Oct. 1999 Ver 1.0

8-BIT SINGLE-CHIP MICROCONTROLLERS

GMS87C1404
GMS87C1408

User's Manual

HYUNDAI MicroElectronics

Table of Contents

GMS87C1404 / GMS87C1408

CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER

1. OVERVIEW

1.1 Description

The GMS87C1404 and GMS87C1408 are an advanced CMOS 8-bit microcontroller with 4K/8K bytes of EPROM. The HYUNDAI MicroElectronics GMS87C1404 and GMS87C1408 are a powerful microcontroller which provides a highly flexible and cost effective solution to many small applications such as controller for battery charger. The GMS87C1404 and GMS87C1408 provide the following standard features: 4K/8K bytes of EPROM, 192 bytes of RAM, 8-bit timer/counter, 8-bit A/D converter, 10-bit high speed PWM output, programmable buzzer driving port, 8-bit serial communication port, on-chip oscillator and clock circuitry. In addition, the GMS87C1404 and GMS87C1408 supports power saving modes to reduce power consumption.

Device name	ROM Size	RAM Size	Package
GMS87C1404	4K bytes	192bytes	28 SKDIP or SOP
GMS87C1408	8K bytes	192bytes	28 SKDIP or SOP

1.2 Features

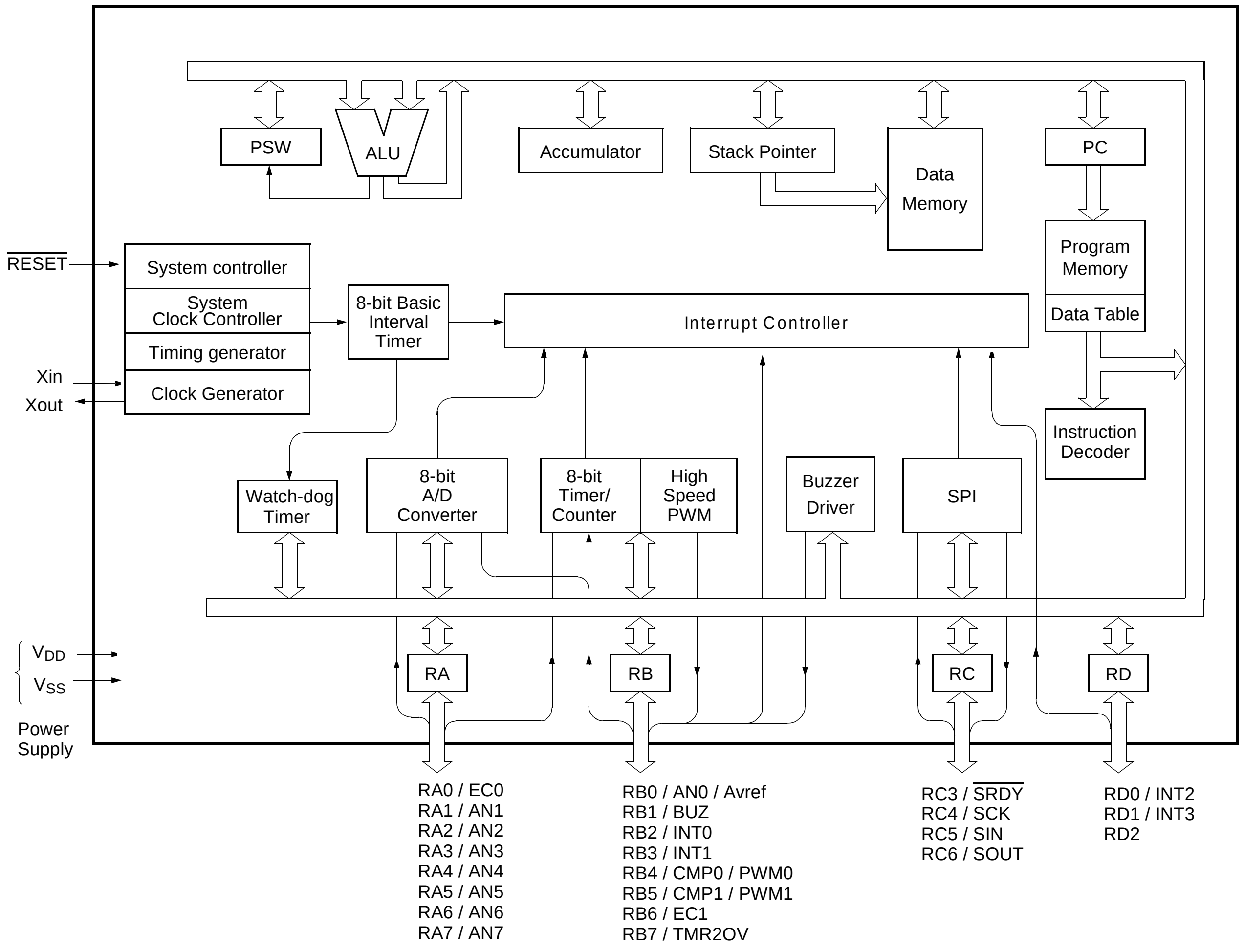
- **4K/8K Bytes On-chip Program Memory (OTP)**
- **192 Bytes of On-chip Data RAM (Included stack memory)**
- **Instruction Cycle Time:**
  - 250nS at 8MHz
- **23 Programmable I/O pins (LED direct driving can be source and sink)**
- **2.5V to 5.5V Wide Operating Range**
- **One 8-bit A/D Converter**
- **One 8-bit Basic Interval Timer**
- **Four 8-bit Timer / Counters**
- **Two 10-bit High Speed PWM Outputs**
- **Watchdog timer (can be operate with internal RC-oscillation)**
- **One 8-bit Serial Peripheral Interface**
- **Twelve Interrupt sources**
  - External input: 4
  - A/D Conversion: 1
  - Serial Peripheral Interface: 1
  - Timer: 6
- **One Programmable Buzzer Driving port**
  - 500Hz ~ 130kHz
- **Oscillator Type**
  - Crystal
  - Ceramic Resonator
- **Noise Immunity Circuit**
  - Power Fail Processor
- **Power Down Mode**
  - STOP mode
  - Wake-up Timer mode

