HYNIX SEMICONDUCTOR INC.
8-BIT SINGLE-CHIP MICROCONTROLLERS

HMS91C8032
HMS97C8032

User's Manual (Ver. 1.02)

REVISION HISTORY

VERSION 1.01 (JUL., 2001) sticker

Add the interrupt control block and changed the P2.0 ~ P2.3 pins schematic block.

VERSION 1.02 (NOV., 2001) sticker

Changed Power-On Reset Circuit.

Version 1.02

Published by
MCU Team

Table of Contents

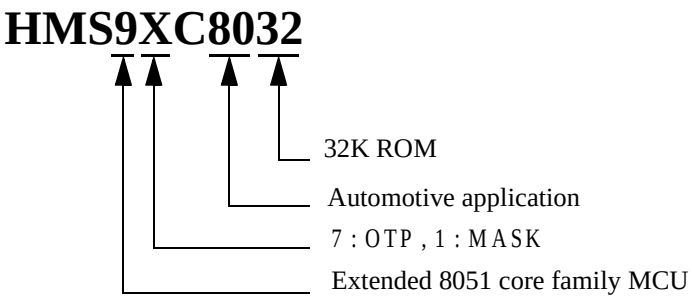
HMS91C8032
HMS97C8032

1. OVERVIEW

1.1 Description

The HMS91C8032 and the HMS97C8032 are a member of the HMS9XC8032 series. This devices are the Digital Tuning System(DTS) with PLL. It has extended Intel 8051 core, 32Kbytes one-time programmable(OTP) ROM. Because this device can be programmed by user, it is suited for applications such as the small-scale production of many different products and rapid development and time-to-market of new products.

- **Extended 8051 core (7.2MHz / 32.768KHz)**
- **1K-Byte Data RAM / 32K-Byte Program ROM**
- **130 MHz Digital PLL block**
- **IFC (Intermediate Frequency Counter)**
- **8-channel 8-bit ADC**
- **Five 16-bit Timers/Counters**
- **Two 3-wire SIO & One UART**
- **18 Interrupts Sources(7 External Interrupts / 5 Timer Interrupts / 3 Serial Port Interrupts / WDT Interrupt / IF Counter Interrupt / ADC Interrupt), Two Priority Levels**
- **Two Power Saving Mode (Idle Mode and Power Down Modes)**
- **5V ±10% Power supply**
- **80-MQFP Package**



1.2 Ordering Information

Device name	ROM Size (bytes)	RAM size	Package	
HMS91C8032	32K	1024 bytes	80MQFP	Mask ROM version
HMS97C8032	32K bytes OTP	1024 bytes	80MQFP	OTP ROM version

1.3 Features

Item		Features
ROM		32K x 8-bit
RAM		1K x 8-bit
Instruction Cycle		With variable instruction execution time function 1.66µs / 3.33µs / 26.6µs (with main system clock : 7.2MHz) 366.2µs (with sub-system clock : 32.768KHz)
Instruction Set		MCS-51 Micro-controller Compatible Instruction Set
I/O Port		CMOS I/O : 62 pins (including 4-open drain ports)
A/D Converter		8-bit resolution x 8-channels
Serial Interface		3-wire serial I/O mode : 2 channels Full duplex UART : 1 channel
Timer/Counter		Five 16-bit timers/event counters Dedicated Watchdog timer
Buzzer(Beep) Output		1.2KHz (fx/6000), 2.4KHz (fx/3000), 4.5KHz (fx/1600), 8.0KHz (fx/900)
Interrupt Source		7 External, 11 Internal Sources
PLL Frequency Synthesizer	Division Mode	Direct division mode (VCOL pin) Pulse swallow mode (VCOH and VCOL pins)
	Reference Frequency	13 types selected by program 1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 18, 20, 25, 50KHz
	Charge Pump	Error out : EO pin
	Phase Detector	Unlock detectable by program
Frequency Counter		Frequency Measurement AMIFC pin : for 450KHz count FMIFC pin : for 10.7MHz count
Standby Function		Idle mode Power-down mode
Reset		Reset by RESET pin Reset by Vdet circuit Vdet circuit: Detection of less than 2.7V (Normal operation mode)
Power Supply Voltage		VDD = 4.5V to 5.5V (with PLL operating) VDD > 1.8V (Data retention mode)
System Clock		Main system clock : 7.2MHz Sub-system clock : 32.768KHz
Package		80 pin plastic MQFP

1.4 Pin Description

Pin Names	Port Names	Alternatives	Functions		Reset
1 2 3 4 5 6 7 8	P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7		8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.	LED drive ability.	Input
9 10 11 12 13 14 15 16	P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7		8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.		Input
17 18 19 20 21 22 23 24	P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	N-ch N-ch N-ch N-ch	8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, P2.4~P2.7pin can use on-chip pullup resister by soft-ware. P2.0~P2.3pin have no pullup	N-channel open drain (P2.0 - P2.3) N-channel open drain voltage : Max. 6V	Input
25 26 27 28 29 30	P3.0 P3.1 P3.2 P3.3 P3.4 P3.5	T0 T1 T2 T3 T4 T2EX	6-bit general purpose bidirectional Pin Input and Output mode selected by 8-bit Port Mode Register.	P3.0 - P3.5 pin can use Timer input pin	Input
31 41 71	VSS1 VSS2 VSS3		Ground		-
32 50 74	VDD! VDD2 VDD3		DC Supply Voltage is 5V +/- 10%. In Power down mode, RAM data guaranteed until 1.8V All VDD pin is connected in system. VDD1 : I/O VDD, VDD2 : core VDD, VDD3 : analog VDD		-
33 34 35 36 37 38 39 40	P4.0 P4.1 P4.2 P4.3 P4.4 P4.5 P4.6 P4.7	INT0 INT1 INT2 INT3 INT4 INT5 INT6 BEEP	8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.	P4.0 - P4.6 is External Interrupt input pin. level/falling detect : 2 pin level/edge detect : 5 pin P4.7 is beep clock putput.	Input

41 42 43 44 45 46 47 48	P5.0 P5.1 P5.2 P5.3 P5.4 P5.5 P5.6 P5.7	TxD RxD SCK1 SO1 SI1 SCK2 SO2 SI2	8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.	TxD, RxD : Asynchronous serial data pin SI1, SI2, SO1, SO2 : Synchronous serial data pin SCK1, SCK2 : Clock pin for Synchronous serial data	Input
49 50 51 52 53 54 55 56	P6.0 P6.1 P6.2 P6.3 P6.4 P6.5 P6.6 P6.7		8-bit general purpose bidirectional Pin Input and Output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.		Input
57	TstEn	Ground	This pin is oniy ground. Chip test pin		GND
60	Avref+		A/D converter reference voltage input pin In AD converter, signal from ANI0 ~ ANI7 change to digital signal by reference between AVref+ and VSS.		
61 62 63 64 65 66 67 68	P7.0 P7.1 P7.2 P7.3 P7.4 P7.5 P7.6 P7.7	ANI0 ANI1 ANI2 ANI3 ANI4 ANI5 ANI6 ANI7	8-bit general purpose bidirectional Pin Input and output mode selected by 8-bit Port Mode Register. In Input mode, pin can use on-chip pullup resister by software.	A/D converter 8-channel analog input pin If pin is not used by A/D converter input, can use to general-purpose bidirectional pin. Input voltage in ANI0 - ANI7 is between Avref+ and VSS.	Input
69	AMIFC		AM IF input pin		
70	FMIFC		FM IF input pin		
72	VCOH		FM band VCO frequency input pin		
73	VCOL		AM band VCO frequency input pin		
75	EO		Error output pin in PLL part (charge pump output) If tuning freq. = VCO freq., EO pin is floating. If tuning freq. > VCO freq., EO pin is high. If tuning freq. = VCO freq., EO pin is low.		
76	RESET		Chip reset pin. Reset is active high.		
77	Xin		Crystal oscillator input pin for main system clock		
78	Xout		Main system clock output pin		
79	Xtin		Crystal oscillator input pin for Sub system clock		
80	Xtout		Sub system clock output pin		

1.5 Pin Diagram

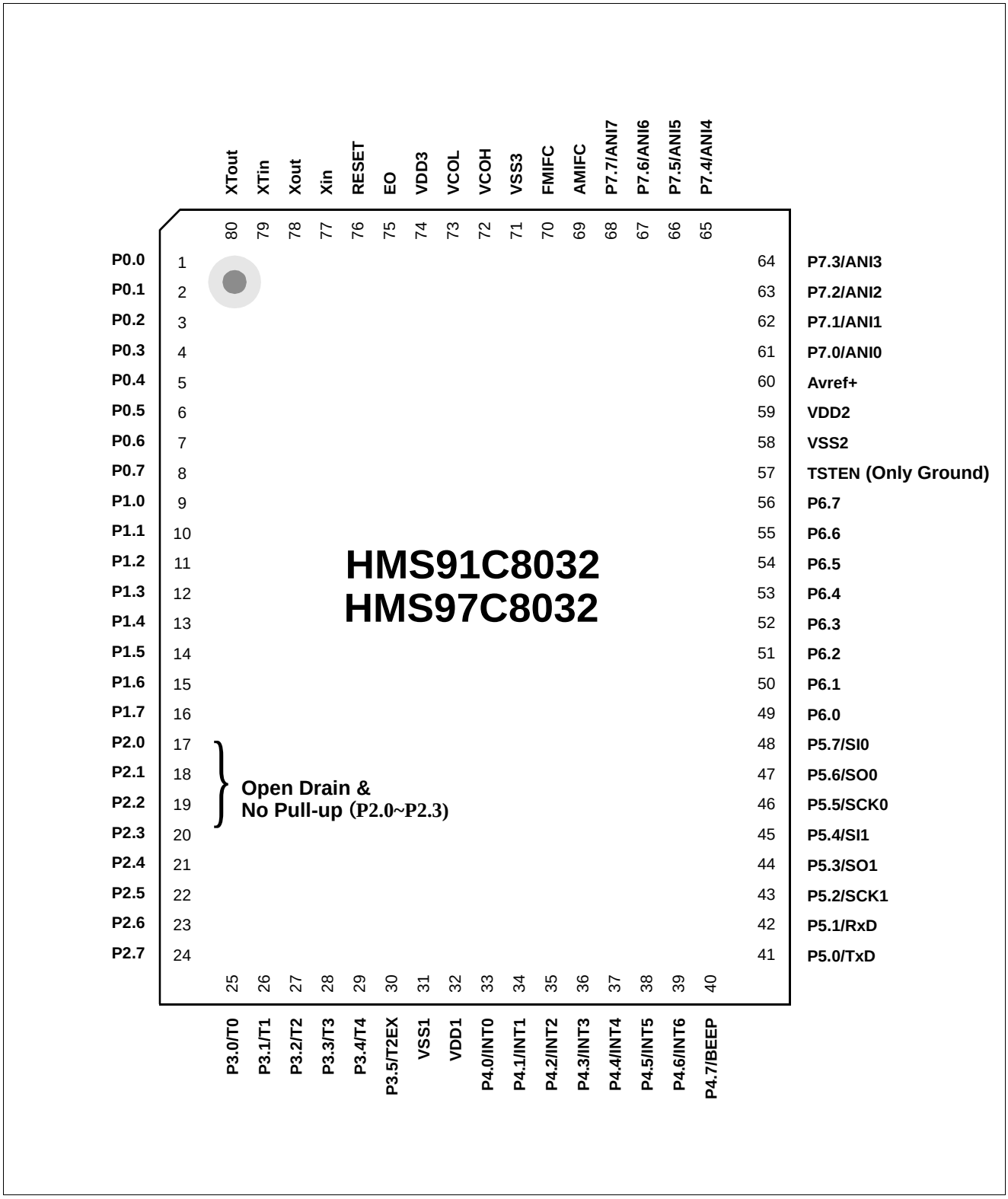


Figure 1-1 HMS9XC8032 Pin Diagram

2. MEMORY ORGANIZATION

All HMS91C8032 devices have separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU.
Program memory (ROM) can only be read, not written to. There can be up to 32K bytes of program memory. In the HMS9XC8032 devices, the Program Memory is provided on-chip.
Data Memory (RAM) occupies a separate address space from Program Memory. In the HMS9XC8032, the data memory is on-chip.

2.1 Program Memory

Figure 2-1 shows a map of the lower part of the Program Memory. After reset, the CPU begins execution from location 0000H.

As shown in Figure 2-2, each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory.
The interrupt service locations are spaced at 8-byte intervals : 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1 and etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.
Program Memory addresses are always 16bits wide, even though the actual amount of Program Memory used may be less than 32K bytes.

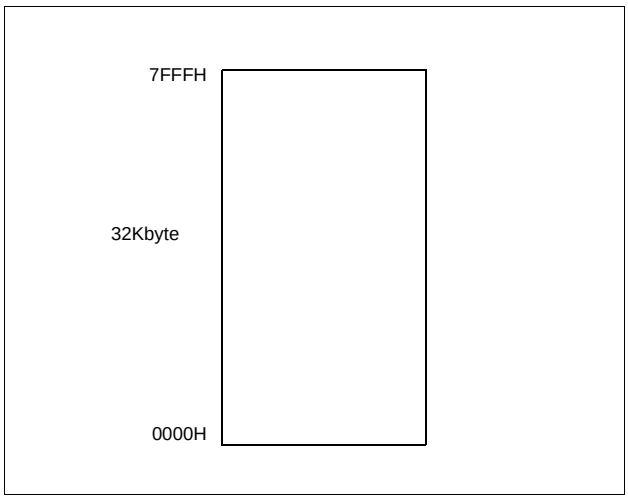


Figure 2-1 Program Mamory

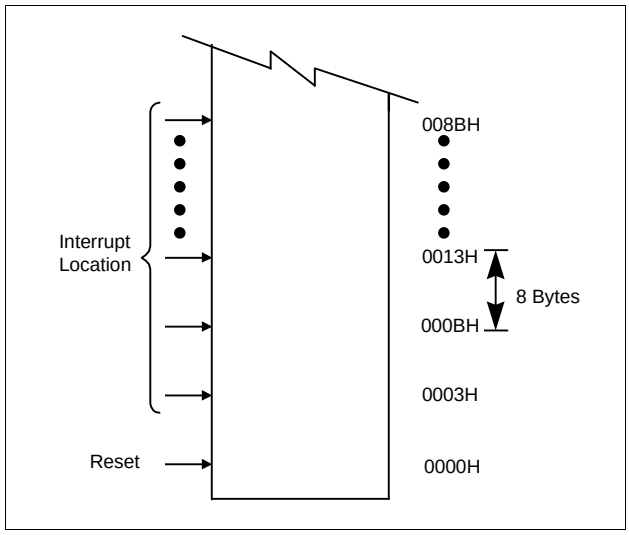


Figure 2-2 Interrupt Location of Program Memory

2.2 Data Memory

Figure 2-3, Figure 2-6 and Figure 2-6 shows the Memory spaces available to the HMS9XC8032 user. HMS9XC8032 can address up to 1kbytes of data memory. 10bits address is configured as follows.

10bits address for READ memory operation = 2bits of RDPG + 8bits of implied address in instruction

10bits address for WRITE memory operation = 2bits of WRPG + 8bits of implied address in instruction

(Where, 0 =< RDPG, WRPG =< 6)

(CAUTIONS: A valid value which can be stored in RDPG and WRPG must be from 0 to 6. 7 is reserved for indirect addressable memory region.(upper 128bytes region) A programmer who set RDPG/WRPG to 7 or greater than 7 will get the invalid memory operation results.)

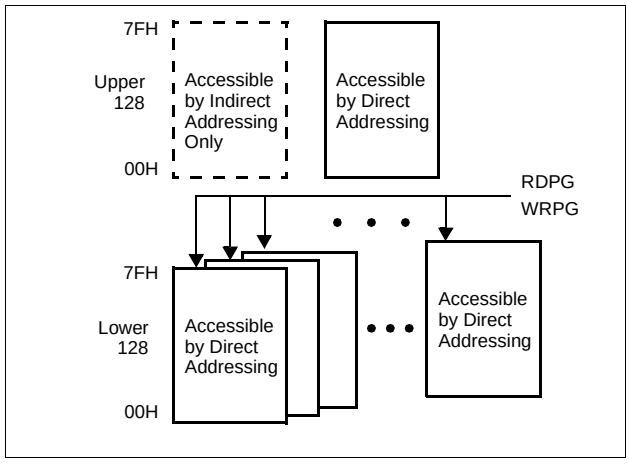


Figure 2-3 Data Memory Structure

Data memory consists of 7 pages, and each page can store 128bytes. According to the value of RDPG(FCH) and WRPG(FDH), HMS9XC8032 selects working memory page. Figure 2-4 shows the generation method of internal data memory address. For example, to read from data memory, HMS9XC8032 references the content of RDPG, generates 10bits address and accesses the corresponding data. The following two cases are equivalent.

```
MOV   00H, A     1)
MOV   R0, A      2)
```

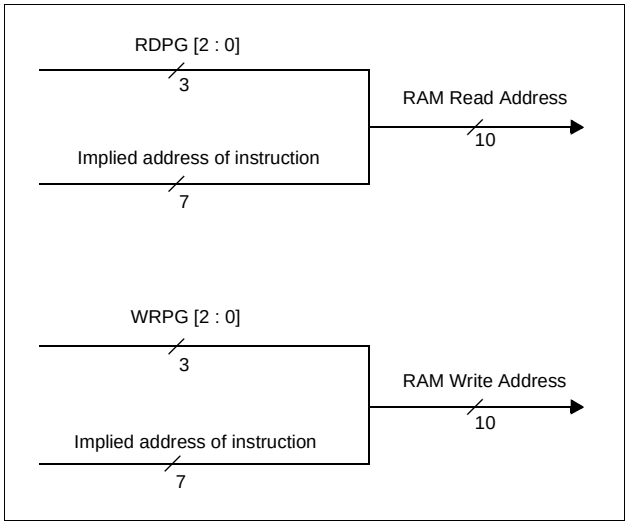


Figure 2-4 Data Memory Address Generation Method

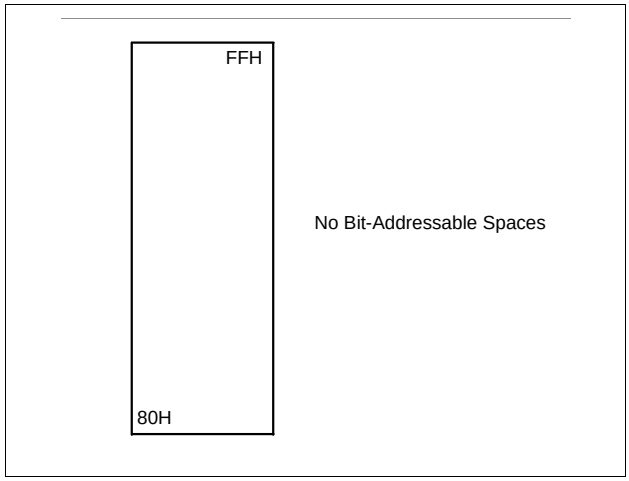


Figure 2-5 Upper 128bytes of Internal RAM

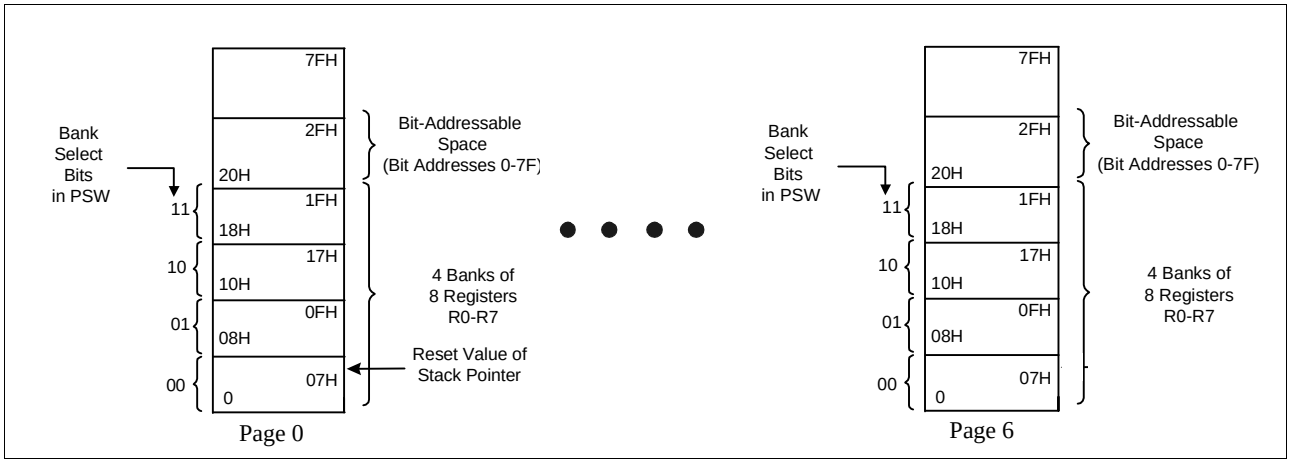


Figure 2-6 Page0 ~ Page6 of Internal RAM

2.3 Special Function Register

Unlike Intel 805X series, HMS9XC8032 has two SFR pages. If the content of SFRPG (address:FFH) is clear to 00H(01H), HMS9XC8032 assumes working SFR page to SFR page 0(1). Byte-addressing only registers in SFR pages have the same address in each SFR pages, but bit addressing registers in SFR page 0 and SFR page 1 are different except ACC, B and PSW.

The Port Data registers are located to SFR page1, and the Peripheral Control registers to SFR page0. Refer to "4.2 Special Function Registers" on page 19.

HMS91C8032/HMS97C8032 Description

3. INSTRUCTION SET

The HMS9XC8032 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

3.1 Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Figure 3-1, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the functions of a Carry bit in arithmetic operations, also serves as the “Accumulator” for a number of Boolean operations.

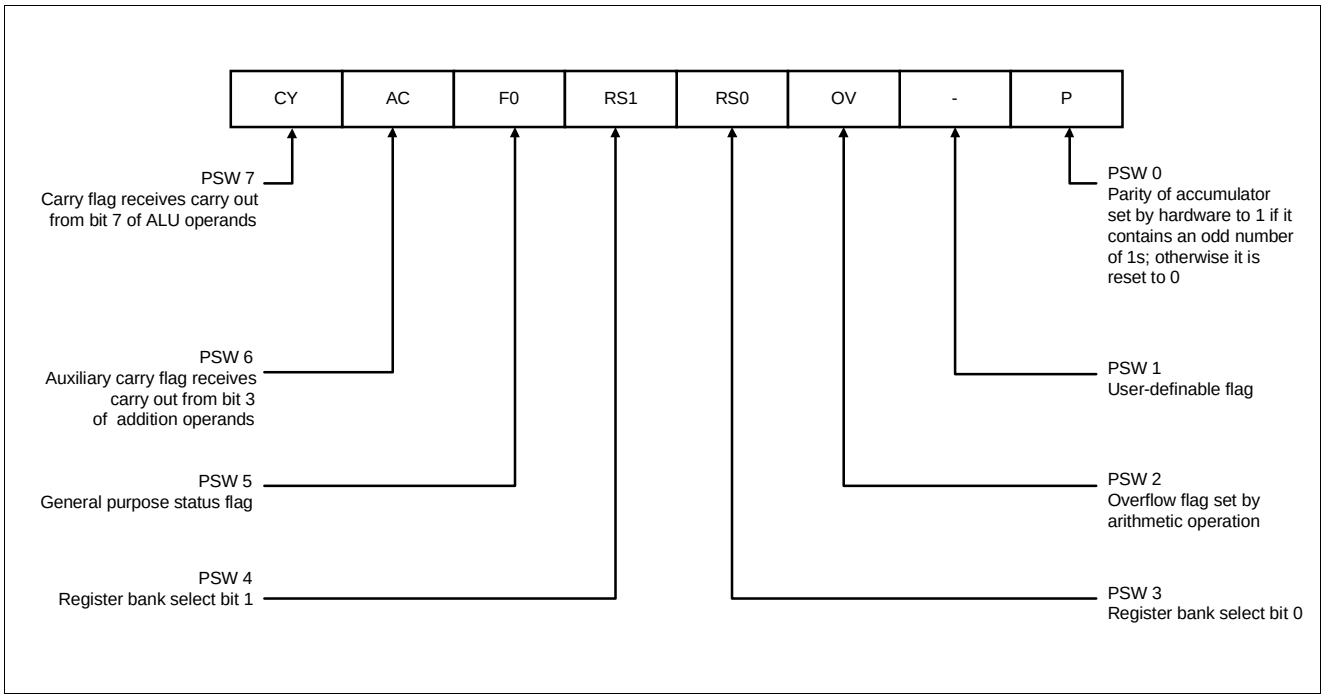


Figure 3-1 PSW (Program Status Word) Register in HMS9XC8032 Devices

RS0 and RS1 are used to select one of the four register banks. Each register bank composed of eight registers.(R0 to R7) The selection of a register bank is made at execution time.

The parity bit reflects the number of 1s in the Accumulator: P= 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even. Two bits in the PSW are uncommitted and may be used as general-purpose status flags.

3.2 Addressing Modes

The addressing modes in the HMS9XC8032 instruction set are as follows:

Direct Addressing

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit "data pointer" register, DPTR.

Register Instructions

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

Register-Specific Instructions

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

Immediate Constants

The value of a constant can follow the opcode in Program Memory. For example,

```
MOV   A,   #100
```

loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

Indexed Addressing

Only Program Memory can be accessed with indexed addressing, and it can be read. This addressing mode is intended for reading look-up tables in Program Memory. A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number.

The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the "case jump" instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

3.3 Arithmetic Instructions

The arithmetic instructions is listed in Table 3-1. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A, <byte> instruction can be written as:

```
ADD     a, 7FH (direct addressing)
ADD     A, @R0   (indirect addressing)
ADD     a, R7    (register addressing)
ADD     A, #127  (immediate constant)
```

Note that any byte in the internal Data Memory space can be incremented without going through the Accumulator.

One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operations is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register.

MNEMONIC	OPERATION	ADDRESSING MODES			
		Dir	Ind	Reg	Imm
ADD A,<byte>	A = A+<byte>	X	X	X	X
ADDC A,<byte>	A = A+<byte>+C	X	X	X	X
SUBB A,<byte>	A = A-<byte>-C	X	X	X	X
IN C	A = A+1	Accumulator only			
INC <byte>	<byte> = <byte>+1	X	X	X	
INC DPTR	DPTR = DPTR+1	Data Pointer only			
DEC A	A = A-1	Accumulator only			
DEC <byte>	<byte> = <byte>-1	X	X	X	
MUL AB	B:A = B x A	ACC and B only			
DIV AB	A = Int[A/B] B = Mod[A/B]	ACC and B only			
DA A	Decimal Adjust	Accumulator only			

Table 3-1 HMS9XC8032 Arithmetic Instructions

Oddly enough, DIV AB finds less use in arithmetic "divide" routines than in radix conversions and programmable shift operations. An example of the use of DIV AB in a radix conversion will be given later. In shift operations, dividing a number by 2n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4μs and leaves the B register holding the bits that were shifted out. The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

3.4 Logical Instructions

Table 3-2 shows list of HMS9XC8032 logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and byte contains 01010011B, then :

ANL A, <byte>

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 3-2.

The ANL A, <byte> instruction may take any of the forms:

ANL A,7FH(direct addressing)
ANL A, @R1 (indirect addressing)
ANL A,R6 (register addressing)
ANL A,#53H (immediate constant)

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in

XRL P1, #0FFH.

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to push it onto the stack in the service routine.

The Rotate instructions (RL A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations. For example, if the Accumulator contains a binary number which is known to be less than 100, it can be quickly converted to BCD by the following code:

MNEMONIC	OPERATION	ADDRESSING MODES			
		Dir	Ind	Reg	Imm
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X
ANL <byte>,A	<byte> = <byte> .AND. A	X			
ANL <byte>,#data	<byte> = <byte> .AND. #data	X			
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X
ORL <byte>,A	<byte> = <byte> .OR. A	X			
ORL <byte>,#data	<byte> = <byte> .OR. #data	X			
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X
XRL <byte>,A	<byte> = <byte> .XOR. A	X			
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X			
CRL A	A = 00H	Accumulator only			
CPL A	A = .NOT. A	Accumulator only			
RL A	Rotate ACC Left 1 bit	Accumulator only			
RLC A	Rotate Left through Carry	Accumulator only			
RR A	Rotate ACC Right 1 bit	Accumulator only			
RRC A	Rotate Right through Carry	Accumulator only			
SWAP A	Swap Nibbles in A	Accumulator only			

Table 3-2 HMS9XC8032 Logical Instructions

```
MOVE    B,#10
DIV     AB
SWAP    A
ADD     A,B
```

Dividing the number by 10 leaves the tens digit in the low nibble of the Accumulator, and the ones digit in the B register. The SWAP and ADD instructions move the tens digit to the high nibble of the Accumulator, and the ones digit to the low nibble.

3.5 Data Transfers

Internal RAM

Table 3-3 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember, the Upper 128 bytes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in HMS9XC8032 devices, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128 bytes of RAM, if they are implemented, but not into SFR space.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A, @Ri instruction is similar, but only the low nibbles are involved in the exchange.

To see how XCH and XCHD can be used to facilitate data manipulations, consider first the problem of shifting and 8-digit BCD number two digits to the right. Figure 3-2 shows how this can be done using XCH instructions. To aid in understanding how the code works, the contents of the registers that are holding the BCD number and the content of the Accumulator are shown alongside each instruction to indicate their status after the instruction has been executed.

After the routine has been executed, the Accumulator contains the two digits that were shifted out on the right. Doing the routine with direct MOVs uses 14 code bytes. The same operation with XCHs uses only 9 bytes and executes almost twice as fast.

To right-shift by an odd number of digits, a one-digit must be executed.

Figure 3-3 shows a sample of code that will right-shift a BCD number one digit, using the XCHD instruction. Again, the contents of the registers holding the number and of the accumulator are shown alongside each instruction.

MNEMONIC	OPERATION	ADDRESSING MODES			
		Dir	Ind	Reg	Imm
MOV A,<src>	A = <src>	X	X	X	X
MOV <dest>,A	<dest> = A	X	X	X	
MOV <dest>,<src>	<dest> = <src>	X	X	X	X
MOV DPTR,#data16	DPTR = 16-bit immediate constant				X
PUSH <src>	INC SP:MOV “@SP”, <src>	X			
POP <dest>	MOV <dest>, “@SP”:DEC SP	X			
XCH A,<byte>	ACC and <byte> exchange data	X	X	X	
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X		

Table 3-3 Data Transfer Instruction that Access Internal Data Memory Space

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

A. Using direct MOVs: 14 bytes, 9us

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A,2EH	00	00	12	34	56	78

B. Using XCHs: 9 bytes, 5us

Figure 3-2 Shifting a BCD Number Two Digits to the Right

		2A	2B	2C	2D	2E	ACC
	MOV R1,#2EH	00	12	34	56	78	XX
	MOV R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH						
LOOP:	MOV A,@R1	00	12	34	56	78	78
	XCHD A,@R0	00	12	34	58	78	76
	SWAP A	00	12	34	58	78	67
	MOV @R1,A	00	12	34	58	67	67
	DEC R1	00	12	34	58	67	67
	DEC R0	00	12	34	58	67	67
	CJNE R1,#2AH,LOOP						
	loop for R1 = 2DH:	00	12	38	45	67	45
	loop for R1 = 2CH:	00	18	23	45	67	23
	loop for R1 = 2BH:	08	01	23	45	67	01
	CLR A	00	01	23	45	67	00
	XCH A,2AH	08	01	23	45	67	08

Figure 3-3 Shifting a BCD Number One Digits to the Right

First, pointers R1 and R0 are set up to point to the two bytes containing the last four BCD digits. Then a loop is executed which leaves the last byte, location 2EH, holding the last two digits of the shifted number. The pointers are decremented, and the loop is repeated for location 2DH. The CJNE instruction (Compare and Jump if Not equal) is a loop control that will be described later.

The loop executed from LOOP to CJNE for R1 = 2EH, 2DH, 2CH, and 2BH. At that point the digit that was originally shifted out on the right has propagated to location 2AH. Since that location should be left with 0s, the lost digit is moved to the Accumulator.

External RAM

HMC9XC8032 series do NOT support external RAM access mode.

3.6 Lookup Tables

Table 3-4 shows the two instructions that are available for reading lookup tables in Program Memory. Since these instructions access only Program Memory, the lookup tables can only be read, not updated.

The mnemonic is MOVC for "move constant." The first MOVC instruction in Table 3-1 can accommodate a table of up to 256 entries numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to the beginning of the table. Then:

```
MOVC    A, @A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the table base, and the table is accessed through a subroutine. First the number of the desired entry is loaded into the Accumulator, and the subroutine is called:

```
MOV     A , ENTRY NUMBER
CALL    TABLE
```

The subroutine "TABLE" would look like this:

```
TABLE:  MOVC    A , @A+PC
        RET
```

The table itself immediately follows the RET (return) instruction is Program Memory. This type of table can have up to 255 entries, numbered 1 through 255. Number 0 cannot be used, because at the time the MOVC instruction is executed, the PC contains the address of the RET instruction. An entry numbered 0 would be the RET opcode itself.

MNEMONIC	OPERATION
MOVC A, @A+DPTR	Read program memory at (A + DPTR)
MOVC A, @A+PC	Read program memory at (A + PC)

Table 3-4 Table B-4 HMS9XC8032 Data Transfer Instruction that Access Internal Data Memory Spcace

3.7 Boolean Instructions

HMS9XC8032 devices contain a complete Boolean (single-bit) processor. One page of the internal RAM contains 128 addressable bits, and the SFR space can support up to 128 addressable bits as well. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software.

The instruction set for the Boolean processor is shown in Table 3-5. All bits accesses are by direct addressing.

Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

Note how easily an internal flag can be moved to a port pin:

```
MOV     C,FLAG
MOV     P1.0,C
```

In this example, FLAG is the name of any addressable bit in the Lower 128 or SFR space. An I/O line (the LSB of Port 1, in this case) is set or cleared depending on whether the flag bit is 1 or 0.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C, etc.). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

MNEMONIC	OPERATION
ANL C,bit	C = A .AND. bit
ANL C,/bit	C = C .AND..NOT. bit
ORL C,bit	C = A .OR. bit
ORL C,/bit	C = C .OR..NOT. bit
MOV C,bit	C = bit
MOV bit,C	bit = C
CLR C	C = 0
CLR bit	bit = 0
SETB C	C = 1
SETB bit	bit = 1
CPL C	C = .NOT.C
CPL bit	bit = .NOT.bit
JC rel	Jump if C = 1
JNC rel	Jump if C = 0
JB bit,rel	Jump if bit = 1
JNB bit,rel	Jump if bit = 0
JBC bit,REL	Jump if bit = 1;CLR bit

Table 3-5 Table B-5 HMS9XC8032 Boolean Instructions

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation. An XRL operation is simple to implement in software. Suppose, for example, it is required to form the Exclusive OR of two bits:

```
C = bit 1  .XRL.  bit2
```

The software to do that could be as follows:

```
        MOV     C , bit1
        JNB     bit2, OVER
        CPL     C
OVER:   (continue)
```

First, bit1 is moved to the Carry. If bit2 = 0, then C now contains the correct result. That is, bit1 .XRL. bit2 = bit1 if bit2 = 0. On the other hand, if bit2 = 1, C now contains the complement of the correct result. It need only be inverted (CPL C) to complete the operation.

This code uses the JNB instruction, one of a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0, the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation. All the PSW bits are directly addressable, so the Parity bit, or the general-purpose flags, for example, are also available to the bit-test instructions.

3.8 Relative Offset

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program memory. However, the destination address assembles to a relative offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed.

The range of the jump is therefore -128 to +127 Program Memory bytes relative to the first byte following the instruction.

3.9 Jump Instructions

Table 3-6 shows the list of unconditional jumps.

MNEMONIC	OPERATION
JMP addr	Jump to addr
JMP @A+DPTR	Jump to A+DPTR
CALL addr	Call subroutine at addr
RET	Return from subroutine
RETI	Return from interrupt
NOP	No operation

Table 3-6 Unconditional Jumps in HMS9XC8032 Devices

The table lists a single "JMP add" instruction, but in fact there are three SJMP, LJMP, and AJMP, which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of -128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64K Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2K block as the instruction following the AJMP.

In all cases the programmer specifies the destination address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the destination address into the correct format for the given instruction. If the format required by the instruction will not support the distance to the specified destination address, a "Destination out of range" message is written into the List file.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically. DPTR is set up with the address of a jump table. In a 5-way branch, for example, an integer 0 through 4 is loaded into the Accumulator. The code to be executed might be as follows:

```
MOV     DPTR,#JUMP TABLE
MOV     A,INDEX_NUMBER
RL      A
JMP     @A+DPTR
```

The RL A instruction converts the index number (0 through 4) to an even number on the range 0 through 8, because each entry in the jump table is 2 bytes long:

```
JUMP TABLE:
AJMP    CASE 0
AJMP    CASE 1
AJMP    CASE 2
AJMP    CASE 3
AJMP    CASE 4
```

Table 3-1 shows a single "CALL addr" instruction, but there are two of them, LCALL and ACALL, which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64K Program Memory space. The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2K block as the instruction following the ACALL.

In any case, the programmer specifies the subroutine address to the assembler in the same way: as a label or as a 16-bit constant. The assembler will put the address into the correct format for the given instructions.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 3-7 shows the list of conditional jumps available to the HMS9XC8032 user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of -128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control. To execute a loop N times, load a counter byte with N and terminate the loop with a DJNZ to the beginning of the loop, as shown below for N = 10:

