

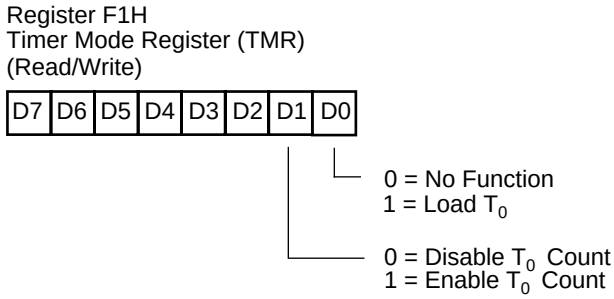


Figure 9-5. Timer Mode Register (TMR) Bit Rate Generation

9.3 UART RECEIVER OPERATION

The receiver consists of a receiver buffer (SIO Register [F0H]), a serial-in, parallel-out shift register, parity checking, and data synchronizing logic. The receiver block diagram is shown as part of Figure 9-1.

9.3.1 Receiver Shift Register

After a hardware reset or after a character has been received, the Receiver Shift Register is initialized to all 1s and the shift clock is stopped. Serial data, input through Port 3 bit 0, is synchronized to the internal clock by two D-type flip-flops before being input to the Shift Register and the start bit detection circuitry.

The start bit detection circuitry monitors the incoming data stream, looking for a start bit (a High-to-Low input transition). When a start bit is detected, the shift clock logic is enabled. The T0 input is divided-by-16 and, when the count equals eight, the divider outputs a shift clock. This clock shifts the start bit into the Receiver Shift Register at the center of the bit time. Before the shift actually occurs, the input is rechecked to ensure that the start bit is valid. If the detected start bit is false, the receiver is reset and the process of looking for a start bit is repeated. If the start bit is valid, the data is shifted into the Shift Register every sixteen counts until a full character is assembled (Figure 9-6).

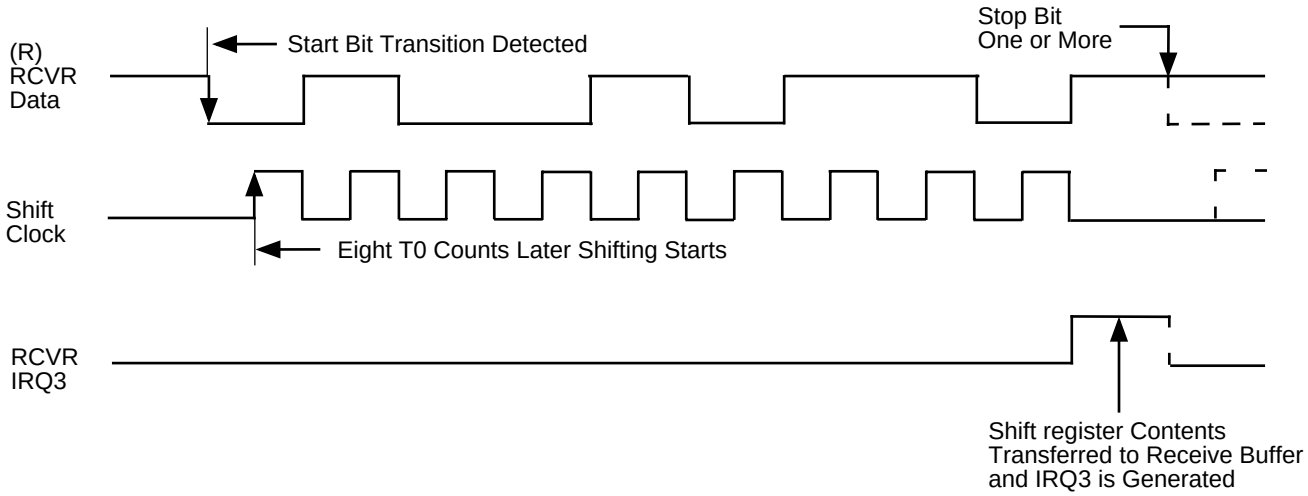


Figure 9-6. Receiver Timing

After a full character has been assembled in the receiver's buffer, SIO Register (F0H), Interrupt Request IRQ3 is generated. The shift clock is stopped and the Shift Register reset to all 1s. The start bit detection circuitry begins monitoring the data input for the next start bit. This cycle allows the receiver to synchronize on the center of the bit time for each incoming character.

9.3.2 Overwrites

Although the receiver is single buffered, it is not protected from being overwritten, so the software must read the SIO Register (F0H) within one character time after the interrupt request (IRQ3). The Z8 does not have a flag to indicate this overrun condition. If polling is used, the IRQ3 bit in the Interrupt Request Register must be reset by software.

9.3.3 Framing Errors

Framing error detection is not supported by the receiver hardware, but by responding to the interrupt request within one character bit time, the software can test for a stop bit on P30. Port 3 bits are always readable, which facilitates break detection. For example, if a null character is received, testing P30 results in a 0 being read.

9.3.4 Parity

The data format supported by the receiver must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the data received will be replaced by a Parity Error Flag. A parity error sets bit 7 to 1, otherwise, bit D7 is set to 0. Figure 9-7 shows these data formats.

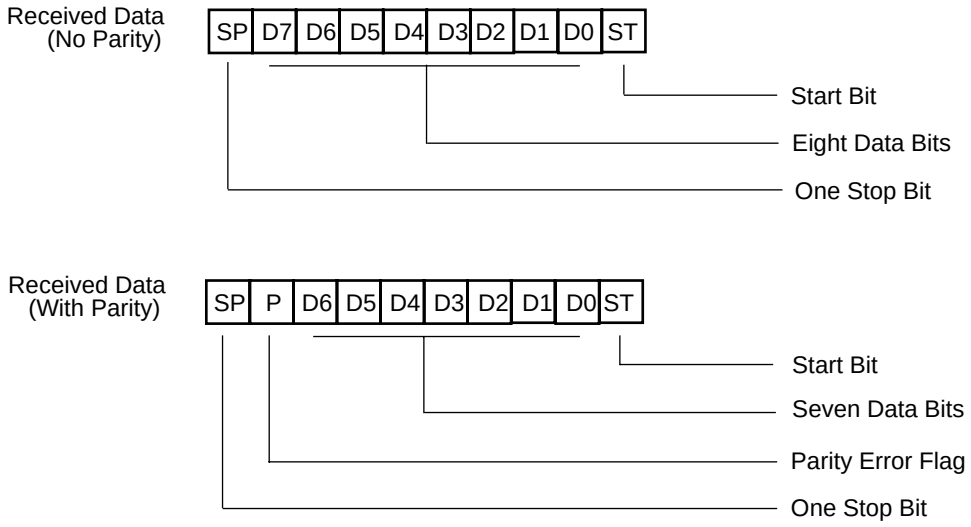


Figure 9-7. Receiver Data Formats

The Z8 hardware supports odd parity only, that is enabled by setting the Port 3 Mode Register bit 7 to 1 (Figure 9-8). If even parity is required, the Parity Mode should be disabled (P3M bit 7 set to 0), and software must calculate the received data's parity.

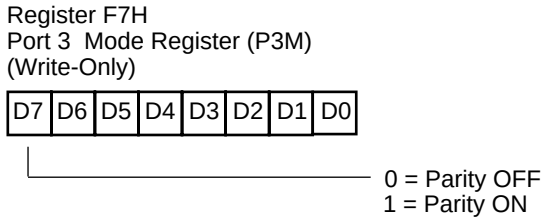


Figure 9-8. Port 3 Mode Register (P3M) Parity

9.4 TRANSMITTER OPERATION

The transmitter consists of a transmitter buffer (SIO Register [F0H]), a parity generator, and associated control logic. The transmitter block diagram is shown as part of Figure 9-1.

After a hardware reset or after a character has been transmitted, the transmitter is forced to a marking state (output always High) until a character is loaded into the transmitter buffer, SIO Register (F0H). The transmitter is loaded by specifying the SIO Register as the destination register of any instruction.

T0's output drives a divide-by-16 counter that in turn generates a shift clock every 16 counts. This counter is reset when the transmitter buffer is written by an instruction. This reset synchronizes the shift clock to the software. The transmitter then outputs one bit per shift clock, through Port 3 bit 7, until a start bit, the character written to the buffer, and two stop bits have been transmitted. After the second stop bit has been transmitted, the output is again forced to a marking state. Interrupt request IRQ4 is generated at this time to notify the processor that the transmitter is ready to accept another character.

9.4.1 Overwrites

The user is not protected from overwriting the transmitter, so it is up to the software to respond to IRQ4 appropriately. If polling is used, the IRQ4 bit in the Interrupt Request Register must be reset.

9.4.2 Parity

The data format supported by the transmitter has a start bit, eight data bits, and at least two stop bits. If parity is on, bit 7 of the data transmitted will be replaced by an odd parity bit. Figure 9-9 shows the transmitter data formats.

Parity is enabled by setting Port 3 Mode Register bit 7 to 1. If even parity is required, the parity mode should be disabled (P3M bit 7 reset to 0), and software must modify the data to include even parity.

Since the transmitter can be overwritten, the user is able to generate a break signal. This is done by writing null characters to the transmitter buffer (SIO Register [F0H]) at a rate that does not allow the stop bits to be output. Each time the SIO Register is loaded, the divide-by-16 counter is resynchronized and a new start bit is output followed by data.

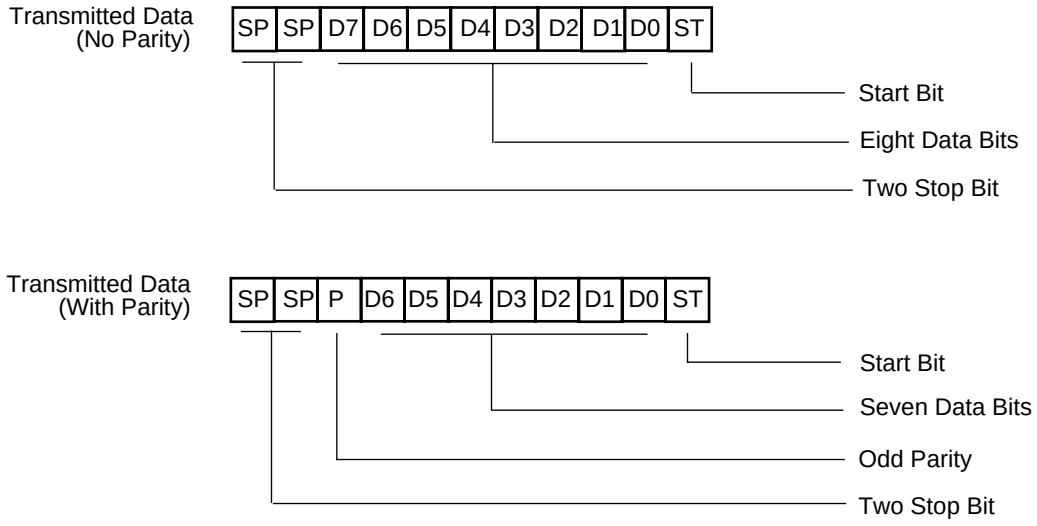


Figure 9-9. Transmitter Data Formats

9.5 UART RESET CONDITIONS

After a hardware reset, the SIO Register contents are undefined, and Serial Mode and parity are disabled. Figures 9-10 and 9-11 show the binary reset values of the SIO Register and its associated mode register P3M.

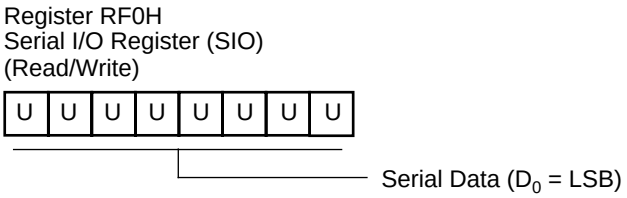


Figure 9-10. SIO Register Reset

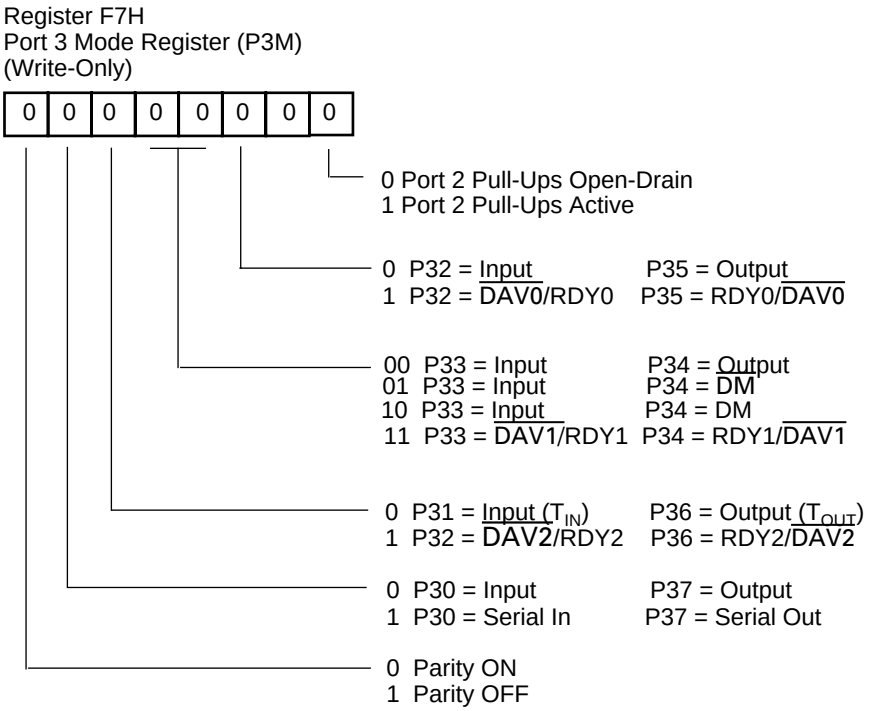


Figure 9-11. P3M Register Reset

9.6 SERIAL PERIPHERAL INTERFACE (SPI)

Select Z8 microcontrollers incorporate a serial peripheral interface (SPI) for communication with other microcontrollers and peripherals. The SPI includes features such as Stop-Mode Recovery, Master/Slave selection, and Compare mode. Table 9-3 contains the pin configuration for the SPI feature when it is enabled. The SPI consists of four registers: SPI Control Register (SCON), SPI Compare Register (SCOMP), SPI Receive/Buffer Register (RxBUF), and SPI Shift Register. SCON is located in bank (C) of the Expanded Register File at address 02.

Table 9-3. SPI Pin Configuration

Name	Function	Pin Location
DI	Data-In	P20
DO	Data-Out	P27
SS	Slave Select	P35
SK	SPI Clock	P34

The SPI Control Register (SCON) (Figure 9-12), is a read/write register that controls Master/Slave selection, interrupts, clock source and phase selection, and error flag. Bit 0 enables/disables the SPI with the default being SPI disabled. A 1 in this location will enable the SPI, and a 0 will disable the SPI. Bits 1 and 2 of the SCON register in Master Mode select the clock rate. The user may choose whether internal clock is divide-by-2, 4, 8, or 16. In Slave Mode, Bit 1 of this register flags the user if an overrun of the RxBUF Register has occurred. The RxCharOverrun flag is only reset by writing a 0 to this bit. In slave mode, bit 2 of the Control Register disables the data-out I/O function. If a 1 is written to this bit, the data-out pin is released to its original port configuration. If a 0 is written to this bit, the SPI shifts out one bit for each bit received. Bit 3 of the SCON Register enables the compare feature of the SPI, with the default being disabled. When the compare feature is enabled, a comparison of the value in the SCOMP Register is made with the value in the RxBUF Register. Bit 4 signals that a receive character is available in the RxBUF Register.

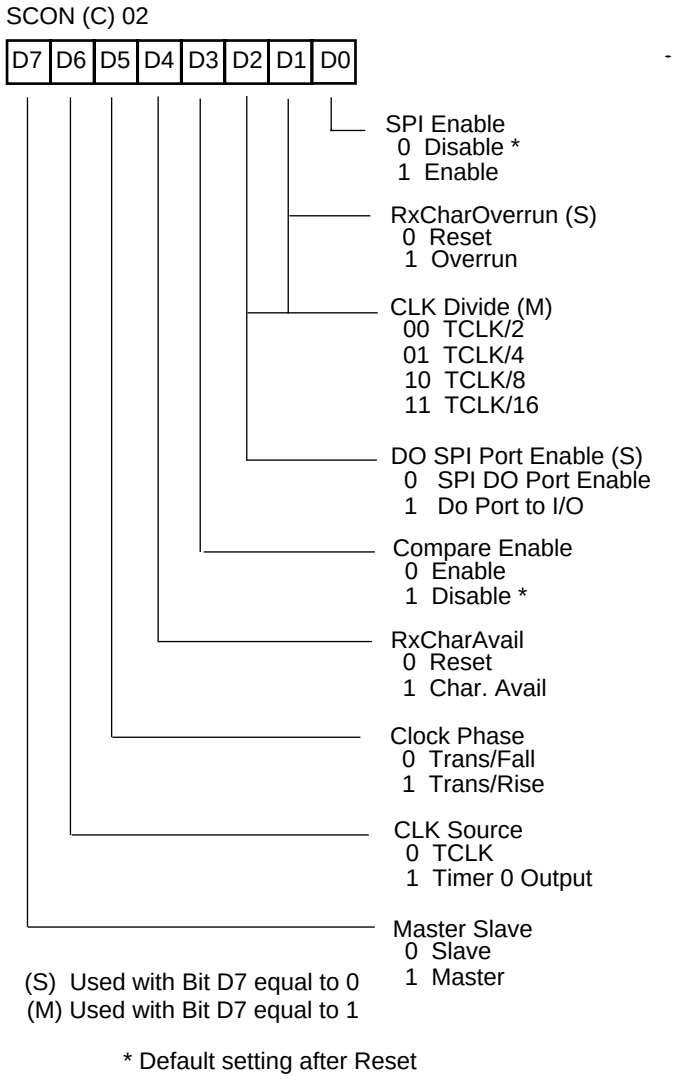


Figure 9-12. SPI Control Register (SCON)

If the associated IRQ3 is enabled, an interrupt is generated. Bit 5 controls the clock phase of the SPI. A 1 in bit 5 allows for receiving data on the clock's falling edge and transmitting data on the clock's rising edge. A 0 allows receiving data on the clock's rising edge and transmitting on the clock's falling edge. The SPI clock source is defined in bit 6. A 1 uses Timer0 output for the SPI clock, and a 0 uses TCLK for clocking the SPI. Finally, bit 7 determines whether the SPI is used as a Master or a Slave. A 1 puts the SPI into Master mode and a 0 puts the SPI into Slave mode.

9.7 SPI OPERATION

The SPI is used in one of two modes: either as system slave, or as system master. Several of the possible system configurations are shown in Figure 9-13. In the slave mode, data transfer starts when the slave select (SS) pin goes active. Data is transferred into the slave's SPI Shift Register through the DI pin, which has the same address as the RxBUF Register. After a byte of data has been received by the SPI Shift Register, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. The next byte of data will be received at this time. The RxBUF Register must be cleared, or a Receive Character Overrun (RxCharOverrun) flag will be set in the SCON Register, and the data in the RxBUF Register will be overwritten. When the communication between the master and slave is complete, the SS goes inactive.When the SPI is activated as a slave, it operates in all system modes: STOP, HALT, and RUN.

Unless disconnected, for every bit that is transferred into the slave through the DI pin, a bit is transferred out through the D0 pin on the opposite clock edge. During slave operation, the SPI clock pin (SK) is an input. In master mode, the CPU must first activate a SS through one of its I/O ports. Next, data is transferred through the master's D0 pin one bit per master clock cycle. Loading data into the shift register initiates the transfer. In master mode, the master's clock will drive the slave's clock. At the conclusion of a transfer, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. Before data is transferred via the D0 pin, the SPI Enable bit in the SCON Register must be enabled.

9.8 SPI COMPARE

When the SPI Compare Enable bit, D3 of the SCON Register is set to 1, the SPI Compare feature is enabled. The compare feature is only valid for slave mode. A compare transaction begins when the (SS) line goes active. Data is received as if it were a normal transaction, but there is no data transmitted to avoid bus contention with other slave devices. When the compare byte is received, IRQ3 is not generated. Instead, the data is compared with the contents of the SCOMP Register. If the data does not match, DO remains inactive and the slave ignores all data until the (SS) signal is reset. If the data received matches the data in the SCOMP register, then a SMR signal is generated. DO is activated if it is not tri-stated by D2 in the SCON Register, and data is received the same as any other SPI slave transaction.

Slaves' not comparing remain in their current mode, whereas slaves' comparing wake from a STOP mode by means of an SMR

9.9 SPI CLOCK

The SPI clock maybe driven by three sources: Timer0, a division of the internal system clock, or the external master when in slave mode. Bit D6 of the SCON Register controls what source drives the SPI clock. A 0 in bit D6 of the SCON Register determines the division of the internal system clock if this is used as the SPI clock source. Divide by 2, 4, 8, or 16 is chosen as the scaler.