1.3 Development Tools

The GMS87C1404 and GMS87C1408 are supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr^{TM}.

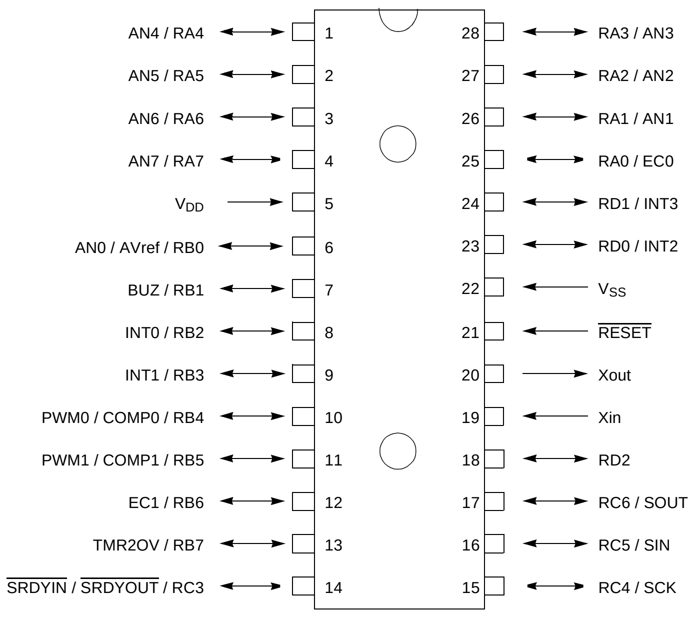
In Circuit Emulators	CHOICE-Dr.
Assembler	HME Macro Assembler
OTP Writer	Dr. Writer