```
MOV     COUNTER,#10
LOOP:(begin loop)
•
•
•
(end loop)
DJNZ    COUNTER, LOOP
(continue)
.
```

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control as in Figure 12. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. In the example of Figure B-3 Shifting a BCD Number One Digits to the Right, the two bytes were data in R1 and the constant 2AH. The initial data in R1 was 2EH.

MNEMONIC	OPERATION	ADDRESSING MODES			
		DIR	IND	REG	IMM
JZ rel	Jump if A = 0	Accumulator only			
JNZ rel	Jump if A ≠ 0	Accumulator only			
DJNZ <byte>,rel	Decrement and jump if not Zero	X		X	
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X	

Table 3-7 Conditional Jumps in HMS9XC8032 Devices

Every time the loop was executed, R1 was decremented, and the looping was to continue until the R1 data reached 2AH.

Another application of this instruction is in "greater than, less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared

3.10 CPU Timing

All HMS9XC8032 microcontrollers have an on-chip oscillator which can be used if desired as the clock source for the CPU. To use the on-chip oscillator, connect a crystal or ceramic resonator between the Xout(XTout) and Xin(XTin) pins of the microcontroller, and capacitors to ground as shown in Figure 3-4 Using the On-Chip Oscillator.

Examples of how to drive the clock with an external oscillator are shown in Figure 3-5. In the CMOS devices (HMS9XC8032, etc.), the signal at the Xout(XTout) pin drives the internal clock generator. The internal clock generator defines the sequence of states that make up the HMS9XC8032 machine cycle.

Main Clock

Xin, Xout : 7.2 MHz

Sub Clock

XTin, XTout : 32.768 KHz

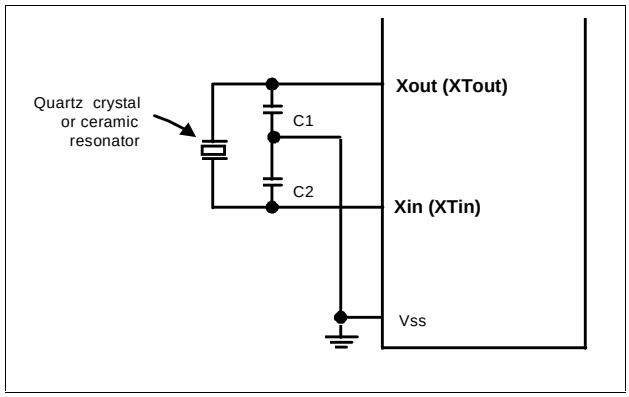


Figure 3-4 sing the On-Chip Oscillator

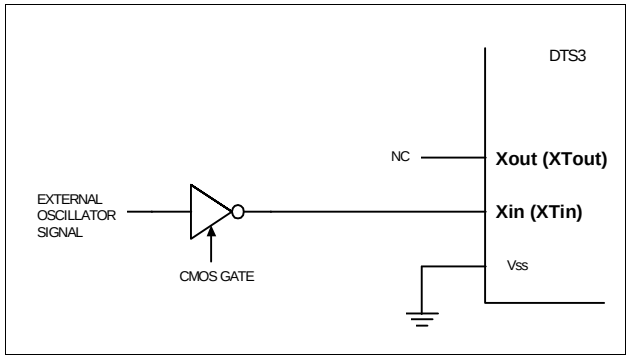


Figure 3-5 Using an External Clock

3.11 Machine Cycles

A machine cycle consists of a sequence of 6 states, numbered S1 through S6. One machine cycle period vary according to the SC-MOD register value. Refer to Figure 3-6

Each state is divided into a Phase 1 half and a Phase 2 half. State Sequence in HMS9XC8032 Devices shows that fetch/execute sequences in states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn't require it. If the instruction being executed doesn't need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 3-6) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

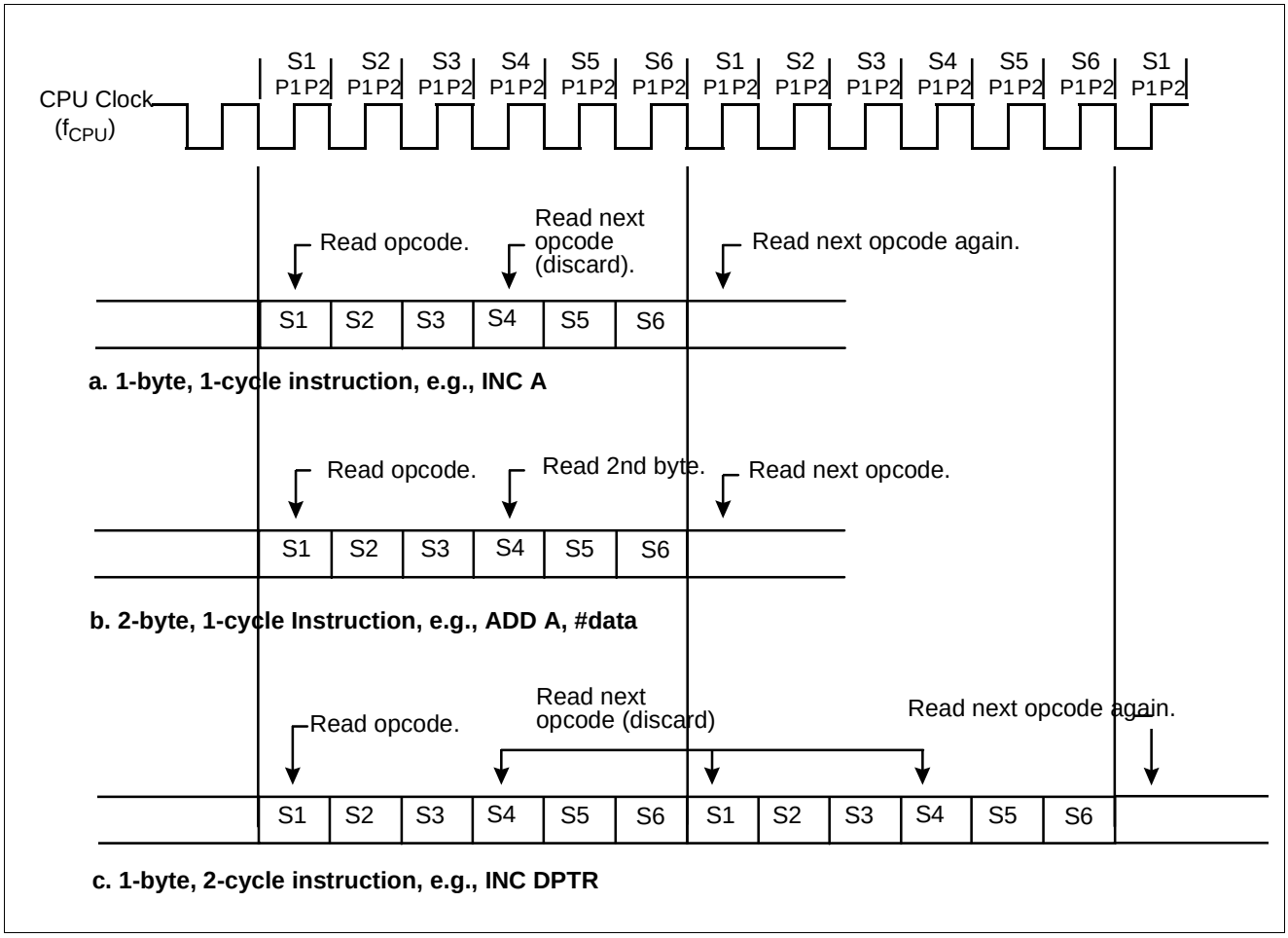


Figure 3-6 State Sequence in HMS9XC8032 Devices

4. HARDWARE DESCRIPTION

This chapter provides a detailed description of the HMS9XC8032 microcontroller (see Figure 4-1) included in this description are the:

- **Clock Genernation Block**
- **Special Function Registers**
- **Timers/Counters**
- **Serial Interface (UART)**
- **Standard Serial Interface (SI01, SIO2)**
- **Port Structure**
- **Watch Dog Timer**
- **Buzzer**
- **IF Counter**
- **PLL**
- **ADC**
- **Interrupts**
- **Reset**
- **Power-On Reset**
- **Power-Saving Modes**
- **On-Chip Oscillators**

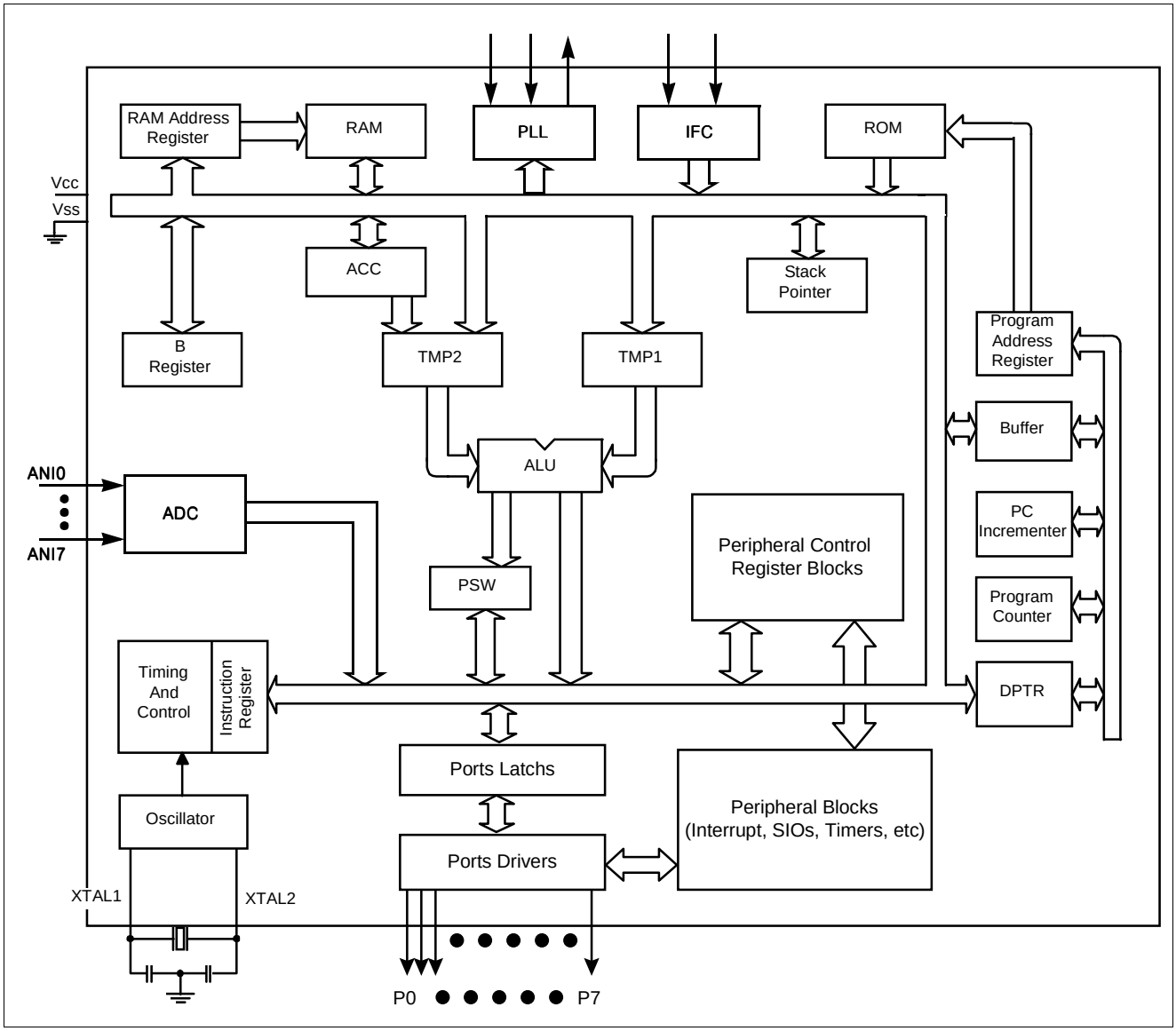


Figure 4-1 HMS9XC8032 Architecture

4.1 Clock Generation Block

Software can control the system clock speed of HMS91C8032 with the SCMOD register. the SCMOD register determine system clock speed and clock source. Figure 4-3 shows the block diagram of the system clock generation block.

Guideline on the CPU clock speed

For determining the speed of CPU clock(f_{CPU}), the following constraints should be satisfied.

The maximum counting rate of timer0~4 in counter mode, should be less than or equal to (1/6)f_{CPU}

The maximum timer clock rate of timer0~4 in timer mode should be less than or equal to (1/2)f_{CPU}

NOTE:

SCMOD[2:0]			Select system clock
0	x	x	fxx
1	0	0	fxx / 2
1	0	1	fxx / 4
1	1	0	fxx / 8
1	1	1	fxx / 16

SCMOD: SELECT CLOCK MODE. : 80H

-	-	-	SCSTOP	SCSW	SCMOD2	SCMOD1	SCMOD0

-	SCMOD.7	Reserved for future use *
-	SCMOD.6	Reserved for future use *
-	SCMOD.5	Reserved for future use *
SCSTOP	SCMOD.4	Software control of the main system oscillator. A logic 1 pulls down the main system oscillator (7.2MHz).
SCSW	SCMOD.3	Software switch control between main system oscillator and sub system oscillator. A logic 1 switches sub system oscillator (32.768KHz).
SCMOD2	SCMOD.2	See NOTES
SCMOD1	SCMOD.1	See NOTES
SCMOD0	SCMOD.0	See NOTES

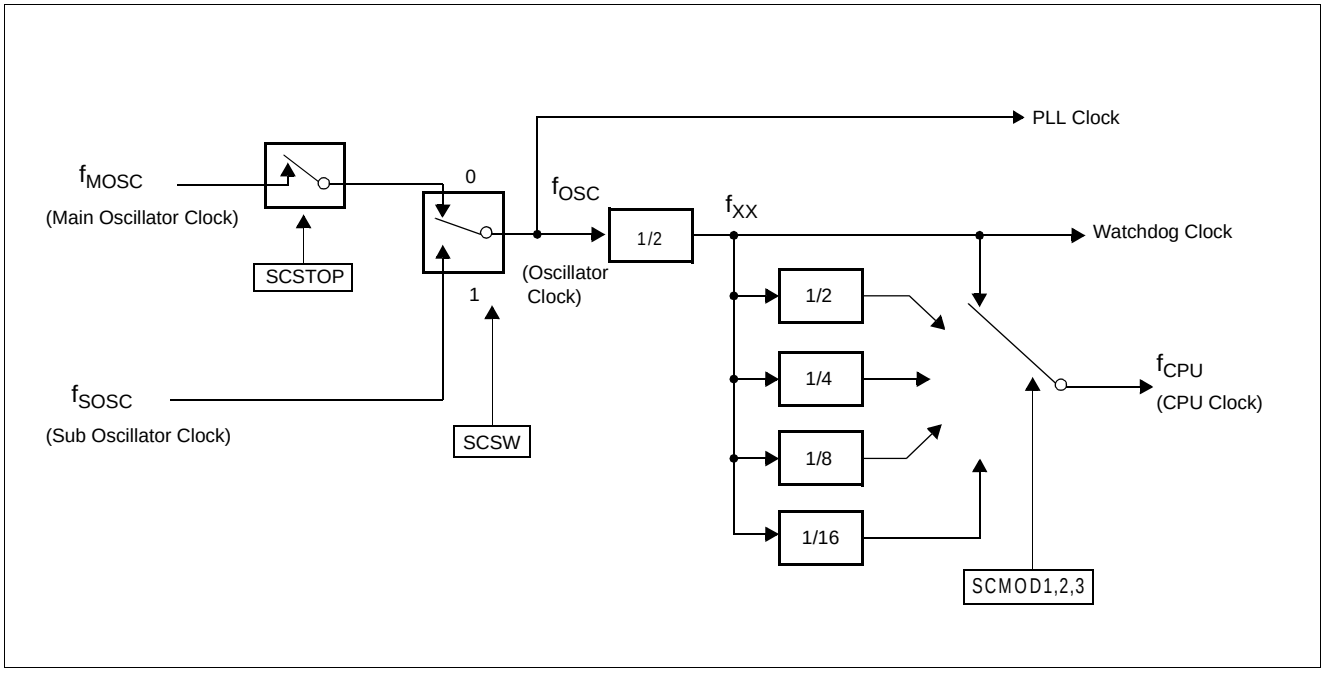


Figure 4-2 System Clock Generation Block

4.2 Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 4-1 and Table 4-2. Note that in the SFRs not all of the addresses are occupied. Unoccupied addresses are not implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have no effect.

User software should not write 1s to these unimplemented locations, since they may be used in other HMS9XC8032 Family products to invoke new features. In that case the reset or inactive values of the new bits will always be 0, and their active values will be 1.

F8	WDTCON	WDTDR			RDPG	WRPG		SFRPG	FF
F0	B	PLLMOD	PLLDRH	PLLDRL	IFCMOD	IFCDR2	IFCDR1	IFCDR0	F7
E8	IR3								EF
E0	ACC								E7
D8	IR2	IT2							DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0	IE3	IP3							C7
B8	IP				P4MOD	P5MOD	P6MOD	P7MOD	BF
B0	IE2	IP2			P0MOD	P1MOD	P2MOD	P3MOD	B7
A8	IE				P4CON	P5CON	P6CON	P7CON	AF
A0	S12CON	SBUF1	SBUF2		P0CON	P1CON	P2CON	P3CON	A7
98	SCON	SBUF							9F
90	T34CON	T34MOD	TL3	TL4	TH3	TH4			97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	SCMOD	SP	DPL	DPH	ADCCON	ADCDR	PLLDEBUG	PCON	87

Bit Addressable

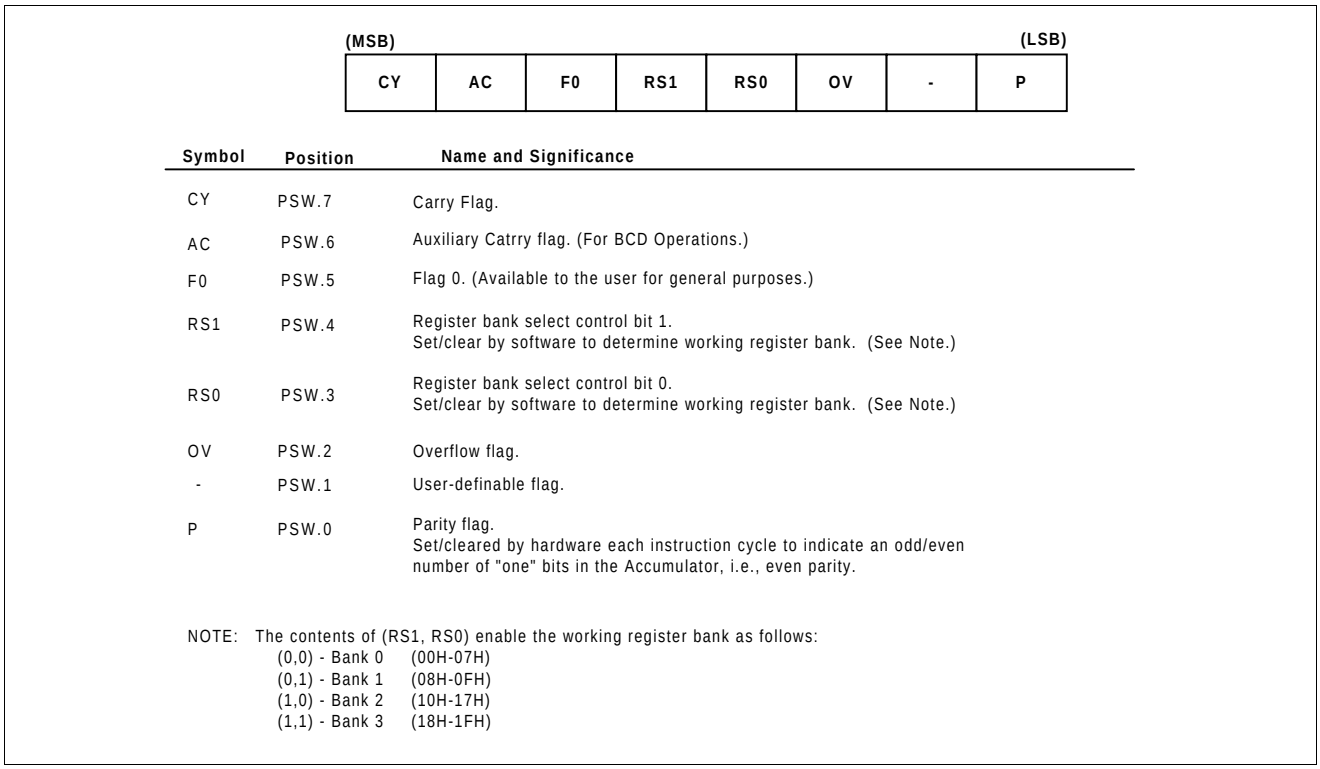
Table 4-1 SFRPG0 SFR Memory Map (8 Bytes)

F8	P7DATA
F0	B
E8	P6DATA
E0	ACC
D8	P5DATA
D0	PSW
C8	P4DATA
C0	
B8	P3DATA
B0	
A8	P2DATA
A0	
98	P1DATA
90	
88	P0DATA
80	

: in this area, the registers of SFRPG0 are the same registers of SFRPG1

: in this area, the registers of SFRPG0 are different from registers of SFRPG1

Table 4-2 SFRPG1 SFR Memory Map (8 Bytes)



(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	-	P

Symbol	Position	Name and Significance
CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Catrry flag. (For BCD Operations.)
F0	PSW.5	Flag 0. (Available to the user for general purposes.)
RS1	PSW.4	Register bank select control bit 1. Set/clear by software to determine working register bank. (See Note.)
RS0	PSW.3	Register bank select control bit 0. Set/clear by software to determine working register bank. (See Note.)
OV	PSW.2	Overflow flag.
-	PSW.1	User-definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the Accumulator, i.e., even parity.

NOTE: The contents of (RS1, RS0) enable the working register bank as follows:
(0,0) - Bank 0 (00H-07H)
(0,1) - Bank 1 (08H-0FH)
(1,0) - Bank 2 (10H-17H)
(1,1) - Bank 3 (18H-1FH)

Figure 4-3 Program Status Word (PSW) Register

Accumulator

ACC is the Accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B Register

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

Program Status Word

The PSW register contains program status information as detailed in Figure 4-3.

Stack Pointer

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at locations 08H. **But, it is forbidden to use the area of 00H to 7FH as the Stack. Thus the stack pointer should be set to the address larger than 7FH when it is initialized.**

Data Pointer

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Serial Data Buffer

SBUF, SBUF1 and SBUF2 are Serial Buffers. SBUF register is used by UART, SBUF1 used by SIO1 and SBUF2 used by SIO2.

The SBUF is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

Unlike SBUF, SBUF1(SBUF2) is one register. If the SIO1(SIO2) run flag is activated, receive and transmit of serial data is done simultaneously using SBUF1(SBUF2).

Timer Registers Basic to HMS9XC8032

Register pairs (THx, TLx) are the 16-bit Counting registers for Timer/Counters 0, 1, 2, 3 and 4, respectively.

Control Register for the HMS9XC8032

Special Function Registers IPx, IEx, TMOD, T34MOD, TCON, T2CON, SCON, S12CON, PCON and etc. contain control and status bits for the various peripherals in HMS9XC8032. They are described in later sections.

Summary of SFR

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE. : D0H

CY	AC	F0	RS1	RS0	OV	-	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (See NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (See NOTE 1).
OV	PSW.2	Overflow Flag.
-	PSW.1	User flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in th accumulator.

NOTE 1: *The value presented by RS0 and RS1 selects the corresponding register bank.*

RS1	RS0	Register Bank	Addresss
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

PCON: POWER CONTROL REGISTER. NOT BIT ADDRESSABLE. : 87H

SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD	PCON.7	Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.
-	PCON.6	Not implemented, reserved for future use.*
-	PCON.5	Not implemented, reserved for future use.*
-	PCON.4	Not implemented, reserved for future use.*
GF1	PCON.3	General purpose flag bit.
GF0	PCON.2	General purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates Power Down operation.
IDL	PCON.0	dle Mode bit. Setting this bit activates Idle Mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

INTERRUPTS:

In order to use any of the interrupt in the DTS3, the following three steps must be taken.

1. Set the EA (Enable All) bit in the IE Register to 1.
2. Set the corresponding individual interrupt enable bit in the IE, IE2 and IE3 register to 1.
3. Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.

Interrupt Source	Vector Address
INTEX0	0003H
INTT0	000BH
INTEX1	0013H
INTT1	001BH
INTS0 (RI & TI)	0023H
INTT2 (TF2 & EXF2)	002BH
INTWDT	0033H
INTIFC	003BH
INTAD	0043H
INTEX2	004BH
INTEX3	0053H
INTEX4	005BH
INTS1	0063H
INTS2	006BH
INTEX5	0073H
INTEX6	007BH
INTT3	0083H
INTT4	008BH

Table 4-3 Intrrupt Vector

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE. : A8H

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	-	IET2	IES0	IET1	IEX1	IET0	IEX0

EA	IE.7	Disables all interrupt. If EA = 0. no interrupt will be acknowledged. IF EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use.*
IET2	IE.5	Enable or disable the Timer 2 overflow or capture interrupt
IES0	IE.4	Enable or disable the serial port interrupt.
IET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
IEX1	IE.2	Enable or disable External Interrupt 1
IET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
IEX0	IE.0	Enable or disable External Interrupt 0.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IE2: INTERRUPT ENABLE REGISTER 2. BIT ADDRESSABLE. : B0H

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

-	-	-	IEX6	IEX5	IEX4	IEX3	IEX2

-	IE2.7	Not implemented, reserved for future use.*
-	IE2.6	Not implemented, reserved for future use.*
-	IE2.5	Not implemented, reserved for future use.*
IEX6	IE2.4	Enable or disable External Interrupt 6
IEX5	IE2.3	Enable or disable External Interrupt 5
IEX4	IE2.2	Enable or disable External Interrupt 4
IEX3	IE2.1	Enable or disable External Interrupt 3
IEX2	IE2.0	Enable or disable External Interrupt 2.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IE3: INTERRUPT ENABLE REGISTER 3. BIT ADDRESSABLE. : C0H

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

-	IEWDT	IEADC	IEIF	IES2	IES1	IET4	IET3

-	IE3.7	Not implemented, reserved for future use.*
IEWDT	IE3.5	Enable or disable Watchdog timer interrupt
IEADC	IE3.6	Enable or disable A/D conversion completion interrupt
IEIF	IE3.4	Enable or disable IF counter interrupt
IES2	IE3.3	Enable or disable SIO2 interrupt
IES1	IE3.2	Enable or disable SIO1 Interrupt
IET4	IE3.1	Enable or disable the Timer 4 overflow interrupt.
IET3	IE3.0	Enable or disable the Timer 3 overflow interrupt.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP0, IP1 and IP2 register must be set to 1.

Remember that while an interrupt service is progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

INTEX0
INTT0
INTEX1
INTT1

INTS0 (RI or TI)
INTT2 (TF2 or EXF2)
INTWDT
INTIFC
INTAD
INTEX2
INTEX3
INTEX4
INTS1
INTS2
INTEX5
INTEX6
INTT3
INTT4

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE. : B8H

If the bit is 0, the corresponding interrupt has a lower priority and If the bit is 1, the corresponding interrupt has a higher priority.

-	-	IPT2	IPS0	IPT1	IPX1	IPT0	IPX0

-	IP.7	Not implemented, reserved for future use.*
-	IP.6	Not implemented, reserved for future use.*
IPT2	IP.5	Defines the Timer 2 interrupt priority level
IPS	IP.4	Defines the Serial Port interrupt priority level.
IPT1	IP.3	Defines the Timer 1 interrupt priority level.
IPX1	IP.2	Defines External Interrupt 1 priority level.
IPT0	IP.1	Defines the Timer 0 interrupt priority level.
IPX0	IP.0	Defines the External Interrupt 0 priority level.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IP2: INTERRUPT PRIORITY REGISTER 2. : B1H

If the bit is 0, the corresponding interrupt has a lower priority and If the bit is 1, the corresponding interrupt has a higher priority.

-	-	-	IPX6	IPX5	IPX4	IPX3	IPX2

-	IP2.7	Not implemented, reserved for future use.*
-	IP2.6	Not implemented, reserved for future use.*
-	IP2.5	Not implemented, reserved for future use.*
IPX6	IP2.4	Defines External Interrupt 6 priority level.
IPX5	IP2.3	Defines External Interrupt 5 priority level.
IPX4	IP2.2	Defines External Interrupt 4 priority level.
IPX3	IP2.1	Defines External Interrupt 3 priority level.
IPX2	IP2.0	Defines External Interrupt 2 priority level.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IP3: INTERRUPT PRIORITY REGISTER 3. : C1H

If the bit is 0, the corresponding interrupt has a lower priority and If the bit is 1, the corresponding interrupt has a higher priority.

-	IPWDT	IPADC	IPIFC	IPS2	IPS1	IPT4	IPT3

-	IP3.7	Not implemented, reserved for future use.*
IPWDT	IP3.6	Defines the Watchdog timer interrupt priority level.
IPADC	IP3.5	Defines ADC interrupt priority level.
IPIFC	IP3.4	Defines IF counter interrupt priority level.
IPS2	IP3.3	Defines SIO2 interrupt priority level.
IPS1	IP3.2	Defines SIO1 Interrupt priority level.
IPT4	IP3.1	Defines the Timer 4 interrupt priority level.
IPT3	IP3.0	Defines the Timer 3 interrupt priority level.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

REQUESTING TO SERVICE ONE OR MORE INTERRUPTS:

IR2: INTERRUPT REQUEST REGISTER 2. BIT ADDRESSABLE. : D8H

-	-	-	IRX6	IRX5	IRX4	IRX3	IRX2

-	IR2.7	Reserved for future use *
-	IR2.6	Reserved for future use *
-	IR2.5	Reserved for future use *
IRX6	IR2.4	External interrupt 6 flag. Set by hardware when External interrupt is detected. Cleared by hardware when interrupt is processed.
IRX5	IR2.3	External interrupt 5 flag. Set by hardware when External interrupt is detected. Cleared by hardware when interrupt is processed.
IRX4	IR2.2	External interrupt 4 flag. Set by hardware when External interrupt is detected. Cleared by hardware when interrupt is processed.
IRX3	IR2.1	External interrupt 3 flag. Set by hardware when External interrupt is detected. Cleared by hardware when interrupt is processed.
IRX2	IR2.0	External interrupt 2 flag. Set by hardware when External interrupt is detected. Cleared by hardware when interrupt is processed.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IR3: INTERRUPT REQUEST REGISTER 3. BIT ADDRESSABLE. : E8H

-	IRWDT	IRADC	IRIFC	IRS2	IRS1	IRT4	IRT3

-	IR3.7	Reserved for future use *
IRWDT	IR3.6	Watchdog timer overflow flag. Set by hardware when WDT overflows. Cleared by hardware as processor vectors to the interrupt service routine.
IRADC	IR3.5	A/D conversion completion flag. Set by hardware when ADC completes. Cleared by hardware as processor vectors to the interrupt service routine.
IRIFC	IR3.4	IF counter interrupt flag. Set by hardware when run time of IF counter reaches to gate time. Cleared by hardware as processor vectors to the interrupt service routine.
IRS2	IR3.3	SIO2 interrupt flag. Set by hardware when one TX/RX is completed. Cleared by hardware as processor

		vectors to the interrupt service routine.
IRS1	IR3.2	SIO1 interrupt flag. Set by hardware when one TX/RX is completed. Cleared by hardware when interrupt is processed.
IRT4	IR3.1	Timer 4 Overflow flag. Set by hardware when the Timer/Counter 4 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
IRT3	IR3.0	Timer 3 Overflow flag. Set by hardware when the Timer/Counter 3 overflows. Cleared by hardware as processor vectors to the interrupt service routine.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IT2: EXTERNAL INTERRUPT TPYE REGISTER 2. BIT ADDRESSABLE. : D9H

IT5M1	IT5M0	IT4M1	IT4M0	IT3M1	IT3M0	IT2M1	IT2M0

IT5M1	IT2.7	See Table 4-4
IT5M0	IT2.6	See Table 4-4
IT4M1	IT2.5	See Table 4-4
IT4M0	IT2.4	See Table 4-4
IT3M1	IT2.3	See Table 4-4
IT3M0	IT2.2	See Table 4-4
IT2M1	IT2.1	See Table 4-4
IT2M0	IT2.0	See Table 4-4

ITxM[1:0]		Select interrupt detect mode
0	0	Both rising & falling edge detection
0	1	Rising edge detect mode
1	0	Falling edge detect mode
1	1	Level (high) detect mode

Table 4-4 Interrupt Detect Mode

TCON: TIMER01/COUNTER01 CONTROL REGISTER. BIT ADDRESSABLE. : 88H

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 Overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON.5	Timer 0 Overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

T34CON: TIMER34/COUNTER34 CONTROL REGISTER. BIT ADDRESSABLE. : 90H

TF4	TR4	TF3	TR3		T3_SUB		T4_SUB

TF4	TCON.7	Timer 4 Overflow flag. Set by hardware when the Timer/Counter 4 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR4	TCON.6	Timer 4 run control bit. Set/cleared by software to turn Timer/Counter 4 ON/OFF.
TF3	TCON.5	Timer 3 Overflow flag. Set by hardware when the Timer/Counter 3 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR3	TCON.4	Timer 3 run control bit. Set/cleared by software to turn Timer/Counter 3 ON/OFF.
-	TCON.3	Reserved for future use *
T3_SUB	TCON.2	Switch main clock to sub clock for timer3 counting. This bit is a write-only register. 0 = Main Osc, 1 = Sub Osc.
-	TCON.1	Reserved for future use *
T4_SUB	TCON.0	Switch main clock to sub clock for timer4 counting. This bit is a write-only register. 0 = Main Osc, 1 = Sub Osc.

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE. : 89H

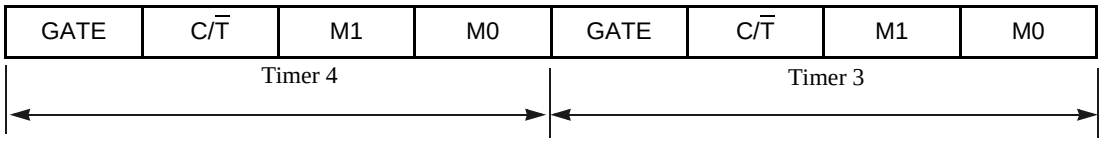
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer 1				Timer 0			

GATE	TMOD.7	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	TMOD.6	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	TMOD.5	Mode selector bit. (See Table 4-5)
M0	TMOD.4	Mode selector bit. (See Table 4-5)
GATE	TMOD.3	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	TMOD.2	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	TMOD.1	Mode selector bit. (See Table 4-5)
M0	TMOD.0	Mode selector bit. (See Table 4-5)

M1	M0	Mode	Operating Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

Table 4-5 Timer 0 and Timer 1 Mode

T34MOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE. : 91H



GATE	T34MOD.7	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	T34MOD.6	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	T34MOD.5	Mode selector bit. (See Table 4-6)
M0	T34MOD.4	Mode selector bit. (See Table 4-6)
GATE	T34MOD.3	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	T34MOD.2	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	T34MOD.1	Mode selector bit. (See Table 4-6)
M0	T34MOD.0	Mode selector bit. (See Table 4-6)

M1	M0	Mode	Operating Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 3) TL3 is an 8-bit Timer/Counter controlled by the standard Timer 3 control bits, TH3 is an 8-bit Timer and is controlled by Timer 4 control bits.
1	1	3	(Timer 4) Timer/Counter 4 stopped.

Table 4-6 Timer 3 and Timer 4 Mode

TIMER SET-UP

TIMER/COUNTER 0 (TIMER/COUNTER 3)

MODE	TIMER 0 (TIMER 3) FUNTION	TMOD (T34MOD)	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

Table 4-7 Timer0 and Timer3 TMOD

MODE	COUNTER 0 (COUNTER 3) FUNTION	TMOD (T34MOD)	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	One 8-bit Counter	07H	0FH

Table 4-8 Counter0 and Counter3 TMOD

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR0 (TR3) by the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on /INT0 (/INT3) when TR0 = 1 (hardware control).

TIMER/COUNTER 1 (TIMER/COUNTER 4)

MODE	TIMER 1 (TIMER 4) FUNTION	TMOD (T34MOD)	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

Table 4-9 Timer0 and Timer3 TMOD

MODE	COUNTER 1 (COUNTER 4) FUNTION	TMOD (T34MOD)	
		INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	-	-

Table 4-10 Counter0 and Counter3 TMOD

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR0 (TR3) by the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on /INT1 (/INT4) when TR1 = 1 (hardware control).

T2CON: TIMER/COUNTER 2 CONTROL REGISTER. BIT ADDRESSABLE. : C8H

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2	T2CON.7	Timer 2 Overflow flag set by hardware and cleared by software. TF2 cannot be set when either RCLK =1 or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX, and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its receive clock in modes 1 & 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its transmit clock in modes 1 & 3. TCLK = 0 causes Timer 1 overflow to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of negative transition on T2EX if Timer 2 is not being used to clock the Serial Port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Software START/STOP control for Timer 2 . A logic 1 starts the Timer.
C/$\overline{T2}$	T2CON.1	Timer or Counter selector. 0 = Internal Timer 1 = External Event Counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, Auto-Reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 =1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the Timer is forced to Auto-Reload on Timer 2 overflow.

TIMER/COUNTER 2 SET-UP

Except for the baud rate generator mode, the value given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the Timer on.

MODE	T2CON	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
BAUD rate generator receive & transmit same baud rate	34H	36H
receive only	24H	26H
transmit only	14H	16H

Table 4-11 Timer 2 Mode

MODE	TMOD	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	02H	0AH
16-bit Capture	03H	0BH

Table 4-12 Counter 2 Mode

NOTES:
1. Capture/Reload occurs only on Timer/Counter overflow.
2. Capture/Reload occurs on Timer/Counter overflow and a 1 to 0 transition on T2EX pin except when Timer 2 is used in the baud rate generating mode.

SCON: SERIAL PORT CONTROL REGISTER.(UART) BIT ADDRESSABLE. : 98H

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial Port mode specifier. (See Table 4-13).
SM1	SCON.6	Serial Port mode specifier. (See Table 4-13).
SM2	SCON.5	Enables the multiprocessor communication feature in modes 2&3. In modes 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.
REN	SCON.4	Set/Cleared by software to Enable/Disable reception.
TB8	SCON.3	The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.
RB8	SCON.2	In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received, In mode 0, RB8 is not used.
TI	SCON.1	Transmit interrupt flag. Set by hardware at the end 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware at the end 8th bit time in mode 0, or halfway through the stop bit in the other modes (except see SM2). Must be cleared by software.

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	$f_{CPU}/6$*
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	$f_{CPU}/32$* or $f_{CPU}/16$*
1	1	3	9-Bit UART	Variable

Table 4-13 UART Mode

* f_{CPU} : CPU Clock Frequency ($f_{OSC}/2$, $f_{OSC}/4$, $f_{OSC}/8$, $f_{OSC}/16$, $f_{OSC}/32$)

f_{OSC} : Oscillator Clock Frequency

SERIAL PORT SET-UP

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

Table 4-14 Serial Port

GENERATING BAUD RATES

Serial Port in Mode 0:

Timer/Counters need to be stop. Only the SCON register needs to be defined.

$$Baud\ Rate = \frac{2 \times f_{CPU}}{12}$$

Serial Port in Mode 1:

Mode 1 has a variable baud rate. The baud rate can be generated by either Timer 1 or Timer 2

USING TIMER/COUNTER 1 TO GENERATE BAUD RATES:

For this purpose, Timer 1 is used in mode 2 (Auto-Reload). Refer to Timer Setup section of this chapter.

$$Baud\ Rate = \frac{K \times 2 \times f_{CPU}}{32 \times 12 \times [256 - (TH1)]}$$

If SMOD = 0, then K =1.

If SMOD = 1, then K = 2. (SMOD is the PCON register).

Most of the timer the user knows the baud rate and needs to know the reload value for TH1.

Therefore, the equation to calculate TH1 can be written as:

$$TH1 = 256 - \frac{K \times 2 \times f_{CPU}}{384 \times Baud\ Rate}$$

TH1 must be an integer value. Rounding off TH1 to the nearest integer may not produce the desired baud rate. In this case, the user may have to choose another crystal frequency.

Since the PCON register is not bit addressable, one way to set the bit is logical ORing the PCON register. (i.e., ORL PCON, #80H). The address of PCON is 87H.

USING TIMER/COUNTER 2 TO GENERATE BAUD RATES:

For this purpose, Timer 2 must be used in the baud rate generating mode. If Timer 2 is being clocked through pin T2 the baud rate is:

$$Baud\ Rate = \frac{Timer2\ Overflow\ Rate}{16}$$

And if it is being clocked internally the baud rate is:

$$Baud\ Rate = \frac{2 \times f_{CPU}}{32 \times [65536 - (RCAP2H, RCAP2L)]}$$

To obtain the reload value for RCAP2H and RCAP2Ll the above can be rewritten as:

$$RCAP2H, RCAP2L = 65535 - \frac{2 \times f_{CPU}}{32 \times Baud\ Rate}$$

SERIAL PORT IN MODE 2:

The baud rate is fixed in this mode and is $^1/_{16}$ or $^1/_{32}$ of the CPU clock depending on the value of the SMOD bit in the PCON register. In this mode, the Timers are used and the clock comes from the internal phase 2 clock.

SMOD = 1

$$Baud\ Rate = \frac{2 \times f_{CPU}}{32}$$

SMOD = 0

$$Baud\ Rate = \frac{2 \times f_{CPU}}{64}$$

To set the SMOD bit: ORL PCON, #80H. The address of PCON is 87H.

SERIAL PORT IN MODE 3:

The baud rate in mode 3 is variable and sets up exactly the same as in mode 1.

S12CON: SIO1 & SIO2 CONTROL REGISTER. BIT ADDRESSABLE. : A0H

SIO2HIZ	SIO2TS	SIO2CK1	SIO2CK0	SIO1HIZ	SIO1TS	SIO1CK1	SIO1CK0

SIO2HIZ	S12CON.7	Software Port control for SiO2. A logic 1 assigns general I/O port to SIO2 port
SIO2TS	S12CON.6	Software START/STOP control for SIO2. A logic 1 starts the SIO2
SIO2CK1	S12CON.5	See Table 4-15
SIO2CK0	S12CON.4	See Table 4-15
SIO1HIZ	S12CON.3	Software Port control for SiO1. A logic 1 assigns general I/O port to SIO1 port
SIO1TS	S12CON.2	Software START/STOP control for SIO1. A logic 1 starts the SIO1
SIO1CK1	S12CON.1	See Table 4-15
SIO1CK0	S12CON.0	See Table 4-15

:

SIO1/2CK1	SIO1/2CK0	Set input/output clock frequency of SIO1 (f_{OSC} = 7.2 MHz)
0	0	Slave mode : External clock
0	1	Master mode : 75KHz
1	0	Master mode : 150KHz
1	1	Master mode : 450KHz

Table 4-15 SIO Clock

PLLMOD : PLL MODE & REFERENCE FREQUENCY SELECT REGISTER. BIT ADDRESSABLE. : F1H

PLLRF3	PLLRF2	PLLRF1	PLLRF0	PLLUL1	PLLUL0	PLLMD1	PLLMD0

PLLRF3	PLLMOD.7	See Table 4-16
PLLRF2	PLLMOD.6	See Table 4-16
PLLRF1	PLLMOD.5	See Table 4-16
PLLRF0	PLLMOD.4	See Table 4-16
PLLUL1	PLLMOD.3	Detects status of unlock FF1 (1.1μs). Set by hardware when PLL locks 900KHz
PLLUL0	PLLMOD.2	Detects status of unlock FF0 (2.2μs). Set by hardware when PLL locks 450KHz
PLLMD1	PLLMOD.1	See Table 4-17
PLLMD0	PLLMOD.0	See Table 4-17

PLLRF3	PLLRF2	PLLRF1	PLLRF0	Reference Frequency of PLL (f_{OSC} = 7.2 MHz)
0	0	0	0	PLL stop
0	0	0	1	1KHz
0	0	1	0	1.25KHz
0	0	1	1	2.5KHz
0	1	0	0	3KHz
0	1	0	1	5KHz
0	1	1	0	6.25KHz
0	1	1	1	9KHz
1	0	0	0	10KHz
1	0	0	1	12.5KHz
1	0	1	0	18KHz
1	0	1	1	20KHz
1	1	0	0	25KHz
1	1	0	1	50KHz
1	1	1	0	Reserved for future use *
1	1	1	1	Reserved for future use *

Table 4-16 PLL Reference Frequency

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

PLLMD1	PLLMD0	Selects of PLL input pin (f_{OSC} = 7.2 MHz)
0	0	Disable VCOL & VCOH pins
0	1	VCOH & VHF mode select
1	0	VCOL & HF mode select
1	1	VCOL & MF mode select

Table 4-17 PLL Mode

IFCMOD : IFC MODE SELECT & CONTROL REGISTER. BIT ADDRESSABLE. : F4H

IFCJR	IFCST	IFCCLR	-	IFCGT1	IFCGT0	IFCMD1	IFCMD0

IFCJR	IFCMOD.7	IF counter judge register. Set by hardware automatically when IF counting is ended, Cleared by hardware automatically when software reads IFCMOD register or IF interrupt service routine is started.
IFCST	IFCMOD.6	Software START/STOP control for IF counter. A logic 1 starts the IF counter.
IFCCLR	IFCMOD.5	A logic 1 resets the IF counter.
	IFCMOD.4	Reserved for future use *
IFCGT1	IFCMOD.3	See Table 4-18
IFCGT0	IFCMOD.2	See Table 4-18
IFCMD1	IFCMOD.1	See Table 4-19
IFCMD0	IFCMOD.0	See Table 4-19

NOTE:

IFCGT1	IFCGT0	Setting of IFC gate time (f_{OSC} = 7.2 MHz)
0	0	8ms
0	1	32ms
1	0	128ms
1	1	Soft

Table 4-18 IFC Gate Time

IFCMD1	IFCMD0	Selects of IFC input
0	X	Disable FMIFC & AMIFC pins
1	0	FMIFC pin select
1	1	AMIFC pin select

Table 4-19 IFC Mode

* User software should not write 1s to reserved bits. These bits may be used in future DTS3 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

IFCDR2 : IF counter data register 2. : F5H

-	-	-	-	IFCDET	IFCDATA18	IFCDATA17	IFCDATA16

-	IFCDR2.7	Reserved for future use
-	IFCDR2.6	Reserved for future use
-	IFCDR2.5	Reserved for future use
-	IFCDR2.4	Reserved for future use
IFCDET	IFCDR2.3	Detection bit of 19bit IF counter overflow. A logic 1 implies the overflow of IF counter. It can be reset by IFCCLR. (See IF Counter Control Register
IFCDATA18	IFCDR2.2	19^{th} bit of 19bit IF counter (MSB)
IFCDATA17	IFCDR2.1	18^{th} bit of 19bit IF counter
IFCDATA16	IFCDR2.0	17^{th} bit of 19bit IF counter

IFCDR1 : IF counter data register 1. : F6H

IFCDATA15	IFCDATA14	IFCDATA13	IFCDATA12	IFCDATA11	IFCDATA10	IFCDATA9	IFCDATA8

IFCDATA15	IFCDR1.7	16^{th} bit of 19bit IF counter
IFCDATA14	IFCDR1.6	15^{th} bit of 19bit IF counter
IFCDATA13	IFCDR1.5	14^{th} bit of 19bit IF counter
IFCDATA12	IFCDR1.4	13^{th} bit of 19bit IF counter
IFCDATA11	IFCDR1.3	12^{th} bit of 19bit IF counter
IFCDATA10	IFCDR1.2	11^{th} bit of 19bit IF counter
IFCDATA9	IFCDR1.1	10^{th} bit of 19bit IF counter
IFCDATA8	IFCDR1.0	9^{th} bit of 19bit IF counter

IFCDR0 : IF counter data register 0. : F7H

IFCDATA7	IFCDATA6	IFCDATA5	IFCDATA4	IFCDATA3	IFCDATA2	IFCDATA1	IFCDATA0

IFCDATA7	IFCDR0.7	8^{th} bit of 19bit IF counter
IFCDATA6	IFCDR0.6	7^{th} bit of 19bit IF counter
IFCDATA5	IFCDR0.5	6^{th} bit of 19bit IF counter
IFCDATA4	IFCDR0.4	5^{th} bit of 19bit IF counter
IFCDATA3	IFCDR0.3	4^{th} bit of 19bit IF counter
IFCDATA2	IFCDR0.2	3^{rd} bit of 19bit IF counter
IFCDATA1	IFCDR0.1	2^{nd} bit of 19bit IF counter
IFCDATA0	IFCDR0.0	1^{st} bit of 19bit IF counter (LSB)

WDTCON: BEEPER & WATCHDOG TIMER CONTROL REGISTER. BIT ADDRESSABLE. : F8H

RUNBEEP	BEEPMD1	BEEPMD0	RUNWDT	WDTMK	WDTMD2	WDTMD1	WDTMD0

RUNBEEP	WDTCON.7	Software START/STOP control for Beeper. A logic 1 starts the Beeper.
BEEPMD1	WDTCON.6	See Table 4-20
BEEPMD0	WDTCON.5	See Table 4-20
RUNWDT	WDTCON.4	Restart Watchdog timer (This bit is automatically cleared to “0” after restart.).
WDTMK	WDTCON.3	Software Enable/Disable NMI(Non Maskable Interrupt) for WDT. A logic 1 makes WDT interrupt NMI
WDTMD2	WDTCON.2	See Table 4-21
WDTMD1	WDTCON.1	See Table 4-21
WDTMD0	WDTCON.0	See Table 4-21

BEEPMD1	BEEPMD0	Select Beeper Clock Frequency (f_{OSC} = 7.2 MHz)
0	0	1.2KHz (f_{OSC} / 6000)
0	1	2.4KHz (f_{OSC} / 3000)
1	0	4.5KHz (f_{OSC} / 1600)
1	1	8KHz (f_{OSC} / 900)

Table 4-20 BEEP Mode

WDTMD2	WDTMD1	WDTMD0	Selects of WDT input (f_{OSC} = 7.2 MHz)
0	0	0	fxx ($f_{XX} = f_{OSC}/2$)
0	0	1	fxx / 2^3
0	1	0	fxx / 2^4
0	1	1	fxx / 2^5
1	0	0	fxx / 2^7
1	0	1	fxx / 2^9
1	1	0	fxx / 2^11
1	1	1	fxx / 2^13

Table 4-21 Watchdog Timer

SCMOD : SYSTEM CLOCK & POWER CONTROL REGISTER. BIT ADDRESSABLE. : 80H

-	-	-	SCSTOP	SCSW	SCMOD2	SCMOD1	SCMOD0

-	SCMOD.7	Reserved for future use *
-	SCMOD.6	Reserved for future use *
-	SCMOD.5	Reserved for future use *
SCSTOP	SCMOD.4	Software control of the main system oscillator. A logic 1 pulls down the main system oscillator (7.2MHz).
SCSW	SCMOD.3	Software switch control between main system oscillator and sub system oscillator. A logic 1 switches sub system oscillator (32.768KHz).
SCMOD2	SCMOD.2	See Table 4-22
SCMOD1	SCMOD.1	See Table 4-22
SCMOD0	SCMOD.0	See Table 4-22

SCMOD2	SCMOD1	SCMOD0	Select system clock
0	x	x	fxx
1	0	0	fxx / 2
1	0	1	fxx / 4
1	1	0	fxx / 8
1	1	1	fxx / 16

Table 4-22 Select System Clock

ADCCON: ADC CONTROL REGISTER. BIT ADDRESSABLE. : 84H

-	ADCEN	-	ADCCH2	ADCCH1	ADCCH0	ADCST	ADCSF

-	ADCCON.7	Reserved for future use *
ADCEN	ADCCON.6	ADC Enable flag. This bit is a write-only register.
-	ADCCON.5	Reserved for future use *
ADCCH2	ADCCON.4	See Table 4-23. This bit is a write-only register.
ADCCH1	ADCCON.3	See Table 4-23. This bit is a write-only register.
ADCCH0	ADCCON.2	See Table 4-23. This bit is a write-only register.
ADCST	ADCCON.1	Software START control for ADC. A logic 1 starts A/D conversion. This bit is a write-only register.
ADCSF	ADCCON.0	A/D conversion completion flag. Set by hardware when ADC operation complete. Cleared by hardware when this flag is read.

ADCCH2	ADCCH1	ADCCH0	Select ADC channel
0	0	0	Select channel 0
0	0	1	Select channel 1
0	1	0	Select channel 2
0	1	1	Select channel 3
1	0	0	Select channel 4
1	0	1	Select channel 5
1	1	0	Select channel 6
1	1	1	Select channel 7

Table 4-23 ADC Channel Select

SFRPG: SFR PAGE REGISTER. NOT BIT ADDRESSABLE. : FFH

-	-	-	-	-	-	-	SFRP

-	SFRPG.7	Reserved for future use *
-	SFRPG.6	Reserved for future use *
-	SFRPG.5	Reserved for future use *
-	SFRPG.4	Reserved for future use *
-	SFRPG.3	Reserved for future use *
-	SFRPG.2	Reserved for future use *
-	SFRPG.1	Reserved for future use *
SFRP	SFRPG.0	Software SFR page0/page1 control flag. A logic 1 switches to SFR page 1.

P0MOD: PORT0 MODE REGISTER. NOT BIT ADDRESSABLE. : B4H

P0MD7	P0MD6	P0MD5	P0MD4	P0MD3	P0MD2	P0MD1	P0MD0

P0MD7	P0MOD.7	Software Input/Output mode control flag for P0.7. A logic 1 changes P0.7 to input mode.
P0MD6	P0MOD.6	Software Input/Output mode control flag for P0.6. A logic 1 changes P0.6 to input mode.
P0MD5	P0MOD.5	Software Input/Output mode control flag for P0.5. A logic 1 changes P0.5 to input mode.
P0MD4	P0MOD.4	Software Input/Output mode control flag for P0.4. A logic 1 changes P0.4 to input mode.
P0MD3	P0MOD.3	Software Input/Output mode control flag for P0.3. A logic 1 changes P0.3 to input mode.
P0MD2	P0MOD.2	Software Input/Output mode control flag for P0.2. A logic 1 changes P0.2 to input mode.
P0MD1	P0MOD.1	Software Input/Output mode control flag for P0.1. A logic 1 changes P0.1 to input mode.
P0MD0	P0MOD.0	Software Input/Output mode control flag for P0.0. A logic 1 changes P0.0 to input mode.

P1MOD: PORT1 MODE REGISTER. NOT BIT ADDRESSABLE. : B5H

P1MD7	P1MD6	P1MD5	P1MD4	P1MD3	P1MD2	P1MD1	P1MD0

P1MD7	P1MOD.7	Software Input/Output mode control flag for P1.7. A logic 1 changes P1.7 to input mode.
P1MD6	P1MOD.6	Software Input/Output mode control flag for P1.6. A logic 1 changes P1.6 to input mode.
P1MD5	P1MOD.5	Software Input/Output mode control flag for P1.5. A logic 1 changes P1.5 to input mode.
P1MD4	P1MOD.4	Software Input/Output mode control flag for P1.4. A logic 1 changes P1.4 to input mode.
P1MD3	P1MOD.3	Software Input/Output mode control flag for P1.3. A logic 1 changes P1.3 to input mode.
P1MD2	P1MOD.2	Software Input/Output mode control flag for P1.2. A logic 1 changes P1.2 to input mode.
P1MD1	P1MOD.1	Software Input/Output mode control flag for P1.1. A logic 1 changes P1.1 to input mode.
P1MD0	P1MOD.0	Software Input/Output mode control flag for P1.0. A logic 1 changes P1.0 to input mode.

P2MOD: PORT2 MODE REGISTER. NOT BIT ADDRESSABLE. : B6H

P2MD7	P2MD6	P2MD5	P2MD4	P2MD3	P2MD2	P2MD1	P2MD0

P2MD7	P2MOD.7	Software Input/Output mode control flag for P2.7. A logic 1 changes P2.7 to input mode.
P2MD6	P2MOD.6	Software Input/Output mode control flag for P2.6. A logic 1 changes P2.6 to input mode.
P2MD5	P2MOD.5	Software Input/Output mode control flag for P2.5. A logic 1 changes P2.5 to input mode.
P2MD4	P2MOD.4	Software Input/Output mode control flag for P2.4. A logic 1 changes P2.4 to input mode.
P2MD3	P2MOD.3	Software Input/Output mode control flag for P2.3. A logic 1 changes P2.3 to input mode.
P2MD2	P2MOD.2	Software Input/Output mode control flag for P2.2. A logic 1 changes P2.2 to input mode.
P2MD1	P2MOD.1	Software Input/Output mode control flag for P2.1. A logic 1 changes P2.1 to input mode.
P2MD0	P2MOD.0	Software Input/Output mode control flag for P2.0. A logic 1 changes P2.0 to input mode.

P3MOD: PORT3 MODE REGISTER. NOT BIT ADDRESSABLE. : B7H

P3MD7	P3MD6	P3MD5	P3MD4	P3MD3	P3MD2	P3MD1	P3MD0

-	P3MOD.7	Reserved for future use.
-	P3MOD.6	Reserved for future use.
P3MD5	P3MOD.5	Software Input/Output mode control flag for P3.5. A logic 1 changes P3.5 to input mode.
P3MD4	P3MOD.4	Software Input/Output mode control flag for P3.4. A logic 1 changes P3.4 to input mode.
P3MD3	P3MOD.3	Software Input/Output mode control flag for P3.3. A logic 1 changes P3.3 to input mode.
P3MD2	P3MOD.2	Software Input/Output mode control flag for P3.2. A logic 1 changes P3.2 to input mode.
P3MD1	P3MOD.1	Software Input/Output mode control flag for P3.1. A logic 1 changes P3.1 to input mode.
P3MD0	P3MOD.0	Software Input/Output mode control flag for P3.0. A logic 1 changes P3.0 to input mode.

P4MOD: PORT4 MODE REGISTER. NOT BIT ADDRESSABLE. : BCH

P4MD7	P4MD6	P4MD5	P4MD4	P4MD3	P4MD2	P4MD1	P4MD0

P4MD7	P4MOD.7	Software Input/Output mode control flag for P4.7. A logic 1 changes P4.7 to input mode.
P4MD6	P4MOD.6	Software Input/Output mode control flag for P4.6. A logic 1 changes P4.6 to input mode.
P4MD5	P4MOD.5	Software Input/Output mode control flag for P4.5. A logic 1 changes P4.5 to input mode.
P4MD4	P4MOD.4	Software Input/Output mode control flag for P4.4. A logic 1 changes P4.4 to input mode.
P4MD3	P4MOD.3	Software Input/Output mode control flag for P4.3. A logic 1 changes P4.3 to input mode.
P4MD2	P4MOD.2	Software Input/Output mode control flag for P4.2. A logic 1 changes P4.2 to input mode.
P4MD1	P4MOD.1	Software Input/Output mode control flag for P4.1. A logic 1 changes P4.1 to input mode.
P4MD0	P4MOD.0	Software Input/Output mode control flag for P4.0. A logic 1 changes P4.0 to input mode.

P5MOD: PORT5 MODE REGISTER. NOT BIT ADDRESSABLE : BDH

P5MD7	P5MD6	P5MD5	P5MD4	P5MD3	P5MD2	P5MD1	P5MD0

P5MD7	P5MOD.7	Software Input/Output mode control flag for P5.7. A logic 1 changes P5.7 to input mode.
P5MD6	P5MOD.6	Software Input/Output mode control flag for P5.6. A logic 1 changes P5.6 to input mode.
P5MD5	P5MOD.5	Software Input/Output mode control flag for P5.5. A logic 1 changes P5.5 to input mode.
P5MD4	P5MOD.4	Software Input/Output mode control flag for P5.4. A logic 1 changes P5.4 to input mode.
P5MD3	P5MOD.3	Software Input/Output mode control flag for P5.3. A logic 1 changes P5.3 to input mode.
P5MD2	P5MOD.2	Software Input/Output mode control flag for P5.2. A logic 1 changes P5.2 to input mode.
P5MD1	P5MOD.1	Software Input/Output mode control flag for P5.1. A logic 1 changes P5.1 to input mode.
P5MD0	P5MOD.0	Software Input/Output mode control flag for P5.0. A logic 1 changes P5.0 to input mode.

P6MOD: PORT6 MODE REGISTER. NOT BIT ADDRESSABLE. : BEH

P6MD7	P6MD6	P6MD5	P6MD4	P6MD3	P6MD2	P6MD1	P6MD0

P6MD7	P6MOD.7	Software Input/Output mode control flag for P6.7. A logic 1 changes P6.7 to input mode.
P6MD6	P6MOD.6	Software Input/Output mode control flag for P6.6. A logic 1 changes P6.6 to input mode.
P6MD5	P6MOD.5	Software Input/Output mode control flag for P6.5. A logic 1 changes P6.5 to input mode.
P6MD4	P6MOD.4	Software Input/Output mode control flag for P6.4. A logic 1 changes P6.4 to input mode.
P6MD3	P6MOD.3	Software Input/Output mode control flag for P6.3. A logic 1 changes P6.3 to input mode.
P6MD2	P6MOD.2	Software Input/Output mode control flag for P6.2. A logic 1 changes P6.2 to input mode.
P6MD1	P6MOD.1	Software Input/Output mode control flag for P6.1. A logic 1 changes P6.1 to input mode.
P6MD0	P6MOD.0	Software Input/Output mode control flag for P6.0. A logic 1 changes P6.0 to input mode.

P7MOD: PORT7 MODE REGISTER. NOT BIT ADDRESSABLE. : BFH

P7MD7	P7MD6	P7MD5	P7MD4	P7MD3	P7MD2	P7MD1	P7MD0

P7MD7	P7MOD.7	Software Input/Output mode control flag for P7.7. A logic 1 changes P7.7 to input mode.
P7MD6	P7MOD.6	Software Input/Output mode control flag for P7.6. A logic 1 changes P7.6 to input mode.
P7MD5	P7MOD.5	Software Input/Output mode control flag for P7.5. A logic 1 changes P7.5 to input mode.
P7MD4	P7MOD.4	Software Input/Output mode control flag for P7.4. A logic 1 changes P7.4 to input mode.
P7MD3	P7MOD.3	Software Input/Output mode control flag for P7.3. A logic 1 changes P7.3 to input mode.
P7MD2	P7MOD.2	Software Input/Output mode control flag for P7.2. A logic 1 changes P7.2 to input mode.
P7MD1	P7MOD.1	Software Input/Output mode control flag for P7.1. A logic 1 changes P7.1 to input mode.
P7MD0	P7MOD.0	Software Input/Output mode control flag for P7.0. A logic 1 changes P7.0 to input mode.

P0CON: PORT0 CON REGISTER. NOT BIT ADDRESSABLE. : A4H

P0CON7	P0CON6	P0CON5	P0CON4	P0CON3	P0CON2	P0CON1	P0CON0

P0CON7	P0CON.7	Software Enable/Disable pull-up TR control flag for P0.7. A logic 1 pulls up P0.7
P0CON6	P0CON.6	Software Enable/Disable pull-up TR control flag for P0.6. A logic 1 pulls up P0.6.
P0CON5	P0CON.5	Software Enable/Disable pull-up TR control flag for P0.5. A logic 1 pulls up P0.5.
P0CON4	P0CON.4	Software Enable/Disable pull-up TR control flag for P0.4. A logic 1 pulls up P0.4.
P0CON3	P0CON.3	Software Enable/Disable pull-up TR control flag for P0.3. A logic 1 pulls up P0.3.
P0CON2	P0CON.2	Software Enable/Disable pull-up TR control flag for P0.2. A logic 1 pulls up P0.2.
P0CON1	P0CON.1	Software Enable/Disable pull-up TR control flag for P0.1. A logic 1 pulls up P0.1.
P0CON0	P0CON.0	Software Enable/Disable pull-up TR control flag for P0.0. A logic 1 pulls up P0.0.

P1CON: PORT1 CON REGISTER. NOT BIT ADDRESSABLE. : A5H

P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	P1CON1	P1CON0