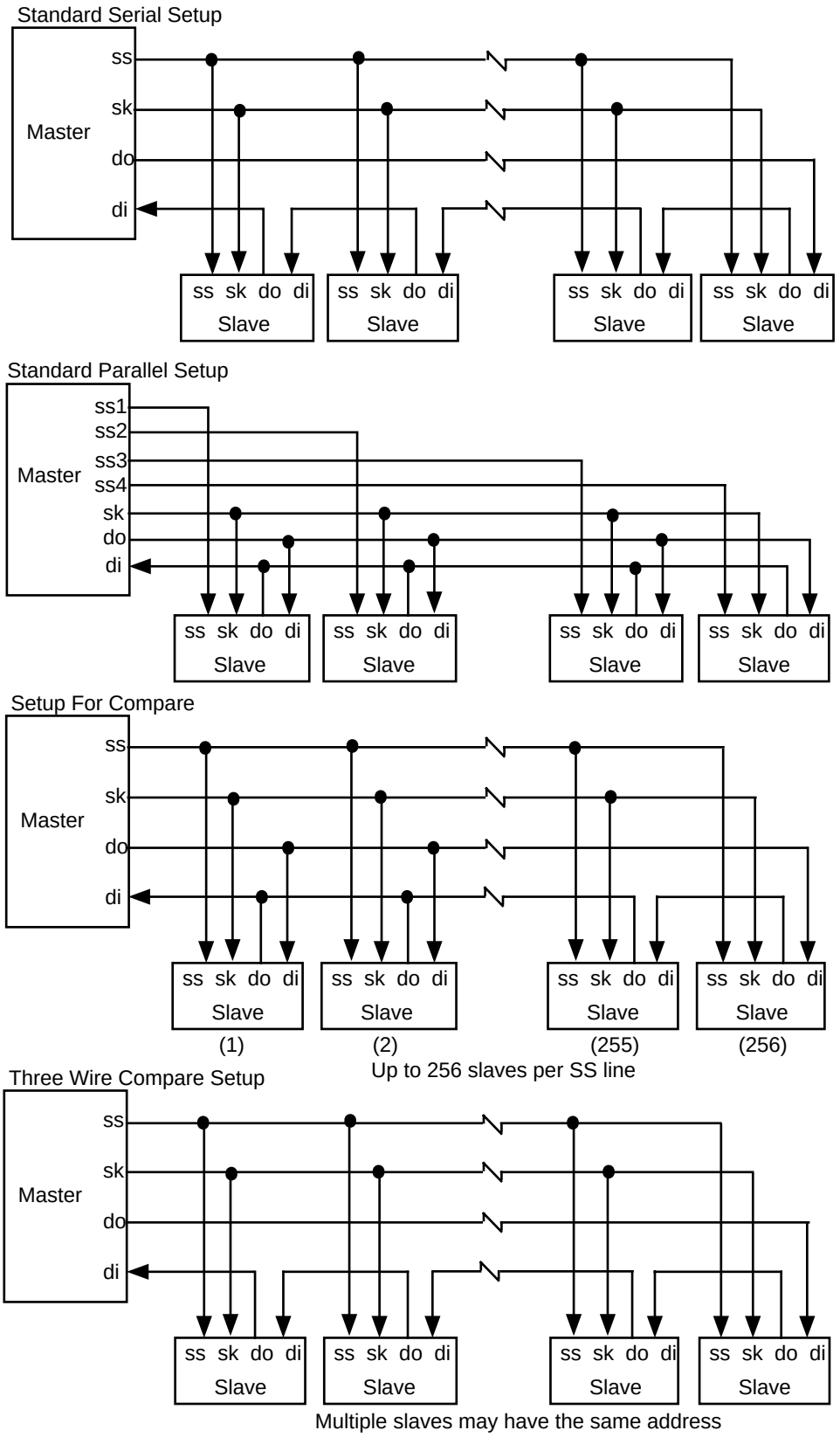


Figure 9-13. SPI System Configuration

9.10 RECEIVE CHARACTER AVAILABLE AND OVERRUN

When a complete data stream is received, an interrupt is generated and the RxCharAvail bit in the SCON Register is set. Bit 4 in the SCON Register is for enabling or disabling the RxCharAvail interrupt. The RxCharAvail bit is available for interrupt polling purposes and is reset when the RxBUF Register is read. RxCharAvail is generated in both master and slave modes. While in slave mode, if the RxBUF is not read before the next data stream is received and loaded into the RxBUF Register, Receive Character Overrun (RxCharOverrun) occurs. Since there is no need for clock control in slave mode, bit D1 in the SPI Control Register is used to log any RxCharOverrun (Figure 9-14 and Figure 9-15).

No	Parameter	Min	Units
1	DI to SK Setup	10	ns
2	SK to D0 Valid	15	ns
3	SS to SK Setup	.5 Tsk	ns
4	SS to D0 Valid	15	ns
5	SK to DI Hold Time	10	ns

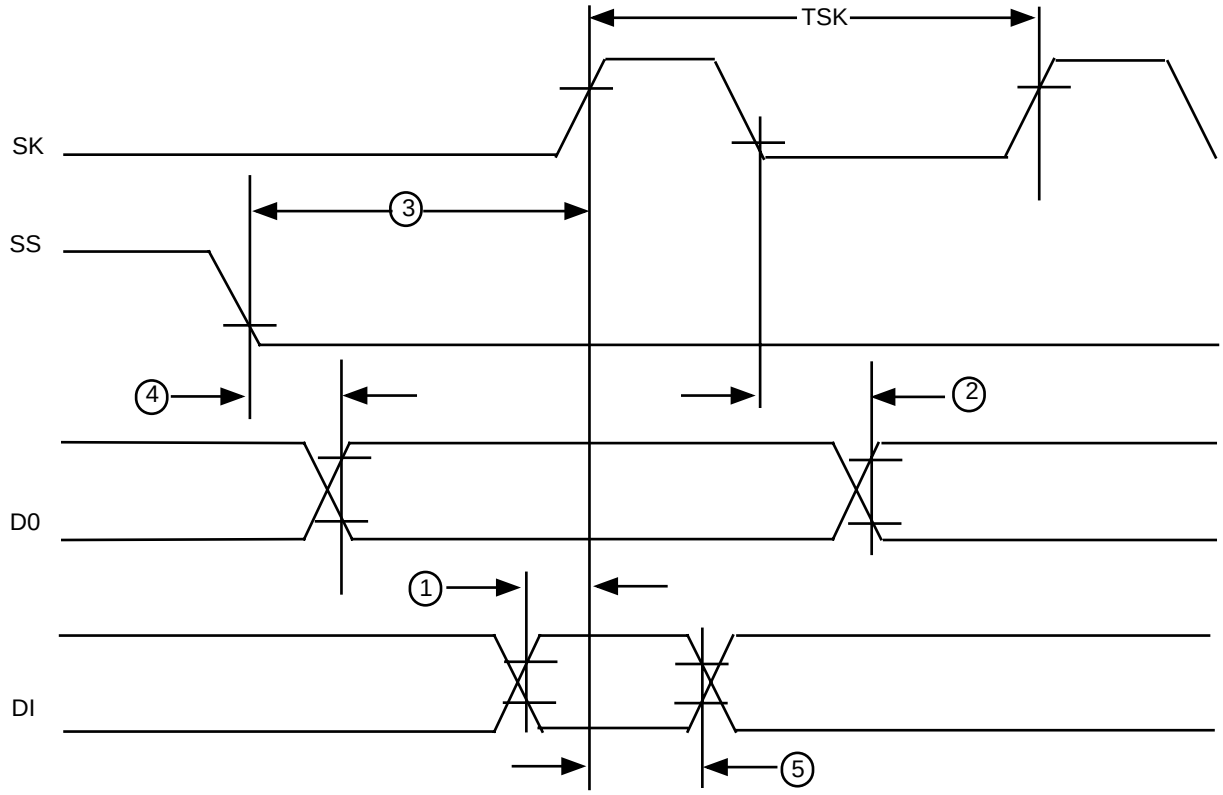


Figure 9-14. SPI Timing

9.10 RECEIVE CHARACTER AVAILABLE AND OVERRUN (Continued)

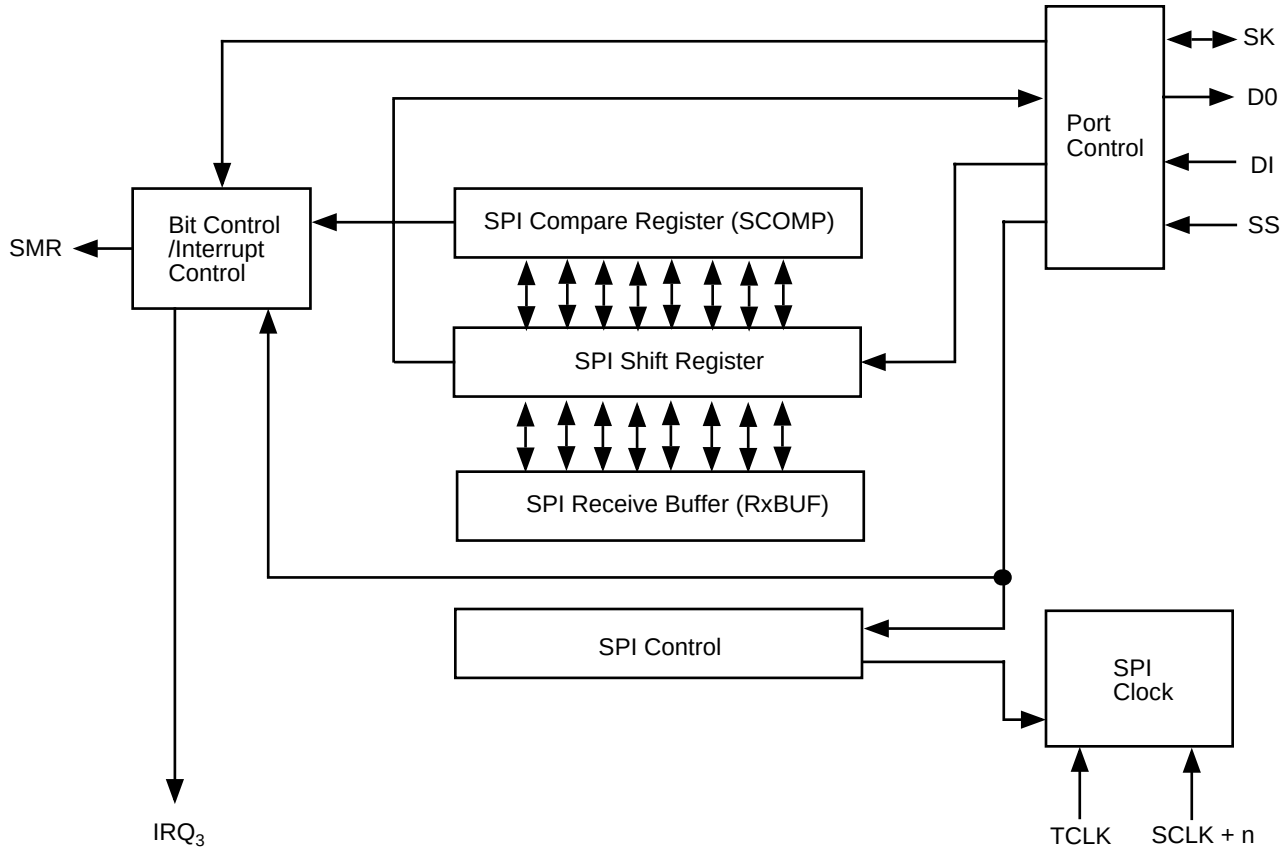


Figure 9-15. SPI Logic

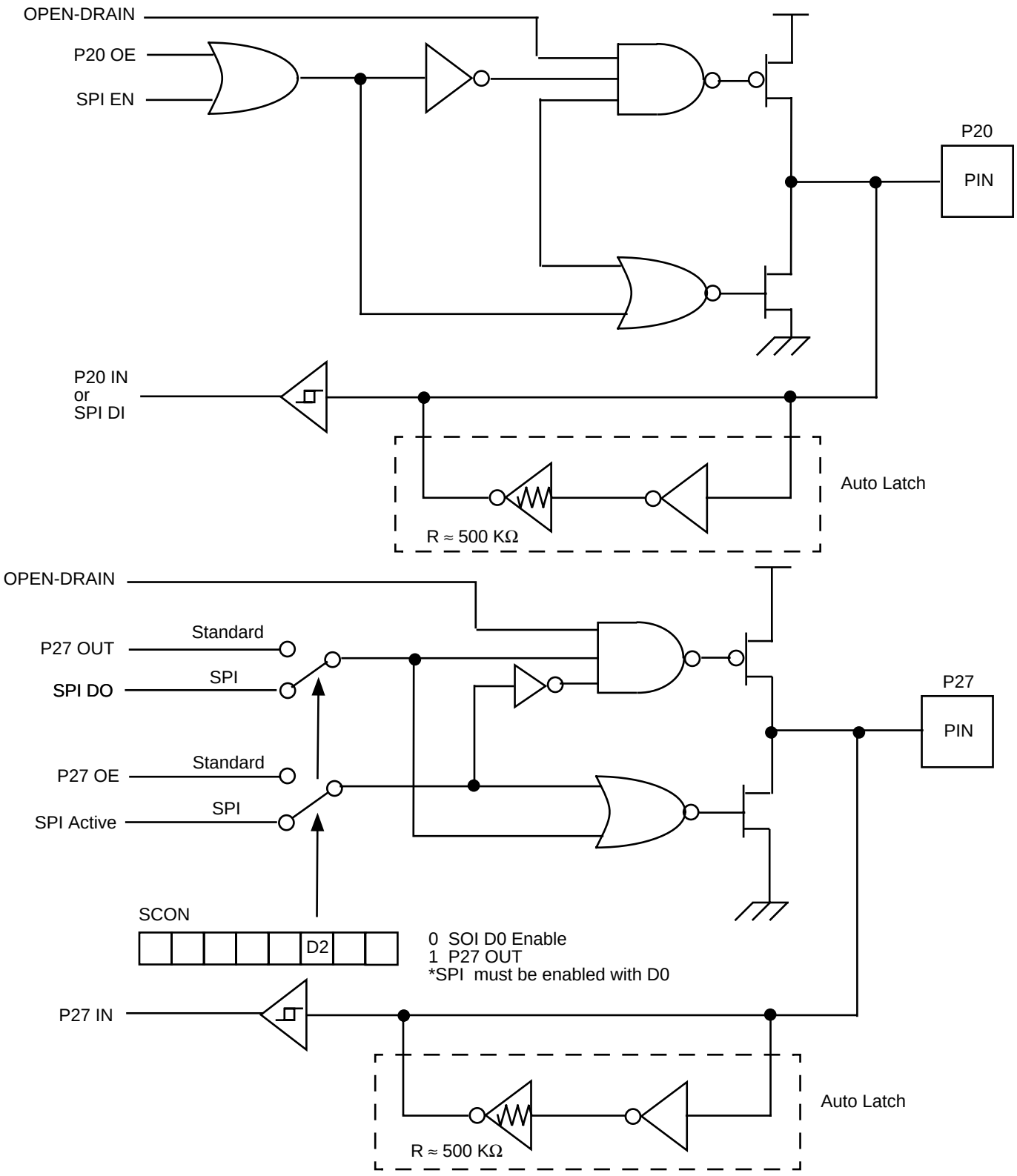


Figure 9-16. SPI Data In/Out Configuration

9.10 RECEIVE CHARACTER AVAILABLE AND OVERRUN (Continued)

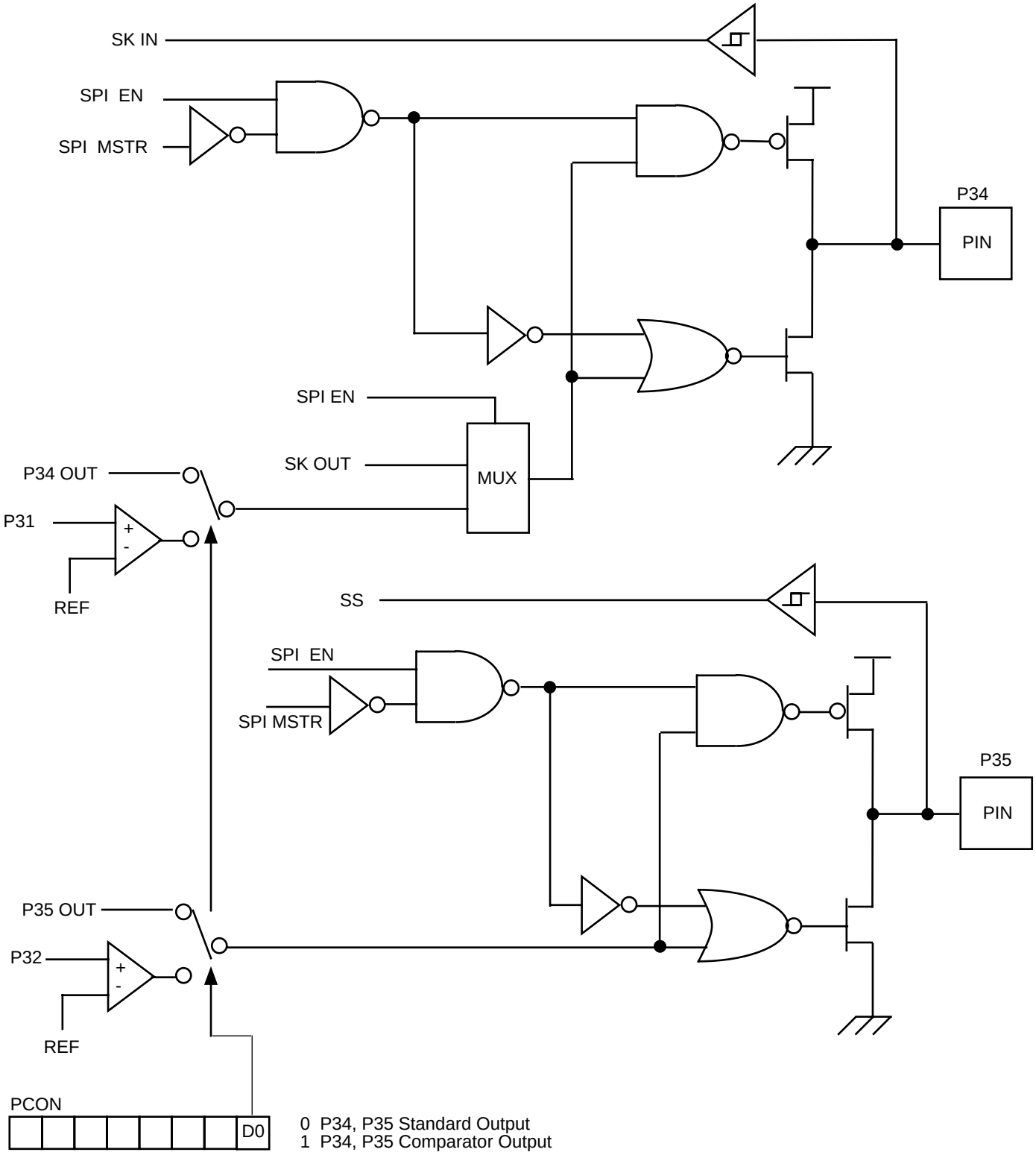


Figure 9-17. SPI Clock / SPI Slave Select Output Configuration

Totally Logical

CHAPTER 10
EXTERNAL INTERFACE

10.1 INTRODUCTION

The Z8 can be a microcontroller with 20 pins for external memory interfacing. The external memory interface on the Z8 is generally for either RAM or ROM. This is only available for devices featuring Port 0, Port 1, R/$\overline{W}$, $\overline{DM}$, $\overline{AS}$, and $\overline{DS}$. Please refer to specific product specifications for availability of these features.

The Z8 has a multiplexed external memory interface. In the multiplexed mode, eight pins from Port 1 form an Address/Data Bus (AD7-AD0), eight pins from Port 0 form a High Address Bus (A15-A8). Three additional pins provide the Address Strobe, Data Strobe, and the Read/Write Signal. Figure 10-1 shows the Z8 external interface pins.

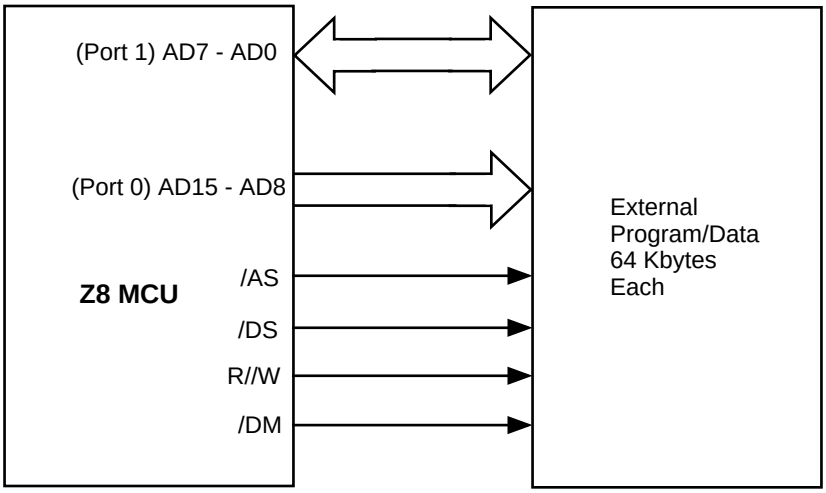


Figure 10-1. Z8 External Interface Pins

10.2 PIN DESCRIPTIONS

The following sections briefly describe the pins associated with the Z8 MCU® external memory interface.

10.2.1 /AS

Address Strobe (output, active Low). Address Strobe is pulsed Low once at the beginning of each machine cycle. The rising edge of $\overline{AS}$ indicates the address, Read/Write (R/$\overline{W}$), and Data Memory ($\overline{DM}$) signals are valid for program or data memory transfers. In some cases, the Z8 address strobe is pulsed low regardless of accessing external or internal memory. Please refer to specific product specifications for $\overline{AS}$ operation.

10.2.2 $\overline{DS}$

Data Strobe (Output, Active Low). Data Strobe provides the timing for data movement to or from the Address/Data bus for each external memory transfer. During a Write Cycle, data out is valid at the leading edge of the $\overline{DS}$. During a Read Cycle, data in must be valid prior to the trailing edge of the $\overline{DS}$.

10.2.3 R/$\overline{W}$

Read/Write (Output). Read/Write determines the direction of data transfer for memory transactions. R/$\overline{W}$ is Low when writing to program or data memory, and High for all other transactions.

10.2.4 DM

Data Memory (Output). Data Memory provides a signal to separate External Program Memory from External Data Memory. It is a programmable function on pin P34. Data memory is active low for External Data Memory accesses and high for External Program Memory accesses.

10.2.5 P07 - P00

High Address Lines A15 -A8 (Outputs can be CMOS- or TTL- compatible. Please refer to product specifications for actual type). A15-A8 provide the High Address lines for the memory interface. Port 0 - 1 mode register must have bits D7 = 1 and D1 = 1 to configure Port 0 as A15 - A8 (Figure 10-2).

10.2.6 P17 - P10

Address/Data Lines AD7 - AD0 (inputs/outputs, TTL-compatible). AD7-AD0 is a multiplexed Address/Data memory interface. The lower eight Address lines (A7-A0) are multiplexed with Data lines (D7-D0). Port 0 - 1 mode register must have bits D4 = 1 and D3 = 0 to configure Port 1 as AD7 - AD0 (Figure 10-2).

10.2.7 /RESET

Reset (input, active Low). $\overline{RESET}$ initializes the Z8. When $\overline{RESET}$ is deactivated, program execution begins from program location 000CH. If held Low, $\overline{RESET}$ acts as a register file protect during power-down and power-up sequences. To avoid asynchronous and noisy reset problems, the Z8 is equipped with a reset filter of four external clocks ($4T_PC$). If the external $\overline{RESET}$ signal is less than $4T_PC$ in duration, no reset will occur. On the fifth clock after the $\overline{RESET}$ is detected, an internal reset signal is latched and held for an internal register count of 18 or more external clocks, or for the duration of the external $\overline{RESET}$, whichever is longer. Please refer to specific product specifications for length of reset delay time.

10.2.8 XTAL1, XTAL2.

Crystal1, Crystal2 (Oscillator input and output). These pins connect a parallel-resonant crystal, ceramic resonator, LC, RC network, or external single-phase clock to the on-chip oscillator input. Please refer to the device product specifications for information on availability of RC oscillator features.

10.3 EXTERNAL ADDRESSING CONFIGURATION

The minimum bus configuration uses Port 1 as a multiplexed address/data port (AD7 - AD0), allowing access to 256 bytes of external memory. In this configuration, the eight low-order bits (A0 - A7) are multiplexed with the data (D7 - D0).