2. BLOCK DIAGRAM

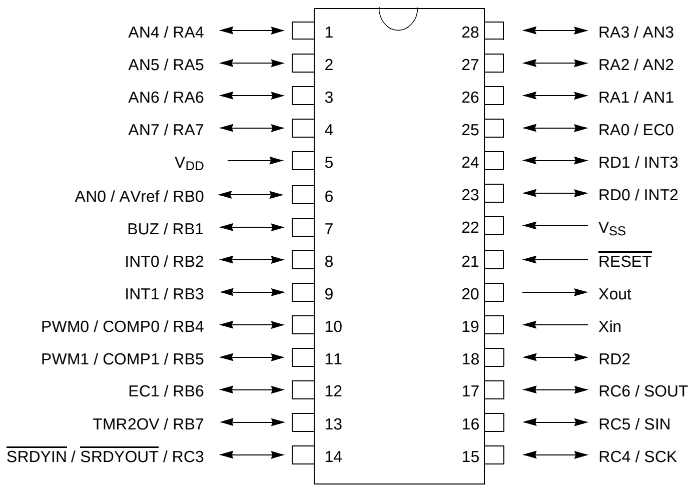


3. PIN ASSIGNMENT

28 SKINNY DIP

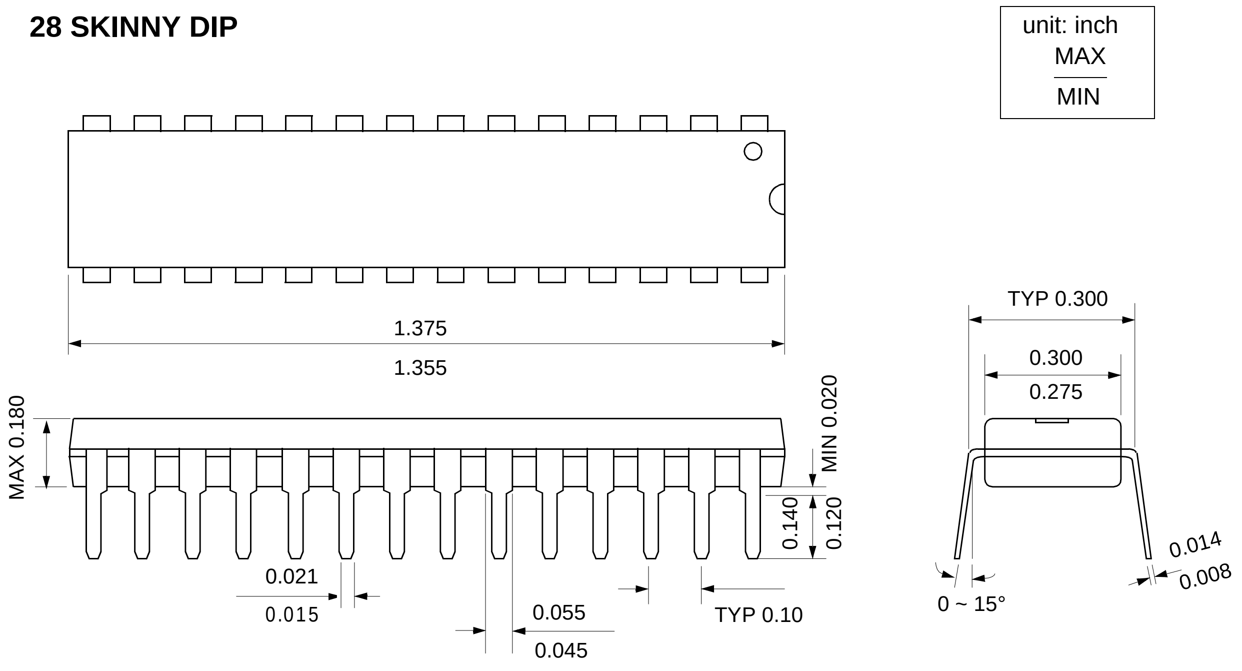


28 SOP

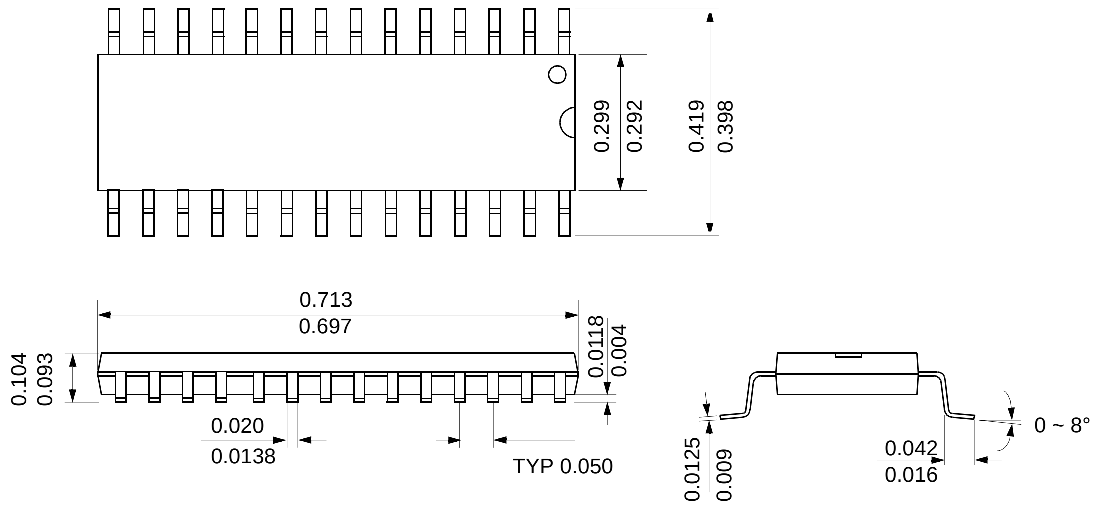


4. PACKAGE DIAGRAM

28 SKINNY DIP



28 SOP



5. PIN FUNCTION

V_{DD}: Supply voltage.

V_{SS}: Circuit ground.

$\overline{\text{RESET}}$: 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.

RA0~RA7: RA is an 8-bit, CMOS, bidirectional I/O port. RA pins can be used as outputs or inputs according to “1” or “0” written the their Port Direction Register(RAIO).

Port pin	Alternate function
RA0	EC0 (Event Counter Input Source)
RA1	AN1 (Analog Input Port 1)
RA2	AN2 (Analog Input Port 2)
RA3	AN3 (Analog Input Port 3)
RA4	AN4 (Analog Input Port 4)
RA5	AN5 (Analog Input Port 5)
RA6	AN6 (Analog Input Port 6)
RA7	AN7 (Analog Input Port 7)

Table 5-1 RA Port

In addition, RA serves the functions of the various special features in Table 5-1 .

RB0~RB7: RB is a 8-bit, CMOS, bidirectional I/O port. RB pins can be used as outputs or inputs according to “1” or “0” written the their Port Direction Register(RBIO).

RB serves the functions of the various following special features in Table 5-2

Port pin	Alternate function
RB0	AN0 (Analog Input Port 0)
