

ABOV SEMICONDUCTOR
8-BIT SINGLE-CHIP MICROCONTROLLERS

HMS87C1904

HMS87C1902

User's Manual (Ver. 1.1)



REVISION HISTORY

VERSION 1.1 (MAR. 2006) This Book

The company name, MagnaChip Semiconductor Ltd. changed to ABOV Semiconductor Co., Ltd..

Add Pb free package.

VERSION 1.0 (DEC. 2004)

VERSION 0.0 (OCT. 2004)

Preliminary

Version 1.1

Published by

FAE Team

©2006 ABOV Semiconductor Co., Ltd. All right reserved.

Additional information of this manual may be served by ABOV Semiconductor offices in Korea or Distributors and Representatives.

ABOV Semiconductor reserves the right to make changes to any information here in at any time without notice.

The information, diagrams and other data in this manual are correct and reliable; however, ABOV Semiconductor is in no way responsible for any violations of patents or other rights of the third party generated by the use of this manual.

1. OVERVIEW	1
Description	1
Features	1
Development Tools	2
Ordering Information	3
2. BLOCK DIAGRAM	4
3. PIN ASSIGNMENT	5
4. PACKAGE DIAGRAM	6
5. PIN FUNCTION	7
6. PORT STRUCTURES	8
7. ELECTRICAL CHARACTERISTICS	11
Absolute Maximum Ratings	11
Recommended Operating Conditions	11
A/D Converter Characteristics	11
DC Electrical Characteristics	12
AC Characteristics	13
Typical Characteristics	14
8. MEMORY ORGANIZATION	16
Registers	16
Program Memory	18
Data Memory	22
Addressing Mode	25
9. I/O PORTS	29
R0 and R0IO registers	29
10. CLOCK GENERATOR	31
Oscillation Circuit	32
Internal 4MHz Oscillator	33
11. BASIC INTERVAL TIMER	34
12. TIMER / COUNTER	35
8-bit Timer/Counter Mode	36
16-bit Timer/Counter Mode	38
8-bit Compare Output (16-bit)	38
8-bit Capture Mode	38
16-bit Capture Mode	41
PWM Mode	41
13. ANALOG TO DIGITAL CONVERTER	44
14. INTERRUPTS	47
Interrupt Sequence	49
External Interrupt	51
15. WATCHDOG TIMER	52
16. POWER SAVING MODE	53
Minimizing Current Consumption	58
17. RESET	60

18. POWER FAIL DETECTOR 62

19. OTP PROGRAMMING 64

 DEVICE CONFIGURATION AREA 64

20. APPENDIX i

 Instruction Map i

 Alphabetic order table of instruction ii

 Instruction Table by Function vii

HMS87C1904 / HMS87C1902

CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER

1. OVERVIEW

1.1 Description

The HMS87C1904 is an advanced CMOS 8-bit microcontroller with 4K/2K bytes of EPROM. The ABOV HMS87C1904 is a powerful microcontroller which provide a highly flexible and cost effective solution to many small applications. The HMS87C1904 provides the following standard features: 4K/2K bytes of EPROM, 128bytes of RAM, 8-bit timer/counter, 8-bit A/D converter, 10-bit high speed PWM output, Power on reset circuit, on-chip oscillator and clock circuitry. In addition, the HMS87C1904 supports power saving modes to reduce power consumption.

This document is **only explained for the base of HMS87C1904.**

Device name	EPROM	RAM	EXT.INT	I/O	Operating Voltage	Package
HMS87C1904	4K bytes	128bytes	2	5 I/O + 1 Input	2.5 ~ 5.5V	8 PDIP or SOP
HMS87C1902	2K bytes	128bytes	2	5 I/O + 1 Input	2.5 ~ 5.5V	8 PDIP or SOP

1.2 Features

- **4K/2K Bytes On-chip Program Memory**
- **128 Bytes of On-chip Data RAM (Included stack memory)**
- **Instruction Cycle Time:**
 - 250nS at 8MHz
- **Programmable I/O pins (LED direct driving can be source and sink)**
 - Input/Output : 5
 - Input only : 1
- **8-bit A/D Converter**
 - 4 channels
- **One 8-bit Basic Interval Timer**
- **Two 8-bit Timer / Counters**
- **One 10-bit High Speed PWM Outputs**
- **Watchdog timer**
- **Eight Interrupt sources**
 - Keyscan
- **External INT0**
- **A/D Conversion**
- **Timer: 4**
- **PFD**
- **Noise Immunity Circuit**
 - Power Fail Processor (3.2V or 2.4V)
- **Power Down Mode**
 - STOP mode (excluding HF mode)
 - SLEEP mode
- **Operating Frequency**
 - LF : X-tal, Resonator (32KHz ~ 200KHz)
 - ERC : RC Oscillation (200KHz ~ 4MHz)
 - XT : X-tal, Resonator (455KHz ~ 4MHz)
 - HF : X-tal, Resonator (4MHz ~ 12MHz)
 - IRC : Internal RC-oscillation (4MHz)
- **Operating Voltage**
 - 2.5 ~ 5.5V (LF, ERC, XT, IRC)
 - 4.5 ~ 5.5V (HF)
- **Operating Temperature**
 - -40 ~ 85°C

1.3 Development Tools

The HMS87C1904 is supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr™ and OTP programmers.

The marco assembler operates under the MS-Windows 95/98/ME/2000/XP™.

The OTP programmer can be supplied three types of programmer such as single programmer (PGM-plus™), universal stand-alone type single programmer (CHOICE-SIGMA™) and gang type programmer (CHOICE-GANG4™).

In Circuit Emulators	CHOICE-Dr.™
Assembler	ABOV Macro Assembler
OTP Programmer	Single Programmer : PGM-plus™
	Universal Programmer : CHOICE-SIGMA™
	Gang Programmer : CHOICE-GANG4™



Figure 1-2 OTP Single Programmer PGM-plus™



Figure 1-1 In Circuit Emulator CHOICE-Dr.™



Figure 1-3 OTP Gang Programmer CHOICE-GANG4™

1.4 Ordering Information

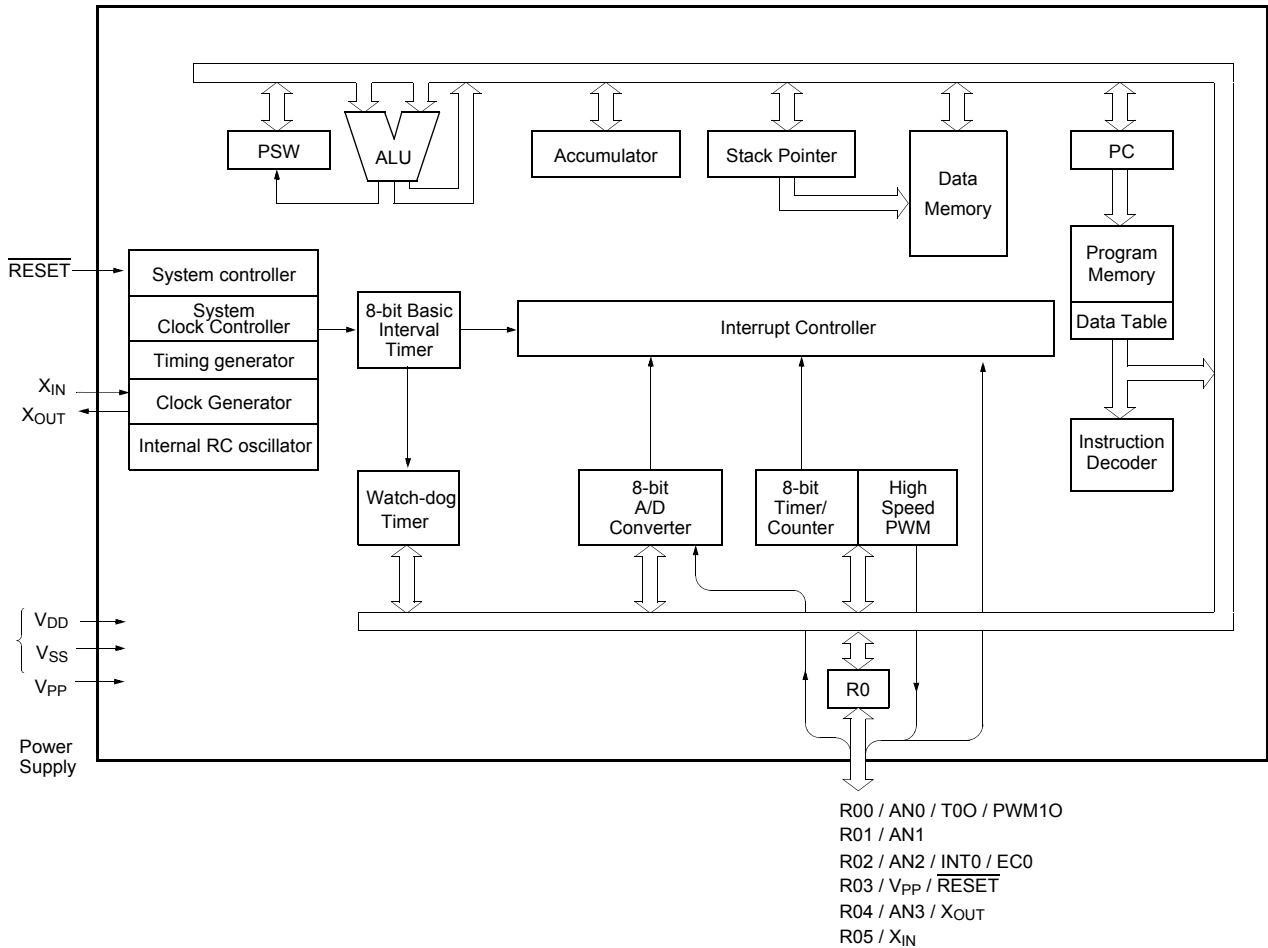
ROM Size	Package Type	Ordering Device Code	Operating Temperature
4K bytes (OTP)	8 PDIP	HMS87C1904	-40 ~ +85°C
	8 SOP	HMS87C1904 D	
2K bytes (OTP)	8 PDIP	HMS87C1902	
	8 SOP	HMS87C1902 D	

Pb free package :

The “P” suffix will be added at the original part number.

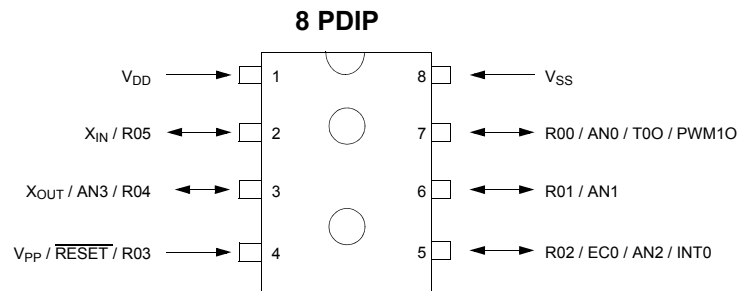
For example; HMS87C1904 (Normal package), HMS87C1904 P (Pb free package)

2. BLOCK DIAGRAM

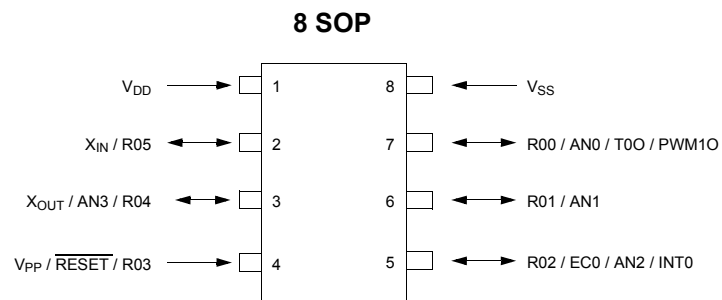


3. PIN ASSIGNMENT

HMS87C1904(2)



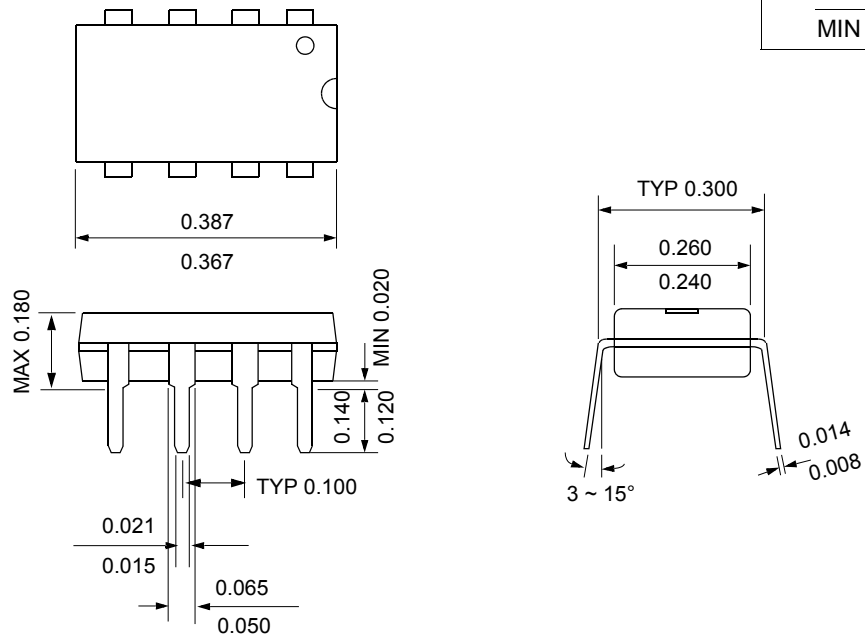
HMS87C1904(2)D



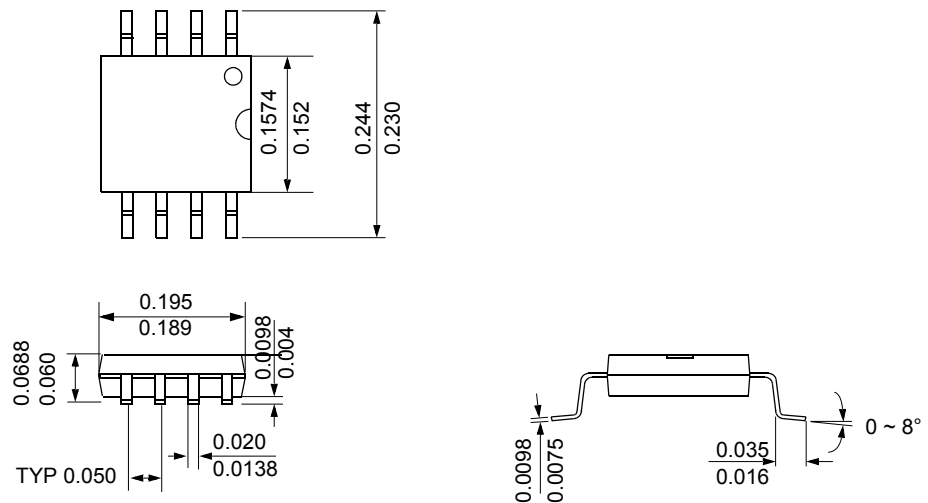
4. PACKAGE DIAGRAM

8 PDIP

unit: inch
 MAX
 MIN



8 SOP



5. PIN FUNCTION

V_{DD}: Supply voltage.

V_{SS}: Circuit ground.

RESET: Reset the MCU.

X_{IN}: Input to the inverting oscillator amplifier and input to the internal main clock operating circuit.

X_{OUT}: Output from the inverting oscillator amplifier.

R00–R05: R0 is a 6-bit, CMOS, bidirectional I/O port except for input only pin R03. R0 pins can be used as outputs or inputs according to “1” or “0” written the their Port Direction Register(R0IO).In addition, R0 serves the functions of the various special features in Table 5-1.

Port pin	Alternate function
R00	AN0 (Analog Input Port 0) T0O (Timer0 Output) PWM1O (PWM Output)
R01	AN1 (Analog Input Port 1)
R02	AN2 (Analog Input Port 2) EC0 (Event Counter Input 0) INT0 (External Interrupt Input Port 0)
R03	RESET (RESET) V _{PP} (EPROM Programming Voltage Input)
R04	AN3 (Analog Input Port34) X _{OUT} (Oscillator Output)
RA5	X _{IN} (Oscillator Input)

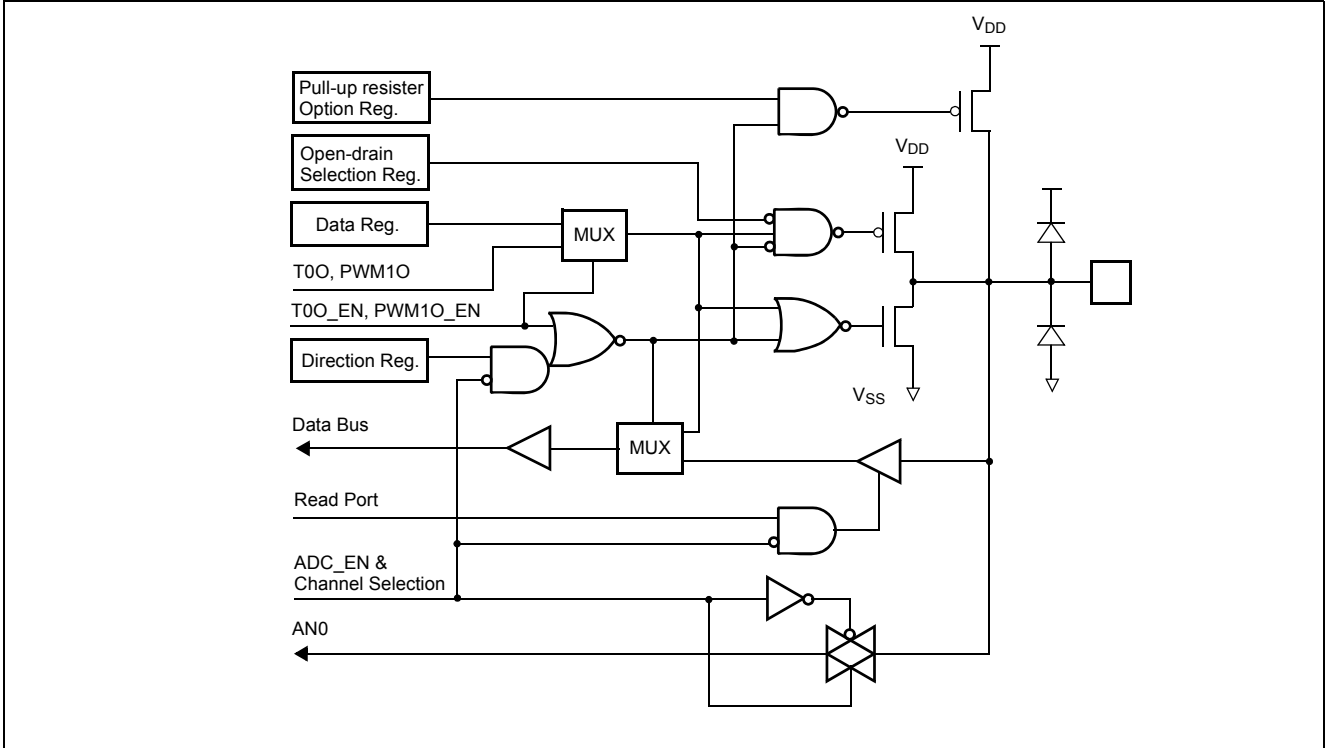
Table 5-1 R0 Port

PIN NAME	PIN No.	In/Out	Function			
			Basic	Alternate		
V _{DD}	1	-	Supply voltage			
V _{SS}	8	-	Circuit ground			
R00 / AN0 / T0O / PWM1O	7	I/O (Input)	General I/O ports	Analog Input Port 0	Timer 0 Output	PWM Output
R01 / AN1	6	I/O (Input)		Analog Input Port 1		
R02 / AN2 / EC0 / INT0	5	I/O (Input)		Analog Input Port 2	External Event Counter Input 0	External Interrupt Input 0
R03 / RESET / V _{PP}	4	I (Input)	Input Port	RESET Input	Programming Voltage Input	
R04 / AN3 / X _{OUT}	3	I/O (Input)	General I/O ports	Analog Input Port 3	Oscillator Output (XT, LP mode)	
R05 / X _{IN}	2	Input		Oscillator Input or External Clock Source Input		

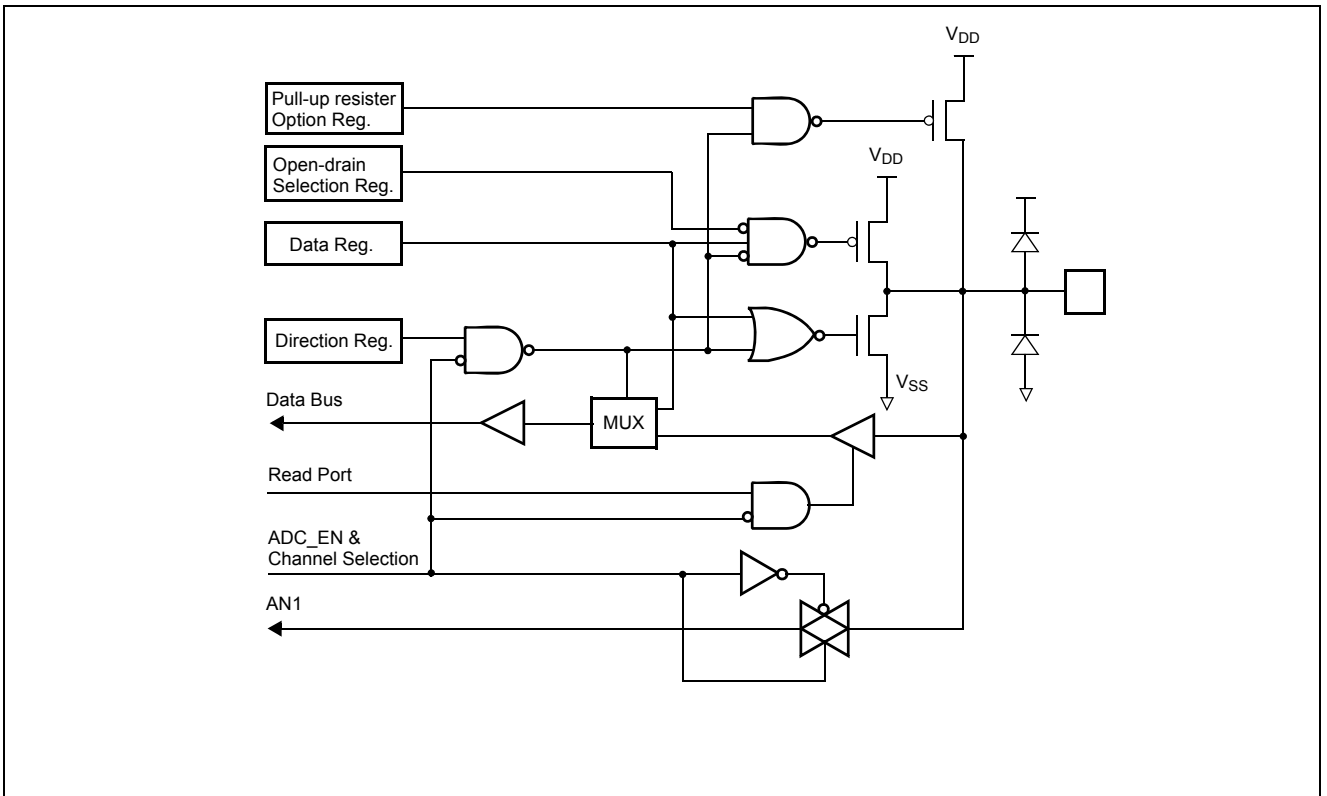
Table 5-2 Pin Description

6. PORT STRUCTURES

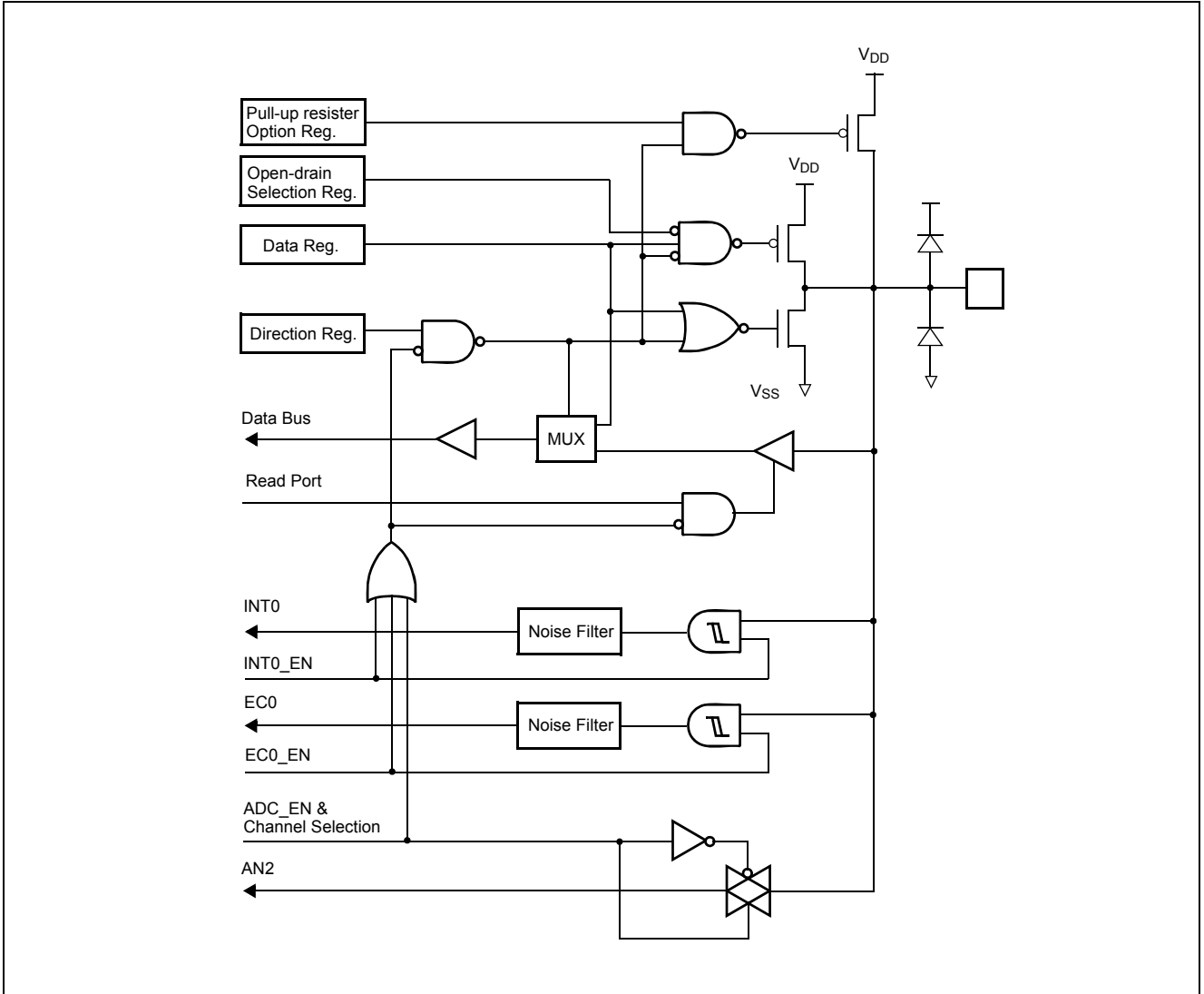
• R00/AN0/T00/PWM10



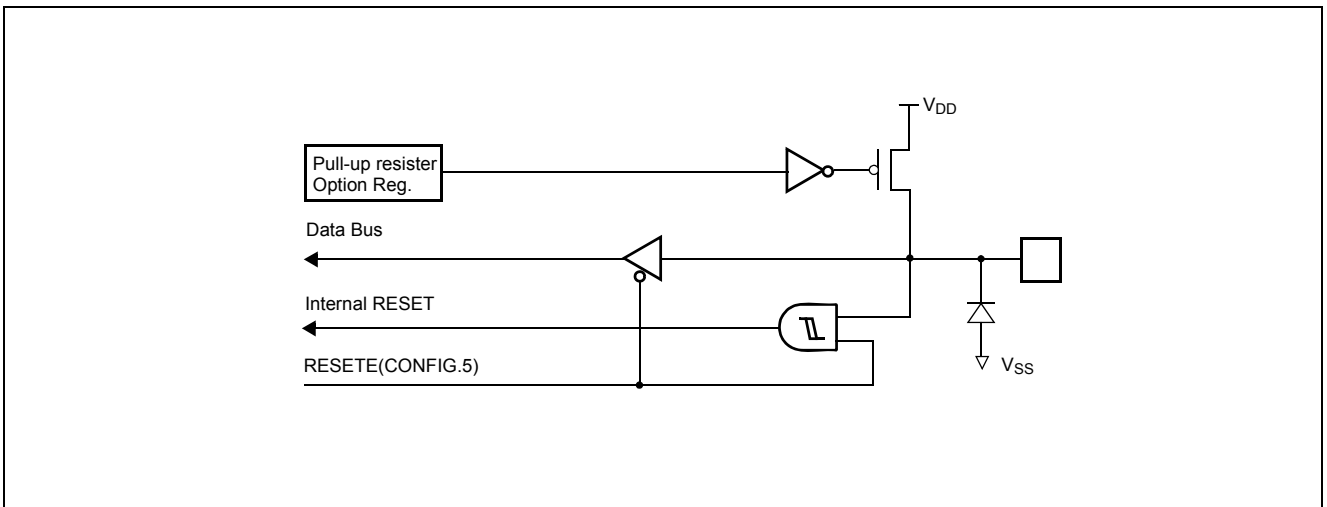
• R01/AN1



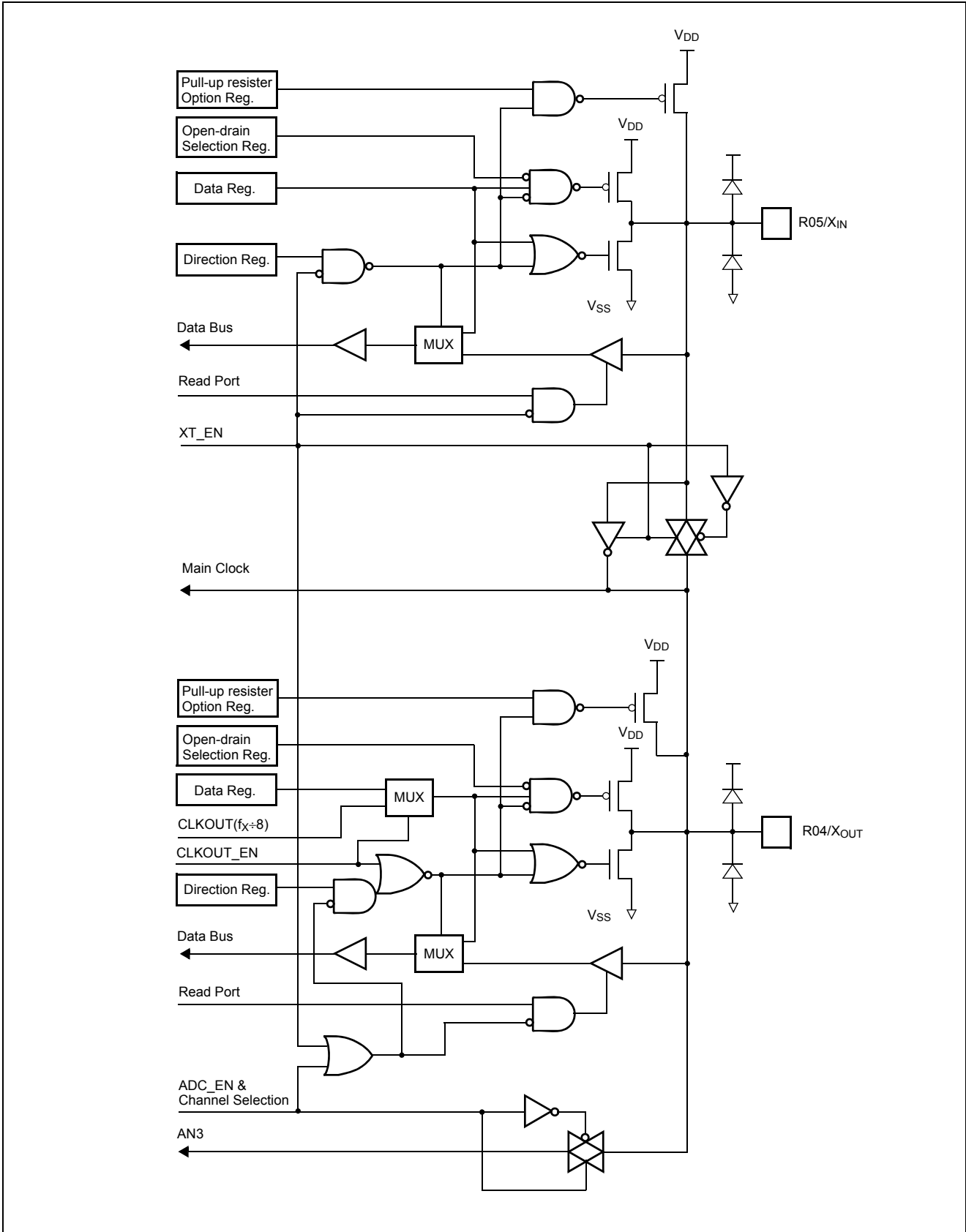
• R02/AN2/EC0/INT0



• R03/V_{PP}/RESET



• R04(AN3/X_{OUT}) R05(X_{IN})



7. ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

Supply voltage	-0.3 to +6.5 V
Storage Temperature	-65 to +125 °C
Voltage on any pin with respect to Ground (V_{SS})	-0.3 to $V_{DD}+0.3$
Maximum current out of V_{SS} pin	150 mA
Maximum current into V_{DD} pin	75 mA
Maximum current sunk by (I_{OL} per I/O Pin)	30 mA
Maximum output current sourced by (I_{OH} per I/O Pin)	15 mA
Maximum current (ΣI_{OL})	150 mA

Maximum current (ΣI_{OH})..... 100 mA

Note: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7.2 Recommended Operating Conditions

Parameter	Symbol	Condition	Specifications		Unit
			Min.	Max.	
Supply Voltage	V_{DD}	$f_{XIN}=32KHz\sim 4MHz$	2.5	5.5	V
		$f_{XIN}=32KHz\sim 12MHz$	4.5	5.5	
Operating Temperature	T_{OPR}	$V_{DD}=2.5\sim 5.5V$	-40	85	°C

7.3 A/D Converter Characteristics

($T_A=25^\circ C$, $V_{SS}=0V$, $V_{DD}=5.12V$ @ $f_{XIN}=4MHz$)

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}		V_{SS}	-	V_{DD}	V
Overall Accuracy	N_{ACC}		-	-	± 2.0	LSB
Non-Linearity Error	N_{NLE}		-	-	± 2.0	LSB
Differential Non-Linearity Error	N_{DNLE}		-	-	± 2.0	LSB
Zero Offset Error	N_{ZOE}		-	-	± 2.0	LSB
Full Scale Error	N_{FSE}		-	-	± 2.0	LSB
Gain Error	N_{NLE}		-	-	± 2.0	LSB
Conversion Time	T_{CONV}	$f_{XIN}=4MHz$	-	-	30	μS
Current Flowing Between AV_{DD} and AV_{SS}	I_{AVDD}		-	1.2	2.8	mA

7.4 DC Electrical Characteristics

($T_A = -40 \sim 85^\circ\text{C}$ at $V_{DD} = 2.5 \sim 5.5\text{V}$, $V_{SS} = 0\text{V}$),

Parameter	Symbol	Pin	Condition	Specifications			Unit	
				Min.	Typ.	Max.		
Input High Voltage	V_{IH1}	$\overline{\text{RESET}}$		$0.8 V_{DD}$	-	V_{DD}	V	
	V_{IH2}	Hysteresis Input ¹		$0.8 V_{DD}$				
	V_{IH3}	Normal Input(TTL)		2				
	V_{IH4}	X_{IN}	RC OSC	$0.8 V_{DD}$				
	V_{IH5}		HF, XT, LF OSC	$0.7 V_{DD}$				
Input Low Voltage	V_{IL1}	$\overline{\text{RESET}}$		0	-	0.6	V	
	V_{IL2}	Hysteresis Input ¹						
	V_{IL3}	Normal Input						
	V_{IL4}	X_{IN}	RC OSC					0.2 V_{DD}
	V_{IL5}		HF, XT, LF OSC					0.3 V_{DD}
Output High Voltage	V_{OH}	All Output Port	$V_{DD} = 4.5\text{V}$, $I_{OH} = -5\text{mA}$	$V_{DD} - 1$	-	-	V	
Output Low Voltage	V_{OL}	All Output Port	$V_{DD} = 4.5\text{V}$, $I_{OL} = 10\text{mA}$	-	-	1	V	
Pull-up Resistor ²	I_P	R00, R01, R02, R04, R05	$V_{DD} = 4.5\text{V}$	50	100	150	K Ω	
Input High Leakage Current	I_{IH}	All Pins	$V_{DD} = 5\text{V}$	-	-	5	μA	
Input Low Leakage Current	I_{IL}	All Pins	$V_{DD} = 5\text{V}$	-5	-	-	μA	
Hysteresis	$ V_T $	Hysteresis Input ¹	$V_{DD} = 5\text{V}$	0.5	-	-	V	
V_{DD} rise rate to ensure Power-on Reset	S_{VDD}			0.05			V/mS	
PFD Voltage	V_{PFD1}	V_{DD}	PFD Level = 0, Low	1.9	2.4	2.9	V	
	V_{PFD2}		PFD Level = 1, High	2.8	3.2	3.7		
Internal RC WDT Period	T_{RCWDT}		$V_{DD} = 5.0\text{V}$	60		200	μS	
Operating Current	I_{DD}	V_{DD}	HF	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 8\text{MHz}$		5	10	mA
			ERC	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 8\text{MHz}$		5	9	
			IRC	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 4\text{MHz}$		2.5	3.5	
			XT	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 4\text{MHz}$		3	5	
			LF	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 32\text{KHz}$		0.4	0.8	
Power Down Current (ERC, IRC, XT mode)	I_{STOP}^3	V_{DD}	$V_{DD} = 5.5\text{V}$, $f_{XIN} = 4\text{MHz}$	-	2	4	μA	
	I_{SLEEP}			-	2	3	mA	
Internal Calibrated RC Frequency	F_{IRC}	V_{DD}	$V_{DD} = 5\text{V}$	3.6	4	4.4	MHz	

Parameter	Symbol	Pin	Condition	Specifications			Unit
				Min.	Typ.	Max.	
RAM Data Retention Voltage	V _{DR}	V _{DD}		-	1.5	-	V

1. Hysteresis Input: EC0, INT0
2. This parameter is valid when the bit of R0PU is selected and set the input mode or interrupt input function.
3. In the HF mode, the STOP mode can not be used.