Port 0 can be programmed to provide either four additional address lines (A11- A8), which increases the addressable memory to 4K bytes, or eight additional address lines (A15 - A8), which increases the addressable external memory up to 64K bytes. It is required to add a NOP after configuring Port 0 / Port 1 for external addressing before jumping to external memory execution.

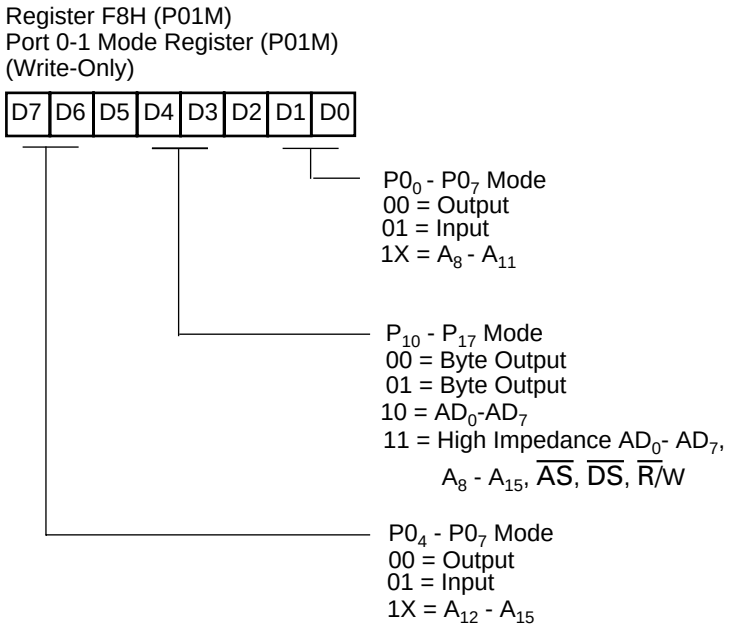


Figure 10-2. External Address Configuration

10.4 EXTERNAL STACKS

The Z8 architecture supports stack operations in either the Z8 Standard Register File or External Data Memory. A stack's location is determined by bit 2 in the Port 0-1 Mode Register (F8H). If bit 2 is set to 0, the stack is in External Data Memory. (Figure 10-3).

The instruction used to change the stack selection bit should not be immediately followed by the instructions RET or IRET, because this will cause indeterminate program flow. After a $\overline{\text{RESET}}$, the internal stack is selected.

Please note that if Port 0 is configured as A15 - A8 and the stack is selected as internal, any stack operation will cause the contents in register FEH to be displayed on Port 0.

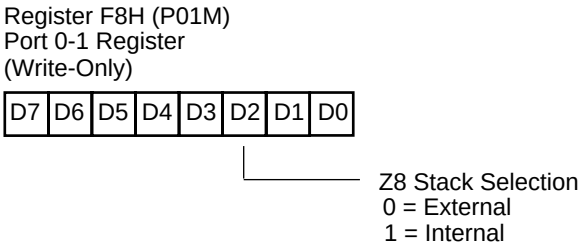


Figure 10-3. Z8 Stack Selection

10.5 DATA MEMORY

The two Z8 external memory spaces, data and program, are addressed as two separate spaces of up to 64 Kbytes each. External Program Memory and External Data Memory are logically selected by the Data Memory select output ($\overline{\text{DM}}$). $\overline{\text{DM}}$ is made available on Port 3, bit 4 (P34) by setting bit 4 and bit 3 in the Port 3 Mode Register (F7H) to 10 or 01 (Figure 10-4). $\overline{\text{DM}}$ is active Low during the execution of the LDE, LDEI instructions, and High for the execution of program instructions. $\overline{\text{DM}}$ is also active Low during the execution of CALL, POP, PUSH, RET and IRET instructions if the stack resides in External Data Memory. After a $\overline{\text{RESET}}$, $\overline{\text{DM}}$ is not selected.

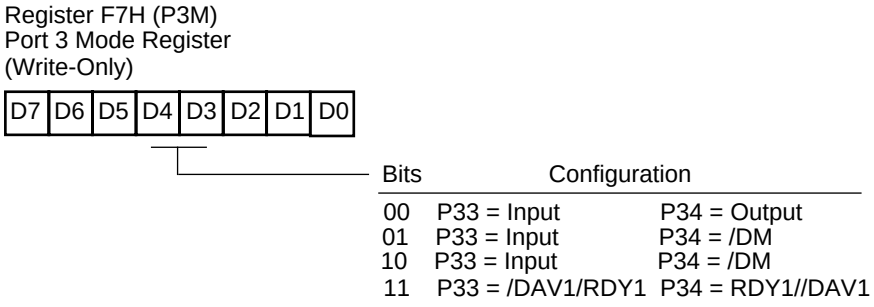
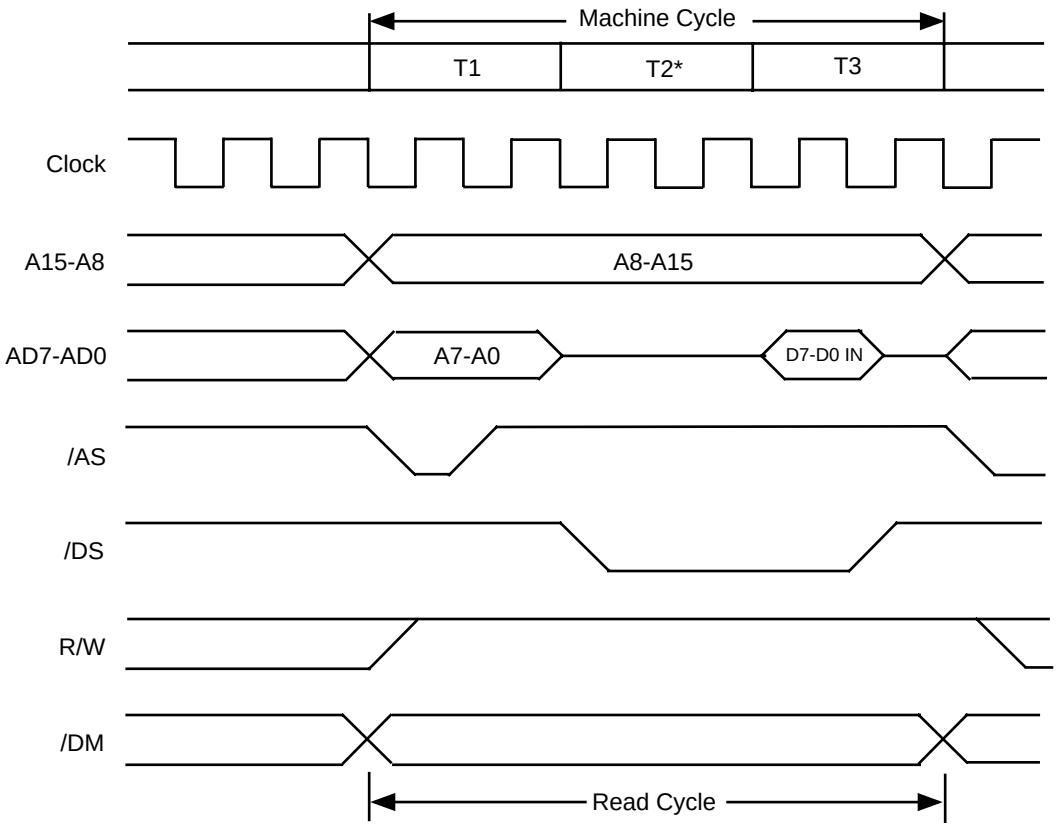


Figure 10-4. Port 3 Data Memory Operation

10.6 BUS OPERATION

Typical data transfers between the Z8 MCU and External Memory are illustrated in Figures 10-5 and 10-6. Machine cycles can vary from six to 12 clock periods depending on the operation being performed. The notations used to describe the basic timing periods of the Z8 are machine cycles (Mn), timing states (Tn), and clock periods. All timing references are made with respect to the output signals $\overline{AS}$ and $\overline{DS}$. The clock is shown for clarity only and does not have a specific timing relationship with other signals.



*Port inputs are strobed during T2, which is two internal systems clocks before the execution cycle of the current instruction.

Figure 10-5. External Instruction Fetch or Memory Read Cycle

10.6 BUS OPERATION (Continued)

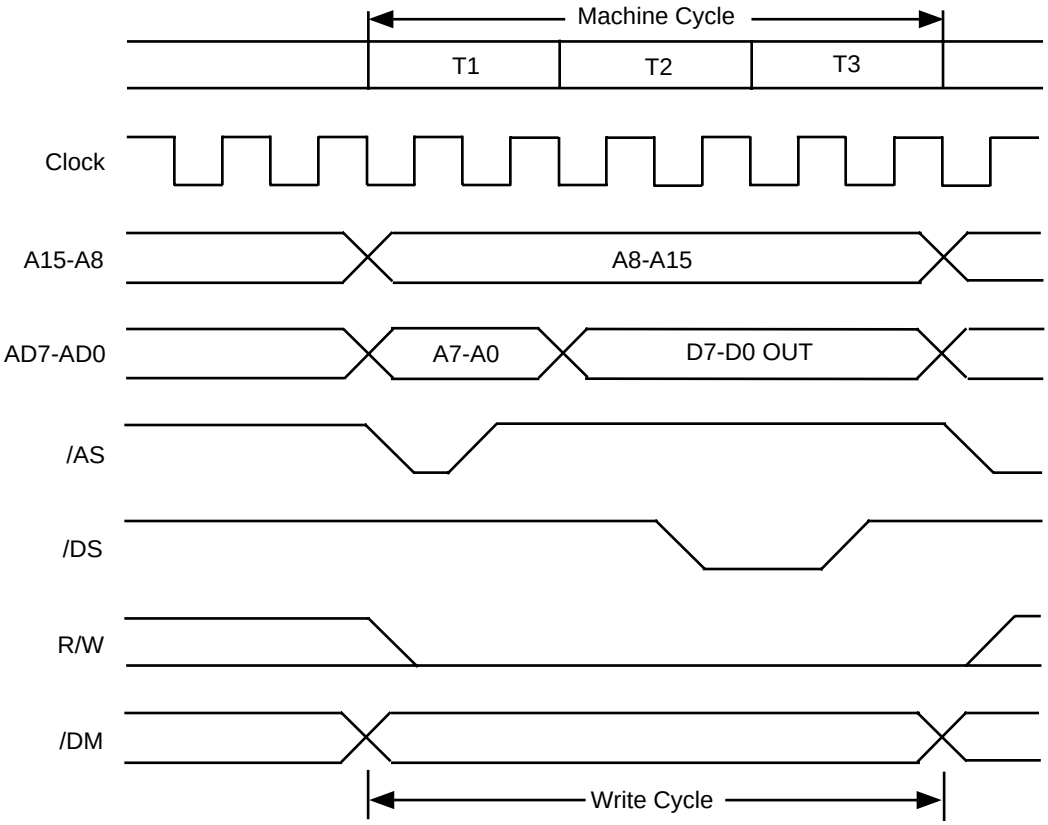


Figure 10-6. External Memory Write Cycle

10.6.1 Address Strobe ($\overline{AS}$)

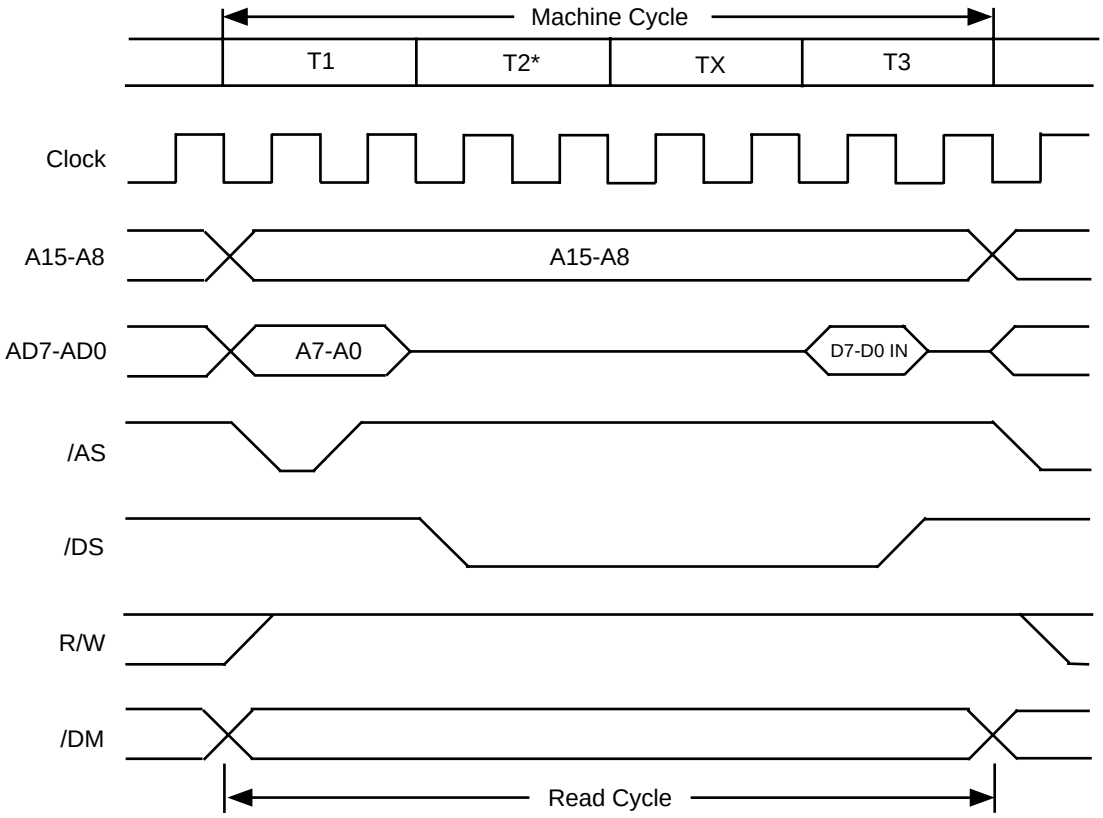
All transactions start with $\overline{AS}$ driven Low and then raised High by the Z8 MCU. The rising edge of $\overline{AS}$ indicates that R/$\overline{W}$, $\overline{DM}$ (if used), and the address outputs are valid. The address outputs (AD7-AD0), remain valid only during MnT1 and typically need to be latched using $\overline{AS}$. Address outputs (A15-A8) remain stable throughout the machine cycle, regardless of the addressing mode.

10.6.2 Data Strobe (/DS)

The Z8 uses $\overline{DS}$ to time the actual data transfer. For Write operations (R/$\overline{W}$ = Low), a Low on $\overline{DS}$ indicates that valid data is on the AD7-AD0 lines. For Read operations (R/W = High), the bus is placed in a high-impedance state before driving $\overline{DS}$ Low, so the addressed device can put its data on the bus. The Z8 samples this data prior to raising $\overline{DS}$ High.

10.7 EXTENDED BUS TIMING

Some products can accommodate slow memory access time by automatically inserting an additional software controlled state time (Tx). This stretches the $\overline{DS}$ timing by two clock periods. Figures 10-7 and 10-8 illustrate extended external memory Read and Write cycles.



*Port inputs are strobed during T2, which is two internal system clocks before the execution of the current instruction.

Figure 10-7. Extended External Instruction Fetch or Memory Read Cycle

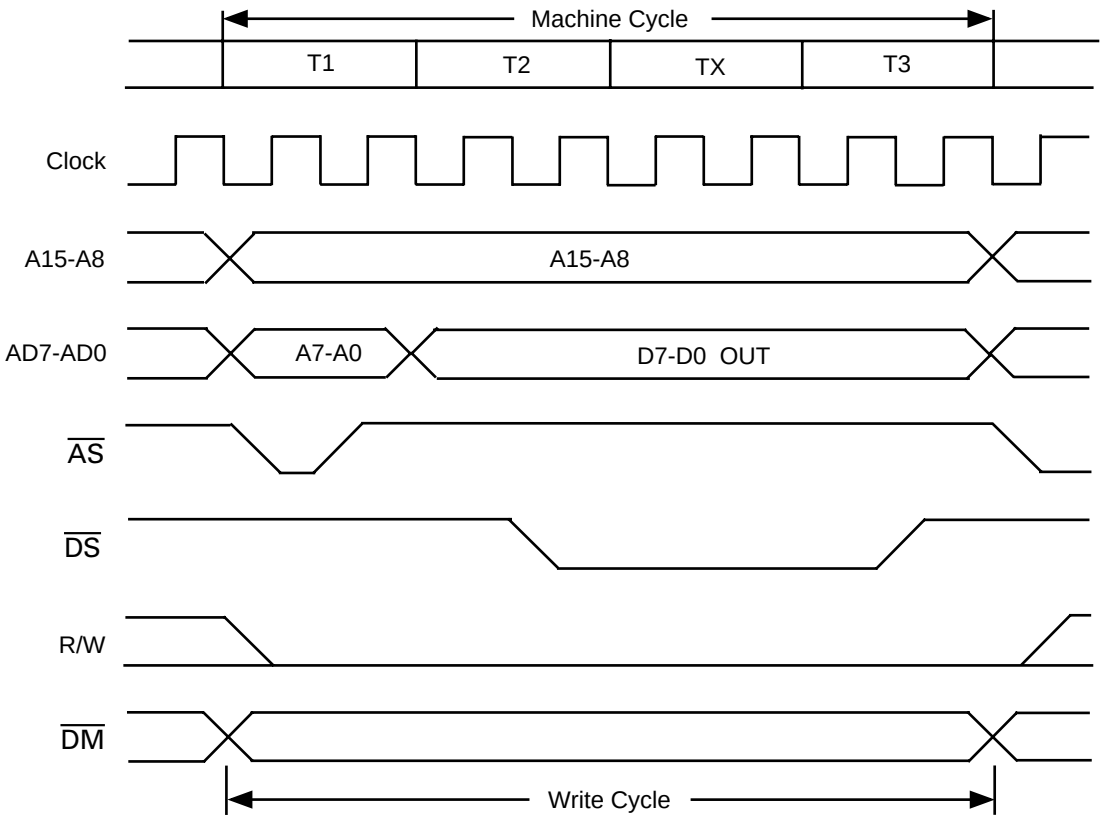


Figure 10-8. Extended External Memory Write Cycle

Timing is extended by setting bit D5 in the Port 0-1 Mode Register (F8H) to 1 (Figure 10-9). After a RESET, this bit is set to 0.

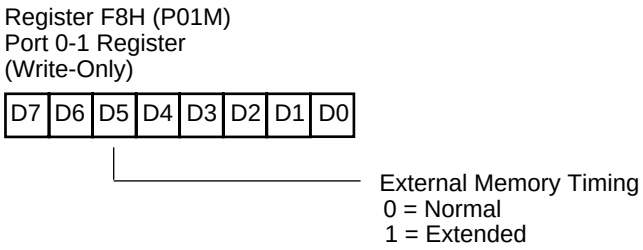


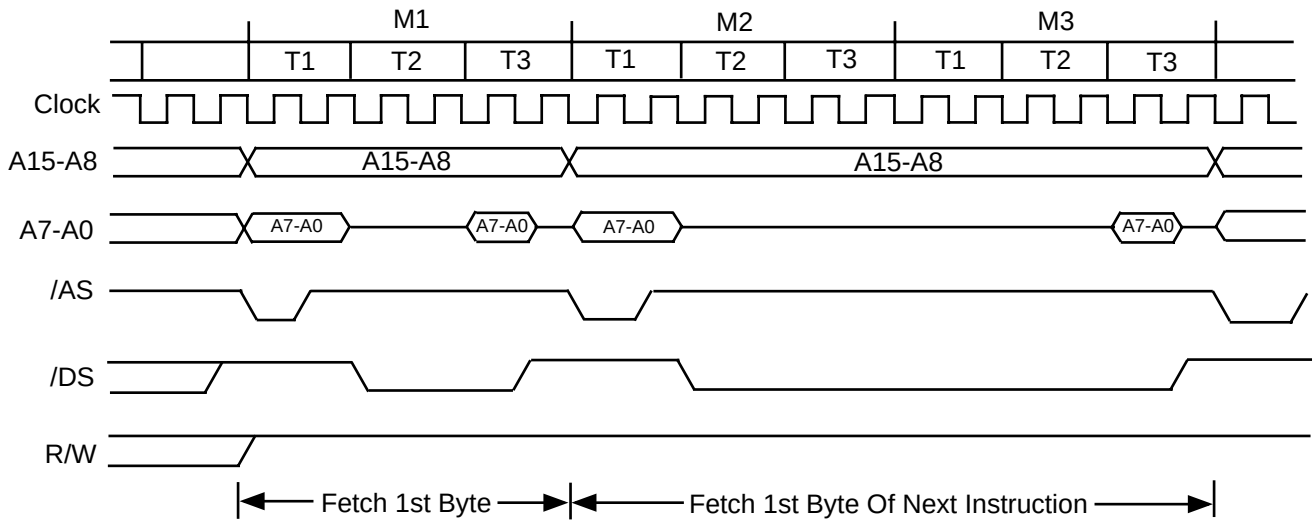
Figure 10-9. Extended Bus Timing

10.8 INSTRUCTION TIMING

The High throughput of the Z8 is due, in part, to the use of an instruction pipeline, in which the instruction fetch and execution cycles are overlapped. During the execution of the current instruction, the opcode of the next instruction is fetched. Instruction pipelining is illustrated in Figure 10-10.

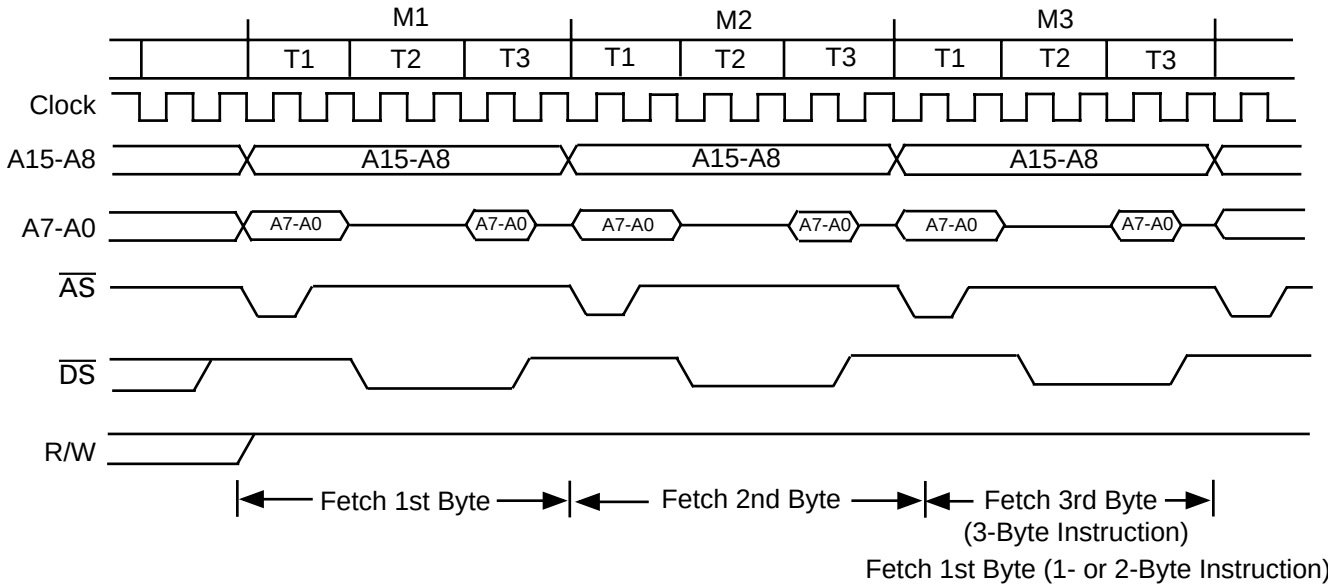
Figures 10-10 and 10-11 show typical instruction cycle timing for instructions fetched from memory. For those instructions that require execution time longer than that of the overlapped fetch, or reference program or data memory as part of their execution, the pipe must be flushed.