P1CON7	P1CON.7	Software Enable/Disable pull-up TR control flag for P1.7. A logic 1 pulls up P1.7.
P1CON6	P1CON.6	Software Enable/Disable pull-up TR control flag for P1.6. A logic 1 pulls up P1.6.
P1CON5	P1CON.5	Software Enable/Disable pull-up TR control flag for P1.5. A logic 1 pulls up P1.5.
P1CON4	P1CON.4	Software Enable/Disable pull-up TR control flag for P1.4. A logic 1 pulls up P1.4.
P1CON3	P1CON.3	Software Enable/Disable pull-up TR control flag for P1.3. A logic 1 pulls up P1.3.
P1CON2	P1CON.2	Software Enable/Disable pull-up TR control flag for P1.2. A logic 1 pulls up P1.2.
P1CON1	P1CON.1	Software Enable/Disable pull-up TR control flag for P1.1. A logic 1 pulls up P1.1.
P1CON0	P1CON.0	Software Enable/Disable pull-up TR control flag for P1.0. A logic 1 pulls up P1.0.

P2CON: PORT2 CON REGISTER. NOT BIT ADDRESSABLE. : A6H

P2CON7	P2CON6	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P2CON0

P2CON7	P2CON.7	Software Enable/Disable pull-up TR control flag for P2.7. A logic 1 pulls up P2.7.
P2CON6	P2CON.6	Software Enable/Disable pull-up TR control flag for P2.6. A logic 1 pulls up P2.6.
P2CON5	P2CON.5	Software Enable/Disable pull-up TR control flag for P2.5. A logic 1 pulls up P2.5.
P2CON4	P2CON.4	Software Enable/Disable pull-up TR control flag for P2.4. A logic 1 pulls up P2.4.
P2CON3	P2CON.3	Use not bit. P2.3 have no pulls up TR.
P2CON2	P2CON.2	Use not bit. P2.2 have no pulls up TR.
P2CON1	P2CON.1	Use not bit. P2.1 have no pulls up TR.
P2CON0	P2CON.0	Use not bit. P2.0 have no pulls up TR.

P3CON: PORT3 CON REGISTER. NOT BIT ADDRESSABLE. : A7H

P3CON7	P3CON6	P3CON5	P3CON4	P3CON3	P3CON2	P3CON1	P3CON0

-	P3CON.7	Reserved for future use.
-	P3CON.6	Reserved for future use.
P3CON5	P3CON.5	Software Enable/Disable pull-up TR control flag for P3.5. A logic 1 pulls up P3.5.
P3CON4	P3CON.4	Software Enable/Disable pull-up TR control flag for P3.4. A logic 1 pulls up P3.4.
P3CON3	P3CON.3	Software Enable/Disable pull-up TR control flag for P3.3. A logic 1 pulls up P3.3.
P3CON2	P3CON.2	Software Enable/Disable pull-up TR control flag for P3.2. A logic 1 pulls up P3.2.
P3CON1	P3CON.1	Software Enable/Disable pull-up TR control flag for P3.1. A logic 1 pulls up P3.1.
P3CON0	P3CON.0	Software Enable/Disable pull-up TR control flag for P3.0. A logic 1 pulls up P3.0.

P4CON: PORT4 CON REGISTER. NOT BIT ADDRESSABLE. : ACH

P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0

P4CON7	P4CON.7	Software Enable/Disable pull-up TR control flag for P4.7. A logic 1 pulls up P4.7.
P4CON6	P4CON.6	Software Enable/Disable pull-up TR control flag for P4.6. A logic 1 pulls up P4.6.
P4CON5	P4CON.5	Software Enable/Disable pull-up TR control flag for P4.5. A logic 1 pulls up P4.5.
P4CON4	P4CON.4	Software Enable/Disable pull-up TR control flag for P4.4. A logic 1 pulls up P4.4.
P4CON3	P4CON.3	Software Enable/Disable pull-up TR control flag for P4.3. A logic 1 pulls up P4.3.
P4CON2	P4CON.2	Software Enable/Disable pull-up TR control flag for P4.2. A logic 1 pulls up P4.2.
P4CON1	P4CON.1	Software Enable/Disable pull-up TR control flag for P4.1. A logic 1 pulls up P4.1.
P4CON0	P4CON.0	Software Enable/Disable pull-up TR control flag for P4.0. A logic 1 pulls up P4.0.

P5CON: PORT5 CON REGISTER. NOT BIT ADDRESSABLE. : ADH

P5CON7	P5CON6	P5CON5	P5CON4	P5CON3	P5CON2	P5CON1	P5CON0

P5CON7	P5CON.7	Software Enable/Disable pull-up TR control flag for P5.7. A logic 1 pulls up P5.7.
P5CON6	P5CON.6	Software Enable/Disable pull-up TR control flag for P5.6. A logic 1 pulls up P5.6.
P5CON5	P5CON.5	Software Enable/Disable pull-up TR control flag for P5.5. A logic 1 pulls up P5.5.
P5CON4	P5CON.4	Software Enable/Disable pull-up TR control flag for P5.4. A logic 1 pulls up P5.4.
P5CON3	P5CON.3	Software Enable/Disable pull-up TR control flag for P5.3. A logic 1 pulls up P5.3.
P5CON2	P5CON.2	Software Enable/Disable pull-up TR control flag for P5.2. A logic 1 pulls up P5.2.
P5CON1	P5CON.1	Software Enable/Disable pull-up TR control flag for P5.1. A logic 1 pulls up P5.1.
P5CON0	P5CON.0	Software Enable/Disable pull-up TR control flag for P5.0. A logic 1 pulls up P5.0.

P6CON: PORT6 CON REGISTER. NOT BIT ADDRESSABLE. : AEH

P6CON7	P6CON6	P6CON5	P6CON4	P6CON3	P6CON2	P6CON1	P6CON0

P6CON7	P6CON.7	Software Enable/Disable pull-up TR control flag for P6.7. A logic 1 pulls up P6.7.
P6CON6	P6CON.6	Software Enable/Disable pull-up TR control flag for P6.6. A logic 1 pulls up P6.6.
P6CON5	P6CON.5	Software Enable/Disable pull-up TR control flag for P6.5. A logic 1 pulls up P6.5.
P6CON4	P6CON.4	Software Enable/Disable pull-up TR control flag for P6.4. A logic 1 pulls up P6.4.
P6CON3	P6CON.3	Software Enable/Disable pull-up TR control flag for P6.3. A logic 1 pulls up P6.3.
P6CON2	P6CON.2	Software Enable/Disable pull-up TR control flag for P6.2. A logic 1 pulls up P6.2.
P6CON1	P6CON.1	Software Enable/Disable pull-up TR control flag for P6.1. A logic 1 pulls up P6.1.
P6CON0	P6CON.0	Software Enable/Disable pull-up TR control flag for P6.0. A logic 1 pulls up P6.0.

P7CON: PORT7 CON REGISTER. NOT BIT ADDRESSABLE. : AFH

P7CON7	P7CON6	P7CON5	P7CON4	P7CON3	P7CON2	P7CON1	P7CON0

P7CON7	P7CON.7	Software Enable/Disable pull-up TR control flag for P7.7. A logic 1 pulls up P7.7.
P7CON6	P7CON.6	Software Enable/Disable pull-up TR control flag for P7.6. A logic 1 pulls up P7.6.
P7CON5	P7CON.5	Software Enable/Disable pull-up TR control flag for P7.5. A logic 1 pulls up P7.5.
P7CON4	P7CON.4	Software Enable/Disable pull-up TR control flag for P7.4. A logic 1 pulls up P7.4.
P7CON3	P7CON.3	Software Enable/Disable pull-up TR control flag for P7.3. A logic 1 pulls up P7.3.
P7CON2	P7CON.2	Software Enable/Disable pull-up TR control flag for P7.2. A logic 1 pulls up P7.2.
P7CON1	P7CON.1	Software Enable/Disable pull-up TR control flag for P7.1. A logic 1 pulls up P7.1.
P7CON0	P7CON.0	Software Enable/Disable pull-up TR control flag for P7.0. A logic 1 pulls up P7.0.

4.3 Timer/Counters (Timer0, Timer1 and Timer2)

The HMS9XC8032 has five 16-bit Timer/Counter registers: Timer 0, Timer 1, Timer2, Timer 3 and Timer 4. All of them can be configured to operate either as timers or event counters. In this chapter, Timer0, Timer1 and Timer2 which are compatible with Intel 8052 are described. Timer3 and Timer4 are described in Part C: Timer/Counters (Timer3 and Timer4) chapter.

In the "Timer" function, the register is incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 6 CPU clock periods, the count rate is 1/6 of the CPU clock frequency.

In the "Counter" function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S2P1 of the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (12 CPU clock periods) to recognize a 1-to-0 transition, the maximum count rate is 1/12 of the CPU clock frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full cycle. In addition to the "Timer" or "Counter" selection, Timer 0 and Timer1 have four operating modes from which to select.

Timer 0 and Timer 1

The "Timer" or "Counter" function is selected by control bits C/T in the Special Function Register TMOD. These Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Modes 0, 1, and 2 are the same for Timers/Counters. Mode 3 is different. The four operating modes are described in the following text.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance	Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.	IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt Processed.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.	IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling Edge/low level triggered external Interrupt.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.	IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt Processed.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.	IT0	TCON.0	Interrupt 1 Type control bit. Set/cleared by software to specify falling Edge/low level triggered external Interrupt.

Figure 4-4 TCON Control Reigster

Mode 0

Putting either Timer into Mode 0 makes it look like an 8048 Timer, which is an 8-bit Counter with a divide-by-32 prescaler. Figure 4-6 shows the Mode 0 operation as it applies to Timer 1.

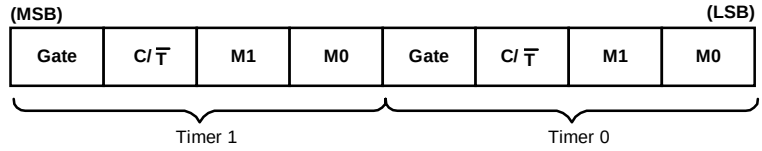
In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TF1. The counted input is enabled to the Timer when TR1 = 1 and either GATE = 0 or /INT1 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input /INT1, to facilitate pulse width measurements). TR1 is a control bit in the Special Function Register TCON (TCON Control Reigster). GATE is in TMOD.

The 13-bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 are indeterminate and should be ignored. Setting the run flag does not clear the registers.

Mode 0 operation is the same for the Timer 0 as for Timer 1. Substitute TR0, TF0, and /INT0 for the corresponding Timer 1 signals in Figure 4-6. There are two different GATE bits, one for Timer 1 and one for Timer 0.

Mode 1

Mode 1 is the same as Mode 0, except that the Timer register is being run with all 16 bits.



		M1	M0	Operating
Gate	Gating Control when set. Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.	0	0	13-bit Timer/Counter
		0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded : there is no prescaler.
C/ T	Timer or Counter Selector cleared for Timer operation (imput from internal system clock). Set for Counter operation (input from "Tx" input pin).	1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
		1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
		1	1	(Timer 1) Timer/Counter 1 stopped.

Figure 4-5 TMOD Register

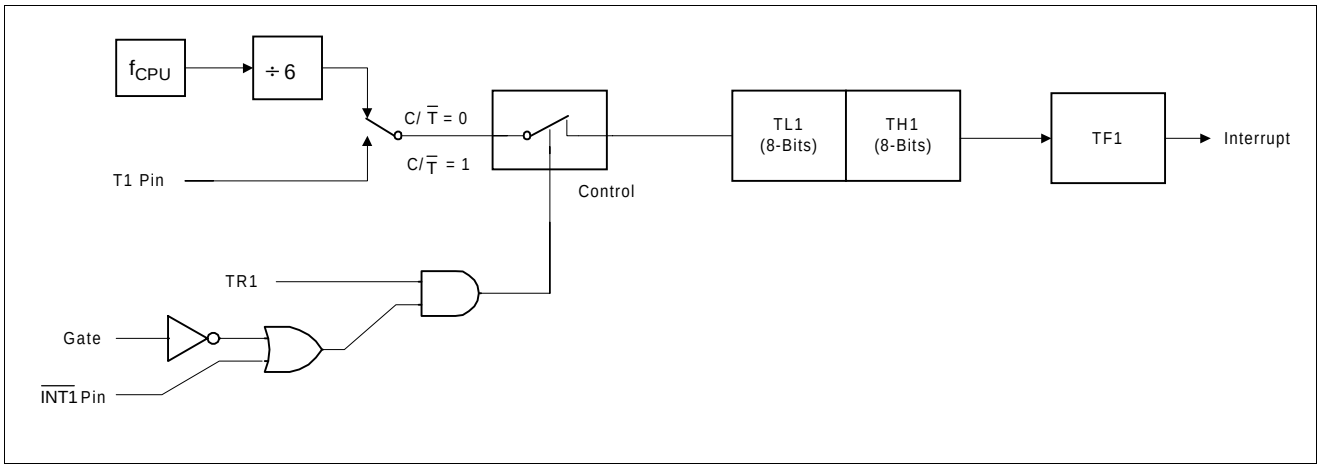


Figure 4-6 Timer/Counter Mode 0: 13-bit Counter

Mode 2

Mode 2 configures the Timer register as an 8-bit Counter (TL1) with automatic reload, as shown in Figure 4-7. Overflow from TL1 not only sets TF1, but also reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.Mode 2 operation is the same for Timer/Counter 0.

Mode 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 4-8. TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now controls the "Timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer on the counter. With Timer 0 in Mode 3, an HMS9XC8032 can look like it has three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in fact, in any application not requiring an interrupt.

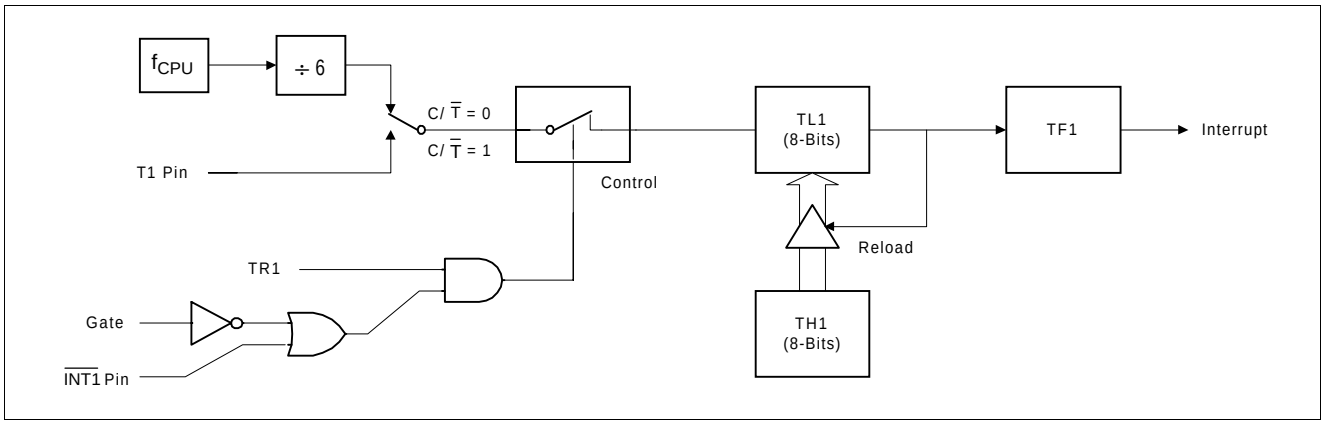


Figure 4-7 Timer/Counter Mode 2: 8-bit Auto-reload

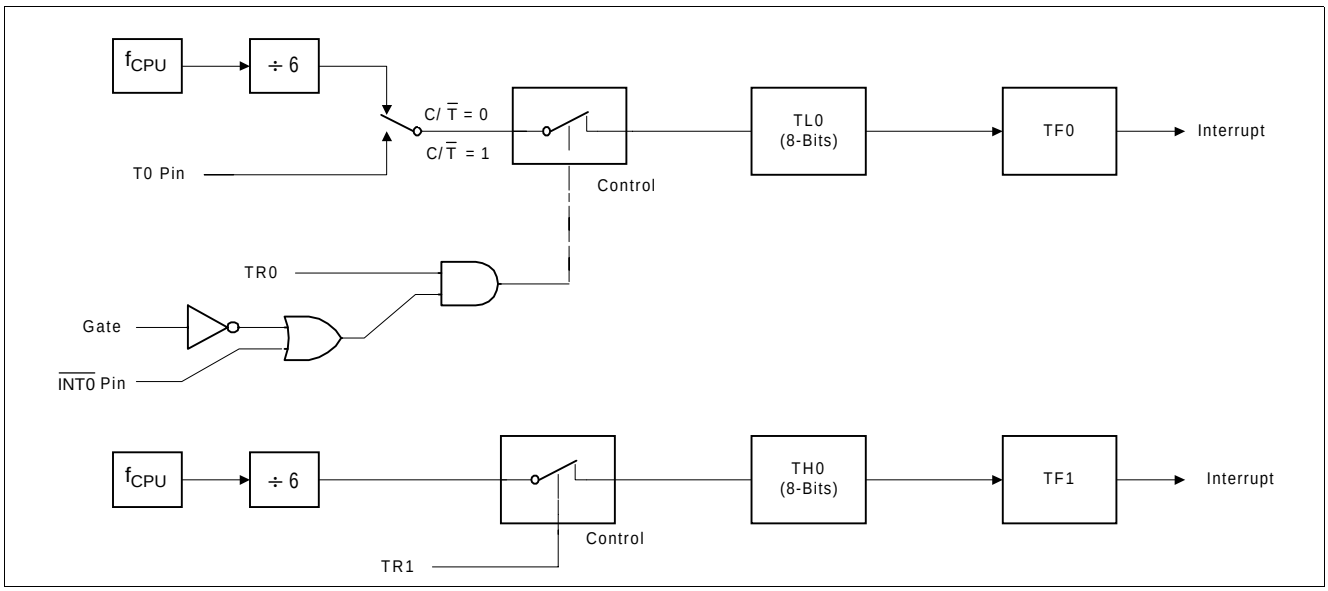


Figure 4-8 Timer/Counter Mode 3: Two 8-bit Counters

Timer 2

In addition to timer/counter 0, 1, 3 and 4 of the HMS9XC8032, the HMS9XC8032 contains timer/counter 2. Like timer 0, 1, 3 and 4, timer 2 can operate as either an event timer or as an event counter. This is selected by bit C/T2 in the special function register T2CON (see Figure 4-9). It has three operating modes: capture, auto-load, and baud rate generator, which are selected by bits in the T2CON as shown in Table 4-24.In the Capture Mode there are two options which are selected by bit EXEN2 in T2CON. if EXEN2 = 0, then Timer 2 is a 16-bit timer or counter which upon overflowing sets bit TF2, the Timer 2 overflow bit, which can be used to generate an interrupt. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2 like TF2 can generate an interrupt. The Capture Mode is illustrated in Figure 4-10.

In the auto-reload mode, there are again two options, which are selected by bit EXEN2 in T2CON. If EXEN2 = 0, then when Timer 2 rolls over it not only sets TF2 but also causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. If EXEN2 = 1, then Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX will also trigger the 16-bit reload and set EXF2. The auto-reload mode is illustrated in Standard Serial Interface (UART)Figure 4-11.

The baud rate generation mode is selected by RCLK = 1 and/or TCLK = 1. It will be described in conjunction with the serial port.

Timer/Counter 2 Set-up

T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the timer on.

(MSB) (LSB)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflow to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/$\overline{T2}$	T2CON.1	Timer or Counter select. (Timer 2) 0 = Internal timer ($f_{CPU}/6$) 1 = External event counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON.0	Capture/Reload flag. When set, capture will occur on negative transition at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and timer is forced to auto-reload on Timer 2 overflow.

Figure 4-9 Timer/Counter 2 Control Register (T2CON)

RCLK + TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud rate generator
1	X	0	(off)

Table 4-24 Timer2 Operating Modes

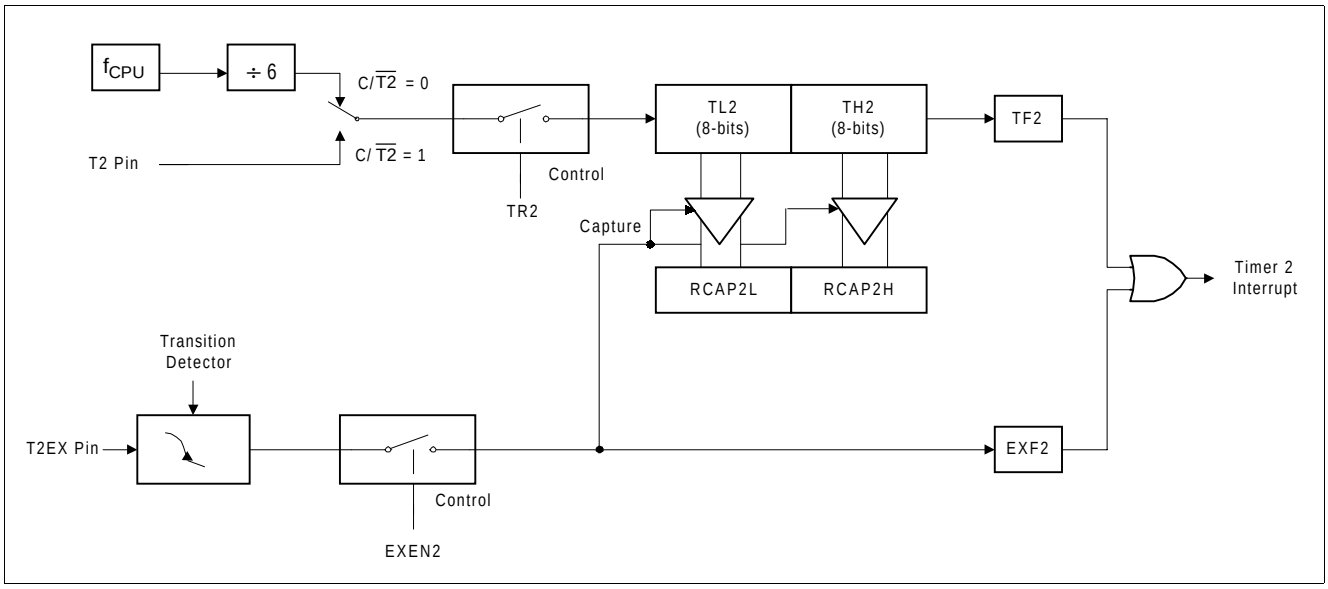


Figure 4-10 Timer2 in Capture Mode

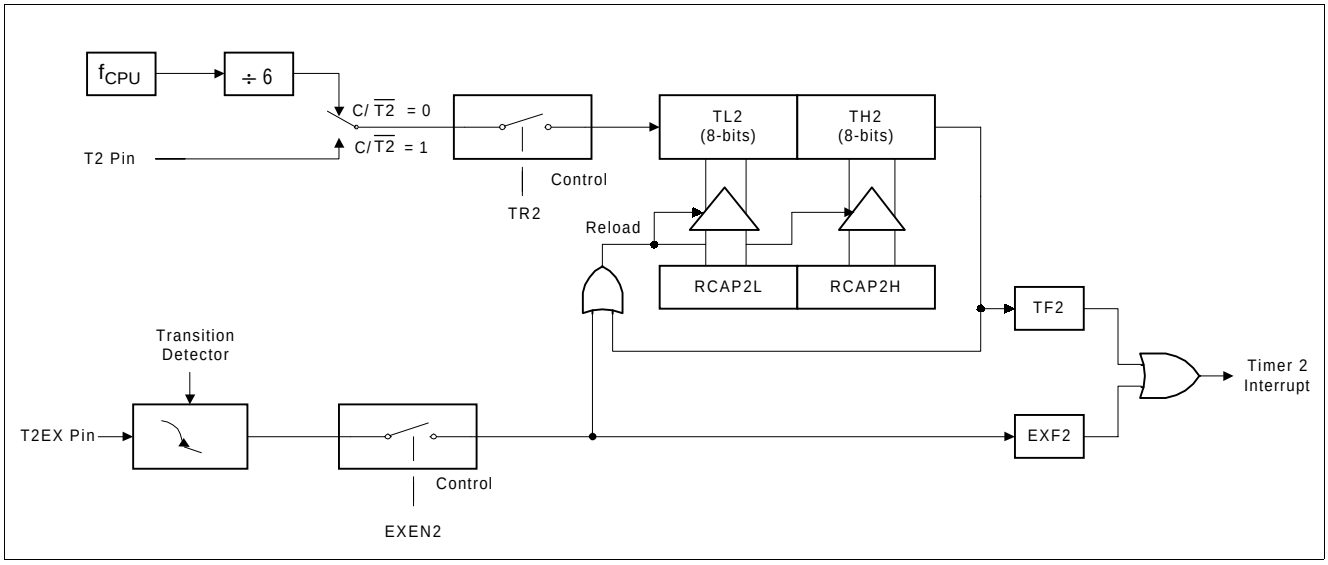


Figure 4-11 Timer 2 in Auto-Reload Mode

4.4 Timer/Counters (Timer3 and Timer4)

HMS9XC8032 has five 16-bit general-purpose Timer/Counter. Timer0, Timer1 and Timer2, which are compatible with genuine 8052, are described in "4.3 Timer/Counters (Timer0, Timer1 and Timer2)" on page 43. It is a clone in functional level between Timer0 and Timer3, and between Timer4 and Timer1. But Timer3(Timer4) has a little difference from Timer0(Timer1). It is the counting clock source for Timer/Counter that make a difference of Timer3(Timer4) from Timer0(Timer1).

* The f_{CPU} and the f_{SOSC} are shown in Figure 4-2

The counting clock sources of Timer0 and Timer1 are Xtin/12 for Timer and external signal like T0, T1 and T2 for Counter. But for the counting clock sources of Timer3 and Timer4, Xtin2 for Timer is added to the above two sources. In Timer3 and Timer4, to select Xtal2 for counting clock source, turn on T3_SUB for Timer3 or T4_SUB for Timer4 in T34CON.

.

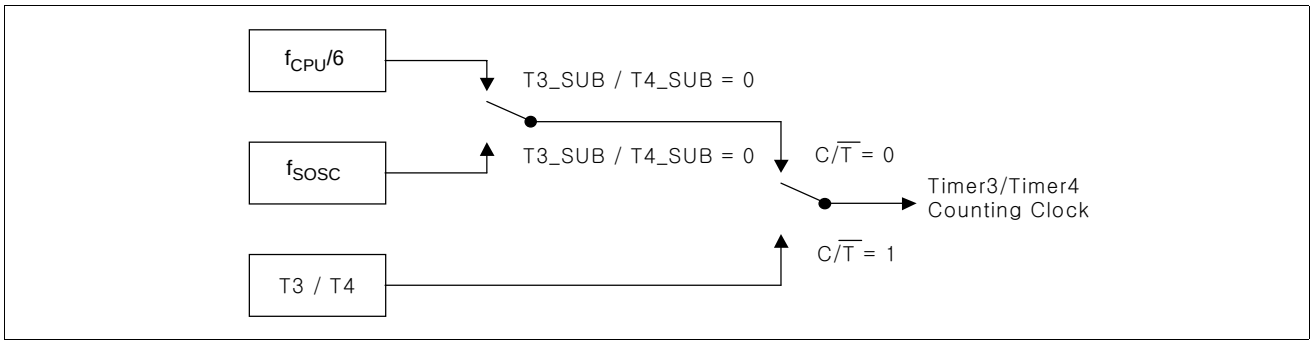
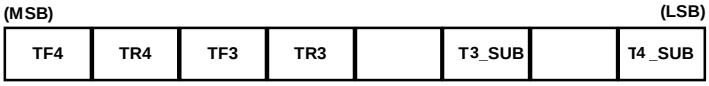
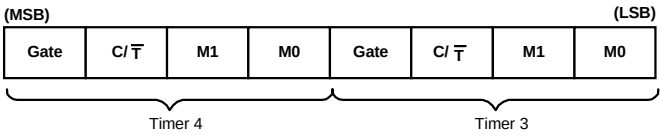


Figure 4-12 Clock Counting Sources fot Timer3/Counter3 and Timer4/Counter4



Symbol	Position	Name and Significance	Symbol	Position	Name and Significance
TF4	TCON.7	Timer 4 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.		TCON.3	
TR4	TCON.6	Timer 4 Run control bit. Set/cleared by software to turn Timer/Counter on/off.	T3_SUB	TCON.2	Switch main clock to sub clock for Timer 3 counting. This bit is a write-only register. 0 = Main Osc, 1 = Sub Osc.
TF3	TCON.5	Timer 3 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.		TCON.1	
TR3	TCON.4	Timer 3 Run control bit. Set/cleared by software to turn Timer/Counter on/off.	T4_SUB	TCON.0	Switch main clock to sub clock for Timer 4 counting. This bit is a write-only register. 0 = Main Osc, 1 = Sub Osc.

Figure 4-13 T34CON Register



		M1	M0	Operating
Gate	Gating Control when set. Timer/Counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set.	0	0	13-bit Timer/Counter
		0	1	16-bit Timer/Counter "THx" and "TLx" are cascaded : there is no prescaler.
C/$\overline{T}$	Timer or Counter Selector cleared for Timer operation (imput from internal system clock). Set for Counter operation (input from "Tx" input pin).	1	0	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
		1	1	(Timer 3) TL3 is an 8-bit Timer/Counter controlled by the standard Timer 3 control bits. TH3 is an 8-bit timer only controlled by Timer 4 control bits.
		1	1	(Timer 4) Timer/Counter 4 stopped.

Figure 4-14 FiT34MOD Register

4.5 Standard Serial Interface (UART)

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the register. (However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost.) The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received (LSB first). The baud rate is fixed at 1/6 the CPU clock frequency.

Mode 1: 10 bits are transmitted (through TxD) or received (through RxD) : a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted (through TxD) or received (through RxD) : start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in Special Function Register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/16 or 1/32 the CPU clock frequency.

Mode 3: 11 bits are transmitted (through TxD) or received (through RxD) : a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable.
In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1.

Multiprocessor Communications

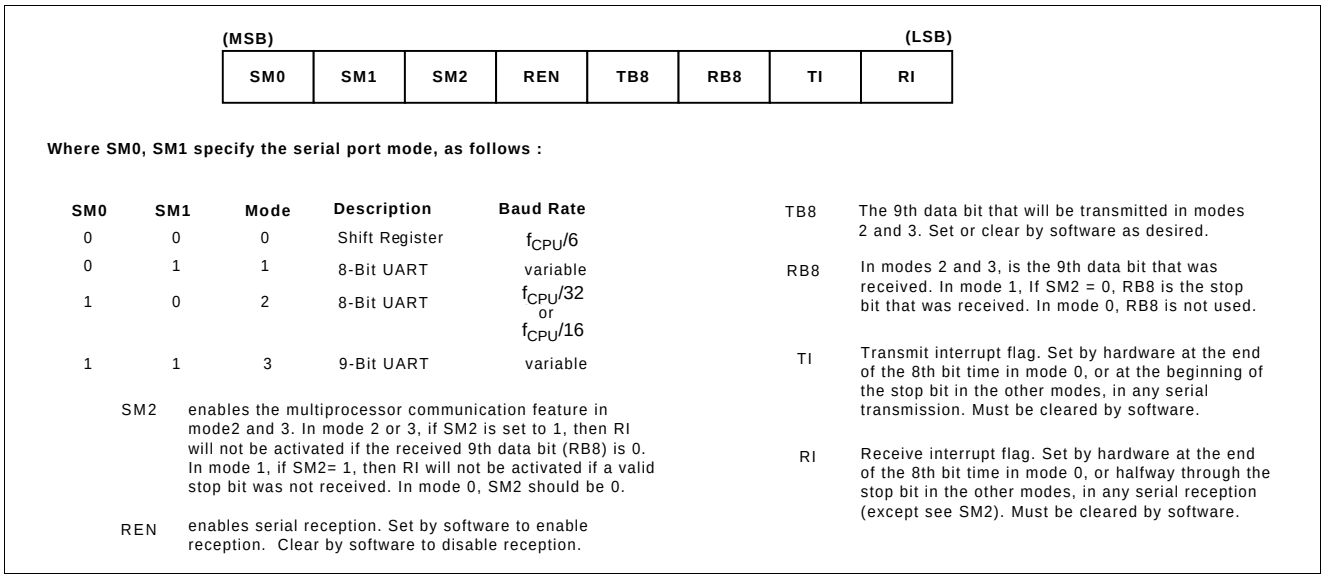
Modes 2 and 3 have a special provision for multiprocessor communications. In these modes, 9 data bits are received. The 9th one goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON. A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

Serial Port Control Register

The serial port control and status register is the Special Function Register SCON, shown in Figure 4-15. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).



(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows :

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	$f_{CPU}/6$
0	1	1	8-Bit UART	variable
1	0	2	8-Bit UART	$f_{CPU}/32$ or $f_{CPU}/16$
1	1	3	9-Bit UART	variable

SM2 enables the multiprocessor communication feature in mode2 and 3. In mode 2 or 3, if SM2 is set to 1, then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2= 1, then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.

REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.

TB8 The 9th data bit that will be transmitted in modes 2 and 3. Set or clear by software as desired.

RB8 In modes 2 and 3, is the 9th data bit that was received. In mode 1, If SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

TI Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.

RI Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit in the other modes, in any serial reception (except see SM2). Must be cleared by software.

Figure 4-15 Serial Port Control Register (SCON)

* f_{CPU} : CPU clock
* The f_{CPU} is shown in Figure 4-2

Baud Rates

The baud rate in Mode 0 is fixed:

$$Mode\ 0\ Baud\ Rate = f_{CPU}/6$$

The baud rate in Mode 2 depends on the value of bit SMOD = 0 (which is the value on reset), the baud rate is 1/32 the CPU clock frequency. If SMOD = 1, the baud rate is 1/16 the CPU clock frequency.

$$Mode\ 2\ Baud\ Rate = \frac{2^{SMOD}}{32} \times f_{CPU}$$

In the HMS9XC8032, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate.

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows

$$Mode\ 1,3\ Baud\ Rate = \frac{2^{SMOD}}{32} \times (Timer\ 1\ Overflow\ Rate)$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either "timer" or "counter" operation, and in any of its 3 running modes. In the most typical applications, it is configured for "timer" operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case the baud rate is given by the formula:

$$Mode\ 1,3\ Baud\ Rate = \frac{2^{SMOD}}{16} \times \frac{f_{CPU}}{12 \times [256 - (TH1)]}$$

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the Timer 1 interrupt to do a 16-bit software reload. Figure 12 lists various commonly used baud rates and how they can be obtained from Timer 1. be obtained from Timer 1.

Using Timer/Counter 2 to Generate Baud Rates

In the HMS9XC8032, Timer 2 selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (see Figure B-14 Timer/Counter 2 Control Register (T2CON)). Note that the baud rate for transmit and receive can be simultaneously different. Setting RCLK and/or TCLK puts Timer into its baud rate generator mode.

The baud rate generator mode is similar to the auto-reload mode, in that a roll over in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAL2H and RCAP2L, which are preset by software.

Now, the baud rates in Modes 1 and 3 are determined at Timer 2's overflow rate as follows:

$$Mode\ 1,3\ Baud\ Rate = \frac{Timer\ 2\ Overflow\ Rate}{16}$$

The timer can be configured for either "timer" or "counter" oper-

ation. In the most typical applications, it is configured for “timer” operation (C/T2 = 0). “Timer” operation is a little different for Timer 2 when it’s being used as a baud rate generator. Normally, as a timer it would increment every machine cycle (thus at the 1/6 the CPU clock frequency). In the case, the baud rate is given by the formula:

$$\text{Mode 1,3 Baud Rate} = \frac{f_{CPU}}{16 \times [65536 - (RCAP2H, RCAP2L)]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 also be used as the baud rate generating mode. This mode is valid only if RCLK + TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in “timer” function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the timer is being incremented every state time, and the results of a read or write may not be accurate. The RCAP registers may be read, but should not be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

More About Mode 0

Serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed a 1/6 the CPU clock frequency.

Figure 4-16 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal at S6P2 also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write to SBUF" and activation of SEND.

SEND enables the output of the shift register to the alternate output function line of RxD and also enable SHIFT CLOCK to the alternate output function line of TxD. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1, and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the
transmit shift are shifted to the right one position.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control block to do one last shift and then deactivate SEND and set T1. Both of these actions occur at S1P1. Both of these actions occur at S1P1 of the 10th machine cycle after "write to SBUF."

Reception is initiated by the condition REN = 1 and R1 = 0. At S6P2 of the next machine cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of TxD. SHIFT CLOCK makes transitions at S3P1 and S6P1 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the RxD pin at S5P2 of the same machine cycle.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared as RI is set.

More About Mode 1

Ten bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first). and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. In the HMS9XC8032 the baud rate is determined by the Timer 1 overflow rate.

Figure 4-17 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit receive.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF” signal.)

The transmission begins with activation of SEND which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeros are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs

at the 10th divide-by-16 rollover after "write to SBUF."

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the reset of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1. R1 = 0, and
2. Either SM2 = 0, or the received stop bit = 1.

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RxD.

More About Modes 2 and 3

Eleven bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the th data bit goes into RB8 in SCON. The baud rate is programmable to either 1/16 or 1/32 the CPU clock frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1.

Figure 4-18 and Figure 4-19 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write to SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "write to SBUF" signal.)

The transmission begins with activation of SEND, which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data bits shift out to the right, zeros are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate SEND and set TI. This occurs at the 11th divide-by 16 rollover after "write to SUBF."

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register.

At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of R-D. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI.

The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1. RI = 0, and
2. Either SM2 = 0, or the received 9th data bit = 1

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RxD input.

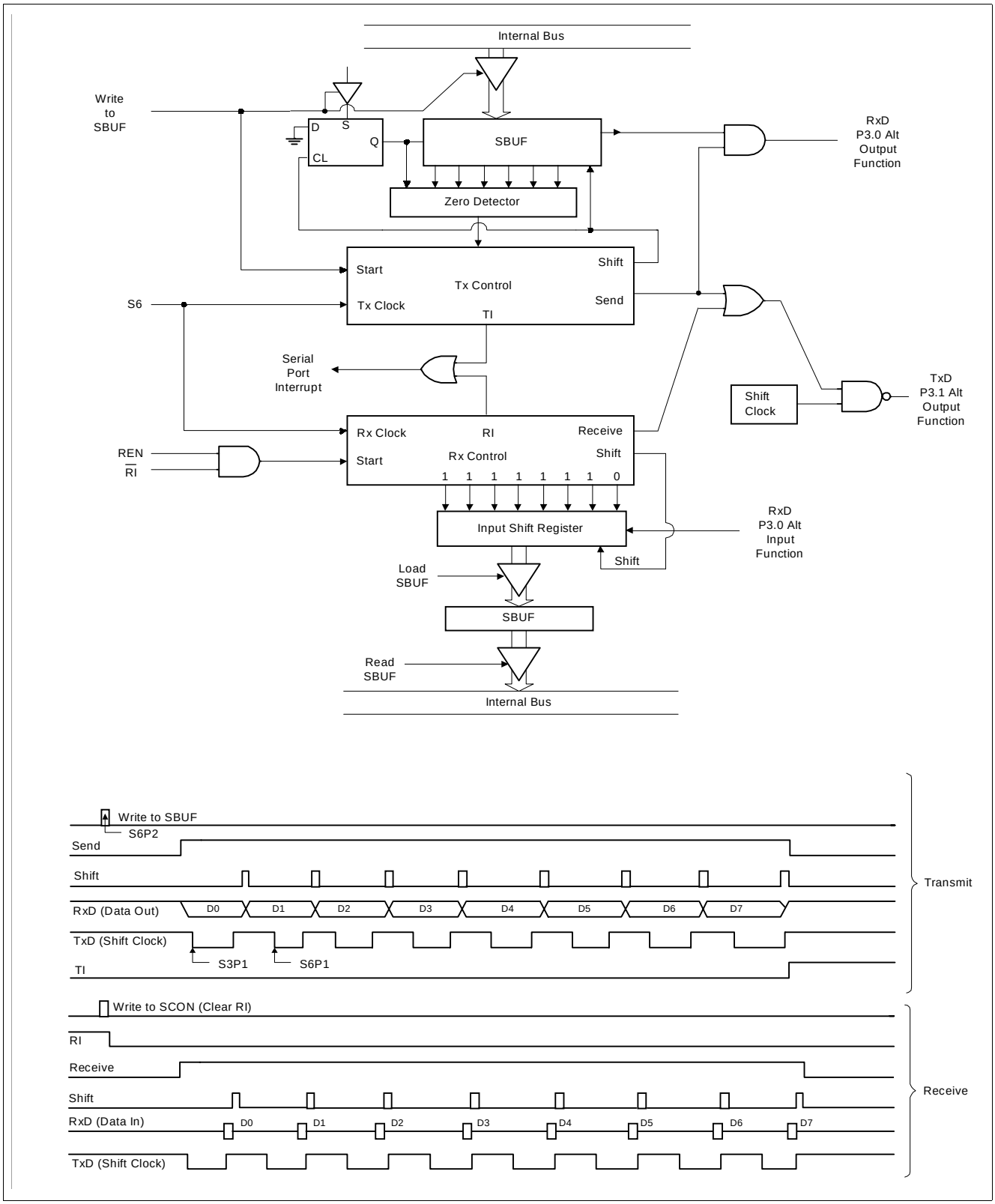


Figure 4-16 Serial Port Mode 0

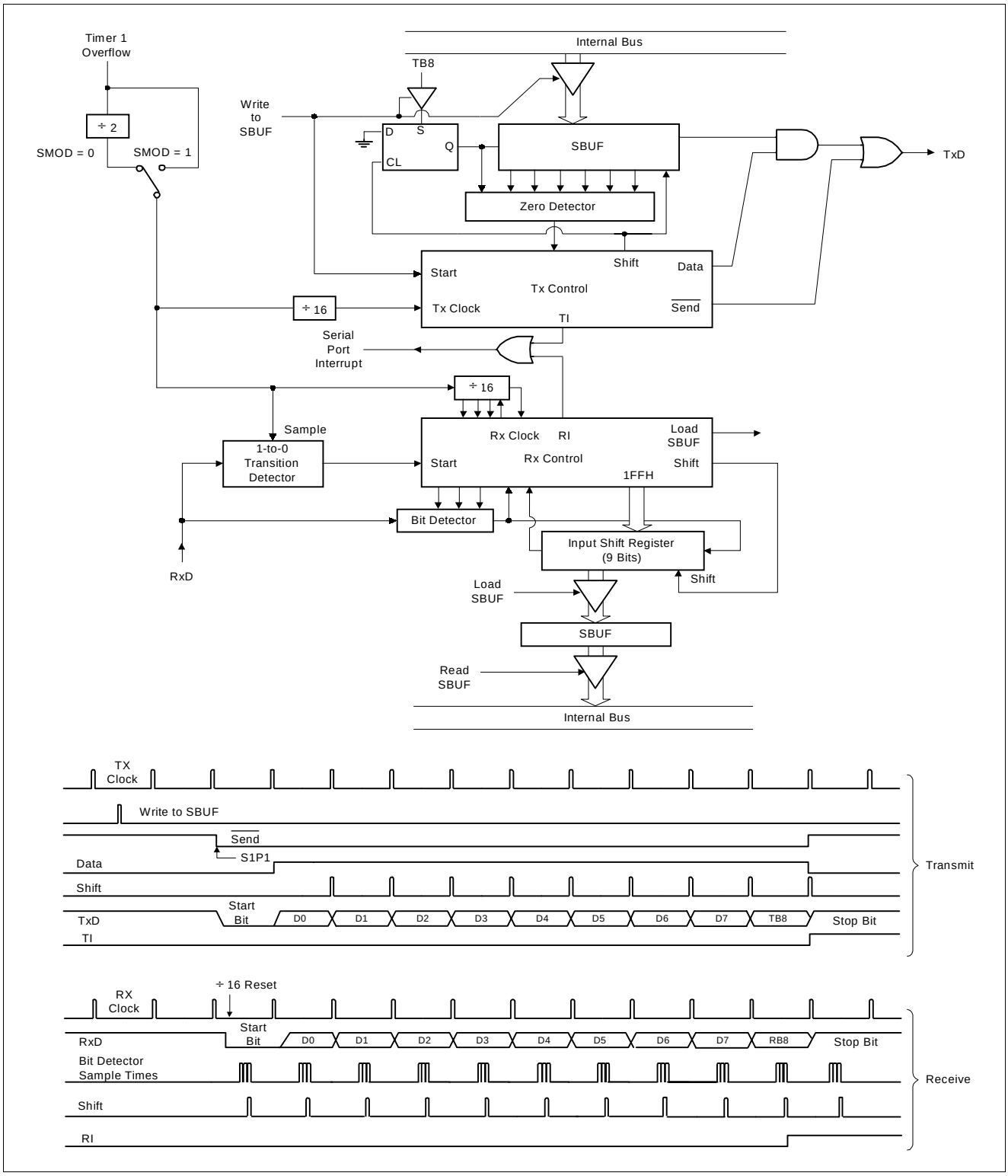


Figure 4-17 Serial Port Mode 1

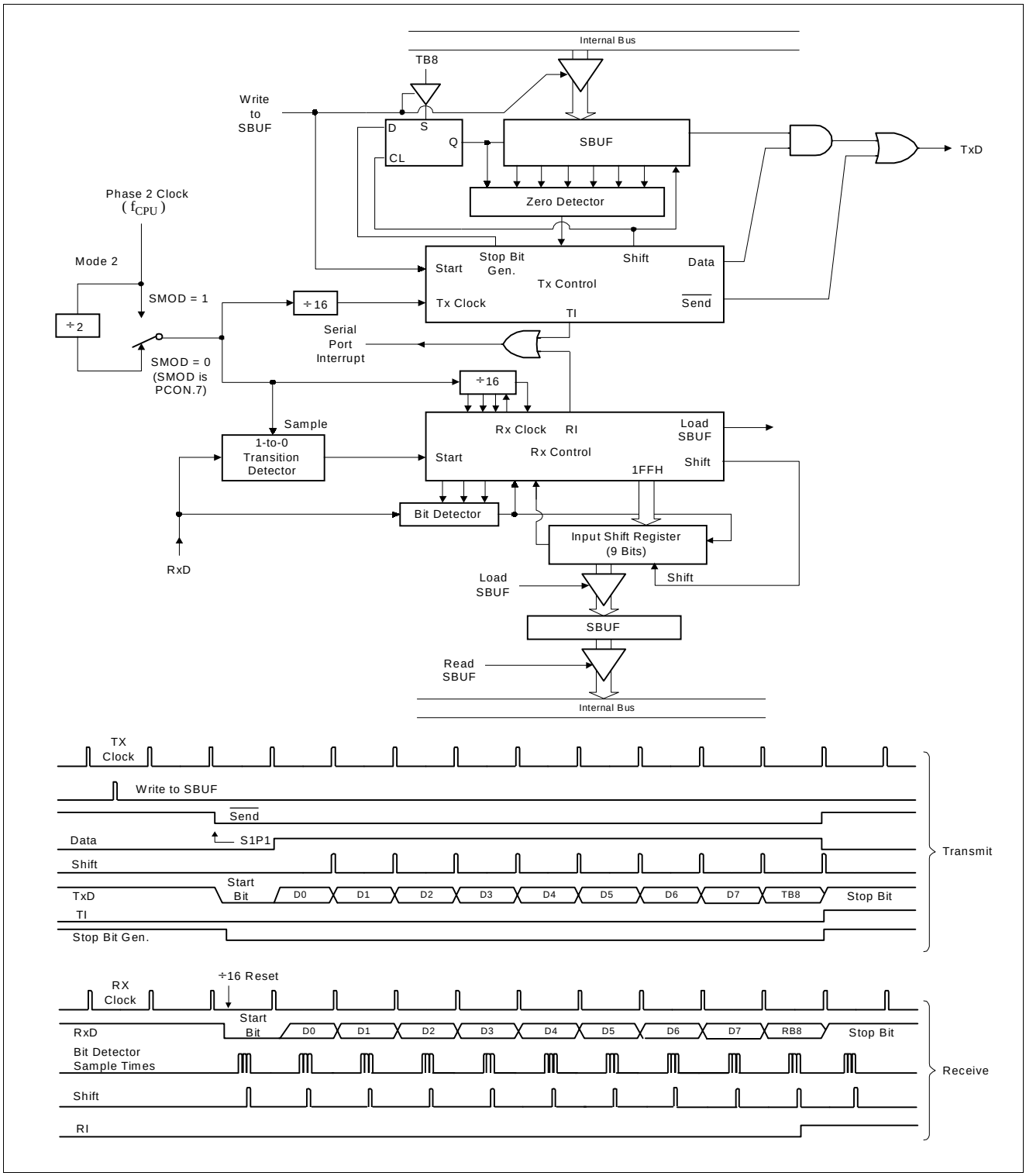


Figure 4-18 Serial Port Mode 2

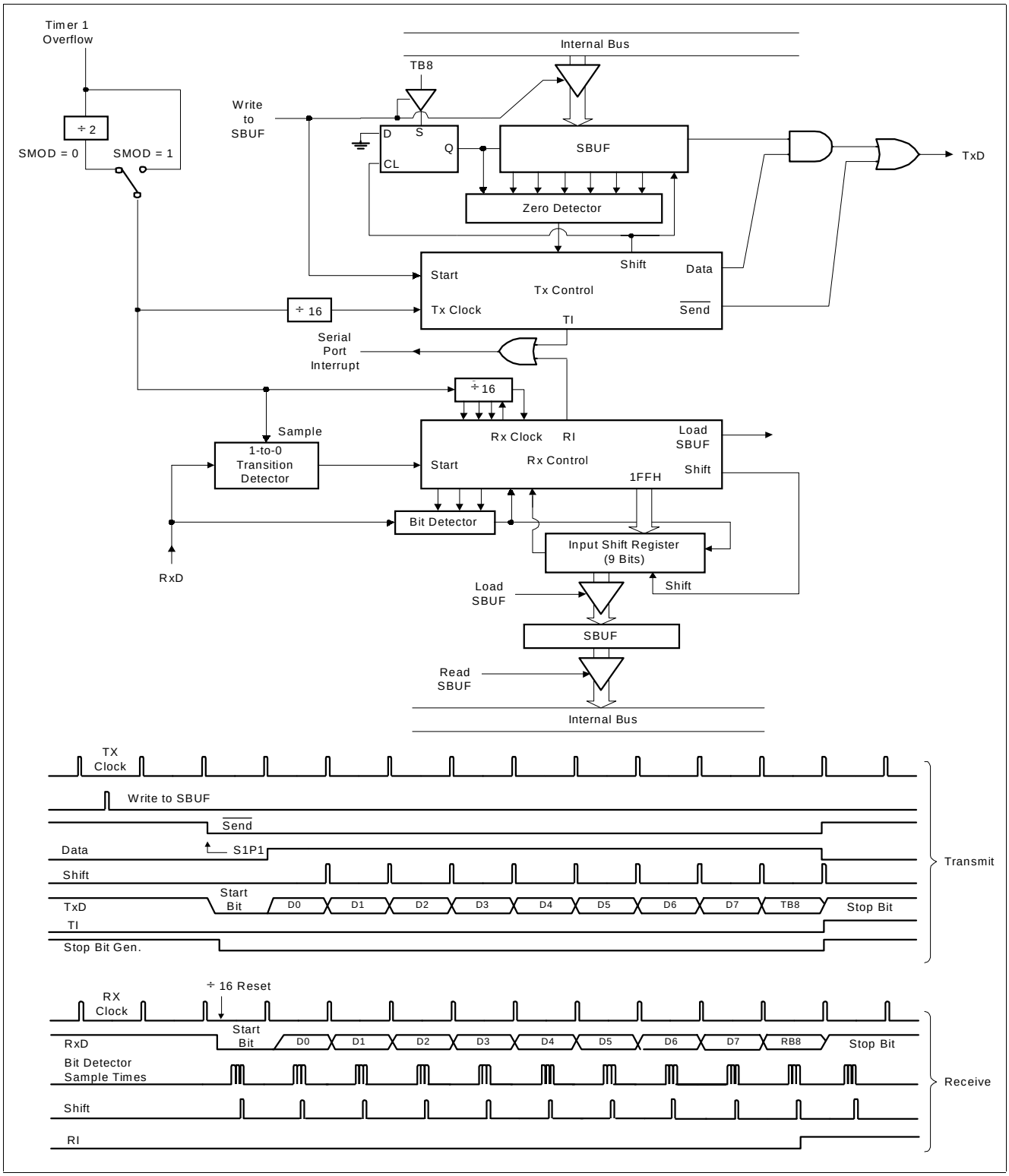


Figure 4-19 Serial Port Mode 3

4.6 Standard Serial Interface (SIO 1, SIO 2)

Configuration of Serial Interface

Figure 4-20 shows the block diagram of the SIO1 and SIO2.

As shown in Figure 4-20, the shift clock control section of the SIO is composed of a clock input/output pin block , clock generation block, wait control block, and clock count block. The serial data control section is composed of a serial data input/output pin block and SBUF1 and SBUF2. These blocks are controlled by the flags of the control register. Writing of data into and reading of data from the SBUF1 and SBUF2 are performed via the data buffer. The functions of each block are outlined in Outline of function of serial interface section

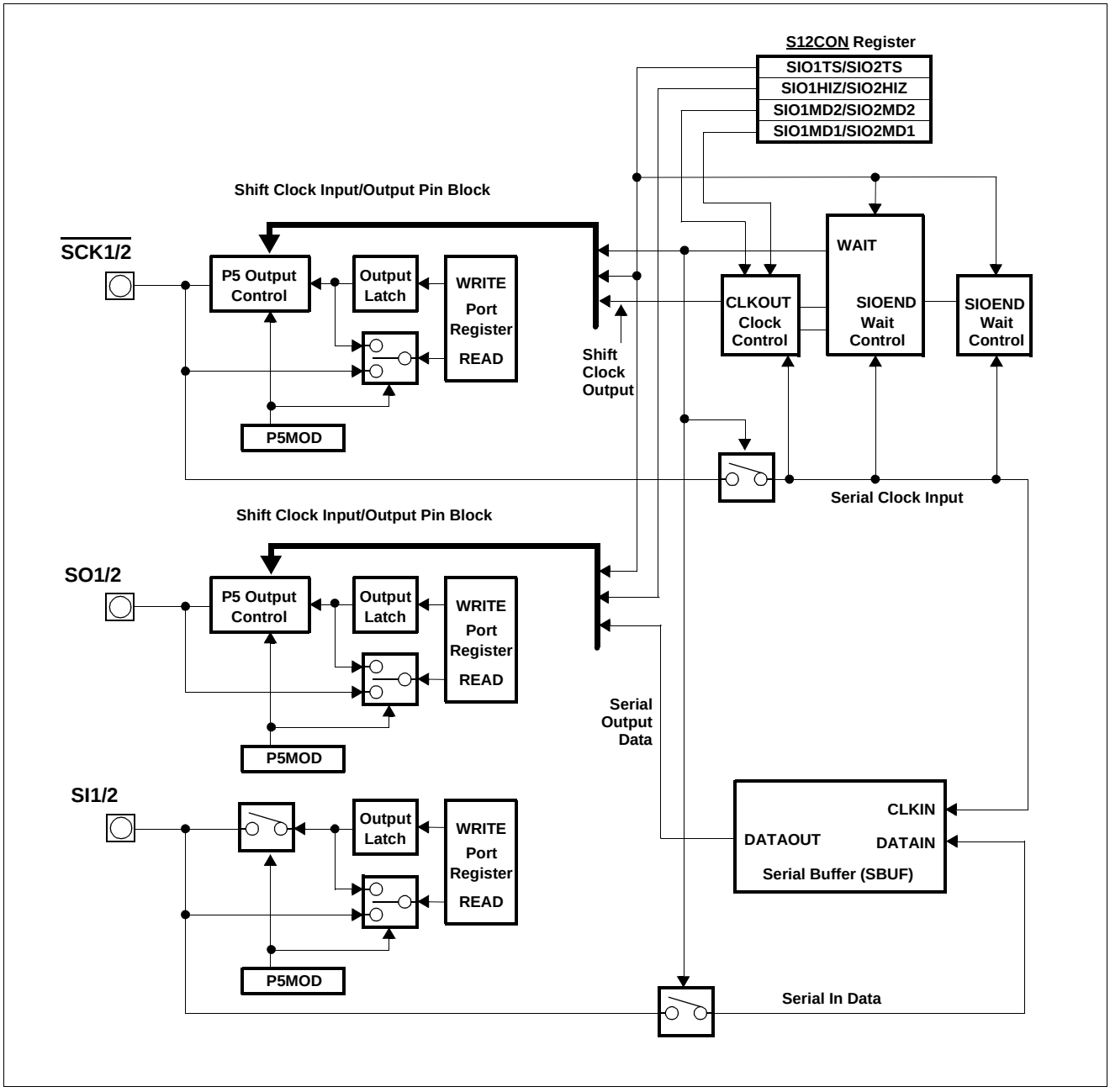


Figure 4-20 SIO Block Diagram

Outline of function of serial interface

The SIO1 and SIO2 permits use of 3-wire serial I/O system. The SIO1 and SIO2 uses SCK pin, SI pin and SO pin. The SIO1 and SIO2 permits selection of internal clock and external clock, and also permits selection of the reception and transmission operations. The following sections indicate the functions of blocks of the SIO1 and SIO2.

Shift clock input/output pin block

This block is used for selecting the shift clock input/ output pin. This selection of the shift clock input/output pin is performed by the serial I/O mode select register. See Shift clock and serial data input/output control block section.

Serial data input/output pin block

This block is used for selecting the shift data input/ output pin. This selection of the shift data input/output pin is performed by the serial I/O mode select register. See Shift clock and serial data input/output control block section.

Clock generation block

This block selects the clock frequency of the shift clock, and also controls the shift clock output timing. Selection of the clock frequency is performed by the serial I/O clock select register. See Clock Generation Block section.

Clock counter

The clock counter counts the number of the rising edges of the clocks output from the shift clock output pin, and issues signal at 8^{th} clock (SIOEND signal). The SIOEND signal is used to put the serial communication into a wait (pause). See Clock Counter section.

Serial Buffer (SBUF1 and SBUF2)

This is a shift register which sets the serial out data and stores the serial in data. This register performs shift operation to input or output data by the clock input of the shift clock input pin. Setting of the output data and reading of input data are performed via the data buffer. See Serial Buffer (SBUF1, SBUF2) section.

Wait control block

This block controls the wait (pause) and wait cancel (communication operation) of serial communication. Wait cancel of serial communication is performed by the serial I/O mode select register. See Wait Block section.

Shift clock and serial data input/output control block

The shift clock and serial data input/output control block controls the setting of pins and sending and receiving operation related to the SIO1 and SIO2. These are controlled by the serial I/O mode select register. The configuration and function of the serial I/O mode select register are explained in *Configuration and function of serial I/O mode select register* section. The setting status of each pin by the serial I/O mode select register is explained in *Setting of Each pin by serial I/O mode select register* section.

Configuration and function of serial I/O mode select register

The configuration and function of the serial I/O mode select register are explained below. SIO1CK1 and SIO2CK0 flags select between internal clock and external clock, and also set the frequency of internal clock. For the clock, see Clock Generation Block Section. SIO2TS flag sets the wait and wait cancel state of the SIO1 and SIO2. For the wait operation, see Wait Block section.

S12CON: SIO1 & SIO2 CONTROL REGISTER. BIT ADDRESSABLE. : A0H

SIO2TS	SIO2HIZ	SIO2CK1	SIO2CK0	SIO1TS	SIO1HIZ	SIO1CK1	SIO1CK0

SIO2TS	S12CON.7	Software START/STOP control for SIO2. A logic 1 starts the SIO2
SIO2HIZ	S12CON.6	Software Port control for SIO2. A logic 1 assigns general I/O port to SIO2 port
SIO2CK1	S12CON.5	See Table 5-24
SIO2CK0	S12CON.4	See Table 5-24
SIO1TS	S12CON.3	Software START/STOP control for SIO1. A logic 1 starts the SIO1
SIO1HIZ	S12CON.2	Software Port control for SIO1. A logic 1 assigns general I/O port to SIO1 port
SIO1CK1	S12CON.1	See Table 5-24
SIO1CK0	S12CON.0	See Table 5-24

SIO1CK1/SIO2CK1	SIO1CK0/SIO2CK0	Set input/output clock frequency of SIO1/SIO2 (f_{SC})
0	0	Slave mode : External clock
0	1	Master mode : 75KHz (f_{XX} / 48)
1	0	Master mode : 150KHz (f_{XX} / 24)
1	1	Master mode : 450KHz (f_{XX} / 8)

Table 4-25 SIO1 and SIO2 Control Register

Setting of Each pin by serial I/O mode select register

The setting of each pin also requires handling of the input/output setting flags. When using SO pin as serial out pin, SO pin must be set as the output port by the port5 mode select register (P5MOD). Similarly, SI pin must be set as input port. When using the external clock, SCK pin must be set as the general purpose input port. It must be set as output port when using the internal clock.

Clock Generation Block

The clock generation block controls the clock generation and clock output timing when the internal clock is used (master operation mode). The internal clock frequency $\mathbf{f_{SC}}$ is set by SIO2CK1 and SIO2CK0 flags of the serial I/O mode select register. The shift clock is output until the value of the clock counter, to be mentioned later, reaches “8”. *Internal shift clock generation timing* section shows the clock output waveform and generation timing.

Internal shift clock generation timing

(1) Wait cancel from initialization state

The initialization state indicated the state where the internal clock operation mode is selected and “high” level is output to SCK pin which is set as output pin. During the wait state, “High” level is output to the shift clock pin.

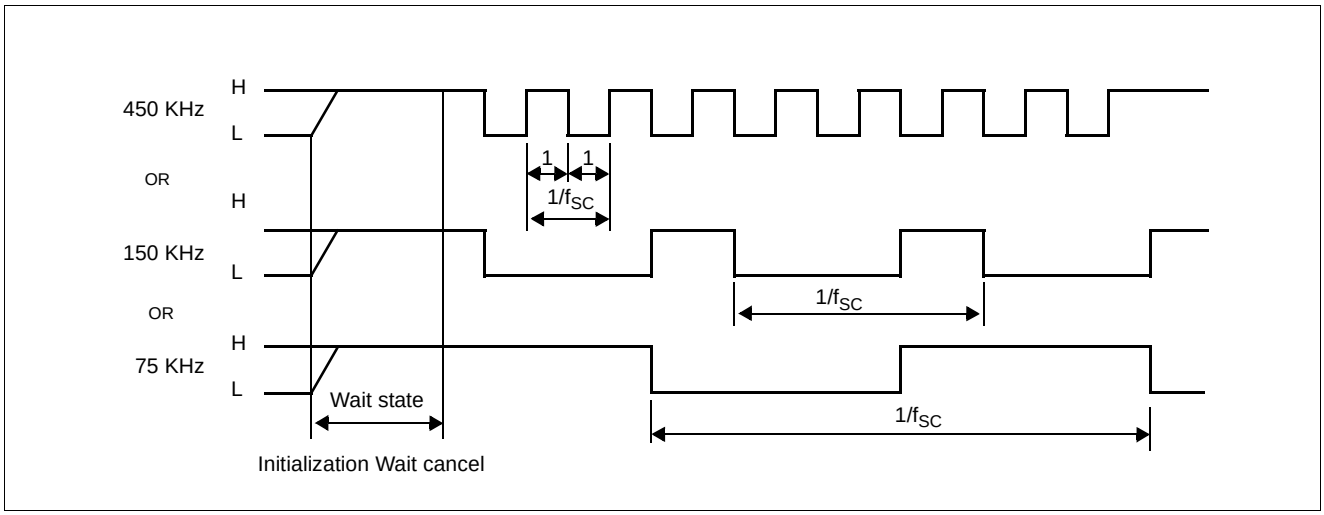
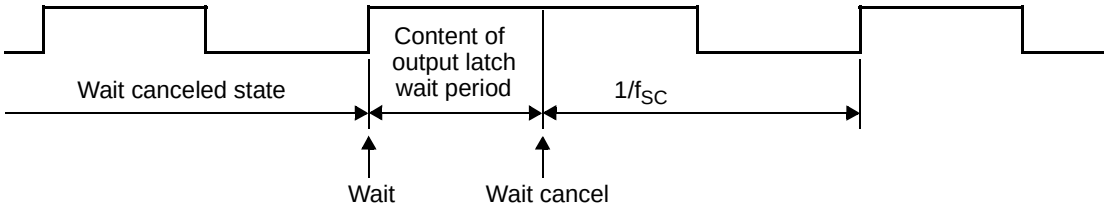


Figure 4-21 SIO Clock (f_{SC})

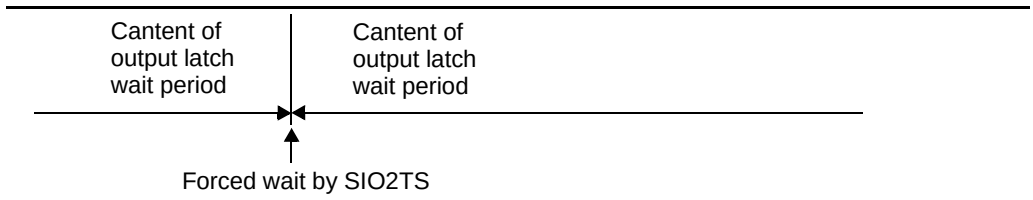
(2)When wait operation is performed

For the details of wait operation, see Wait Block section 21.19.

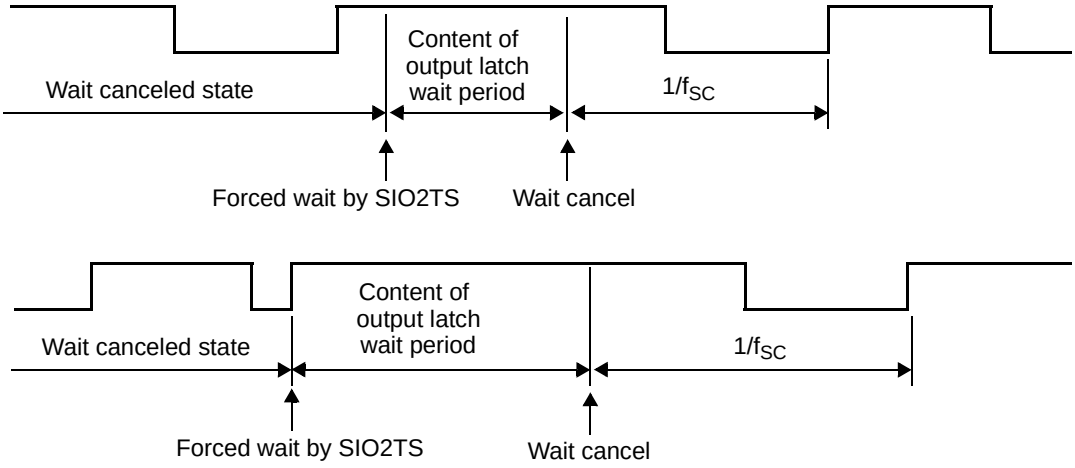
(a) Ordinary wait with clock counter reached “8”



(b)Forced wait during a wait



(c) Forced wait during wait canceled state



(D) Wait cancel during wait canceled state

No change occurs in the clock output waveform. The clock counter is not reset.

(e) When clock frequency change and wait cancel are effected at the same time.

The setting of clock frequency and cancellation of wait are performed by the register of the same address, and cancellation of wait (setting of SIO2TS flag) and changing of the clock frequency can be performed by single instruction. If wait cancellation and clock frequency change are performed at the same time, the same state is resulted as the wait cancel state from the initialization state mentioned in item (1) above.

Clock Counter

The operation of clock counter is shown in Figure 4-22. The

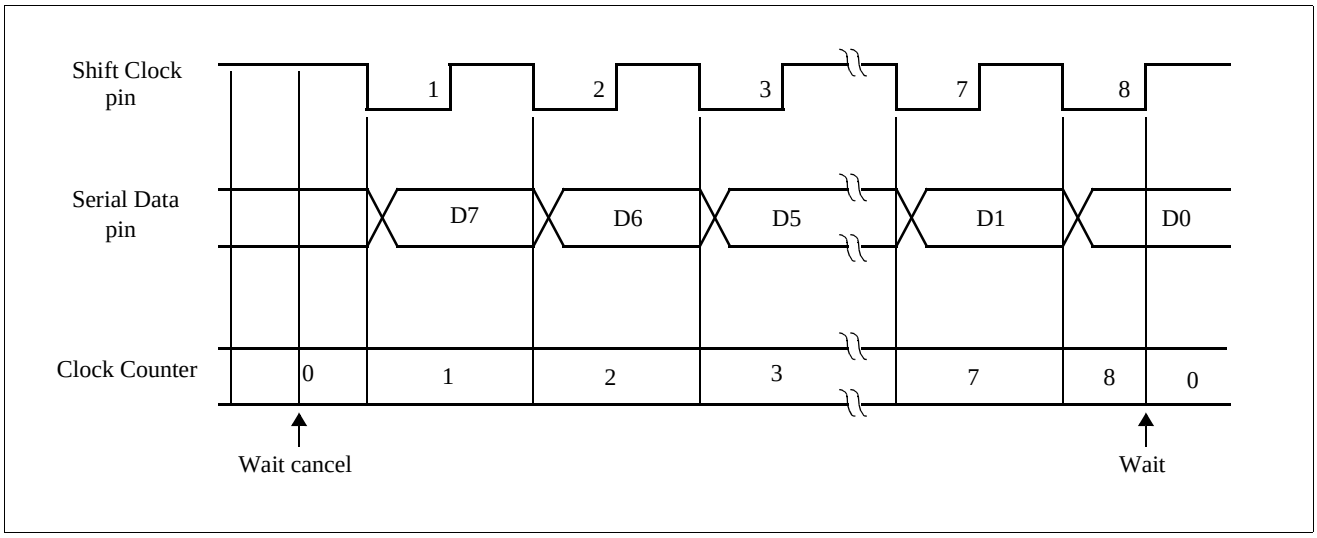


Figure 4-22 Clock Counter Operation

initial value of the clock counter is 0, and counter value increments (+1) upon each detection of the falling edge of the clock pin waveform. When counted up to 8, the counter is reset to 0 at the rising edge of next shift clock. The serial communication is put to wait state at the time the clock counter is reset to 0.

Clock Counter Reset 0 Condition

The clock counter resetting conditions are listed below:

(1) Power ON

(2) Writing of 0 into SIO2TS flag

(3) Rising of shift clock when wait is canceled and clock counter is 9.

Serial Buffer (SBUF1, SBUF2)

The serial buffer (SBUF1 and SBUF2) is an 8-bit shift register which is used to set the serial out data and read the serial in data. Setting (writing) of data to and reading of data from the serial buffer are performed respectively by MOV instruction. The data shift operation of the serial buffer is performed in synchronization with the clock applied to the shift clock pin (SCK pin). The content of the most significant bit of the serial buffer is output to serial data pin in synchronization with the falling edge of the shift clock. The data of the serial data pin is read into the least significant bit of the serial buffer in synchronization with the rising edge of the clock waveform.

Operation of Serial buffer section shows the operation and precautions concerning this shift register. *Precautions in Data setting and Data reading* Section shows precautions concerning data writing into and data reading from the serial buffer. During the wait state, the serial buffer does not perform data shift operation.

Operation of Serial buffer

The operation is shown below.

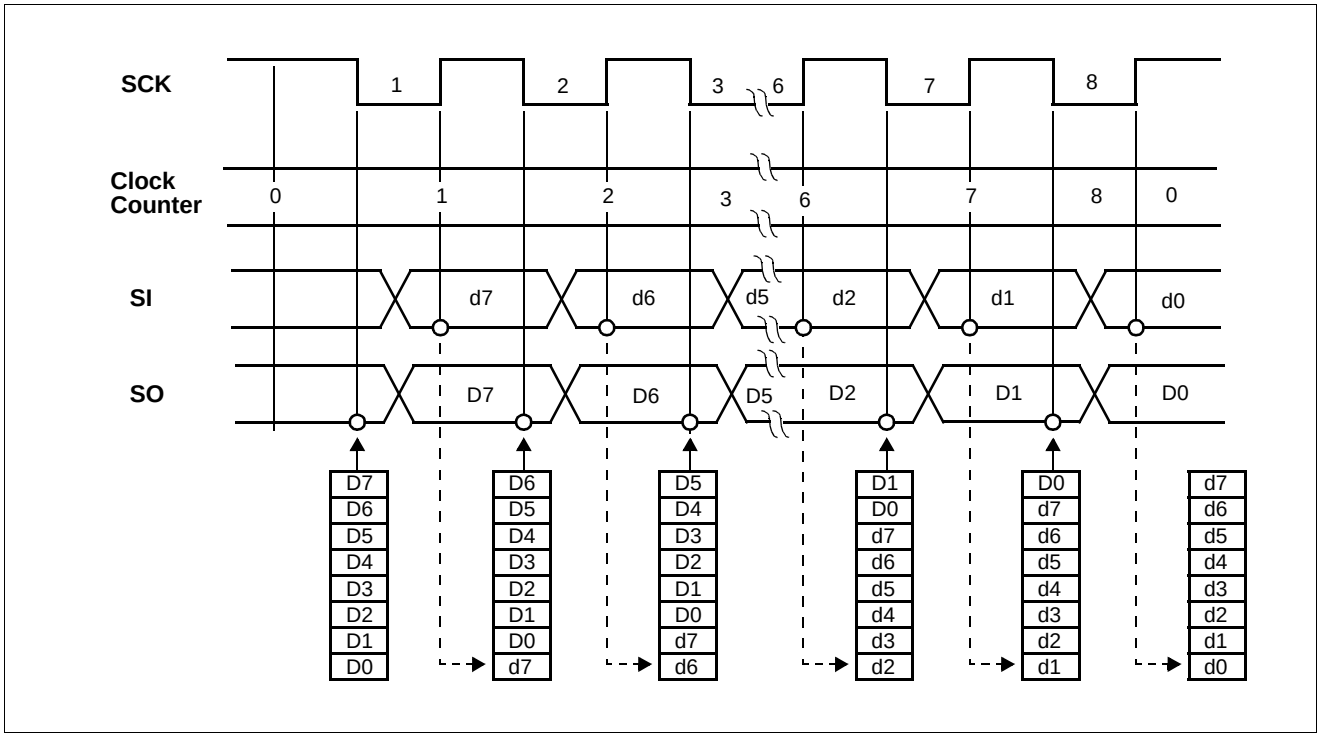


Figure 4-23 SIO1 and SIO2 Timing Diragram

Data shift operation of Serial buffer

Serial I/O system	
Serial input operation	**Serial output operation**
The status of SI is entered by shifting from LSB at eth rising edge of shift clock pin wave form. If the SI pin is set as input port, the content of output latch is entered.	The data is output to SO pin by shifting from MSB at the falling edge of shift clock pin waveform. If the SO pin is set as input port, if if SIO2HIZ flag is 0, then no serial output is provide.

Table 4-26

Precautions in Data Setting and Data reading

Data writing into the serial buffer is performed by MOV instruction. Reading of data is performed by MOV instruction. Data setting and data reading must be performed while the wait status exists. During the wait cancel, data setting and data carrying may fail depending on the status of the shift clock pin.

Wait Block

The wait block controls pause (wait) and cancel of communication of the SIO1 and SIO2. This control is performed by the SIO2TS flag. *Wait Operation and Precautions* section shows the wait operation and precautions.

Wait Operation and Precautions

The wait state means a state when the clock generation block, serial buffer, etc. stop their operation, and the serial communication is suspended. When the wait state if canceled, serial communication operation is started. Wait state is canceled by writing 1 into SIO2TS flag. When 1 is written into the SIO2TS flag, the internal clock is output to the shift clock output pin (master operation mode), and the serial buffer and clock counter start operation. When the clock counter is 8 and shift clock rises, the wait cancel state turns into the wait state. In this case, the SIO2TS flag is reset (0) automatically. The operation status of serial communication can be known by detecting the content of SIO2 TS flag while the wait is canceled. After starting the serial communication by writing 1 to SIO2TS flag, the data can be read or set by detecting the

SIO2TS flag turning to 0. This means that correct data setting and reading may fail if data setting or data reading is executed to the serial buffer during the wait canceled state. See *Precautions in data setting and data reading* section. Writing of 0 to SIO2TS flag during the wait cancel state causes the wait state to be established. This is called as “forced wait”.

An example of wait operation is shown below.

When wait is canceled, the serial data is output at the falling edge of the next clock, and the flag turns into the wait canceled state. When eight shift clock pulses are entered, the value of the output latch (usually high level) is output from the shift clock pin, and this causes the operation of the clock counter and serial buffer to be stopped. Note that correct data will not be set if data writing to and data reading from the serial buffer are attempted while the wait is in the canceled state and the shift clock pin is at high level. If data is written into the serial buffer while the wait is in the canceled state and the shift clock pin is at low level, the content of MSB will be output to the serial data output pin at the time when MOV instruction is executed. If forced wait is effected during the wait canceled state, a wait state is resumed upon writing of 0 into SIO2TS flag.

Usage of SIO1 and SIO2

Figure 4-25 and Figure 4-26 shows the input/output blocks and communication method of the SIO. As shown in Figure 4-25 and Figure 4-26, there are internal clock operation mode and external clock operation mode, and each mode permits transmission and reception. Master and slave operation modes are selected by SIOxCK1 and SIOxCK0 flags. Reception and transmission are set according to the pins used. In the master operation mode, SCK pin outputs internal clock. In this case, however, the SCK pin must be set as output port. In the slave operation mode, SCK pin is set in the floating state for receiving external clock. In this case, however, the SCK pin must be set as input port. Serial data is output from SO pin at the falling edge of the shift clock irrespective of the internal clock or external clock. In this case, however, SO pin must be set as output port, and SIO2HIZ flag be set. Serial data is input to the serial buffer as the status of SI pin at the rising edge of the shift clock irrespective of the internal clock or external clock. SCK pin reads the current status of output latch during a wait , or reads the status of the current pin during a wait cancel. SO pin reads the current status of output latch.

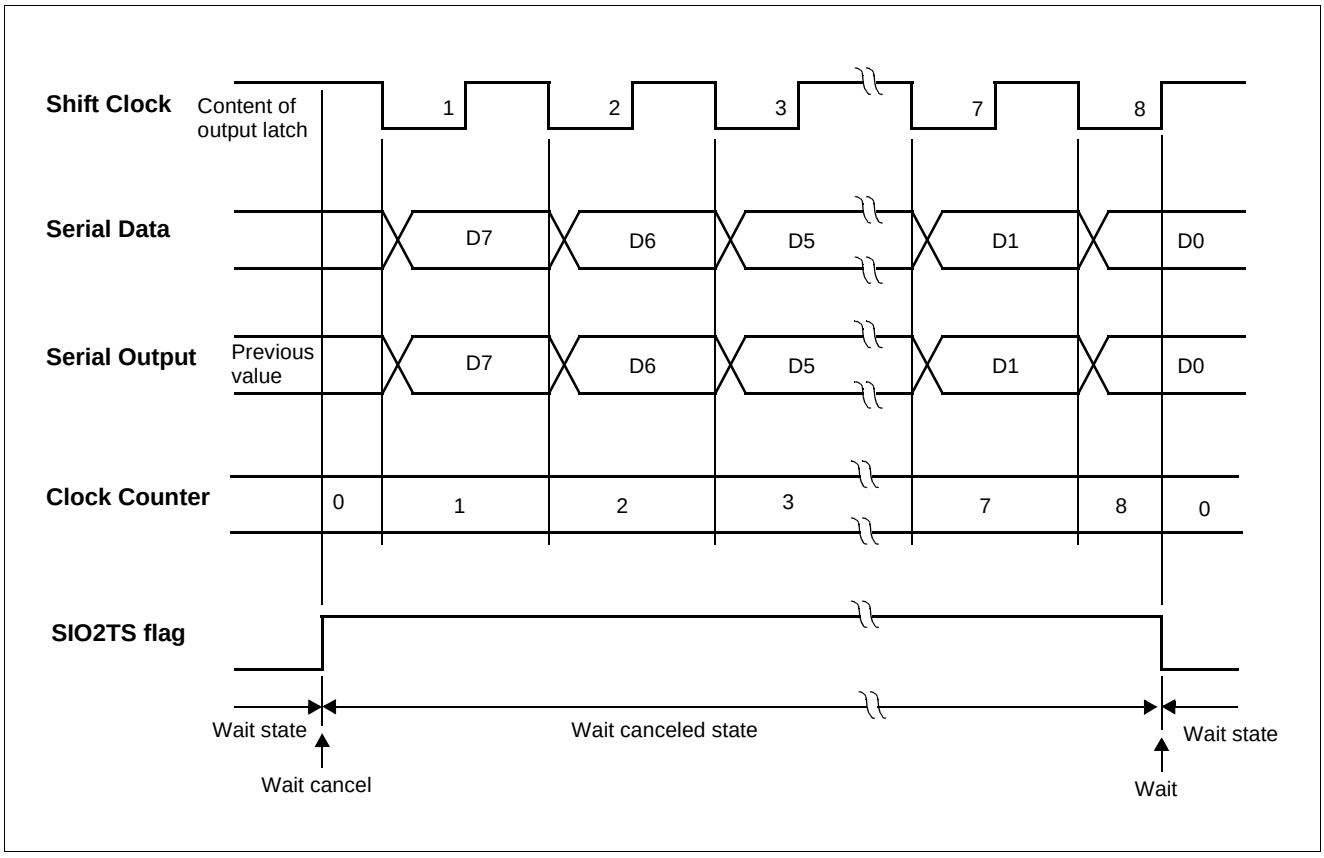


Figure 4-24 Example of Wait Operation

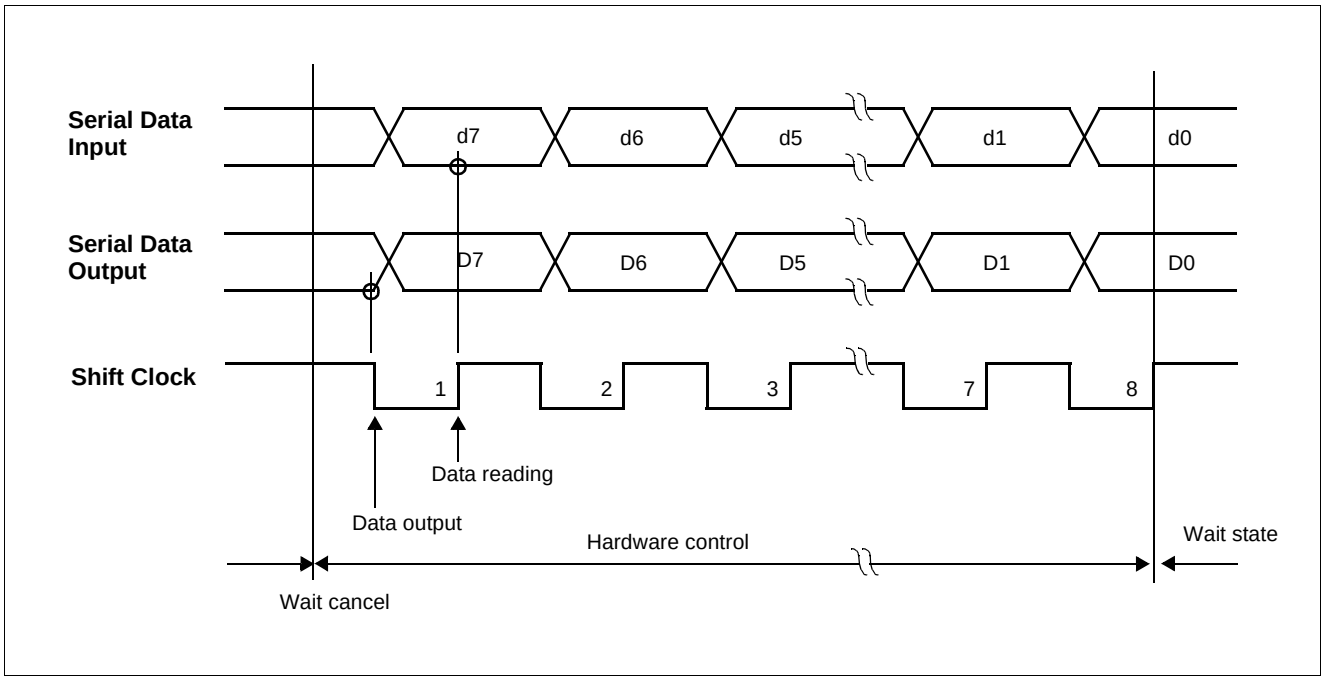


Figure 4-25 Input/Ouitput Block of the SIO and Communication Method

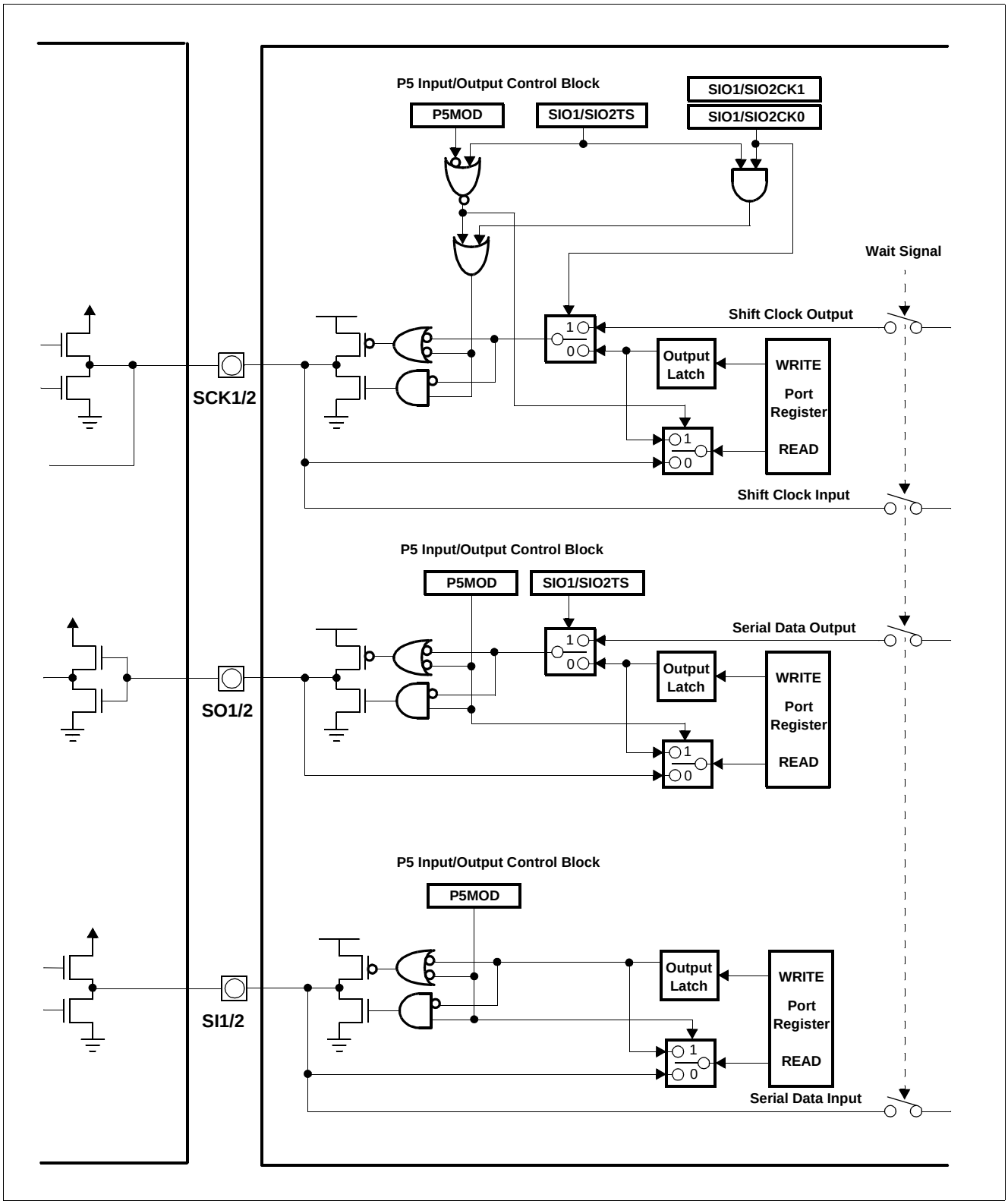


Figure 4-26 Operation of Each Mode of the SIO

4.7 Port Structure and Operation

Ports 0 to 7

The direction of each port is controlled by the value of PXMOD register and On/Off control of pull-up transistor in ports except P2.0, P2.1, P2.2 and P2.3 is selected by the content of PXCON register. P0DATA, P1DATA, P2DATA, P3DATA, P4DATA, P5DATA, P6DATA and P7DATA are the SFR latches of Ports 0, 1, 2, 3, 4, 5, 6 and 7, respectively. Writing a one to a bit of a port SFR causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low. When used as an input, the external state of a port pin will be held in the port SFR (i.e., if the external state of a pin is low, the corresponding port SFR bit will contain a 0, if it is high, the bit will contain a 1).

All eight ports in the HMS9XC8032 are bi-directional.

All the Port 3, Port 4, Port 5 and Port 7 pins are multifunctional. They are not only port pins, but also serve the functions of various special features as listed below:

Port Pin	Alternate Function
P3.0	T0 (Timer/Counter 0 External Input)
P3.1	T1 (Timer/Counter 1 External Input)
P3.2	T2 (Timer/Counter 2 External Input)
P3.3	T3 (Timer/Counter 3 External Input)
P3.4	T4 (Timer/Counter 4 External Input)
P3.5	T2EX (Timer/Counter 2 Capture/Reload Trigger)
P4.0	/INT0 (External Interrupt 0)
P4.1	/INT1 (External Interrupt 1)
P4.2	/INT2 (External Interrupt 2)
P4.3	/INT3 (External Interrupt 3)
P4.4	/INT4 (External Interrupt 4)
P4.5	/INT5 (External Interrupt 5)
P4.6	/INT6 (External Interrupt 6)
P4.7	Beeper Output
P5.0	TxD (serial output port)
P5.1	RxD (serial input port)
P5.2	SCK1 (SIO1 clock port)
P5.3	SO1 (SIO1 output port)
P5.4	SI1 (SIO1 input port)
P5.5	SCK2 (SIO2 clock port)
P5.6	SO2 (SIO2 output port)
P5.7	SI2 (SIO2 input port)
P7.0	ANI0 (Analog input channel 0 for ADC)
P7.1	ANI1 (Analog input channel 1 for ADC)
P7.2	ANI2 (Analog input channel 2 for ADC)
P7.3	ANI3 (Analog input channel 3 for ADC)
P7.4	ANI4 (Analog input channel 4 for ADC)
P7.5	ANI5 (Analog input channel 5 for ADC)
P7.6	ANI6 (Analog input channel 6 for ADC)
P7.7	ANI7 (Analog input channel 7 for ADC)

I/O Configurations

Figure 4-27 and Figure 4-28 shows a simplified functional diagram in each of the ports. The bit latch (one bit in the port's SFR) is represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read pin" signal from the CPU. Some instructions that read a port activate the "read latch" signal, and others activate the "read pin" signal.

All ports have internal pullups controlled by the user software, except Port2 low nibble. Port2.0 - Port2.3 have open drain outputs. Each I/O line can be independently used as an input or an output.

All the port latches in the HMS9XC8032 have 1s written to them by the reset function. If a 0 is subsequently written to a port latch, it can be reconfigured as an input by writing a 1 to it.

Writing to a Port

Users, who want to use port as output, must set PXMOD register as output. When a port specify as output mode, attempt to read from the port will not be guaranteed.

In the execution of an instruction that changes the value in a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction.

Consequently, the new value in the port latch won't actually appear at the output pin until the next Phase 1, which will be at S1P1 of the next machine cycle.

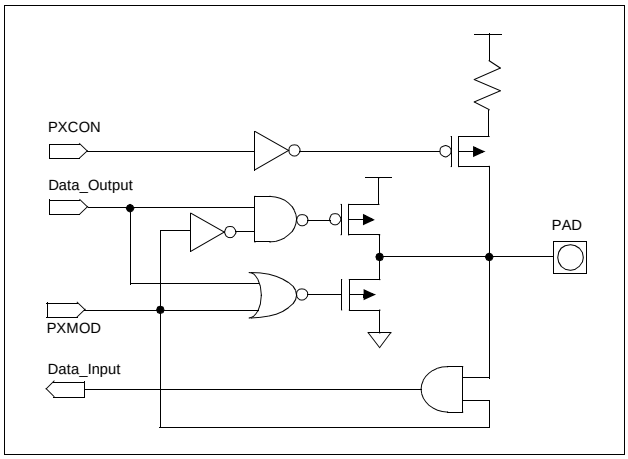


Figure 4-27 P0 ~ P7 Ports Schematic (Except P2.0, P2.1, P2.2 and P2.3)

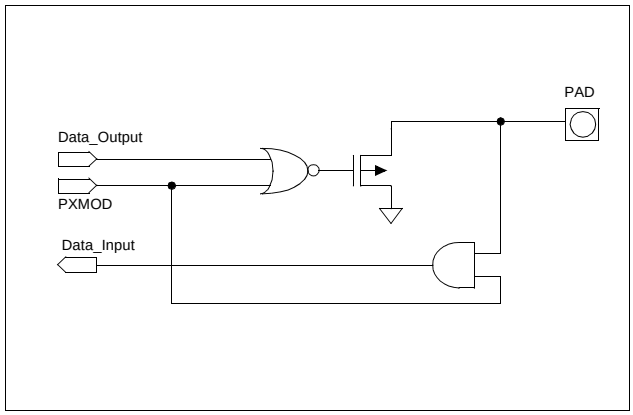


Figure 4-28 P2.0, P2.1, P2.2 and P2.3 Ports Schematic

Read-Modify-Write Feature

Some instructions that read a port read the latch and others read the pin. Which ones do latch and others read the pin. The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and rewrite it to the latch. These are called "read -modify-write" instructions. The instructions listed below are read-modify-write instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch.

The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the based of a transistor. When a 1 is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as a 0. Reading the latch rather than the pin will return the correct value of 1.