	AVref (External Analog Reference Pin)
RB1	BUZ (Buzzer Driving Output Port)
RB2	INT0 (External Interrupt Input Port 0)
RB3	INT1 (External Interrupt Input Port 1)
RB4	PWM0 (PWM0 Output)
	COMP0 (Timer1 Compare Output)
RB5	PWM1 (PWM1 Output)
	COMP1 (Timer3 Compare Output)
RB6	EC1 (Event Counter Input Source)
RB7	TMR2OV (Timer2 Overflow Output)

Table 5-2 RB Port

RC3~RC6: RC is a 4-bit, CMOS, bidirectional I/O port. RC pins can be used as outputs or inputs according to “1” or “0” written the their Port Direction Register(RCIO).

RC serves the functions of the serial interface following special features in Table 5-3 .

Port pin	Alternate function
RC3	$\overline{\text{SRDYIN}}$ (SPI Ready Input)
	$\overline{\text{SRDYOUT}}$ (SPI Ready Output)
RC4	SCKI (SPI CLK Input)
	SCKO (SPI CLK Output)
RC5	SIN (SPI Serial Data Input)
RC6	SOUT (SPI Serial Data Output)

Table 5-3 RC Port

RD0~RD2: RD is a 3-bit, CMOS, bidirectional I/O port. RC pins can be used as outputs or inputs according to “1” or “0” written the their Port Direction Register(RDIO).

RD serves the functions of the external interrupt following special features in Table 5-4

Port pin	Alternate function
RD0	INT2 (External Interrupt Input Port 2)
RD1	INT3 (External Interrupt Input Port 3)
RD2	