Figures 10-10 and 10-11 assume the XTAL/2 clock mode is selected.



* Port inputs are strobed during T2, which is two internal system clocks before the execution cycle of the current installation

Figure 10-10. Instruction Cycle Timing (One-Byte Instructions)



*Port inputs are strobed during T2, which is two internal system clocks before the execution cycle of the current instruction.

Figure 10-11. Instruction Cycle Timing (2- and 3-Byte Instructions)

10.9 Z8 RESET CONDITIONS

After a hardware reset, extended timing is set to accommodate slow memory access during the configuration routine, $\overline{DM}$ is inactive, the stack resides in the register file. Port 0, 1, and 2 are reset to input mode. Port 2 is set to Open-Drain Mode.

Totally Logical

CHAPTER 11
ADDRESSING MODES

11.1 INTRODUCTION

11.1.1 Z8 Addressing Modes

The Z8 microcontroller provides six addressing modes:

- Register (R)
- Indirect Register (IR)
- Indexed (X)
- Direct (D)
- Relative (RA)
- Immediate (IM)

With the exception of immediate data and condition codes, all operands are expressed as register file, Program Memory, or Data Memory addresses. Registers are accessed using 8-bit addresses in the range of 00H-FFH. The Program Memory or Data Memory is accessed using 16-bit addresses (register pairs) in the range of 0000H-FFFFH.

Working Registers are accessed using 4-bit addresses in the range of 0-15 (0H-FH). The address of the register being accessed is formed by the combination of the upper four bits in the Register Pointer (R253) and the 4-bit working register address supplied by the instruction.

Registers can be used in pairs to designate 16-bit values or memory addresses. A Register Pair must be specified as an even-numbered address in the range of 0, 2,...., 14 for Working Registers, or 4, 6,....238 for actual registers.

In the following definitions of Z8 Addressing Modes, the use of'register' can also imply register pair, working register, or working register pair, depending on the context.

Note: See the product data sheet for exact program, data, and register memory types and address ranges available.

11.2 Z8 REGISTER ADDRESSING (R)

In 8-bit Register Addressing mode, the operand value is equivalent to the contents of the specified register or register pair.

In the Register Addressing (Figure 11-1), the destination and/or source address specified corresponds to the actual register in the register file.

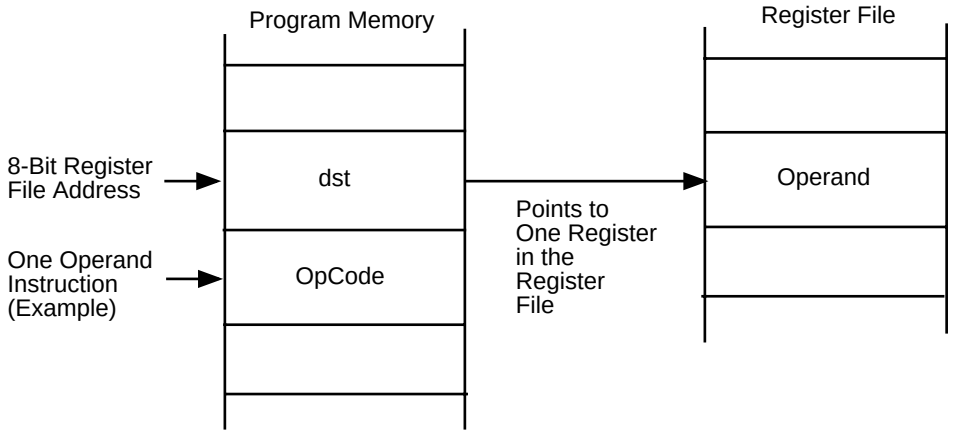


Figure 11-1. 8-Bit Register Addressing

In 4-bit Register Addressing (Figure 11-2), the destination and/or source addresses point to the Working Register within the current Working Register Group. This 4-bit address is combined with the upper four bits of the Register Pointer to form the actual 8-bit address of the affected register.

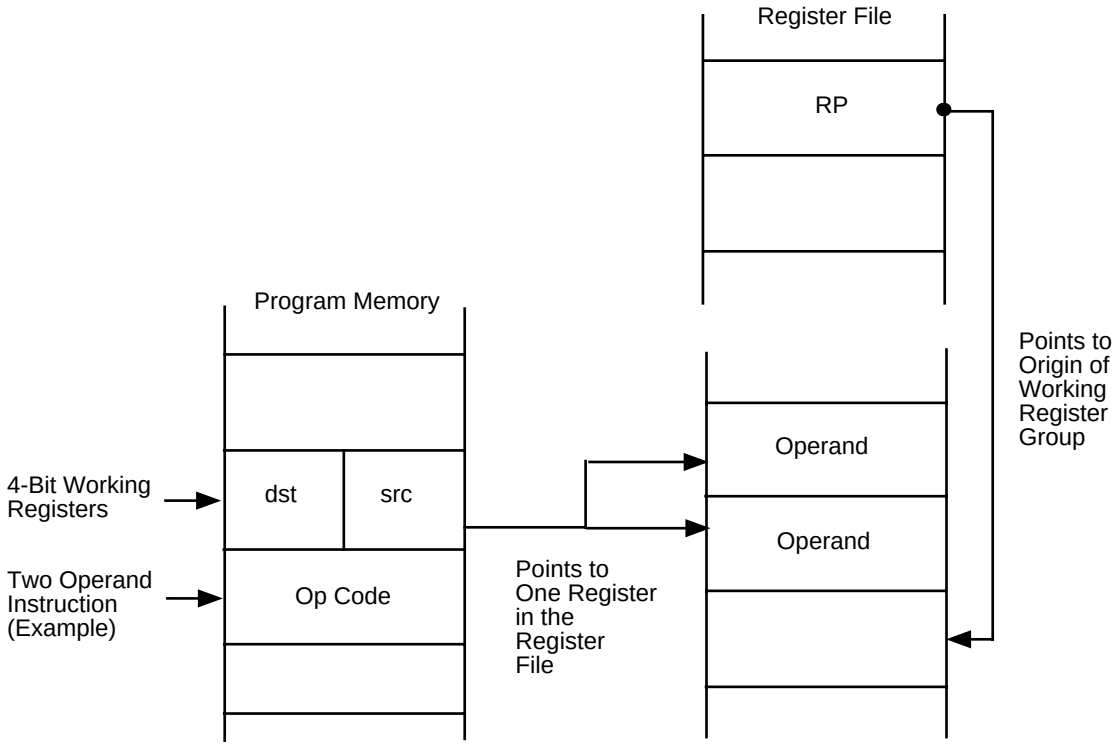


Figure 11-2. 4-Bit Register Addressing

11.3 Z8 INDIRECT REGISTER ADDRESSING (IR)

In the Indirect Register Addressing Mode, the contents of the specified register are equivalent to the address of the operand (Figures 11-3 and 11-4).

Depending upon the instruction selected, the specified register contents points to a Register, Program Memory, or an External Data Memory location.

When accessing program memory or External Data Memory, register pairs or Working Register pairs are used to hold the 16-bit addresses.

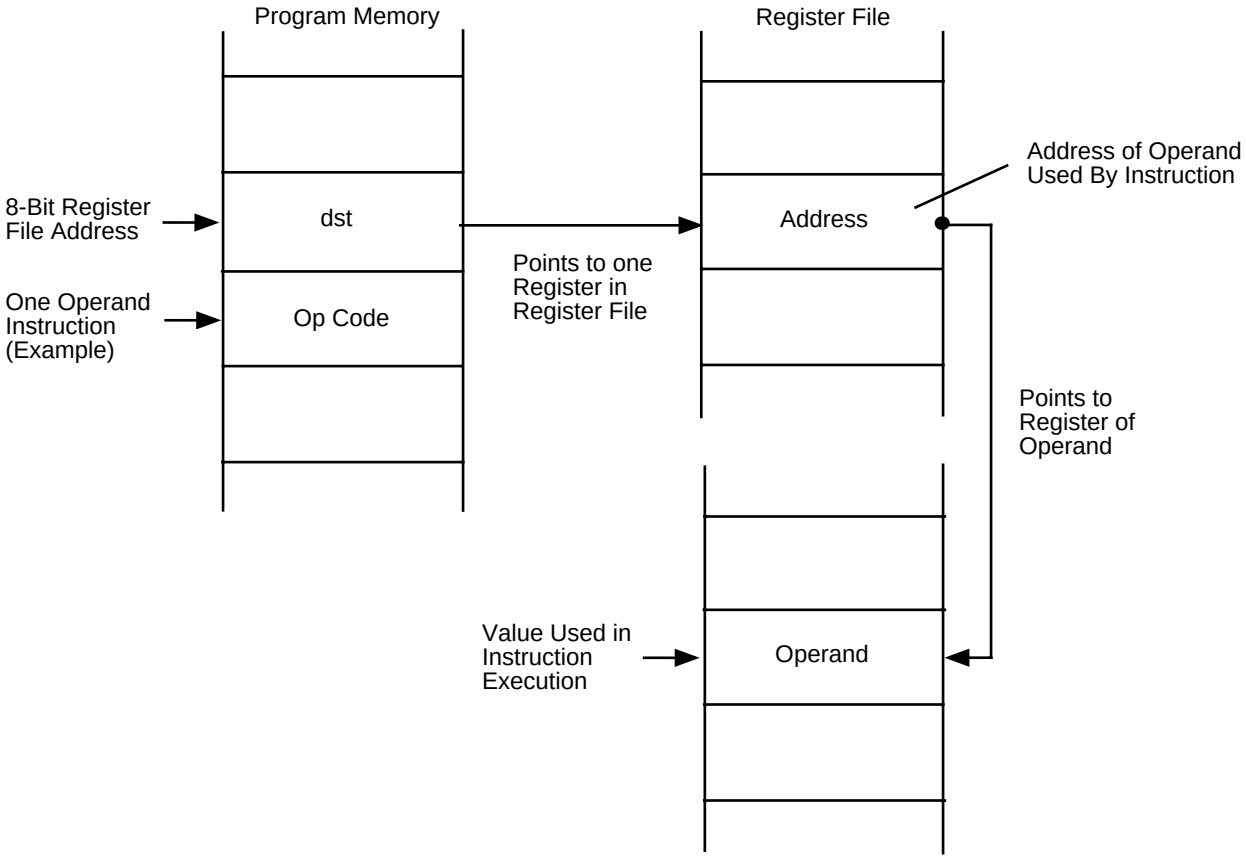


Figure 11-3. 4-Bit Register Addressing

11.3 Z8 INDIRECT REGISTER ADDRESSING (IR) (Continued)

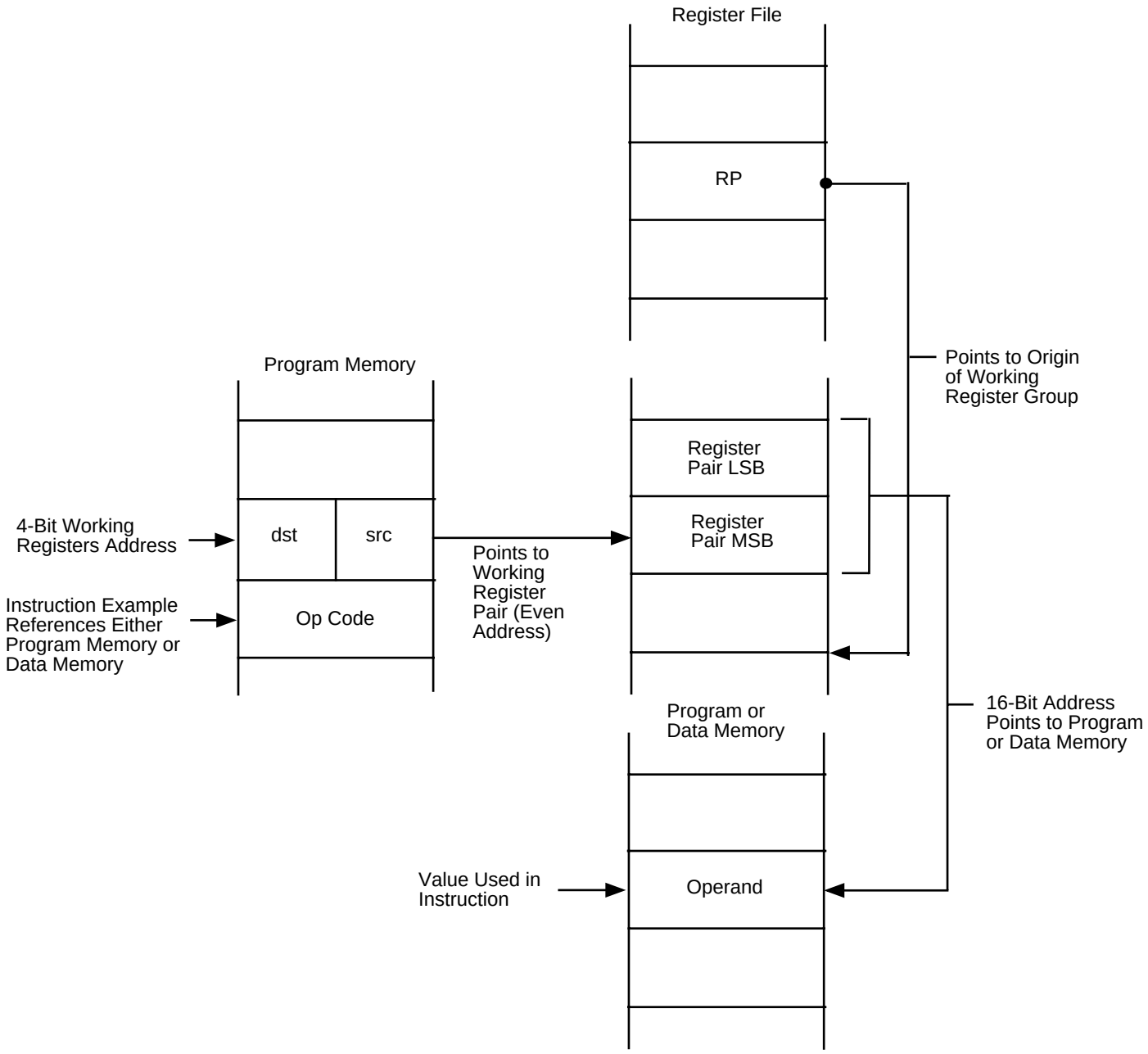


Figure 11-4. Indirect Register Addressing to Program or Data Memory

11.4 Z8 INDEXED ADDRESSING (X)

The Indexed Addressing Mode is used only by the Load (LD) instruction. 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 to obtain the address of the operand. Figure 11-5 illustrates this addressing convention.

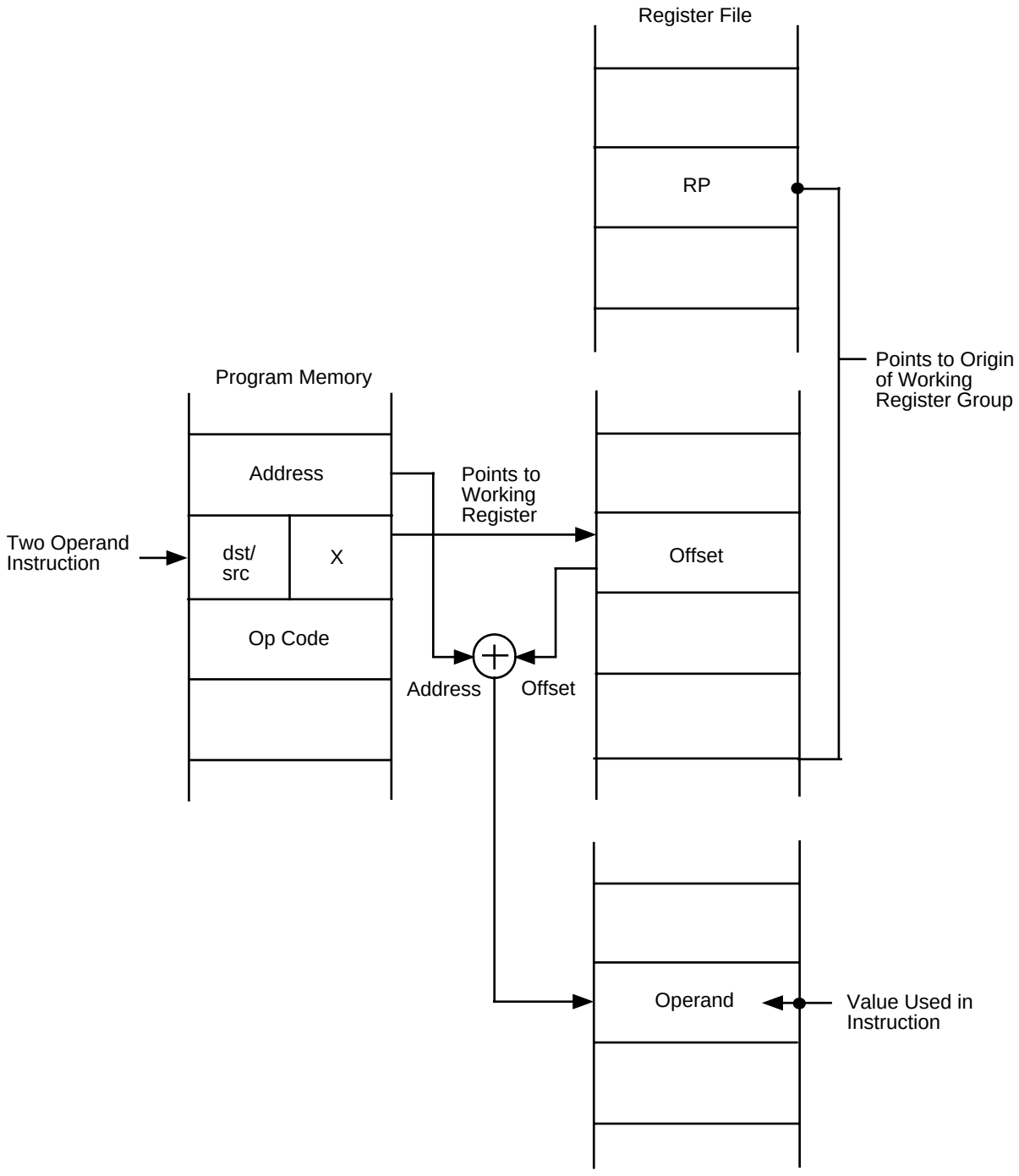


Figure 11-5. Indexed Register Addressing

11.5 Z8 DIRECT ADDRESSING (DA)

The Direct Addressing mode, as shown in Figure 11-6, specifies the address of the next instruction to be executed. Only the Conditional Jump (JP) and Call (CALL) instructions use this addressing mode.

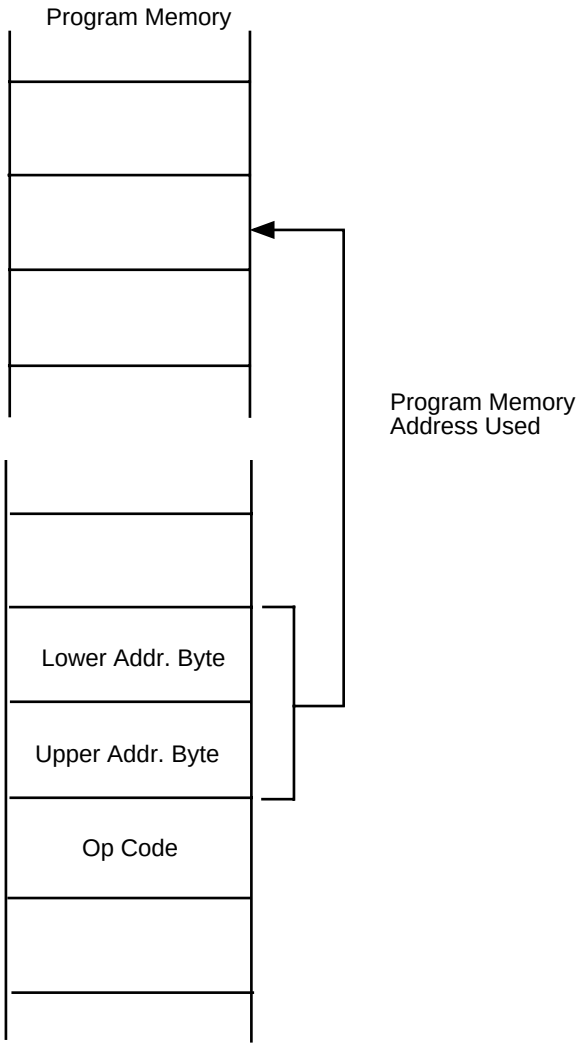


Figure 11-6. Direct Addressing

11.6 Z8 RELATIVE ADDRESSING (RA)

In the Relative Addressing mode, illustrated in Figure 11-7, the instruction specifies a two's-complement signed displacement in the range of –128 to +127. This is added to the contents of the PC to obtain the address of the next instruction to be executed. The PC (prior to the add) consists of the address of the instruction following the Jump Relative (JR) or Decrement and Jump if Non-Zero (DJNZ) instruction. JR and DJNZ are the only instructions which use this addressing mode.