```
ANL             (logical AND, e.g., ANL P1, A)
ORL             (logical OR, e.g., ORL P2, A)
XRL             (logical EX-OR, e.g., XRL P3, A)
JBC             (jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)
CPL             (complement bit, e.g., CPL P3.0)
INC             (increment, e.g., INC P2)
DEC             (decrement, e.g., DEC P2)
DJNZ            (decrement and jump if not zero, e.g., DJNZ P3, LABEL)
MOV   PX.Y,C    (move carry bit to bit Y of Port X)
CLR   PX.Y      (clear bit Y of Port X)
SET   PX.Y      (set bit Y of Port X)
```

4.8 Watch Dog Timer

Watchdog Timer Functions

The watchdog timer has the following functions.

- **Non-maskable watchdog timer interrupt**
- **Maskable watchdog timer interrupt**

Watchdog Timer Configuration

The watchdog timer consists of the following hardware.

WDTCON: BEEPER & WATCHDOG TIMER CONTROL REGISTER. BIT ADDRESSABLE. : F8H

RUNBEEP	*BEEPMD1*	*BEEPMD0*	**RUNWDT**	**WDTMK**	**WDTMD2**	**WDTMD1**	**WDTMD0**

RUNWDT	WDTCON.4	Restart watchdog timer (This bit is automatically cleared to “0” after restart.).
WDTMK	WDTCON.3	Software Enable/Disable NMI (Non Maskable Interrupt) for WDT. A logic 1 makes WDT interrupt NMI
WDTMD2	WDTCON.2	See Table 4-27
WDTMD1	WDTCON.1	See Table 4-27
WDTMD0	WDTCON.0	See Table 4-27

WDTMD[2:0]			Selects of WDT input
0	0	0	f_{XX}
0	0	1	f_{XX} / 2^3
0	1	0	f_{XX} / 2^4
0	1	1	f_{XX} / 2^5
1	0	0	f_{XX} / 2^7
1	0	1	f_{XX} / 2^9
1	1	0	f_{XX} / 2^11
1	1	1	fxx / 2^13

Table 4-27 Selects of WDT

* The f_{XX} is shown in Figure 4-2 on page 18

WDTDR: WATCHDOG TIMER DATA REGISTER. : F9H

WDTDR7	**WDTDR6**	**WDTDR5**	**WDTDR4**	**WDTDR3**	**WDTDR2**	**WDTDR1**	**WDTDR0**

* WDTDR0 ~ 7 is counting value of the watchdog timer.

Watchdog Timer Operations

When WDTRUN flag is set to 1, the 8-bit watchdog timer begins to increment with the selected watchdog timer counting clock. The initial value of this 8-bit counter is determined by WDTDR register.

Watchdog timer interrupt

If the counter continues to increment and overflow is generated, the watchdog timer interrupt occurs. The types of the watchdog

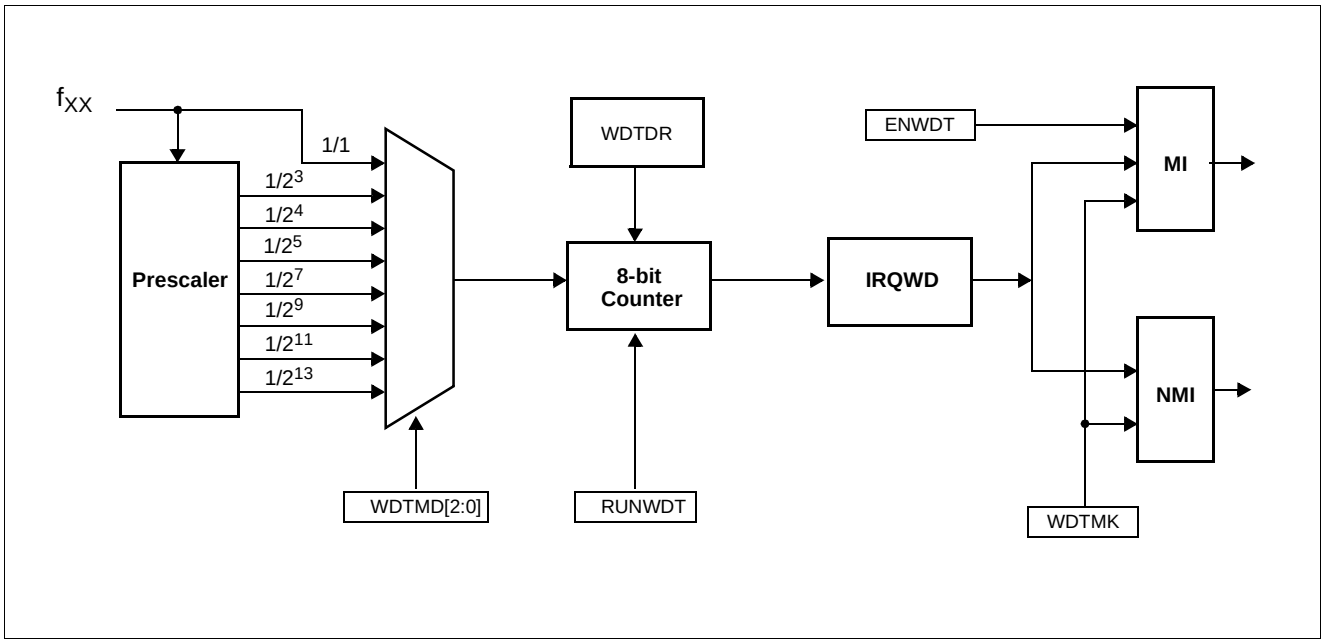


Figure 4-29 Watchdog Timer Block Dragram

timer interrupt(Maskable Interrupt or Non Maskable Interrupt) are selected by WDTMK flag. If maskable interrupt is selected by WDTMK flag, the watchdog timer interrupt can be disabled by IEWDT flag of the IE3 register. Refer Figure 4-29.

Watchdog timer restart

After the watchdog timer starts, resetting RUNWDT flag to 1 makes the 8-bit watchdog counter restart from the initial value determined by WDTDR register. Once the watchdog timer starts, setting RUNWDT flag to 1 does not stop the watchdog timer.

The watchdog timer continues operating in the IDLE mode but it stops in the Power Down mode.

WDTMD[2:0]	Inadvertent Program Loop Detection Time
000	OSC (0.139us)
001	OSC / 2^3 (1.1us)
010	OSC / 2^4 (2.2us)
011	OSC / 2^5 (4.4us)
100	OSC / 2^7 (17.8us)
101	OSC / 2^9 (71.1us)
110	OSC / 2^11 (284us)
111	OSC / 2^13 (1138us)

Table 4-28 Watchdog Timer Inadvertent Program Loop Detection Times

NOTE: OSC : System clock frequency

4.9 Buzzer

Buzzer Output Control Circuit Functions

The buzzer output control circuit outputs 1.2KHz, 2.4KHz, 4.5KHz, 8KHz frequency square waves. The buzzer frequency selected with the watchdog timer register(WDTCON) is output from the P4.7/BEEP pin.

Follow the procedure below to output the buzzer frequency.

(1) Select the buzzer output grequency with bits 5 to 7 of WDTCON.

(2) Set the P4.7 output latch to 1.

(3) Set the P4.7 port mode register to output mode.

Buzzer Output Control Circuit Configuration

The buzzer output control circuit consists of the following hardware.

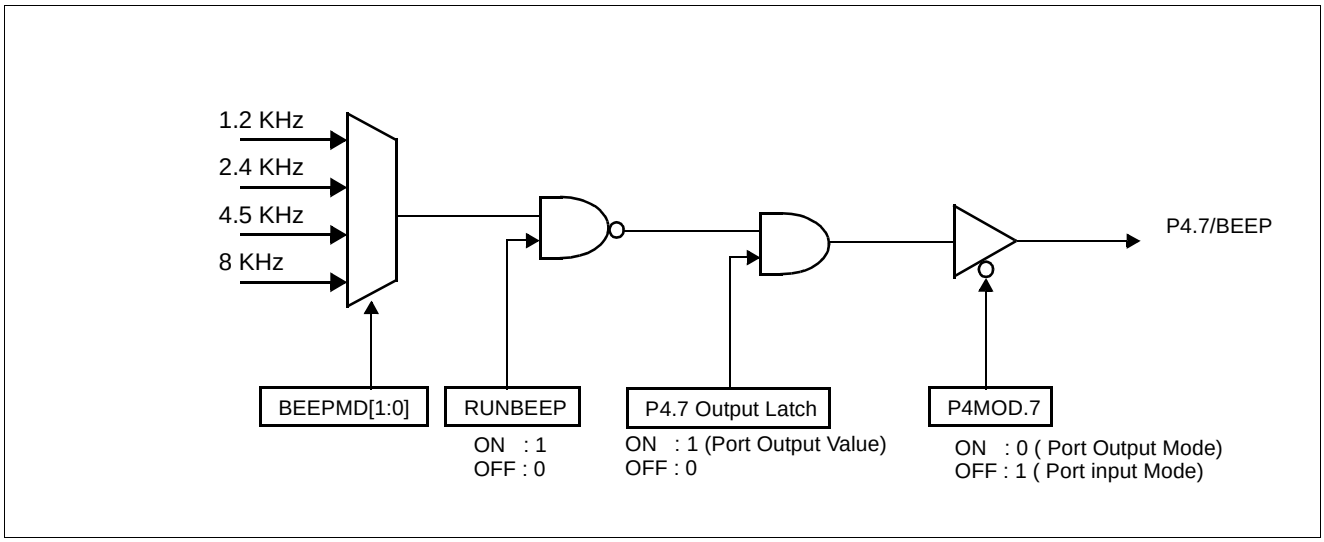


Figure 4-30 Buzzer Output Control Circuit Block Diagram

WDTCON: BEEPER & WATCHDOG TIMER CONTROL REGISTER. BIT ADDRESSABLE. : F8H

RUNBEEP	BEEPMD1	BEEPMD0	*RUNWDT*	*WDTMK*	*WDTMD2*	*WDTMD1*	*WDTMD0*

RUNBEEP	WDTCON.7	Software START/STOP control for Beeper. A logic 1 starts the Beeper.
BEEPMD1	WDTCON.6	See Table 4-29
BEEPMD0	WDTCON.5	See Table 4-29

BEEPMD[1:0]		Select Beeper Clock Frequency (f_{OSC} = 7.2 MHz)
0	0	1.2KHz (f_{OSC} / 6000)
0	1	2.4KHz (f_{OSC} / 3000)
1	0	4.5KHz (f_{OSC} / 1600)
1	1	8KHz (f_{OSC} / 900)

Table 4-29 Select Beeper Clock

Buzzer Control Register

The following two types of registers are used to control the buzzer output function.

Watchdog timer mode register (WDTCON)

Port mode register 4 (P4MOD)

(1) Watchdog timer mode register (WDTCON)

This register sets the buzzer output frequency.

NOTE: Besides setting the buzzer output frequency, WDTCON sets the watchdog timer count clock.

Watchdog TimerMode Register Format.

4.10 IF Counter

Function of Frequency Counter

The frequency counter counts the intermediate frequency (IF) of a tuner. It counts the intermediate frequency input to the FMIFC or AMIFC pin for a specific time (8ms, 32ms, 128ms or soft) with a 19-bit counter. The count value of the frequency counter is stored to the IF counter register. Figure 4-31 shows a block diagram of IF counter.

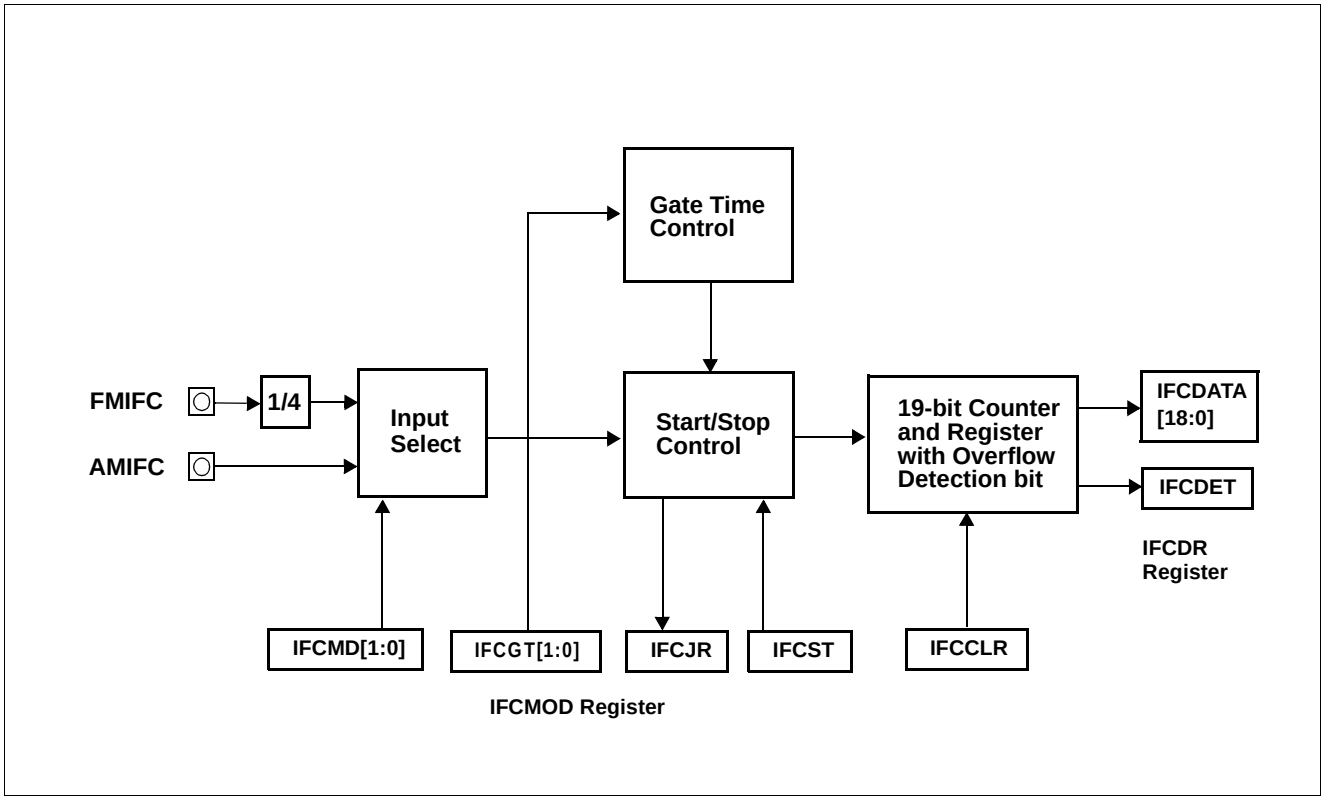


Figure 4-31 Frequency Counter Block Diagram

(1) Input select block

Input select block selects one of counter modes. Refer to IF Counter Control Register section for the details.

(2) Gate time control block

The gate time control block sets a gate time (count time).

(3) Start/stop control block

The start/stop control block starts IF counter data register counting and detects the end of counting.

(4) IF counter register block

The IF counter register block is a 19-bit register that counts up the input frequency during the set gate time. The counted value is stored to the IF counter register (IFC). The value of this register is reset to 00000H at reset. When the count value reaches 3FFFH, the overflow detection bit in IFCDR2 is set. The value of overflow detection bit is cleared by reset or writing 1 to IFCCLR.

IF Counter Control Register

The frequency counter is controlled by the following three registers.

IF counter mode register (IFCMOD)

IF counter data register (IFCDR2, IFCDR1, IFCDR0)

IFCMOD: IF counter mode register. : F4H

IFCJR	IFCST	IFCCLR	-	IFCGT1	IFCGT0	IFCMD1	IFCMD0

IFCJR	IFCMOD.7	IF counter judge register. Set by hardware automatically when IF counting is ended, Cleared by hardware automatically when software reads IFCMOD register.
IFCST	IFCMOD.6	Software START/STOP control for IF counter. A logic 1 starts the IF counter.
IFCCLR	IFCMOD.5	A logic 1 resets the IF counter data registers.
	IFCMOD.4	Reserved for future use *
IFCGT1	IFCMOD.3	See Table 4-30
IFCGT0	IFCMOD.2	See Table 4-30
IFCMD1	IFCMOD.1	See Table 4-31
IFCMD0	IFCMOD.0	See Table 4-31

IFCGT[1:0]		Setting of IFC gate time	
0	0	8ms	$1/(f_{XX}/28800)$
0	1	32ms	$1/(f_{XX}/115200)$
1	0	128ms	$1/(f_{XX}/460800)$
1	1	Soft *	

Table 4-30 IFC gate time

IFCMD[1:0]		Selects of IFC input
0	X	Disable FMIFC & AMIFC pins
1	0	FMIFC pin select
1	1	AMIFC pin select

Table 4-31 Selects of IFC inpu

* Software controls IFC gate time. IF counts during IFCST flag is high.

IF Counter Data Register

IF counter data registers (IFCDR2, IFCDR1 and IFCDR0) are read only registers. Attempt to write these registers is not allowed. These registers are valid when counting operation of IF counter is terminated normally.

IFCDR2: IF counter data register 2. : F5H

-	-	-	-	IFCDET	IFCDATA18	IFCDATA17	IFCDATA16

-	IFCDR2.7	Reserved for future use
-	IFCDR2.6	Reserved for future use
-	IFCDR2.5	Reserved for future use
-	IFCDR2.4	Reserved for future use
IFCDET	IFCDR2.3	Detection bit of 19bit IF counter overflow. A logic 1 implies the overflow of IF counter. It can be reset by IFCCLR. (See IF Counter Control Register
IFCDATA18	IFCDR2.2	19^{th} bit of 19bit IF counter (MSB)
IFCDATA17	IFCDR2.1	18^{th} bit of 19bit IF counter
IFCDATA16	IFCDR2.0	17^{th} bit of 19bit IF counter

IFCDR1: IF counter data register 1. : F6H

IFCDATA15	IFCDATA14	IFCDATA13	IFCDATA12	IFCDATA11	IFCDATA10	IFCDATA9	IFCDATA8

IFCDATA15	IFCDR1.7	16th bit of 19bit IF counter
IFCDATA14	IFCDR1.6	15th bit of 19bit IF counter
IFCDATA13	IFCDR1.5	14th bit of 19bit IF counter
IFCDATA12	IFCDR1.4	13th bit of 19bit IF counter
IFCDATA11	IFCDR1.3	12th bit of 19bit IF counter
IFCDATA10	IFCDR1.2	11th bit of 19bit IF counter
IFCDATA9	IFCDR1.1	10th bit of 19bit IF counter
IFCDATA8	IFCDR1.0	9th bit of 19bit IF counter

IFCDR0: IF counter data register 0. : F7H

IFCDATA7	IFCDATA6	IFCDATA5	IFCDATA4	IFCDATA3	IFCDATA2	IFCDATA1	IFCDATA0

IFCDATA7	IFCDR0.7	8th bit of 19bit IF counter
IFCDATA6	IFCDR0.6	7th bit of 19bit IF counter
IFCDATA5	IFCDR0.5	6th bit of 19bit IF counter
IFCDATA4	IFCDR0.4	5th bit of 19bit IF counter
IFCDATA3	IFCDR0.3	4th bit of 19bit IF counter
IFCDATA2	IFCDR0.2	3rd bit of 19bit IF counter
IFCDATA1	IFCDR0.1	2nd bit of 19bit IF counter
IFCDATA0	IFCDR0.0	1st bit of 19bit IF counter (LSB)

* User software should not write 1s to reserved bits. These bits may be used in future HMS9XC8032 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

Operation of Frequency Counter

(1) Select an input pin, mode and gate time using the IF counter mode register. Figure 4-32 shows a block that selects an input pin and mode.

(2) Set IFCCLR bit of the IF counter mode register to 1, and clears the data of the IF counter register.

(3) Set IFCST of the IF counter mode register to 1.

(4) The gate is opened only for the set gate time since 1KHz internal signal has risen after IFCST was set. If the gate time is set to be opened, the gate is opened as soon as it has been specified to be opened. IFCJR of the IF counter gate judge register is automatically set to 1 as soon as IFCST has been set to 1. When the gate time has expired, IFCJR bit of the IF counter gate judge register is automatically cleared to 0. If it is specified that the gate be open, however, IFCJR is not automatically cleared. In this case, set a gate time. Figure 4-33 shows the gate timing of the frequency counter.

(5) While the gate opens, the IF counter register counts the input frequency of the selected AMIFC or FMIFC pin. If the FMIFC pin is used in the FMIF count mode, however the input frequency is divided by quarter before if is counted.

The relationship between count value N (decimal), input frequencies, and gate time is shown below.

(1) FMIF count mode (FMIFC pin)

$$F_{FMIFC} = N / T_{GATE} \times 4 \text{ (KHz)}$$

N : FMIF Count Register Value

Example) FMIFC : 10.7 MHz

Gate Time : 32 ms

$N = (F_{FMIFC} / 4) \times T_{GATE} = (10.7MHz/4) \times 32ms$

= 85600 (decimal) = 14E60H (hexadecimal)

(2) AMIF count mode (AMIFC pin)

$F_{AMIFC} = N / T_{GATE}$ (KHz)

N : AMIF Count Register Value

Example) AMIFC : 450 MHz

Gate Time : 32 ms

$N = F_{AMIFC} \times T_{GATE} = 450\ KHz \times 32ms$

= 14400 (decimal) = 3840H (hexadecimal)

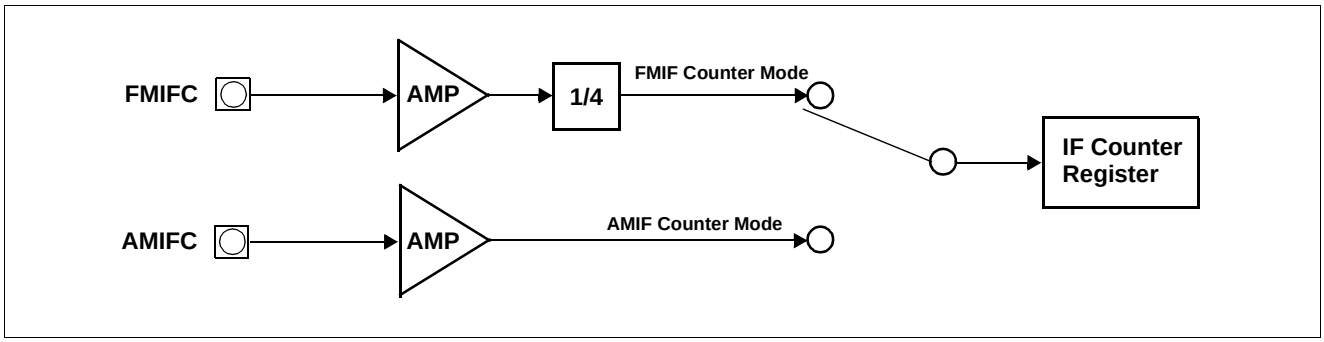


Figure 4-32 Input Pin and Mode Selection Block Diagram

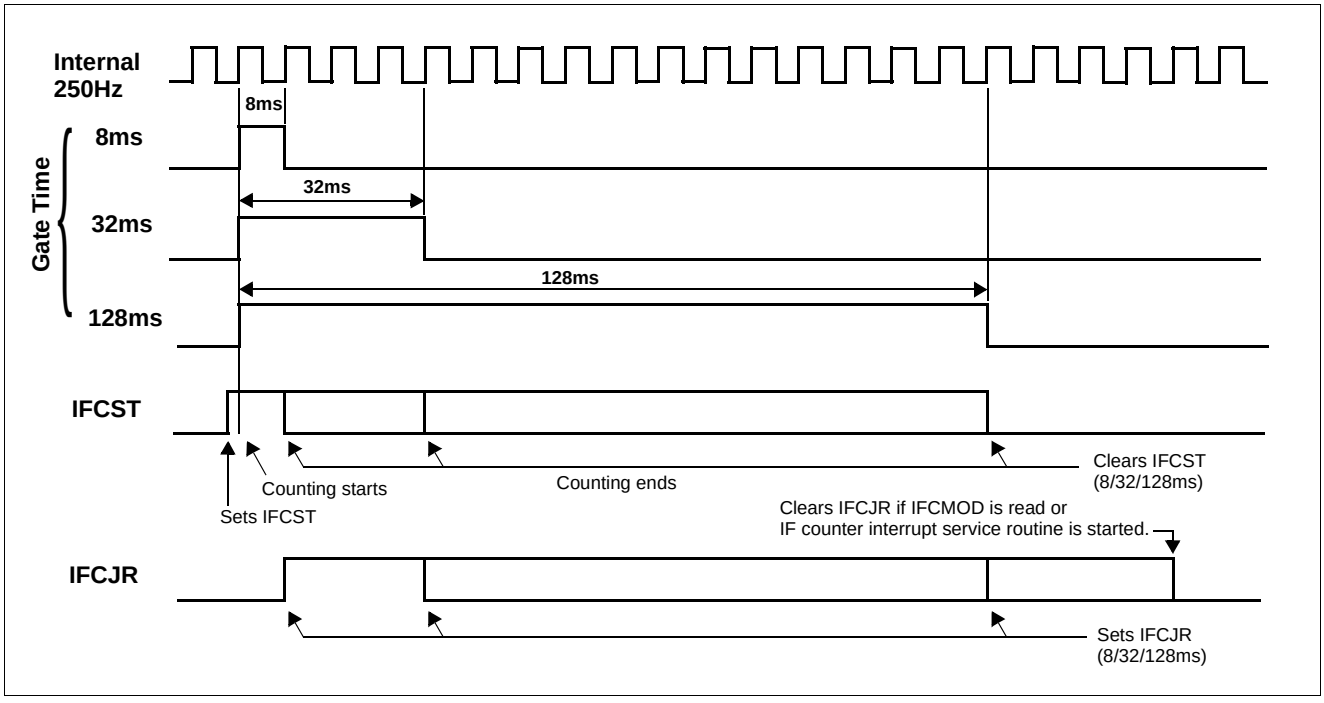


Figure 4-33 Gate Timing of Frequency Counter

Notes on Frequency Counter

(1) Notes on using frequency counter

Because signals are input to the frequency counter from an input pin (FMIFC or AMIFC pin) with an AC amplifier as shown in Figure 4-34Because signals are input to the frequency counter

from an input pin (FMIFC or AMIFC pin) with an AC amplifier as shown in Figure 4-34, cut the DC component of the input signals by using capacitor C. If the FMIFC or AMIFC pin is selected by the IF counter mode select register, switch SW1 turns ON, and switch SW2 turns OFF. As a result, the voltage on the pin is about 1/2VDD. Unless the voltage has risen to a sufficient intermediate level at this time, counting may not be performed normally because the AC amplifier is not in the normal operating range. Therefore, make sure that sufficient wait time elapses after a pin has been selected and before counting is started (IFCST = 1)., cut the DC component of the input signals by using capacitor C. If the FMIFC or AMIFC pin is selected by the IF counter mode select register, switch SW1 turns ON, and switch SW2 turns OFF. As a result, the voltage on the pin is about 1/2VDD. Unless the voltage has risen to a sufficient intermediate level at this time, counting may not be performed normally because the AC amplifier is not in the normal operating range. Therefore, make sure that sufficient wait time elapses after a pin has been selected and before counting is started (IFCST = 1).

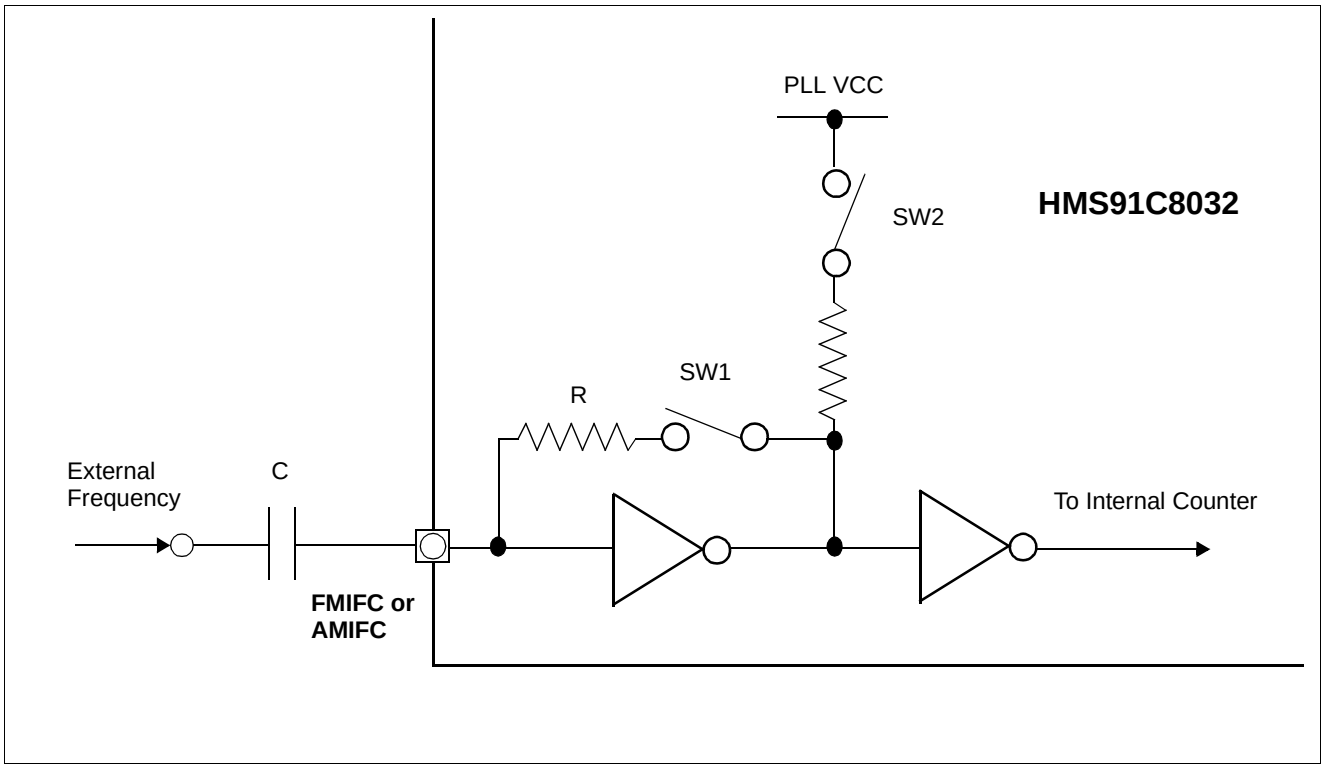


Figure 4-34 Frquency Counter Input Pin Circuit

(2) Error of frequency counter

Error of gate time

The gate time of the frequency counter is created by dividing 7.2MHz. Therefore, if 7.2MHz is shifted “+x”ppm, the gate time is also shifted “-x”ppm.

Count error

The frequency counter counts the frequency at the rising edge of the input signal. If a high level is input to the pin when the gate is opened, therefore, one excess pulse is counted. When the gate is closed, however, counting is not affected by the status of the pin. Therefore, the count error is “maximum + 1”.

4.11 PLL

The phase locked loop (PLL) frequency synthesizer is used to lock medium frequency (MF), high frequency (HF), and very high frequency (VHF) signals to a fixed frequency using a phase difference comparison system.

PLL Frequency Synthesizer Configuration

Figure 4-35 shows the PLL frequency synthesizer block diagram.

As shown in Figure 4-35, the PLL frequency synthesizer consists of an input selection circuit, programmable divider (PD), phase comparator (Phase-DET) and reference frequency generator (RFG). These blocks are connected to charge pump, an external low-pass filter (LPF) and voltage controlled oscillator (VCO). The PLL frequency synthesizer also has an internal CMOS operational amplifier so that it can be used as an external low-pass filter amplifier.

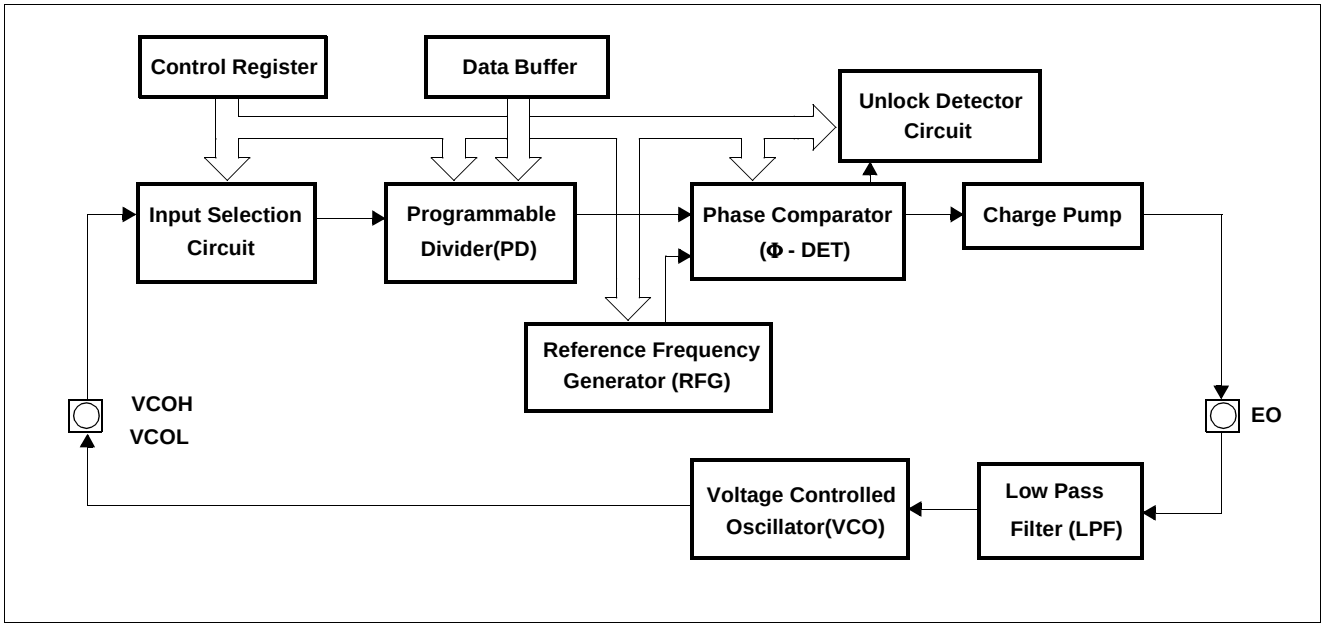


Figure 4-35 PLL Frequency Synthesizer Block Diagram

PLL Frequency Synthesizer Functions

The PLL frequency synthesizer divides the frequency of a signal from the VCOH pin or VCOL pin using a programmable divider and outputs the phase difference between the divided frequency and reference frequency from EO pin.

Input Selection Circuit

The input selection circuit selects the pin to which the signal output from an external voltage controlled oscillator is input. A VCOH or VCOL pin is selected as the input pin using a PLL mode select register (see IInput Selection Circuit and Programmable Divider Configuration section)

Programmable Divider

The programmable divider divides the frequency of a signal from the VCOH or VCOL pin at the frequency division ratio that is set using a program.

A direct frequency division system or pulse swallow system can be selected using a PLL mode select register. The frequency division value is set via the data buffer using a PLL data register. (See IInput Selection Circuit and Programmable Divider Configuration section)

Reference Frequency Generator

The reference frequency generator produces the reference frequency that is compared using a phase comparator. Twelve reference frequencies can be selected using a PLL reference mode select register. (See Reference Frequency Generator section)

Phase Comparator and Unlock Detector Circuit

The phase comparator compares the frequency divided signal output from a programmable divider and the signal from a reference frequency generator and outputs the phase difference.

The unlock detector circuits detected the PLL unlock state. The PLL unlock state is detected according to a PLL unlock flip-flop judge register, PLLUL1 and PLLUL0. (See Twelve reference frequencies (1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 25, 50KHz) can be selected using a PLL reference mode select register. The PLL reference mode select register is described in PLL Mode Select Register Configuration and Functions sectionPhase Comparator, charge pump and unlock detector circuit configuration section)

Charge Pump

The charge pump outputs the signal from a phase comparator to the EO pins as high, low, and floating output signals. (See Twelve

reference frequencies (1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 25, 50KHz) can be selected using a PLL reference mode select register. The PLL reference mode select register is described in PLL Mode Select Register Configuration and Functions sectionPhase Comparator, charge pump and unlock detector circuit configuration section)

IInput Selection Circuit and Programmable Divider Configuration

Figure 4-36 shows the input selection circuit and programmable divider configuration.

As shown in Figure 4-36, the input selection circuit consists of a VCOH pin, VCOL pin, and two input amplifiers. The programmable divider consists of a prescaler(1/16, 1/17), swallow counter (SC), programmable counter (PC), and frequency division selection switch.

Input Selection Circuit and Programmable Divider Functions

The input selection circuit and programmable divider select the input pin and frequency division system of a PLL frequency synthesizer.

A VCOH or VCOL pin can be selected as the input pin, and a direct frequency division system or pulse swallow system can be selected as the frequency division system.

The programmable divider divides a frequency according to the values of PLL data register using a swallow counter and a programmable counter. Figure 4-36 shows the input pins (VCOH and VCOL) and frequency division systems. The content of PLL mode register controls the input pin and the frequency division system for the programmable counter. The configuration and functions of the PLL mode select register are described in *PLL Mode Select Register Configuration and Functions* section.

The frequency division value of the programmable divider is set via the data buffer using a PLL data register.

Programmable Divider and PLL Data Register section describes the programmable divider and PLL data register.

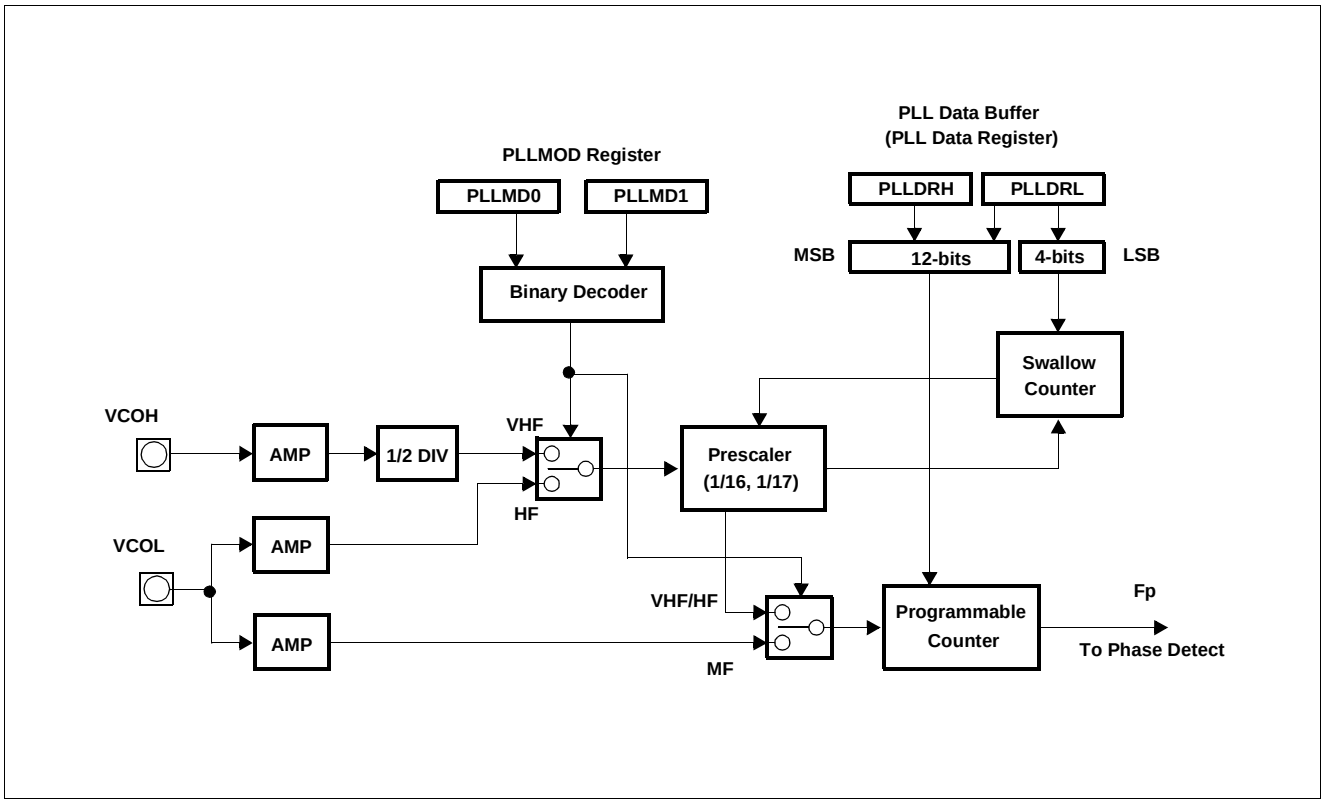


Figure 4-36 Input Selection Circuit and Programmable Divider Configuration

PLL Mode Select Register (PLLMOD) Configuration and Functions

The PLL mode select register sets the frequency division system and input pin of a PLL frequency synthesizer. The PLL mode select register configuration and functions are shown below. Steps (1) through (4) below describe the frequency division outline.

Frequency division system	Pin used	Input frequency (MHz)	Input amplitude (Vp-p)	Possible frequency division value
Direct frequency division (MF)	VCOL	0.5 to 30	0.1	16 to $2^{12} - 1$
Pulse swallow (HF)	VCOL	5 to 40	0.1	256 to $2^{16} - 1$
Pulse swallow (VHF)	VCOH	9 to 150	0.1	256 to $2^{16} - 1$

Figure 4-37 Input Pin and Frequency Division System

PLLMOD : PLL Mode Register. : F1H

PLLRF3	PLLRF2	PLLRF1	PLLRF0	PLLUL1	PLLUL0	PLLMD1	PLLMD0

PLLRF3	PLLMOD.7	See Table 4-32
PLLRF2	PLLMOD.6	See Table 4-32
PLLRF1	PLLMOD.5	See Table 4-32
PLLRF0	PLLMOD.4	See Table 4-32
PLLUL1	PLLMOD.3	Detects status of unlock FF1 (1.1μs). Set by hardware at 900KHz sampling when PLL is unlock state. Cleared by software when PLL mode register is read.
PLLUL0	PLLMOD.2	Detects status of unlock FF0 (2.2μs). Set by hardware at 450KHz sampling when PLL is unlock state. Cleared by software when PLL mode register is read.
PLLMD1	PLLMOD.1	See Table 4-33
PLLMD0	PLLMOD.0	See Table 4-33

PLLRF[3:0]				Reference Frequency of PLL (f_{OSC} = 7.2MHz)
0	0	0	0	PLL stop
0	0	0	1	1KHz
0	0	1	0	1.25KHz
0	0	1	1	2.5KHz
0	1	0	0	3KHz
0	1	0	1	5KHz
0	1	1	0	6.25KHz
0	1	1	1	9KHz
1	0	0	0	10KHz
1	0	0	1	12.5KHz
1	0	1	0	18KHz
1	0	1	1	20KHz
1	1	0	0	25KHz
1	1	0	1	50KHz
1	1	1	0	Reserved for future use *
1	1	1	1	Reserved for future use *

Table 4-32 Reference Frequency of PLL

* User software should not write 1s to reserved bits. These bits may be used in future HMS9XC8032 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

PLLMD[1:0]		Selects of PLL input pin
0	0	Disable VCOL & VCOH pins
0	1	VCOH & VHF mode select
1	0	VCOL & HF mode select
1	1	VCOL & MF mode select

Table 4-33 PLL MODE

(1) Direct frequency division system (MF)

The VCOL pin is used, and the VCOH pin is pulled down. The direct frequency division system divides the frequency using only a programmable counter.

(2) Pulse swallow system (HF)

The VCOL pin is used, and the VCOH pin is pulled down. The pulse swallow system divides the frequency using a swallow counter and a programmable counter.

(3) Pulse swallow system (VHF)

The VCOH pin is used, and the VCOL pin is pulled down. The pulse swallow system divides the frequency in a swallow counter and programmable counter.

(4) VCOL and VCOH pin disable

VCOL and VCOH pins are pulled down internally.

Programmable Divider and PLL Data Register

The programmable divider divides the frequency of a signal from the VCOH and VCOL pins according to the value of PLL mode register. The swallow counter consists of a 4-bit binary down counter, and the programmable counter consists of a 12-bit binary down counter. The frequency division value of the swallow counter and programmable counter is set via the data buffer using a PLL data register.

The PLL data register can be read and written using MOV instruction. The frequency division value is called value N.

The relation between the PLL data register and data buffer is described below. For more details of the frequency division value (N) setting in each frequency division system, see Use of PLL Frequency Synthesizer section.

(1) PLL data register and data buffer

In the direct frequency division system, the high-order 12bits are valid. In the pulse swallow system, all 16 bits are valid. The 12 bits in the direct frequency division system are set in a program counter. The high-order 12 bits in the pulse swallow system are set in a program counter, and the low-order 4bits are set in a swallow counter.

(2) Relation between frequency division value N and frequency division output frequency of programmable divider

Relation between frequency division value N and frequency division output frequency of programmable divider, f_N, is shown below. For more information, see Use of PLL Frequency Synthesizer section.

A. Direct frequency division (MF)

$f_N = f_{in} / N$ where N is 12bits

B. Pulse swallow system (HF and VHF)

$f_N = f_{in} / N$ where N is 16bits

Reference Frequency Generator

Figure 4-38 shows the reference frequency generator configuration.

As shown in Figure 4-38, the reference frequency generator divides a crystal oscillation frequency of 7.2MHz and generates reference frequency f_r of a PLL frequency synthesizer.

Twelve reference frequencies (1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 25, 50KHz) can be selected using a PLL reference mode select register. The PLL reference mode select register is described in *PLL Mode Select Register Configuration and Functions* section-Phase Comparator, charge pump and unlock detector circuit configuration

Figure 4-39 shows the phase comparator, charge pump and unlock detector circuit configuration. The phase comparator compares the phase of frequency division output f_N from a programmable divider and that of reference frequency output fr from a reference frequency generator and outputs up request (UPB) and down request (DWB) signals. The charge pump outputs the output of the phase comparator from error output pin (EO). The unlock detector circuit consisting of unlock flip-flop detects the unlock state of a PLL frequency synthesizer. .

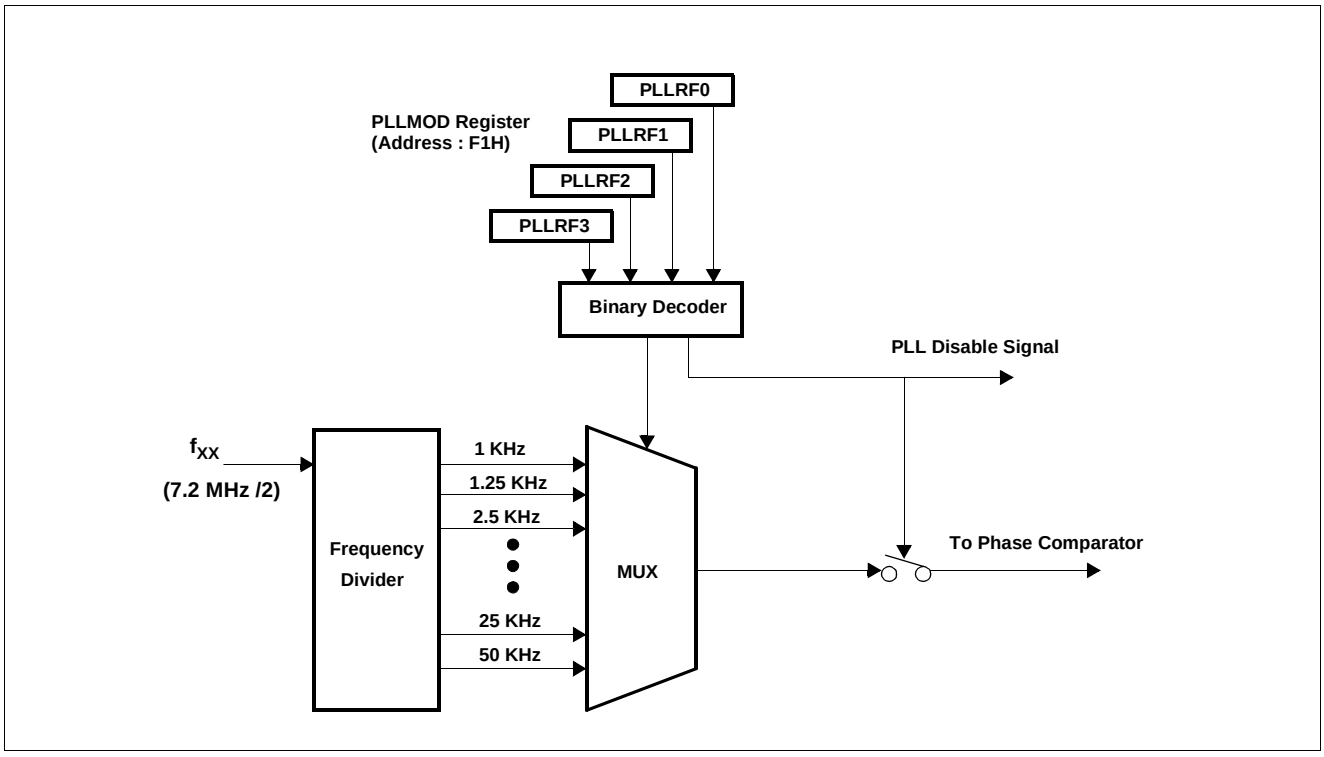


Figure 4-38 Referenc Frequency Generator Configuration

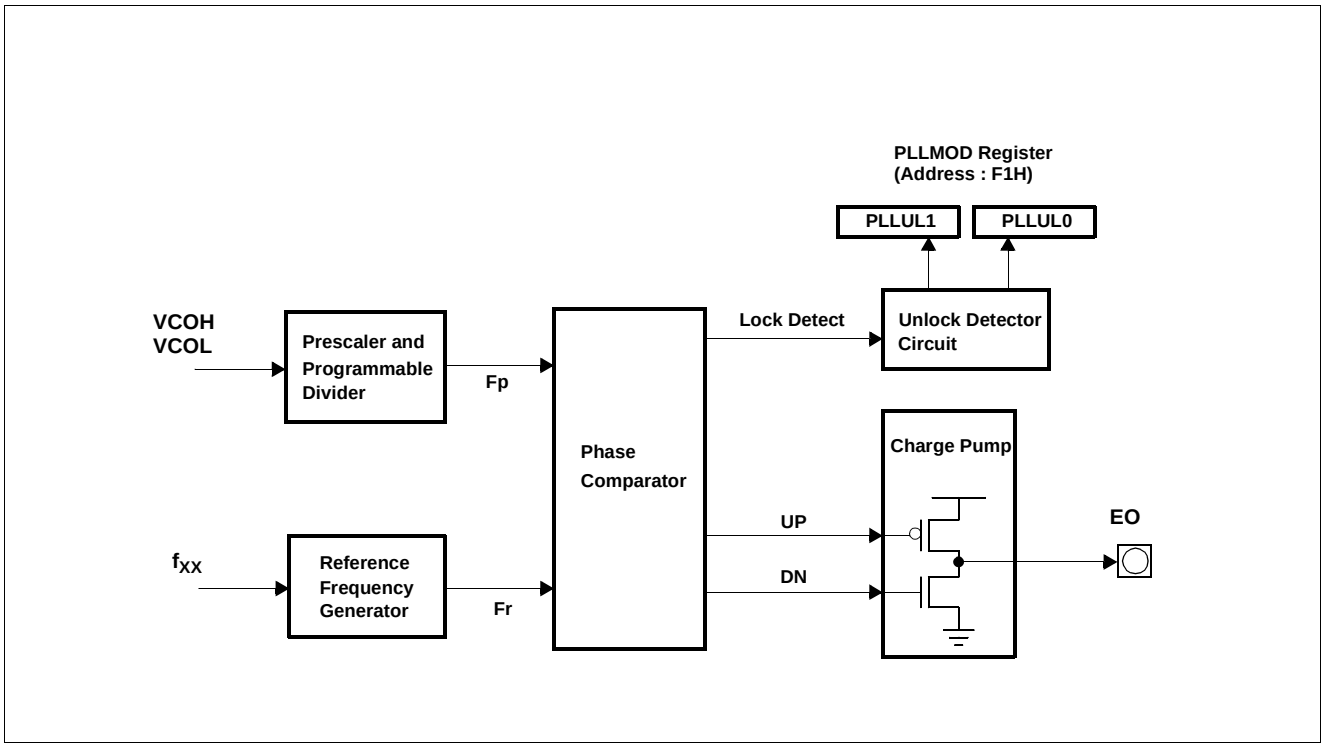


Figure 4-39 Phase Comparator, Charge Pump and Unlock Detector Circuit Configuration

Phase Comparator Functions

As shown in Figure 4-39, the comparator compares the phase of frequency division output "f_N" from a programmable divider and that of reference frequency output f_r from a reference frequency generator and outputs up request (UP) and down request (DN) signal. The UP signal is activated to low if divided frequency f_N is higher than reference frequency f_r. The DN signal is activated to high if the former is lower than the latter.

shows the reference frequency (f_r), divided frequency (f_N), UP signal, and DN signal. The up and down request signals are input to the charge pump and unlock detector circuit.

Charge Pump

As shown in Figure 4-39, the charge pump outputs the UP and DN signal from a phase comparator from error output pins.

The relation between the error output pin output, divided frequency f_N, and reference frequency fr is shown below

Reference frequency fr > Divided frequency f_N : Low level output

Reference frequency fr < Divided frequency f_N : High level output

Reference frequency fr = Divided frequency f_N : Floating

Unlock Detector Circuit

As shown in Figure 4-39, the unlock detector circuit detects the unlock state of a PLL frequency synthesizer using the up and down request signals from a phase comparator.

The UP and DN signal cause EO to be a low or high signal when the PLL frequency synthesizer is in unlock state. An unlock flip-flop (FF) is set to high when PLL is in unlock state. The unlock FF state is detected using a PLL unlock flip-flop judge register. An unlock flip-flop is set according to the period of reference frequency, fr, selected at that time. The unlock flip-flop is also reset when the PLL unlock flip-flop judge register information is read using a MOV command. This unlock flip-flop must thus be detected at a period longer than period of reference frequency f_r. (1/fr)

The PLL unlock flip-flop judge register that is a read only register is reset when the register information is read in a window using a MOV command. The unlock flip-flop is set at a period of reference frequency fr. Therefore, this register must be read at a period longer than period of a reference frequency (1/fr) when it is read in the window register.

Use of PLL Frequency Synthesizer

The data below is required to control a PLL frequency synthesizer.

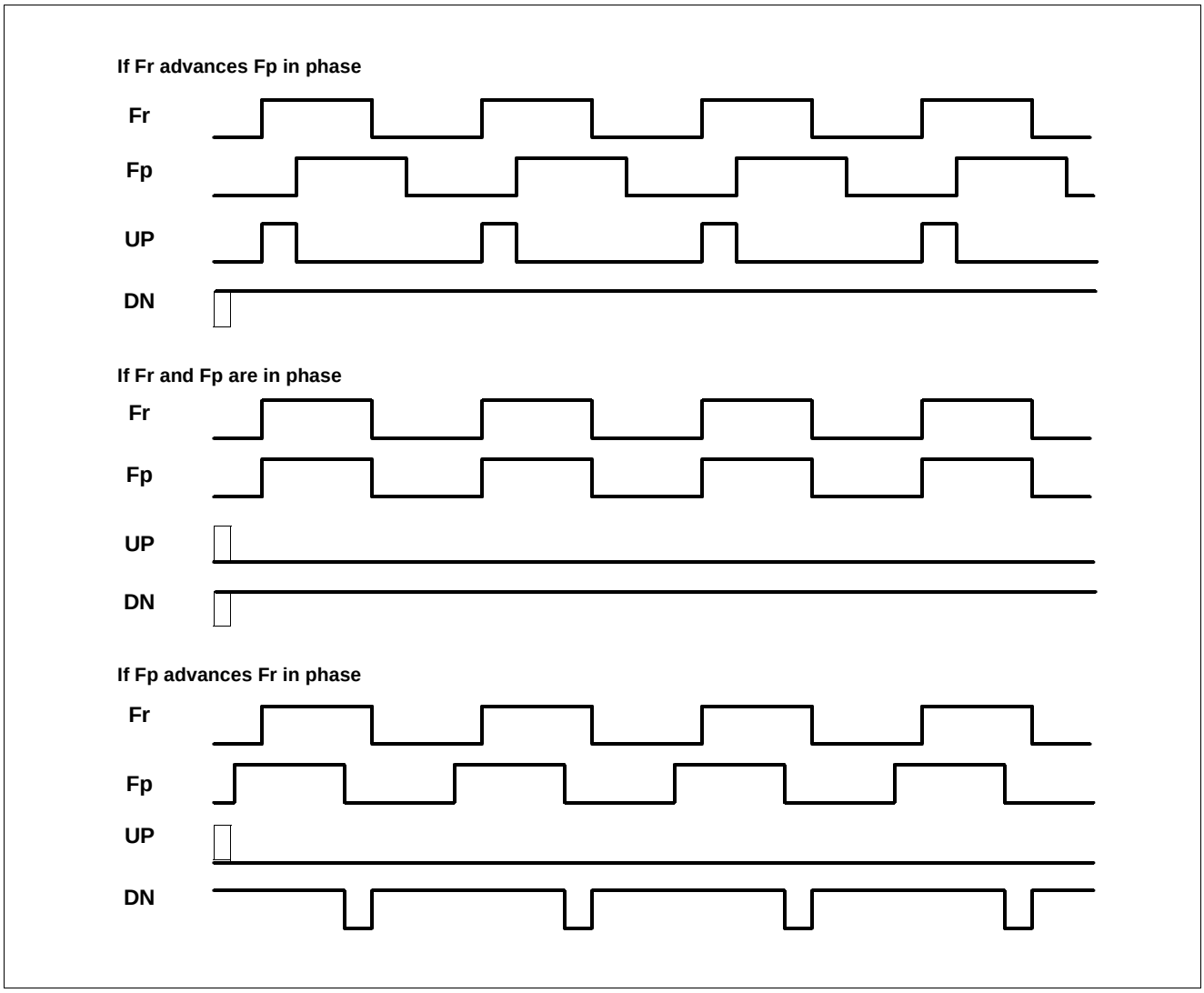


Figure 4-40 Relation Between f_r, f_N, UPB and DWB signals

(1) Frequency division system : Direct frequency division (MF) and pulse swallow (HF and VHF)

(2) Pin used : VCOL and VCOH pins

(3) Reference frequency : f_r

(4) Frequency division value : N

Setting the PLL data in each frequency division system (MF, HF and VHF) is described in this section

Direct Frequency Division System

(1) Frequency division system selection

The direct frequency division system is selected using a PLL mode select register.

(2) Pin used

The VCOL pin can operate when the direct frequency division system is selected.

(3) Reference frequency fr setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$N = f_{VCOL} / fr$

where f_{VCOL} : Input frequency at VCOL pin

fr : Reference frequency

(5) PLL data setting example

The data used to receive the MW-band broadcasting station below is set as follows:

Receive frequency : 1260KHz (MW band)

Reference frequency : 9KHz

Intermediate frequency : +450KHz

Frequency division value N is given by

$N = f_{VCOL} / fr = (1260 + 450) / 9 = 190$ (decimal) = 0BEH (hexadecimal)

Pulse Swallow System (HF)

(1) Frequency division system selection

The pulse swallow system is selected using a PLL mode select register.

(2) Pin used

The VCOL pin can operate when the pulse swallow system is selected.

(3) Reference frequency fr setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$N = f_{VCOL} / f_r$

where f_{VCOL} : Input frequency at VCOL pin

f_r : Reference frequency

(5) PLL data setting example

The data used to receive the SW-band broadcasting station below is set as follows:

Receive frequency : 25.50MHz (SW band)

Reference frequency : 5KHz

Intermediate frequency : +450KHz

Frequency division value N is given by

$N = f_{VCOL} / fr = (25500 + 450) / 5 = 5190$ (decimal) = 1446H (hexadecimal)

Pulse Swallow System (VHF)

(1) Frequency division system selection

The pulse swallow system is selected using a PLL mode select register.

(2) Pin used

The VCOH pin can operate when the pulse swallow system is selected.

(3) Reference frequency fr setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$N = f_{VCOH} / f_r$

where f_{VCOH} : Input frequency at VCOH pin

f_r : Reference frequency

(5) PLL data setting example

The data used to receive the FM-band broadcasting station below is set as follows:

Receive frequency : 100.0MHz (FM band)

Reference frequency : 25KHz

Intermediate frequency : +10.7MHz

Frequency division value N is given by

$N = f_{VCOH} / f_r = (100.0 + 10.7) / 0.025 = 4428$ (decimal) = 114CH (hexadecimal)

Data is set in a PLL data register and PLL mode select register as shown below.

4.12 ADC

The analog-to-digital converter (A/D) allows conversion of an analog input signal to a corresponding 8-bit digital value. The A/D module has eight analog inputs, which are multiplexed into one sample and hold. The output of the sample and hold is the input into the converter, which generates the result via successive approximation. The analog supply voltage is connected to Avref+ of ladder resistance of A/D module.

The A/D module has two registers which are the control register ADCCON and A/D result register ADCDR. The register ADCCON, shown in Figure C-33 ADC Block Diagram, controls the operation of the A/D converter module. The Port7 pins can be configured as analog inputs or digital I/O. To use analog inputs, I/O is selected input mode by P7MOD register.

ADCCON: AD CONVERTER CONTROL REGISTER. : 84H

-	ADCEN	-	ADCCH2	ADCCH1	ADCCH0	ADCST	ADCSF

-	ADCCON.7	Reserved for future use *
ADCEN	ADCCON.6	ADC Enable flag
-	ADCCON.5	Reserved for future use *
ADCCH2	ADCCON.4	See Table 4-34
ADCCH1	ADCCON.3	SeeTable 4-34
ADCCH0	ADCCON.2	See Table 4-34
ADCST	ADCCON.1	Software START control for ADC. A logic 1 starts A/D conversion.
ADCSF	ADCCON.0	A/D conversion completion flag. Set by hardware when ADC operation complete. Cleared by hardware when this flag is read.

ADCCH[2:0]			Select ADC channel
0	0	0	Select channel 0
0	0	1	Select channel 1
0	1	0	Select channel 2
0	1	1	Select channel 3
1	0	0	Select channel 4
1	0	1	Select channel 5
1	1	0	Select channel 6
1	1	1	Select channel 7

Table 4-34 ADCCON Register

How to Use A/D Converter

The processing of conversion is start when the start bit ADST is set to 1. After one cycle, it is cleared by hardware. ADCDR contains the results of the A/D conversion. When the conversion is completed, the result is loaded into the ADCDR, the A/D conversion status bit ADSF is set to 1, and the A/D interrupt flag AIF is set. The block diagram of the A/D module is shown in Figure 4-41. The A/D status bit ADSF is set automatically when A/D conversion is completed, cleared when A/D conversion is in process.

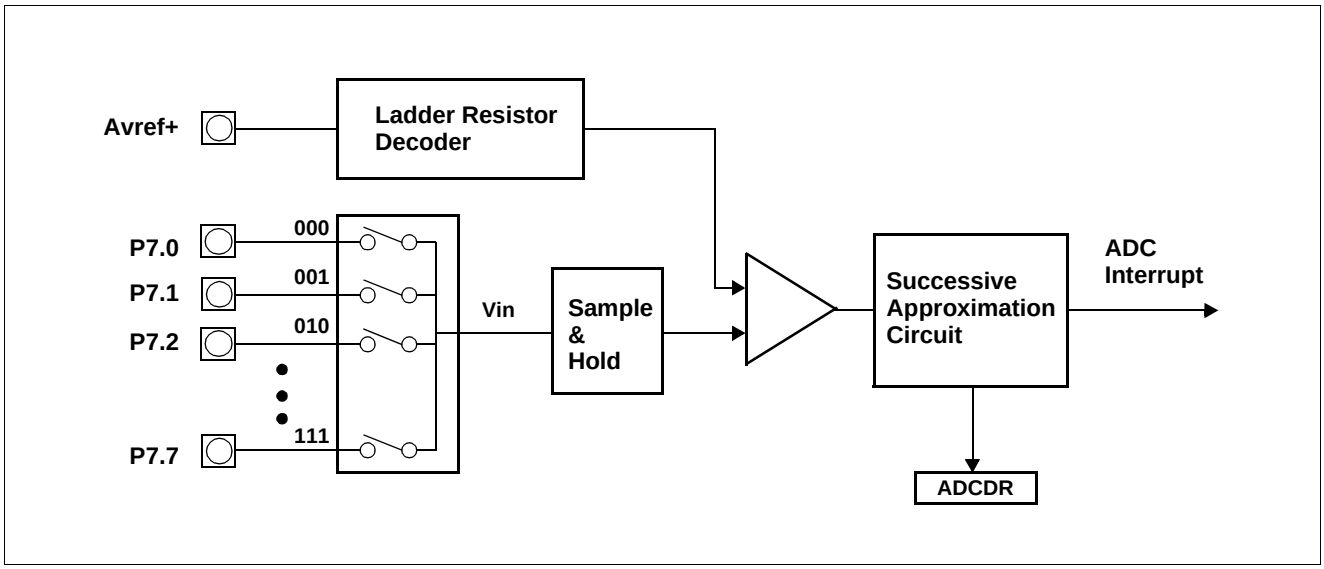


Figure 4-41 ADC Block Diagram

Guideline on ADC

Programmers who want to use ADC in HMS91C8032 series should follow the recommended rules.

1. To enter the power down mode, programmers should power off the ADC using ADCCON.6 flag. When ADC is on, though HMS91C8032 is in the power down mode, static leakage current may be.

2. While ADC is converting analog input, HMS91C8032 core should do nothing except NOP instruction. This is the reason that some instructions would disrupt the ADC result. So, interrupt function should be disabled because when unexpected interrupt is called, some instructions in the interrupt routine may disrupt the ADC result. Example code is as follows. **ADC conversion time can be calculate by 21*6*(1/f_{XX}) seconds.** If f_{MOSC} is 7.2MHz and f_{CPU} is 1/2 f_{MOSC}, conversion time is approximately 22 machine cycles. So, at least 22 NOP instructions are required for ADC conversion.

Example code)