7.5 AC Characteristics

(T_A=-40~+85°C, V_{DD}=5V±10%, V_{SS}=0V)

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Operating Frequency	f _{CP}	X _{IN}	0.032	-	12	MHz
System Clock Cycle Time	t _{sys}	X _{IN}	0.166	-	62.5	μS
External Clock Pulse Width	t _{CPW}	X _{IN}	-	t _{sys} /2	-	nS
External Clock Transition Time	t _{RCP} , t _{FCP}	X _{IN}	-	-	5	nS
Oscillation Stabilizing Time	t _{ST}	X _{IN} , X _{OUT}	-	-	20	mS
External Input Pulse Width	t _{EPW}	INT0, EC0	2	-	-	t _{sys}
RESET Input Width	t _{RST}	RESET	1	4	30	μS

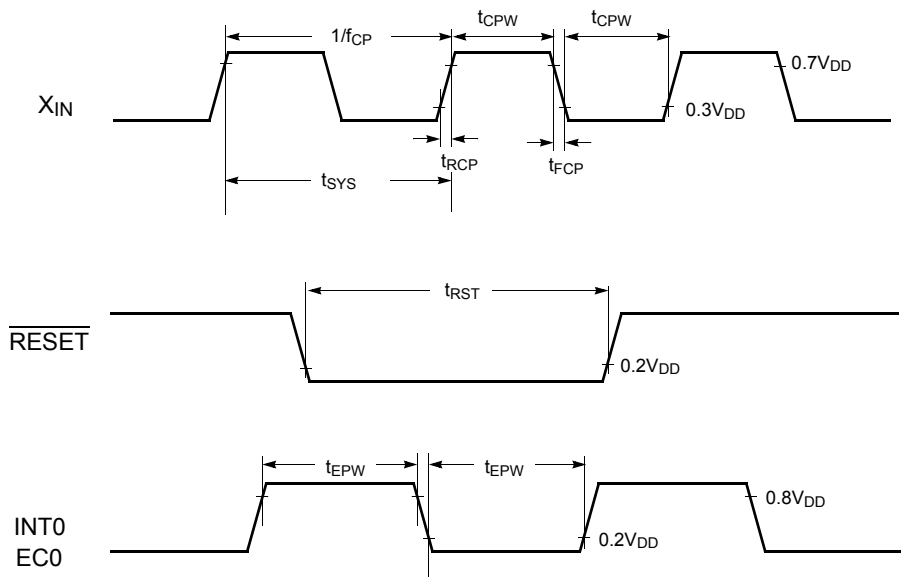


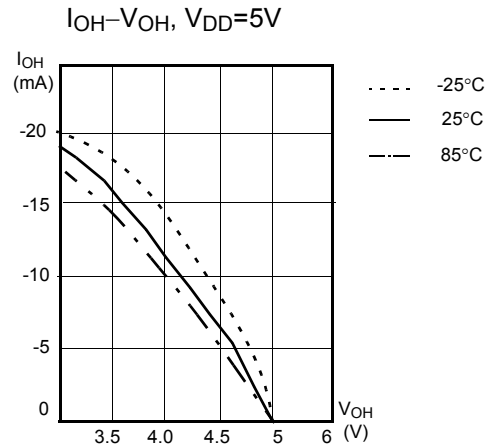
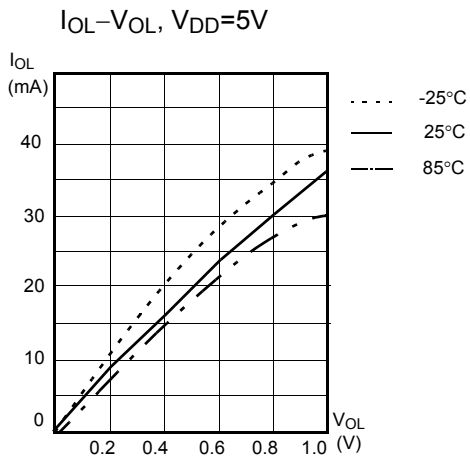
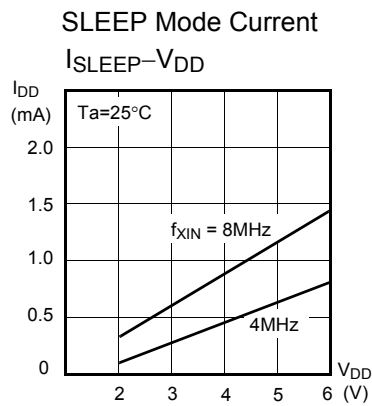
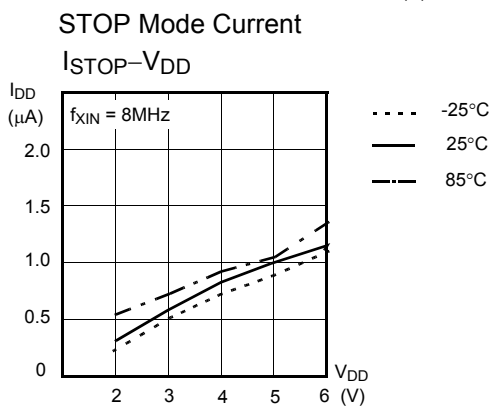
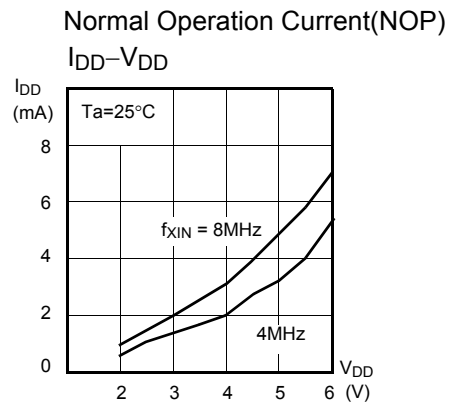
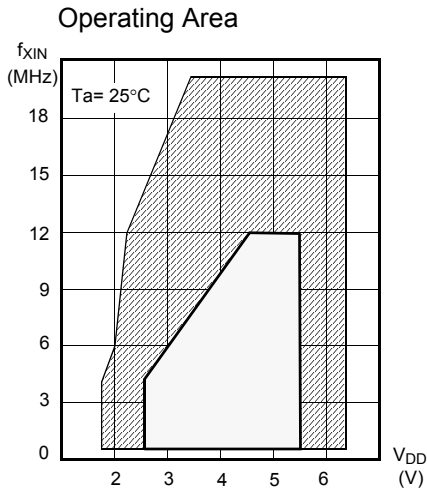
Figure 7-1 Timing Chart

7.6 Typical Characteristics

These graphs and tables provided in this section are for design guidance only and are not tested or guaranteed.

In some graphs or tables the data presented are outside specified operating range (e.g. outside specified V_{DD} range). This is for information only and devices are guaranteed to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. “Typical” represents the mean of the distribution while “max” or “min” represents (mean + 3σ) and (mean - 3σ) respectively where σ is standard deviation



R _{EXT}	C _{EXT}	With External Capacitor		Without External Capacitor	
		f _{osc} @ 5V, 25°C	f _{osc} @ 3V, 25°C	f _{osc} @ 5V, 25°C	f _{osc} @ 3V, 25°C
4.7KΩ	10pF	3.90MHz	2.70MHz	5.43MHz	3.66MHz
	20pF	3.19MHz	2.22MHz		
	30pF	2.74MHz	1.72MHz		
10KΩ	10pF	2.13MHz	1.69MHz	3.20MHz	2.48MHz
	20pF	1.79MHz	1.36MHz		
	30pF	1.58MHz	1.18MHz		
20KΩ	10pF	1.23MHz	0.97MHz	1.88MHz	1.52MHz
	20pF	0.96MHz	0.78MHz		
	30pF	0.84MHz	0.63MHz		

Table 7-1 RC Oscillation Frequencies (with C_{EXT} and without C_{EXT})

8. MEMORY ORGANIZATION

The HMS87C1904 has separate address spaces for Program memory and Data Memory. Program memory can only be read, not written to. It can be up to 2K/4K bytes of

Program memory. Data memory can be read and written to up to 128 bytes including the stack area.

8.1 Registers

This device has six registers that are the Program Counter (PC), a Accumulator (A), two index registers (X, Y), the Stack Pointer (SP), and the Program Status Word (PSW). The Program Counter consists of 16-bit register.

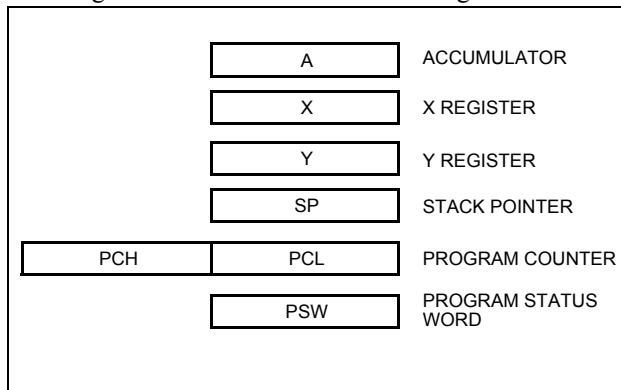


Figure 8-1 Configuration of Registers

Accumulator: The Accumulator is an 8-bit general purpose register, used for data operation such as transfer, temporary saving, and conditional judgement, etc.

The Accumulator can be used as a 16-bit register with Y Register as shown below.

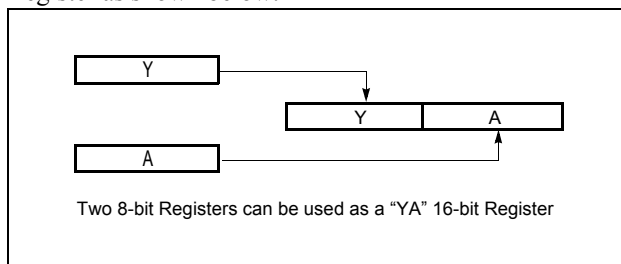


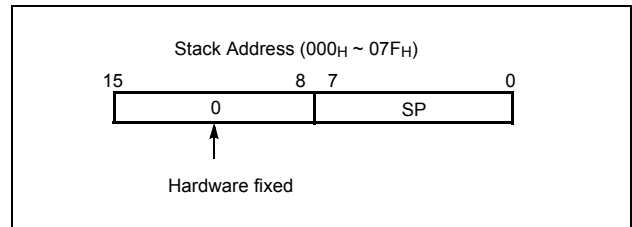
Figure 8-2 Configuration of YA 16-bit Register

X, Y Registers: In the addressing mode which uses these index registers, the register contents are added to the specified address, which becomes the actual address. These modes are extremely effective for referencing subroutine tables and memory tables. The index registers also have increment, decrement, comparison and data transfer functions, and they can be used as simple accumulators.

Stack Pointer: The Stack Pointer is an 8-bit register used for occurrence interrupts and calling out subroutines. Stack Pointer identifies the location in the stack to be accessed (save or restore).

Generally, SP is automatically updated when a subroutine call is executed or an interrupt is accepted. However, if it is used in excess of the stack area permitted by the data memory allocating configuration, the user-processed data may be lost.

The stack can be located at any position within 00H to 7FH of the internal data memory. The SP is not initialized by hardware, requiring to write the initial value (the location with which the use of the stack starts) by using the initialization routine. Normally, the initial value of "7FH" is used.



Note: The Stack Pointer must be initialized by software because its value is undefined after RESET.

```
Example: To initialize the SP
LDX    #07FH
TXSP                      ; SP ← 7FH
```

Program Counter: The Program Counter is a 16-bit wide which consists of two 8-bit registers, PCH and PCL. This counter indicates the address of the next instruction to be executed. In reset state, the program counter has reset routine address (PCH:0FFH, PCL:0FEH).

Program Status Word: The Program Status Word (PSW) contains several bits that reflect the current state of the CPU. The PSW is described in Figure 8-3. It contains the Negative flag, the Overflow flag, the Break flag the Half Carry (for BCD operation), the Interrupt enable flag, the Zero flag, and the Carry flag.

[Carry flag C]

This flag stores any carry or borrow from the ALU of CPU after an arithmetic operation and is also changed by the Shift Instruction or Rotate Instruction.

[Zero flag Z]

This flag is set when the result of an arithmetic operation or data transfer is "0" and is cleared by any other result.

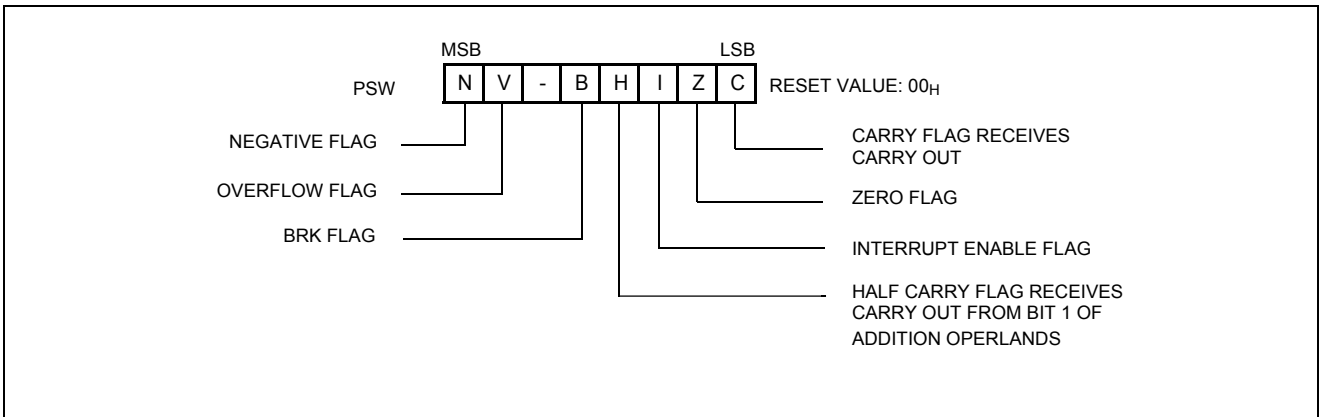


Figure 8-3 PSW (Program Status Word) Register

[Interrupt disable flag I]

This flag enables/disables all interrupts except interrupt caused by Reset or software BRK instruction. All interrupts are disabled when cleared to “0”. This flag immediately becomes “0” when an interrupt is served. It is set by the EI instruction and cleared by the DI instruction.

[Half carry flag H]

After operation, this is set when there is a carry from bit 3 of ALU or there is no borrow from bit 4 of ALU. This bit can not be set or cleared except CLRV instruction with Overflow flag (V).

[Break flag B]

This flag is set by software BRK instruction to distinguish BRK from TCALL instruction with the same vector address.

dress.

[Overflow flag V]

This flag is set to “1” when an overflow occurs as the result of an arithmetic operation involving signs. An overflow occurs when the result of an addition or subtraction exceeds +127(7FH) or -128(80H). The CLRV instruction clears the overflow flag. There is no set instruction. When the BIT instruction is executed, bit 6 of memory is copied to this flag.

[Negative flag N]

This flag is set to match the sign bit (bit 7) status of the result of a data or arithmetic operation. When the BIT instruction is executed, bit 7 of memory is copied to this flag.

8.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but these devices have 4K/2K bytes program memory space only physically implemented. Accessing a location above FFFF_H will cause a wrap-around to 0000_H.

Figure 8-4, shows a map of Program Memory. After reset, the CPU begins execution from reset vector which is stored in address FFFE_H and FFFF_H as shown in Figure 8-5.

As shown in Figure 8-4, each area is assigned a fixed location in Program Memory. Program Memory area contains the user program.

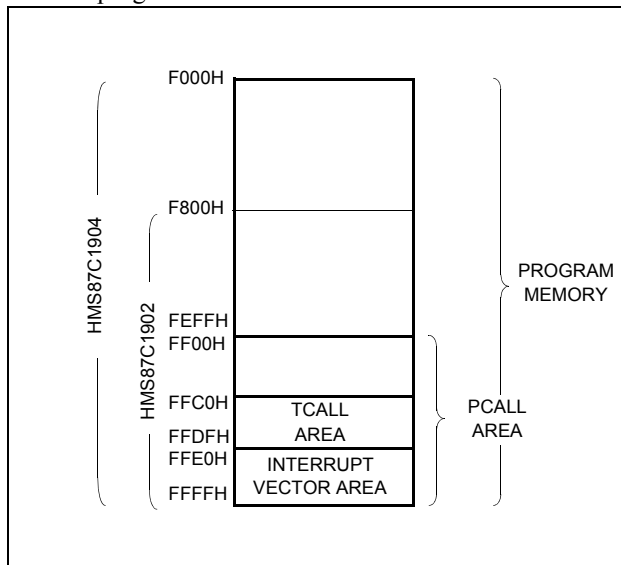


Figure 8-4 Program Memory Map

Page Call (PCALL) area contains subroutine program to reduce program byte length by using 2 bytes PCALL instead of 3 bytes CALL instruction. If it is frequently called, it is more useful to save program byte length.

Example: Usage of TCALL

```

LDA    #5
TCALL  0FH      ; 1BYTE INSTRUCTION
          :      ; INSTEAD OF 3 BYTES
          :      ; NORMAL CALL

;
; TABLE CALL ROUTINE
;
FUNC_A: LDA    LRG0
        RET

;
FUNC_B: LDA    LRG1
        RET

;
; TABLE CALL ADD. AREA
;
        ORG    0FFC0H
        DW    FUNC_A
        DW    FUNC_B
    
```

Table Call (TCALL) causes the CPU to jump to each TCALL address, where it commences the execution of the service routine. The Table Call service area spaces 2-byte for every TCALL: 0FFC0_H for TCALL15, 0FFC2_H for TCALL14, etc., as shown in Figure 8-6.

The interrupt causes the CPU to jump to specific location, where it commences the execution of the service routine. The External interrupt 0, for example, is assigned to location 0FFFA_H. The interrupt service locations spaces 2-byte interval: 0FFF8_H and 0FFF9_H for Timer/Counter 0 Interrupt, 0FFFA_H and 0FFFB_H for External Interrupt 0, etc.

As for the area from 0FF00_H to 0FFFF_H, if any area of them is not going to be used, its service location is available as general purpose Program Memory.

In the case of using C-compiler, it must be define the code macro as example of next page.

Address	Vector Area Memory
0FFE0H	-
E2	-
E4	-
E6	-
E8	-
EA	-
EC	-
EE	Power Fail Detect Interrupt Vector Area
F0	Basic Interval Interrupt Vector Area
F2	Watchdog Timer Interrupt Vector Area
F4	A/D Converter Interrupt Vector Area
F6	Timer/Counter 1 Interrupt Vector Area
F8	Timer/Counter 0 Interrupt Vector Area
FA	External Interrupt 0 Vector Area
FC	Key Scan Interrupt Vector Area
FE	RESET Vector Area

NOTE:
“-” means reserved area.

Figure 8-5 Interrupt Vector Area

Example: The usage software example of Vector address and the initialize part using C-compiler .

```
// The Vector Address must be define as below bold characters. //

#define INT0    __attribute__ ((interrupt ("0")))
#define INT1    __attribute__ ((interrupt ("1")))
#define INT2    __attribute__ ((interrupt ("2")))
#define INT3    __attribute__ ((interrupt ("3")))
#define INT4    __attribute__ ((interrupt ("4")))
#define INT5    __attribute__ ((interrupt ("5")))
#define INT6    __attribute__ ((interrupt ("6")))
#define INT7    __attribute__ ((interrupt ("7"))) /* FFEE-FFEF, (PFD) */
#define INT8    __attribute__ ((interrupt ("8"))) /* FFF0-FFF1, (BIT) */
#define INT9    __attribute__ ((interrupt ("9"))) /* FFF2-FFF3, (WDT) */
#define INT10   __attribute__ ((interrupt ("10"))) /* FFF4-FFF5, (ADC) */
#define INT11   __attribute__ ((interrupt ("11"))) /* FFF6-FFF7, (T1) */
#define INT12   __attribute__ ((interrupt ("12"))) /* FFF8-FFF9, (T0) */
#define INT13   __attribute__ ((interrupt ("13"))) /* FFFA-FFFB, (INT0) */
#define INT14   __attribute__ ((interrupt ("14"))) /* FFFC-FFFD, (KSCAN) */

void    INT7    PFD(void);
void    INT8    BIT(void);
void    INT9    WDT(void);
void    INT10   ADC(void);
void    INT11   BaseTim(void);
void    INT12   T0(void);
void    INT13   INT0(void);
void    INT14   KSCAN(void);

        :
        :

void PFD(void)
{
//      :
//      user program written here
//      :
}

void KSCAN(void)
{
//      :
//      user program written here
//      :
}
```

Example: The usage software example of Vector address and the initialize part using Assembler .

```

ORG    0FFE0H

DW    NOT_USED      ; (0FFE0)
DW    NOT_USED      ; (0FFE2)
DW    NOT_USED      ; (0FFE4)
DW    NOT_USED      ; (0FFE6)
DW    NOT_USED      ; (0FFE8)
DW    NOT_USED      ; (0FFEA)
DW    NOT_USED      ; (0FFEC)
DW    PFD_INT       ; (0FFEE) Power Fail Detect
DW    BIT_INT       ; (0FFF0) Basic Interval Timer
DW    WDT_INT       ; (0FFF2) Watchdog Timer
DW    AD_INT        ; (0FFF4) A/D
DW    TMR1_INT      ; (0FFF6) Timer-1
DW    TMR0_INT      ; (0FFF8) Timer-0
DW    INT0          ; (0FFFA) Int.0
DW    KEY_SCAN      ; (0FFFC) Key Scan
DW    RESET         ; (0FFFE) Reset

ORG    0F000H

;*****
;          MAIN      PROGRAM      *
;*****
;
RESET:  DI          ;Disable All Interrupts
        LDX        #0
RAM_CLR: LDA        #0          ;RAM Clear(!0000H->!007FH)
        STA        {X}+
        CMPX       #080H
        BNE        RAM_CLR
;
        LDX        #07FH          ;Stack Pointer Initialize
        TXSP
;
        CALL       INITIAL      ;
;
        LDM        R0, #0        ;Normal Port 0
        LDM        R0IO, #0000_0010B ;R01 Output, Others Input
        :
        :
        LDM        PFDR, #0      ;Enable Power Fail Detector
        :
        :

```

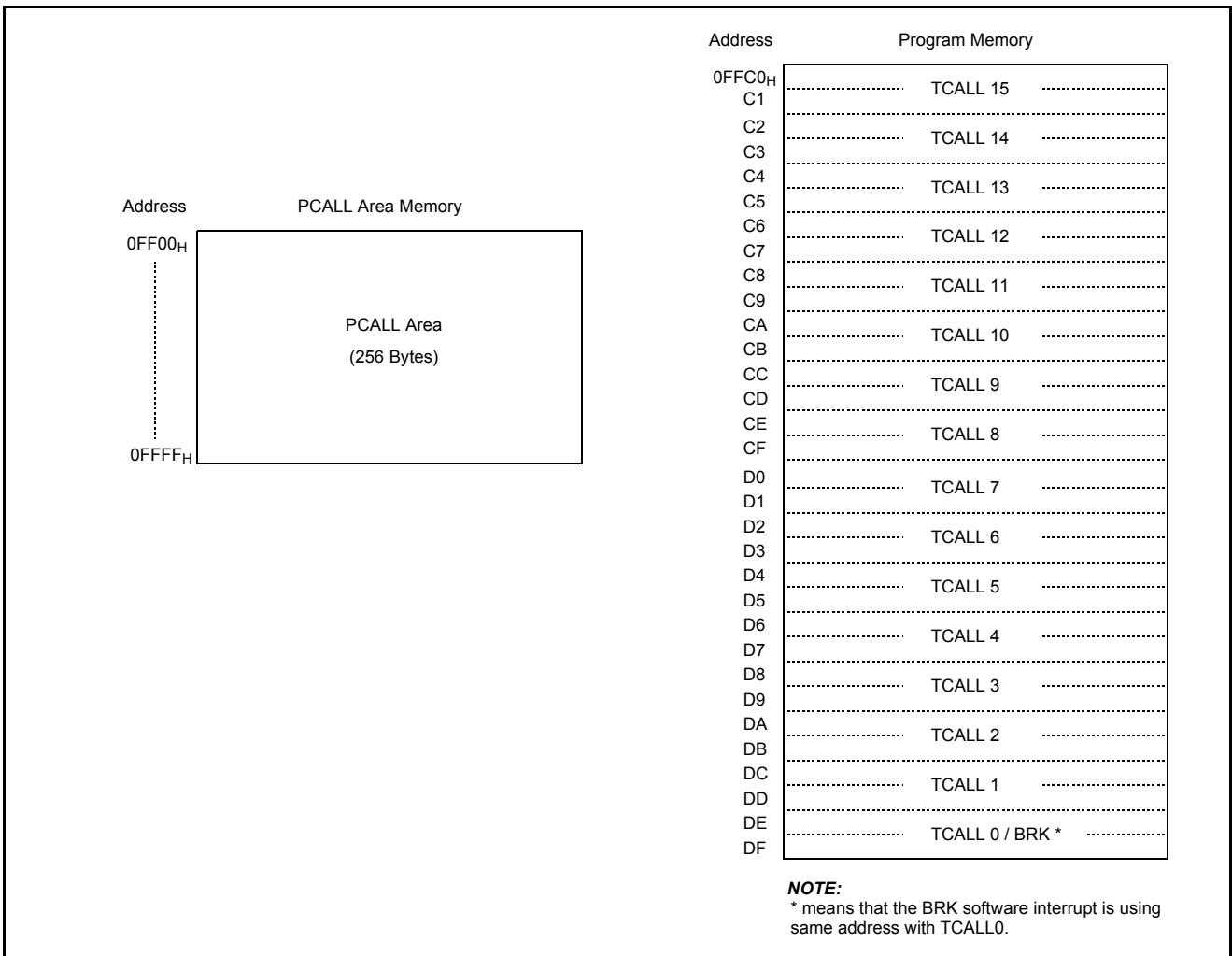
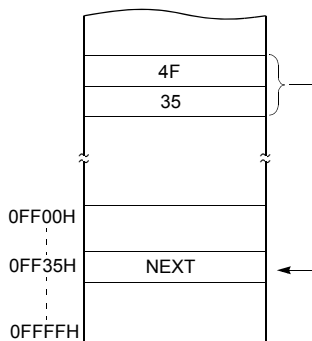



Figure 8-6 PCALL and TCALL Memory Area

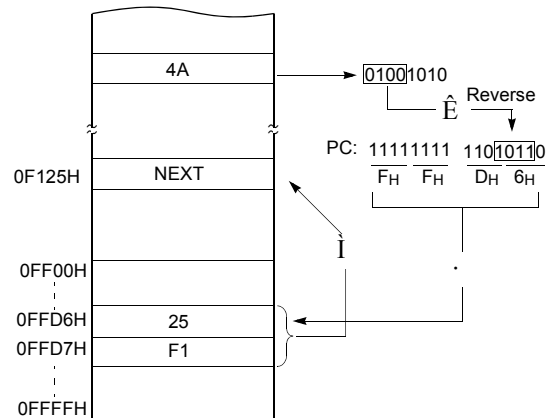
PCALL → rel

4F35 PCALL 35H



TCALL → n

4A TCALL 4



8.3 Data Memory

Figure 8-7 shows the internal Data Memory space available. Data Memory is divided into two groups, a user RAM (including Stack) and control registers.

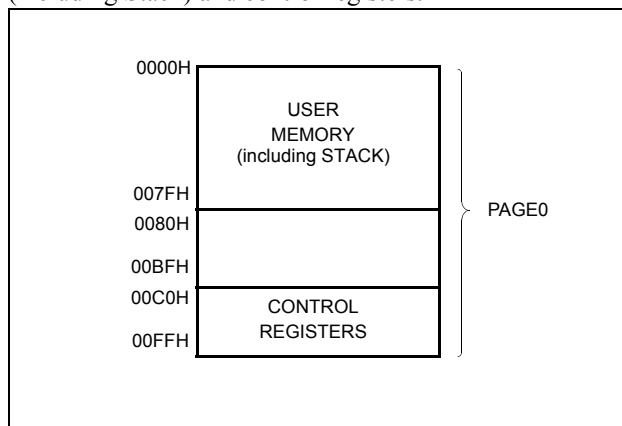


Figure 8-7 Data Memory Map

User Memory

The HMS87C1904 has 128×8 bits for the user memory (RAM).

Control Registers

The control registers are used by the CPU and Peripheral function blocks for controlling the desired operation of the device. Therefore these registers contain control and status bits for the interrupt system, the timer/ counters, analog to digital converters and I/O ports. The control registers are in address range of 0C0H to 0FFH.

Note that unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

More detailed informations of each register are explained in each peripheral section.

Note: Write only registers can not be accessed by bit manipulation instruction. Do not use read-modify-write instruction. Use byte manipulation instruction.

Example; To write at CKCTRL

```
LDM CKCTRL,#09H ;Divide ratio +16
```

Address	Symbol	R/W	RESET Value	Addressing mode
0C0H	R0	R/W	--00_000	byte, bit ¹
0C1H	R0IO	W	--00_000	byte ²
0C2H	R0OD	W	--00_000	byte
0C3H	R0PU	W	--00_0000	byte
0C4H	R0FUNC	W	----_0000	byte
0C5H	R0KS	W	0000_0000	byte
0D0H	TM0	R/W	--00_0000	byte, bit
0D1H	T0	R	0000_0000	byte
0D1H	TDR0	W	1111_1111	byte
0D1H	CDR0	R	0000_0000	byte
0D2H	TM1	R/W	0000_0000	byte, bit
0D3H	TDR1	W	1111_1111	byte
0D3H	T1PPR	W	1111_1111	byte
0D4H	T1	R	0000_0000	byte
0D4H	CDR1	R	0000_0000	byte
0D4H	T1PDR	R/W	0000_0000	byte, bit
0D5H	PWM1HR	W	----_0000	byte
0E0H	BITR	R	undefined	byte
0E0H	CKCTRL	W	0-01_0111	byte
0E2H	WDTR	R	undefined	byte
0E2H	WDTR	W	0111_1111	byte
0E3H	IENH	R/W	0000_0000	byte, bit
0E5H	IRQH	R/W	0000_0000	byte, bit
0E7H	IEDS	R/W	0000_0000	byte, bit
0E8H	ADCM	R/W	--0-_0001	byte, bit
0E9H	ADCR	R	Undefined	byte
0ECH	SR	R/W	0000_0000	byte
0EDH	ICR	W	0000_0000	byte
0EEH	PFDR	R/W	----_000	byte, bit
0EFH	SSCR	W	0000_0000	byte

Table 8-1 Control Registers

1. "byte, bit" means that register can be addressed by not only bit but byte manipulation instruction.
2. "byte" means that register can be addressed by only byte manipulation instruction. On the other hand, do not use any read-modify-write instruction such as bit manipulation for clearing bit.

Note: Several names are given at same address. Refer to below table.