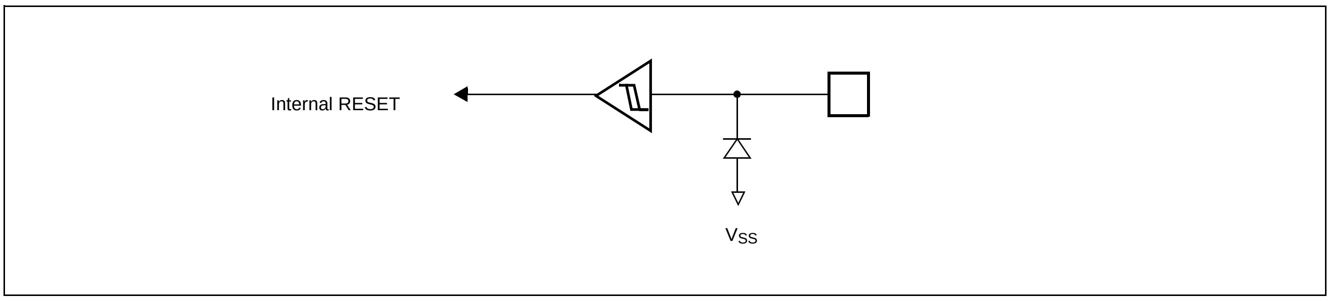
Table 5-4 RD Port

PIN NAME	Pin No.	In/Out	Function	
V_{DD}	5	-	Supply voltage	
V_{SS}	22	-	Circuit ground	
$\overline{RESET}$	21	I	Reset signal input	
X_{IN}	19	I		
X_{OUT}	20	O		
RA0 (EC0)	25	I/O (Input)		External Event Counter input 0
RA1 (AN1)	26	I/O (Input)		Analog Input Port 1
RA2 (AN2)	27	I/O (Input)		Analog Input Port 2
RA3 (AN3)	28	I/O (Input)		Analog Input Port 3
RA4 (AN4)	1	I/O (Input)	8-bit general I/O ports	Analog Input Port 4
RA5 (AN5)	2	I/O (Input)		Analog Input Port 5
RA6 (AN6)	3	I/O (Input)		Analog Input Port 6
RA7 (AN7)	4	I/O (Input)		Analog Input Port 7
RB0 (AVref/AN0)	6	I/O (Input)		Analog Input Port 0 / Analog Reference
RB1 (INT0)	7	I/O (Input)		External Interrupt Input 0
RB2 (INT1)	8	I/O (Input)		External Interrupt Input 1
RB3 (BUZ)	9	I/O (Output)		Buzzer Driving Output
RB4 (PWM0/COMP0)	10	I/O (Output/Output)	8-bit general I/O ports	PWM0 Output or Timer1 Compare Output
RB5 (PWM1/COMP1)	11	I/O (Output/Output)		PWM1 Output or Timer3 Compare Output
RB6 (EC1)	12	I/O (Output/Output)		External Event Counter input 1
RB7 (TMR2OV)	13	I/O (Output/Output)		Timer2 Overflow Output
RC3 ($\overline{SRDYIN}$/$\overline{SRDYOUT}$)	14	I/O (Input/Output)		SPI READY Input/Output
RC4 (SCK)	15	I/O (Input/Output)	4-bit general I/O ports	SPI CLK Input/Output
RC5 (SIN)	16	I/O (Input)		SPI DATA Input
RC6 (SOUT)	17	I/O (Output)		SPI DATA Output
RD0 (INT2)	23	I/O (Input)		External Interrupt Input 2
RD1 (INT3)	24	I/O (Input)	3-bit general I/O ports	External Interrupt Input 3
RD2	18	I/O		