```
; Interrupt should be disabled
mov     adccon, #0e2h ; start ADC operation
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
mov     a, adcdr ; read the conversion result
```

4.13 Interrupts

The HMS9XC8032 provides 18 interrupt sources. (The 7 external interrupts and 11 internal interrupts) Among the 7 external interrupts (INT0 through INT6), the 6 External Interrupts (INT0 through INT5) can be configured as either level-activated or transition-activated, depending on bits in Register IT2, and the external interrupt source, INT6, can only be transition-activated. The flags that actually generate these interrupts are bits IR0 and IR1 registers. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was a level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer0, Timer1, Timer2, Timer3 and Timer4 interrupts are generated by TF0, TF1, TF2 (T2EX), TF3 and TF4 which are set when rollover in their respective Timer/Counter registers (except see Timer 0 and Timer 3 in Mode 3) occurs. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The UART interrupt is generated by the logical OR of RI and TI. And SIO1 and SIO2 Interrupt is generated by IRS1 and IRS2. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine will normally have to determine whether it was RI or TI that generated the interrupt, and the bit will have to be cleared in software.

The IF counter, ADC, WDT Interrupt is generated by IRIF, IRADC and IRWDT. Neither of these flags is cleared by hardware when the service routine is vectored to. Especially, WDT interrupt can be NMI (Non Maskable Interrupt).

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be canceled in software.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in special Function Register IE, IE2 and IE3. IE also contains a global disable bit, EA, which disables all interrupts at once.

Priority Level Structure

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in Special Function Register IP, IP2 and IP3. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high priority interrupt can't be interrupted by any other interrupt source.

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, an internal poling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence as follows:

Source	Priority Within Level
(Highest)	
INTEX0	External interrupt 0
INTT0	Timer0/Counter0 interrupt
INTEX1	External interrupt 1
INTT1	Timer1/Counter1 interrupt
INTS0 (RI or TI)	UART interrupt
INTT2 (TF2 or EXF2)	Timer2/Counter2 interrupt
INTWDT	WDT interrupt
INTIFC	IF Counter interrupt
INTAD	ADC interrupt
INTEX2	External interrupt 2
INTEX3	External interrupt 3
INTEX4	External interrupt 4
INTS1	SIO1 interrupt
INTS2	SIO2 interrupt
INTEX5	External interrupt 5
INTEX6	External interrupt 6
INTT3	Timer3/Counter3 interrupt
INTT4	Timer4/Counter4 interrupt
(Lowest)	

Note that the “priority within level” structure is only used to resolve simultaneous requests of the same priority level.

The IP, IP2 and IP3 register contains a number of unimplemented bits. IP.7, IP.6, IP2.7, IP2.6, IP2.5 and IP3.7 are reserved in the HMS9XC8032. User software should not write 1s to these positions, since they may be used in other HMS9XC8032 Family products.

How interrupts Are Handled

The interrupt flags are sampled at S5P2 of every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.

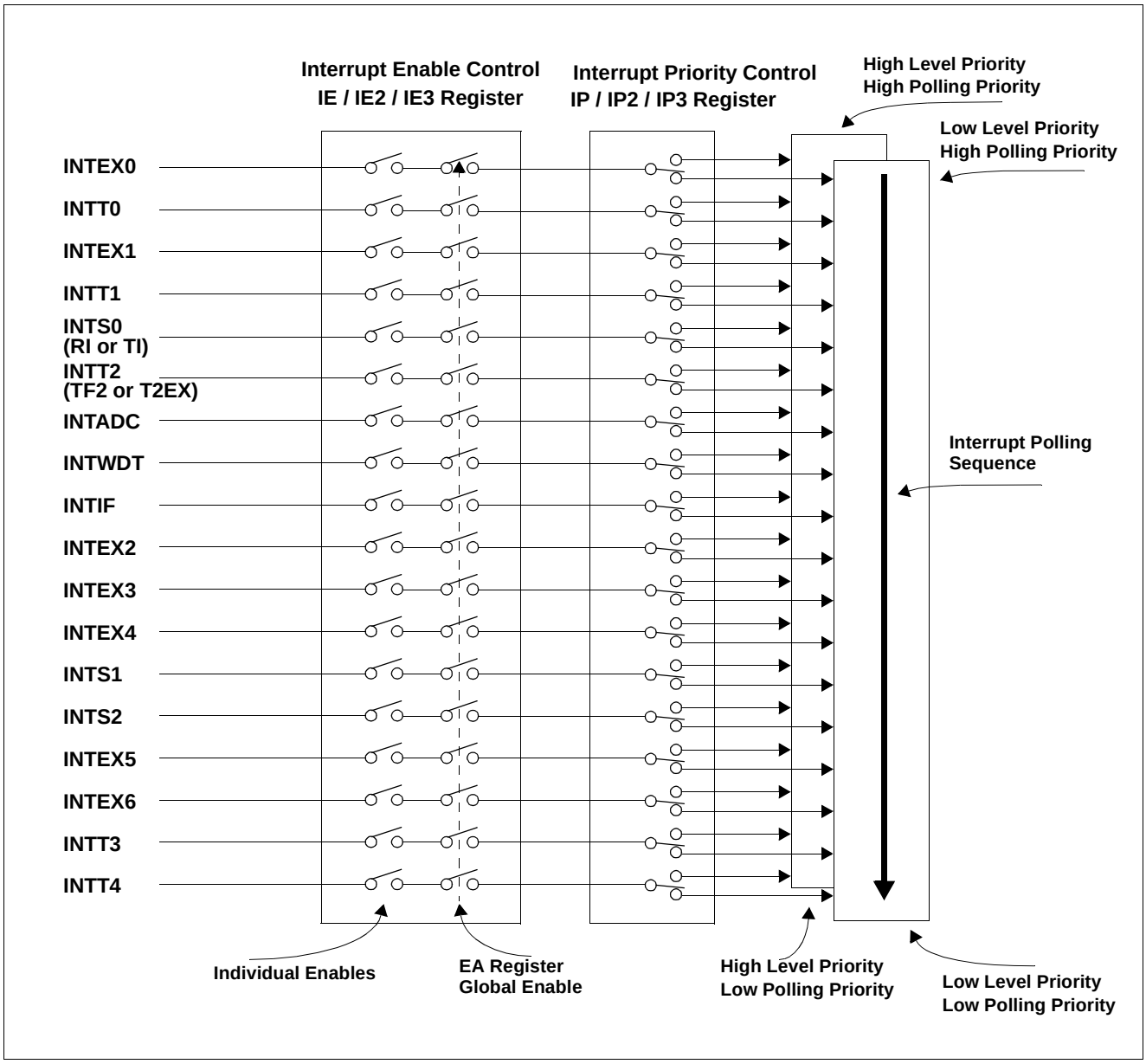


Figure 4-42 Interrupt Control Block

2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.

3. The RETI instruction in progress or any write to the IEs or IPs registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine, Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IEs or IPs, then at least one more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at S5P2 of the previous machine cycle. Note that if an interrupt flag is active but not being responded to for one of the above conditions, and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is now.

The polling cycle/LCALL sequence is illustrated in Figure 4-43.

Note that if an interrupt of higher priority level goes active prior

to S5P2 of the machine cycle labeled C3 in Figure 20, then in accordance with the above rules it will be vectored to during C5 and C6, without any instruction of the lower priority routine having been executed.

Thus the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine.
In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the Serial Port flag. This has to be done in the user's software. It clears an external interrupt flag (IE or IE2) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter on to the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to, as shown below:

Source	Vector Address
INTEX0	0003H
INTT0	000BH
INTEX1	0013H
INTT1	001BH
INTS0 (RI & TI)	0023H
INTT2 (TF2 & EXF2)	002BH
INTWDT	0033H
INTIFC	003BH
INTAD	0043H
INTEX2	004BH
INTEX3	0053H
INTEX4	005BH
INTS1	0063H
INTS2	006BH
INTEX5	0073H
INTEX6	007BH
INTT3	0083H
INTT4	008BH

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where in left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress, making future interrupts impossible.

External Interrupts

The external sources can be programmed to be level-activated or transition-activated by setting or clearing bit IT2 Register. Since the external interrupt pins are sampled once each machine cycle, an input high or low should hold for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one machine cycle, and then hold it low for at least one machine cycle. This is done to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.
If the external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Response Time

The /INTx levels are inverted and latched into IE and IE2 register at S5P2 of every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycle elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. Figure 4-43 shows interrupt response timings.

A longer response time would result if the request is blocked by one of the 3 previously listed conditions. If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is no in its final cycle, the additional wait time cannot be more the 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV).

Thus, in a single interrupt system, the response time is always more than 3 cycles and less than 9 cycles.
Single-Step Operation

The HMS9XC8032 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been executed, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-step operation is to program one of the external interrupts (e.g., $\overline{\text{INT0}}$) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$   ; Wait Till INT0 Goes High
```

```
JB    P3.2,$   ; Wait Till INT0 Goes Low
RETI  ; Go Back and Execute One Instruction
```