Addr.	When read			When write	
	Timer Mode	Capture Mode	PWM Mode	Timer Mode	PWM Mode
D1H	T0	CDR0	-	TDR0	-
D3H	-			TDR1	T1PPR
D4H	T1	CDR1	T1PDR	-	T1PDR
EC0	BITR			CKCTLR	

Table 8-2 Various Register Name in Same Address

Stack Area

The stack provides the area where the return address is saved before a jump is performed during the processing routine at the execution of a subroutine call instruction or the acceptance of an interrupt.

When returning from the processing routine, executing the subroutine return instruction [RET] restores the contents of the program counter from the stack; executing the interrupt return instruction [RETI] restores the contents of the program counter and flags.

The save/restore locations in the stack are determined by the stack pointed (SP). The SP is automatically decreased after the saving, and increased before the restoring. This means the value of the SP indicates the stack location number for the next save.



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C0H	R0	R0 Port Data Register							
C1H	R0IO	R0 Port Direction Register							
C2H	R0OD	R0 Port Open-drain Selection Register							
C3H	R0PU	R0 Port Pull-up Selection Register							
C4H	R0FUNC	R0 Port Function Selection Register							
C5H	R0KS	R0 Port Key Scan Selection Register							
D0H	TM0	-	-	CAP0	T0CK2	T0CK1	T0CK0	T0CN	T0ST
D1H	T0/TDR0/ CDR0	Timer0 Register / Timer0 Data Register / Capture0 Data Register							
D2H	TM1	POL	T16BIT	PWM1E	-	T1CK1	T1CK0	T1CN	T1ST
D3H	TDR1/ T1PPR	Timer1 Data Register / PWM1 Period Register							
D4H	T1/CDR1/ T1PDR	Timer1 Register / Capture1 Data Register / PWM1 Duty Register							
D5H	PWM1HR	PWM1 High Register							
E0H	BITR ¹	Basic Interval Timer Data Register							
E0H	CKCTLR ¹	RCCLK	-	RCWDT	WDTON	BTCL	BTS2	BTS1	BTS0
E2H	WDTR	WDTCL	7-bit Watchdog Counter Register						
E3H	IENH	KSCANE	INT0E	T0E	T1E	ADE	WDTE	BITE	PFDE
E5H	IRQH	KSCANIF	INT0IF	T0IF	T1IF	ADIF	WDTIF	BITIF	PFDF
E7H	IEDS	-	-	-	-	IED1H	IED1L	IED0H	IED0L
E8H	ADCM	-	-	ADEN	-	ADS1	ADS0	ADST	ADSF
E9H	ADCR	ADC Result Data Register							
ECH	SR	Status Register							
EDH	ICR	Internal Oscillator Calibration Register							
EEH	PFDR ²								
EFH	SSCR	Stop & Sleep Mode Control Register							

Table 8-3 Control Registers of HMS87C1904

These registers of shaded area can not be accessed by bit manipulation instruction as "SET1, CLR1", but should be accessed by register operation instruction as "LDM dp,#imm".

1. The register BITR and CKCTLR are located at same address. Address E0H is read as BITR, written to CKCTLR.
2. The register PFDR only be implemented on devices, not on In-circuit Emulator.

8.4 Addressing Mode

The HMS87C1904 uses six addressing modes;

- Register addressing
- Immediate addressing
- Direct page addressing
- Absolute addressing
- Indexed addressing
- Register-indirect addressing

(1) Register Addressing

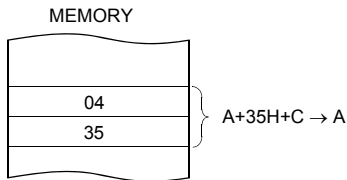
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

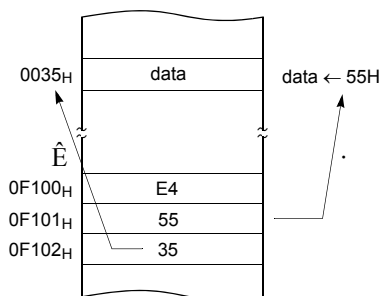
In this mode, second byte (operand) is accessed as a data immediately.

Example:

```
0435   ADC   #35H
```



```
E45535  LDM  35H, #55H
```

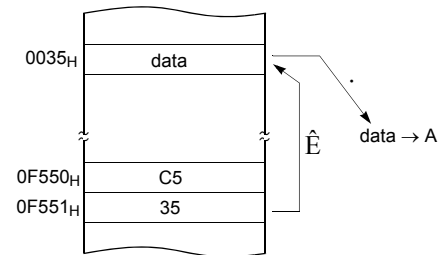


(3) Direct Page Addressing → dp

In this mode, a address is specified within direct page.

Example;

```
C535   LDA   35H           ;A ←RAM[35H]
```



(4) Absolute Addressing → !abs

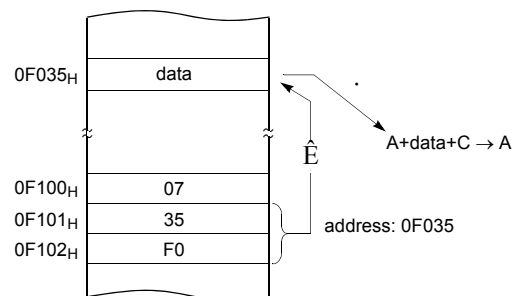
Absolute addressing sets corresponding memory data to Data, i.e. second byte(Operand I) of command becomes lower level address and third byte (Operand II) becomes upper level address.

With 3 bytes command, it is possible to access to whole memory area.

ADC, AND, CMP, CMPX, CMPY, EOR, LDA, LDX, LDY, OR, SBC, STA, STX, STY

Example;

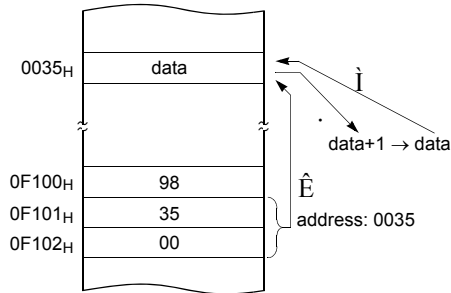
```
0735F0  ADC   !0F035H     ;A ←ROM[0F035H]
```



The operation within data memory (RAM)
ASL, BIT, DEC, INC, LSR, ROL, ROR

Example; Addressing accesses the address 0135H.

```
983500 INC !0035H ; A ←RAM[035H]
```



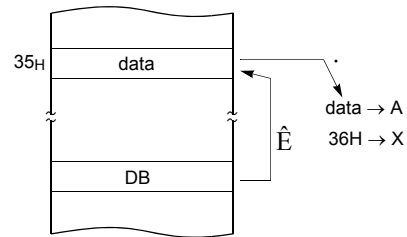
X indexed direct page, auto increment → {X}+

In this mode, a address is specified within direct page by the X register and the content of X is increased by 1.

LDA, STA

Example; X=35H

```
DB LDA {X}+
```



(5) Indexed Addressing

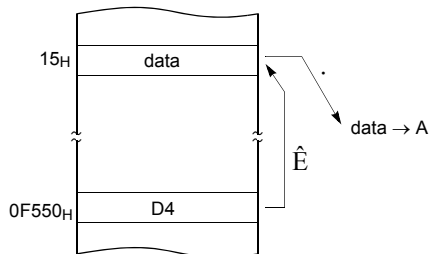
X indexed direct page (no offset) → {X}

In this mode, a address is specified by the X register.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA, XMA

Example; X=15H

```
D4 LDA {X} ; ACC←RAM[X].
```



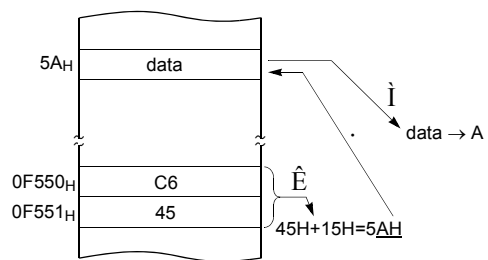
X indexed direct page (8 bit offset) → dp+X

This address value is the second byte (Operand) of command plus the data of X-register. And it assigns the memory in Direct page.

ADC, AND, CMP, EOR, LDA, LDY, OR, SBC, STA
STY, XMA, ASL, DEC, INC, LSR, ROL, ROR

Example; X=015H

```
C645 LDA 45H+X
```



Y indexed direct page (8 bit offset) → dp+Y

This address value is the second byte (Operand) of command plus the data of Y-register, which assigns Memory in Direct page.

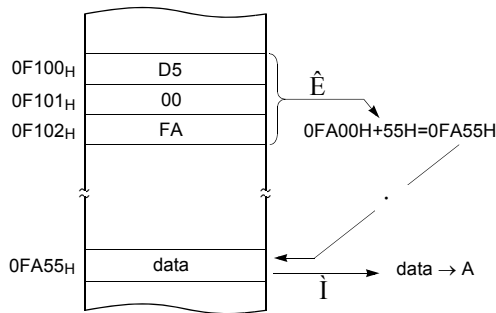
This is same with above (2). Use Y register instead of X.

Y indexed absolute →!abs+Y

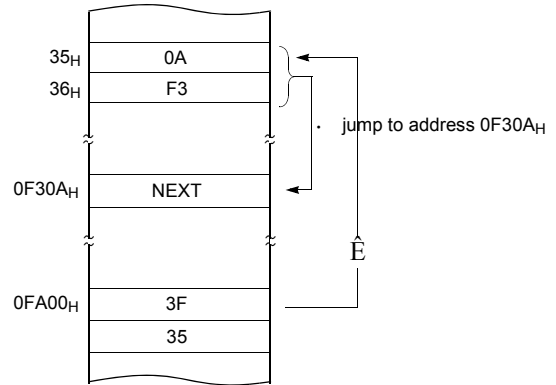
Sets the value of 16-bit absolute address plus Y-register data as Memory. This addressing mode can specify memory in whole area.

Example; Y=55H

```
D500FA LDA !0FA00H+Y
```



```
3F35 JMP [35H]
```



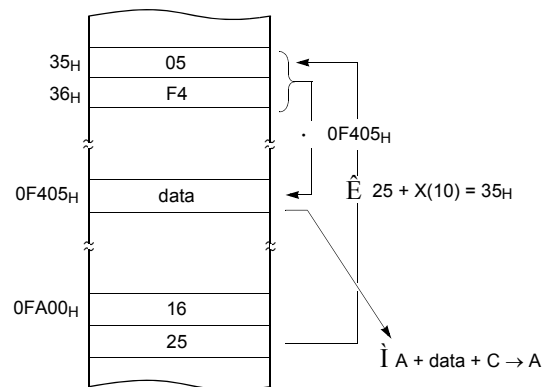
X indexed indirect → [dp+X]

Processes memory data as Data, assigned by 16-bit pair memory which is determined by pair data [dp+X+1][dp+X] Operand plus X-register data in Direct page.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA

Example; X=10H

```
1625 ADC [25H+X]
```



(6) Indirect Addressing

Direct page indirect → [dp]

Assigns data address to use for accomplishing command which sets memory data(or pair memory) by Operand. Also index can be used with Index register X,Y.

JMP, CALL

Example;

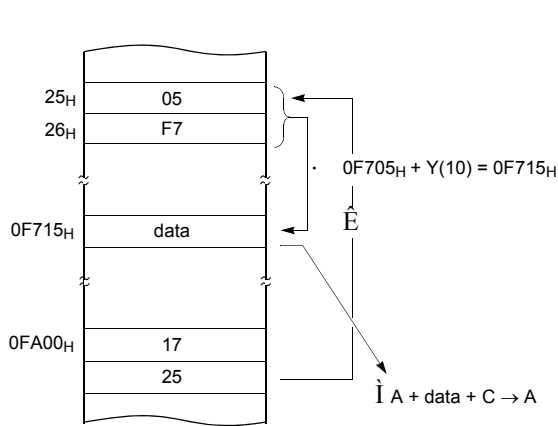
Y indexed indirect → [dp]+Y

Processes memory data as Data, assigned by the data [dp+1][dp] of 16-bit pair memory paired by Operand in Direct page plus Y-register data.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA

Example; Y=10H

```
1725   ADC    [25H]+Y
```



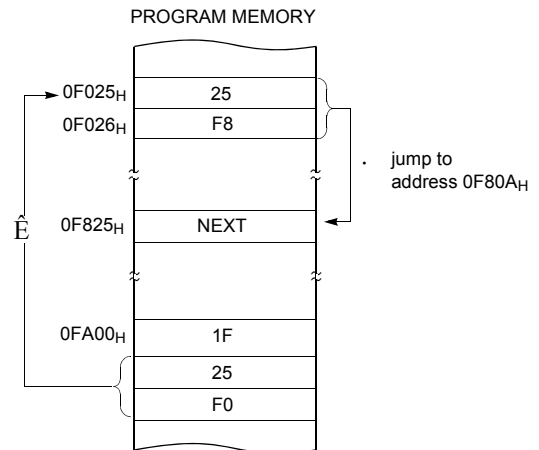
Absolute indirect → [!abs]

The program jumps to address specified by 16-bit absolute address.

JMP

Example;

```
1F25F0   JMP    [!0F025H]
```



9. I/O PORTS

The HMS87C1904 has one port, R0. This port pins may be multiplexed with an alternate function for the peripheral features on the device. In general, when a initial reset state, all ports are used as a general purpose input port.

All pins have data direction registers which can set these ports as output or input. An “1” in the port direction register defines the corresponding port pin as output. Conversely, write “0” to the corresponding bit to specify as an input pin. For example, to use the even numbered bit of R0 as output ports and the odd numbered bits as input ports, write “15H” to address C1H (R0 direction register) during initial setting as shown in Figure 9-1.

Reading data register reads the status of the pins whereas writing to it will write to the port latch.

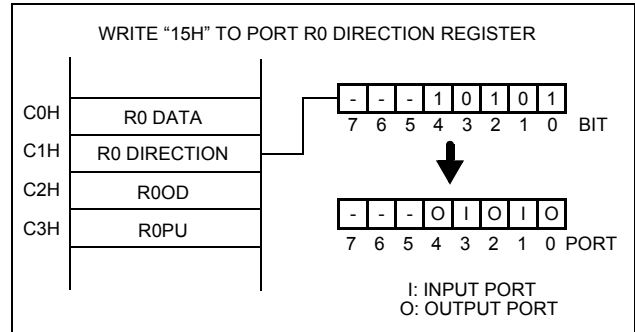


Figure 9-1 Example of port I/O assignment

9.1 R0 and R0IO registers

R0 is a five bidirectional I/O and one input-only port (address C0H). Each port can be set individually as input and output through the R0IO register (address C1H).

R00~R02, and R04 pins are multiplexed with Analog Input Port(AN0~AN2 and AN3) and R00 port is multiplexed with Timer0 output(T0O) or PWM1output (PWM1O).

The control register R0FUNC (address C4H) controls to select alternate function. After reset, this value is “0”, port may be used as general I/O ports. To select alternate function such as Analog Input or External Event Counter Input, write “1” to the corresponding bit of R0FUNC.Regardless of the direction register R0IO, R0FUNC is selected to use as alternate functions, port pin can be used as a corresponding alternate features.

PORT	R0FUNC.3~0	Description
R05/X _{IN}	Internal CLK ¹	RA5 (Normal I/O Port)
	External CLK	X _{IN}
R04/AN3/ X _{OUT}	Internal CLK	RA4 (Normal I/O Port) AN4 (ADS1.0=11)
	External CLK	X _{OUT}
R03/V _{PP} / RESET	RESETE=1 ²	RA3 (Normal I/O Port)
	PGM Mode	V _{PP}
	RESETE=0	RESET
R02/AN2/ EC0/INT0	INT0=0 EC0=0	RA2 (Normal I/O Port) AN2 (ADS1.0=10)
	INT0=1	INT0
	EC0=1	EC0 (T0CK2~0=111)
R01/AN1	-	R01 (Normal I/O Port) AN1 (ADS1.0=01)
R00/AN0/ T0O/PWM1O	T0O/ PWM1O=0	R00 (Normal I/O Port) AN0 (ADS1.0=00)
	T0O/ PWM1O=0	PWM1O

1. This function is programmed by writing option.
2. This bit is located in Configuration area(201FH) and it is programmed by writing option.

Figure 9-2 Alternate Functions of Port 0

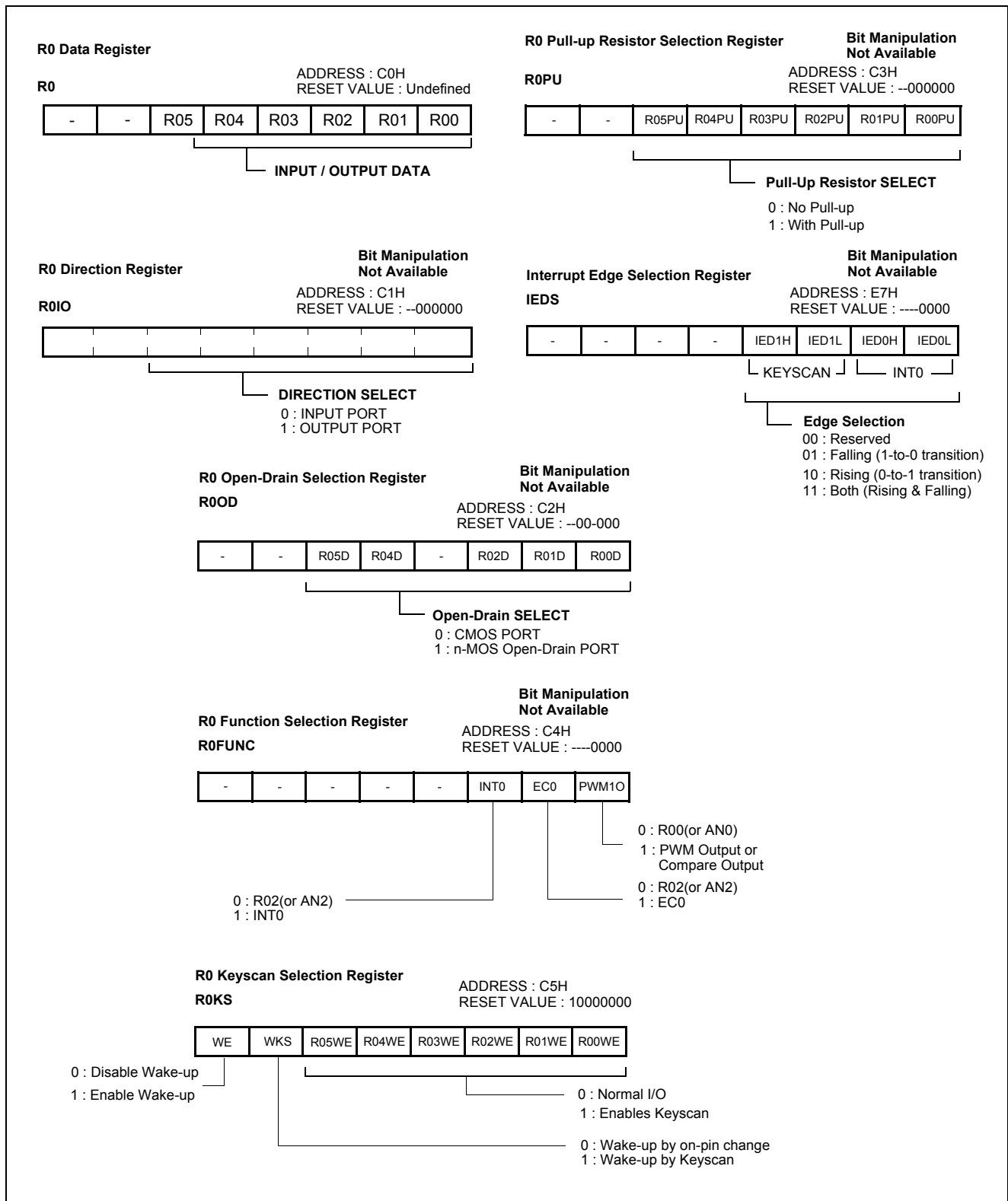


Figure 9-3 Registers of Port R0

10. CLOCK GENERATOR

The clock generator produces the basic clock pulses which provide the system clock to be supplied to the CPU and peripheral hardware. The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator connected to the

X_{IN} and X_{OUT} pins. External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the X_{IN} pin and open the X_{OUT} pin.

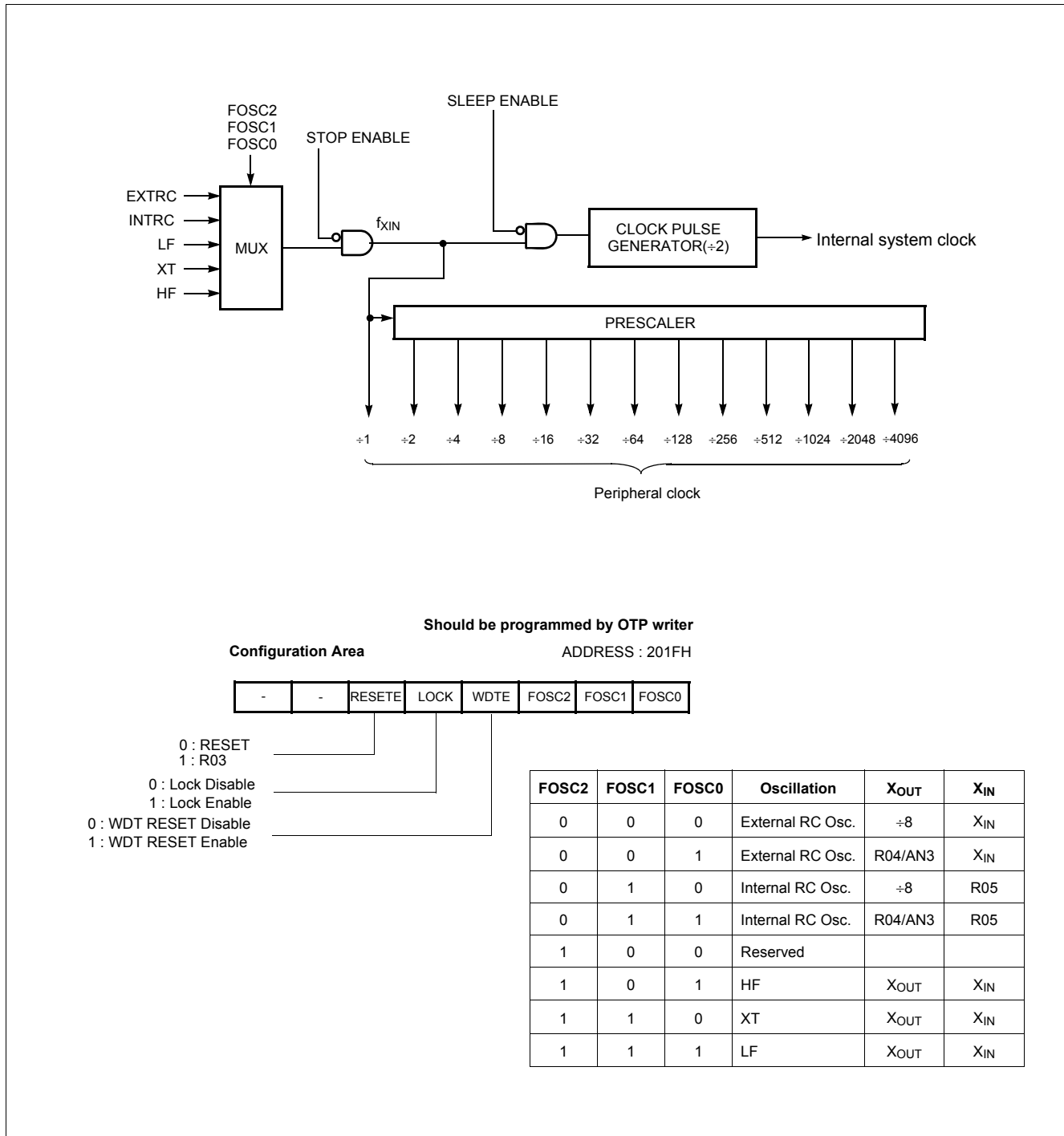


Figure 10-1 Block Diagram of Clock Pulse Generator

10.1 Oscillation Circuit

X_{IN} and X_{OUT} are the input and output, respectively, a inverting amplifier which can be set for use as an on-chip oscillator, as shown in Figure 10-2.

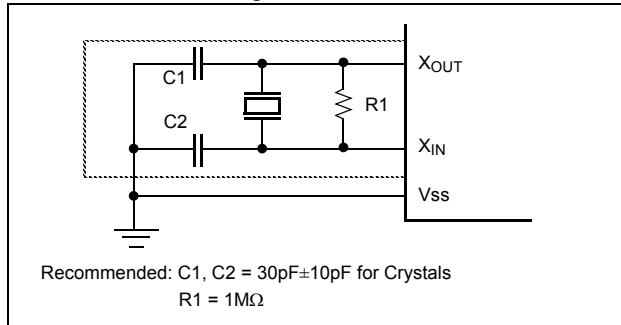


Figure 10-2 Oscillator Connections

To drive the device from an external clock source, X_{out} should be left unconnected while X_{in} is driven as shown in Figure 10-3. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.

Oscillation circuit is designed to be used either with a ceramic resonator or crystal oscillator. Since each crystal and ceramic resonator have their own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

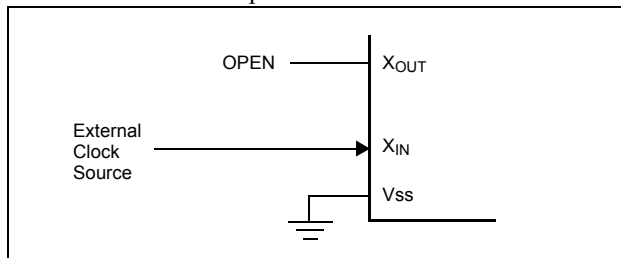


Figure 10-3 External Clock Connections

Note: When using a system clock oscillator, carry out wiring in the broken line area in Figure 10-2 to prevent any effects from wiring capacitances.

- Minimize the wiring length.
- Do not allow wiring to intersect with other signal conductors.
- Do not allow wiring to come near changing high current.
- Set the potential of the grounding position of the oscillator capacitor to that of V_{SS} . Do not ground to any ground pattern where high current is present.
- Do not fetch signals from the oscillator.

In addition, the HMS87C1904 has an ability for the external RC oscillated operation. It offers additional cost savings for **timing insensitive applications**. The RC oscillator frequency is a function of the supply voltage, the external resistor (R_{EXT}) and capacitor (C_{EXT}) values, and the operating temperature.

The user needs to take into account variation due to tolerance of external R and C components used.

Figure 10-4 shows how the RC combination is connected to the HMS87C1904.

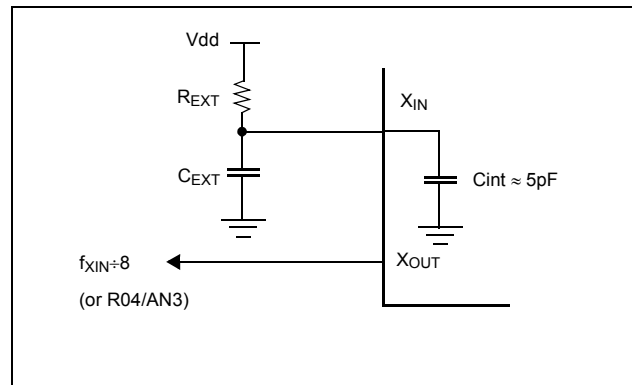


Figure 10-4 RC Oscillator Connections

External capacitor (C_{EXT}) can be omitted for more cost saving. However, the characteristics of external R only oscillation are more variable than external RC oscillation.

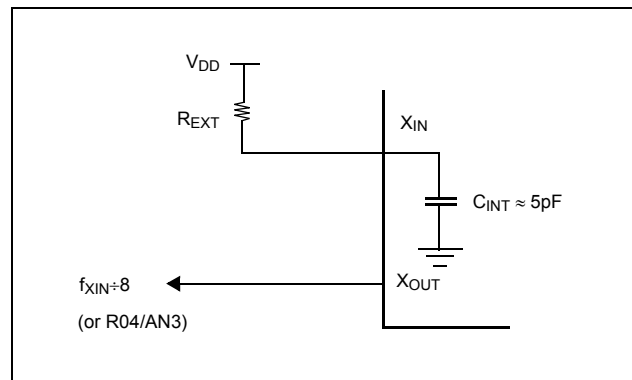


Figure 10-5 R Oscillator Connections

The oscillator frequency, divided by 8, is output from the X_{out} pin, and can be used for test purpose or to synchronize other logic.

10.2 Internal 4MHz Oscillator

The HMS87C1904 has an internal RC ring oscillator that provides a fixed 4MHz (typical) system clock without any external components.

This oscillation is performed by setting the oscillation selection bits of Configuration Area(201F_H) and it should be programmed Option writing in the OTP writer.

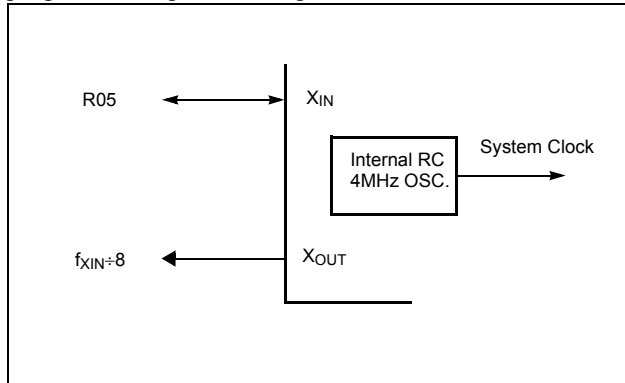


Figure 10-6 Internal RC Oscillator

The oscillator frequency, divided by 8, is output from the Xout pin, and can be used for test purpose or to synchronize other logic.

In the internal oscillation mode, X_{IN} and X_{OUT} pins serve as normal I/O (R04/AN3 and R03) port by setting the Option writing in the OTP wrtier.

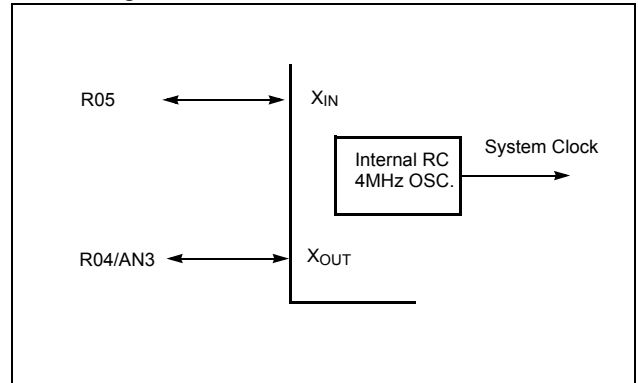


Figure 10-7 Internal RC Oscillator within Normal Port

11. BASIC INTERVAL TIMER

The HMS87C1904 has one 8-bit Basic Interval Timer that is free-run, can not stop. Block diagram is shown in Figure 11-1. The 8-bit Basic interval timer register (BITR) is increased every internal count pulse which is divided by prescaler. Since prescaler has divided ratio by 8 to 1024, the count rate is 1/8 to 1/1024 of the oscillator frequency. As the count overflows from FF_H to 00_H, this overflow causes to generate the Basic interval timer interrupt. The BITIF is interrupt request flag of Basic interval timer.

When write “1” to bit BTCL of CKCTLR, BITR register is cleared to “0” and restart to count-up. The bit BTCL becomes “0” after one machine cycle by hardware.

If the STOP instruction executed after writing “1” to bit

RCWDT of CKCTLR, it goes into the internal RC oscillated watchdog timer mode. In this mode, all of the block is halted except the internal RC oscillator, Basic Interval Timer and Watchdog Timer. More detail informations are explained in Power Saving Function. The bit WDTON decides Watchdog Timer or the normal 7-bit timer

Note: All control bits of Basic interval timer are in CKCTLR register which is located at same address of BITR (address E0_H). Address E0_H is read as BITR, written to CKCTLR. Therefore, the CKCTLR can not be accessed by bit manipulation instruction.

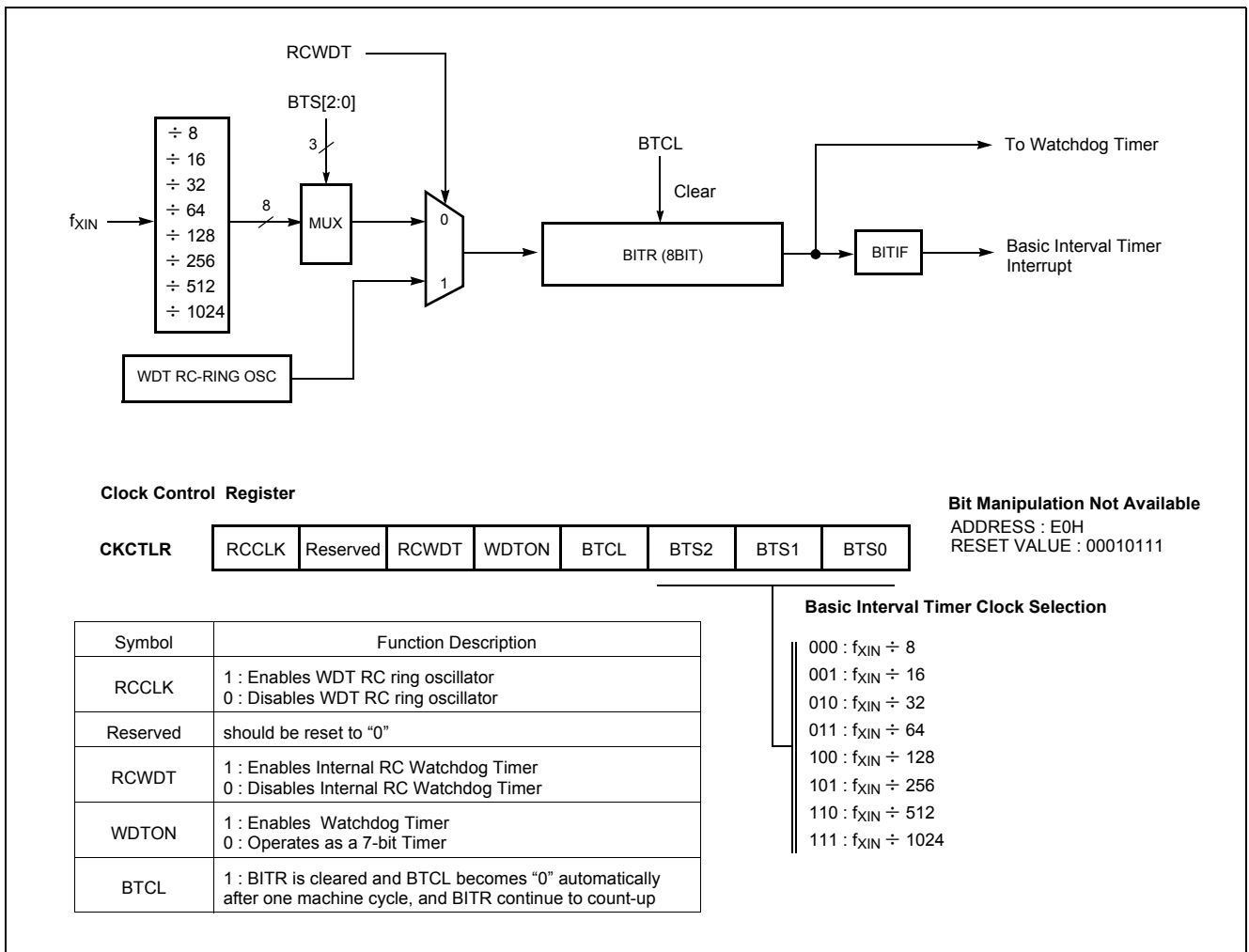


Figure 11-1 Block Diagram of Basic Interval Timer

12. TIMER / COUNTER

The HMS87C1904 has two Timer/Counter registers. Each module can generate an interrupt to indicate that an event has occurred (i.e. timer match).