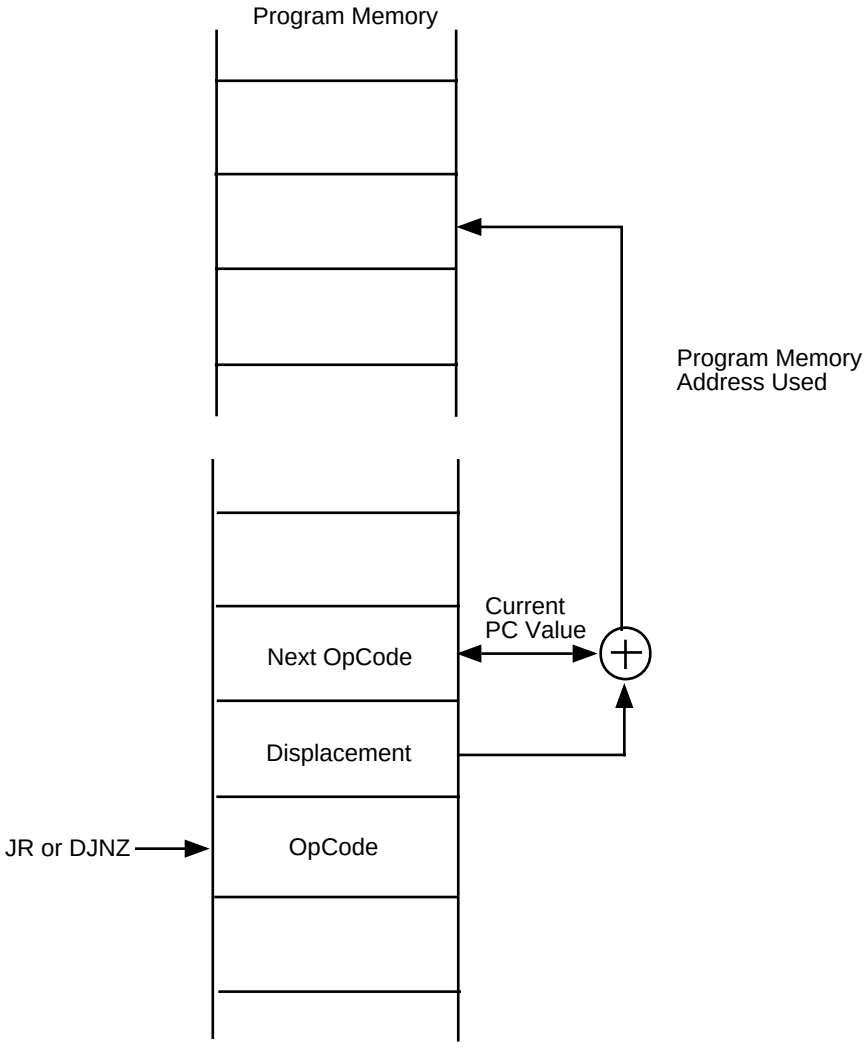


Figure 11-7. Relative Addressing

11.7 Z8 IMMEDIATE DATA ADDRESSING (IM)

Immediate data is considered an “addressing mode” for the purposes of this discussion. It is the only addressing mode that does not indicate a register or memory address as the source operand. The operand value used by the instruction is the value supplied in the operand field itself. Because an immediate operand is part of the instruction, it is always located in the Program Memory address space (Figure 11-8).

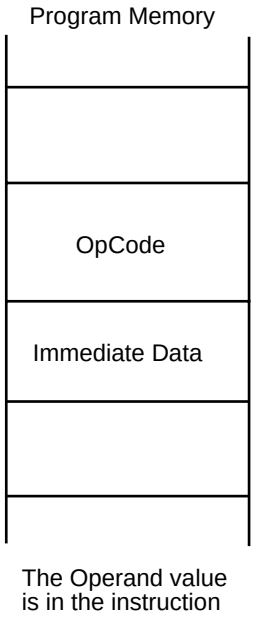


Figure 11-8. Immediate Data Addressing

Totally Logical

CHAPTER 12
INSTRUCTION SET

12.1 Z8 FUNCTIONAL SUMMARY

Z8 instructions can be divided functionally into the following eight groups:

- Load
- Bit Manipulation
- Arithmetic
- Block Transfer
- Logical
- Rotate and Shift
- Program Control
- CPU Control

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

Table 12-1. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant
LDE	dst, src	Load External
POP	dst	Pop
PUSH	src	Push

Table 12-2. Arithmetic Instructions

Mnemonic	Operands	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

Table 12-3. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical Exclusive OR

Table 12-4. Program Control Instructions

Mnemonic	Operands	Instruction
CALL	dst	Call Procedure
DJNZ	dst, src	Decrement and Jump Non-Zero
IRET		Interrupt Return
JP	cc, dst	Jump
JR	cc, dst	Jump Relative
RET		Return

Table 12-5. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
TCM	dst, src	Test Complement Under Mask
TM	dst, src	Test Under Mask
AND	dst, src	Bit Clear
OR	dst, src	Bit Set
XOR	dst, src	Bit Complement

Table 12-6. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant Auto Increment
LDEI	dst, src	Load External Auto Increment

Table 12-7. Rotate and Shift Instructions

Mnemonic	Operands	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 12-8. CPU Control Instructions

Mnemonic	Operands	Instruction
CCF		Complement Carry Flag
DI		Disable Interrupts
EI		Enable Interrupts
HALT		Halt
NOP		No Operation
RCF		Reset Carry Flag
SCF		Set Carry Flag
SRP	src	Set Register Pointer
STOP		Stop
WDH		WDT Enable During HALT
WDT		WDT Enable or Refresh

12.2 PROCESSOR FLAGS

The Flag Register (FCH) informs the user of the current status of the Z8. The flags and their bit positions in the Flag Register are shown in Figure 12-1.

The Z8 Flag Register contains six bits of status information which are set or cleared by CPU operations. Four of the bits (C, V, Z and S) can be tested for use with conditional Jump instructions. Two flags (H and D) cannot be tested and are used for BCD arithmetic. The two remaining bits in the Flag Register (F1 and F2) are available to the user, but they must be set or cleared by instructions and are not usable with conditional Jumps.

As with bits in the other control registers, the Flag Register bits 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 (Load Immediate).

Note: The Watch-Dog Timer (WDT) instruction effects the Flags accordingly: Z=1, S=0, V=0.

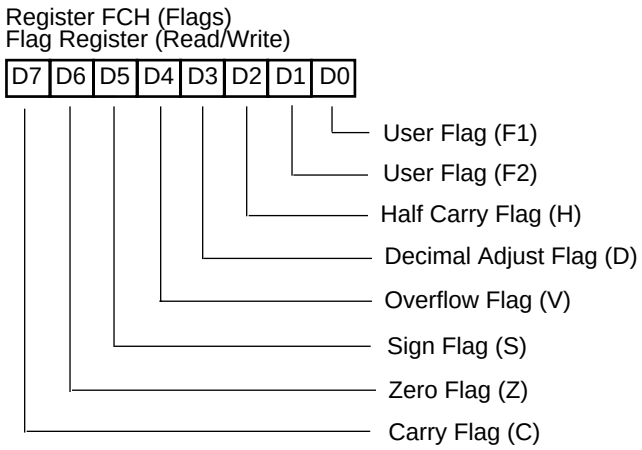


Figure 12-1. Z8 Flag Register

12.2.1 Carry Flag (C)

The Carry Flag is set to 1 whenever the result of an arithmetic operation generates a carry or a borrow the high order bit 7. Otherwise, the Carry Flag is cleared to 0.

Following Rotate and Shift instructions, the Carry Flag contains the last value shifted out of the specified register.

An instruction can set, reset, or complement the Carry Flag.

IRET may change the value of the Carry Flag when the Flag Register, saved in the Stack, is restored.

12.2.2 Zero Flag (Z)

For arithmetic and logical operations, the Zero Flag is set to 1 if the result is zero. Otherwise, the Zero Flag is cleared to 0.

If the result of testing bits in a register is 00H, the Zero Flag is set to 1. Otherwise the Zero Flag is cleared to 0.

If the result of a Rotate or Shift operation is 00H, the Zero Flag is set to 1. Otherwise, the Zero Flag is cleared to 0.

IRET changes the value of the Zero Flag when the Flag Register saved in the Stack is restored. The WDT Instruction sets the Zero Flag to a 1.

12.2.3 Sign Flag (S)

The Sign Flag stores the value of the most significant bit of a result following an arithmetic, logical, Rotate, or Shift operation.

When performing arithmetic operations on signed numbers, binary two's-complement notation is used to represent and process information. A positive number is identified by a 0 in the most significant bit position (bit 7); therefore, the Sign Flag is also 0.

A negative number is identified by a 1 in the most significant bit position (bit 7); therefore, the Sign Flag is also 1.

IRET changes the value of the Sign Flag when the Flag Register saved in the Stack is restored.

12.2.4 Overflow Flag (V)

For signed arithmetic, Rotate, and Shift operations, the Overflow Flag is set to 1 when the result is greater than the maximum possible number (>127) or less than the minimum possible number (<–128) that can be represented in two's-complement form. The Overflow Flag is set to 0 if no overflow occurs.

Following logical operations the Overflow Flag is set to 0.

IRET changes the value of the Overflow Flag when the Flag Register saved in the Stack is restored.

12.2.5 Decimal Adjust Flag (D)

The Decimal Adjust Flag is used for BCD arithmetic. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag specifies what type of instruction was last executed so that the subsequent Decimal Adjust (DA) operation can function properly. Normally, the Decimal Adjust Flag cannot be used as a test condition.

After a subtraction, the Decimal Adjust Flag is set to 1. Following an addition it is cleared to 0.

IRET changes the value of the Decimal Adjust Flag when the Flag Register saved in the Stack is restored.

12.2.6 Half Carry Flag (H)

The Half Carry Flag is set to 1 whenever an addition generates a carry bit 3 (Overflow) or a subtraction generates a borrow bit 3. The Half Carry Flag is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. As in the case of the Decimal Adjust Flag, the user does not normally access this flag.

IRET changes the value of the Half Carry Flag when the Flag Register saved in the Stack is restored.

12.3 CONDITION CODES

The C, Z, S, and V Flags control the operation of the 'Conditional' Jump instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 4-7 of the conditional instructions.

Condition codes and flag settings are summarized in Tables 12-9, 12-10, and 12-11. Notation for the flags and how they are affected are as follows:

Table 12-9. Z8 Flag Definitions

Flag	Description
C	Carry Flag
Z	Zero Flag
S	Sign Flag
V	Overflow Flag
D	Decimal Adjust Flag
H	Half Carry Flag

Table 12-10. Flag Settings Definitions

Symbol	Definition
0	Cleared to 0
1	Set to 1
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Table 12-11. Condition Codes

Binary	HEX	Mnemonic	Definition	Flag Settings
0000	0	F	Always False	–
1000	8	(blank)	Always True	–
0111	7	C	Carry	C = 1
1111	F	NC	No Carry	C = 0
0110	6	Z	Zero	Z = 1
1110	E	NZ	Non-Zero	Z = 0
1101	D	PL	Plus	S = 0
0101	5	Ml	Minus	S = 1
0100	4	OV	Overflow	V = 1
1100	C	NOV	No Overflow	V = 0
0110	6	EQ	Equal	Z = 1
1110	E	NE	Not Equal	Z = 0
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
0001	1	LT	Less Than	(S XOR V) = 1
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
1111	F	UGE	Unsigned Greater Than or Equal	C = 0
0111	7	ULT	Unsigned Less Than	C = 1
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1

12.4 NOTATION AND BINARY ENCODING

In the detailed instruction descriptions that make up the rest of this chapter, operands and status flags are represented by a notational shorthand. Operands, condition codes, address modes, and their notations are as follows (Table 12-12):

Table 12-12. Notational Shorthand

Notation	Address Mode	Operand	Range *
cc	Condition Code		See condition codes
r	Working Register	Rn	n = 0 – 15
R	Register or Working Register	Reg Rn	Reg. represents a number in the range of 00H to FFH n = 0 – 15
RR	Register Pair or Working Register Pair	Reg RRp	Reg. represents an even number in the range of 00H to FEH p = 0, 2, 4, 6, 8, 10, 12, or 14
Ir	Indirect Working Register	@Rn	n = 0 –15
IR	Indirect Register or Indirect Working Register	@Reg @Rn	Reg. represents a number in the range of 00H to FFH n = 0– 15
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair or Working Register Pair	@Reg @RRp	Reg. represents an even number in the range 00H to FFH p = 0, 2, 4, 6, 8, 10, 12, or 14
X	Indexed	Reg (Rn)	Reg. represents a number in the range of 00H to FFH and n = 0 – 15
DA	Direct Address	Addrs	Addrs. represents a number in the range of 00H to FFH
RA	Relative Address	Addrs	Addrs. represents a number in the range of +127 to –128 which is an offset relative to the address of the next instruction
IM	Immediate	#Data	Data is a number between 00H to FFH

*See the device product specification to determine the exact register file range available. The register file size varies by the device type.

Additional symbols used are:

Table 12-13. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag Register (FCH)
RP	Register Pointer (FDH)
IMR	Interrupt Mask Register (FBH)
#	Immediate Operand Prefix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix
B	Binary Number Suffix
OPC	Opcode

Assignment of a value is indicated by the symbol “¨”. For example,

dst ¨ dst + src

indicates 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.

12.4.1 Assembly Language Syntax

For proper instruction execution, Z8 assembly language syntax requires ‘dst, src’ be specified, in that order. The following instruction descriptions show the format of the object code produced by the assembler. This binary format should be followed by users who prefer manual program coding or who intend to implement their own assembler.

Example: If the contents of registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code is:

ASM:	ADD	43H,	08H	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

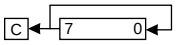
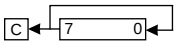
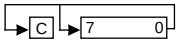
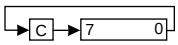
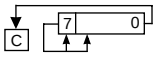
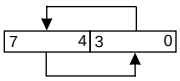
In general, whenever an instruction format requires an 8-bit register address, that address can specify any register location in the range 0 - 255 or a Working Register R0 - R15. If, in the above example, register 08H is a Working Register, the assembly syntax and resulting object code would be:

ASM:	ADD	43H,	08H	(ADD dst, src)
OBJ:	04	08	43	(OPC src, dst)

Note: See the device product specification to determine the exact register file range available. The register file size varies by device type

12.5 Z8 INSTRUCTION SUMMARY

Instruction and Operation	Address Mode dst	Address Mode src	Op Code Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst ← dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst ← dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst ← dst AND src	†		5[ ]	–	*	*	0	–	–
CALL dst SP ← SP – 2 and PC ← dst or @ SP ← PC	DA IRR		D6 D4	–	–	–	–	–	–
CCF C ←NOT C			EF	*	–	–	–	–	–
CLR dst dst ← 0	R IR		B0 B1	–	–	–	–	–	–
COM dst dst ← NOT dst	R IR		60 61	–	*	*	0	–	–
CP dst, src dst – src	†		A[ ]	*	*	*	*	–	–
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	–	–
DEC dst dst ← dst – 1	R IR		00 01	–	*	*	*	–	–
DECW dst dst ← dst – 1	R R IR		80 81	–	*	*	*	–	–
DI dst IMR(7) ← 0			8 F	–	–	–	–	–	–
DJNZr, dst r ← r – 1 if r ≠ 0 PC ← PC + dst Range:+127, -128	RA		rA r=0-F	–	–	–	–	–	–
EI IMR(7) ← 1			9 F	–	–	–	–	–	–
HALT			7 F	–	–	–	–	–	–
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	–	*	*	*	–	–
INCW dst dst ← dst + 1	RR IR		A0 A1	–	*	*	*	–	–
IRET FLAGS←@SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; and IMR(7) – 1			B F	*	*	*	*	*	*
JP cc, dst if cc is true, then PC ← dst	DA IRR		cD c = 0 – F 30	–	–	–	–	–	–
JR cc, dst if cc is true, PC ← PC + dst Range: +127, –128	RA		cB c = 0 – F	–	–	–	–	–	–
LD dst, src dst ← src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	r C r 8 r 9 r = 0 – F C 7 D 7 E 3 F 3 E 4 E 5 E 6 E 7 F 5	–	–	–	–	–	–
LDC dst, src dst ← src	r Irr	Irr r	C 2 D 2	–	–	–	–	–	–
LDCI dst, src dst ← src r ← r + 1 or rr ←rr + 1	Ir Irr	Irr r	C 3 D 3	–	–	–	–	–	–
LDE dst, src dst ← src	r Irr	Irr r	82 92	–	–	–	–	–	–
LDEI dst, src dst ← src and r ← r + 1 or rr ←rr + 1	r Irr	Irr r	C 2 D 2	–	–	–	–	–	–
NOP			FF	–	–	–	–	–	–
OR dst, src dst ← dst OR src	†		4[ ]	–	*	*	0	–	–

Instruction and Operation	Address Mode dst	Address Mode src	Op Code Byte (Hex)	Flags Affected C	Z	S	V	D	H
POP dst dst ← @SP and SP ← SP + 1	R IR		50 51	–	–	–	–	–	–
PUSH src SP ← SP – 1 and @SP ← src	R IR		70 71	–	–	–	–	–	–
RCF C ← 0			C F	0	–	–	–	–	–
RET PC ← @SP; SP ← SP + 2			A F	–	–	–	–	–	–
RL dst 	R IR		90 91	*	*	*	*	–	–
RLC dst 	R IR		10 11	*	*	*	*	–	–
RR dst 	R IR		E 0 E 1	*	*	*	*	–	–
RRC dst 	R IR		C 0 C 1	*	*	*	*	–	–
SBC dst, src dst ← dst – src – C	†		3[ ]	*	*	*	*	1	*
SCF C ← 1			D F	1	–	–	–	–	–
SRA dst 	R IR		D 0 D 1	*	*	*	0	–	–
SRP dst RP ← src	Im		31	–	–	–	–	–	–
STOP			6 F	–	–	–	–	–	–
SUB dst, src dst ← dst – src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	–	–

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
TCM dst, src (NOT dst) AND src	†		6[ ]	–	*	*	0	–	–
TM dst, src dst AND src	†		7[ ]	–	*	*	0	–	–
WDH			4 F	–	X	X	X	–	–
WDT			5 F	–	X	X	X	–	–
XOR dst, src dst AND src XOR src	†		7[ ]	–	*	*	0	–	–

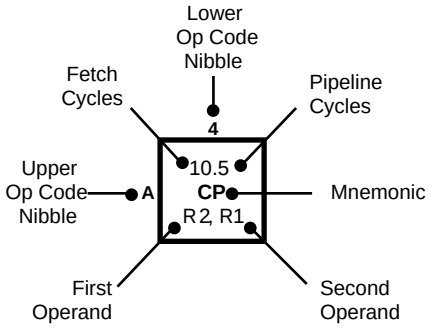
Note: † These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair. For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address dst	Mode src	Lower Op Code Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

12.5.1 Op Code Map

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								6.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** lrr1, r2	18.0 **LDCI** lrr1, Ir2	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2				3				2				3		1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: Blank areas are reserved.

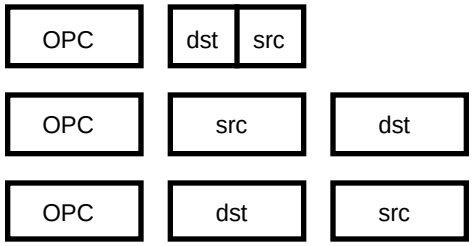
*2-byte instruction appears as a 3-byte instruction

12.6 INSTRUCTION DESCRIPTION AND FORMATS

ADC
ADD WITH CARRY

ADC
Add With Carry
ADC dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	12	r	r
6	13	r	Ir
10	14	R	R
10	15	R	IR
10	16	R	IM
10	17	IR	IM

Operation:

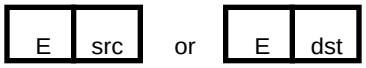
dst <— dst + src + C
The source operand, along with the setting of the Carry (C) Flag, is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected. In multiple precision arithmetic, this instruction permits the carry from the addition of low order operands to be carried into the addition of high order operands.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is zero; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared.
H: Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise.

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R3 contains 16H, the C Flag is set to 1, and Working Register R11 contains 20H, the statement:

ADC R3, R11
Op Code: 12 3B

leaves the value 37H in Working Register R3. The C, Z, S, V, D, and H Flags are all cleared.

ADC
ADD WITH CARRY

Example:

If Working Register R16 contains 16H, the C Flag is not set, Working Register R10 contains 20H, and Register 20H contains 11H, the statement:

ADC R16, @R10
Op Code: 13 FA

leaves the value 27H in Working Register R16. The C, Z, S, V, D, and H Flags are all cleared.

Example

If Register 34H contains 2EH, the C Flag is set, and Register 12H contains 1BH, the statement:

ADC 34H, 12H
Op Code: 14 12 34

leaves the value 4AH in Register 34H. The H Flag is set, and the C, Z, S, V, and D Flags are cleared.

Example:

If Register 4BH contains 82H, the C Flag is set, Working Register R3 contains 10H, and Register 10H contains 01H, the statement:

ADC 4BH, @R3
Op Code: 15 E3 4B

leaves the value 84H in Register 4BH. The S Flag is set, and the C, Z, V, D, and H Flags are cleared.

Example:

If Register 6CH contains 2AH, and the C Flag is not set, the statement:

ADC 6CH, #03H
Op Code: 16 6C 03

leaves the value 2DH in Register 6CH. The C, Z, S, V, D, and H Flags are all cleared.

Example:

If Register D4H contains 5FH, Register 5FH contains 4CH, and the C Flag is set, the statement:

ADC @D4H, #02H
Op Code: 17 D4 02

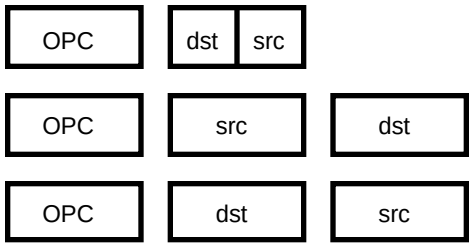
leaves the value 4FH in Register 5FH. The C, Z, S, V, D, and H Flags are all cleared.

ADD
ADD

ADD
Add

ADD dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	02	r	r
6	03	r	Ir
10	04	R	R
10	05	R	IR
10	06	R	IM
10	07	IR	IM

Operation:

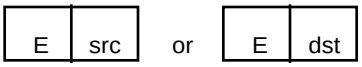
dst <— dst + src
The source operand is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is zero; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared.
H: Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise.

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

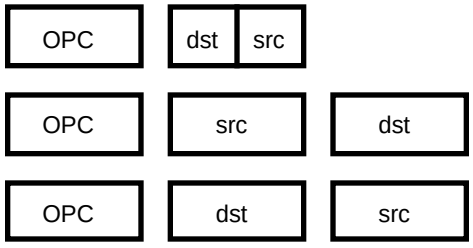
If Working Register R3 contains 16H and Working Register R11 contains 20H, the statement:

ADD
ADD

ADD
Add

ADD dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	02	r	r
6	03	r	Ir
10	04	R	R
10	05	R	IR
10	06	R	IM
10	07	IR	IM

Operation:

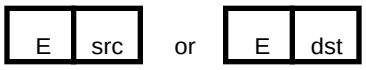
dst <— dst + src
The source operand is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is zero; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared.
H: Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise.

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the OpCode.



Example:

If Working Register R3 contains 16H and Working Register R11 contains 20H, the statement:

ADD R3, R11
OpCode: 02 3B

leaves the value 36H in Working Register R3. The C, Z, S, V, D, and H Flags are all cleared.