Now if the $\overline{INT0}$ pin is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until $\overline{INT0}$ is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of $\overline{INT0}$. One step of the task program is executed each time $\overline{INT0}$ is pulsed.

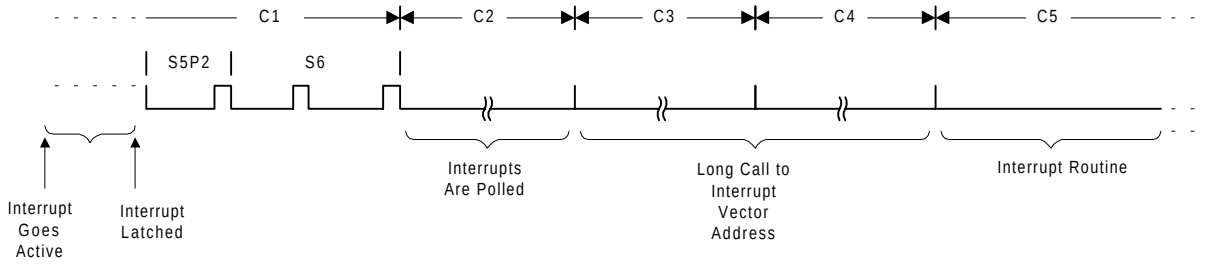
Simulating a Third Priority Level in Software

Some applications require more than two priority levels that are provided by on-chip hardware in HMS9XC8032 devices. In these cases, relatively simple software can be written to produce the same effect as a third priority level. First, interrupts that are to have higher priority than 1 are assigned to priority 1 in the Interrupt Priority (IP) register. The service routines for priority 1 interrupts that are supposed to be interruptible by priority 2 interrupts are written to include the following code :

```
        PUSH      IE
        MOV       IE,#MASK
        CALL      LABEL
        ***************************
        (execute service routine)
        ***************************
        POP       IE
        RET
LABEL:            RET1
```

As soon as any priority interrupt is acknowledged, the Interrupt Enable (IE) register is redefined so as to disable all but priority 2 interrupts. Then a CALL to LABEL executes the RETI instruction, which clears the priority 1 interrupt that is enabled can be serviced, but only priority 2 interrupts are enabled.