Timer 0 and Timer 1 can be used either two 8-bit Timer/Counter or the one 16-bit Timer/Counter by combining them.

In the “timer” function, the register is increased every internal clock input. Thus, one can think of it as counting internal clock input. Since the least clock consists of 2 and the most clock consists of 2048 oscillator periods, the count rate is 1/2 to 1/2048 of the oscillator frequency in Timer0. And Timer1 can use the same clock source too. In addition, Timer1 has more fast clock source (1/1 to 1/8).

In the “counter” function, the register is increased in response to a 0-to-1 (rising edge) transition at its corresponding external input pin, EC0(Timer 0).

Note: If changing the Timer value or starting again, it should be stop the timer clock firstly, and then set Timer register value.

```
Ex)  LDM  TM0, #00001100B
      LDM  TDR, #7FH
      LDM  TM0, #00010111B
```

In addition the “capture” function, the register is increased in response external interrupt same with timer function. When external interrupt edge input, the count register is captured into capture data register CDRx.

Timer1 is shared with “PWM” function and “Compare output” function

It has seven operating modes: “8-bit timer/counter”, “16-bit timer/counter”, “8-bit capture”, “16-bit capture”, “8-bit compare output”, “16-bit compare output” and “10-bit PWM” which are selected by bit in Timer mode register TMx as shown in Figure 12-1 and Table 12-1.

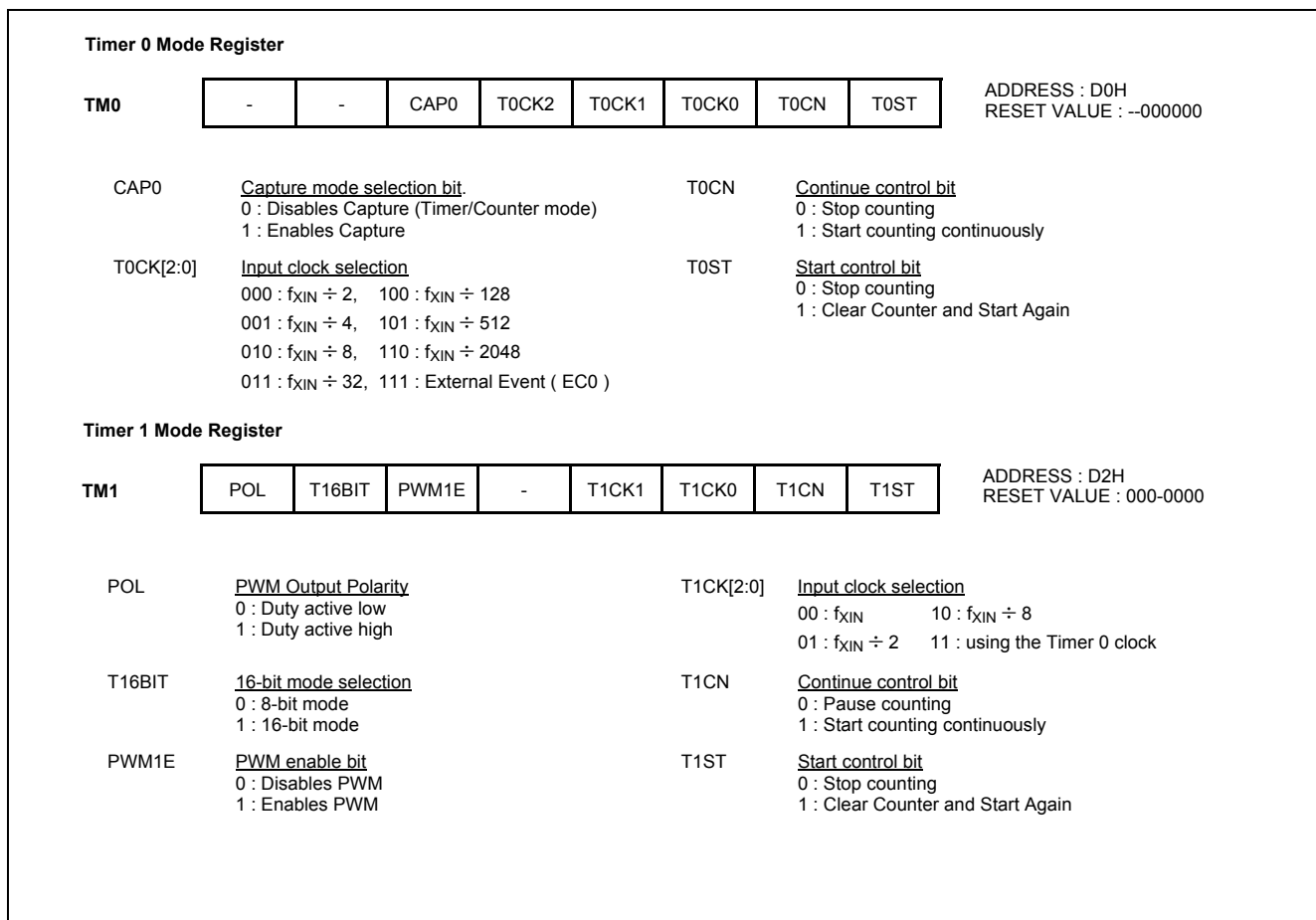


Figure 12-1 Timer Mode Register (TM0, TM1)

T16BIT	CAP0	PWM1E	T0CK[2:0]	T1CK[1:0]	PWM1O	TIMER 0	TIMER1
0	0	0	XXX	XX	0	8-bit Timer	8-bit Timer
0	0	0	111	XX	0	8-bit Event Counter	8-bit Capture
0	1	0	XXX	XX	1	8-bit Capture	8-bit Compare output
0	X ¹	1	XXX	XX	1	8-bit Timer/Counter	10-bit PWM
1	0	0	XXX	11	0	16-bit Timer	
1	0	0	111	11	0	16-bit Event Counter	
1	1	0	XXX	11	0	16-bit Capture	
1	0	0	XXX	11	1	16-bit Compare output	

Table 12-1 Operating Modes of Timer 0 and Timer 1

1. X: The value "0" or "1" corresponding your operation.

12.1 8-bit Timer/Counter Mode

The HMS87C1904 has four 8-bit Timer/Counters, Timer 0 and Timer 1 as shown in Figure 12-2.

The "timer" or "counter" function is selected by mode registers TMx as shown in Figure 12-1 and Table 12-1. To use

as an 8-bit timer/counter mode, bit CAP0 of TM0 is cleared to "0" and bits T16BIT of TM1 should be cleared to "0"(Table 12-1).

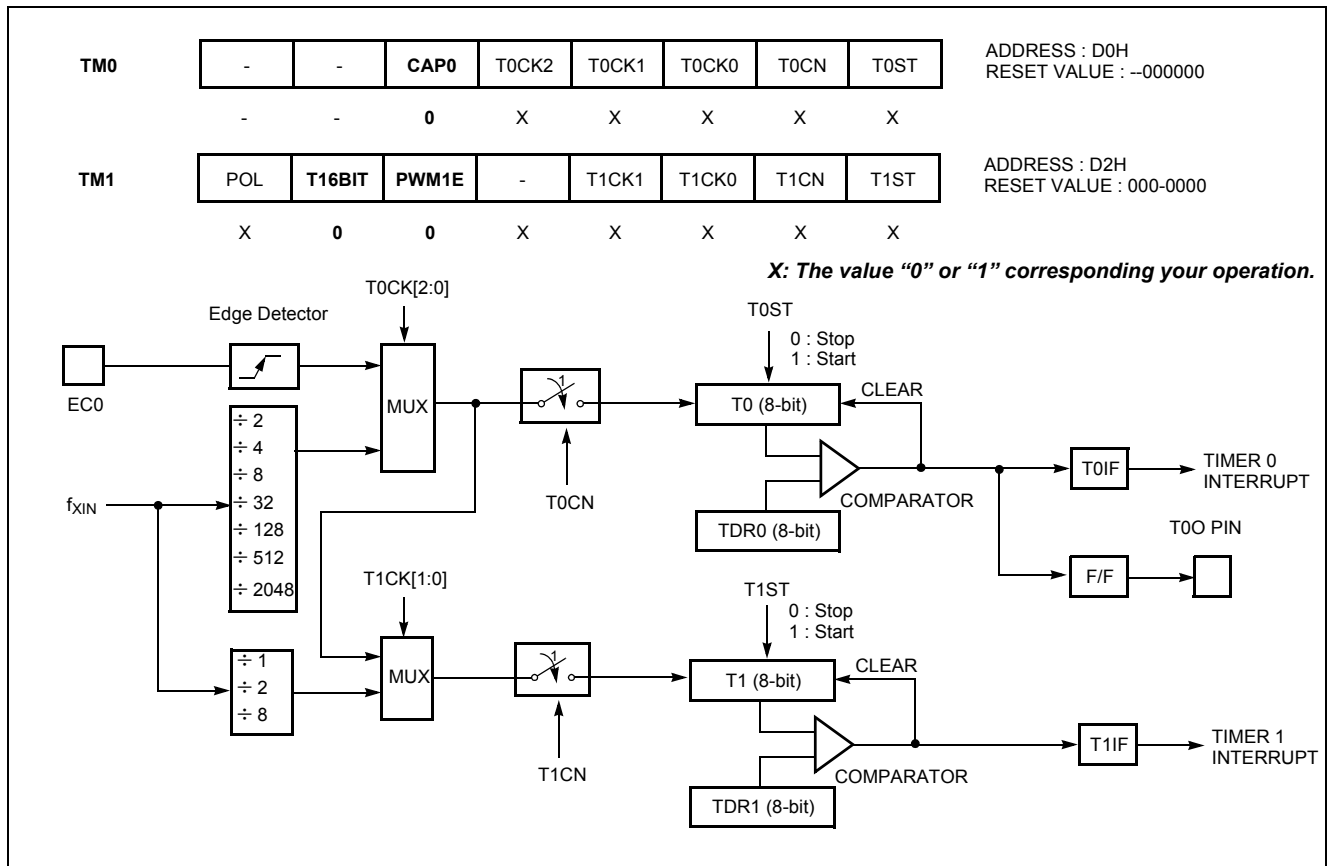


Figure 12-2 8-bit Timer / Counter Mode

These timers have each 8-bit count register and data register. The count register is increased by every internal or external clock input. The internal clock has a prescaler divide ratio option of 2, 4, 8, 32, 128, 512, 2048 (selected by control bits T0CK2, T0CK1 and T0CK0 of register TM0) and 1, 2, 8 (selected by control bits T1CK1 and T1CK0 of register TM1). In the Timer 0, timer register T0 increases from 00_H until it matches TDR0 and then reset to 00_H. The match output of Timer 0 generates Timer 0 interrupt

(latched in T0F bit). As TDRx and Tx register are in same address, when reading it as a Tx, written to TDRx.

In counter function, the counter is increased every 0-to 1 (rising edge) transition of EC0 pin. In order to use counter function, the bit R02 of the R0 Direction Register R0IO is set to "0". The Timer 0 can be used as a counter by pin EC0 input, but Timer 1 can not.

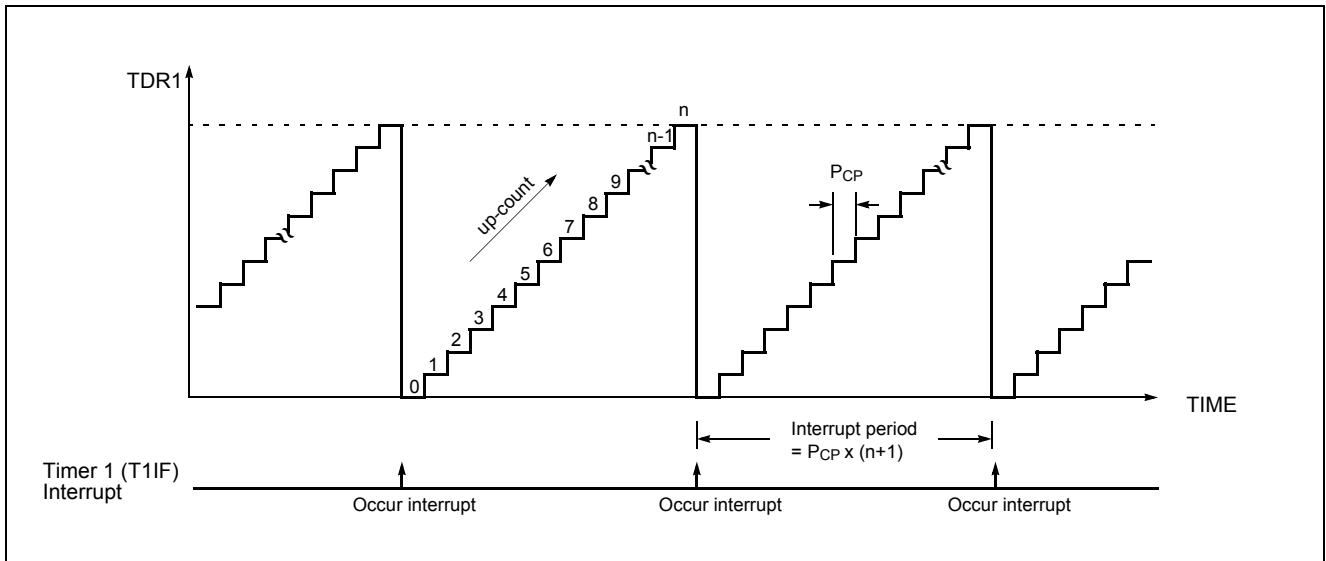


Figure 12-3 Counting Example of Timer Data Registers

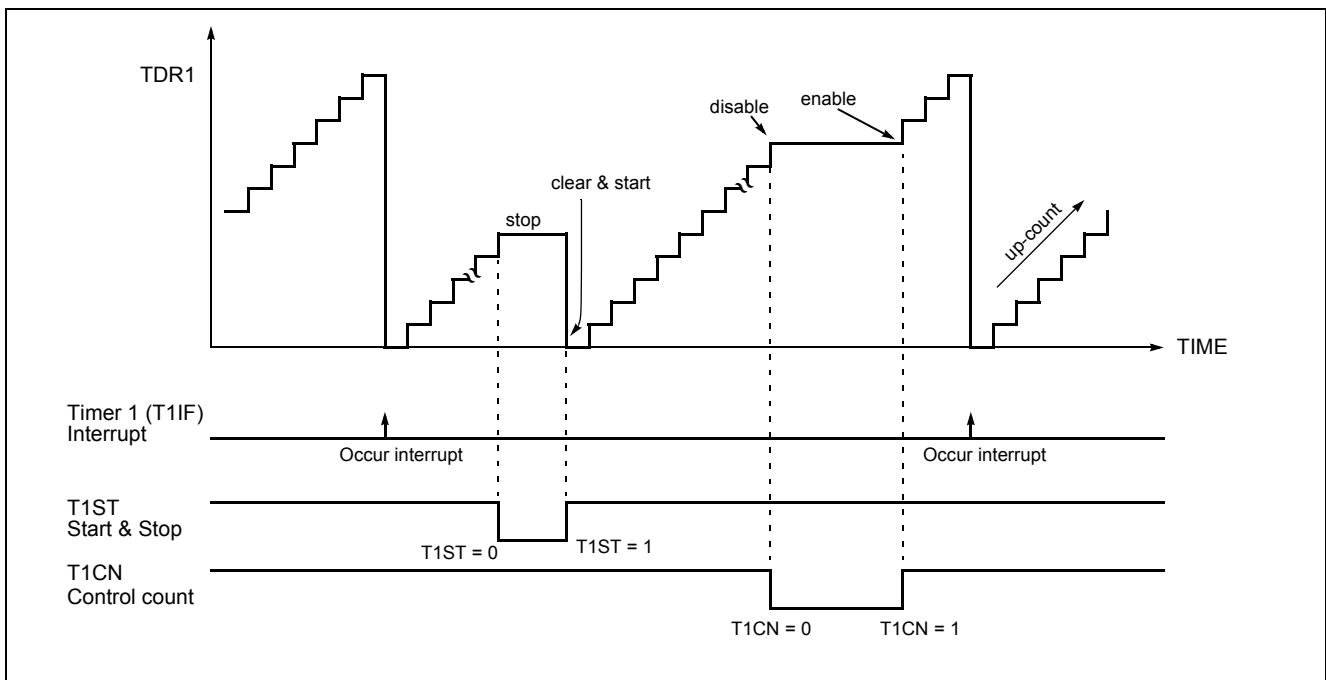


Figure 12-4 Timer Count Operation

12.2 16-bit Timer/Counter Mode

The Timer register is being run with 16 bits. A 16-bit timer/counter register T0, T1 are increased from 0000_H until it matches TDR0, TDR1 and then resets to 0000_H. The match output generates Timer 0 interrupt not Timer 1 interrupt.

The clock source of the Timer 0 is selected either internal or external clock by bit T0CK2, T0CK1 and T0CK0.

In 16-bit mode, the bits T1CK1, T1CK0 and T16BIT of TM1 should be set to “1” respectively.

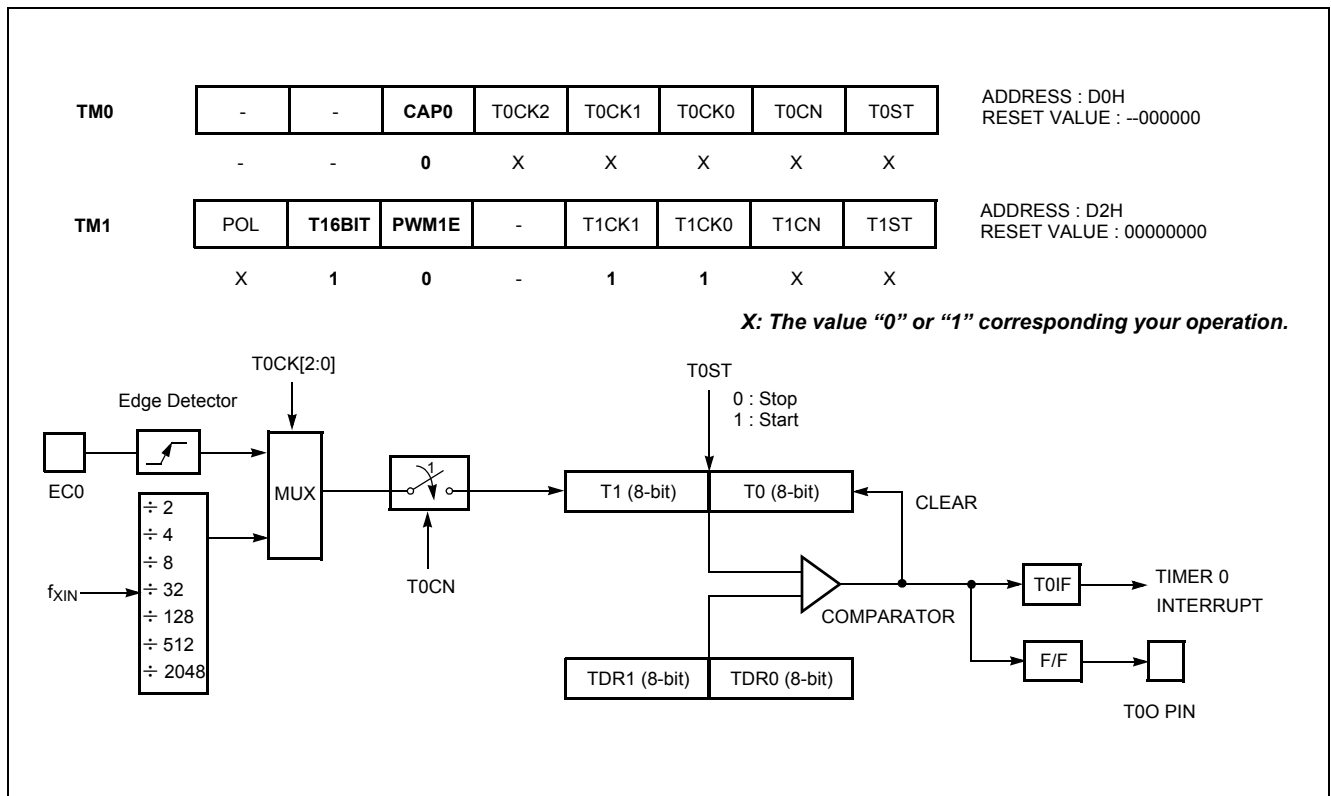


Figure 12-5 16-bit Timer / Counter Mode

12.3 8-bit Compare Output (16-bit)

The HMS87C1904 has a function of Timer Compare Output. To pulse out, the timer match can go to port pin(TIO) as shown in Figure 12-2 and Figure 12-5. Thus, pulse out is generated by the timer match. These operation is implemented to pin, R00/TIO/PWM1O.

This pin output the signal having a 50: 50 duty square

wave, and output frequency is same as below equation.

$$f_{COMP} = \frac{\text{Oscillation Frequency}}{2 \times \text{Prescaler Value} \times (TDR + 1)}$$

In this mode, the bit PWM0 of R0 function register (R0FUNC) should be set to “1”, and the bit PWM1E of timer1 mode register (TM1) should be set to “0”.

In addition, 16-bit Compare output mode is available, also.

12.4 8-bit Capture Mode

The Timer 0 capture mode is set by bit CAPO of timer mode register TM0 as shown in Figure 12-6.

As mentioned above, not only Timer 0 but Timer 1 can also be used as a capture mode.

The Timer/Counter register is increased in response internal or external input. This counting function is same with normal timer mode, and Timer interrupt is generated when timer register T0 (T1) increases and matches TDR0 (TDR1).

This timer interrupt in capture mode is very useful when the pulse width of captured signal is more wider than the maximum period of Timer.

For example, in Figure 12-8, the pulse width of captured signal is wider than the timer data value (FF_H) over 2 times. When external interrupt is occurred, the captured value (13_H) is more little than wanted value. It can be obtained correct value by counting the number of timer overflow occurrence.

Timer/Counter still does the above, but with the added feature that a edge transition at external input INT_x pin causes the current value in the Timer x register (T0,T1), to be captured into registers CDR_x (CDR0, CDR1), respectively. After captured, Timer x register is cleared and restarts by

hardware.

It has three transition modes: “falling edge”, “rising edge”, “both edge” which are selected by interrupt edge selection register IEDS (Refer to External interrupt section). In addition, the transition at INT0 pin generate an interrupt.

Note: The CDR_x, TDR_x and T_x are in same address. In the capture mode, reading operation read the CDR_x, not T_x because the reading path is opened to the CDR_x, and TDR_x is read while writing operation executed.

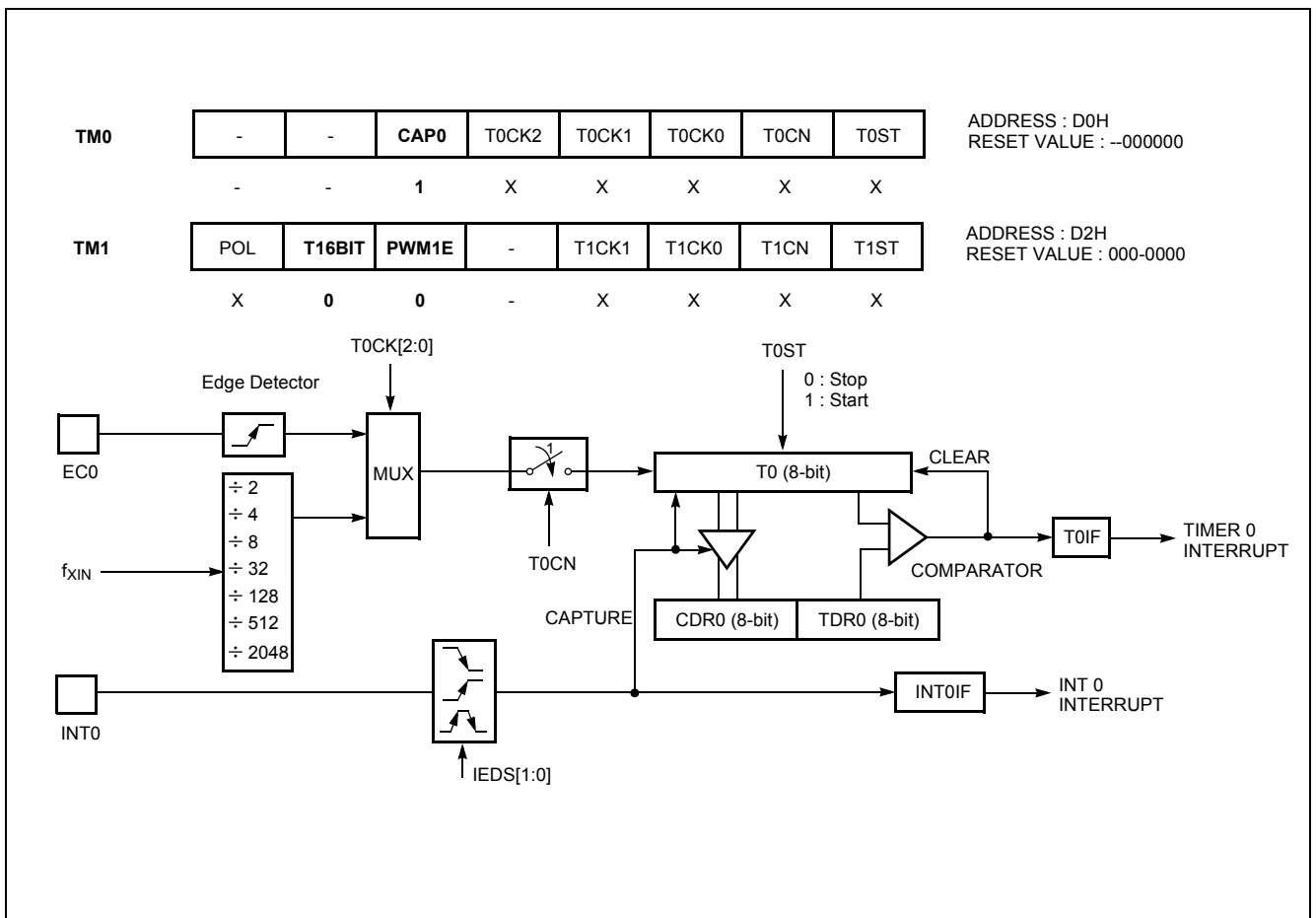


Figure 12-6 8-bit Capture Mode

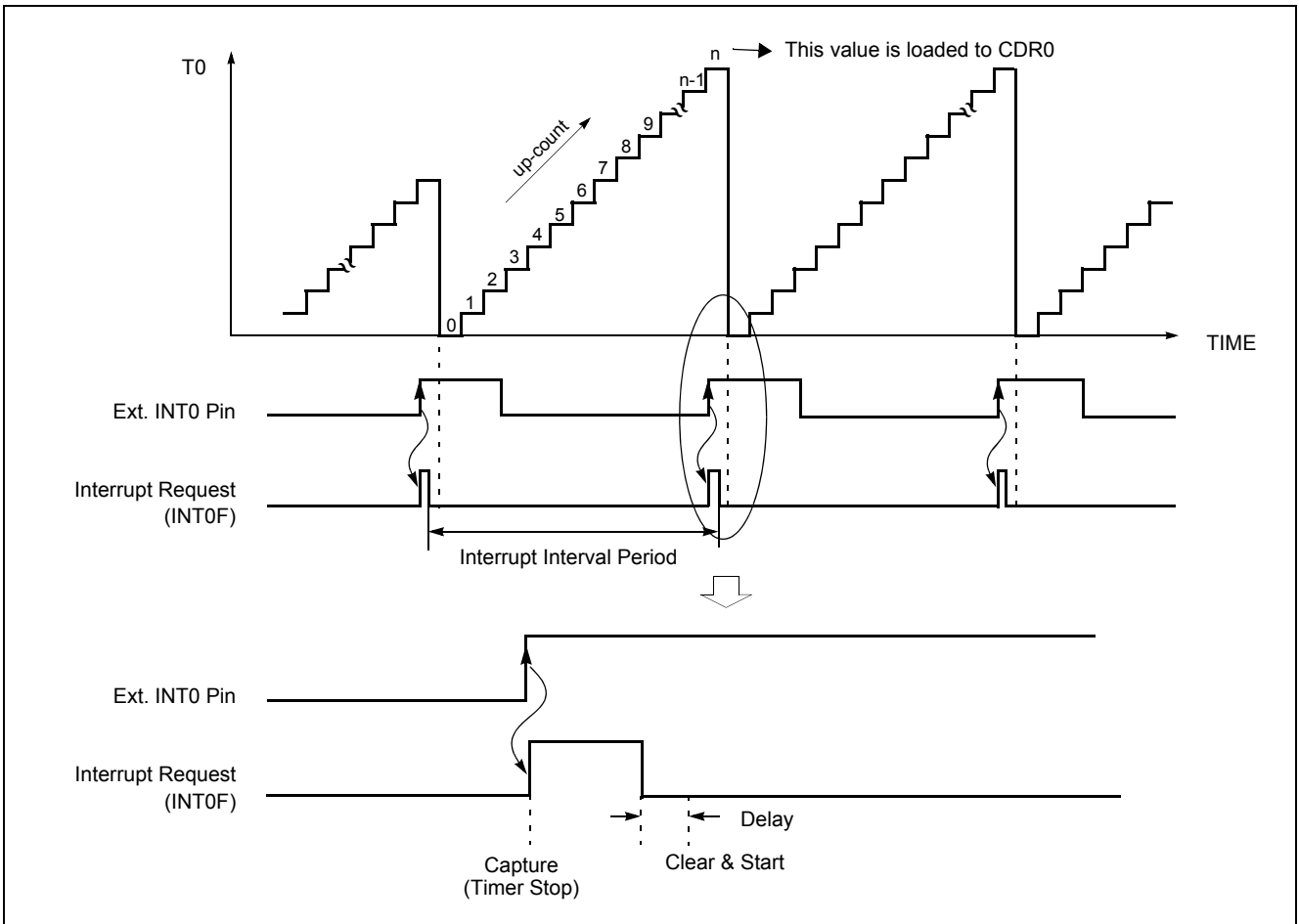


Figure 12-7 Input Capture Operation

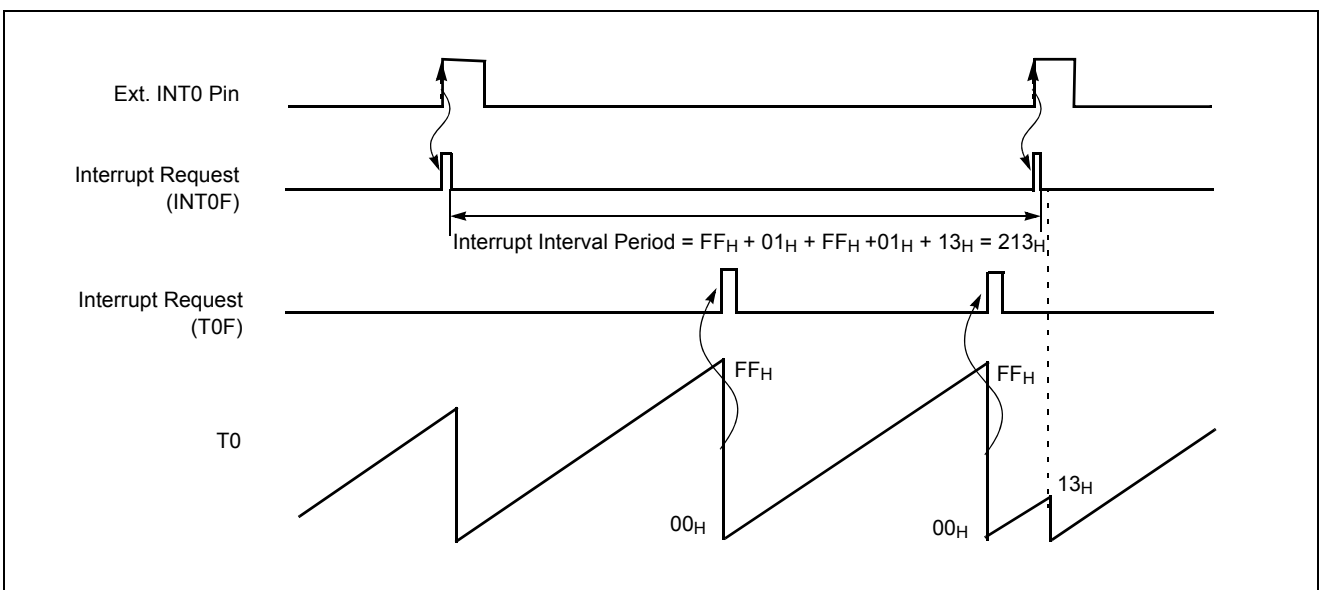


Figure 12-8 Excess Timer Overflow in Capture Mode

12.5 16-bit Capture Mode

16-bit capture mode is the same as 8-bit capture, except that the Timer register is being run will 16 bits.

In 16-bit mode, the bits T1CK1, T1CK0 and T16BIT of TM1 should be set to "1" respectively.

The clock source of the Timer 0 is selected either internal or external clock by bit T0CK2, T0CK1 and T0CK0.