ADD
ADD

Example:

If Working Register R16 contains 16H, Working Register R10 contains 20H, and Register 20H contains 11H, the statement:

ADD R16, @R10
Op Code: 03 FA

leaves the value 27H in Working Register R16. The C, Z, S, V, D, and H Flags are all cleared.

Example:

If Register 34H contains 2EH and Register 12H contains 1BH, the statement:

ADD 34H, 12H
Op Code: 04 12 34

leaves the value 49H in Register 34H. The H Flag is set, and the C, Z, S, V, and D Flags are cleared.

Example

If Register 4BH contains 82H, Working Register R3 contains 10H, and Register 10H contains 01H, the statement:

ADD 3EH, @R3
Op Code: 05 E3 4B

leaves the value 83H in Register 4BH. The S Flag is set, and the C, Z, V, D, and H Flags are cleared.

Example:

If Register 6CH contains 2AH, the statement:

ADD 6CH, #03H
Op Code: 06 6C 03

leaves the value 2DH in Register 6CH. The C, Z, S, V, D, and H Flags are all cleared.

Example:

If Register D4H contains 5FH and Register 5FH contains 4CH, the statement:

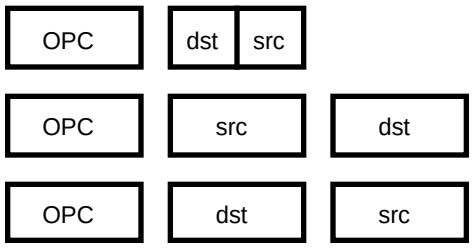
ADD @D4H, #02H
Op Code: 07 D4 02

leaves the value 4EH in Register 5FH. The C, Z, S, V, D, and H Flags are all cleared.

AND
Logical AND

AND
Logical AND
AND dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	52	r	r
6	53	r	Ir
10	54	R	R
10	55	R	IR
10	56	R	IM
10	57	IR	IM

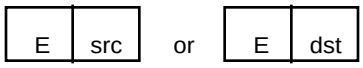
Operation:

dst <— dst AND src
The source operand is logically ANDed with the destination operand. The AND operation results in a 1 being stored whenever the corresponding bits in the two operands are both 1, otherwise a 0 is stored. The result is stored in the destination operand. The contents of the source bit are not affected.

Flags:
- C: Unaffected
- Z: Set if the result is zero; cleared otherwise
- S: Set if the result of bit 7 is set; cleared otherwise
- V: Always reset to 0
- D: Unaffected
- H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R1 contains 34H (00111000B) and Working Register R14 contains 4DH (10001101), the statement:

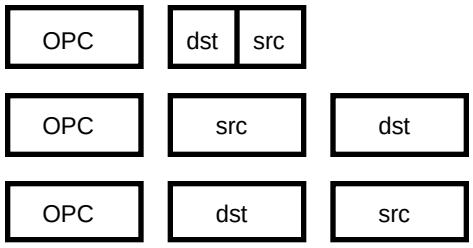
AND R1, R14
Op Code: 52 1E

ADD
ADD

ADD
Add:

ADD dst, src:

Instruction Format:

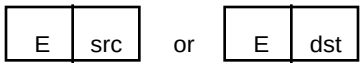


Cycles	OPC (Hex)	Address dst	Mode src
6	02	r	r
6	03	r	Ir
10	04	R	R
10	05	R	IR
10	06	R	IM
10	07	IR	IM

Operation: dst <— dst + src
The source operand is added to the destination operand. Two's complement addition is performed. The sum is stored in the destination operand. The contents of the source operand are not affected.

Flags:
- C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
- Z: Set if the result is zero; cleared otherwise.
- S: Set if the result is negative; cleared otherwise.
- V: Set if an arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
- D: Always cleared.
- H: Set if there is a carry from the most significant bit of the low order four bits of the result; cleared otherwise.

Note: Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the OpCode.



Example: If Working Register R3 contains 16H and Working Register R11 contains 20H, the statement:

ADD R3, R11
OpCode: 02 3B

leaves the value 36H in Working Register R3. The C, Z, S, V, D, and H Flags are all cleared.

AND
Logical AND

Example:

If Working Register R4 contains F9H (11111001B), Working Register R13 contains 7BH, and Register 7BH contains 6AH (01101010B), the statement:

AND R4, @R13
Op Code: 53 4D

leaves the value 68H (01101000B) in Working Register R4. The Z, V, and S Flags are cleared.

Example:

If Register 3AH contains the value F5H (11110101B) and Register 42H contains the value 0AH (00001010), the statement:

AND 3AH, 42H
Op Code: 54 42 3A

leaves the value 00H (00000000B) in Register 3AH. The Z Flag is set, and the V and S Flags are cleared.

Example:

If Working Register R5 contains F0H (11110000B), Register 45H contains 3AH, and Register 3AH contains 7FH (01111111B), the statement:

AND R5, @45H
Op Code: 55 45 E5

leaves the value 70H (01110000B) in Working Register R5. The Z, V, and S Flags are cleared.

Example:

If Register 7AH contains the value F7H (11110111B), the statement:

AND 7AH, #F0H
Op Code: 56 7A F0

leaves the value F0H (11110000B) in Register 7AH. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Working Register R3 contains the value 3EH and Register 3EH contains the value ECH (11101100B), the statement:

AND @R3, #05H
Op Code: 57 E3 05

leaves the value 04H (00000100B) in Register 3EH. The Z, V, and S Flags are cleared.

CALL
CALL PROCEDURE

CALL
Call Procedure
CALL dst

Instruction Format:

	Cycles	OPC (Hex)	Address Mode dst
OPC dst	20	D6	DA
OPC dst	20	D4	IRR

Operation:

SP <— SP - 2
@SP <— PC
PC <— dst
The Stack pointer is decremented by two, the current contents of the Program Counter (PC) (address of the first instruction following the CALL instruction) are pushed onto the top of the Stack, and the specified destination address is then loaded into the PC. The PC now points to the first instruction of the procedure. At the end of the procedure a RET (return) instruction can be used to return to the original program flow. RET will pop the top of the Stack and replace the original value into the PC.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address mode IRR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register Pair RR12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.

E	dst

CALL
Call Procedure

Example:

If the contents of the PC are 1A47H and the contents of the SP (Registers FEH and FFH) are 3002H, the statement:

CALL 3521H
Op Code: D6 35 21

causes the SP to be decremented to 3000H, 1A4AH (the address following the CALL instruction) to be stored in external data memory 3000 and 3001H, and the PC to be loaded with 3521H. The PC now points to the address of the first statement in the procedure to be executed.

Example:

If the contents of the PC are 1A47H, the contents of the SP (Register FFH) are 72H, the contents of Register A4H are 34H, and the contents of Register Pair 34H are 3521H, the statement:

CALL @A4H
Op Code: D4 A4

causes the SP to be decremented to 70H, 1A4AH (the address following the CALL instruction) to be stored in R70H and 71H, and the PC to be loaded with 3521H. The PC now points to the address of the first statement in the procedure to be executed.

CCF
COMPLEMENT CARRY FLAG

CCF
Complement Carry Flag
CCF

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	EF

Operation:

C <— NOT C
The C Flag is complemented. If C = 1, then it is changed to C = 0; or, if C = 0, then it is changed to C = 1.

Flags:
C: Complemented
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If the C Flag contains a 0, the statement:

CCF
Op Code: EF

will change the C Flag from C = 0 to C = 1.

CLR
CLEAR

CLR
CLEAR
CLR dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	B0	R
6	B1	IR

Operation:

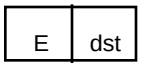
dst <— 0
The destination operand is cleared to 00H.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R6 contains AFH, the statement:

CLR R6
Op Code: B0 E6

will leave the value 00H in Working Register R6.
If Register A5H contains the value 23H, and Register 23H contains the value FCH, the statement:

CLR @A5H
Op Code: B1 A5

will leave the value 00H in Register 23H.

COM
COMPLEMENT

COM
Complement
COM dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	60	R
6	61	IR

Operation:

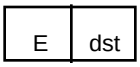
dst <— NOT dst
The contents of the destination operand are complemented (one's complement). All 1 bits are changed to 0, and all 0 bits are changed to 1.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if result bit 7 is set; cleared otherwise.
V: Always reset to 0.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Register 08H contains 24H (00100100B), the statement:

COM 08H
Op Code: 60 08

leaves the value DBH (11011011) in Register 08H. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Register 08H contains 24H, and Register 24H contains FFH (11111111B), the statement:

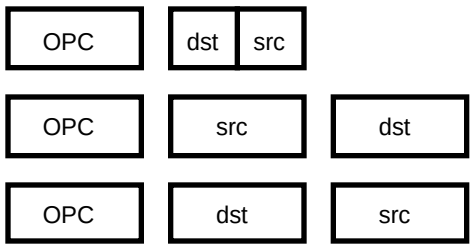
COM @08H
Op Code: 61 08

leaves the value 00H (00000000B) in Register 24H. The Z Flag is set, and the V and S Flags are cleared.

CP
COMPARE

CP
Compare
CP dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	A2	r	r
6	A3	r	Ir
10	A4	R	R
10	A5	R	IR
10	A6	R	IM
10	A7	IR	IM

Operation:

dst - src
The source operand is compared to (subtracted from) the destination operand, and the appropriate flags are set accordingly. The contents of both operands are unaffected.

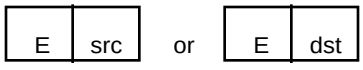
Flags:

C: Cleared if there is a carry from the most significant bit of the result. Set otherwise indicating a borrow.
Z: Set if the result is zero; cleared otherwise.
S: Set if result bit 7 is set (negative); cleared otherwise.
V: Set if arithmetic overflow occurs; cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.

Example:



If Working Register R3 contains 16H and Working Register R11 contains 20H, the statement:

CP R3, R11
Op Code: A2 3B

sets the C and S Flags, and the Z and V Flags are cleared.

CP
COMPARE

Example:

If Working Register R15 contains 16H, Working Register R10 contains 20H, and Register 20H contains 11H, the statement:

CP R16, @R10
Op Code: A3 FA

clears the C, Z, S, and V Flags.

Example:

If Register 34H contains 2EH and Register 12H contains 1BH, the statement:

CP 34H,12H
Op Code: A4 12 34

clears the C, Z, S, and V Flags.

Example:

If Register 4BH contains 82H, Working Register R3 contains 10H, and Register 10H contains 01H, the statement:

CP 4BH, @R3
Op Code: A5 E3 4B

sets the S Flag, and clears the C, Z, and V Flags.

Example:

If Register 6CH contains 2AH, the statement:

CP 6CH, #2AH
Op Code: A6 6C 2A

sets the Z Flag, and the C, S, and V Flags are all cleared.

Example:

If Register D4H contains FCH, and Register FCH contains 8FH, the statement:

CP @D4H, 7FH
Op Code: A7 D4 FF

sets the V Flag, and the C, Z, and S Flags are all cleared.

DA
DECIMAL ADJUST

DA
Decimal Adjust
DA dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
8	40	R
8	41	IR

Operation:

dst <— DA dst

The destination operand is adjusted to form two 4-bit BCD digits following a binary addition or subtraction operation on BCD encoded bytes. For addition (ADD and ADC) or subtraction (SUB and SBC), the following table indicates the operation performed.

Instruction	Carry Before DA	Bits 7-4 Value (HEX)	H Flag Before DA	Bits 3-0 Value (HEX)	Number Added To Byte	Carry After DA
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
	0	0-9	0	0-9	00	0
SUB	0	0-8	1	6-F	FA	0
SBC	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

If the destination operand is not the result of a valid addition or subtraction of BCD digits, the operation is undefined.

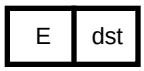
Flags:

C: Set if there is a carry from the most significant bit; cleared otherwise (see table above).
Z: Set if the result is zero; cleared otherwise.
S: Set if result bit 7 is set (negative); cleared otherwise.
D Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For

example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If addition is performed using the BCD value 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic.

```
  0001 0101 =
  15H
+ 0010 0111 =
  27H
-------------
  0011 1100 =
  3CH
```

If the result of the addition is stored in Register 5FH, the statement:

DA 5FH
Op Code: 40 5F

adjusts this result so the correct BCD representation is obtained.

```
0011 1100 =
3CH
0000 0110 =
06H
-----------
0100 0010 =
42H
```

Register 5F now contains the value 42H. The C, Z, and S Flags are cleared, and V is undefined.

Example:

If addition is performed using the BCD value 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic.

```
  0001 0101 =
  15H
+ 0010 0111 =
  27H
-------------
  0011 1100 =
  3CH
```

Register 45F contains the value 5FH, and the result of the addition is stored in Register 5FH, the statement:

DA @45H
Op Code: 40 45

adjusts this result so the correct BCD representation is obtained.

```
0011 1100 = 3CH
0000 0110 = 06H
---------------
0100 0010 = 42H
```

Register 5F now contains the value 42H. The C, Z, and S Flags are cleared, and V is undefined.

DEC
DECREMENT

DEC
Decrement
DEC dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	00	R
6	01	IR

Operation:

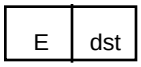
dst <— dst - 1
The contents of the destination operand are decremented by one.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise
S: Set if the result of bit 7 is set (negative); cleared otherwise
V: Set if arithmetic overflow occurs; cleared otherwise
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R10 contains 2A%, the statement:

DEC R10
Op Code: 00 EA

leaves the value 29H in Working Register R10. The Z, V, and S Flags are cleared.

Example:

If Register B3H contains CBH, and Register CBH contains 01H, the statement:

DEC @B3H
Op Code: 01 B3

leaves the value 00H in Register CBH. The Z Flag is set, and the V and S Flags are cleared.

DECW
DECREMENT WORD

DECW
Decrement Word
DECW dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
10	80	RR
10	81	IR

Operation:

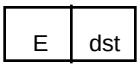
dst <— dst - 1
The contents of the destination (which must be an even address) operand are decremented by one. The destination operand can be a Register Pair or a Working Register Pair.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise
S: Set if the result of bit 7 is set (negative); cleared otherwise
V: Set if arithmetic overflow occurs; cleared otherwise
D: Unaffected
H: Unaffected

Note:

Address modes RR or IR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register Pair R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Register Pair 30H and 31H contain the value 0AF2H, the statement:

DECW 30H
Op Code: 80 30

leaves the value 0AF1H in Register Pair 30H and 31H. The Z, V, and S Flags are cleared.

Example:

If Working Register R0 contains 30H and Register Pairs 30H and 31H contain the value FAF3H, the statement:

DECW @R0
Op Code: 81 E0

leaves the value FAF2H in Register Pair 30H and 31H. The S Flag is set, and the Z and V Flags are cleared.

DI
DISABLE INTERRUPTS

DI
Disable Interrupts
DI

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	8F

Operation:

IMR (7) <— 0
Bit 7 of Control Register FBH (the Interrupt Mask Register) is reset to 0. All interrupts are disabled, although they remain “potentially” enabled. (For instance, the Global Interrupt Enable is cleared, but not the individual interrupt level enables.)

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Control Register FBH contains 8AH (10001010) (interrupts IRQ1 and IRQ3 are enabled), the statement:

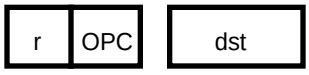
DI
Op Code: 8F

sets Control Register FBH to 0AH (00001010B) and disables these interrupts.

DJNZ
DECREMENT AND JUMP IF NON-ZERO

DJNZ
Decrement and Jump if Non-zero
DJNZ r, dst

Instruction Format:



Cycles		OPC (Hex)	Address Mode dst
12	if jump taken	rA	RA
10	if jump not taken	(r = 0 to F)	

Operation:

r <— r - 1;
If r <> 0, PC <— PC + dst
The specified Working Register being used as a counter is decremented. If the contents of the specified Working Register are not zero after decrementing, then the relative address is added to the Program Counter (PC) and control passes to the statement whose address is now in the PC. The range of the relative address is +127 to –128. The original value of the PC is the address of the instruction byte following the DJNZ statement. When the specified Working Register counter reaches zero, control falls through to the statement following the DJNZ instruction.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

The Working Register being used as a counter must be one of the Registers from 04H to EFH. Use of one of the I/O ports, control or peripheral registers will have undefined results.

Example:

DJNZ is typically used to control a “loop” of instructions. In this example, 12 bytes are moved from one buffer area in the register file to another. The steps involved are:

- Load 12 into the counter (Working Register R6).
- Set up the loop to perform the moves.
- End the loop with DJNZ.

The assembly listing required for this routine is as follows:

```
        LD R6, 12          ;Load Counter
LOOP    LD R9, @R6         ;Move one byte to
:
        LD @R6, R9         ;new location
        DJNZ R6,           ;Decrement and Loop until
        LOOP
                           counter = 0
```

EI
ENABLE INTERRUPTS

EI
Enable Interrupts
EI

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	9F

Operation:

IMR (7) <— 0
Bit 7 of Control Register FBH (the Interrupt Mask Register) is set to 1. This allows potentially enabled interrupts to become enabled.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Control Register FBH contains 0AH (00001010) (interrupts IRQ1 and IRQ3 are selected), the statement:

EI
Op Code: 9F

sets Control Register FBH to 8AH (10001010B) and enables IRQ1 and IRQ3.

HALT
HALT

HALT
Halt
HALT

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	7F

Operation:

The HALT instruction turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and the external interrupts IRQ1, IRQ2, and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

In order to enter HALT mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. The user must execute a NOP immediately before the execution of the HALT instruction.

Example:

Assuming the Z8 is in normal operation, the statements:

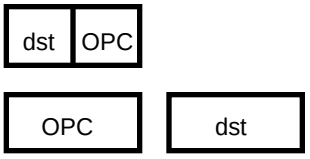
```
NOP
HALT
Op Codes:  FF  7F
```

place the Z8 into HALT mode.

INC
INCREMENT

Inc
Increment

Instruction Format:



Cycles	OPC (Hex)	Address Mode dst
6	rE	r
6	20	R
6	21	IR

Operation:

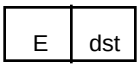
dst <— dst + 1
The contents of the destination operand are incremented by one.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if the result of bit 7 is set (negative); cleared otherwise.
V: Set if arithmetic overflow occurs; cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R10 contains 2AH, the statement:

INC R10
Op Code: AE

leaves the value 2BH in Working Register R10. The Z, V, and S Flags are cleared.

Example:

If Register B3H contains CBH, the statement:

INC B3H
Op Code: 20 B3

leaves the value CCH in Register CBH. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Register B3H contains CBH and Register BCH contains FFH, the statement:

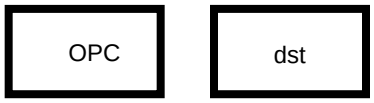
INC @B3H
Op Code: 21 B3

leaves the value 00H in Register CBH. The Z Flag is set, and the V and S Flags are cleared.

INCW
INCREMENT WORD

INCW
Increment Word
INCW dst

Instruction Format:



Cycles	OPC (Hex)	Address Mode dst
10	A0	RR
10	A1	IR
10	A0	R

Operation:

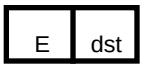
dst <— dst - 1
The contents of the destination (which must be an even address) operand is decremented by one. The destination operand can be a Register Pair or a Working Register Pair.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if the result of bit 7 is set (negative); cleared otherwise.
V: Set if arithmetic overflow occurs; cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes RR or IR can be used to specify a 4-bit Working Register Pair. In this format, the destination Working Register Pair operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register Pair R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code



Example:

If Register Pairs 30H and 31H contain the value 0AF2H, the statement:

INCW 30H
Op Code: A0 30

leaves the value 0AF3H in Register Pair 30H and 31H. The Z, V, and S Flags are cleared.

Example:

If Working Register R0 contains 30H, and Register Pairs 30H and 31H contain the value FAF3H, the statement:

INCW @R0
Op Code: A1 E0

leaves the value FAF4H in Register Pair 30H and 31H. The S Flag is set, and the Z and V Flags are cleared.

IRET
INTERRUPT RETURN

IRET
Interrupt RETURN
IRET

Instruction Format:

	Cycles	OPC (Hex)
OPC	16	BF

Operation:

FLAGS <— @SP
SP <— SP + 1
PC <— @SP
SP <— SP + 2
IMR (7) <— 1
This instruction is issued at the end of an interrupt service routine. It restores the Flag Register (Control Register FCH) and the PC. It also re-enables any interrupts that are potentially enabled.

Flags:

C: Restored to original setting before the interrupt occurred.
Z: Restored to original setting before the interrupt occurred.
S: Restored to original setting before the interrupt occurred.
V: Restored to original setting before the interrupt occurred.
D: Restored to original setting before the interrupt occurred.
H: Restored to original setting before the interrupt occurred.

Example:

If Stack Pointer Low Register FFH currently contains the value 45H, Register 45H contains the value 00H, Register 46H contains 6FH, and Register 47 Contains E4H, the statement:

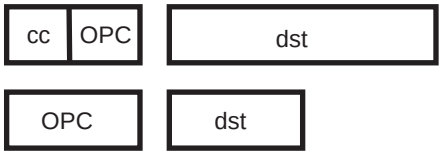
IRET
Op Code: BF

restores the FLAG Register FCH with the value 00H, restores the PC with the value 6FE4H, re-enables the interrupts, and sets the Stack Pointer Low to 48H. The next instruction to be executed will be at location 6FE4H.

JP
JUMP

JP
JUMP
JP cc, dst

Instruction Format:



Cycles		OPC (Hex)	Address Mode dst
12	if jump taken	ccD	DA
10	if not taken	cc = 0 to F	
8		30	IRR

Operation:

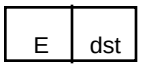
If cc (condition code) is true, then PC <— dst
A conditional jump transfers Program Control to the destination address if the condition specified by cc (condition code) is true. Otherwise, the instruction following the JP instruction is executed. See Section 12.3 for a list of condition codes.
The unconditional jump simply replaces the contents of the Program Counter with the contents of the register pair specified by the destination operand. Program Control then passes to the instruction addressed by the PC.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address mode IRR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If the Carry Flag is set, the statement:

JP C, 1520H
Op Code: 7D 15 20

replaces the contents of the Program Counter with 1520H and transfers program control to that location. If the Carry Flag had not been set, control would have fallen through to the statement following the JP instruction.

Example:

If Working Register Pair RR2 contains the value 3F45H, the statement:

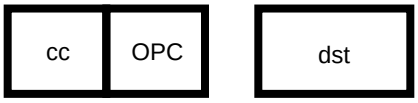
JP @RR2
Op Code: 30 E2

replaces the contents of the PC with the value 3F45H and transfers program control to that location.