POPing IE restores the original enable byte. Then a normal RET (rather than another RETI) is used to terminate the service routine.



This is the fastest possible response when C2 is the final cycle of an instrcution other than RETI or an access to IE or IP.

Figure 4-43 Interrupt Response Timing Diagram

4.14 Reset

The reset input is the RST pin, which is the input to a Schmitt Trigger.

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), *while the oscillator is running.* The CPU responds by generating an internal reset.

The external reset signal is asynchronous to the internal clock. The RST pin is sampled during State 5 Phase of every machine cycle. The port pins will maintain their current activities for 19 oscillator periods after a logic 1 has been sampled at the RST pin; that is, for 19 to 31 oscillator periods after the external reset signal has been applied to the RST pin.

The internal reset algorithm writes 0s to all the SFRs except the port latches, the Stack Pointer, and SBUF, The port latches are initialized to FFH, the Stack Pointer to 07H, and SBUF is indeterminate.

The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

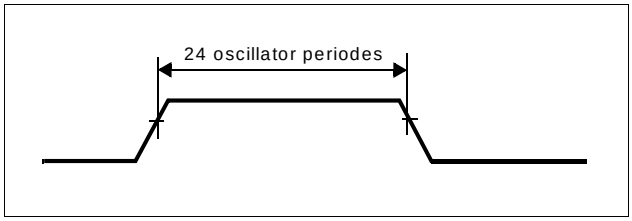


Figure 4-44 Reset Timing

4.15 Power-On Reset

An automatic reset can be obtained when Vcc is turned on by connecting the RST pin to Vcc through a 10µf capacitor. CMOS devices do not require external resistor although its presence does no harm, because they have an internal pulldown on the RST pin.

On power up, Vcc rise time does not exceed 10 millisecond and the oscillator start-up time will depend on the oscillator frequency. This power-on reset circuit is shown in Figure 4-45.

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the value of the capacitor and the rate at which it charges. To ensure a good reset, the RST pin must be high long enough to allow the oscillator time to start-up (normally a few ms) plus two machine cycles.

Note that the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them.

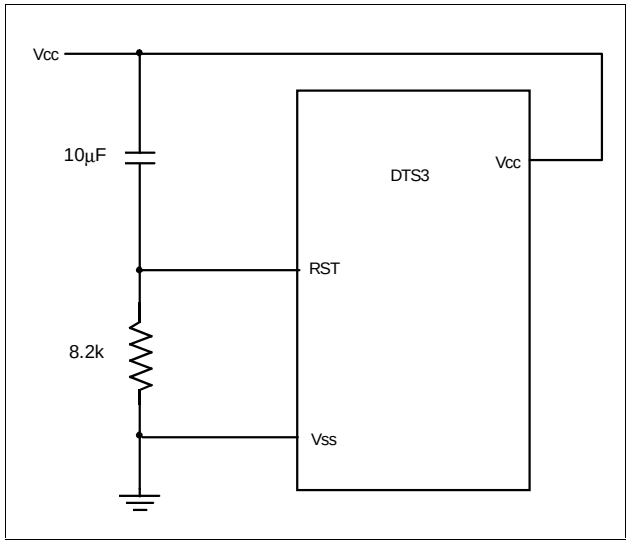


Figure 4-45 Power-On Reset Circuit

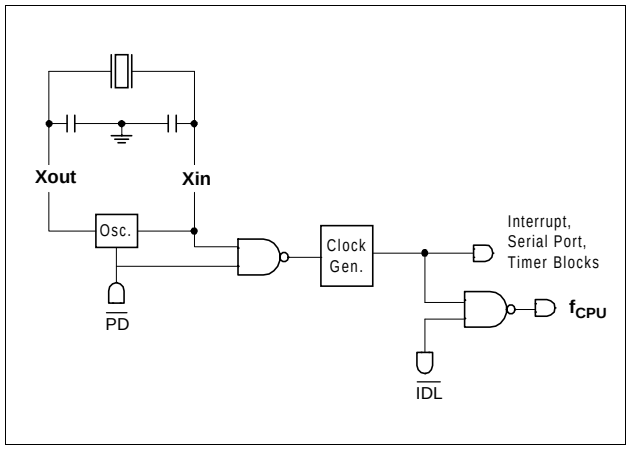


Figure 4-46 Idle and Power Down Hardware

With this circuit, reducing Vcc quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited, and will not harm the device.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs, specifically the Program Counter, may not get properly initialized.

4.16 Power-Saving Modes of Operation

For applications where power consumption is critical the CMOS version provides power reduced modes of operation as a standard feature.

CMOS versions have two power reducing modes, Idle and Power Down. The input through which backup power is supplied during these operations is Vcc. Figure 4-46 shows the internal circuitry which implements these features. In the Idle modes (IDL = 1), the oscillator continues to run and the Interrupt, Serial Port, and Timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In Power Down (PD=1), the oscillator is frozen. The Idle and Power Down Modes are activated by setting bits in Special Function Register PCON. The address of this register is 87H. Figure 4-47 details its contents.

Idle Mode

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode. In the Idle mode, the internal clock signal is gated off to the CPU but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety; the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle was activated.

There is one way to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be service, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during on Idle. For example, an instruction that activates Idle can also set on or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

Power-Down Mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, the contents of the on-chip RAM and Special Function Registers are maintained. The port pins output the values held by their respective SFRs. The only exit from Power Down is a hardware reset. Reset redefines all the SFRs, but does not change the on-chip RAM. In the Power Down mode of operation, Vcc can be reduced to as low as 2V. Care must be taken, however, to ensure that Vcc is not reduced before the Power Down mode is invoked, and that Vcc is restored to its normal operating level, before the Power Down mode is terminated. The reset that terminates Power Down also frees the oscillator. The reset should not be activated before Vcc is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize (normally less than 10ms).

(MSB) (LSB)

SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2 or 3.
-	PCON.6	Reserved.
-	PCON.5	Reserved.
-	PCON.4	Reserved.
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power-down bit. Setting this bit activates power-down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1s are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000). User software should never write 1s to unimplemented bits, since they may be used in future products.

Figure 4-47 Power Control Register (PCON)

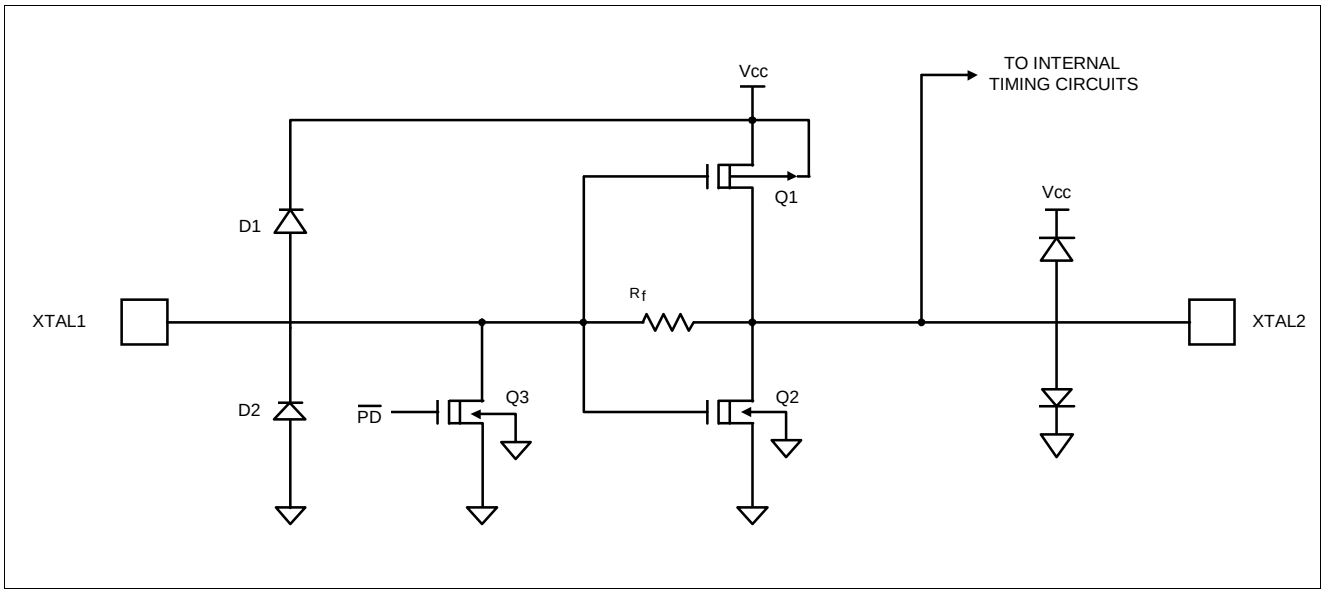


Figure 4-48 On-Chip Oscillator Circuitry in the CMOS Version of the HMS9XC8032

4.17 The On-Chip Oscillators

The on-chip oscillator circuitry for the HMS9XC8032, shown in Figure 4-48, consists of a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator.

The HMS9XC8032 is able to turn off its oscillator under software control (by writing a 1 to the PD bit in PCON), and the internal clocking circuitry is driven by the signal at XTAL2.

The feedback resistor R_f in Figure 4-50 consists of paralleled n- and p-channel FETs controlled by the PD bit, such that R_f is opened when PD = 1. The diodes D1 and D2 which act as clamps to Vcc and Vss, are parasitic to the R_f FETs. The oscillator can be used with the external components, as shown in Figure 4-50. Typically, C1 = C2 = 30pF when the feedback element is a quartz crystal, and C1 = C2 = 47pF when a ceramic resonator is used.

To drive the CMOS parts with an external clock source, apply the external clock signal to XTAL2, and leave XTAL1 float, as shown in Figure 4-50.

In the CMOS parts the internal timing circuits are driven by the signal at XTAL2.

7.2 MHz Oscillator : Xout, Xin

C1 = C2 = 30pF ° ± 10pF

32.768 KHz Oscillator : XTout, XTin

C1 = C2 = 100pF ° ± 20pF

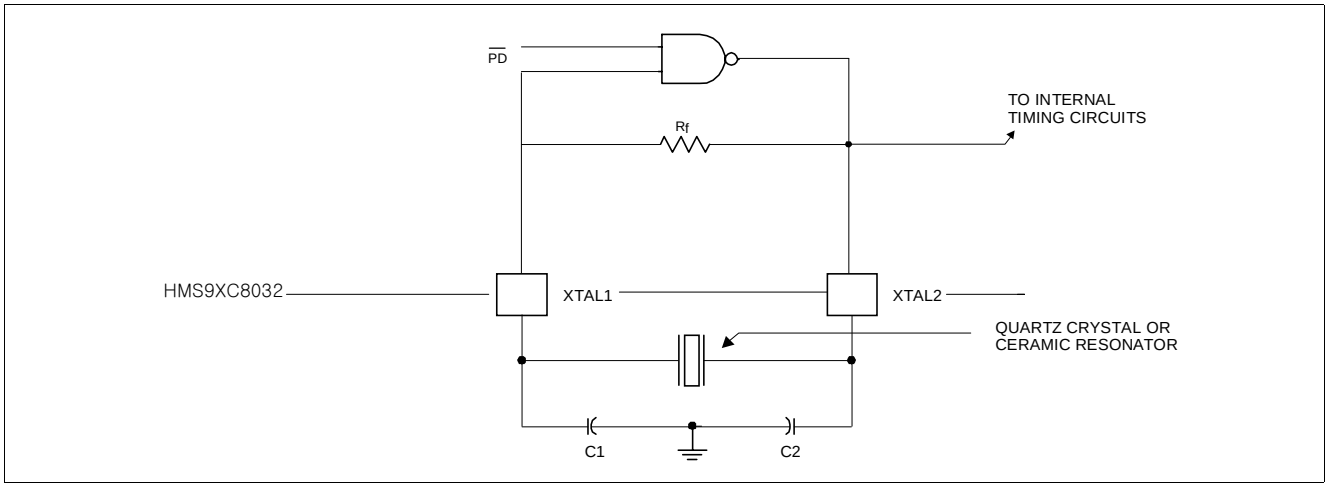


Figure 4-49 Using the CMOS On-Chip Oscillator

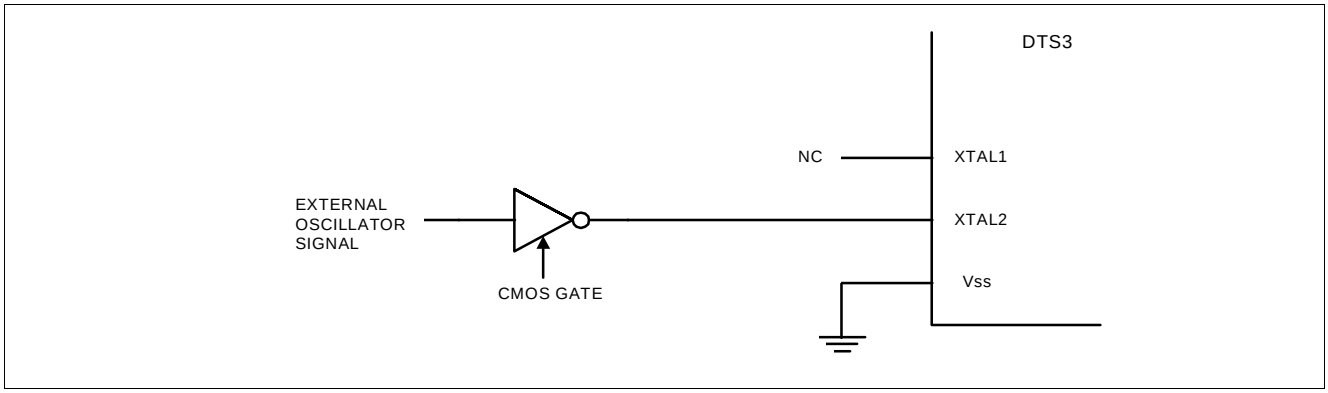


Figure 4-50 Driving the CMOS Family Parts with an External Clock Source

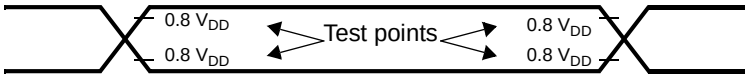
5. ELECTRICAL CHARACTERISTICS

5.1 Operating Conditions

Symbol	Descriptions	Min	Max	Units
T_A	Ambient Temperature Under Bias	-40	+85	ºC
V_{DD}	Supply Voltage	4.5	5.5	V
f_{OSC}	Oscillator Frequency		7.2 (32.768)	MHz (KHz)

5.2 AC Characteristics

AC TIMING TEST POINT

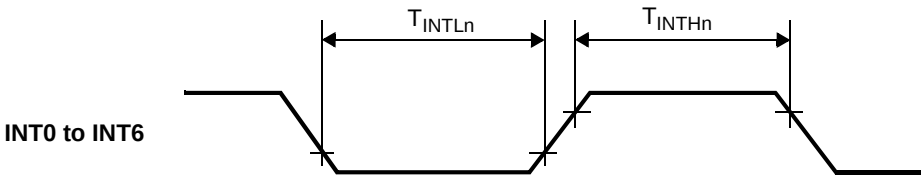


BASIC OPERATION (T_A = -40 to +85 ºC, V_{DD} = 4.5 to 5.5V)

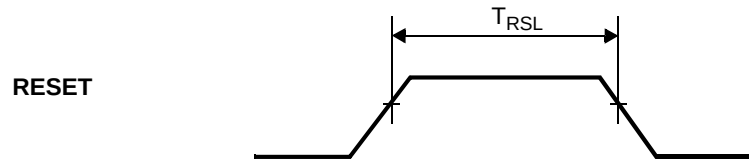
Parameter	Symbol	Variable	MIN.	TYP.	MAX.	Unit
Oscillator frequency	f_x		0	7.2	10	MHz
Interrupt input high/ low-level width	T_{INTHn}/ T_{INTLn}	Minimum : 13*(1/f_x)	1.8Note			µs
RESET high level width	T_{RSL}	Minimum : 30*(1/f_x)	4.17Note			µs
T0,T1,T2,T3,T4 input frequency	f_{Tm}	Maximum : f_{Tm} = f_x/28			3.89Note	µs
T0,T1,T2,T3,T4 input High/low level width	T_{THm}/ T_{TLm}	Minimum : 13*(1/f_x)	1.8Note			µs

** **Note.** When f_x is 7.2 MHz.*

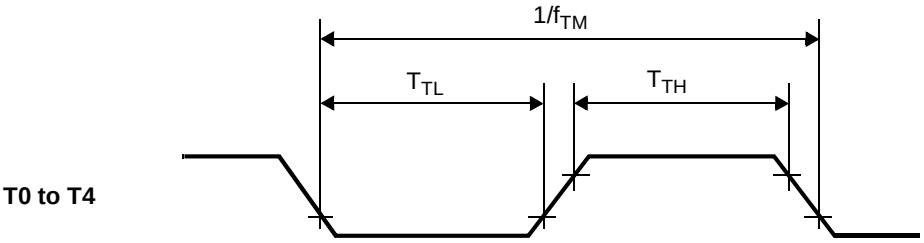
INTERRUPT TIMING WAVEFORM



RESET TIMING WAVEFORM



TIMER INPUT TIMING WAVEFORM



SERIAL INTERFACE(SIO) (T_A = -40° to +85°, V_{DD} = 3.5 to 5.5 V)

- **3-wire serial I/O mode (SCK0 … internal clock output)**

Parameter	Symbol	Variable	MIN.	TYP.	MAX.	Unit
SCK0 cycle time	T_{KCY1}	Minimum : $(1/f_x)*2*8$	2220 Note1			ns
SCK0 high/low-level width	T_{KH1} / T_{KL1}	Minimum : $T_{KCY1}/2-100$	1010 Note1			ns
SIO setup time (to SCK0)	T_{SIK1}		300			ns
SIO hold time (to SCK0)	T_{KSI1}		400			ns
SO0 output delay time from SCK0	T_{KSO1}	C = 100 pF Note2			300	ns

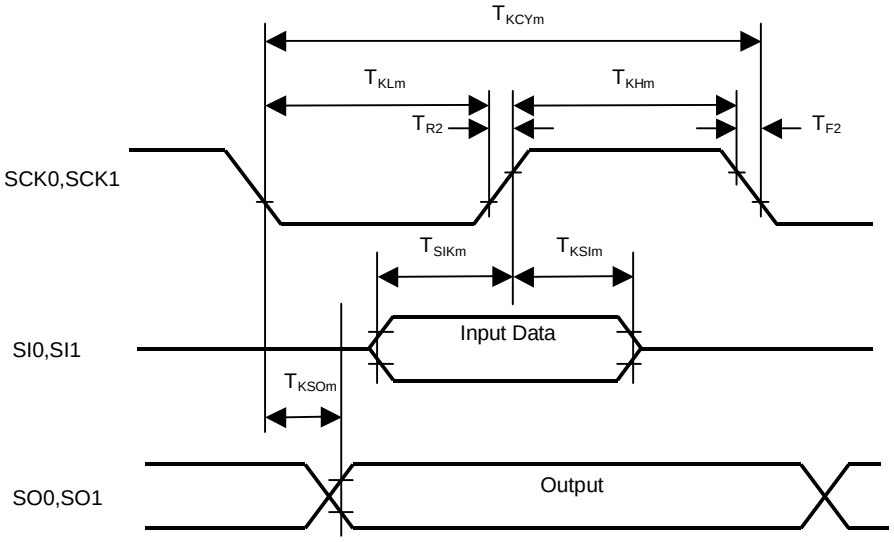
**** Note 1.*** *When f_x is 7.2 MHz*
2. *C is the load capacitance of SO0 output line.*

3-wire serial I/O mode (SCK0 … external clock input)* Note 1. When f_X is 7.2MHz.

Parameter	Symbol	Variable	MIN.	TYP.	MAX.	Unit
SCK0 cycle time	T_{KCY2}	Minimum : $(1/f_x)*2*8$	2200 Note1			ns
SCK0 high/low-level width	T_{KH2} / T_{KL1}	Minimum : $T_{KCY2}/2-100$	1010 Note1			ns
SIO setup time (to SCK0)	T_{SIK2}		100			ns
SIO hold time (to SCK0)	T_{KSI2}		400			ns
SO0 output delay time from SCK0	T_{KSO2}	C = 100 pF Note2			300	ns
SCK0 at rising or falling edge time	T_{R2} , T_{F2}				100	ns

**** Note 1.*** *When f_x is 7.2 MHz*
2. *C is the load capacitance of SO0 output line.*

3-WIRE SERIAL I/O MODE TIMING WAVEFORMS



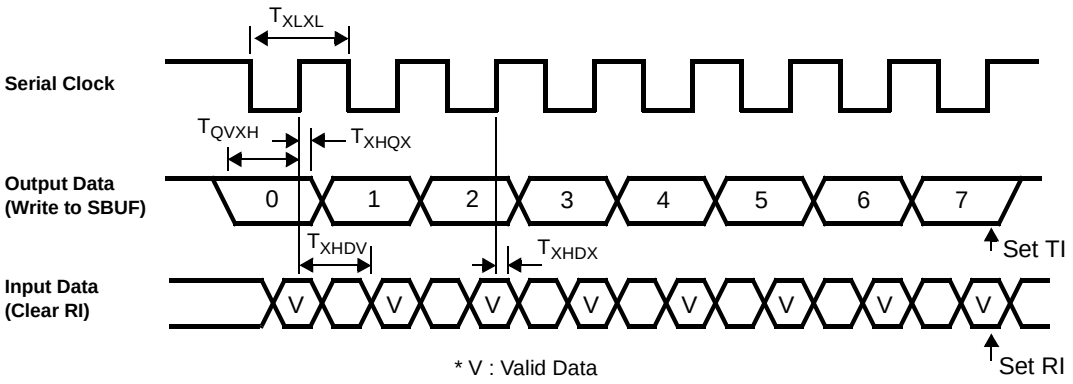
SERIAL PORT(UART) TIMING

Test Conditions : Over Operation Conditions ; Load Capacitance = 80 pF

Parameter	Symbol	Variable	Min	Max	Units
Serial Port Clock Cycle Time	T_{XLXL}	Minimum : 13*(1/f_X)	1.81[Note]		µs
Output Data Setup to Clock Rising Edge	T_{QVXH}		1.39		µs
Output Data Hold After Clock Rising Edge	T_{XHQX}		280		ns
Input Data Hold After Clock Rising Edge	T_{XHDX}		0		ns
Clock Rising Edge to Input Data Valid	T_{XHDV}			1.39	µs

**** Note.*** *When f_X is 7.2MHz.*

SHIFT REGISTER MODE TIMING WAVEFORMS



A/D CONVERTER CHARACTERISTIC (TA = -40° to +85°, VDD = 4.5 to 5.5 V**)**

Parameter	Symbol	Variables	MIN.	TYP.	MAX.	Unit
Resolution			8	8	8	bit
Conversion total error					±3.0	LSB
Conversion time	T_{CONV}	$21*12*(1/f_x)$		35		µs
Sampling time	T_{SAMP}	$4.5*12*(1/f_x)$	$15/f_{XX}$	7.5		µs
Analog input voltage	T_{IAN}		AV_{SS}-0.2		AV_{DD}+0.2	V

PLL CHARACTERISTIC (T_A = -40° to +85°, V_{DD} = 4.5 to 5.5 V)

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Operating Frequency	f_{IN1}	VCOL Pin MF/HF Mode Sine wave input V_{IN} = 0.1 V_{P-P}	0.5		55	MHz
	f_{IN2}	VCOH Pin VHF Mode Sine wave input V_{IN} = 0.1 V_{P-P}	60		160	MHz

IFC CHARACTERISTIC (T_A = -40° to +85°, V_{DD} = 4.5 to 5.5 V)

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Operating Frequency	f_{IN4}	AMIFC Pin AMIF Count Mode Sine wave input V_{IN} = 0.1 V_{P-P} NOTE	0.4		0.5	MHz
	f_{IN5}	FMIFC Pin FMIF Count Mode Sine wave input V_{IN} = 0.1 V_{P-P} NOTE	10		11	MHz
	f_{IN6}	FMIFC Pin AMIF Count Mode Sine wave input V_{IN} = 0.1 V_{P-P} NOTE	0.4		0.5	MHz

Note The condition of a sine wave input of V_{IN} = 0.1 V_{P-P} is the standard value of operation of this device during stand-alone operation, so in consideration of the effect of noise, it is recommended that operation be at an input amplitude condition of V_{IN} = 0.15 V_{P-P}.

5.3 DC Characteristics

Power Specification (HMS 91C8032)

Parameter	Symbol	Test Condition	Typ.	Max.	Unit
Active Mode	I_{DD}	RESET is high (Xtal1 = 7.2 MHz)	8	10	mA
Idle Mode	I_{DD}	CPU stops, Only timer works (Xtal1 = 32.768KHz)	0.8	1	mA
Power Down Mode	I_{DD}	Xtin1, Xtin2 Stuck at VSS	0.5	1	μA

Power Specification (HMS 97C8032)

Parameter	Symbol	Test Condition	Typ.	Max.	Unit
Active Mode	I_{DD}	RESET is high (Xtal1 = 7.2 MHz)	13	16	mA
Idle Mode	I_{DD}	CPU stops, Only timer works (Xtal1 = 32.768KHz)	1.3	2	mA
Power Down Mode	I_{DD}	Xtin1, Xtin2 Stuck at VSS	0.5	1.5	μA

Port Type 1 (P0, P1, P2.4 – P2.7, P3.5 – P3.7, P4.7, P5.3, P5.6, P6)

Parameter	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Input Voltage High	V_{IH}		0.7 V_{DD}		V_{DD}	V
Input Voltage Low	V_{IL}		0		0.3 V_{DD}	V
Output Voltage High	V_{OH}	I_{OH} = -1mA	V_{DD}-1.0			V
		I_{OH} = -100uA	V_{DD}-0.5			V
Output Voltage Low	V_{OL}(P0)	I_{OL} = 15mA		1.0	2.0	V
	V_{OL}(Others)	I_{OL} = 1.6mA			0.4	V
Leakage Current High	I_{LH}	V = Vdd			3	uA
Leakage Current Low	I_{LL}	V = 0			-3	uA

Port Type 2 (P7)

Parameter	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Input Voltage High	V_{IH}		0.7 V_{DD}		V_{DD}	V
Input Voltage Low	V_{IL}		0		0.3 V_{DD}	V
Output Voltage High	V_{OH}	I_{OH} = -1mA	V_{DD}-1.0			V
		I_{OH} = -100uA	V_{DD}-0.5			V
Output Voltage Low	V_{OL}	I_{OL} = 1.6mA			0.4	V
Leakage Current High	I_{LH}	V = Vdd			3	uA
Leakage Current Low	I_{LL}	V = 0			-3	uA

Port Type 3 (P3.0 – P3.4, P4.0 – P4.6, P5.0, P5.1, P5.2, P5.4, P5.5, P5.7)

Parameter	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Input Voltage High	V_{IH}		0.8 V_{DD}		V_{DD}	V
Input Voltage Low	V_{IL}		0		0.2 V_{DD}	V
Output Voltage High	V_{OH}	I_{OH} = -1mA	V_{DD}-1.0			V
		I_{OH} = -100uA	V_{DD}-0.5			V
Output Voltage Low	V_{OL}	I_{OL} = 1.6mA			0.4	V
Leakage Current High	I_{LH}	V = Vdd			3	uA
Leakage Current Low	I_{LL}	V = 0			-3	uA

Port Type 4 (P2.0 – P2.3)

Parameter	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Input Voltage High	V_{IH}		0.7 V_{DD}		V_{DD}	V
Input Voltage Low	V_{IL}		0		0.3 V_{DD}	V
Output Voltage High	V_{OH}	I_{OH} = 15mA		5.0	6.0	V
Output Voltage Low	V_{OL}	I_{OL} = 15mA		1.0	2.0	V
Leakage Current High	I_{LH}	V = Vdd			3	uA
Leakage Current Low	I_{LL}	V = 0			-3	uA

6. INSTRUCTION DEFINITIONS

6.1 Instruction Set Summary

Interrupt Response Time : Refer to Hardware Description Chapter

Instructions that Affect Flag Settings(1)

Instruction	Flag C	OV	AC	Instruction	Flag C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1) Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn - Register R7-R0 of the currently selected Register Bank.

direct - 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].

@Ri - 8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.

#data - 8-bit constant included in instruction.

#data 16 - 16-bit constant included in instruction.

addr 16 - 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 - 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel - Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.

bit - Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	OSC Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A, #data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A, #data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A, #data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24

Instruction Set Summary (Continued)

Mnemonic		Description	Byte	OSC Period
ARITHMETIC OPERATIONS (Continued)				
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A, #data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	OSC Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
AR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move indirect byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12

Instruction Set Summary (Continued)

Mnemonic		Description	Byte	OSC Period
DATA TRANSFER (Continued)				
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to Carry	2	24
ANL	C,/bit AND	Complement of direct bit to Carry	2	24
ORL	C,bit OR	direct bit to Carry	2	24
ORL	C,/bit OR	Complement of direct bit to Carry	2	24
MOV	C,bit Move	direct bit to Carry	2	12
MOV	bit,C Move	Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry is not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24

Mnemonic		Description	Byte	OSC Period
PROGRAM BRANCHING				
ACALL	addr 11	Absolute Subroutine Call	2	24
LCALL	addr 16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr 11	Absolute Jump	2	24
LJMP	addr 16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	3	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

6.2 Instruction Definitions

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The subroutine called must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07H. The label "SUBRTN" is at program memory location 0345H. After executing the instruction,

ACALL SUBRTN

at location 0123H, SP will contain 09H, internal RAM locations 08H and 09H will contain 25H and 01H, respectively, and the PC will contain 0345H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: ACALL

(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A,<src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B). The instruction,

ADD A,R0

will leave 6DH (01101101B) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	1 r r r

Operation: ADD
(A) ← (A) + (Rn)

ADD A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 1	direct address

Operation: ADD
(A) ← (A) + (direct)

ADD A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 1 1 i

Operation: ADD
(A) ← (A) + (Ri)

ADD A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 0	0 1 0 0	immediate data

Operation: ADD
(A) ← (A) + #data

ADDC A,<src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) with the carry flag set. The instruction,

ADDC A,R0

will leave 6EH (01101110B) in the Accumulator with AC cleared and both the Carry flag and OV set to 1.

ADDC A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	1 r r r

Operation: ADDC

(A) ← (A) + (C) + (R_n)

ADDC A,direct

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 1	direct address

Operation: ADDC

(A) ← (A) + (C) + (direct)

ADDC A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 1 1 i

Operation: ADDC
(A) ← (A) + (C) + ((Ri))

ADDC A,#data

Bytes: 2

Cycles: 1

Encoding:

0 0 1 1	0 1 0 0	immediate data

Operation: ADDC
(A) ← (A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123H. The instruction,

AJMP JMPADR

is at location 0345H and will load the PC with 0123H.

Bytes: 2

Cycles: 2

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: AJMP
(PC) ← (PC) + 2
(PC_{10-0}) ← page address

ANL <dest-byte> , <src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 55H (01010101B) then the instruction,

ANL A,R0

will leave 41H (01000001B) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1,#01110011B

will clear bits 7, 3, and 2 of output port 1.

ANL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	1 r r r

Operation: ANL
(A) ← (A) ∧ (Rn)

ANL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 1	direct address

Operation: ANL
(A) ← (A) ∧ (direct)

ANL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 1	0 1 1 i

Operation: ANL
(A) ← (A) ∧ ((Ri))

ANL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 1 0 0	immediate data

Operation: ANL
(A) ← (A) ∧ #data

ANL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 0 1	0 0 1 0	direct address

Operation: ANL
(direct) ← (direct) ∧ (A)

ANL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 0 1	0 0 1 1	direct address	immediate data

Operation: ANL
(direct) ← (direct) ∧ #data

ANL C,<src-bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```
MOV  C,P1.0     ;LOAD CARRY WITH INPUT PIN STATE

ANL   C,ACC.7  ;AND CARRY WITH ACCUM.  BIT 7

ANL   C,/OV      ;AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C,bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address

Operation: ANL

(C) ← (C) ∧ (bit)

ANL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address

Operation: ANL

(C) ← (C) ∧ ¬ (bit)

CJNE <dest-byte>,<src-byte>,rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34H. Register 7 contains 56H. The first instruction in the sequence,

```
                CJNE    R7,#60H,NOT_EQ
;               ...     .....                   ; R7 = 60H.
NOT_EQ:         JC      REQ_LOW                 ; IF R7 < 60H.
;               ...     .....                   ; R7 > 60H.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then the instruction,

```
WAIT:   CJNE  A,P1,WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34H.)

CJNE A,direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 1	direct address	rel. address

Operation:

(PC) ← (PC) + 3
IF (A) < > (direct)
THEN
(PC) ← (PC) + ***relative offset***

IF (A) < (direct)
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE A,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 0 0	immediate data	rel. address

Operation:

(PC) ← (PC) + 3
IF (A) < > data
THEN
(PC) ← (PC) + relative offset

IF (A) < data
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE Rn,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	1 r r r	immediate data	rel. address

Operation:

(PC) ← (PC) + 3
IF (Rn) < > data
THEN
(PC) ← (PC) + relative offset

IF (Rn) < data
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE @Ri,#data,rel

Bytes: 3

Cycles: 2

Encoding:

1 0 1 1	0 1 1 i	immediate data	rel. address

Operation: **(PC) ← (PC) + 3**
IF ((Ri)) < > data
THEN
(PC) ← (PC) + relative offset

IF (Ri) < data
THEN
(C) ← 1
ELSE
(C) ← 0

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits set on zero). No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CLR A

will leave the Accumulator set to 00H (00000000B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 0 0

Operation: CLR
(A) ← 0

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5DH (01011101B). The instruction,

CLR P1.2

will leave the port set to 59H (01011001B).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 0 1 1

Operation: CLR
(C) ← 0

CLR bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 0 1 0	bit address

Operation: CLR
(bit) ← 0

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5CH (01011100B). The instruction,

CPL A

will leave the Accumulator set to 0A3H (10100011B).

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	0 1 0 0

Operation: CPL
(A) ← ¬ (A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CLR can operate on the carry or any directly addressable bit.

Note.- When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: Port 1 has previously been written with 5BH (01011011B). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5BH (01011011B).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1 0 1 1	0 0 1 1

Operation: CPL
(C) ← ¬ (C)

CPL bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 1	0 0 1 0	bit address

Operation: CPL
(bit) ← ¬ (bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carry-out of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H. or 66H to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A cannot simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56H (01010110B) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67H (01100111B) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC   A,R3
DA A
```

will first perform a standard twos-complement binary addition, resulting in the value 0BEH (10111110B) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the Accumulator initially holds 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD   A, # 99H
DA A
```

will leave the carry set and 29H in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 - 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 0 0

Operation:

DA
-contents of Accumulator are BCD
IF [[(A_{3-0}) > 9] V [(AC) = 1]]
THEN$(A_{3-0}) \leftarrow (A_{3-0}) + 6$
AND

IF [[(A_{7-4}) > 9] V [(C) = 1]]
THEN $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00H will underflow to 0FFH. No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7FH (01111111B). Internal RAM locations 7EH and 7FH contain 00H and 40H, respectively. The instruction sequence,

DEC @R0

DEC R0

DEC @R0

will leave register 0 set to 7EH and internal RAM locations 7EH and 7FH set to 0FFH and 3FH.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 0 0

Operation: DEC
(A) ← (A) - 1

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	1 r r r

Operation: DEC
(Rn) ← (Rn) - 1

DEC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 1	0 1 0 1	direct address

Operation: DEC
(direct) ← (direct) - 1

DEC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 1 i r

Operation: DEC
((Ri)) ← ((Ri)) - 1

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00H, the values returned in the Accumulator and B-register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBH or 11111011B) and B contains 18 (12H or 00010010B). The instruction,

DIV AB

will leave 13 in the Accumulator (0DH or 00001101B) and the value 17 (11H or 00010001B) in B, since 251 = (13 X 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 4

Encoding:

1 0 0 0	0 1 0 0

Operation: DIV
$(A)_{15-8}$ ← (A)/(B)
$(B)_{7-0}$

DJNZ <byte>,<rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00H will underflow to 0FFH. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note.- When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Internal RAM locations 40H, 50H, and 60H contain the values 01H, 70H, and 15H. respectively. The instruction sequence,

```
DJNZ  40H,LABEL_1
DJNZ  50H,LABEL_2
DJNZ  60H,LABEL_3
```

will cause a jump to the instruction at label LABEL-2 with the values 00H, 6FH, and 15H in the three RAM locations. The first jump was not taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
MOV R2,#8
TOGGLE:  CPL P1.7
DJNZ R2,TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1.
Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn,rel

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	1 r r r	rel. address

Operation: DJNZ

(PC) ← (PC) + 2
(Rn) ← (Rn) - 1
IF (Rn) > 0 or (Rn) < 0
THEN
(PC) ← (PC) + rel

DJNZ direct,rel

Bytes: 3

Cycles: 2

Encoding:

1 1 0 1	0 1 0 1	direct address	rel. address

Operation: DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) - 1
IF (direct) > 0 or (direct) < 0
THEN
(PC) ← (PC) + rel

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFH will overflow to 00H. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note.- When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7EH (01111110B). Internal RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. The instruction sequence,

```
INC   @R0
INC   R0
INC   @R0
```

will leave register 0 set to 7FH and internal RAM locations 7EH and 7FH holding (respectively) 00H and 41H.

INC A

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 0 0

Operation: INC
(A) ← (A) + 1

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	1 r r r

Operation: INC
(Rn) ← (Rn) + 1

INC direct

Bytes: 2

Cycles: 1

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation: INC
(direct) ← (direct) + 1

INC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation: INC
((Ri)) ← ((Ri)) + 1

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 216) is per-formed; an overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12H and 0FEH. respectively. The instruction sequence,

```
INC DPTR
INC DPTR
INC DPTR
```

will change DPH and DPL to 13H and 01H.

Bytes: 1

Cycles: 2

Encoding:

1 0 1 0	0 0 1 1

Operation: INC
(DPTR) ← (DPTR) + 1

JB bit,rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010B. The Accumulator holds 56 (01010110B). The instruction sequence,

```
JB  P1.2,LABEL1

JB  ACC.2,LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 0	0 0 0 0	bit address	rel. address

Operation:

JB
(PC) ← (PC) + 3
IF (bit) = 1
THEN
(PC) ← (PC) + rel

JBC bit,rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. The bit will not be cleared if it is already a zero. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.
Note.- When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: The Accumulator holds 56H (01010110B). The instruction sequence,

```
JBC  ACC.3,LABEL1
JBC  ACC.2,LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52H (01010010B).

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 0 0	bit address	rel. address

Operation:

JBC
(PC) ← (PC) + 3
IF (bit) = 1
THEN
(bit) ← 0
(PC) ← (PC) + rel

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC LABEL1
CPL C
JC LABEL 2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 0	0 0 0 0	rel. address

Operation:

JC
(PC) ← (PC) + 2
IF (C) = 1
THEN
(PC) ← (PC) + rel

JMP @A + DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
            MOV   DPTR, # JMP_TBL
            JMP   @A + DPTR
JMP_TBL:    AJMP  LABEL0
            AJMP  LABEL1
            AJMP  LABEL2
            AJMP  LABEL3
```

If the Accumulator equals 04H when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0 1 1 1	0 0 1 1

Operation:

JMP
(PC) ← (A) + (DPTR)

JNB bit,rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port I is 11001010B. The Accumulator holds 56H (01010110B). The instruction sequence,

```
JNB   P1.3,LABEL1
JNB   ACC.3,LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 2

Encoding:

0 0 1 1	0 0 0 0	bit address	rel. address

Operation:

JNB
(PC) ← (PC) + 3
IF (bit) = 0
THEN
(PC) ← (PC) + rel

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC LABEL1
CPL C
JNC LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 0 1	0 0 0 0	rel. address

Operation:

JNC
(PC) ← (PC) + 2
IF (C) = 0
THEN
(PC) ← (PC) + rel

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00H. The instruction sequence,

```
JNZ   LABEL1
INC   A
JNZ   LABEL2
```

will set the Accumulator to 01H and continue at label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 0 0	rel. address

Operation:

```
JNZ
(PC) ← (PC) + 2
IF  (A) _ 0
    THEN
        (PC) ← (PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the address indicated otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01H. The instruction sequence,

```
JZ    LABEL1
DEC   A
JZ    LABEL2
```

will change the Accumulator to 00H and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 2

Encoding:

0 1 1 0	0 0 0 0	rel. address

Operation:

```
JZ
(PC) ← (PC) + 2
IF  (A) = 0
    THEN
        (PC) ← (PC) + rel
```

LCALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64K-byte program memory address space No flags are affected.

Example: Initially the Stack Pointer equals 07H. The label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction,

LCALL SUBRTN

at location 0123H, the Stack Pointer will contain 09H, internal RAM locations 08H and 09H will contain 26H and 01H, and the PC will contain 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 1	0 0 1 0	addr15-addr8	addr7-addr0

Operation: LCALL

(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) +1
((SP)) ← (PC_{15-8})
(PC) ← $addr_{15-0}$

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64K program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H- The instruction,

LJMP JMPADR

at location 0123H will load the program counter with 1234H.

Bytes: 3

Cycles: 2

Encoding:

0 0 0 0	0 0 1 0	addr15-addr8	addr7-addr0

Operation: LJMP

(PC) ← $addr_{15-0}$

MOV <dest-byte>,<src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30H holds 40H. The value of RAM location 40H is 10H. The data present at input port I is 11001010B (0CAH).