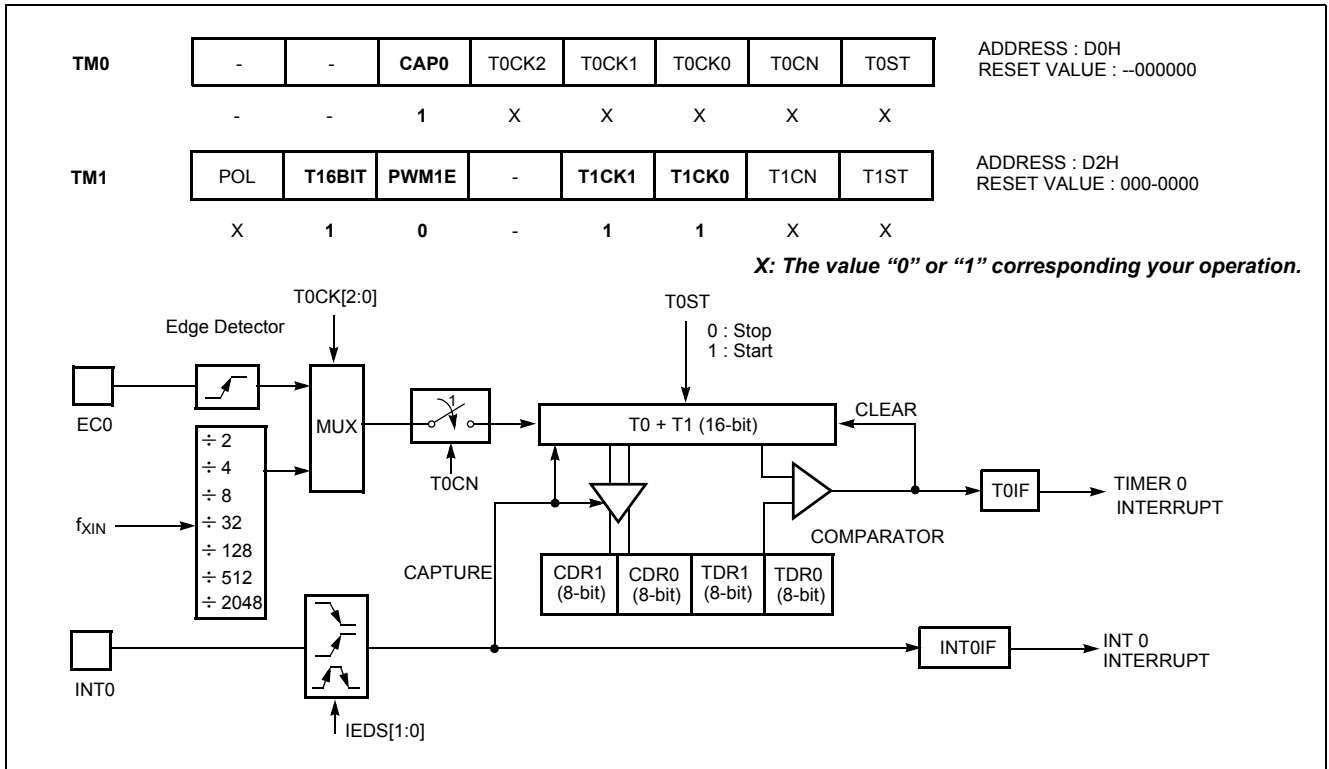


Figure 12-9 16-bit Capture Mode

12.6 PWM Mode

The HMS87C1904 has a high speed PWM (Pulse Width Modulation) functions which shared with Timer1.

In PWM mode, pin R00/T00/PWM10 outputs up to a 10-bit resolution PWM output. This pin should be configure as a PWM output by setting "1" bit PWM10 in R0FUNC register.

The period of the PWM output is determined by the T1PPR (PWM1 Period Register) and PWM1HR[3:2] (bit3,2 of PWM1 High Register) and the duty of the PWM output is determined by the T1PDR (PWM1 Duty Register) and PWM1HR[1:0] (bit1,0 of PWM1 High Register).

The user writes the lower 8-bit period value to the T1PPR and the higher 2-bit period value to the PWM1HR[3:2].

And writes duty value to the T1PDR and the PWM1HR[1:0] same way.

The T1PDR is configure as a double buffering for glitchless PWM output. In Figure 12-10, the duty data is transferred from the master to the slave when the period data matched to the counted value. (i.e. at the beginning of next duty cycle)

$$PWM\ Period = [PWM1HR[3:2]T1PPR] \times Source\ Clock$$

$$PWM\ Duty = [PWM1HR[1:0]T1PDR] \times Source\ Clock$$

The relation of frequency and resolution is in inverse proportion. Table 12-2 shows the relation of PWM frequency vs. resolution.

If it needed more higher frequency of PWM, it should be reduced resolution.

Resolution	Frequency		
	T1CK[1:0]=00(125nS)	T1CK[1:0]=01(250nS)	T1CK[1:0]=10(1uS)
10-bit	7.8KHz	3.9KHz	0.98KHz
9-bit	15.6KHz	7.8KHz	1.95KHz
8-bit	31.2KHz	15.6KHz	3.90KHz
7-bit	62.5KHz	31.2KHz	7.81KHz

Table 12-2 PWM Frequency vs. Resolution at 8MHz

The bit POL of TM1 decides the polarity of duty cycle.

If the duty value is set same to the period value, the PWM output is determined by the bit POL (1: High, 0: Low). And if the duty value is set to "00H", the PWM output is determined by the bit POL (1: Low, 0: High).

It can be changed duty value when the PWM output. However the changed duty value is output after the current period is over. And it can be maintained the duty value at present output when changed only period value shown as Figure 12-12. As it were, the absolute duty time is not changed in varying frequency. But the changed period value must greater than the duty value.

Note: If changing the Timer1 to PWM function, it should be stop the timer clock firstly, and then set period and duty register value. If user writes register values while timer is in operation, these register could be set with certain values.

```

Ex) LDM TM1,#00H
     LDM T1PPR,#00H
     LDM T1PDR,#00H
     LDM PWM1HR,#00H
     LDM R0FUNC,#0000_0001B
     LDM TM1,#1010_1011B
    
```

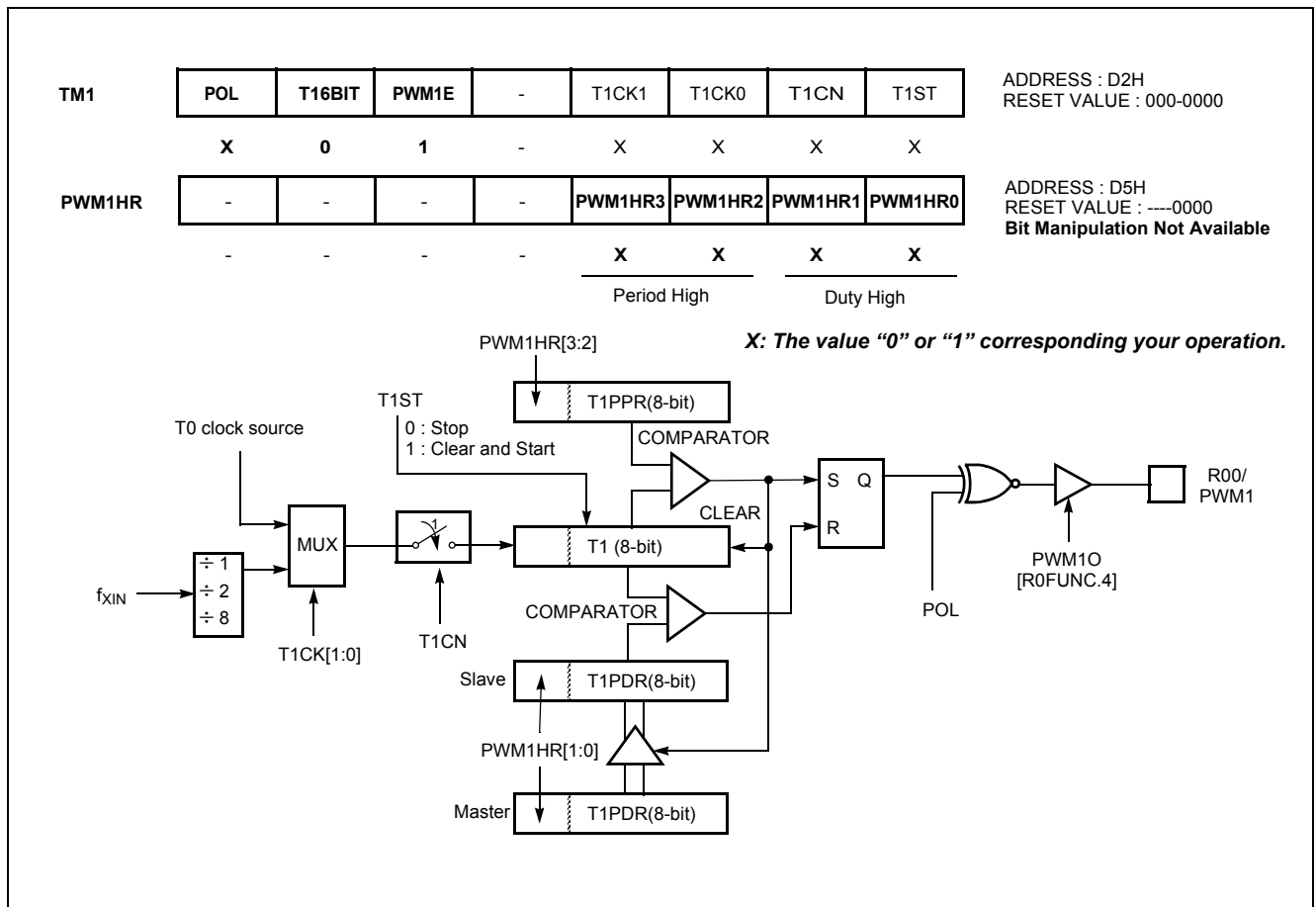


Figure 12-10 PWM Mode

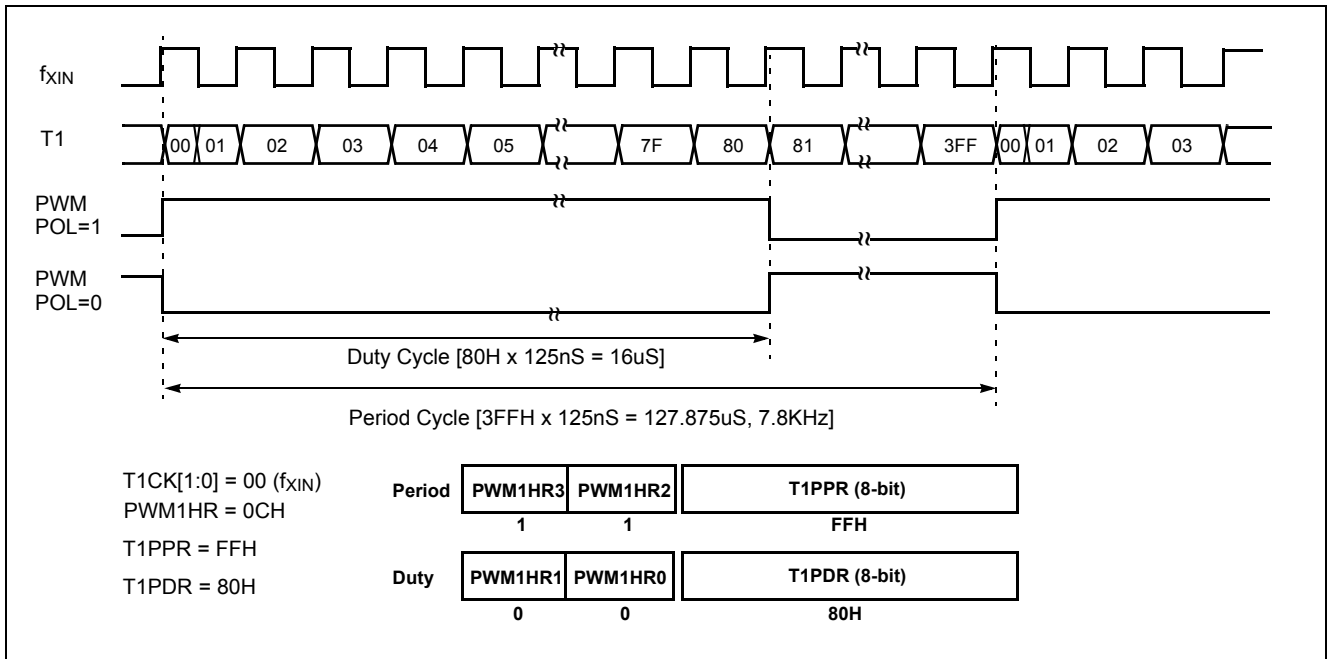


Figure 12-11 Example of PWM at 8MHz

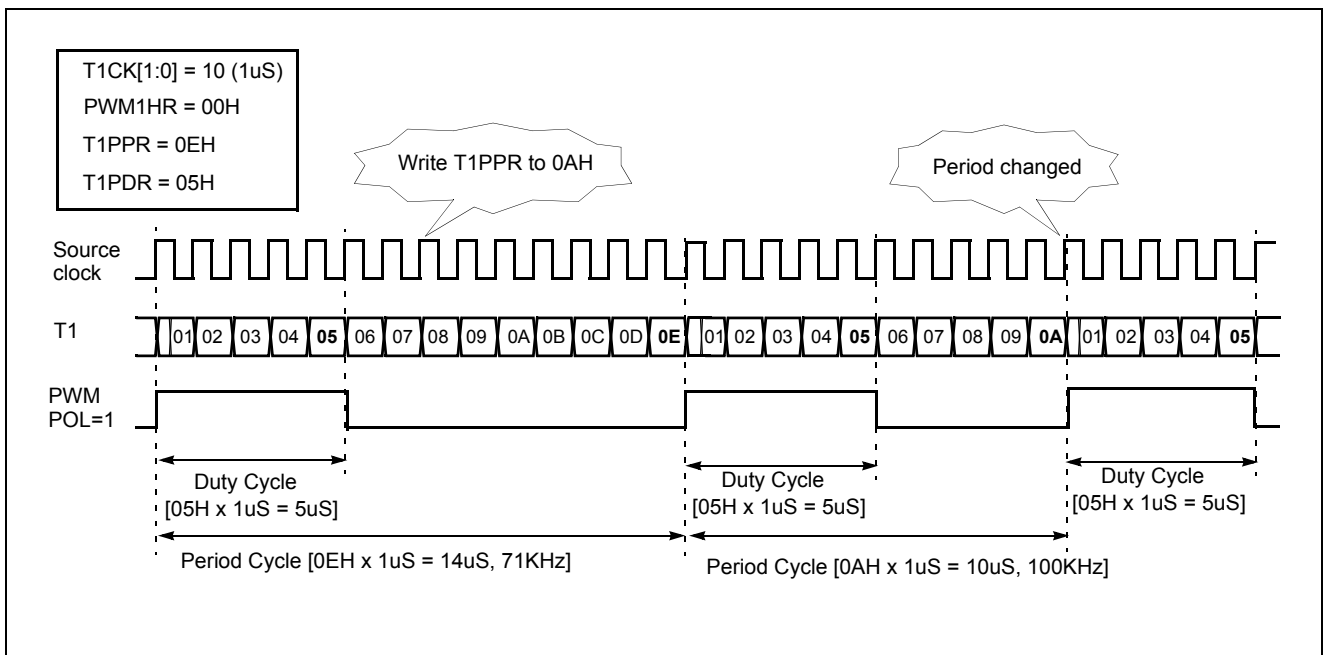


Figure 12-12 Example of Changing the Period in Absolute Duty Cycle (@8MHz)

13. ANALOG TO DIGITAL CONVERTER

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 A/D module has two registers which are the control register ADCM and A/D result register ADCR. The ADCM register, shown in Figure 13-2, controls the operation of the A/D converter module. The port pins can be configured as analog inputs or digital I/O.

To use analog inputs, A/D converter should be enabled by setting the bit ADEN of ADCM register to "1" and each port is assigned analog input port by setting the corre-

sponding channel to be converted the bit ADS[1:0] in ADCM register.

The processing of conversion start when the start bit ADST is set to "1". After one cycle, it is cleared by hardware. The register ADCR contains the results of the A/D conversion. When the conversion is completed, the result is loaded into the ADCR, the A/D conversion status bit ADSF is set to "1", and the A/D interrupt flag ADIF is set. The block diagram of the A/D module is shown in Figure 13-1. The A/D status bit, ADSF is set automatically when A/D conversion is completed, cleared when A/D conversion is in process. The conversion time takes maximum 10 μ S (at $f_{XIN}=8$ MHz).

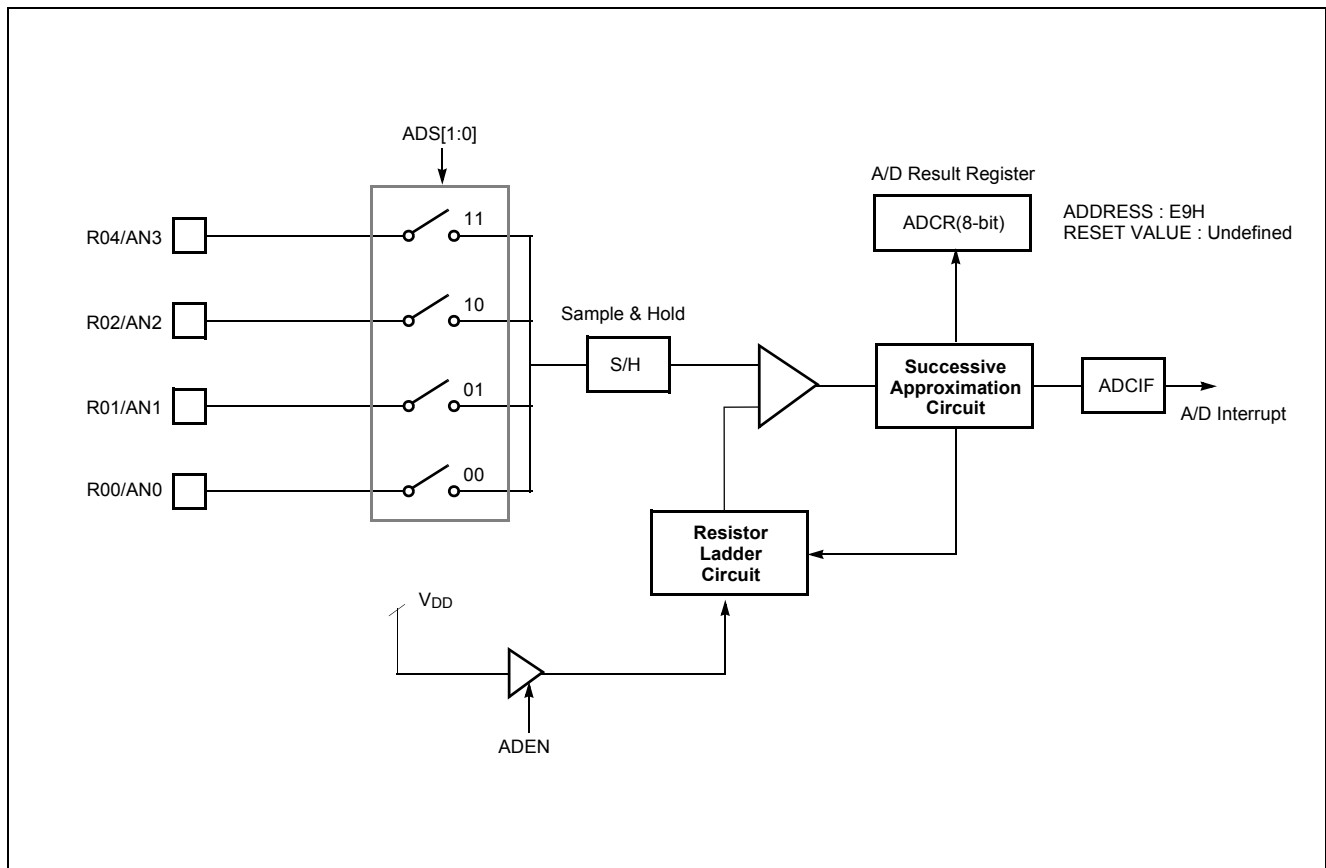


Figure 13-1 A/D Converter Block Diagram

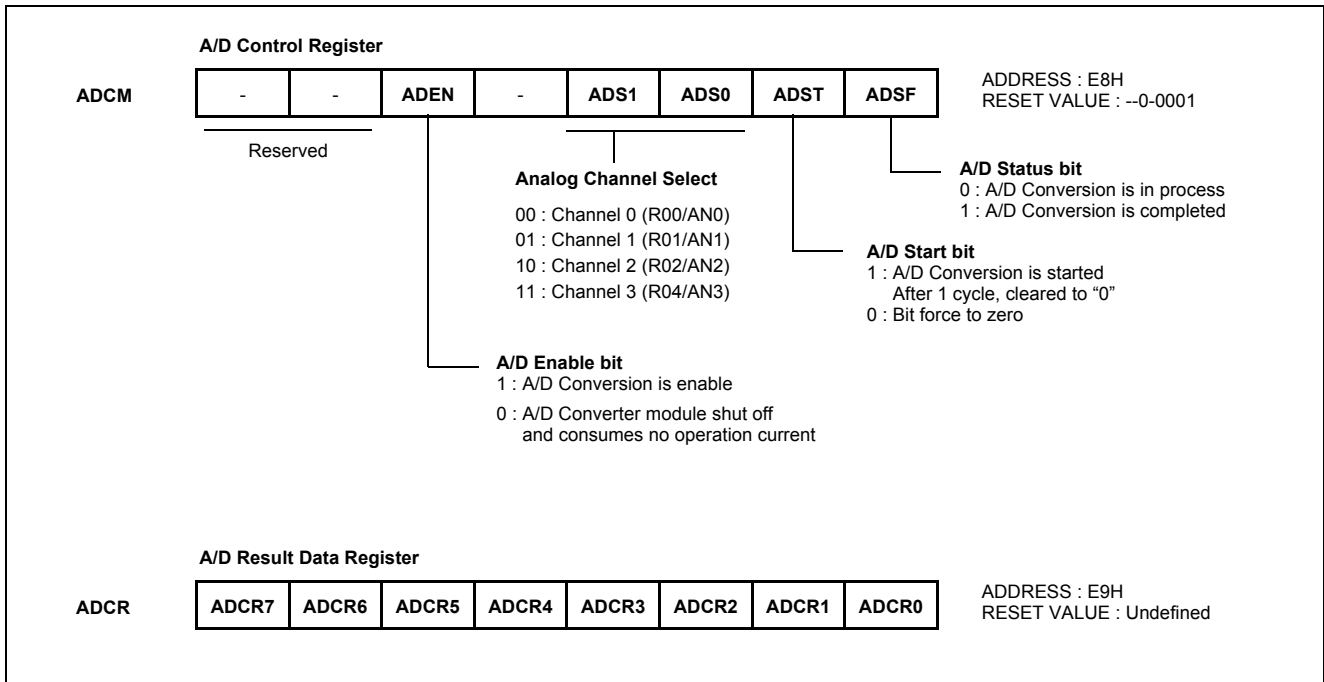


Figure 13-2 A/D Converter Registers

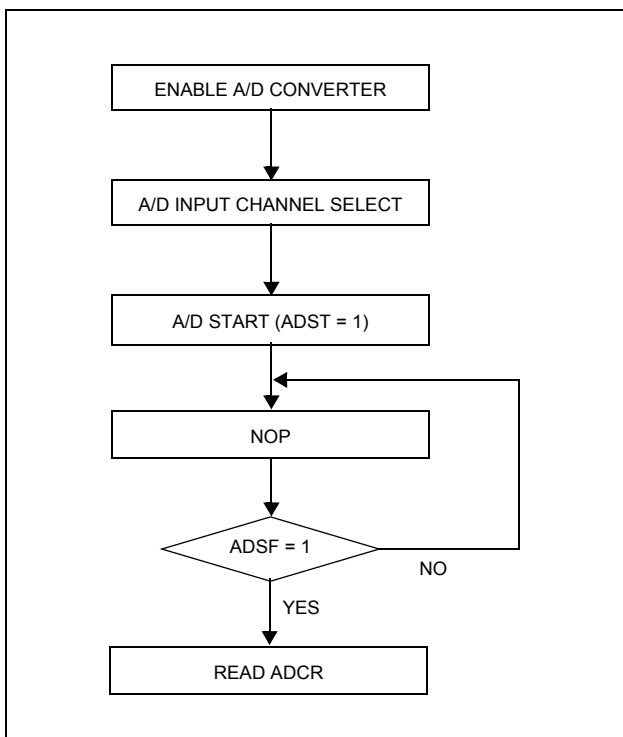


Figure 13-3 A/D Converter Operation Flow

A/D Converter Cautions

(1) Input range of AN0 to AN3

The input voltage of AN0 to AN3 should be within the specification range. In particular, if a voltage above V_{DD} or below V_{SS} is input (even if within the absolute maximum rating range), the conversion value for that channel can not be determinate. The conversion values of the other channels may also be affected.

(2) Noise countermeasures

In order to maintain 8-bit resolution, attention must be paid to noise on pins V_{DD} and AN0 to AN3. Since the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally as shown in Figure 13-4 in order to reduce noise.

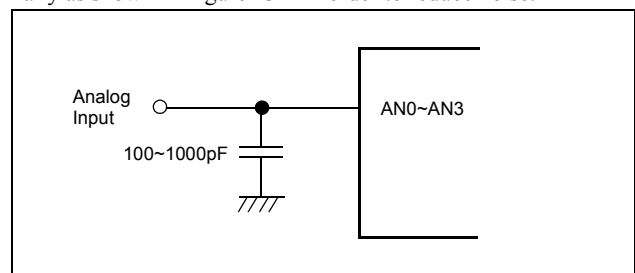


Figure 13-4 Analog Input Pin Connecting Capacitor

(3) Pins AN0~AN3

The analog input pins AN0 to AN3 also function as input/output port (PORT R0) pins. When A/D conversion is performed with any of pins AN0 to AN3 selected, be sure not to execute a PORT input instruction while conversion is in progress, as this may reduce the conversion resolution.

Also, if digital pulses are applied to a pin adjacent to the pin in the process of A/D conversion, the expected A/D conversion value may not be obtainable due to coupling noise. Therefore, avoid applying pulses to pins adjacent to

the pin undergoing A/D conversion.

(4) AV_{DD} pin input impedance

A series resistor string of approximately 10K Ω is connected between the V_{DD} pin and the V_{SS} pin.

Therefore, if the output impedance of the reference voltage source is high, this will result in parallel connection to the series resistor string between the V_{DD} pin and the V_{SS} pin, and there will be a large reference voltage error.

14. INTERRUPTS

The HMS87C1904 interrupt circuits consist of Interrupt enable register (IENH), Interrupt request flags of IRQH, Interrupt Edge Selection Register (IEDS), priority circuit and Master enable flag ("I" flag of PSW). The configuration of interrupt circuit is shown in Figure 14-1 and Interrupt priority is shown in Table 14-1.

The External Interrupts INT0 can each be transition-activated (1-to-0, 0-to-1 and both transition).

The flags that actually generate these interrupts are bit INT0IF in Register IRQH. When an external interrupt is generated, the flag that generated it is cleared by the hard-

ware when the service routine is vectored to only if the interrupt was transition-activated.

The Timer 0 and Timer 1 Interrupts are generated by T0IF and T1IF, which are set by a match in their respective timer/counter register. The AD converter Interrupt is generated by ADCIF which is set by finishing the analog to digital conversion. The Watch dog timer Interrupt is generated by WDTIF which set by a match in Watch dog timer register (when the bit WDTON is set to "0"). The Basic Interval Timer Interrupt is generated by BITIF which is set by a overflowing of the Basic Interval Timer Register (BITR).

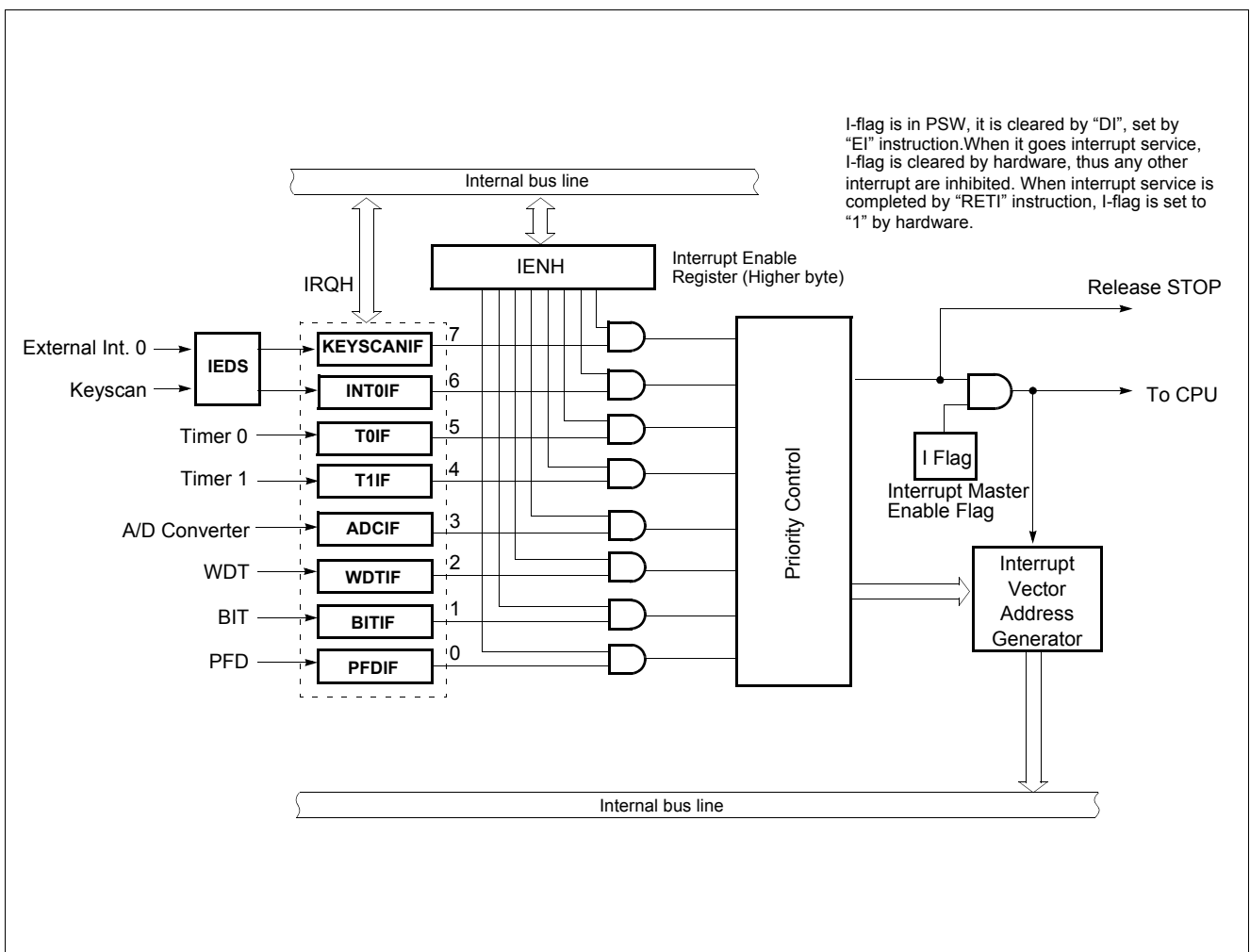


Figure 14-1 Block Diagram of Interrupt Function

The interrupts are controlled by the interrupt master enable flag I-flag (bit 2 of PSW), the interrupt enable register (IENH) and the interrupt request flags (in IRQH) except Power-on reset and software BRK interrupt.

Interrupt enable registers are shown in Figure 14-2. These registers are composed of interrupt enable flags of each interrupt source, these flags determines whether an interrupt will be accepted or not. When enable flag is “0”, a corresponding interrupt source is prohibited. Note that PSW contains also a master enable bit, I-flag, which disables all interrupts at once.

Reset/Interrupt	Symbol	Priority	Vector Addr.
Hardware Reset	RESET	-	FFFE _H
Keyscan Interrupt	KEYSCAN	1	FFFC _H
External Interrupt 0	INT0	2	FFFA _H
Timer 0	Timer 0	3	FFF8 _H
Timer 1	Timer 1	4	FFF6 _H
A/D Converter	A/D C	5	FFF4 _H
Watch Dog Timer	WDT	6	FFF2 _H
Basic Interval Timer	BIT	7	FFF0 _H
PFD	PFD	8	FFEE _H

Table 14-1 Interrupt Priority

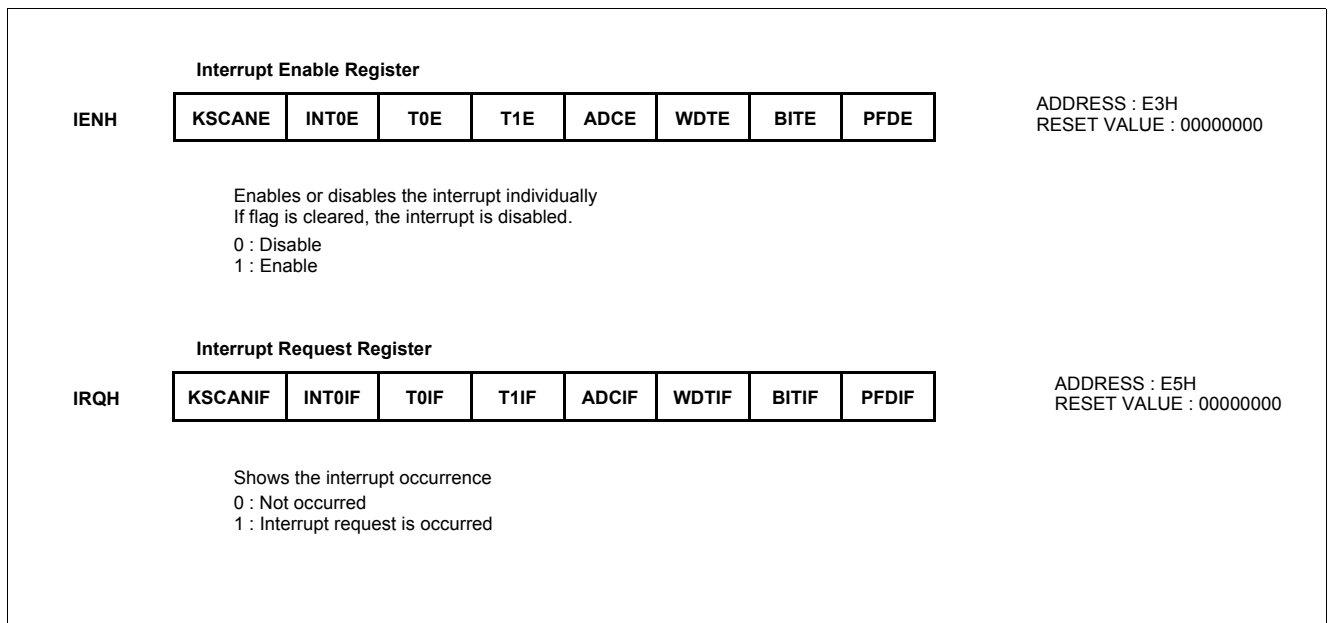


Figure 14-2 Interrupt Enable Registers and Interrupt Request Registers

When an interrupt is occurred, the I-flag is cleared and disable any further interrupt, the return address and PSW are pushed into the stack and the PC is vectored to. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt request flag bits.

The interrupt request flag bit(s) must be cleared by software before re-enabling interrupts to avoid recursive interrupts. The Interrupt Request flags are able to be read and written.

14.1 Interrupt Sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to “0” by a reset or an instruction. Interrupt acceptance sequence requires $8 f_{OSC}$ ($2 \mu s$ at $f_{XIN}=4MHz$) after the completion of the current instruction execution. The interrupt service task is terminated upon execution of an interrupt return instruction [RETI].