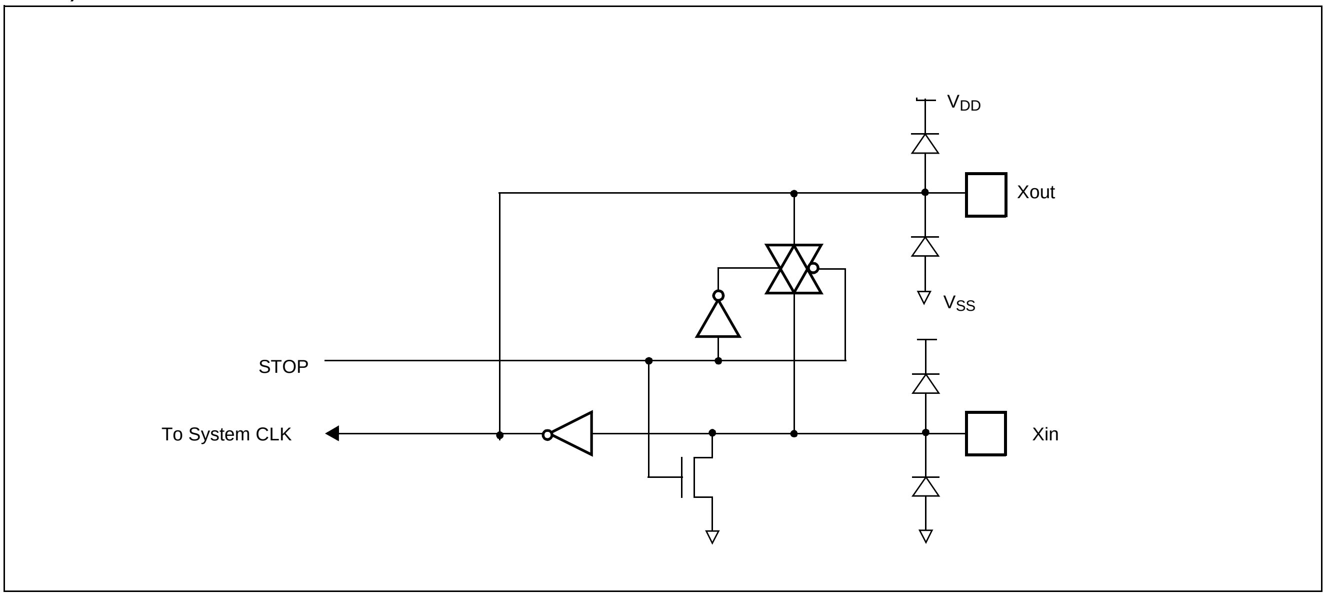
Table 5-5 Pin Description

6. PORT STRUCTURES

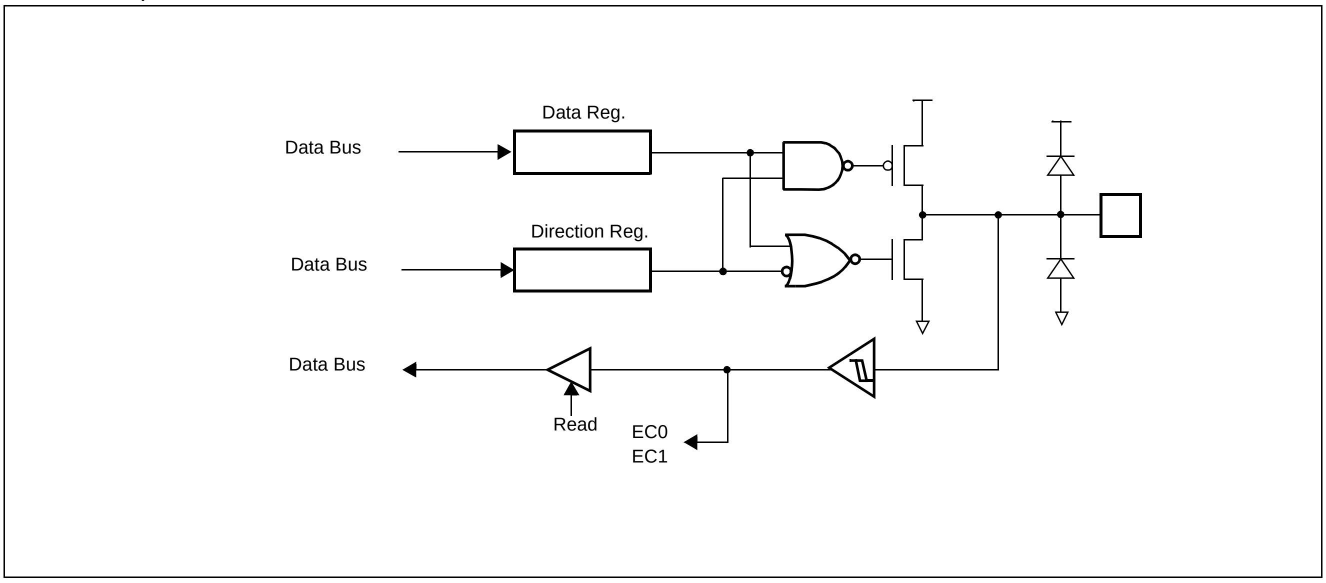