JR
JUMP RELATIVE

JR
Jump Relative
JR cc, dst

Instruction Format:



Cycles	OPC (Hex)	Address Mode dst
12 if jump taken	ccB	RR
10 if jump not taken	cc = 0 to F	

Operation:

If cc is true, PC <— PC + dst
If the condition specified by the “cc” is true, the relative address is added to the PC and control passes to the instruction located at the address specified by the PC (See Section 12.3 for a list of condition codes). Otherwise, the instruction following the JR instruction is executed. The range of the relative address is +127 to –128, and the original value of the PC is taken to be the address of the first instruction byte following the JR instruction.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If the result of the last arithmetic operation executed is negative, the next four statements (which occupy a total of seven bytes) are skipped with the statement:

JR MI, #9
Op Code: 5B 09

If the result was not negative, execution would have continued with the instruction following the JR instruction.

Example:

A short form of a jump –45 is:

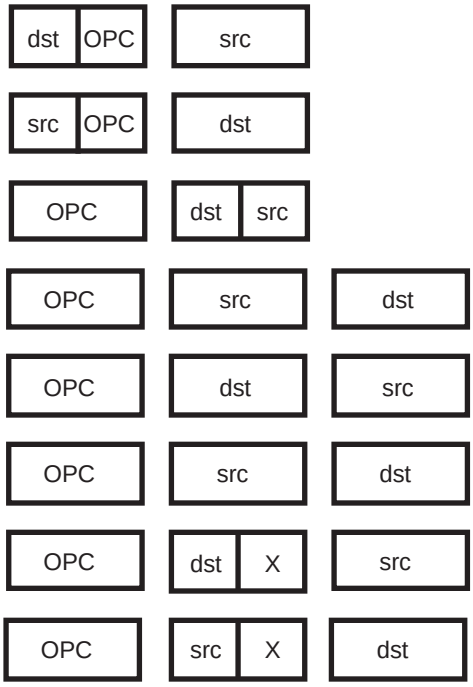
JR #–45
Op Code: 8B D3

The condition code is “blank” in this case, and is assumed to be “always true.”

LD
LOAD

LD
Load
LD dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode dst	Address Mode src
6	rC	r	IM
6	r8	r	R
6	r9 r = 0 to F	R*	r
6	E3	r	Ir
6	F3	Ir	r
10	E4	R	R
10	E5	R	IR
10	E6	R	IM
10	E7	IR	IM
10	F5	IR	R
10	C7	r	X
10	D7	X	r

* In this instance, only a full 8-bit register can be used.

Operation:

dst <— src

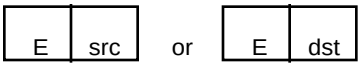
The contents of the source operand are loaded into the destination operand. The contents of the source operand are not affected.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



LD

LOAD

Example:

The statement:

LD R15, #34H
Op Code: FC 34

loads the value 34H into Working Register R15.

Example:

If Register 34H contains the value FCH, the statement:

LD R14, 34H
Op Code: F8 34

loads the value FCH into Working Register R15. The contents of Register 34H are not affected.

Example:

If Working Register R14 contains the value 45H, the statement:

LD 34H, R14
Op Code: E9 34

loads the value 45H into Register 34H. The contents of Working Register R14 are not affected.

Example:

If Working Register R12 contains the value 34H, and Register 34H contains the value FFH, the statement:

LD R13, @R12
Op Code: E3 DC

loads the value FFH into Working Register R13. The contents of Working Register R12 and Register R34 are not affected.

Example:

If Working Register R13 contains the value 45H, and Working Register R12 contains the value 00H the statement:

LD @R13, R12
Op Code: F3 DC

loads the value 00H into Register 45H. The contents of Working Register R12 and Working Register R13 are not affected.

Example:

If Register 45H contains the value CFH, the statement:

LD 34H, 45H
Op Code: E4 45 34

loads the value CFH into Register 34H. The contents of Register 45H are not affected.

LD
LOAD

Example:

If Register 45H contains the value CFH and Register CFH contains the value FFH, the statement:

LD 34H, @45H
Op Code: E5 45 34

loads the value FFH into Register 34H. The contents of Register 45H and Register CFH are not affected.

Example:

The statement:

LD 34H, #A4H
Op Code: E6 34 A4

loads the value A4H into Register 34H.

Example:

If Working Register R14 contains the value 7FH, the statement:

LD @R14, #FCH
Op Code: E7 EE FC

loads the value FCH into Register 7FH. The contents of Working Register R14 are not affected.

Example:

If Register 34H contains the value CFH and Register 45H contains the value FFH, the statement:

LD @34H, 45H
Op Code: F5 45 34

loads the value FFH into Register CFH. The contents of Register 34H and Register 45H are not affected.

Example:

IIf Working Register R0 contains the value 08H and Register 2CH (24H + 08H = 2CH) contains the value 4FH, the statement:

LD R10, 24H(R0)
Op Code: C7 A0 24

loads Working Register R10 with the value 4FH. The contents of Working Register R0 and Register 2CH are not affected.

Example:

If Working Register R0 contains the value 0BH and Working Register R10 contains 83H the statement:

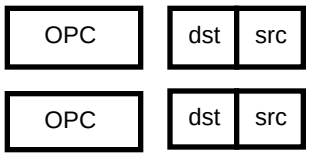
LD F0H(R0), R10
Op Code: D7 A0 F0

loads the value 83H into Register FBH (F0H + 0BH = FBH). Since this is the Interrupt Mask Register, the LOAD statement has the effect of enabling IRQ0 and IRQ1. The contents of Working Registers R0 and R10 are unaffected by the load.

LDC
LOAD CONSTANT

LDC
Load Constant
LDC dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode src	Address Mode dst
12	C2	r	Irr
12	D2	Irr	r

Operation:

dst <— src
This instruction is used to load a byte constant from program memory into a Working Register, or vice versa. The address of the program memory location is specified by a Working Register Pair. The contents of the source operand are not affected.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Working Register Pair R6 and R7 contain the value 30A2H and program memory location 30A2H contains the value 22H, the statement:

LDC R2, @RR6
Op Code: C2 26

loads the value 22H into Working Register R2. The value of program memory location 30A2H is unchanged by the load.

Example:

If Working Register R2 contains the value 22H, and Working Register Pair R6 and R7 contains the value 10A2H, the statement:

LDC @RR6, R2
Op Code: D2 26

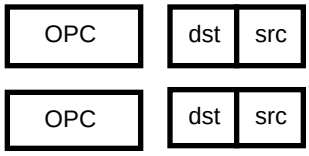
loads the value 22H into program memory location 10A2H. The value of Working Register R2 is unchanged by the load.

Note: This instruction format is valid only for MCUs which can address external program memory.

LDCI
LOAD CONSTANT AUTO-INCREMENT

LDCI
Load Constant Auto-increment
LDCI dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode src	Address Mode dst
18	C3	Ir	Irr
18	D3	Irr	Ir

Operation:

dst <— src
r <— r + 1
rr <— rr + 1
This instruction is used for block transfers of data between program memory and the Register File. The address of the program memory location is specified by a Working Register Pair, and the address of the Register File location is specified by Working Register. The contents of the source location are loaded into the destination location. Both addresses in the Working Registers are then incremented automatically. The contents of the source operand are not affected.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Working Register Pair R6-R7 contains 30A2H, program memory location 30A2H and 30A3H contain 22H and BCH respectively, and Working Register R2 contains 20H, the statement:

LDCI @R2, @RR6
Op Code: C3 26

loads the value 22H into Register 20H. Working Register Pair RR6 is incremented to 30A3H and Working Register R2 is incremented to 21H. A second

LDCI @R2, @RR6
Op Code: C3 26

loads the value BCH into Register 21H. Working Register Pair RR6 is incremented to 30A4H and Working Register R2 is incremented to 22H.

LDCI
LOAD CONSTANT AUTO-INCREMENT

Example:

If Working Register R2 contains 20H, Register 20H contains 22H, Register 21H contains BCH, and Working Register Pair R6-R7 contains 30A2H, the statement:

LDCI @RR6, @R2
Op Code: D3 26

loads the value 22H into program memory location 30A2H. Working Register R2 is incremented to 21H and Working Register Pair R6-R7 is incremented to 30A3H. A second

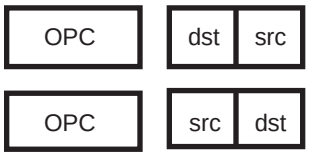
LDCI @RR6, @R2
Op Code: D3 26

loads the value BCH into program memory location 30A3H. Working Register R2 is incremented to 22H and Working Register Pair R6-R7 is incremented to 30A4H.

LDE
LOAD EXTERNAL DATA

LDE
Load External Data
LDE dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode src	Address Mode dst
12	82	r	Irr
12	92	Irr	r

Operation:

dst <— src
This instruction is used to load a byte from external data memory into a Working Register or vice versa. The address of the external data memory location is specified by a Working Register Pair. The contents of the source operand are not affected.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Working Register Pair R6 and R7 contain the value 40A2H and external data memory location 40A2H contains the value 22H, the statement:

LDE R2, @RR6
Op Code: 82 26

loads the value 22H into Working Register R2. The value of external data memory location 40A2H is unchanged by the load.

Example:

If Working Register Pair R6 and R7 contain the value 404AH and Working Register R2 contains the value 22H, the statement:

LDE @RR6, R2
Op Code: 92 26

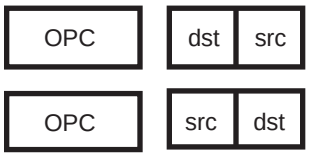
loads the value 22H into external data memory location 404AH

Note: This instruction format is valid only for MCUs which can address external data memory.

LDEI
LOAD EXTERNAL DATA AUTO-INCREMENT

LDEI
Load External Data Auto-increment
LDEI dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode src	Address Mode dst
18	83	Ir	Irr
18	93	Irr	Ir

Operation:

dst <— src
r <— r + 1
rr <— rr + 1
This instruction is used for block transfers of data between external data memory and the Register File. The address of the external data memory location is specified by a Working Register Pair, and the address of the Register File location is specified by a Working Register. The contents of the source location are loaded into the destination location. Both addresses in the Working Registers are then incremented automatically. The contents of the source are not affected.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If Working Register Pair R6 and R7 contains 404AH, external data memory location 404AH and 404BH contain ABH and C3H respectively, and Working Register R2 contains 22H, the statement:

LDEI @R2, @RR6
Op Code: 83 26

loads the value ABH into Register 22H. Working Register Pair RR6 is incremented to 404BH and Working Register R2 is incremented to 23H. A second

LDEI @R2, @RR6
Op Code: 83 26

loads the value C3H into Register 23H. Working Register Pair RR6 is incremented to 404CH and Working Register R2 is incremented to 24H.

LDEI
LOAD EXTERNAL DATA AUTO-INCREMENT

Example:

If Working Register R2 contains 22H, Register 22H contains ABH, Register 23H contains C3H, and Working Register Pair R6 and R7 contains 404AH, the statement:

LDEI @RR6, @R2
Op Code: 93 26

loads the value ABH into external data memory location 404AH. Working Register R2 is incremented to 23H and Working Register Pair RR6 is incremented to 404BH. A second

LDEI @RR6, @R2
Op Code: 93 26

loads the value C3H into external data memory location 404BH. Working Register R2 is incremented to 24H and Working Register Pair RR6 is incremented to 404CH.

Note: This instruction format is valid only for MCUs which can address external data memory.

NOP
NO OPERATION

NOP
No Operation
NOP

Instruction Format:

Cycles	OPC (Hex)
6	FF

Operation

No action is performed by this instruction. It is typically used for timing delays or clearing the pipeline.

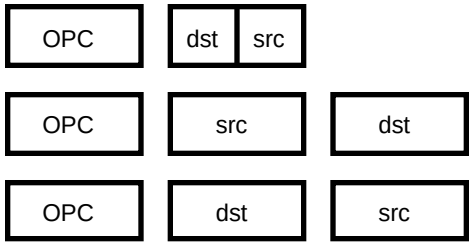
Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

OR
LOGICAL OR

OR
Logical OR
OR dst, src

Instruction Format:



Cycles	OPC (Hex)	Address Mode dst	Address Mode src
6	42	r	r
6	43	r	Ir
10	44	R	R
10	45	R	IR
10	46	R	IM
10	47	IR	IM

Operation:

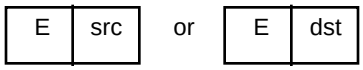
dst <— dst OR src
The source operand is logically ORed with the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. The OR operation results in a one bit being stored whenever either of the corresponding bits in the two operands is a one. Otherwise, a zero bit is stored.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise
S: Set if the result of bit 7 is set; cleared otherwise
V: Always reset to 0
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R1 contains 34H (00111000B) and Working Register R14 contains 4DH (10001101), the statement:

OR R1, R14
Op Code: 42 1E

leaves the value BDH (10111101B) in Working Register R1. The S Flag is set, and the Z and V Flags are cleared.

OR
LOGICAL OR

Example:

If Working Register R4 contains F9H (11111001B), Working Register R13 contains 7BH, and Register 7B contains 6AH (01101010B), the statement:

OR R4, @R13
Op Code: 43 4D

leaves the value FBH (11111011B) in Working Register R4. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Register 3AH contains the value F5H (11110101B) and Register 42H contains the value 0AH (00001010), the statement:

OR 3AH, 42H
Op Code: 44 42 3A

leaves the value FFH (11111111B) in Register 3AH. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Working Register R5 contains 70H (01110000B), Register 45H contains 3AH, and Register 3AH contains 7FH (01111111B), the statement:

OR R5, @45H
Op Code: 45 45 E5

leaves the value 7FH (01111111B) in Working Register R5. The Z, V, and S Flags are cleared.

Example:

If Register 7AH contains the value F3H (11110111B), the statement:

OR 7AH, #F0H
Op Code: 46 7A F0

leaves the value F3H (11110111B) in Register 7AH. The S Flag is set, and the Z and V Flags are cleared.

Example:

If Working Register R3 contains the value 3EH and Register 3EH contains the value 0CH (00001100B), the statement:

OR @R3, #05H
Op Code: 57 E3 05

leaves the value 0DH (00001101B) in Register 3EH. The Z, V, and S Flags are cleared.

POP
POP

POP
Pop
POP dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
10	50	R
10	51	IR

Operation:

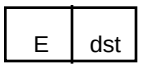
dst <— @SP
SP <— SP + 1
The contents of the location specified by the SP (Stack Pointer) are loaded into the destination operand. The SP is then incremented automatically.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If the SP (Control Registers FEH and FFH) contains the value 70H and Register 70H contains 44H, the statement:

POP 34H
Op Code: 50 34

loads the value 44H into Register 34H. After the POP operation, the SP contains 71H. The contents of Register 70 are not affected.

Example:

If the SP (Control Registers FEH and FFH) contains the value 1000H, external data memory location 1000H contains 55H, and Working Register R6 contains 22H, the statement:

POP @R6
Op Code: 51 E6

loads the value 55H into Register 22H. After the POP operation, the SP contains 1001H. The contents of Working Register R6 are not affected.

PUSH
Push
PUSH src

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
10 Internal Stack 12 External Stack	70	R
10 Internal Stack 10 External Stack	71	IR

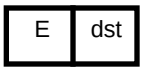
Operation:

SP <— SP - 1
@SP <— src
The contents of the SP (stack pointer) are decremented by one, then the contents of the source operand are loaded into the location addressed by the decremented SP, thus adding a new element to the stack.

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If the SP contains 1001H, the statement:

PUSH FCH
Op Code: 70 FC

stores the contents of Register FCH (the Flag Register) in location 1000H. After the PUSH operation, the SP contains 1000H.

Example:

If the SP contains 61H and Working Register R4 contains FCH, the statement:

PUSH @R4
Op Code: 71 E4

stores the contents of Register FCH (the Flag Register) in location 60H. After the PUSH operation, the SP contains 60H.

RCF
RESET CARRY FLAG

RCF
Reset Carry Flag
RCF

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	CF

Operation:

C <— 0
The C Flag is reset to 0, regardless of its previous value.

Flags:

C: Reset to 0
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Example:

If the C Flag is currently set, the statement:

RCF
Op Code: CF

resets the Carry Flag to 0.

RET
RETURN

RET
Return
RET

Instruction Format:

	Cycles	OPC (Hex)
OPC	14	AF

Operation:

PC <— @SP
SP <— SP + 2
This instruction is normally used to return from a procedure entered by a CALL instruction. The contents of the location addressed by the SP are popped into the PC. The next statement executed is the one addressed by the new contents of the PC. The stack pointer is also incremented by two.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

Each PUSH instruction executed within the subroutine should be countered with a POP instruction in order to guarantee the SP is at the correct location when the RET instruction is executed. Otherwise the wrong address will be loaded into the PC and the program will not operate as desired.

Example:

If SP contains 2000H, external data memory location 2000H contains 18H, and location 2001H contains B5H, the statement:

RET
Op Code: AF

leaves the value 2002H in the SP, and the PC contains 18B5H, the address of the next instruction to be executed.

RL
ROTATE LEFT

RL
Rotate Left
RL dst

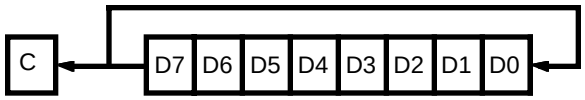
Instruction Format:

	Cycles	OPC (Hex)	Address Mode dst
OPC dst	6	90	R
	6	91	IR

Operation:

C <— dst(7)
dst(0) <— dst(7)
dst(1) <— dst(0)
dst(2) <— dst(1)
dst(3) <— dst(2)
dst(4) <— dst(3)
dst(5) <— dst(4)
dst(6) <— dst(5)
dst(7) <— dst(6)

The contents of the destination operand are rotated left by one bit position. The initial value of bit 7 is moved to the bit 0 position and also into the Carry Flag.

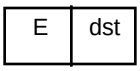


Flags:

C: Set if the bit rotated from the most significant bit position was 1 (i.e., bit 7 was 1).
Z: Set if the result is zero; cleared otherwise.
S: Set if the result in bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



RL
ROTATE LEFT

Example:

If the contents of Register C6H are 88H (10001000B), the statement:

RL C6H
Op Code: 80 C6

leaves the value 11H (00010001B) in Register C6H. The C and V Flags are set, and the S and Z Flags are cleared.

Example:

If the contents of Register C6H are 88H, and the contents of Register 88H are 44H (01000100B), the statement:

RL @C6H
Op Code: 81 C6

leaves the value 88H in Register 88H (10001000B). The S and V Flags are set, and the C and Z Flags are cleared.

RLC
ROTATE LEFT THROUGH CARRY

RLC
Rotate Left Through Carry
RLC dst

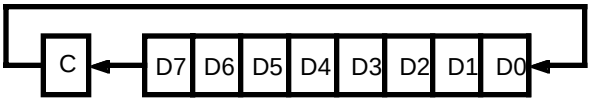
Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	10	R
6	11	IR

Operation:

C<— dst(7)
dst(0) <— C
dst(1) <— dst(0)
dst(2) <— dst(1)
dst(3) <— dst(2)
dst(4) <— dst(3)
dst(5) <— dst(4)
dst(6) <— dst(5)
dst(7) <— dst(6)

The contents of the destination operand along with the C Flag are rotated left by one bit position. The initial value of bit 7 replaces the C Flag and the initial value of the C Flag replaces bit 0.

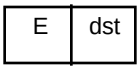


Flags:

C: Set if the bit rotated from the most significant bit position was 1 (i.e., bit 7 was 1).
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



RLC
ROTATE LEFT THROUGH CARRY

Example:

If the C Flag is reset and Register C6 contains 8F (10001111B), the statement:

RLC C6
Op Code: 10 C6

leaves Register C6 with the value 1EH (00011110B). The C and V Flags are set, and S and Z Flags are cleared.

Example:

If the C Flag is reset, Working Register R4 contains C6H, and Register C6 contains 8F (10001111B), the statement:

RLC @R4
Op Code: 11 E4

leaves Register C6 with the value 1EH (00011110B). The C and V Flags are set, and S and Z Flags are cleared.

RR
ROTATE RIGHT

RR
Rotate Right
RR dst

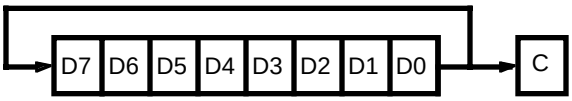
Instruction Format:

	Cycles	OPC (Hex)	Address Mode dst
OPC dst	6	E0	R
	6	E1	IR

Operation:

C <— dst(0)
dst(0) <— dst(1)
dst(1) <— dst(2)
dst(2) <— dst(3)
dst(3) <— dst(4)
dst(4) <— dst(5)
dst(5) <— dst(6)
dst(6) <— dst(7)
dst(7) <— dst(0)

The contents of the destination operand are rotated to the right by one bit position. The initial value of bit 0 is moved to bit 7 and also into the C Flag.

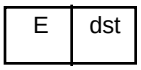


Flags:

C: Set if the bit rotated from the least significant bit position was 1 (i.e., bit 0 was 1).
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



RR
ROTATE RIGHT

Example:

If the contents of Working Register R6 are 31H (00110001B), the statement:

RR R6
Op Code: E0 E6

leaves the value 98H (10011000) in Working Register R6. The C, V, and S Flags are set, and the Z Flag is cleared.

Example:

If the contents of Register C6 are 31H and the contents of Register 31H are 7EH (01111110B), the statement:

RR @C6
Op Code: E1 C6

leaves the value 4FH (00111111) in Register 31H. The C, Z, V, and S Flags are cleared.

RRC
ROTATE RIGHT THROUGH CARRY

RRC
Rotate Right Through Carry
RRC dst

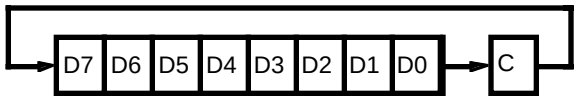
Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	C0	R
6	C1	IR

Operation:

C <— dst(0)
dst(0) <— dst(1)
dst(1) <— dst(2)
dst(2) <— dst(3)
dst(3) <— dst(4)
dst(4) <— dst(5)
dst(5) <— dst(6)
dst(6) <— dst(7)
dst(7) <— C

The contents of the destination operand with the C Flag are rotated right by one bit position. The initial value of bit 0 replaces the C Flag and the initial value of the C Flag replaces bit 7.

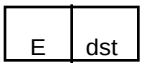


Flags:

C: Set if the bit rotated from the least significant bit position was 1 (i.e., bit 0 was 1).
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred (if the sign of the destination operand changed during rotation); cleared otherwise.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



RRC
ROTATE RIGHT THROUGH CARRY

Example:

If the contents of Register C6H are DDH (11011101B) and the C Flag is reset, the statement:

RRC C6H
Op Code: C0 C6

leaves the value 6EH (01101110B) in register C6H. The C and V Flags are set, and the Z and S Flags are cleared.

Example:

If the contents of Register 2C are EDH, the contents of Register EDH is 00H (00000000B), and the C Flag is reset, the statement:

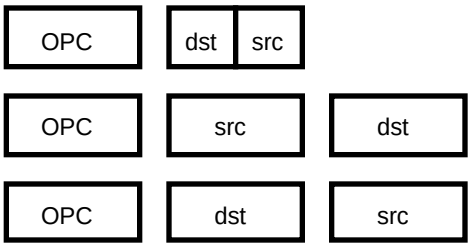
RRC @2CH
Op Code: C1 2C

leaves the value 02H (00000010B) in Register EDH. The C, Z, S, and V Flags are reset.