Interrupt acceptance

1. The interrupt master enable flag (I-flag) is cleared to “0” to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.

2. Interrupt request flag for the interrupt source accepted is cleared to “0”.
3. The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack area. The stack pointer decreases 3 times.
4. The entry address of the interrupt service program is read from the vector table address and the entry address is loaded to the program counter.
5. The instruction stored at the entry address of the interrupt service program is executed.

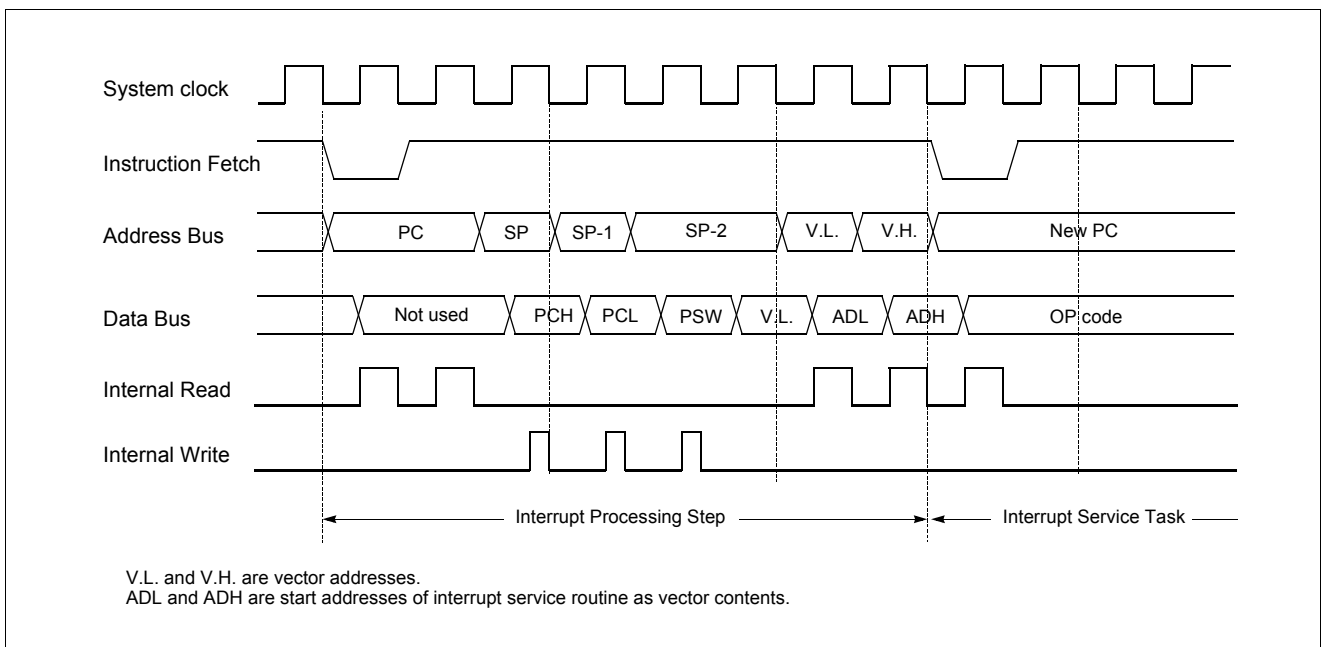
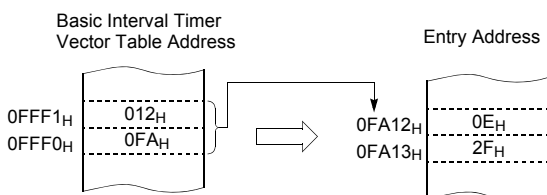


Figure 14-3 Timing chart of Interrupt Acceptance and Interrupt Return Instruction



Correspondence between vector table address for BIT interrupt and the entry address of the interrupt service program.

An interrupt request is not accepted until the I-flag is set to “1” even if a requested interrupt has higher priority than that of the current interrupt being serviced.

When nested interrupt service is required, the I-flag should be set to “1” by “EI” instruction in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags.

Saving/Restoring General-purpose Register

During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but accumulator and other registers are not saved itself. These registers are saved by the software if necessary. Also, when multiple interrupt services are nested, it is necessary to avoid using the same data memory area for saving registers.

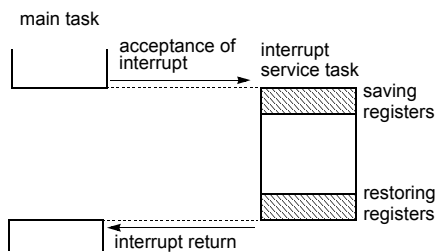
The following method is used to save/restore the general-purpose registers.

Example: Register save using push and pop instructions

```

INTxx:  PUSH    A      ;SAVE ACC.
        PUSH    X      ;SAVE X REG.
        PUSH    Y      ;SAVE Y REG.
        [interrupt processing]
        POP     Y      ;RESTORE Y REG.
        POP     X      ;RESTORE X REG.
        POP     A      ;RESTORE ACC.
        RETI         ;RETURN
    
```

General purpose register save/restore using push and pop instructions;



BRK Interrupt

Software interrupt can be invoked by BRK instruction, which has the lowest priority order.

Interrupt vector address of BRK is shared with the vector of TCALL 0 (Refer to Program Memory Section). When BRK interrupt is generated, B-flag of PSW is set to distinguish BRK from TCALL 0.

Each processing step is determined by B-flag as shown in Figure 14-4.

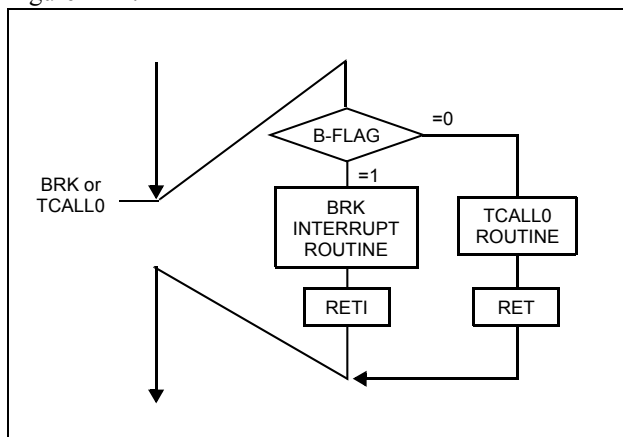


Figure 14-4 Execution of BRK/TCALL0

Multi Interrupt

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the interrupt are received at the same time simultaneously, an internal polling sequence determines by hardware which request is serviced.

However, multiple processing through software for special features is possible. Generally when an interrupt is accepted, the I-flag is cleared to disable any further interrupt. But as user sets I-flag in interrupt routine, some further interrupt can be serviced even if certain interrupt is in progress.

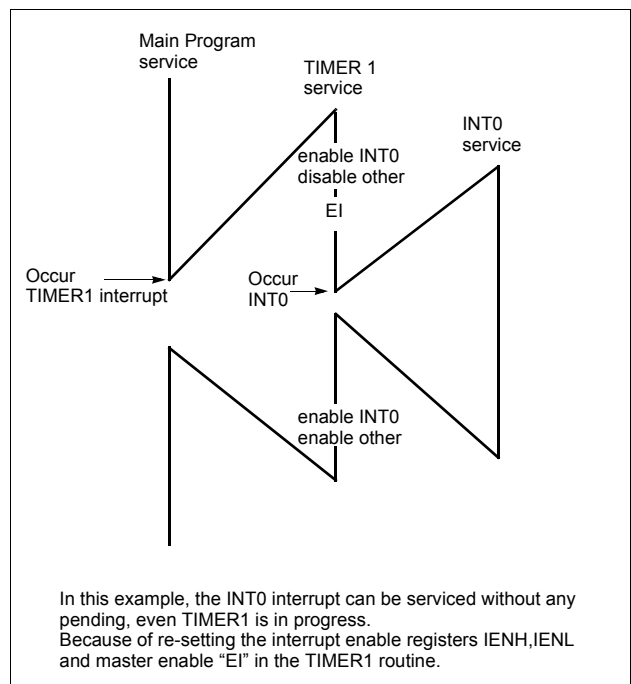


Figure 14-5 Execution of Multi Interrupt

Example: Even though Timer1 interrupt is in progress, INTO interrupt serviced without any suspend.

```

TIMER1:  PUSH    A
        PUSH    X
        PUSH    Y
        LDM    IENH, #40H ; Enable INT0 only
        NOP    ; Disable other
        EI    ; Enable Interrupt
        :
        :
        :
        :
        LDM    IENH, #0FFH ; Enable all interrupts
        POP     Y
        POP     X
        POP     A
        RETI
    
```

14.2 External Interrupt

The external interrupt on INT0 pin is edge triggered depending on the edge selection register IEDS (address 0E7H) as shown in Figure 14-6.

The edge detection of external interrupt has three transition activated mode: rising edge, falling edge, and both edge.

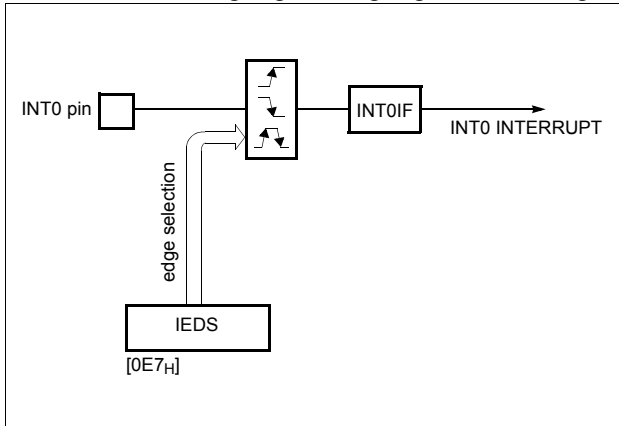


Figure 14-6 External Interrupt Block Diagram

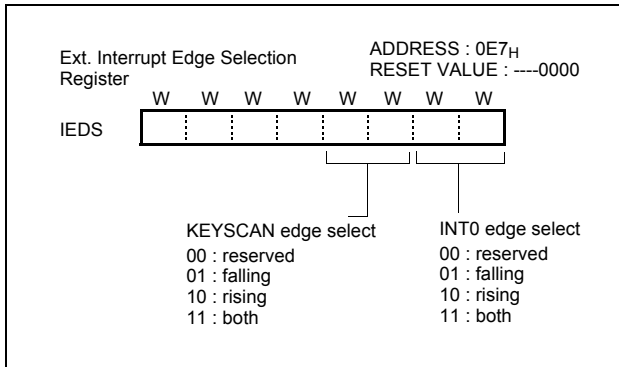


Figure 14-7 External Interrupt Edge Selection Register

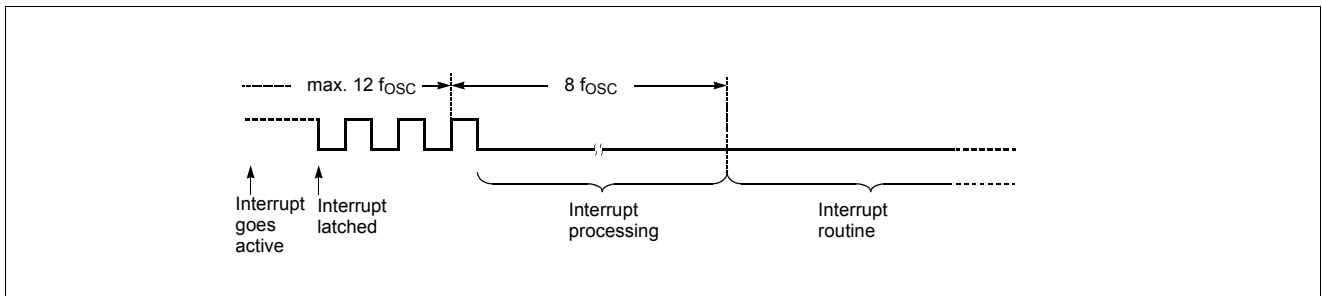


Figure 14-8 Interrupt Response Timing Diagram

Example: To use as an INT0

```

:
:
;**** Set port as an input port R02
LDM R0IO, #1111_1011B
;
;**** Set port as an interrupt port
LDM R0FUNC, #04H
;
;**** Set Falling-edge Detection
LDM IEDS, #0000_0001B
:
    
```

Response Time

The INT0 edge is latched into INT0IF at 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 DIV itself takes twelve cycles. Thus, a minimum of twelve complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine.

Figure 14-8 shows interrupt response timings.

15. WATCHDOG TIMER

The purpose of the watchdog timer is to detect the malfunction (runaway) of program due to external noise or other causes and return the operation to the normal condition. The watchdog timer has two types of clock source.

The first type is an on-chip RC-ring oscillator which does not require any external components. This RC-ring oscillator is separate from the external oscillator of the Xin pin or internal RC oscillator for system clock. It means that the watchdog timer can run even if the clock on the Xin pin of the device has been stopped, for example, by entering the STOP mode.

The internal RC-ring oscillator clock source be activated by setting the bit RCCLK of CKCTRL.

The other type is a prescaled system clock.

The watchdog timer consists of 7-bit binary counter and the watchdog timer data register. The source clock of WDT is overflow of Basic Interval Timer. When the value of 7-bit binary counter is equal to the lower 7 bits of WDTR, the interrupt request flag is generated. This can be used as WDT interrupt or CPU reset signal in accordance with the bit WDTON .

The 7-bit binary counter is cleared by setting WDTCL(bit7 of WDTR) and the WDTCL is cleared automatically after 1 machine cycle.

Note: Because the watchdog timer counter is enabled after clearing Basic Interval Timer and setting the Watchdog Timer Register, maximum error of timer is depend on prescaler ratio of Basic Interval Timer. To using the RESET by WDT, should be programmed the bit WDTE in Configuration Area set to "1" by OTP writer

The RC oscillation period is variable according to the temperature, V_{DD} and process variations from part to part (approximately, 60~200uS, typically 140uS at 5V). The following equation shows the RC oscillated watchdog timer time-out.

$$T_{RCWDT} = CLK_{RC} \times 2^8 \times [(WDTR.6 \sim 0) + 1]$$

where, CLK_{RC} = 60~200uS, typically 140uS

In addition, this watchdog timer can be used as a simple 7-bit timer by interrupt WDTIF. The interval of watchdog timer interrupt is decided by Basic Interval Timer. Interval equation is as below.

$$T_{WDT} = [WDTR.6 \sim 0] \times \text{Interval of BIT}$$

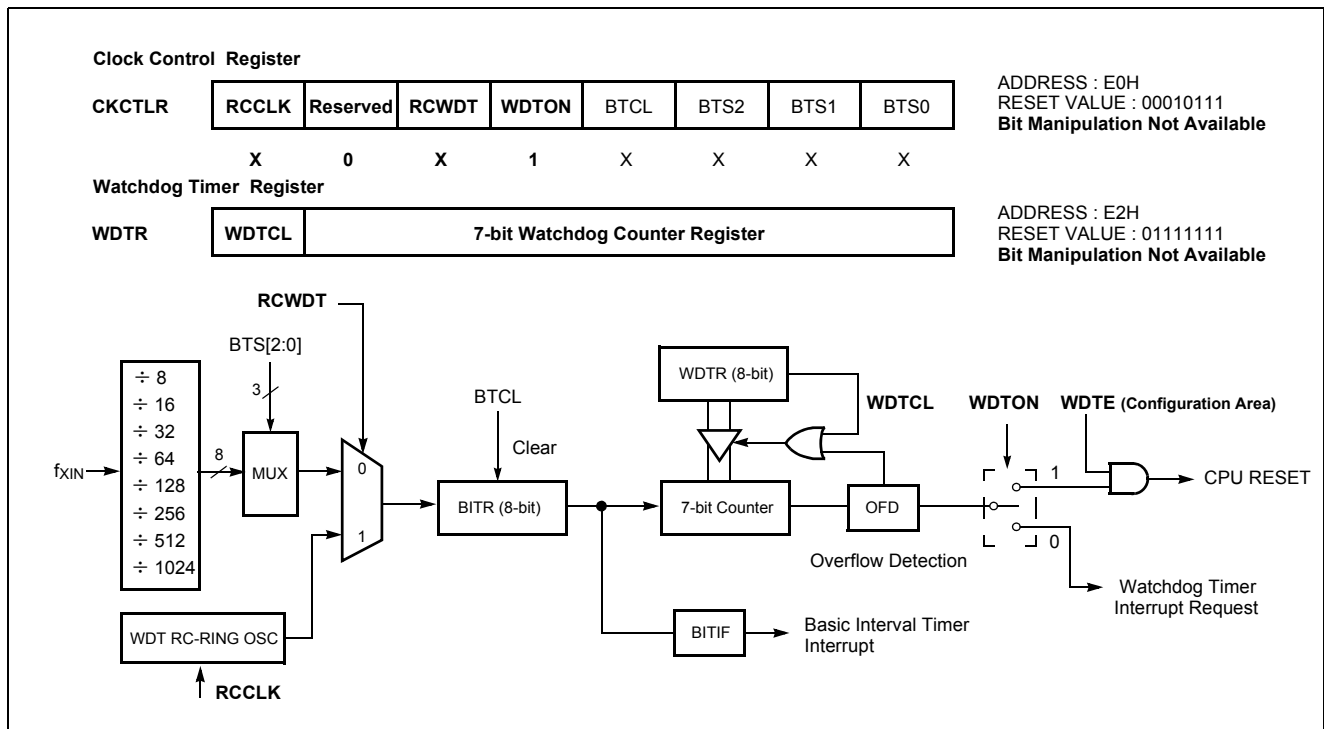


Figure 15-1 Block Diagram of Watchdog Timer

16. POWER SAVING MODE

For applications where power consumption is a critical factor, this device provides two kinds of power saving functions, STOP mode and SLEEP mode.

The STOP mode is activated by execution of STOP instruction after setting the corresponding value (5A_H) of SSCR. And the SLEEP mode is activated by setting the valut (0F_H) of SSCR

Note: Before executing STOP instruction, clear all interrupt request flag. Because if the interrupt request flag is set before STOP instruction, the MCU runs as if it doesn't perform STOP instruction, even though the STOP instruction is completed. So insert line to clear interrupt request flags (IRQH) before STOP instruction as shown each example.

Table 16-1 shows the status of each Power Saving Mode

Peripheral Operation	STOP Mode	SLEEP Mode
RAM	Retain	Retain
Control Registers	Retain	Retain
I/O Ports	Retain	Retain
CPU	Stop	Stop
Timer0	Stop Operates when the Event Counter Mode	Operation
Basic Interval Timer	Stop Operates when the RCWDT Mode	Operation
Watchdog Timer	Stop Operates when the RCWDT Mode	Operation
Oscillation	Stop	Oscillation
Prescaler	Stop & Retain	Active
Entering Condition	Setting 5A _H to SSCR, then STOP	Setting only 0F _H to SSCR
Power Saving Release Source	RESET, INT0, Timer0(EC0), KEYSCAN POR, RCWDT, On-Pin Change	RESET, All Interrupt Sources

Table 16-1 Power Saving Mode

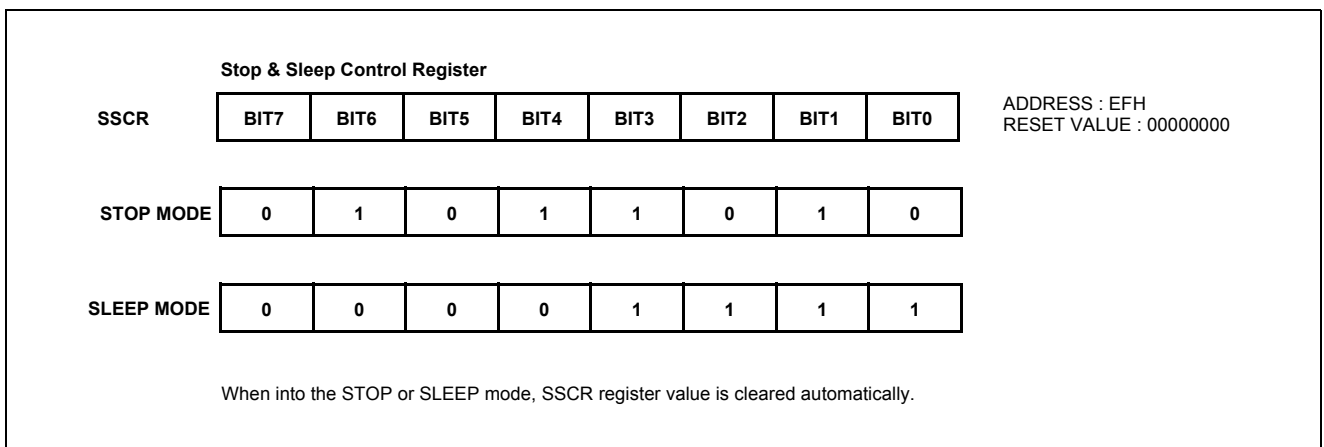


Figure 16-1 Stop & Sleep Control Register

16.1 Stop Mode

In the Stop mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip

RAM and Control registers are held. The port pins out the values held by their respective port data register, port di-

rection registers. Oscillator stops and the systems internal operations are all held up.

- The states of the RAM, registers, and latches valid immediately before the system is put in the STOP state are all held.
- The program counter stop the address of the instruction to be executed after the instruction

“STOP” which starts the STOP operating mode.

The Stop mode is activated by execution of STOP instruction after setting the value SSCR(Stop & Sleep Control Register) to “5AH”. (This register value is cleared automatically)

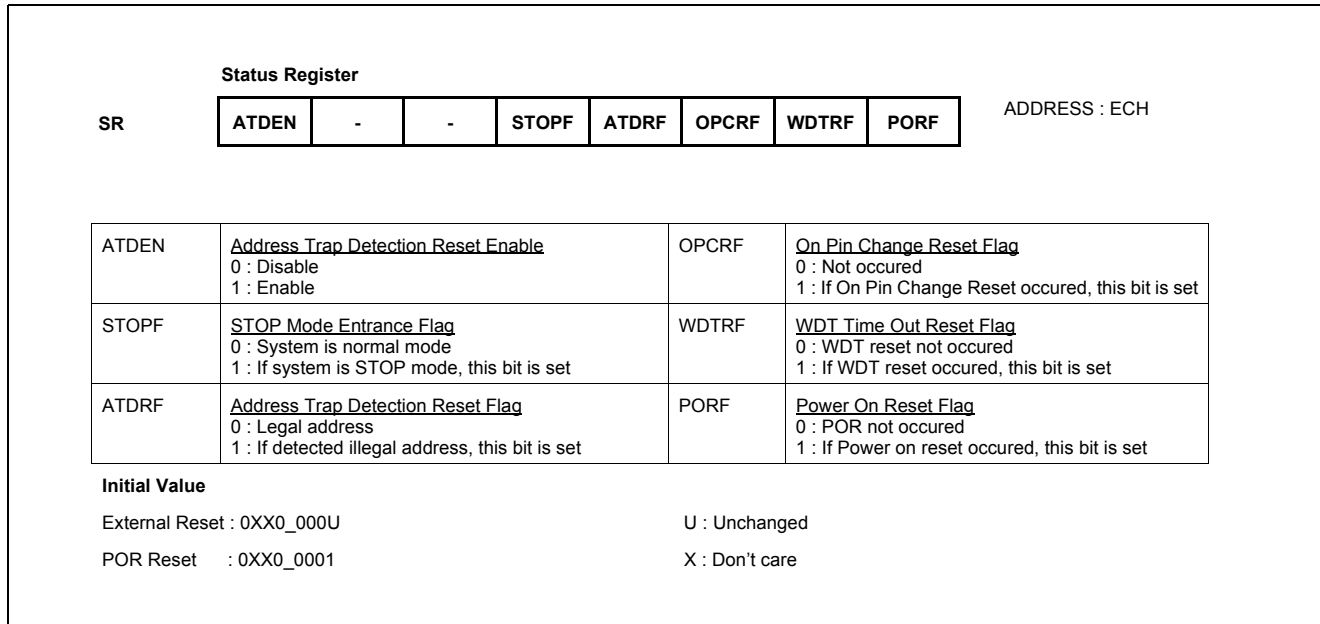


Figure 16-2 Status Register

In the Stop mode of operation, V_{DD} can be reduced to minimize power consumption. Care must be taken, however, to ensure that V_{DD} is not reduced before the Stop mode is invoked, and that V_{DD} is restored to its normal operating level, before the Stop mode is terminated.

The reset should not be activated before V_{DD} is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize.

Note: After STOP instruction, at least two or more NOP instruction should be written

```

Ex)  LDM SSCR, #5AH
      LDM IRQH, #0
      STOP
      NOP
      NOP
    
```

Note: If selected HF mode, this STOP mode can not be used.

In the STOP operation, the dissipation of the power associated with the oscillator and the internal hardware is low-

ered; however, the power dissipation associated with the pin interface (depending on the external circuitry and program) is not directly determined by the hardware operation of the STOP feature. This point should be little current flows when the input level is stable at the power voltage level (V_{DD}/V_{SS}), however, when the input level gets higher than the power voltage level (by approximately 0.3 to 0.5V), a current begins to flow. Therefore, if cutting off the output transistor at an I/O port puts the pin signal into the high-impedance state, a current flow across the ports input transistor, requiring to fix the level by pull-up or other means.

Release the STOP mode

The exit from STOP mode is hardware reset or external interrupt including Keyscan and On-pin change. Reset re-defines all the Control registers but does not change the on-chip RAM. External interrupts allow both on-chip RAM and Control registers to retain their values. But On-pin change causes system RESET.

After releasing STOP mode, instruction execution is divided into two ways by I-flag(bit2 of PSW). If I-flag = 1, the normal interrupt response takes place. If I-

flag = 0, the chip will resume execution starting with the instruction following the STOP instruction. It will not vector to interrupt service routine. (refer to Figure 16-3)

When exit from Stop mode by external interrupt, enough oscillation stabilization time is required to normal operation. Figure 16-4 shows the timing diagram. When release the Stop mode, the Basic interval timer is activated on wake-up. It is increased from 00_H until FF_H. The count overflow is set to start normal operation. Therefore, before STOP instruction, user must be set its relevant prescaler divide ratio to have long enough time (more than 20msec). This guarantees that oscillator has started and stabilized.

By reset, exit from Stop mode is shown in Figure 16-5.

Minimizing Current Consumption in Stop Mode

The Stop mode is designed to reduce power consumption. To minimize the current consumption during Stop mode, the user should turn-off output drivers that are sourcing or sinking current, if it is practical. Weak pull-ups on port pins should be turned off, if possible. All inputs should be either as V_{SS} or at V_{DD} (or as close to rail as possible).

An intermediate voltage on an input pin causes the input buffer to draw a significant amount of current.

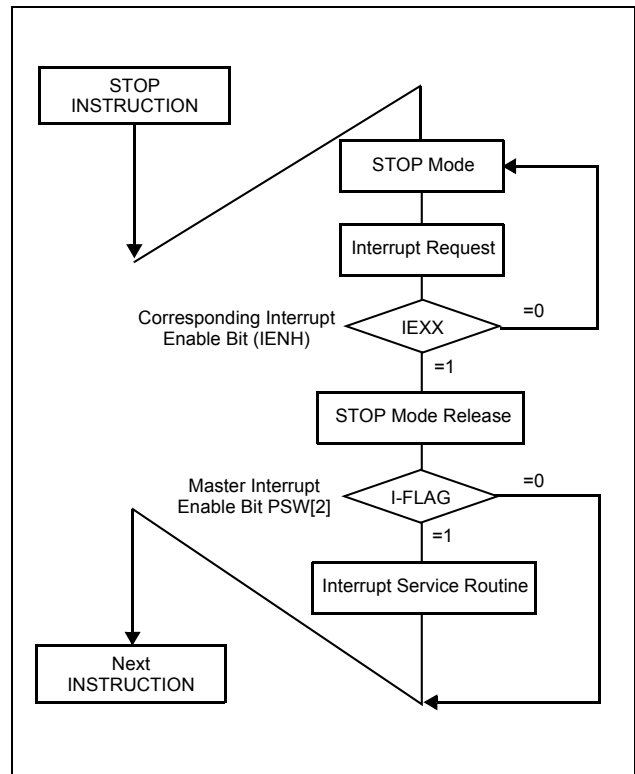


Figure 16-3 STOP Releasing Flow by Interrupts

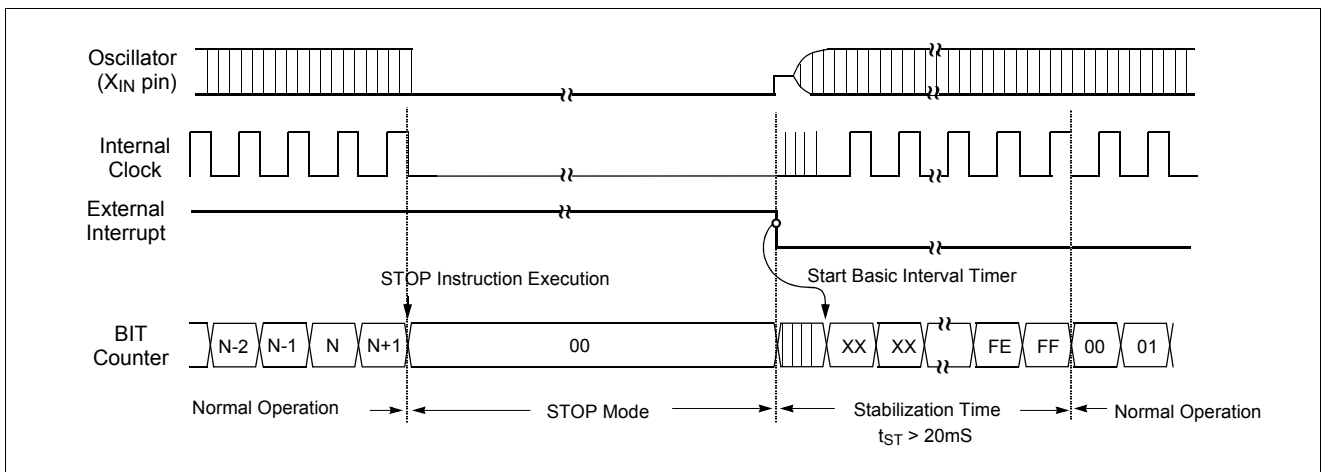


Figure 16-4 Timing of STOP Mode Release by External Interrupt(XT, LF Osc Mode)

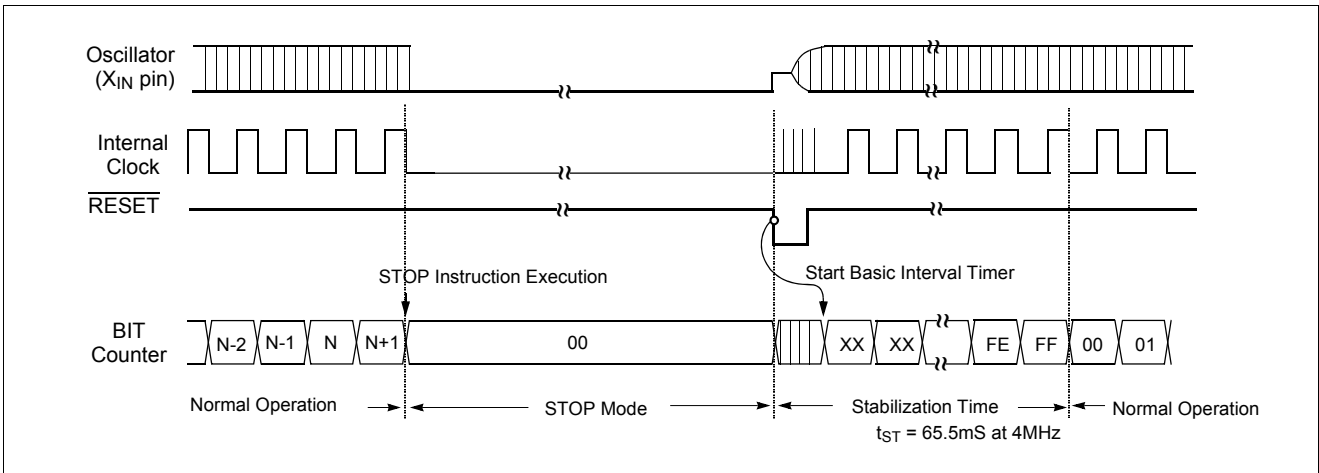


Figure 16-5 Timing of STOP Mode Release by RESET(XT, LF Osc Mode)

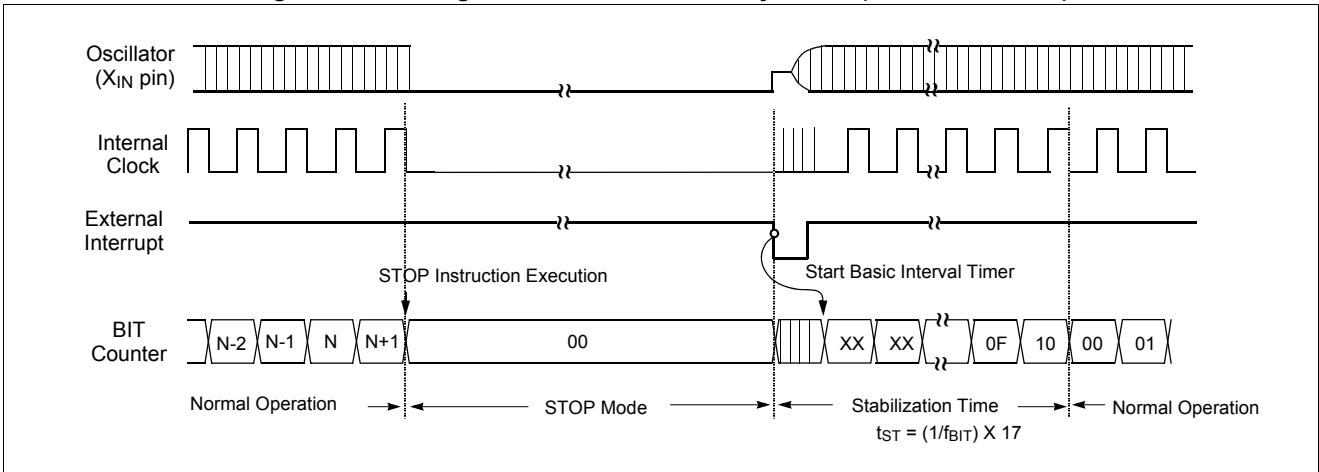


Figure 16-6 Timing of STOP Mode Release by RESET or External Interrupt(ERC, IRC Mode)

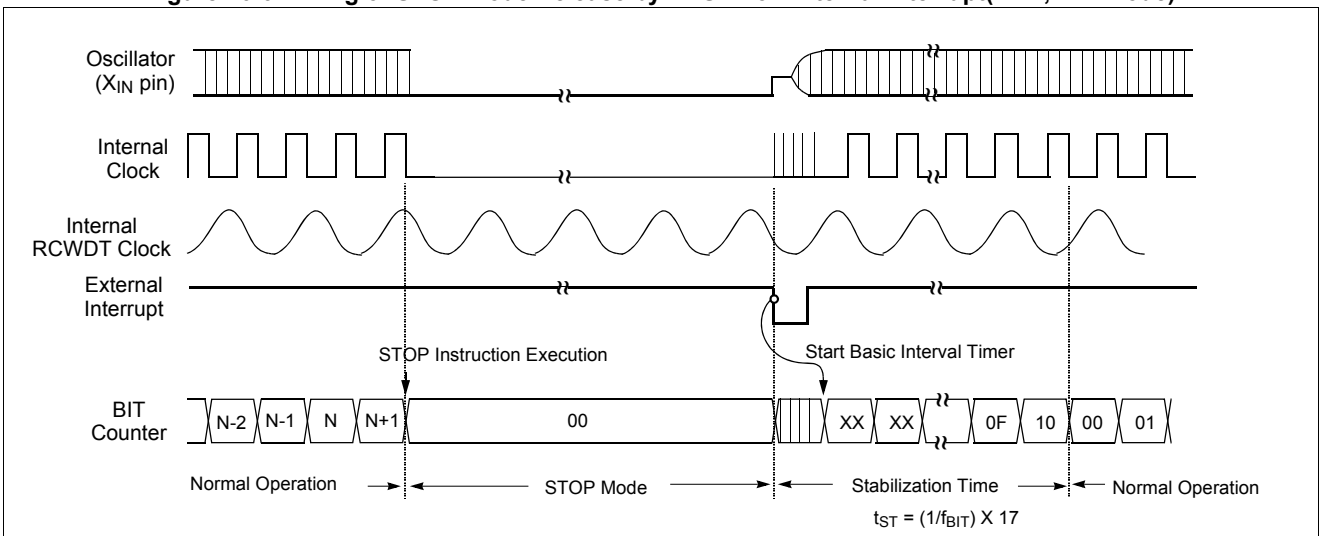


Figure 16-7 Timing of STOP Mode Release by RESET or External Interrupt(ERC, IRC Mode) using RCWDT

16.2 SLEEP Mode

In the SLEEP mode, the CPU operation is stopped. Except the peripherals (Timer0, Basic Interval Timer, and Watchdog Timer), all functions are stopped, but the on-chip RAM and Control registers are held. The port pins out the values held by their respective port data register, port direction registers.