• $\overline{\text{RESET}}$



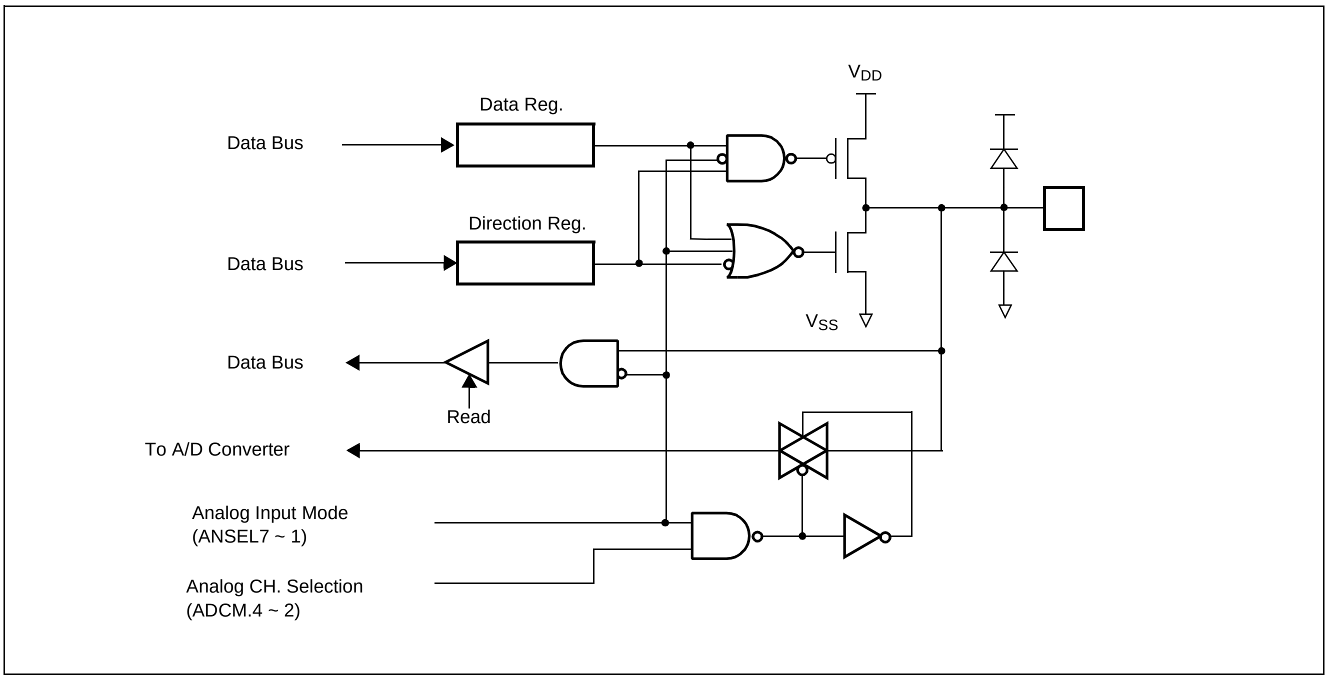
• Xin, Xout