SBC
SUBTRACT WITH CARRY

SBC
Subtract With Carry
SBC dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	32	r	r
6	33	r	Ir
10	34	R	R
10	35	R	IR
10	36	R	IM
10	37	IR	IM

Operation:

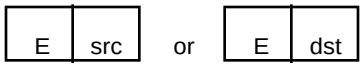
dst <— dst - src - C
The source operand, along with the setting of the C Flag, is subtracted from the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. Subtraction is performed by adding the two's complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry (borrow) from the subtraction of low order operands to be subtracted from the subtraction of high order operands.

Flags:

C: Cleared if there is a carry from the most significant bit of the result; set otherwise, indicating a "borrow."
Z: Set if the result is 0; cleared otherwise.
V: Set if arithmetic overflow occurred (if the operands were of opposite sign and the sign of the result is the same as the sign of the source); reset otherwise.
S: Set if the result is negative; cleared otherwise.
H: Cleared if there is a carry from the most significant bit of the low order four bits of the result; set otherwise indicating a "borrow."
D: Always set to 1.

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



SBC
SUBTRACT WITH CARRY

Example:

Working Register R3 contains 16H, the C Flag is set to 1, and Working Register R11 contains 20H, the statement:

SBC R3, R11
Op Code: 32 3B

leaves the value F5H in Working Register R3. The C, S, and D Flags are set, and the Z, V, and H Flags are all cleared.

Example:

If Working Register R15 contains 16H, the C Flag is not set, Working Register R10 contains 20H, and Register 20H contains 11H, the statement:

SBC R16, @R10
Op Code: 33 FA

leaves the value 05H in Working Register R15. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example

:If Register 34H contains 2EH, the C Flag is set, and Register 12H contains 1BH, the statement:

SBC 34H, 12H
Op Code: 34 12 34

leaves the value 13H in Register 34H. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register 4BH contains 82H, the C Flag is set, Working Register R3 contains 10H, and Register 10H contains 01H, the statement:

SBC 4BH, @R3
Op Code: 35 E3 4B

leaves the value 80H in Register 4BH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register 6CH contains 2AH, and the C Flag is not set, the statement:

SBC 6CH, #03H
Op Code: 36 6C 03

leaves the value 27H in Register 6CH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register D4H contains 5FH, Register 5FH contains 4CH, and the C Flag is set, the statement:

SBC @D4H, #02H
Op Code: 37 D4 02

leaves the value 4AH in Register 5FH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

SCF
SET CARRY FLAG

SCF
Set Carry Flag
SRC

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	DF

Operation:

C <— 1
The C Flag is set to 1, regardless of its previous value.

Flags:

C:	Set to 1
Z	Unaffected
S	Unaffected
V	Unaffected
D	Unaffected
H	Unaffected

Example:

If the C Flag is currently reset, the statement:

SCF
Op Code: DF

sets the Carry Flag to 1.

SRA
SHIFT RIGHT ARITHMETIC

SRA
Shift Right Arithmetic
SRA dst

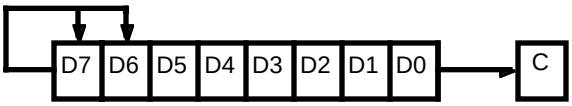
Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	D0	R
6	D1	IR

Operation:

C <— dst(0)
dst(0) <— dst(1)
dst(1) <— dst(2)
dst(2) <— dst(3)
dst(3) <— dst(4)
dst(4) <— dst(5)
dst(5) <— dst(6)
dst(6) <— dst(7)
dst(7) <— dst(7)

An arithmetic shift right by one bit position is performed on the destination operand. Bit 0 replaces the C Flag. Bit 7 (the Sign bit) is unchanged and its value is shifted into bit 6.

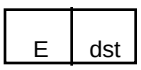


Flags:

C: Set if the bit rotated from the least significant bit position was 1 (i.e., bit 0 was 1).
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to 0.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



SRA
SHIFT RIGHT ARITHMETIC

Example:

If the contents of Working Register R6 are 31H (00110001B), the statement:

SRA R6
Op Code: D0 E6

leaves the value 98H (00011000) in Working Register R6. The C Flag is set, and the Z, V, and S Flags are cleared.

Example:

If Register C6 contains the value DFH, and Register DFH contains the value B8H (10111000B), the statement:

SRA @C6
Op Code: D1 C6

leaves the value DCH (11011100B) in Register DFH. The C, Z, and V Flags are reset, and the S Flag is set.

SRP
SET REGISTER POINTER

SRP
Set Register Pointer
SRP src

Instruction Format:

		Cycles	OPC (Hex)	Address Mode dst
OPC	src	6	31	IM

Operation:

RP <— src
The specified value is loaded into the Register Pointer (RP) (Control Register FDH). Bits 7-4 determine the Working Register Group. Bits 3-0 selects the Expanded Register Bank. Addressing of un-implemented Working Register Group, while using Expanded Register Banks, will point to Bank 0.
Example: SRP TD addresses Working Register Group 7 of Bank 0.

Register Pointer (FDH) Contents (Bin)	Working Register Group (Hex)	Actual Registers (Hex)
1111 0000	F	F0-FF
1110 0000	E	E0-EF
1101 0000	D	D0-DF
1100 0000	C	C0-CF
1011 0000	B	B0-BF
1010 0000	A	A0-AF
1001 0000	9	90-9F
1000 0000	8	80-8F
0111 0000	7	70-7F
0110 0000	6	60-6F
0101 0000	5	50-5F
0100 0000	4	40-4F
0011 0000	3	30-3F
0010 0000	2	20-2F
0001 0000	1	10-1F
0000 0000	0	00-0F

SRP
SET REGISTER POINTER

Register Pointer (FDH) Contents (Hex)	Expanded Register Bank (Hex)	Working Registers (Dec)
xxxx 1111	F	R0-R15
xxxx 1110	E	R0-R15
xxxx 1101	D	R0-R15
xxxx 1100	C	R0-R15
xxxx 1011	B	R0-R15
xxxx 1010	A	R0-R15
xxxx 1001	9	R0-R15
xxxx 1000	8	R0-R15
xxxx 0111	7	R0-R15
xxxx 0110	6	R0-R15
xxxx 0101	5	R0-R15
xxxx 0100	4	R0-R15
xxxx 0011	3	R0-R15
xxxx 0010	2	R0-R15
xxxx 0001	1	R0-R15
xxxx 0000	0	R0-R15

Flags:
C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

When an Expanded Register Bank , other than Bank 0 is selected, access to the Z8 Standard Register File is possible except for the Port Register and general purpose registers 04H to 0FH.
fpr Register Addresses 0H to FH.

Example:

The statement:

SRP F0H Op Code: 31 F0

sets the Register Pointer to access expanded Register Bank 0 and Working Register Group F in the Z8 Standard Register File. All references to Working Registers now affect this group of 16 registers. Registers F0H to FFH can be accessed as Working Registers R0 to R15

Example:

The statement:

SRP 0FH
Op Code: 31 0F

sets the Register Pointer to access Expanded Register Bank F, Reg 00H to Reg 0FH, as the current Working Registers. All references to Working Registers now affect this group of 16 registers. These registers are now accessed as Working Registers R0 to R15. Port Registers are now not accessable.

SRP
SET REGISTER POINTER

Example:

Assume the RP currently addresses the Control and Peripheral Working Register Group and the program has just entered an interrupt service routine. The statement:

SRP 70H
Op Code: 31 70

retains the contents of the Control and Peripheral Registers by setting the RP to 70H (01110000B). Any reference to Working Registers in the interrupt routine will point to registers 70H to 7FH.

STOP
STOP

STOP
Stop
STOP

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	6F

Operation:

This instruction turns off the internal system clock (SCLK) and external crystal (XTAL) oscillation, and reduces the standby current. The STOP mode is terminated by a RESET which causes the processor to restart the application program at address 000CH.

Flags:

C: Unaffected
Z: Unaffected
S: Unaffected
V: Unaffected
D: Unaffected
H: Unaffected

Note:

In order to enter STOP mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. The user must execute a NOP immediately before the execution of the STOP instruction.

Example:

The statements:

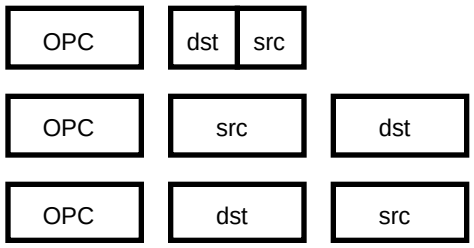
```
NOP
STOP
Op Codes:  FF  6F
```

place the Z8 into STOP mode.

SUB
SUBTRACT

SUB
Subtract
SUB dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	22	r	r
6	23	r	Ir
10	24	R	R
10	25	R	IR
10	26	R	IM
10	27	IR	IM

Operation:

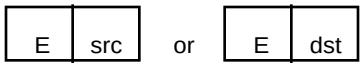
dst <— dst - src
The source operand is subtracted from the destination operand and the result is stored in the destination operand. The contents of the source operand are not affected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:

C: Cleared if there is a carry from the most significant bit of the result; set otherwise, indicating a "borrow."
Z: Set if the result is 0; cleared otherwise.
V: Set if arithmetic overflow occurred (if the operands were of opposite sign and the sign of the result is the same as the sign of the source); reset otherwise.
S: Set if the result is negative; cleared otherwise.
H: Cleared if there is a carry from the most significant bit of the low order four bits of the result; set otherwise indicating a "borrow."
D: Always set to 1.

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

]If Working Register R3 contains 16H, and Working Register R11 contains 20H, the statement:

SUB R3, R11
Op Code: 22 3B

leaves the value F6H in Working Register R3. The C, S, and D Flags are set, and the Z, V, and H Flags are cleared.

SUB
SUBTRACT

Example:

If Working Register R15 contains 16H, Working Register R10 contains 20H, and Register 20H contains 11H, the statement:

SUB R16, @R10
Op Code: 23 FA

leaves the value 05H in Working Register R15. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register 34H contains 2EH, and Register 12H contains 1BH, the statement:

SUB 34H, 12H
Op Code: 24 12 34

leaves the value 13H in Register 34H. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register 4BH contains 82H, Working Register R3 contains 10H, and Register 10H contains 01H, the statement:

SUB 4BH, @R3
Op Code: 25 E3 4B

leaves the value 81H in Register 4BH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register 6CH contains 2AH, the statement:

SUB 6CH, #03H
Op Code: 26 6C 03

leaves the value 27H in Register 6CH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

Example:

If Register D4H contains 5FH, Register 5FH contains 4CH, the statement:

SUB @D4H, #02H
Op Code: 17 D4 02

leaves the value 4AH in Register 5FH. The D Flag is set, and the C, Z, S, V, and H Flags are cleared.

SWAP
SWAP NIBBLES

SWAP
Swap Nibbles
SWAP dst

Instruction Format:

Cycles	OPC (Hex)	Address Mode dst
6	F0	R
6	F1	IR

Operation:

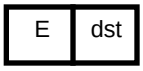
dst(7-4) <—> dst(3-0)
The contents of the lower four bits and upper four bits of the destination operand are swapped.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Undefined
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Register BCH contains B3H (10110011B), the statement:

SWAP B3H
Op Code: F0 B3

will leave the value 3BH (00111011B) in Register BCH. The Z and S Flags are cleared.

Example:

If Working Register R5 contains BCH and Register BCH contains B3H (10110011B), the statement:

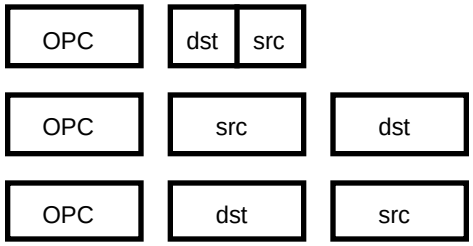
SWAP @R5H
Op Code: F1 E5

will leave the value 3BH (00111011B) in Register BCH. The Z and S Flags are cleared.

TCM
TEST COMPLEMENT UNDER MASK

TCM
Test Complement Under Mask
TCM dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	62	r	r
6	63	r	Ir
10	64	R	R
10	65	R	IR
10	66	R	IM
10	67	IR	IM

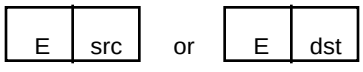
Operation:

(NOT dst) AND src
This instruction tests selected bits in the destination operand for a logical 1 value. The bits to be tested are specified by setting a 1 bit in the corresponding bit position in the source operand (the mask). The TCM instruction complements the destination operand, and then ANDs it with the source mask (operand). The Zero (Z) Flag can then be checked to determine the result. If the Z Flag is set, then the tested bits were 1. When the TCM operation is complete, the destination and source operands still contain their original values.

Flags:
C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to 0.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



TCM
TEST COMPLEMENT UNDER MASK

Example:

If Working Register R3 contains 45H (01000101B) and Working Register R7 contains the value 01H (00000001B) (bit 0 is being tested if it is 1), the statement:

TCM R3, R7
Op Code: 62 37

will set the Z Flag indicating bit 0 in the destination operand is 1. The V and S Flags are cleared.

Example:

If Working Register R14 contains the value F3H (11110011B), Working Register R5 contains CBH, and Register CBH contains 88H (10001000B) (bit 7 and bit 3 are being tested if they are 1), the statement:

TCM R14, @R5
Op Code: 63 E5

will reset the Z Flag, because bit 3 in the destination operand is not a 1. The V and S Flags are also cleared.

Example:

If Register D4H contains the value 04H (000001000B), and Working Register R0 contains the value 80H (10000000B) (bit 7 is being tested if it is 1), the statement:

TCM D4H, R0
Op Code: 64 E0 D4

will reset the Z Flag, because bit 7 in the destination operand is not a 1. The S Flag will be set, and the V Flag will be cleared.

Example:

If Register DFH contains the value FFH (11111111B), Register 07H contains the value 1FH, and Register 1FH contains the value BDH (10111101B) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are being tested if they are 1), the statement:

TCM DFH, @07H
Op Code: 65 07 DF

will set the Z Flag indicating the tested bits in the destination operand are 1. The S and V Flags are cleared.

Example:

If Working Register R13 contains the value F2H (11110010B), the statement:

TCM R13, #02H
Op Code: 66 ED, 02

tests bit 1 of the destination operand for 1. The Z Flag will be set indicating bit 1 in the destination operand was 1. The S and V Flags are cleared.

Example:

If Register 5DH contains A0H, and Register A0H contains 0FH (00001111B), the statement:

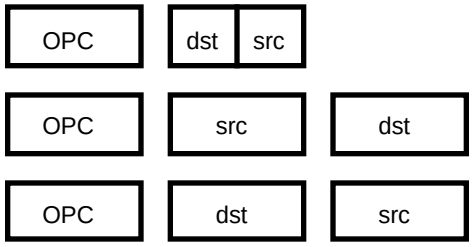
TCM @5D, #10H
Op Code: 67 5D 10

tests bit 4 of the Register A0H for 1. The Z Flag will be reset indicating bit 1 in the destination operand was not 1. The S and V Flags are cleared.

TM
TEST UNDER MASK

TM
Test Under Mask
TM dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	72	r	r
6	73	r	Ir
10	74	R	R
10	75	R	IR
10	76	R	IM
10	77	IR	IM

Operation:

dst AND src

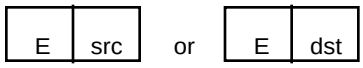
This instruction tests selected bits in the destination operand for a 0 logical value. The bits to be tested are specified by setting a 1 bit in the corresponding bit position in the source operand (the mask). The TM instruction ANDs the destination operand with the source operand (the mask). The Zero (Z) Flag can then be checked to determine the result. If the Z Flag is set, then the tested bits were 0. When the TM operation is complete, the destination and source operands still contain their original values.

Flags:

C: Unaffected
Z: Set if the result is zero; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to 0.
D: Unaffected
H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R3 contains 45H (01000101B) and Working Register R7 contains the value 02H (00000010B) (bit 1 is being tested if it is 0), the statement:

TM R3, R7
Op Code: 72 37

will set the Z Flag indicating bit 1 in the destination operand is 0. The V and S Flags are cleared.

TM
TEST UNDER MASK

Example

Working Register R14 contains the value F3H (11110011B), Working Register R5 contains CBH, and Register CBH contains 88H (10001000B) (bit 7 a bit 3 are being tested if they are 0), the statement:

TM R14, @R5
Op Code: 73 E5

will reset the Z Flag, because bit 7 iin the destination operand is not a 0. The S Flag will be set, and the V Flag is cleared.

Example:

If Register D4H contains the value 08H (00001000B), and Working Register R0 contains the value 04H (00000100B) (bit 2 is being tested if it is 0), the statement:

TM D4H, R0
Op Code: 74 E0 D4

will set the Z Flag, because bit 2 in the destination operand is a 0. The S and V Flags will be cleared.

Example:

If Register DFH contains the value 00H (00000000B), Register 07H contains the value 1FH, and Register 1FH contains the value BDH (10111101B) (bit 7, bit 5, bit 4, bit 3, bit 2, and bit 0 are being tested if they are 0), the statement:

TM DFH, @07H
Op Code: 75 07 DF

will set the Z Flag indicating the tested bits in the destination operand are 0. The S is set, and the V Flag is cleared.

Example:

If Working Register R13 contains the value F1H (11110001B), the statement:

TM R13, #02H
Op Code: 76 ED, 02

tests bit 1 of the destination operand for 0. The Z Flag will be set indicating bit 1 in the destination operand was 0. The S and V Flags are cleared.

Example:

If Register 5DH contains A0H, and Register A0H contains 0FH (00001111B), the statement:

TM @5D, #10H
Op Code: 77 5D 10

tests bit 4 of the Register A0H for 0. The Z Flag will be set indicating bit 4 in the destination operand was 0. The S and V Flags are cleared.

WDH
WATCH-DOG TIMER ENABLE DURING HALT MODE

WDH
Watch-Dog Timer Enable During HALT Mode
WDH

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	4F

Operation:

When this instruction is executed it will enable the WDT (Watch-Dog Timer) during HALT mode. If this instruction is not executed the WDT will stop when entering HALT mode. This instruction does not clear the counter, it just makes it possible to have the WDT function running during HALT mode. A WDH instruction executed without executing WDT (5FH) has no effect.

Flags:

C: Unaffected
Z: Undefined
S: Undefined
V: Undefined
D: Unaffected
H: Unaffected

Note:

The WDH instruction should not be used following any instruction in which the condition of the flags is important.

Example:

If the WDT is enabled, the statement:

WDH
Op Code: .BYTE 4FH

will enable the WDT in HALT mode.

Note: This instruction format is valid only for the Z86C04/C08 and Z86E04/E07/E08.

WDT
WATCH-DOG TIMER

WDT
Watch-Dog Timer
WDT

Instruction Format:

	Cycles	OPC (Hex)
OPC	6	5F

Operation:

The WDT (Watch-Dog Timer) is a retriggerable one shot timer that will reset the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction. Each subsequent execution of the WDT instruction refreshes the timer and prevents the WDT from timing out.

Flags:
C: Unaffected
Z: Undefined
S: Undefined
V: Undefined
D: Unaffected
H: Unaffected

Note:

The WDT instruction should not be used following any instruction in which the condition of the flags is important.

Example:

If the WDT is enabled, the statement:

```
WDT
Op Code:  .BYTE   5FH
```

refreshes the Watch-Dog Timer.

Example:

The first execution of the statement:

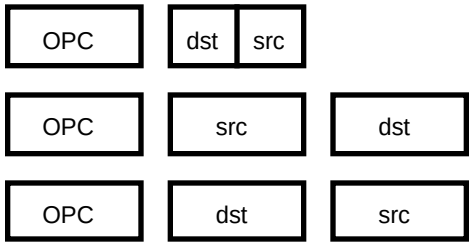
```
WDT
Op Code: .BYTE   5FH
```

enables the Watch-Dog Timer.

XOR
LOGICAL EXCLUSIVE OR

XOR
Logical Exclusive OR
XOR dst, src

Instruction Format:



Cycles	OPC (Hex)	Address dst	Mode src
6	B2	r	r
6	B3	r	Ir
10	B4	R	R
10	B5	R	IR
10	B6	R	IM
10	B7	IR	IM

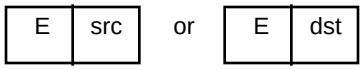
Operation:

dst <— dst XOR src
The source operand is logically EXCLUSIVE ORed with the destination operand. The XOR operation results in a 1 being stored in the destination operand whenever the corresponding bits in the two operands are different, otherwise a 0 is stored. The contents of the source operand are not affected.

Flags:
- C: Unaffected
- Z: Set if the result is zero; cleared otherwise.
- S: Set if the result of bit 7 is set; cleared otherwise.
- V: Always reset to 0
- D: Unaffected
- H: Unaffected

Note:

Address modes R or IR can be used to specify a 4-bit Working Register. In this format, the source or destination Working Register operand is specified by adding 1110B (EH) to the high nibble of the operand. For example, if Working Register R12 (CH) is the destination operand, then ECH will be used as the destination operand in the Op Code.



Example:

If Working Register R1 contains 34H (00111000B) and Working Register R14 contains 4DH (10001101B), the statement:

XOR R1, R14
Op Code: B2 1E

leaves the value BDH (10111101B) in Working Register R1. The Z, and V Flags are cleared, and the S Flag is set.

XOR
LOGICAL EXCLUSIVE OR

Example

If Working Register R4 contains F9H (11111001B), Working Register R13 contains 7BH, and Register 7B contains 6AH (01101010B), the statement:

XOR R4, @R13
Op Code: B3 4D

leaves the value 93H (10010011B) in Working Register R4. The S Flag is set, and the Z, and V Flags are cleared.

Example:

If Register 3AH contains the value F5H (11110101B) and Register 42H contains the value 0AH (00001010B), the statement:

XOR 3AH, 42H
Op Code: B4 42 3A

leaves the value FFH (11111111B) in Register 3AH. The S Flag is set, and the C and V Flags are cleared.

Example:

If Working Register R5 contains F0H (11110000B), Register 45H contains 3AH, and Register 3A contains 7F (01111111B), the statement:

XOR R5, @45H
Op Code: B5 45 E5

leaves the value 8FH (10001111B) in Working Register R5. The S Flag is set, and the C and V Flags are cleared.

Example:

If Register 7AH contains the value F7H (11110111B), the statement:

XOR 7AH, #F0H
Op Code: B6 7A F0

leaves the value 07H (00000111B) in Register 7AH. The Z, V and S Flags are cleared.

Example:

If Working Register R3 contains the value 3EH and Register 3EH contains the value 6CH (01101100B), the statement:

XOR @R3, #05H
Op Code: B7 E3 05

leaves the value 69H (01101001B) in Register 3EH. The Z, V, and S Flags are cleared.