The SLEEP mode is activated by setting the value SSCR(Stop & Sleep Control Register) to “0FH”. (This register value is cleared automatically)

Note: After setting SSCR, at least two or more NOP instruction should be written

```

Ex)  LDM  IRQH, #0
      LDM  SSCR, #0FH
      NOP
      NOP
    
```

Release the SLEEP mode

The exit from SLEEP mode is hardware reset or all interrupt sources including Keyscan and On-pin change. The Reset initializes all the Control registers but does not change the on-chip RAM. Interrupts allow both on-chip RAM and Control registers to retain their values. But On-pin change causes system RESET.

If I-flag = 1, the normal interrupt response takes place. If I-flag = 0, the chip will resume execution starting with the instruction following the STOP instruction. It will not vector to interrupt service routine (refer to Figure 16-3).

When exit from SLEEP mode by external interrupt or timer0 overflow, the oscillation stabilization time is not required to normal operation. Because this mode do not stop the on-chip oscillator shown as Figure 16-8.

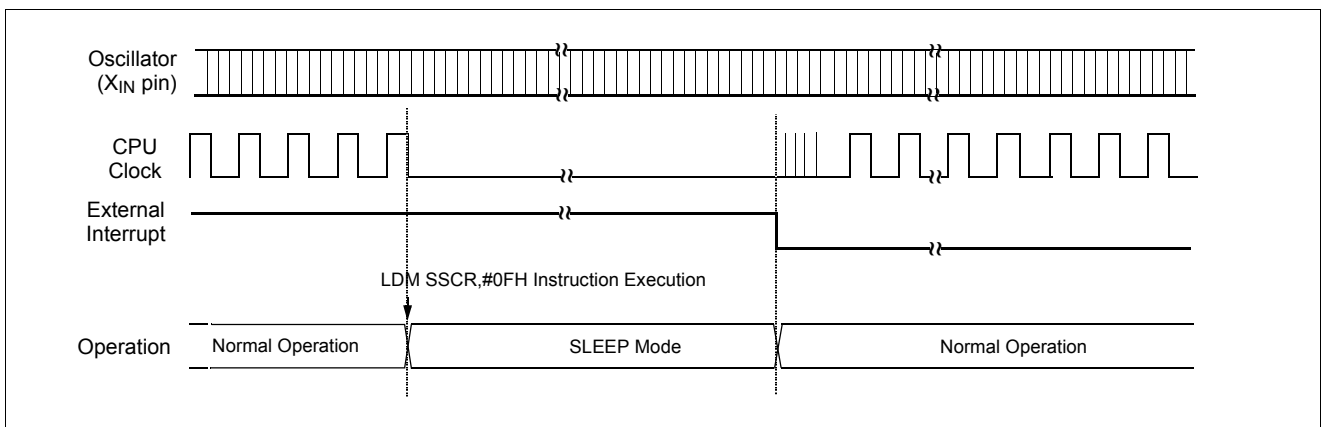


Figure 16-8 SLEEP Mode Releasing by External Interrupt

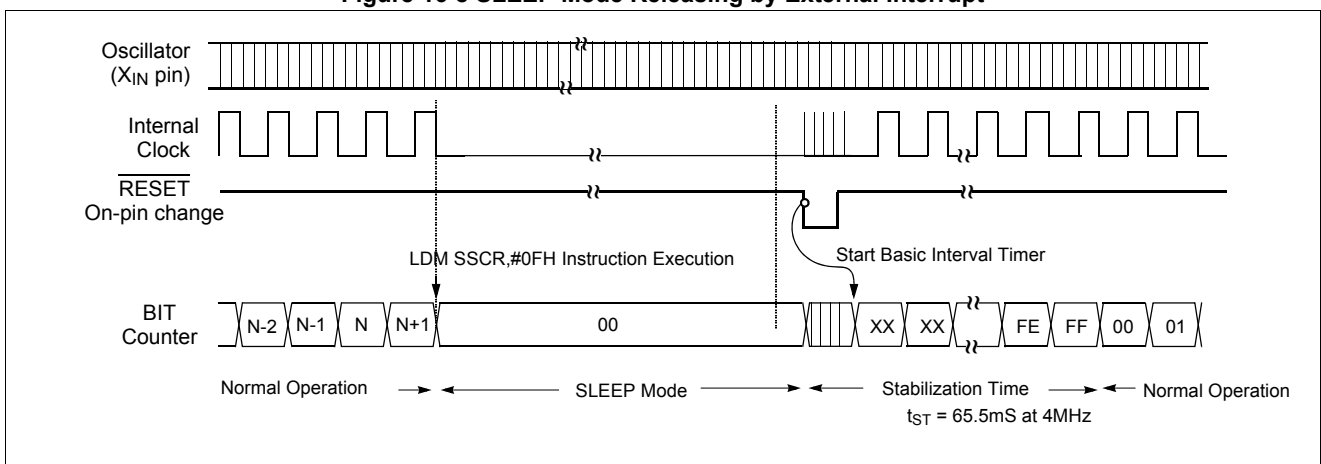


Figure 16-9 SLEEP Mode Releasing by RESET(On-pin change)

16.3 Minimizing Current Consumption

The Stop mode is designed to reduce power consumption. To minimize current drawn during Stop mode, the user should turn-off output drivers that are sourcing or sinking current, if it is practical.

Note: *In the STOP operation, the power dissipation associated with the oscillator and the internal hardware is lowered; however, the power dissipation associated with the pin interface (depending on the external circuitry and program) is not directly determined by the hardware operation of the STOP feature. This point should be little current flows when the input level is stable at the power voltage level (V_{DD}/V_{SS}); however, when the input level becomes higher than the power voltage level (by approximately 0.3V), a current begins to flow. Therefore, if cutting off the output transistor at an I/O port puts the pin signal into the high-impedance state, a current flow across the ports input transistor, requiring it to fix the level by pull-up or other means.*

It should be set properly that current flow through port doesn't exist.

First consider the setting to input mode. Be sure that there is no current flow after considering its relationship with external circuit. In input mode, the pin impedance viewing from external MCU is very high that the current doesn't flow.

But input voltage level should be V_{SS} or V_{DD} . Be careful that if unspecified voltage, i.e. if uncertain voltage level (not V_{SS} or V_{DD}) is applied to input pin, there can be little current (max. 1mA at around 2V) flow.

If it is not appropriate to set as an input mode, then set to output mode considering there is no current flow. Setting to High or Low is decided considering its relationship with external circuit. For example, if there is external pull-up resistor then it is set to output mode, i.e. to High, and if there is external pull-down register, it is set to low.

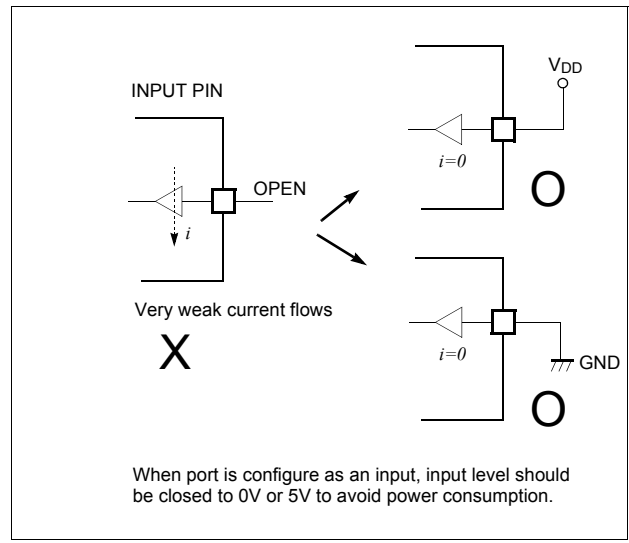
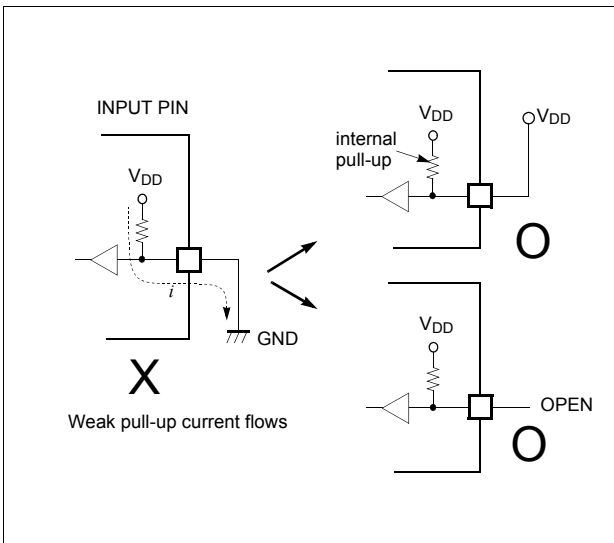


Figure 16-10 Application Example of Unused Input Port

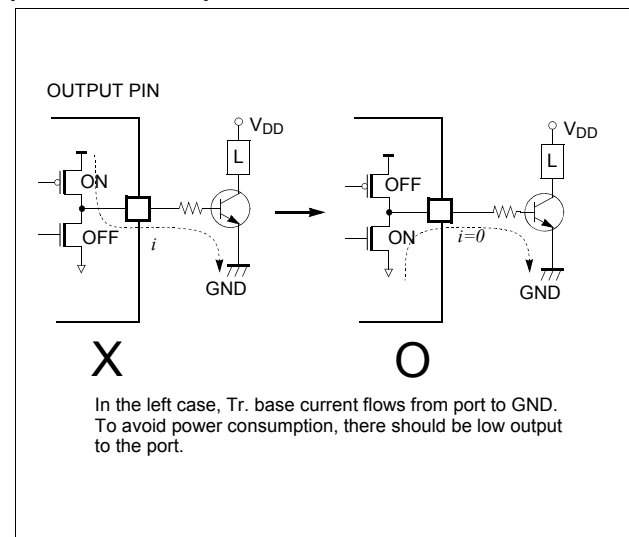
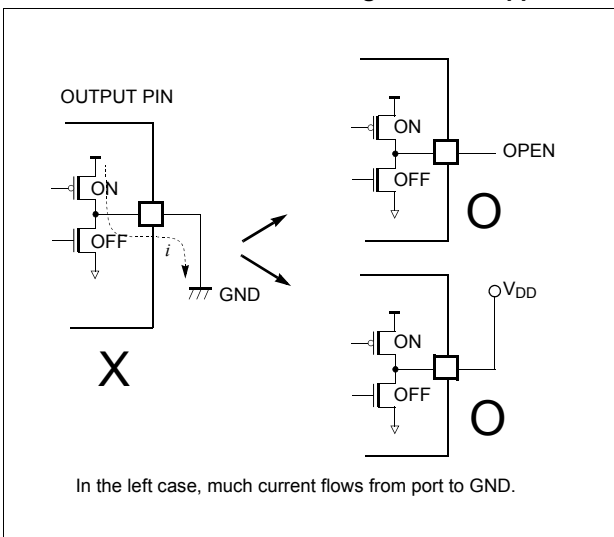


Figure 16-11 Application Example of Unused Output Port

17. RESET

The HMS87C1904 supports various kinds of reset as below.

- Power-On Reset(POR)
- Watchdog Timer Timeout Reset
- On Pin Change Reset
- Power-Fail Detection(PFD) Reset
- On Pin Change Reset during Power Down Mode
- $\overline{\text{RESET}}$ (external reset circuitry)

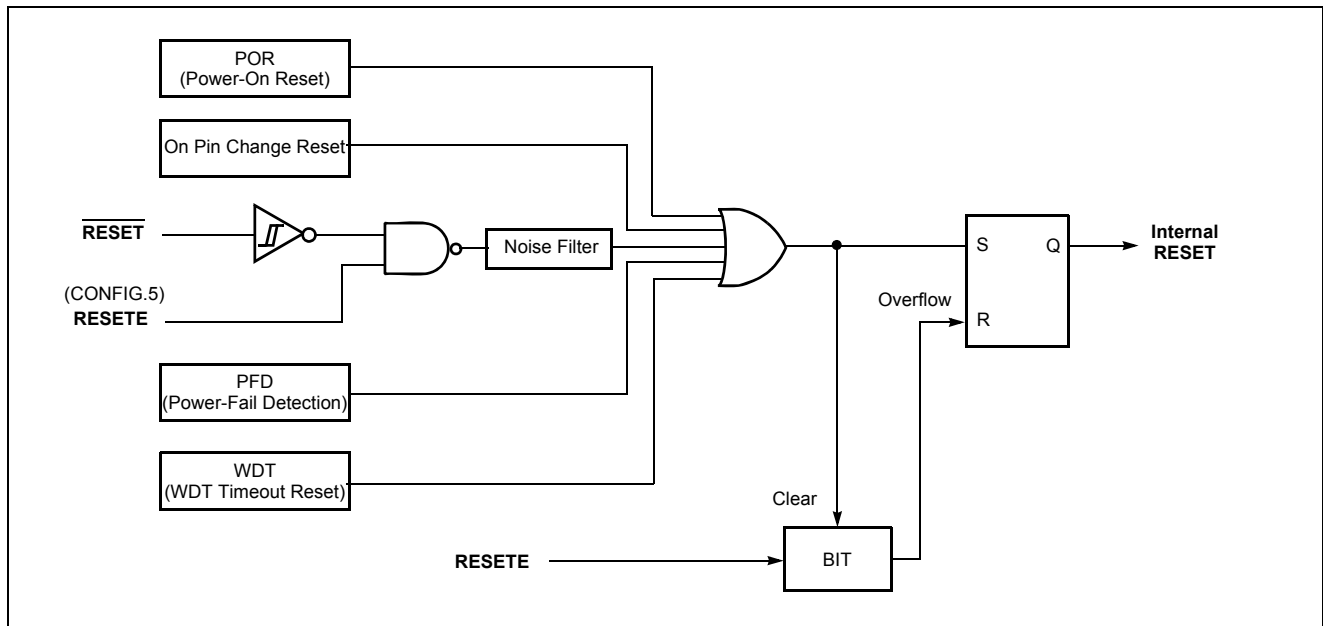


Figure 17-1 RESET Block Diagram

The on-chip POR circuit holds down the device in RESET until V_{DD} has reached a high enough level for proper operation. In this case, just tie the $\overline{\text{RESET}}$ pin to V_{DD} or using the normal input port R03 by clearing the bit “RESETE” of Configuration Area(201FH) in the OTP pro-

gramming. This will eliminate external components.

When the device starts normal operation, its operating parameters(voltage, frequency, temperature...etc) must be met.

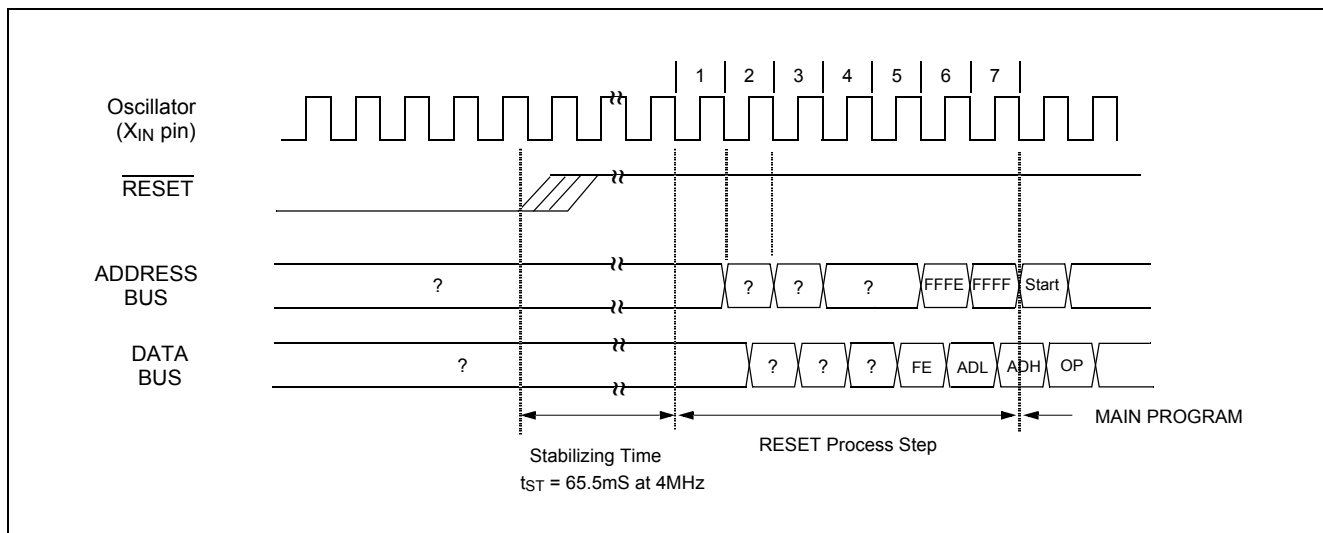


Figure 17-2 Timing Diagram after RESET

The reset input is the $\overline{\text{RESET}}$ pin, which is the input to a Schmitt Trigger. A reset is accomplished by holding the $\overline{\text{RESET}}$ pin low for at least 8 oscillator periods, while the oscillator running. After reset, 65.5ms (at 4 MHz) add with 7 oscillator periods are required to start execution as shown in Figure 17-2.

Internal RAM is not affected by reset. When V_{DD} is turned on, the RAM content is indeterminate. Therefore, this RAM should be initialized before reading or testing it.

Initial state of each register is shown as Table 8-1.

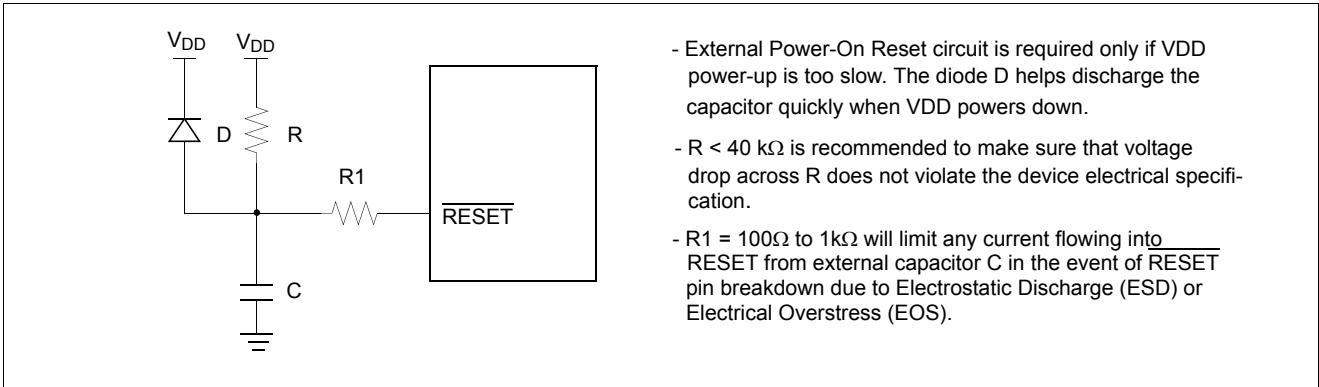


Figure 17-3 External Power-on RESET Circuit (For Slow V_{DD} Power-up)

18. POWER FAIL DETECTOR

The HMS87C1904 has an on-chip power fail detection circuitry to immunize against power noise. A Power Fail Detector Register, PFDR can enable or disable the Power fail Detect circuitry. And it can also select the detection level, the using noise filter and the result of power fail situation as system RESET or System Clock Freeze.

The PFDE bit enables the system voltage detector and the bit PFDLVL bit selects the detection level high or low. Detection high level is 3.2V and low level is 2.4V typically. This parameter is variable according to the V_{DD}, temperatur and process variations from part to part.

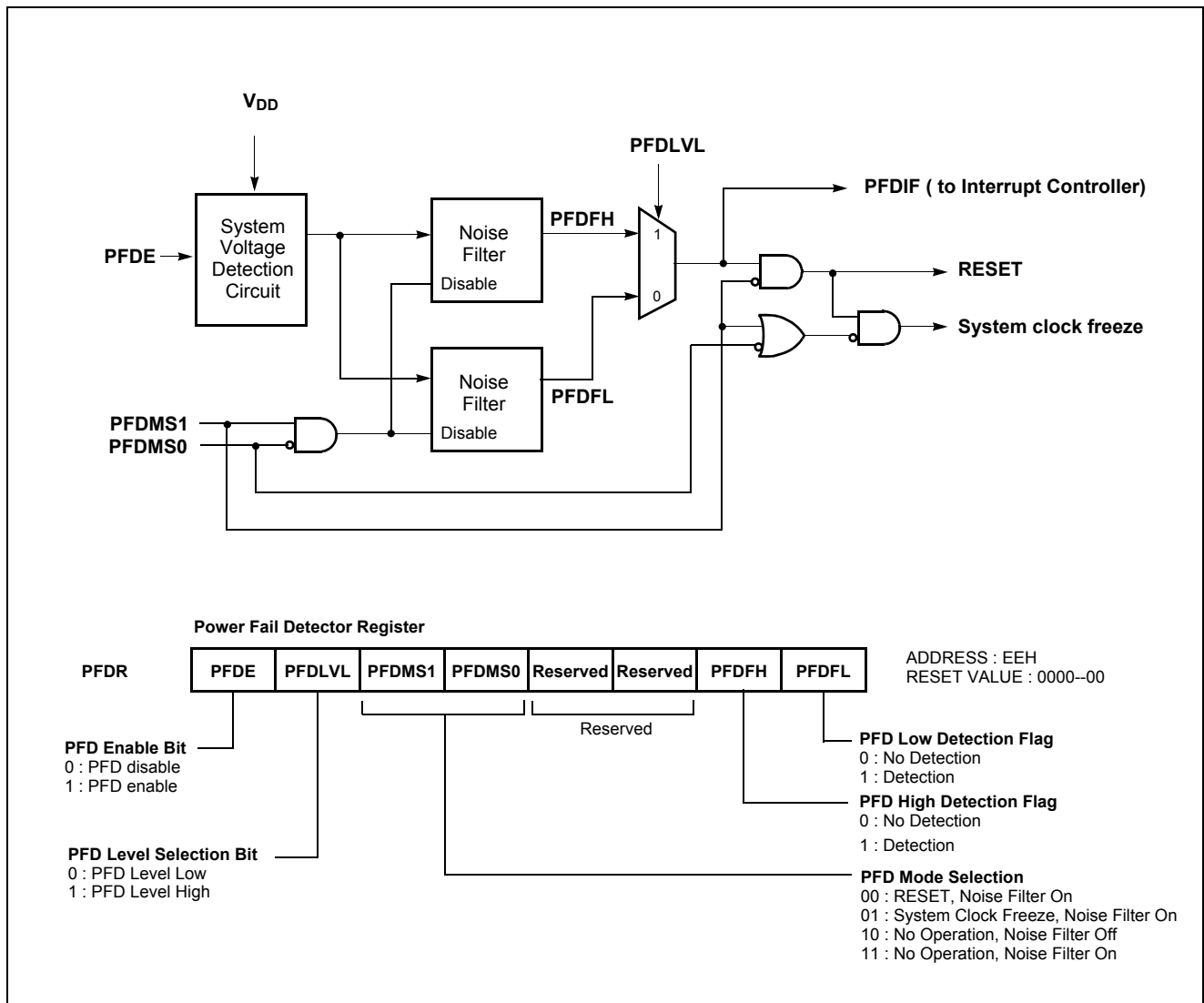


Figure 18-1 Power Fail Detector

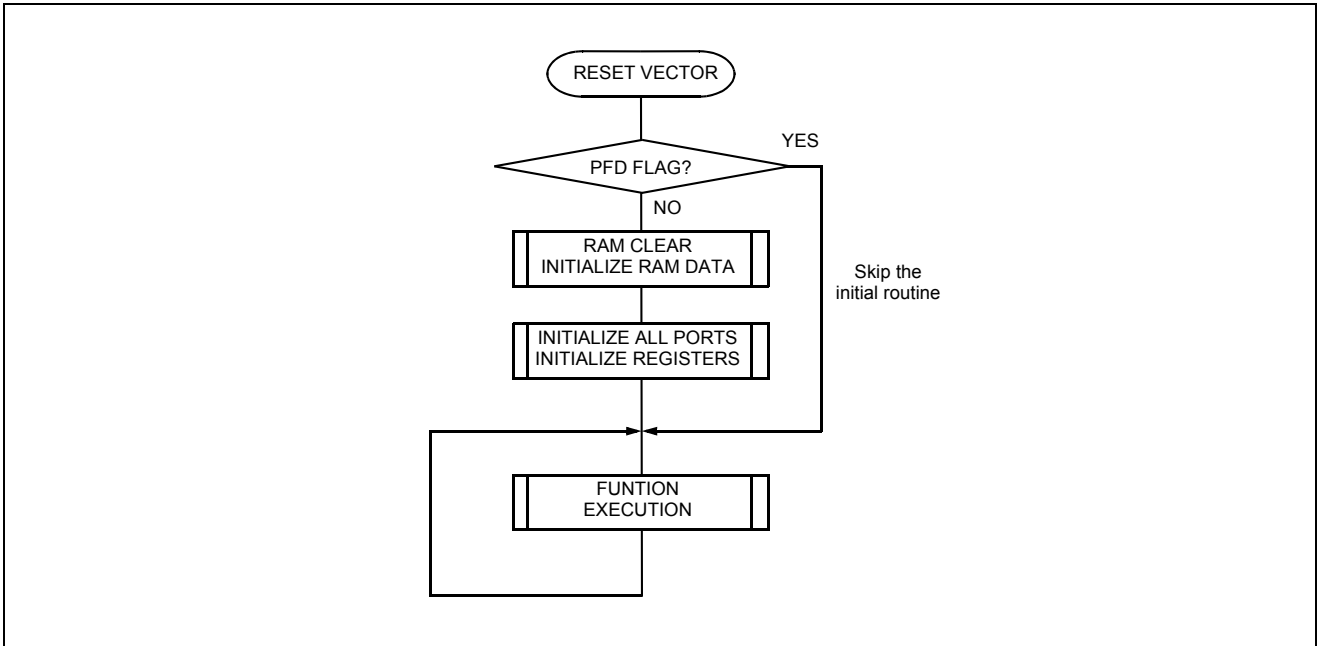


Figure 18-2 Example S/W of RESET by Power fail

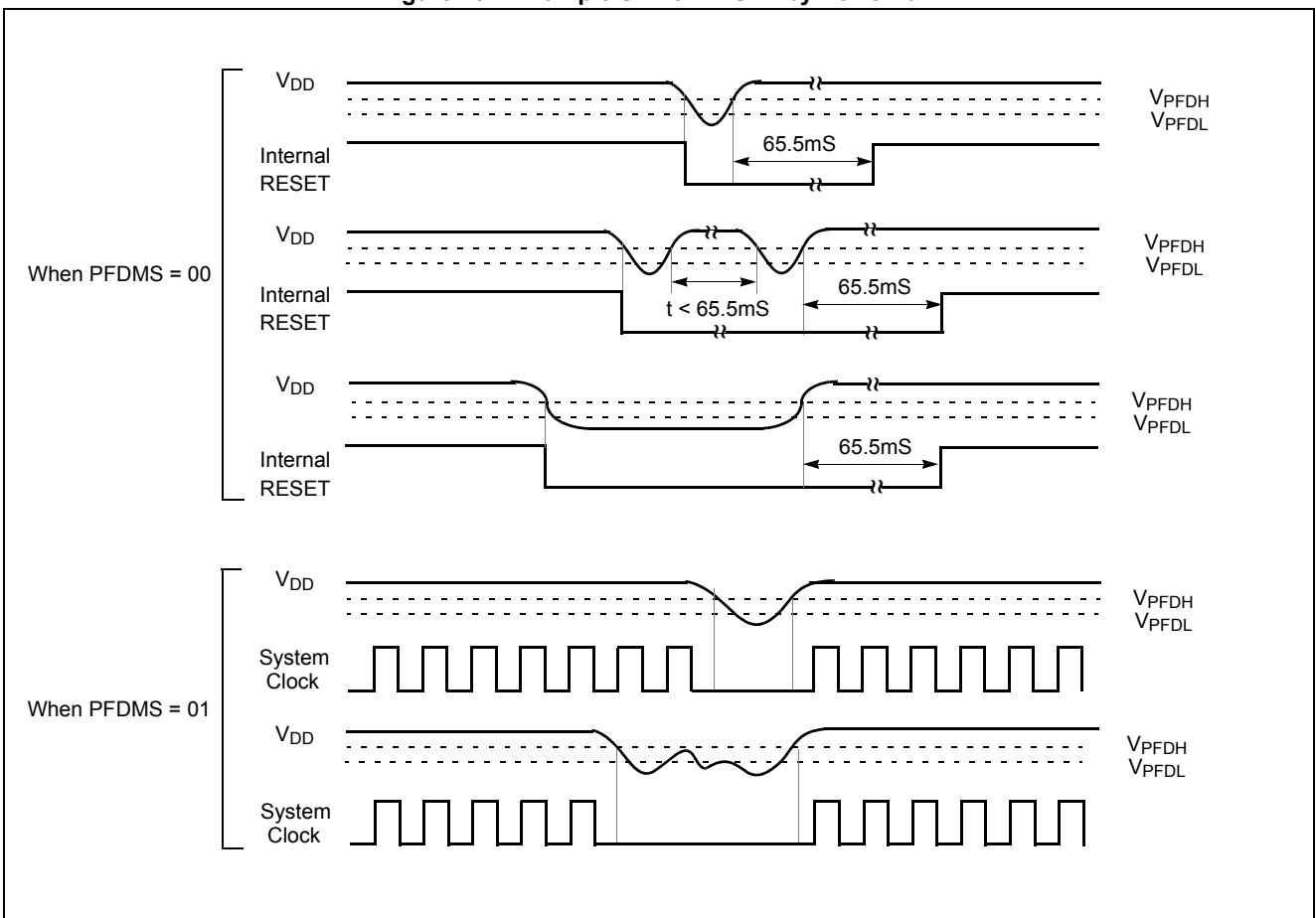


Figure 18-3 Power Fail Processor Situations (at 4MHz)

APPENDIX

1. APPENDIX

1.1 Instruction Map

LOW HIGH	0000 00	00001 01	00010 02	00011 03	00100 04	00101 05	00110 06	00111 07	01000 08	01001 09	01010 0A	01011 0B	01100 0C	01101 0D	01110 0E	01111 0F
000	NOP	SET1 dp.bit	BBS A.bit,rel	BBS dp.bit,rel	ADC #imm	ADC dp	ADC dp+X	ADC !abs	ASL A	ASL dp	TCALL 0	SETA1 .bit	BIT dp	POP A	PUSH A	BRK
001	CLRC	//	//	//	SBC #imm	SBC dp	SBC dp+X	SBC !abs	ROL A	ROL dp	TCALL 2	CLRA1 .bit	COM dp	POP X	PUSH X	BRA rel
010	CLRG	//	//	//	CMP #imm	CMP dp	CMP dp+X	CMP !abs	LSR A	LSR dp	TCALL 4	NOT1 M.bit	TST dp	POP Y	PUSH Y	PCALL Upage
011	DI	//	//	//	OR #imm	OR dp	OR dp+X	OR !abs	ROR A	ROR dp	TCALL 6	OR1 OR1B	CMPX dp	POP PSW	PUSH PSW	RET
100	CLRV	//	//	//	AND #imm	AND dp	AND dp+X	AND !abs	INC A	INC dp	TCALL 8	AND1 AND1B	CMPY dp	CBNE dp+X	TXSP	INC X
101	SETC	//	//	//	EOR #imm	EOR dp	EOR dp+X	EOR !abs	DEC A	DEC dp	TCALL 10	EOR1 EOR1B	DBNE dp	XMA dp+X	TSPX	DEC X
110	SETG	//	//	//	LDA #imm	LDA dp	LDA dp+X	LDA !abs	TXA	LDY dp	TCALL 12	LDC LDCB	LDX dp	LDX dp+Y	XCN	DAS
111	EI	//	//	//	LDM dp,#imm	STA dp	STA dp+X	STA !abs	TAX	STY dp	TCALL 14	STC M.bit	STX dp	STX dp+Y	XAS	STOP