```
MOV  R0, # 30H ;R0 < = 30H
MOV A,@R0 ;A <= 40H
MOV R1,A ;R1 < = 40H
MOV B,@R1 ;B < = 10H
MOV @R1,P1 ;RAM (40H) < = 0CAH
MOV P2,P1 ;P2 #0CAH
```

leaves the value 30H in register 0, 40H in both the Accumulator and register 1, 10H in register B, and 0CAH (11001010B) both in RAM location 40H and output on port 2.

MOV A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	1 r r r

Operation: MOV
(A) ← (Rn)

***MOV A,direct**

Bytes: 2

Cycles: 1

Encoding:

1 1 1 0	0 1 0 1	direct address

Operation: MOV
(A) ← (direct)

MOV A, ACC is not a vaild instruction.

MOV A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 1 0	0 1 1 i

Operation: MOV
(A) ← ((Ri))

MOV A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 1	0 1 0 0	immediate data

Operation: MOV
(A) ← #data

MOV Rn,A

Bytes: 1

Cycles: 1

Encoding:

1 1 1 1	1 r r r

Operation: MOV
(Rn) ← (A)

MOV Rn,direct

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	1 r r r	direct addr.

Operation: MOV
(Rn) ← (direct)

MOV Rn,#dara

Bytes: 2
Cycles: 1

Encoding:

0 1 1 1	1 r r r	immediate data

Operation: MOV
(A) ← #data

MOV direct,A

Bytes: 2
Cycles: 1

Encoding:

1 1 1 1	0 1 0 1	direct address

Operation: MOV
(direct) ← (A)

MOV direct,Rn

Bytes: 2
Cycles: 2

Encoding:

1 0 0 0	1 r r r	direct address

Operation: MOV
(direct) ← (Rn)

MOV direct,direct

Bytes: 3
Cycles: 1

Encoding:

1 1 1 0	0 1 0 1	direct addr.(src)	dir. addr.(dest)

Operation: MOV
(direct) ← (direct)

MOV direct,@Ri

Bytes: 2
Cycles: 2

Encoding:

1 0 0 0	0 1 1 i	direct address

Operation: MOV
(direct) ← ((Ri))

MOV direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 1	0 1 0 1	direct address	immediate data

Operation: MOV
(direct) ← #data

MOV <dest-bit>,<src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101IB. The data previously written to output Port 1 is 35H (00110101B).

```
MOV  P1.3,C
MOV  C,P3.3
MOV  P1.2,C
```

will leave the carry cleared and change Port I to 39H (00111001B).

MOC C,bit

Bytes: 2

Cycles: 1

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation: MOV
(C) ← (bit)

MOV bit,C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation: MOV
(bit) ← (C)

MOV DPTR,#data16

Function:	Load Data Pointer with a 16-bit constant
Description:	The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected. This is the only instruction which moves 16 bits of data at once.
Example:	The instruction, MOV DPTR, # 1234H will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.
Bytes:	3
Cycles:	2

Encoding:

1 0 0 1	0 0 0 0	immed. data15-8	immed. data7-0

Operation: MOV

(DPTR) ← #data$_{15\text{-}0}$

DPH □ DPL ← #data$_{15\text{-}8}$ □ #data$_{7\text{-}0}$

MOV A,@A + <base-reg>

Function:	Move Code byte
Description:	The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.
Example:	A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive. REL_PC: INC A MOVC A,@A+PC RET DB 66H DB 77H DB 88H DB 99H If the subroutine is called with the Accumulator equal to 01H, it will return with 77H in the Accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A,@A + PC

Bytes: 1

Cycles: 2

Encoding:

1 0 0 0	0 0 1 1

Operation: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFH) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50H). Register B holds the value 160 (0A0H). The instruction,

MUL AB

will give the product 12,800 (3200H), so B is changed to 32H (00110010B) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 4

Encoding:

1 0 1 0	0 1 0 0

Operation: MUL
$(A)_{7-0}$ ← (A) X (B)
$(B)_{15-8}$

NOP

Function:	No Operation
Description:	Execution continues at the following instruction. Other than the PC, no registers or flags are affected.
Example:	It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence, CLR P2.7 NOP NOP NOP NOP SETB P2.7
Bytes:	1
Cycles:	1
Encoding:	0 0 0 0 | 0 0 0 0
Operation:	NOP **(PC) ← (PC) + 1**

ORL <dest-byte>,<src-byte>

Function:	Logical-OR for byte variables
Description:	ORL Performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data. Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.
Example:	If the Accumulator holds 0C3H (11000011B) and R0 holds 55H (01010101B) then the instruction, ORL A,R0 will leave the Accumulator holding the value 0D7H (11010111B). When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction, ORL P1,#00110010B will set bits 5, 4, and 1 of output Port 1.

ORL A,Rn

Bytes: 1
Cycles: 1

Encoding:

0 1 0 0	1 r r r

Operation: ORL
(A) ← (A) ∨ (Rn)

ORL A,direct

Bytes: 2
Cycles: 1

Encoding:

0 1 0 0	0 1 0 1	direct address

Operation: ORL
(A) ← (A) ∨ (direct)

ORL A,@Ri

Bytes: 1
Cycles: 1

Encoding:

0 1 0 0	0 1 1 i

Operation: ORL
(A) ← (A) ∨ ((Ri))

ORL A,#data

Bytes: 2
Cycles: 1

Encoding:

0 1 0 0	0 1 0 0	immediate data

Operation: ORL
(A) ← (A) ∨ #data

ORL direct,A

Bytes: 2
Cycles: 1

Encoding:

0 1 0 0	0 0 1 0	direct address

Operation: ORL
(direct) ← (direct) ∨ (A)

ORL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 0 0	0 0 1 1	direct address	immediate data

Operation: ORL
(direct) ← (direct) ∨ #data

ORL C,<src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise . A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV  C,P1.0 ;LOAD CARRY WITH INPUT PIN P10

ORL  C,ACC.7 ;OR CARRY WITH THE ACC. BIT 7

ORL  C,/OV ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C,bit

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation: ORL
(C) ← (C) ∨ (bit)

ORL C,/bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation: ORL
(C) ← (C) ∨ (bit)

POP direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by one. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32H, and internal RAM locations 30H through 32H contain the values 20H, 23H, and 01H, respectively. The instruction sequence,

```
POP  DPH
POP  DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123H. At this point the instruction,

```
POP  SP
```

will leave the Stack Pointer set to 20H. Note that in this special case the Stack Pointer was decremented to 2FH before being loaded with the value popped (20H).

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	0 0 0 0	direct address

Operation:
POP
(direct) ← ((SP))
(SP) ← (SP) - 1

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by one. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09H. The Data Pointer holds the value 0123H. The instruction sequence,

```
PUSH  DPL
PUSH  DPH
```

will leave the Stack Pointer set to 0BH and store 23H and 01H in internal RAM locations 0AH and 0BH, respectively.

Bytes: 2

Cycles: 2

Encoding:

1 1 0 0	0 0 0 0	direct address

Operation:
PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)

RET

Function:	Return from subroutine
Description:	RET pops the high- and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by two. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.
Example:	The Stack Pointer originally contains the value 0BH. Internal RAM locations 0AH and 0BH contain the values 23H and 01H. respectively. The instruction, RET will leave the Stack Pointer equal to the value 09H. Program execution will continue at location 0123H.
Bytes:	1
Cycles:	2
Encoding:	0 0 1 0 | 0 0 1 0
Operation:	RET **$(PC_{15-8}) \leftarrow ((SP))$** **$(SP) \leftarrow (SP) - 1$** **$(PC_{7-0}) \leftarrow ((SP))$** **$(SP) \leftarrow (SP) - 1$**

RETI

Function:	Return from interrupt
Description:	RETI pops the high- and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is not automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower- or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.
Example:	The Stack Pointer originally contains the value 0BH. An interrupt was detected during the instruction ending at location 0122H. Internal RAM locations 0AH and 0BH contain the values 23H and 01H, respectively. The instruction, RETI will leave the Stack Pointer equal to 09H and return program execution to location 0123H.
Bytes:	1
Cycles:	2
Encoding:	0 0 1 1 | 0 0 1 0
Operation:	RETI **$(PC_{15-8}) \leftarrow ((SP))$** **$(SP) \leftarrow (SP) - 1$** **$(PC_{7-0}) \leftarrow ((SP))$** **$(SP) \leftarrow (SP) - 1$**

RL A

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RL A

leaves the Accumulator holding the value 8BH (10001011IB) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0 0 1 0	0 0 1 1

Operation: RL

$(A_{n+1}) \leftarrow (A_n)$, n = 0 - 6

(A0) ← (A7)

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), and the carry is zero. The instruction,

RLC A

leaves the Accumulator holding the value 8BH (10001011B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0 0 1 1	0 0 1 1

Operation: RLC

$(An_{+1}) \leftarrow (A_n)$, n = 0 - 6

(A0) ← (C)

(C) ← (A7)

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B). The instruction,

RR A

leaves the Accumulator holding the value 0E2H (111100010B) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 0 1 1

Operation: RR
$(A_n) \leftarrow (A_{n+1})$, n = 0 - 6
(A7) ← (A0)

RRC A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5H (11000101B), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62 (01100010B) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 0 1 1

Operation: RRC
$(A_n) \leftarrow (A_{n+1})$, n = 0 - 6
(A7) ← (C)
(C) ← (A0)

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34H (00110100B). The instructions,

SETB C

SETB P1.0

will leave the carry flag set to I and change the data output on Port I to 35H (00110101B).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 0 1 1

Operation: SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 1

Encoding:

1 1 0 1	0 0 1 0	bit address

Operation: SETB
(bit) ← 1

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction,

SJMP RELADR

will assemble into location 0100H. After the instruction is executed, the PC will contain the value 0123H.

(Note.- Under the above conditions the instruction following SJMP will be at 102H. Therefore, the displacement byte of the instruction will be the relative offset (0123H-0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 0 0	rel. address

Operation: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A,<src-byte>4

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C9H (11001001B), register 2 holds 54H (01010100B), and the carry flag is set. The instruction,

SUBB A,R2

will leave the value 74H (01110100B) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that OC9H minus 54H is 75H. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A,Rn

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	1 r r r

Operation: SUBB
(A) ← (A) - (C) - (Rn)

SUBB A,direct

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1	0 1 0 1	direct address

Operation: SUBB
(A) ← (A) - (C) - (direct)

SUBB A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 1 i

Operation: SUBB
(A) ← (A) - (C) - ((Ri))

SUBB A,#data

Bytes: 2

Cycles: 1

Encoding:

1 0 0 1	0 1 0 0	immediate data

Operation: SUBB
(A) ← (A) - (C) - #data

SWAP A

Function:	Swap nibbles within the Accumulator
Description:	SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation ran also be thought of as a four-bit rotate instruction. No flags are affected.
Example:	The Accumulator holds the value 0C5H (11000101B). The instruction, SWAP A leaves the Accumulator holding the value 5CH (01011100B).
Bytes:	1
Cycles:	1
Encoding:	1 1 0 0 | 0 1 0 0
Operation:	SWAP (A_{3-0}) (A_{7-4})

XCH A,<byte>

Function:	Exchange Accumulator with byte variable
Description:	XCH loads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.
Example:	R0 contains the address 20H. The Accumulator holds the value 3FH (00111111B). Internal RAM location 20H holds the value 75H (01110101B). The instruction, XCH A,@R0 will leave RAM location 20H holding the value 3FH (00111111B) and 75H (01110101B) in the Accumulator.

XCH A,Rn

Bytes:	1
Cycles:	1
Encoding:	1 1 0 0 | 1 r r r
Operation:	XCH (A) $\rightleftharpoons$ (Rn)

XCH A,direct

Bytes: 2

Cycles: 1

Encoding:

1 1 0 0	0 1 0 1	direct address

Operation: XCH
(A) $\rightleftarrows$ (direct)

XCH A,@Ri

Bytes: 1

Cycles: 1

Encoding:

1 1 0 0	0 1 1 i

Operation: XCH
(A) $\rightleftarrows$ ((Ri))

XCHD A,@Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bit 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bit7-4) of each register are not affected. No flags are affected.

Example: R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction,

XCHD A,@R0

will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 1 1 i

Operation: XCHD
(A_{3-0}) $\rightleftarrows$ ((Ri_{3-0}))

XRL <dest-byte>,<src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flag are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.)

Example: If the Accumulator holds 0C3H (11000011B) and register 0 holds 0AAH (10101010B) then the instruction,

XRL A,R0

will leave the Accumulator holding the value 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL P1,#00110001B

will complement bits 5, 4, and 0 of output Port 1.

XRL A,Rn

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	1 r r r

Operation: XRL

(A) ← (A) ⩛ (Rn)

XRL A,direct

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 1	direct address

Operation: XRL

(A) ← (A) ⩛ (direct)

XRL A,@Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 1 0	0 1 1 i

Operation: XRL

(A) ← (A) ⩛ ((Ri))

XRL A,#data

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 1 0 0	immediate data

Operation: XRL

(A) ← (A) ⩛ #data

XRL direct,A

Bytes: 2

Cycles: 1

Encoding:

0 1 1 0	0 0 1 0	direct address

Operation: XRL

(direct) ← (direct) ⩛ (A)

XRL direct,#data

Bytes: 3

Cycles: 2

Encoding:

0 1 1 0	0 0 1 1	direct address	immediate data

Operation: XRL

(direct) ← (direct) ⩛ #data

7. EPROM CHARACTERISTICS

The HMS97C8032 has internal 32K bytes OTP ROM. The HMS97C8032 is programmed with a modified quick-pulse programming™ algorithm. It differs from older methods in the value used for V_{PP} (programming supply voltage) and in the width and number of the ALE/ pulses. The HMS97C8032 contains two signature bytes that can be read and used by an EPROM programming system to identify the device. The signature bytes identify the device as a manufactured by HEI. Figure 7-1 shows the logic levels for reading the signature bytes. The circuit configuration is shown in Figure 7-2. For programming the program memory, the encryption table, and the lock bits, refer Figure 7-4 and Figure 7-5. Figure 7-3 shows the circuit configuration for normal program memory verification.

7.1 Reading the Signature Bytes:

The HMS97C8032 signature bytes are in locations 030_H and 060_H. To read these bytes, refer Figure 7-1. Location of each signature byte should be represented by 15 bits address. In Figure 7-1, P0[7:0] and P1[6:0] receive lower 8 bits and higher 7 bits of the 15 bits address, respectively. Signature value is read through P6[7:0]. For timing parameters, refer Table 7-3. The row labeled "Read Signature Byte" in Table 7-2 defines the valid states of "CONTROL SIGNALS" in Figure 7-1.

The following table defines the signature values of HMS97C8032 :

Device	Location	Contents	Remarks
HMS97C8032	30_H 60_H	$E0_H$ 58_H	Manufacturer ID Device ID

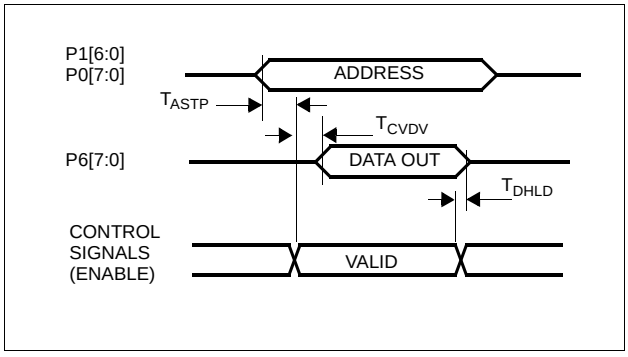


Figure 7-1 Real Signature Waveform

7.2 Modified Quick-Pulse Programming

The waveform for micro-controller quick-pulse programming is shown in Figure 7-4 (See the programming part of Figure 7-4). For timing parameters, refer Table 7-3. Note that the HMS97C8032 is running with a 4 to 6MHz oscillator. The reason the oscillator needs to be running is that the device is executing internal address and program data transfers. Note that the TSTEN/VPP pin must not be allowed to go above the maximum specified V_{PP} level for any amount of time. Even a narrow glitch above that voltage can cause permanent damage to the device. The V_{PP} source should be regulated and free glitches and overshoot.

U : un-programmed, P : programmed

Programming the code memory

The address of the EPROM location to be programmed is applied to P0[7:0] and P1[6:0](0000h ~7FFFh), as shown in Figure 7-2. The code byte to be programmed is applied to P6[7:0]. RESET, (P4.6) and pins of P4 are held at the "Program Code Data" levels indicated in Table 7-2. The P4.7/(ALE) is pulsed low 5 times as shown in Figure 3 to program code data. ***The initial value of every code memory byte is 00h***.

Programming the encryption table

To program the encryption table, the P4.7/(ALE) is pulsed low 25 times as shown in Figure 7-4. The address of the Encryption Array to be programmed is applied to P0[5:0] and a encryption byte to be programmed is applied to P6[7:0]. RESET, (P4.6) and pins of P4 are held at the "Program Encryption Array Address" levels indicated in Table 7-2.

Within the EPROM array are 64 bytes of Encryption Array(00h~3Fh) that are initially not programmed. Every time that a program memory byte is addressed during a verification or read operation, the lower 6 bits (P0[5:0]) of address lines are used to select a byte from the Encryption array. The encryption array byte is then exclusive-NORed (XNOR) with the code byte, creating an Encrypted Verify byte.

The initial value of every encryption byte is 00h. Thus, when a blank program memory byte is read, The HMS97C8032 will return FFh since the initial code value is 00h. It is recommended that whenever the Encryption Array is used, at least one of the Lock Bits be programmed as well.

Programming the lock bits

To program the lock bits, the P4.7/(ALE) is pulsed low 25 times as shown in Figure 7-4 using the "Program lock Bit" levels shown in Table 7-2. For lock bits programming, address and data are not required. After one of the lock bits is programmed, further programming of the code memory and encryption table is disabled. However, the other lock bit can still be programmed.

The following table shows function of each lock bit.

Mode	LB1	LB2	Protection Type
1	U	U	No program lock features
2	P	U	Further programming of the EPROM is Disabled
3	P	P	Same as mode 2, also verify is disabled

Table 7-1 Lock bit function

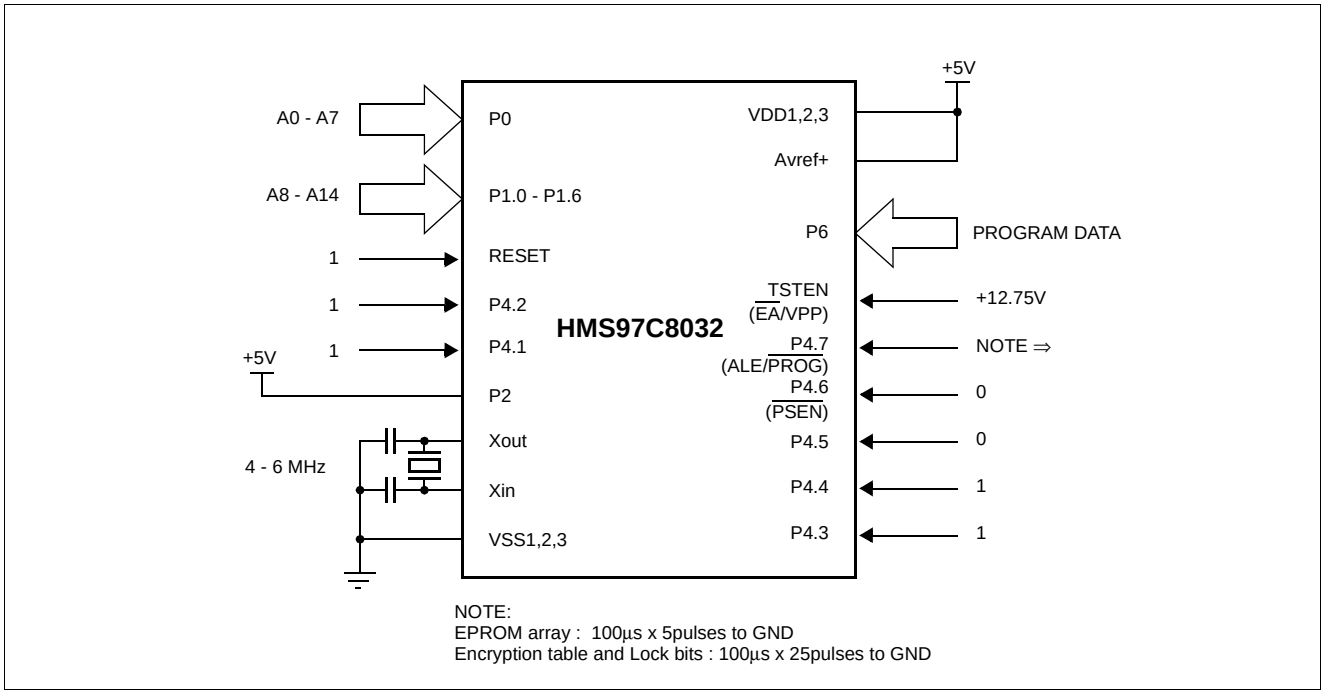
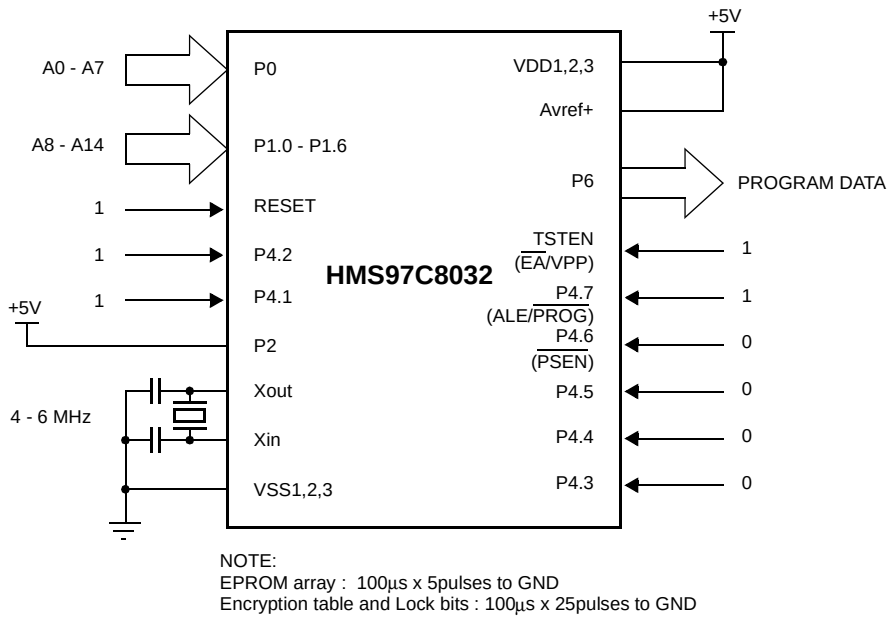


Figure 7-2 Programming Configuration

7.3 Program Verification

If lock bit 2 (LB2 in Table 7-1) has not been programmed, the on-chip program memory can be read out for program verification. The address of the program memory location to be read is applied to P0[7:0] and P1[6:0] (0000h ~7FFFh) as shown in Figure 7-3. The other pins are held at the “Verify Code Data” levels indicated in Table 7-2. The contents of the address location will be emitted on P6[7:0] for this operation. The value on P6[7:0] is always exclusive NORed value of the program code byte and corresponding encryption array byte. The lower 6 bits (P0[5:0]) of address lines are used to select a byte from the Encryption array(00h~3Fh). To restore original code byte, user should know the encryption table. The original code byte could be restored by doing exclusive NOR of the value on P6[7:0] and corresponding encryption array byte.

The encryption table itself cannot be read out. Figure 7-4 shows wave form of program verification waveform (see verification part). Figure 7-5 shows two consecutive program memory read waveform. For timing parameter, refer Table 7-3.



NOTE:
EPROM array : 100μs x 5pulses to GND
Encryption table and Lock bits : 100μs x 25pulses to GND

Figure 7-3 Program Verification

MODE		RESET	P4.6 (PSEN)	P4.7 (ALE)	TETEN (VPP)	P4.5	P4.4	P4.3	P4.2	P4.1
Program Code Data		H	L	⊓	12.75V	L	H	H	H	H
Verify Code Data		H	L	H	H	L	⊔	L	H	H
Program Encryption Array Address (00H ~ 3FH)		H	L	⊓	12.75V	L	H	H	L	H
Program Lock Bits	Bit 1	H	L	⊓	12.75V	H	H	H	H	H
	Bit 2	H	L	⊓	12.75V	H	H	H	L	L
Read Signature Byte		H	L	H	H	L	L	L	L	L

Table 7-2 EPROM programming modes

Notes:

"0" = Valid low for that pin, "1" = Valid high for that pin.

V_{PP} = 12.5V ± 0.25V

V_{CC} = 5V ± 10% during programming and verification.

ALE/ receives 5 (25 for encryption table and lock bits) programming pulses while VPP is held at 12.75V. Each programming pulse is low for 100us (±10us) and high for a minimum of 10us.

5. In "Verify Code Data" mode, the negative edge of P4.4 should be required.

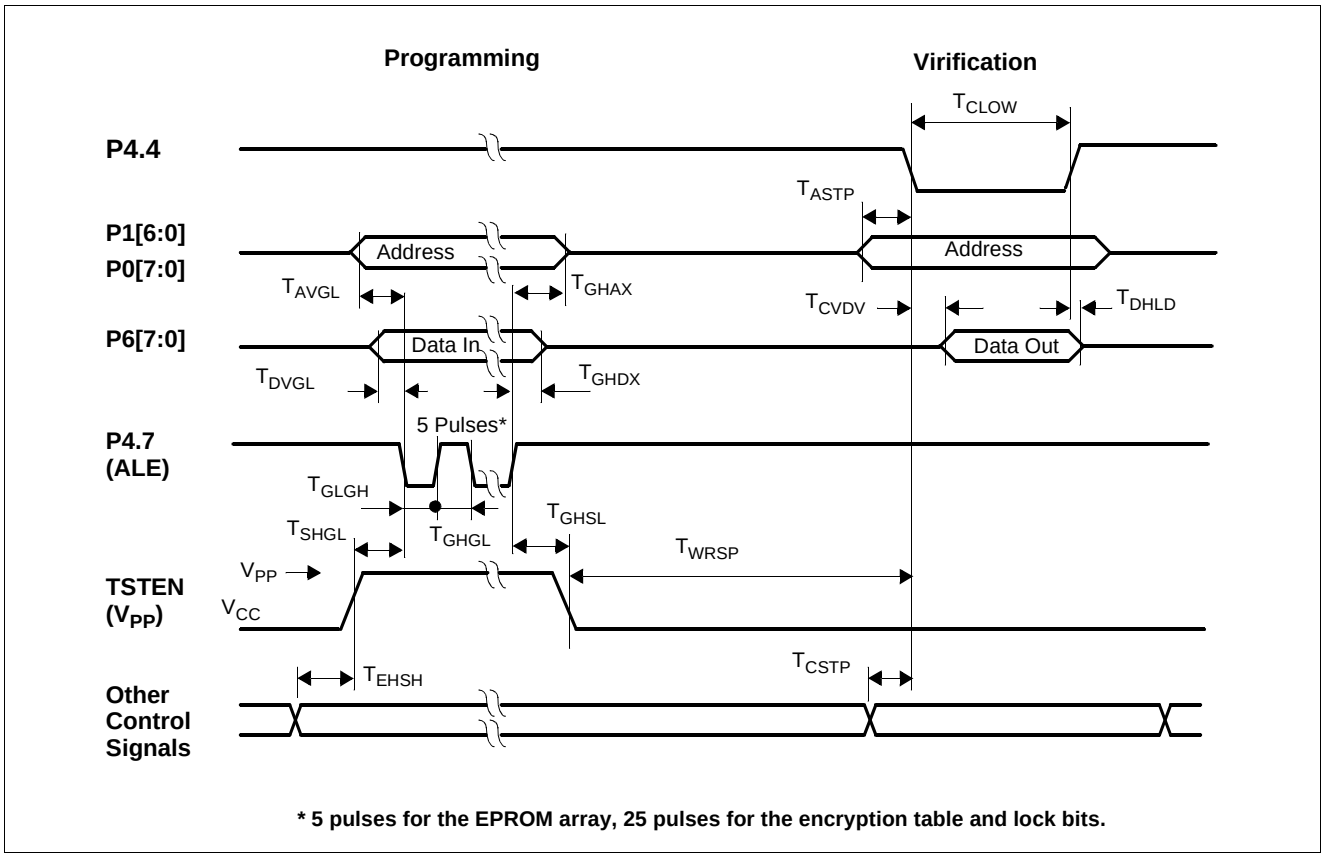


Figure 7-4 EPROM Programming and Verification

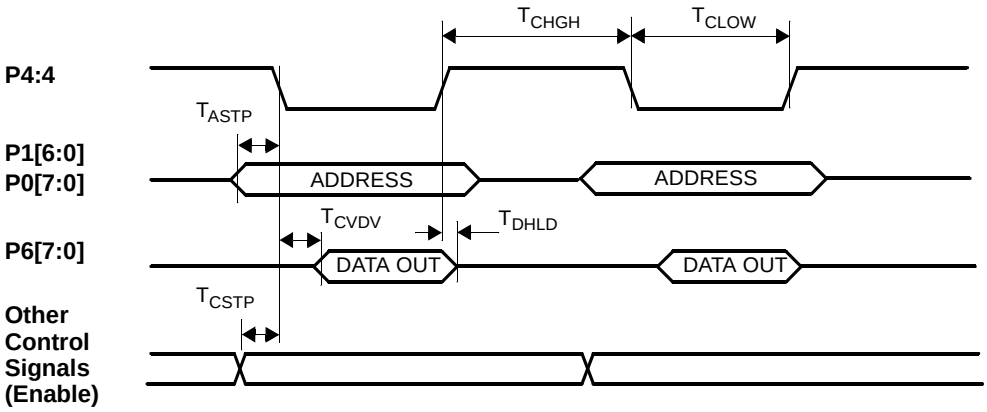


Figure 7-5 Two Consecutive Real Waveform

T_A=21°C to 27°C; V_{CC}= 5V±10%; V_{SS}=0V

Parameter	Symbol	Min	Max	Units
Programming Supply Voltage	V_{PP}	12.25	12.75	V
Programming Supply Current	I_{PP}		75	mA
Oscillator Frequency	$1/T_{CLCL}$	4	6	MH
Address Setup to P4.7(ALE) Low	T_{AVGL}	$48T_{CLCL}$		
Address Hold after P4.7(ALE) High	T_{GHAX}	$48T_{CLCL}$		
Data Setup to P4.7(ALE) Low	T_{DVGL}	$48T_{CLCL}$		
Data Hold after P4.7(ALE) High	T_{GHDX}	$48T_{CLCL}$		
P4.4 High to V_{PP}	T_{EHSH}	$48T_{CLCL}$		
V_{PP} Setup to P4.7(ALE) Low	T_{SHGL}	10		µs
V_{PP} Hold after P4.7(ALE) High	T_{GHSL}	10		µs
P4.7(ALE) Low Width	T_{GLGH}	90	110	µs
P4.7(ALE) High to P4.7(ALE) Low	T_{GHGL}	10		µs
Address Setup to P4.4	T_{ASTP}	2		µs
Control Setup to P4.4	T_{CSTP}	1		µs
Data Hold after P4.4	T_{DHLD}	0	0	µs
Data Valid after P4.4 Low	T_{CVDV}		$48T_{CLCL}$	
P4.4 Minimum High Duration	T_{CHGH}	10		µs
P4.4 Minimum Low Duration	T_{CLOW}	20		µs
Min separation between read and write	T_{WRSP}	300		µs

Figure 7-6 EPROM Programming and Verification Characteristics

8. OTP PROGRAMMING

8.1 HMS97C8032 OTP Programming

Blank Check

Since the initial values of program memory and encryption table memory are all 0s, the HMS97C8032 will return FFh if a blank program memory byte is read. We recommend the following blank check method for a not programmed HMS97C8032 chip.

1.Set every ROM writer program encryption array(00~3h) value to 00h.

2.Read a program memory byte from a HMS97C8032.

3.If the read value in step 2 is FFh, the program memory byte is blank.

Program writing

To burn program file, refer following procedure.

Make program OTP file.

Check blank.

Burn program OTP file (Set chip target address 0000h ~ 7FFFh)

Some ROM writers skip FFh data writing to program memory or encryption array, assuming that the initial value is FFh. But for the HMS97C8032 device , the initial value of program memory byte or encryption array is 00h, ***so you should not skip FFh data write to program memory or encryption array.***

8.2 Device Configuration Data

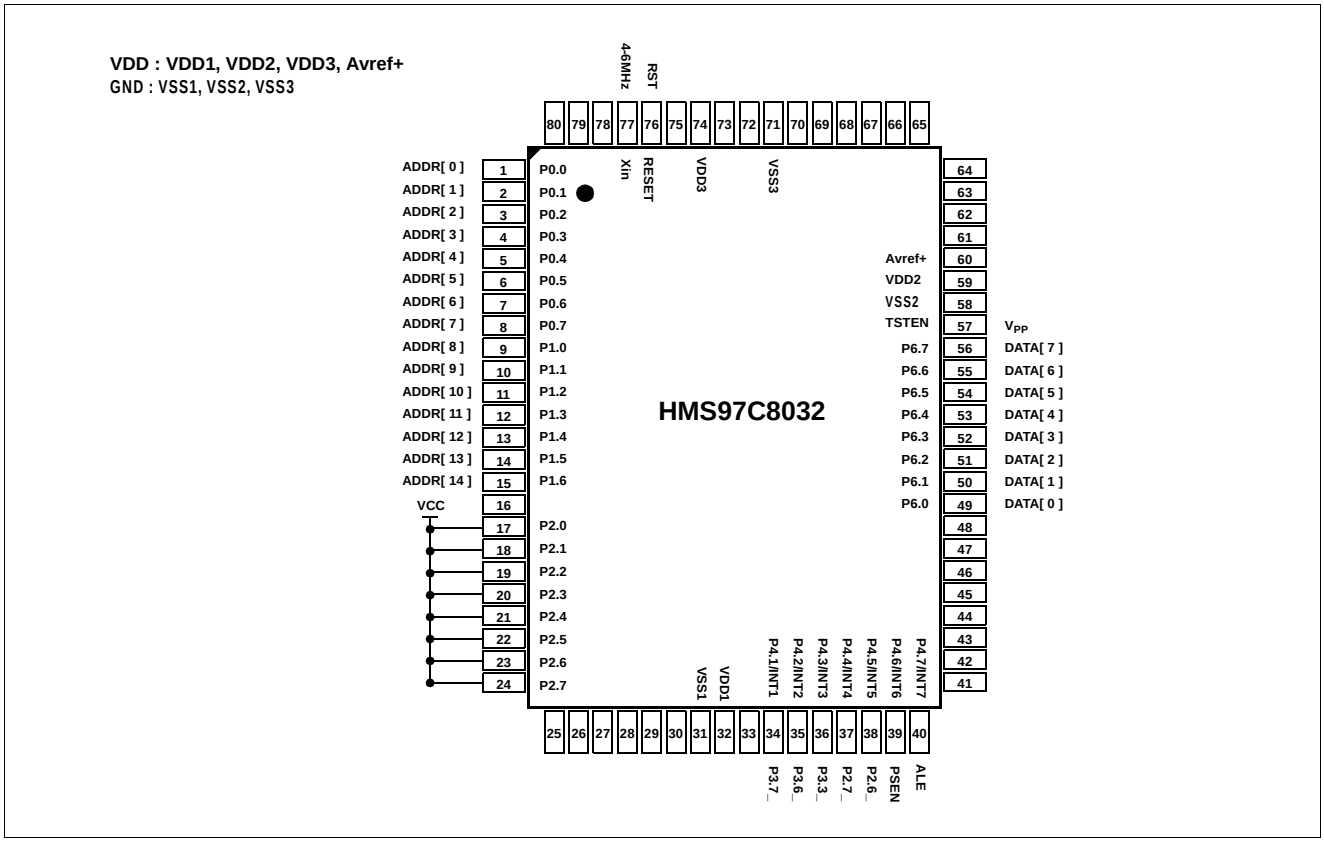


Figure 8-1 Pin Confiuration in OTP Diagram Mode

Intel87C58 (ADAPTER)		HMS97C8032	
P1.0(A0)	1	P0.0(A0)	1
P1.1(A1)	2	P0.1(A1)	2
P1.2(A2)	3	P0.2(A2)	3
P1.3(A3)	4	P0.3(A3)	4
P1.4(A4)	5	P0.4(A4)	5
P1.5(A5)	6	P0.5(A5)	6
P1.6(A6)	7	P0.6(A6)	7
P1.7(A7)	8	P0.7(A7)	8
RESET	9	RESET	76
P3.0	10		
P3.1	11		
P3.2	12		
P3.3_	13	P4.3	36
P3.4(A14)	14	P1.6(A14)	15
P3.5	15		
P3.6_	16	P4.2	35
P3.7_	17	P4.1	34
XTAL2	18	Xout	78
XTAL1	19	Xin	77
VSS	20	VSS1	31
P2.0(A8)	21	P1.0(A8)	9
P2.1(A9)	22	P1.1(A9)	10
P2.2(A10)	23	P1.2(A10)	11
P2.3(A11)	24	P1.3(A11)	12
P2.4(A12)	25	P1.4(A12)	13
P2.5(A13)	26	P1.5(A13)	14
P2.6_	27	P4.5	38
P2.7_	28	P4.4	37
	29	P4.6	39
ALE/	30	P4.7	40
/V_{PP}	31	TSTEN	57
P0.7(D7)	32	P6.7(D7)	56
P0.6(D6)	33	P6.6(D6)	55
P0.5(D5)	34	P6.5(D5)	54
P0.4(D4)	35	P6.4(D4)	53
P0.3(D3)	36	P6.3(D3)	52
P0.2(D2)	37	P6.2(D2)	51
P0.1(D2)	38	P6.1(D1)	50
P0.0(D1)	39	P6.0(D0)	49
VCC	40	VDD1	32

Table 8-1 Pin Mapping Table between Intel87C58 and HMS97C8032

Pin Name	Pin Number	Connect to
P1.7	16	Not Connect
P2.0 ~ P2.7	17 ~ 24	VCC
P3.0 ~ P3.5	25 ~ 30	Not Connect
P4.0	33	Not Connect
P5.0 ~ P5.7	41 ~ 48	Not Connect
VSS2	58	GND
VDD2	59	VCC
Avref+	60	VCC
P7.0 ~ P7.7	61 ~ 68	Not Connect
AMIFC	69	Not Connect
FMIFC	70	Not Connect
VSS3	71	GND
VCOH	72	Not Connect
VCOL	73	Not Connect
VDD3	74	VCC
EO	75	Not Connect
XTin	79	Not Connect
XTout	80	Not Connect

Table 8-2 Connection of Other Pins of HMS97C8032 in OTP Mode

9. DEVELOPMENT TOOLS

The HMS97C8032 and HMS91C8032 are supported by a macro assembler, an in-circuit emulator iC1000 HMS9X8032 and OTP programmers. For mode detail, refer to OTP Programming chapter. Macro assembler operates under the MS-Windows 95/98^{TM}.

Please contact sales part of Hynix Semiconductor.

.

Software	- MS- Window base assembler - Linker / Debugger
Hardware (Emulator)	- iSYSTEM. www.isystem.com - iC1000 . POD HMS9XC8032
OTP programmer	- Universal single programmer. - ALL-11 of HI-LO Systems - ADAPTER : OA97C80XX-80QF-1420 - www.hilosystems.com.tw

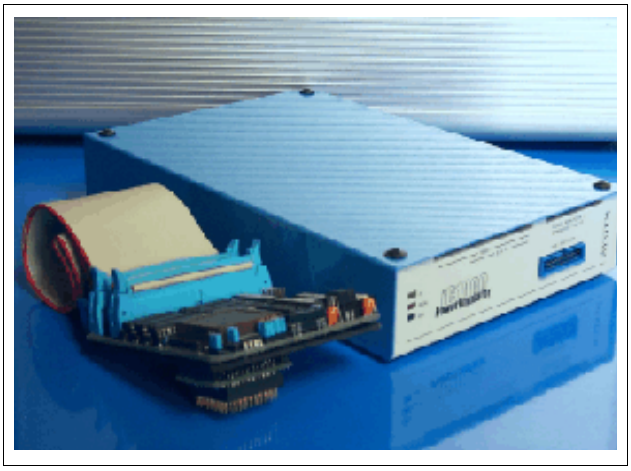
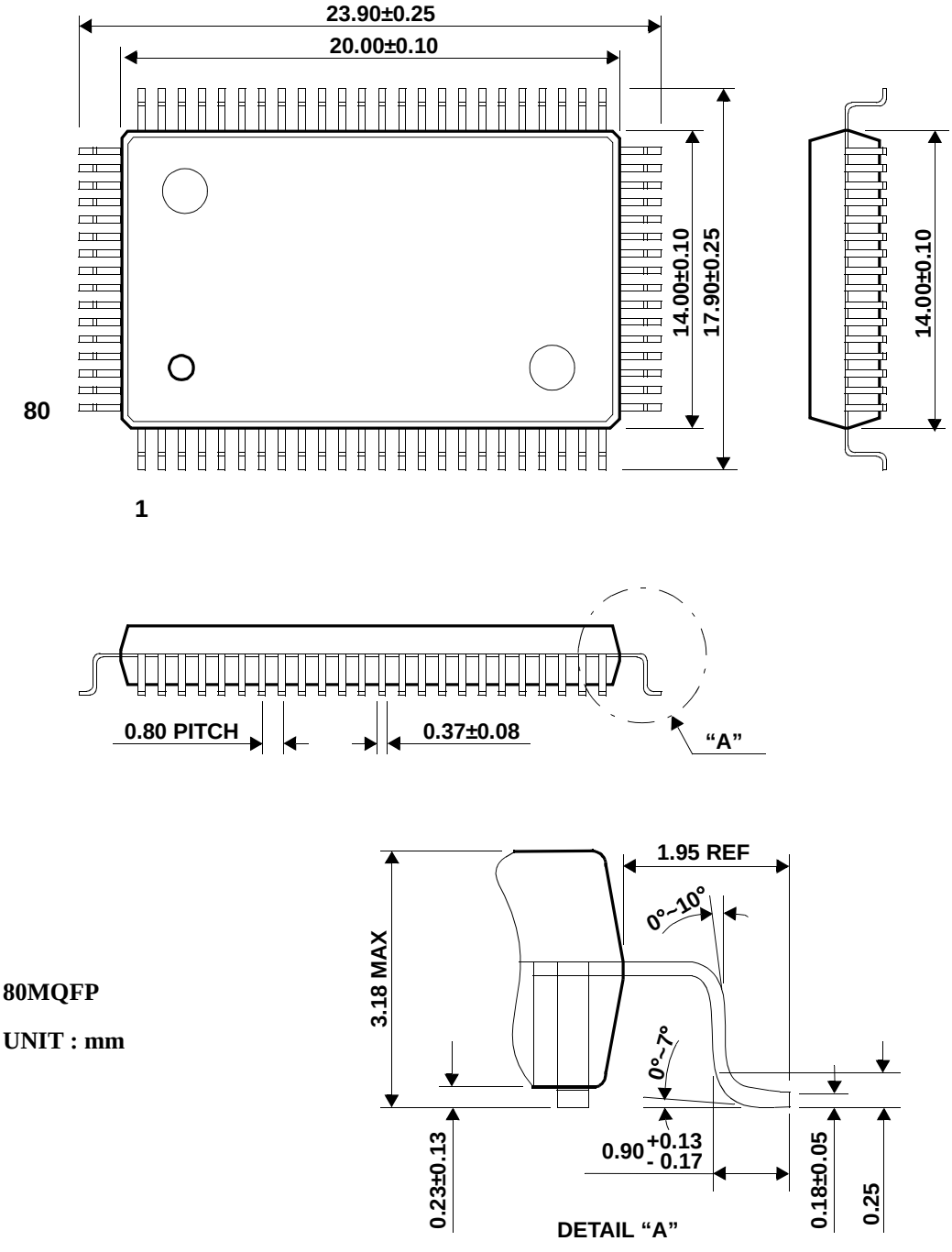


Figure 9-1 iC1000 Emulator with POD HMS9XC8032

Figure 9-2 ALL-11 Programmer with adapter

10. PACKAGE DIMENSION

10.1 HMS97C8032/91C8032 (80 pin package)



80MQFP

UNIT : mm