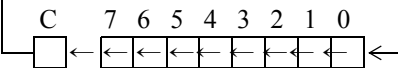
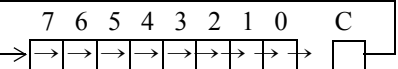
LOW HIGH	10000 10	10001 11	10010 12	10011 13	10100 14	10101 15	10110 16	10111 17	11000 18	11001 19	11010 1A	11011 1B	11100 1C	11101 1D	11110 1E	11111 1F
000	BPL rel	CLR1 dp.bit	BBC A.bit,rel	BBC dp.bit,rel	ADC {X}	ADC !abs+Y	ADC [dp+X]	ADC [dp]+Y	ASL !abs	ASL dp+X	TCALL 1	JMP !abs	BIT !abs	ADDW dp	LDX #imm	JMP [!abs]
001	BVC rel	//	//	//	SBC {X}	SBC !abs+Y	SBC [dp+X]	SBC [dp]+Y	ROL !abs	ROL dp+X	TCALL 3	CALL !abs	TEST !abs	SUBW dp	LDY #imm	JMP [dp]
010	BCC rel	//	//	//	CMP {X}	CMP !abs+Y	CMP [dp+X]	CMP [dp]+Y	LSR !abs	LSR dp+X	TCALL 5	MUL	TCLR1 !abs	CMPW dp	CMPX #imm	CALL [dp]
011	BNE rel	//	//	//	OR {X}	OR !abs+Y	OR [dp+X]	OR [dp]+Y	ROR !abs	ROR dp+X	TCALL 7	DBNE Y	CMPX !abs	LDYA dp	CMPY #imm	RETI
100	BMI rel	//	//	//	AND {X}	AND !abs+Y	AND [dp+X]	AND [dp]+Y	INC !abs	INC dp+X	TCALL 9	DIV	CMPY !abs	INCW dp	INC Y	TAY
101	BVS rel	//	//	//	EOR {X}	EOR !abs+Y	EOR [dp+X]	EOR [dp]+Y	DEC !abs	DEC dp+X	TCALL 11	XMA {X}	XMA dp	DECW dp	DEC Y	TYA
110	BCS rel	//	//	//	LDA {X}	LDA !abs+Y	LDA [dp+X]	LDA [dp]+Y	LDY !abs	LDY dp+X	TCALL 13	LDA {X}+	LDX !abs	STYA dp	XAY	DAA
111	BEQ rel	//	//	//	STA {X}	STA !abs+Y	STA [dp+X]	STA [dp]+Y	STY !abs	STY dp+X	TCALL 15	STA {X}+	STX !abs	CBNE dp	XYX	NOP

1.2 Alphabetic order table of instruction

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADC #imm	04	2	2	Add with carry.	NV -- H - ZC
2	ADC dp	05	2	3	$A \leftarrow A + (M) + C$	
3	ADC dp + X	06	2	4		
4	ADC !abs	07	3	4		
5	ADC !abs+Y	15	3	5		
6	ADC [dp+X]	16	2	6		
7	ADC [dp]+Y	17	2	6		
8	ADC {X}	14	1	3		
9	ADDW dp	1D	2	5	16-bits add without carry : $YA \leftarrow YA + (dp+1)(dp)$	NV -- H - ZC
10	AND #imm	84	2	2	Logical AND	N ----- Z -
11	AND dp	85	2	3	$A \leftarrow A \wedge (M)$	
12	AND dp + X	86	2	4		
13	AND !abs	87	3	4		
14	AND !abs+Y	95	3	5		
15	AND [dp+X]	96	2	6		
16	AND [dp] + Y	97	2	6		
17	AND {X}	94	1	3		
18	AND1 M.bit	8B	3	4	Bit AND C-flag : $C \leftarrow C \wedge (M.bit)$	----- C
19	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : $C \leftarrow C \wedge (M.bit)$	----- C
20	ASL A	08	1	2	Arithmetic shift left	N ----- ZC
21	ASL dp	09	2	4	$ \begin{array}{cccccccc} & C & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \square & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \leftarrow "0" \end{array} $	
22	ASL dp + X	19	2	5		
23	ASL !abs	18	3	5		
24	BBC A.bit,rel	y2	2	4/6		
25	BBC dp.bit,rel	y3	3	5/7	if(bit) = 0, then $PC \leftarrow PC + rel$	-----
26	BBS A.bit,rel	x2	2	4/6	Branch if bit clear :	-----
27	BBS dp.bit,rel	x3	3	5/7	if(bit) = 1, then $PC \leftarrow PC + rel$	
28	BCC rel	50	2	2/4	Branch if carry bit clear : if(C) = 0, then $PC \leftarrow PC + rel$	MM ----- Z -
29	BCS rel	D0	2	2/4	Branch if carry bit set : If (C) = 1, then $PC \leftarrow PC + rel$	-----
30	BEQ rel	F0	2	2/4	Branch if equal : if (Z) = 1, then $PC \leftarrow PC + rel$	-----
31	BIT dp	0C	2	4	Bit test A with memory :	MM ----- Z -
32	BIT !abs	1C	3	5	$Z \leftarrow A \wedge M, N \leftarrow (M_7), V \leftarrow (M_6)$	
33	BMI rel	90	2	2/4	Branch if minus : if (N) = 1, then $PC \leftarrow PC + rel$	-----
34	BNE rel	70	2	2/4	Branch if not equal : if (Z) = 0, then $PC \leftarrow PC + rel$	-----
35	BPL rel	10	2	2/4	Branch if not minus : if (N) = 0, then $PC \leftarrow PC + rel$	-----
36	BRA rel	2F	2	4	Branch always : $PC \leftarrow PC + rel$	-----
37	BRK	0F	1	8	Software interrupt: $B \leftarrow "1", M(SP) \leftarrow (PC_H), SP \leftarrow SP - 1,$ $M(s) \leftarrow (PC_L), SP \leftarrow S - 1, M(SP) \leftarrow PSW,$ $SP \leftarrow SP - 1, PC_L \leftarrow (0FFDE_H), PC_H \leftarrow (0FFDF_H)$	--- 1 - 0 --
38	BVC rel	30	2	2/4	Branch if overflow bit clear : If (V) = 0, then $PC \leftarrow PC + rel$	-----
39	BVS rel	B0	2	2/4	Branch if overflow bit set : If (V) = 1, then $PC \leftarrow PC + rel$	-----

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
40	CALL !abs	3B	3	8	Subroutine call	
41	CALL [dp]	5F	2	8	$M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP-1$, $M(SP) \leftarrow (PC_L)$, $SP \leftarrow SP-1$ if !abs, $PC \leftarrow abs$; if [dp], $PC_L \leftarrow (dp)$, $PC_H \leftarrow (dp+1)$	-----
42	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal ;	
43	CBNE dp + X, rel	8D	3	6/8	If $A \neq (M)$, then $PC \leftarrow PC + rel$.	-----
44	CLR1 dp.bit	y1	2	4	Clear bit : $(M.bit) \leftarrow "0"$	-----
45	CLR1A A.bit	2B	2	2	Clear A.bit : $(A.bit) \leftarrow "0"$	-----
46	CLRC	20	1	2	Clear C-flag : $C \leftarrow "0"$	-----0
47	CLRG	40	1	2	Clear G-flag : $G \leftarrow "0"$	--0-----
48	CLRV	80	1	2	Clear V-flag : $V \leftarrow "0"$	-0--0----
49	CMP #imm	44	2	2	Compare accumulator contents with memory contents $A - (M)$	N-----ZC
50	CMP dp	45	2	3		
51	CMP dp + X	46	2	4		
52	CMP !abs	47	3	4		
53	CMP !abs + Y	55	3	5		
54	CMP [dp + X]	56	2	6		
55	CMP [dp] + Y	57	2	6		
56	CMP {X}	54	1	3		
57	CMPW dp	5D	2	4	Compare YA contents with memory pair contents : $YA - (dp+1)(dp)$	N-----ZC
58	CMPX #imm	5E	2	2	Compare X contents with memory contents $X - (M)$	N-----ZC
59	CMPX dp	6C	2	3		
60	CMPX !abs	7C	3	4		
61	CMPY #imm	7E	2	2	Compare Y contents with memory contents $Y - (M)$	N-----ZC
62	CMPY dp	8C	2	3		
63	CMPY !abs	9C	3	4		
64	COM dp	2C	2	4	1's complement : $(dp) \leftarrow \overline{(dp)}$	N-----Z-
65	DAA	DF	1	3	Decimal adjust for addition	N-----ZC
66	DAS	CF	1	3	Decimal adjust for subtraction	N-----ZC
67	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal : if $(M) \neq 0$, then $PC \leftarrow PC + rel$.	-----
68	DBNE Y,rel	7B	2	4/6		
69	DEC A	A8	1	2	Decrement $M \leftarrow M - 1$	N-----Z-
70	DEC dp	A9	2	4		
71	DEC dp + X	B9	2	5		
72	DEC !abs	B8	3	5		
73	DEC X	AF	1	2		
74	DEC Y	BE	1	2		
75	DECW dp	BD	2	6	Decrement memory pair : $(dp+1)(dp) \leftarrow \{(dp+1)(dp)\} - 1$	N-----Z-
76	DI	60	1	3	Disable interrupts : $I \leftarrow "0"$	-----0--
77	DIV	9B	1	12	Divide : $YA \div A \rightarrow Q:A, R:Y$	NV--H-Z-
78	EI	E0	1	3	Enable interrupts : $I \leftarrow "1"$	-----1--

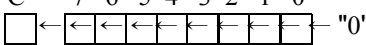
NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC												
79	EOR #imm	A4	2	2	Exclusive OR $A \leftarrow A \oplus (M)$	N-----Z-												
80	EOR dp	A5	2	3														
81	EOR dp + X	A6	2	4														
82	EOR !abs	A7	3	4														
83	EOR !abs + Y	B5	3	5														
84	EOR [dp + X]	96	2	6														
85	EOR [dp] + Y	97	2	6														
86	EOR {X}	94	1	3														
87	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : $C \leftarrow C \oplus (M.bit)$	-----C												
88	EOR1B M.bit	AB	3	5	Bit exclusive-OR C-flag and NOT : $C \leftarrow \overline{C \oplus (M.bit)}$	-----C												
89	INC A	88	1	2	Increment $(M) \leftarrow (M) + 1$	N-----ZC												
90	INC dp	89	2	4														
91	INC dp + X	99	2	5														
92	INC !abs	98	3	5														
93	INC X	8F	1	2														
94	INC Y	9E	1	2														
95	INCW dp	9D	2	6			Increment memory pair : $(dp+1)(dp) \leftarrow \{(dp+1)(dp)\} + 1$	N-----Z-										
96	JMP !abs	1B	3	3	Unconditional jump PC ← jump address	-----												
97	JMP [!abs]	1F	3	5														
98	JMP [dp]	3F	2	4														
99	LDA #imm	C4	2	2	Load accumulator $A \leftarrow (M)$	N-----Z-												
100	LDA dp	C5	2	3														
101	LDA dp + X	C6	2	4														
102	LDA !abs	C7	3	4														
103	LDA !abs + Y	D5	3	5														
104	LDA [dp + X]	D6	2	6														
105	LDA [dp]+Y	D7	2	6														
106	LDA {X}	D4	1	3														
107	LDA {X}+	DB	1	4	X-register auto-increment : $A \leftarrow (M), X \leftarrow X + 1$													
108	LDC M.bit	CB	3	4	Load C-flag : $C \leftarrow (M.bit)$	-----C												
109	LDCB M.bit	CB	3	4	Load C-flag with NOT : $C \leftarrow \overline{(M.bit)}$	-----C												
110	LDM dp,#imm	E4	3	5	Load memory with immediate data : $(M) \leftarrow imm$	-----												
111	LDX #imm	1E	2	2	Load X-register $X \leftarrow (M)$	N-----Z-												
112	LDX dp	CC	2	3														
113	LDX dp + Y	CD	2	4														
114	LDX !abs	DC	3	4														
115	LDY #imm	3E	2	2	Load X-register $Y \leftarrow (M)$	N-----Z-												
116	LDY dp	C9	2	3														
117	LDY dp + Y	D9	2	4														
118	LDY !abs	D8	3	4														
119	LDYA dp	7D	2	5	Load YA : $YA \leftarrow (dp+1)(dp)$	N-----Z-												
120	LSR A	48	1	2	Logical shift right "0" → <table style="display: inline-table; border-collapse: collapse;"><tr><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td><td style="border: 1px solid black; width: 15px; height: 15px;"></td></tr></table> → <table style="display: inline-table; border-collapse: collapse;"><tr><td style="border: 1px solid black; width: 15px; height: 15px;"></td></tr></table>													N-----ZC
121	LSR dp	49	2	4														
122	LSR dp + X	59	2	5														
123	LSR !abs	58	3	5														
124	MUL	5B	1	9	Multiply : $YA \leftarrow Y \times A$	N-----Z-												
125	NOP	FF	1	2	No operation	-----												
126	NOT1 M.bit	4B	3	5	Bit complement : $(M.bit) \leftarrow \overline{(M.bit)}$	-----												

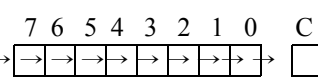
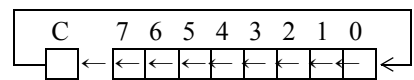
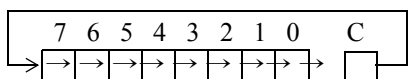
NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
127	OR #imm	64	2	2	Logical OR $A \leftarrow A \vee (M)$	N-----Z-
128	OR dp	65	2	3		
129	OR dp + X	66	2	4		
130	OR !abs	67	3	4		
131	OR !abs + Y	75	3	5		
132	OR [dp + X]	76	2	6		
133	OR [dp] + Y	77	2	6		
134	OR {X}	74	1	3		
135	OR1 M.bit	6B	3	5	Bit OR C-flag : $C \leftarrow C \vee (M.bit)$	-----C
136	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : $C \leftarrow C \vee \overline{(M.bit)}$	-----C
137	PCALL	4F	2	6	U-page call : $M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP - 1$, $M(SP) \leftarrow (PC_L)$, $SP \leftarrow SP - 1$, $PC_L \leftarrow (upage)$, $PC_H \leftarrow "OFF_H"$	-----
138	POP A	0D	1	4	Pop from stack $SP \leftarrow SP + 1$, Reg. $\leftarrow M(SP)$	-----
139	POP X	2D	1	4		
140	POP Y	4D	1	4		
141	POP PSW	6D	1	4		
142	PUSH A	0E	1	4	Push to stack $M(SP) \leftarrow Reg.$ $SP \leftarrow SP - 1$	-----
143	PUSH X	2E	1	4		
144	PUSH Y	4E	1	4		
145	PUSH PSW	6E	1	4		
146	RET	6F	1	5	Return from subroutine : $SP \leftarrow SP + 1$, $PC_L \leftarrow M(SP)$, $SP \leftarrow SP + 1$, $PC_H \leftarrow M(SP)$	-----
147	RETI	7F	1	6	Return from interrupt : $SP \leftarrow SP + 1$, $PSW \leftarrow M(SP)$, $SP \leftarrow SP + 1$, $PC_L \leftarrow M(SP)$, $SP \leftarrow SP + 1$, $PC_H \leftarrow M(SP)$	(restored)
148	ROL A	28	1	2	Rotate left through carry 	N-----ZC
149	ROL dp	29	2	4		
150	ROL dp + X	39	2	5		
151	ROL !abs	38	3	5		
152	ROR A	68	1	2	Rotate right through carry 	N-----ZC
153	ROR dp	69	2	4		
154	ROR dp + X	79	2	5		
155	ROR !abs	78	3	5		
156	SBC #imm	24	2	2	Subtract with carry $A \leftarrow A - (M) - \overline{(C)}$	NV--HZC
157	SBC dp	25	2	3		
158	SBC dp + X	26	2	4		
159	SBC !abs	27	3	4		
160	SBC !abs + Y	35	3	5		
161	SBC [dp + X]	36	2	6		
162	SBC [dp] + Y	37	2	6		
163	SBC {X}	34	1	3		
164	SET1 dp.bit	x1	2	4	Set bit : (M.bit) $\leftarrow "1"$	-----
165	SETA1 A.bit	0B	2	2	Set A.bit : (A.bit) $\leftarrow "1"$	-----
166	SETC	A0	1	2	Set C-flag : $C \leftarrow "1"$	-----1
167	SETG	C0	1	2	Set G-flag : $G \leftarrow "1"$	--1-----

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
168	STA dp	E5	2	3	Store accumulator contents in memory	
169	STA dp + X	E6	2	4	(M) ← A	
170	STA !abs	E7	3	4		
171	STA !abs + Y	F5	3	5		
172	STA [dp + X]	F6	2	6		-----
173	STA [dp] + Y	F7	2	6		
174	STA {X}	F4	1	3		
175	STA {X}+	FB	1	4	X-register auto-increment : (M) ← A, X ← X + 1	
176	STC M.bit	EB	3	6	Store C-flag : (M.bit) ← C	-----
177	STOP	EF	1	3	Stop mode (halt CPU, stop oscillator)	-----
178	STX dp	EC	2	4	Store X-register contents in memory	
179	STX dp + Y	ED	2	5	(M) ← X	-----
180	STX !abs	FC	3	5		
181	STY dp	E9	2	4	Store Y-register contents in memory	
182	STY dp + X	F9	2	5	(M) ← Y	-----
183	STY !abs	F8	3	5		
184	STYA dp	DD	2	5	Store YA : (dp+1)(dp) ← YA	-----
185	SUBW dp	3D	2	5	16-bits subtract without carry : YA ← YA - (dp+1)(dp)	NV -- H - ZC
186	TAX	E8	1	2	Transfer accumulator contents to X-register : X ← A	N ----- Z -
187	TAY	9F	1	2	Transfer accumulator contents to Y-register : Y ← A	N ----- Z -
188	TCALL n	nA	1	8	Table call : M(SP) ← (PC _H), SP ← SP -1, M(SP) ← (PC _L), SP ← SP -1 PC _L ← (Table vector L), PC _H ← (Table vector H)	-----
189	TCLR1 !abs	5C	3	6	Test and clear bits with A : A - (M), (M) ← (M) ∧ (A)	N ----- Z -
190	TSET1 !abs	3C	3	6	Test and set bits with A : A - (M), (M) ← (M) ∨ (A)	N ----- Z -
191	TSPX	AE	1	2	Transfer stack-pointer contents to X-register : X ← SP	N ----- Z -
192	TST dp	4C	2	3	Test memory contents for negative or zero : (dp) - 00 _H	N ----- Z -
193	TXA	C8	1	2	Transfer X-register contents to accumulator : A ← X	N ----- Z -
194	TXSP	8E	1	2	Transfer X-register contents to stack-pointer : SP ← X	N ----- Z -
195	TYA	BF	1	2	Transfer Y-register contents to accumulator : A ← Y	N ----- Z -
196	XAX	EE	1	4	Exchange X-register contents with accumulator : X ↔ A	-----
197	XAY	DE	1	4	Exchange Y-register contents with accumulator : Y ↔ A	-----
198	XCN	CE	1	5	Exchange nibbles within the accumulator: A ₇ ~ A ₄ ↔ A ₃ ~ A ₀	N ----- Z -
199	XMA dp	BC	2	5	Exchange memory contents with accumulator	
200	XMA dp + X	AD	2	6	(M) ↔ A	N ----- Z -
201	XMA {X}	BB	1	5		
202	XYX	FE	1	4	Exchange X-register contents with Y-register : X ↔ Y	-----

1.3 Instruction Table by Function

1.3.1 Arithmetic/Logic Operation

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADC #imm	04	2	2	Add with carry. $A \leftarrow A + (M) + C$	NV -- H - ZC
2	ADC dp	05	2	3		
3	ADC dp + X	06	2	4		
4	ADC !abs	07	3	4		
5	ADC !abs+Y	15	3	5		
6	ADC [dp+X]	16	2	6		
7	ADC [dp]+Y	17	2	6		
8	ADC {X}	14	1	3		
9	AND #imm	84	2	2	Logical AND $A \leftarrow A \wedge (M)$	N ----- Z -
10	AND dp	85	2	3		
11	AND dp + X	86	2	4		
12	AND !abs	87	3	4		
13	AND !abs+Y	95	3	5		
14	AND [dp+X]	96	2	6		
15	AND [dp] + Y	97	2	6		
16	AND {X}	94	1	3		
17	ASL A	08	1	2	Arithmetic shift left $C \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0$ 	N ----- ZC
18	ASL dp	09	2	4		
19	ASL dp + X	19	2	5		
20	ASL !abs	18	3	5		
21	CMP #imm	44	2	2	Compare accumulator contents with memory contents $A - (M)$	N ----- ZC
22	CMP dp	45	2	3		
23	CMP dp + X	46	2	4		
24	CMP !abs	47	3	4		
25	CMP !abs + Y	55	3	5		
26	CMP [dp + X]	56	2	6		
27	CMP [dp] + Y	57	2	6		
28	CMP {X}	54	1	3		
29	CMPX #imm	5E	2	2	Compare X contents with memory contents $X - (M)$	N ----- ZC
30	CMPX dp	6C	2	3		
31	CMPX !abs	7C	3	4		
32	CMPY #imm	7E	2	2	Compare Y contents with memory contents $Y - (M)$	N ----- ZC
33	CMPY dp	8C	2	3		
34	CMPY !abs	9C	3	4		
35	COM dp	2C	2	4	1's complement : $(dp) \leftarrow \overline{(dp)}$	N ----- Z -
36	DAA	DF	1	3	Decimal adjust for addition	N ----- ZC
37	DAS	CF	1	3	Decimal adjust for subtraction	N ----- ZC
38	DEC A	A8	1	2	Decrement $M \leftarrow M - 1$	N ----- Z -
39	DEC dp	A9	2	4		
40	DEC dp + X	B9	2	5		
41	DEC !abs	B8	3	5		
42	DEC X	AF	1	2		
43	DEC Y	BE	1	2		
44	DIV	9B	1	12		

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
45	EOR #imm	A4	2	2	Exclusive OR $A \leftarrow A \oplus (M)$	N-----Z-
46	EOR dp	A5	2	3		
47	EOR dp + X	A6	2	4		
48	EOR !abs	A7	3	4		
49	EOR !abs + Y	B5	3	5		
50	EOR [dp + X]	96	2	6		
51	EOR [dp] + Y	97	2	6		
52	EOR {X}	94	1	3		
53	INC A	88	1	2	Increment $(M) \leftarrow (M) + 1$	N-----ZC
54	INC dp	89	2	4		
55	INC dp + X	99	2	5		
56	INC !abs	98	3	5		
57	INC X	8F	1	2		
58	INC Y	9E	1	2		
59	LSR A	48	1	2	Logical shift right "0" → 	N-----ZC
60	LSR dp	49	2	4		
61	LSR dp + X	59	2	5		
62	LSR !abs	58	3	5		
63	MUL	5B	1	9	Multiply : $YA \leftarrow Y \times A$	N-----Z-
64	OR #imm	64	2	2	Logical OR $A \leftarrow A \vee (M)$	N-----Z-
65	OR dp	65	2	3		
66	OR dp + X	66	2	4		
67	OR !abs	67	3	4		
68	OR !abs + Y	75	3	5		
69	OR [dp + X]	76	2	6		
70	OR [dp] + Y	77	2	6		
71	OR {X}	74	1	3		
72	ROL A	28	1	2	Rotate left through carry 	N-----ZC
73	ROL dp	29	2	4		
74	ROL dp + X	39	2	5		
75	ROL !abs	38	3	5		
76	ROR A	68	1	2	Rotate right through carry 	N-----ZC
77	ROR dp	69	2	4		
78	ROR dp + X	79	2	5		
79	ROR !abs	78	3	5		
80	SBC #imm	24	2	2	Subtract with carry $A \leftarrow A - (M) - \overline{(C)}$	NV--HZC
81	SBC dp	25	2	3		
82	SBC dp + X	26	2	4		
83	SBC !abs	27	3	4		
84	SBC !abs + Y	35	3	5		
85	SBC [dp + X]	36	2	6		
86	SBC [dp] + Y	37	2	6		
87	SBC {X}	34	1	3		
88	TST dp	4C	2	3	Test memory contents for negative or zero : (dp) - 00 _H	N-----Z-
89	XCN	CE	1	5	Exchange nibbles within the accumulator: $A_7 \sim A_4 \leftrightarrow A_3 \sim A_0$	N-----Z-

1.3.2 Register / Memory Operation

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	LDA #imm	C4	2	2	Load accumulator	N-----Z-
2	LDA dp	C5	2	3	$A \leftarrow (M)$	
3	LDA dp + X	C6	2	4		
4	LDA !abs	C7	3	4		
5	LDA !abs + Y	D5	3	5		
6	LDA [dp + X]	D6	2	6		
7	LDA [dp]+Y	D7	2	6		
8	LDA {X}	D4	1	3		
9	LDA {X}+	DB	1	4	X-register auto-increment : $A \leftarrow (M), X \leftarrow X + 1$	
10	LDM dp,#imm	E4	3	5	Load memory with immediate data : $(M) \leftarrow \text{imm}$	-----
11	LDX #imm	1E	2	2	Load X-register	N-----Z-
12	LDX dp	CC	2	3	$X \leftarrow (M)$	
13	LDX dp + Y	CD	2	4		
14	LDX !abs	DC	3	4		
15	LDY #imm	3E	2	2	Load X-register	N-----Z-
16	LDY dp	C9	2	3	$Y \leftarrow (M)$	
17	LDY dp + Y	D9	2	4		
18	LDY !abs	D8	3	4		
19	STA dp	E5	2	3	Store accumulator contents in memory	-----
20	STA dp + X	E6	2	4	$(M) \leftarrow A$	
21	STA !abs	E7	3	4		
22	STA !abs + Y	F5	3	5		
23	STA [dp + X]	F6	2	6		
24	STA [dp] + Y	F7	2	6		
25	STA {X}	F4	1	3		
26	STA {X}+	FB	1	4	X-register auto-increment : $(M) \leftarrow A, X \leftarrow X + 1$	
27	STX dp	EC	2	4	Store X-register contents in memory	-----
28	STX dp + Y	ED	2	5	$(M) \leftarrow X$	
29	STX !abs	FC	3	5		
30	STY dp	E9	2	4	Store Y-register contents in memory	-----
31	STY dp + X	F9	2	5	$(M) \leftarrow Y$	
32	STY !abs	F8	3	5		
33	TAX	E8	1	2	Transfer accumulator contents to X-register : $X \leftarrow A$	N-----Z-
34	TAY	9F	1	2	Transfer accumulator contents to Y-register : $Y \leftarrow A$	N-----Z-
35	TSPX	AE	1	2	Transfer stack-pointer contents to X-register : $X \leftarrow \text{SP}$	N-----Z-
36	TXA	C8	1	2	Transfer X-register contents to accumulator : $A \leftarrow X$	N-----Z-
37	TXSP	8E	1	2	Transfer X-register contents to stack-pointer : $\text{SP} \leftarrow X$	N-----Z-
38	TYA	BF	1	2	Transfer Y-register contents to accumulator : $A \leftarrow Y$	N-----Z-
39	XAX	EE	1	4	Exchange X-register contents with accumulator : $X \leftrightarrow A$	-----
40	XAY	DE	1	4	Exchange Y-register contents with accumulator : $Y \leftrightarrow A$	-----
41	XMA dp	BC	2	5	Exchange memory contents with accumulator	N-----Z-
42	XMA dp + X	AD	2	6	$(M) \leftrightarrow A$	
43	XMA {X}	BB	1	5		
44	XYX	FE	1	4	Exchange X-register contents with Y-register : $X \leftrightarrow Y$	-----

1.3.3 16-Bit Operation

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADDW dp	1D	2	5	16-bits add without carry : $YA \leftarrow YA + (dp+1)(dp)$	NV -- H - ZC
2	CMPW dp	5D	2	4	Compare YA contents with memory pair contents : $YA - (dp+1)(dp)$	N - - - - - ZC
3	DECW dp	BD	2	6	Decrement memory pair : $(dp+1)(dp) \leftarrow \{(dp+1)(dp)\} - 1$	N - - - - - Z -
4	INCW dp	9D	2	6	Increment memory pair : $(dp+1)(dp) \leftarrow \{(dp+1)(dp)\} + 1$	N - - - - - Z -
5	LDYA dp	7D	2	5	Load YA : $YA \leftarrow (dp+1)(dp)$	N - - - - - Z -
6	STYA dp	DD	2	5	Store YA : $(dp+1)(dp) \leftarrow YA$	- - - - - - -
7	SUBW dp	3D	2	5	16-bits subtract without carry : $YA \leftarrow YA - (dp+1)(dp)$	NV -- H - ZC

1.3.4 Bit Manipulation

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	AND1 M.bit	8B	3	4	Bit AND C-flag : $C \leftarrow C \wedge (M.bit)$	- - - - - C
2	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : $C \leftarrow C \wedge (M.bit)$	- - - - - C
3	BIT dp	0C	2	4	Bit test A with memory :	MM - - - - - Z -
4	BIT !abs	1C	3	5	$Z \leftarrow A \wedge M, N \leftarrow (M_7), V \leftarrow (M_6)$	
5	CLR1 dp.bit	y1	2	4	Clear bit : $(M.bit) \leftarrow "0"$	- - - - - - -
6	CLR1A A.bit	2B	2	2	Clear A.bit : $(A.bit) \leftarrow "0"$	- - - - - - -
7	CLRC	20	1	2	Clear C-flag : $C \leftarrow "0"$	- - - - - 0
8	CLRG	40	1	2	Clear G-flag : $G \leftarrow "0"$	- - 0 - - - -
9	CLRv	80	1	2	Clear V-flag : $V \leftarrow "0"$	- 0 - - - 0 - - -
10	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : $C \leftarrow C \oplus (M.bit)$	- - - - - C
11	EOR1B M.bit	AB	3	5	Bit exclusive-OR C-flag and NOT : $C \leftarrow C \oplus \sim(M.bit)$	- - - - - C
12	LDC M.bit	CB	3	4	Load C-flag : $C \leftarrow (M.bit)$	- - - - - C
13	LDCB M.bit	CB	3	4	Load C-flag with NOT : $C \leftarrow \sim(M.bit)$	- - - - - C
14	NOT1 M.bit	4B	3	5	Bit complement : $(M.bit) \leftarrow \sim(M.bit)$	- - - - - - -
15	OR1 M.bit	6B	3	5	Bit OR C-flag : $C \leftarrow C \vee (M.bit)$	- - - - - C
16	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : $C \leftarrow C \vee \sim(M.bit)$	- - - - - C
17	SET1 dp.bit	x1	2	4	Set bit : $(M.bit) \leftarrow "1"$	- - - - - - -
18	SETA1 A.bit	0B	2	2	Set A.bit : $(A.bit) \leftarrow "1"$	- - - - - - -
19	SETC	A0	1	2	Set C-flag : $C \leftarrow "1"$	- - - - - 1
20	SETG	C0	1	2	Set G-flag : $G \leftarrow "1"$	- - 1 - - - -
21	STC M.bit	EB	3	6	Store C-flag : $(M.bit) \leftarrow C$	- - - - - - -
22	TCLR1 !abs	5C	3	6	Test and clear bits with A : $A - (M), (M) \leftarrow (M) \wedge (A)$	N - - - - - Z -
23	TSET1 !abs	3C	3	6	Test and set bits with A : $A - (M), (M) \leftarrow (M) \vee (A)$	N - - - - - Z -

1.3.5 Branch / Jump Operation

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BBC A.bit,rel	y2	2	4/6	Branch if bit clear :	-----
2	BBC dp.bit,rel	y3	3	5/7	if(bit) = 0, then PC ← PC + rel	-----
3	BBS A.bit,rel	x2	2	4/6	Branch if bit clear :	-----
4	BBS dp.bit,rel	x3	3	5/7	if(bit) = 1, then PC ← PC + rel	-----
5	BCC rel	50	2	2/4	Branch if carry bit clear : if(C) = 0, then PC ← PC + rel	MM ---- Z -
6	BCS rel	D0	2	2/4	Branch if carry bit set : If (C) =1, then PC ← PC + rel	-----
7	BEQ rel	F0	2	2/4	Branch if equal : if (Z) = 1, then PC ← PC + rel	-----
8	BMI rel	90	2	2/4	Branch if minus : if (N) = 1, then PC ← PC + rel	-----
9	BNE rel	70	2	2/4	Branch if not equal : if (Z) = 0, then PC ← PC + rel	-----
10	BPL rel	10	2	2/4	Branch if not minus : if (N) = 0, then PC ← PC + rel	-----
11	BRA rel	2F	2	4	Branch always : PC ← PC + rel	-----
12	BVC rel	30	2	2/4	Branch if overflow bit clear : If (V) = 0, then PC ← PC + rel	-----
13	BVS rel	B0	2	2/4	Branch if overflow bit set : If (V) = 1, then PC ← PC + rel	-----
14	CALL !abs	3B	3	8	Subroutine call	-----
15	CALL [dp]	5F	2	8	M(SP) ← (PC _H), SP ← SP-1, M(SP) ← (PC _L), SP ← SP-1 if !abs, PC ← abs ; if [dp], PC _L ← (dp), PC _H ← (dp+1)	-----
16	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal ;	-----
17	CBNE dp + X, rel	8D	3	6/8	If A ≠ (M), then PC ← PC + rel.	-----
18	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal :	-----
19	DBNE Y,rel	7B	2	4/6	if (M) ≠ 0, then PC ← PC + rel.	-----
20	JMP !abs	1B	3	3	Unconditional jump	-----
21	JMP [!abs]	1F	3	5	PC ← jump address	-----
22	JMP [dp]	3F	2	4		-----
23	PCALL	4F	2	6	U-page call : M(SP) ← (PC _H), SP ← SP -1, M(SP) ← (PC _L), SP ← SP -1, PC _L ← (upage), PC _H ← "OFF _H "	-----
24	TCALL n	nA	1	8	Table call : M(SP) ← (PC _H), SP ← SP -1, M(SP) ← (PC _L), SP ← SP -1 PC _L ← (Table vector L), PC _H ← (Table vector H)	-----

1.3.6 Control Operation & etc.

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BRK	0F	1	8	Software interrupt: B ← "1", M(SP) ← (PC _H), SP ← SP - 1, M(s) ← (PC _L), SP ← S - 1, M(SP) ← PSW, SP ← SP - 1, PC _L ← (OFFDE _H), PC _H ← (OFFDF _H)	--- 1 - 0 --
2	DI	60	1	3	Disable interrupts : I ← "0"	----- 0 --
3	EI	E0	1	3	Enable interrupts : I ← "1"	----- 1 --

NO.	MNEMONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
4	NOP	FF	1	2	No operation	-----
5	POP A	0D	1	4	Pop from stack	----- (restored)
6	POP X	2D	1	4	SP ← SP + 1, Reg. ← M(SP)	
7	POP Y	4D	1	4		
8	POP PSW	6D	1	4		
9	PUSH A	0E	1	4	Push to stack	-----
10	PUSH X	2E	1	4	M(SP) ← Reg. SP ← SP - 1	
11	PUSH Y	4E	1	4		
12	PUSH PSW	6E	1	4		
13	RET	6F	1	5	Return from subroutine : SP ← SP+1, PC _L ← M(SP), SP ← SP+1, PC _H ← M(SP)	-----
14	RETI	7F	1	6	Return from interrupt : SP ← SP+1, PSW ← M(SP), SP ← SP+1, PC _L ← M(SP), SP ← SP+1, PC _H ← M(SP)	(restored)
15	STOP	EF	1	3	Stop mode (halt CPU, stop oscillator)	-----