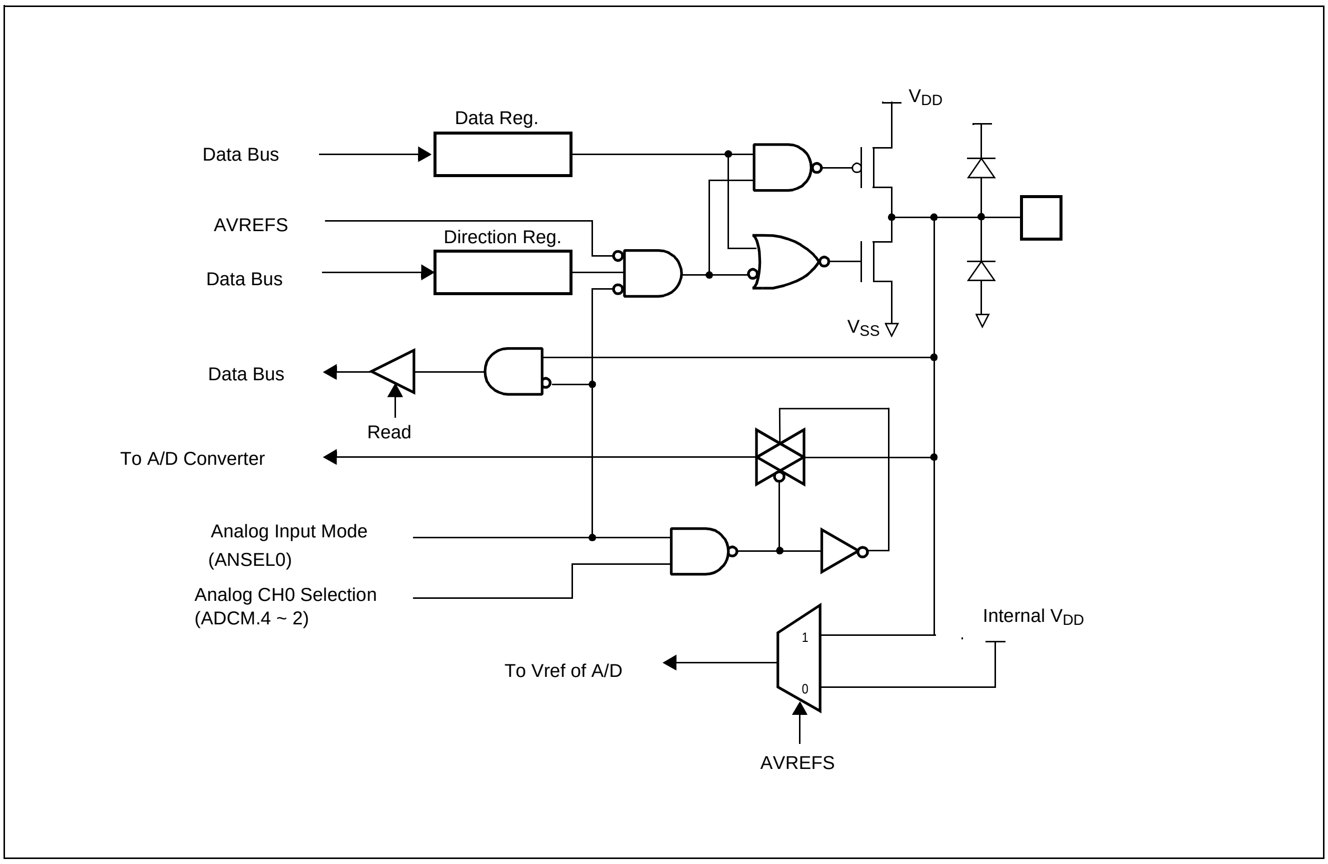
• RA0/EC0, RB6/EC1



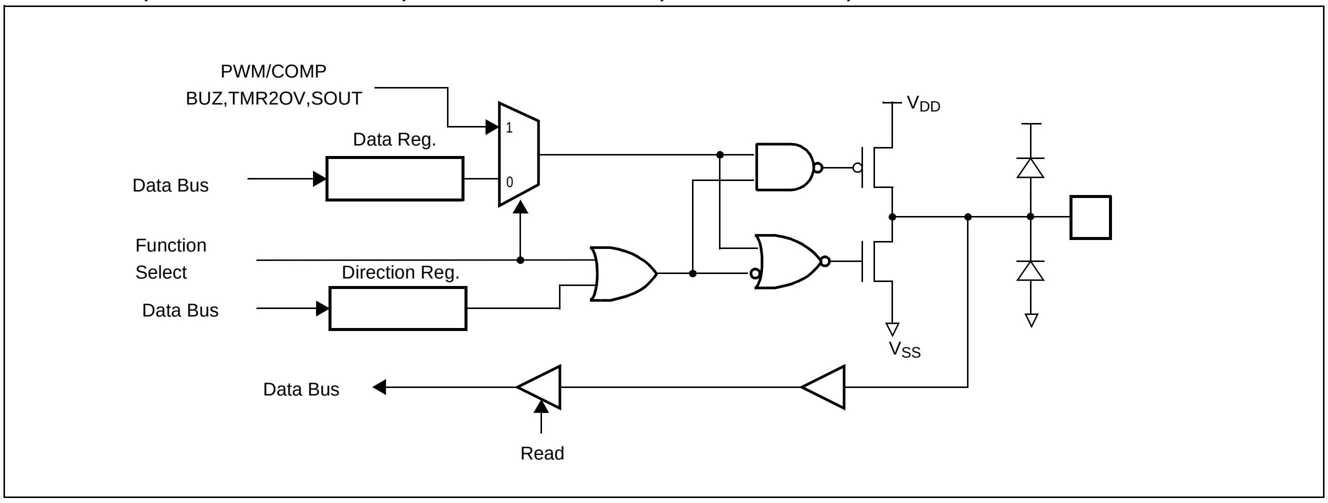
- **RA1/AN1 ~ RA7/AN7**



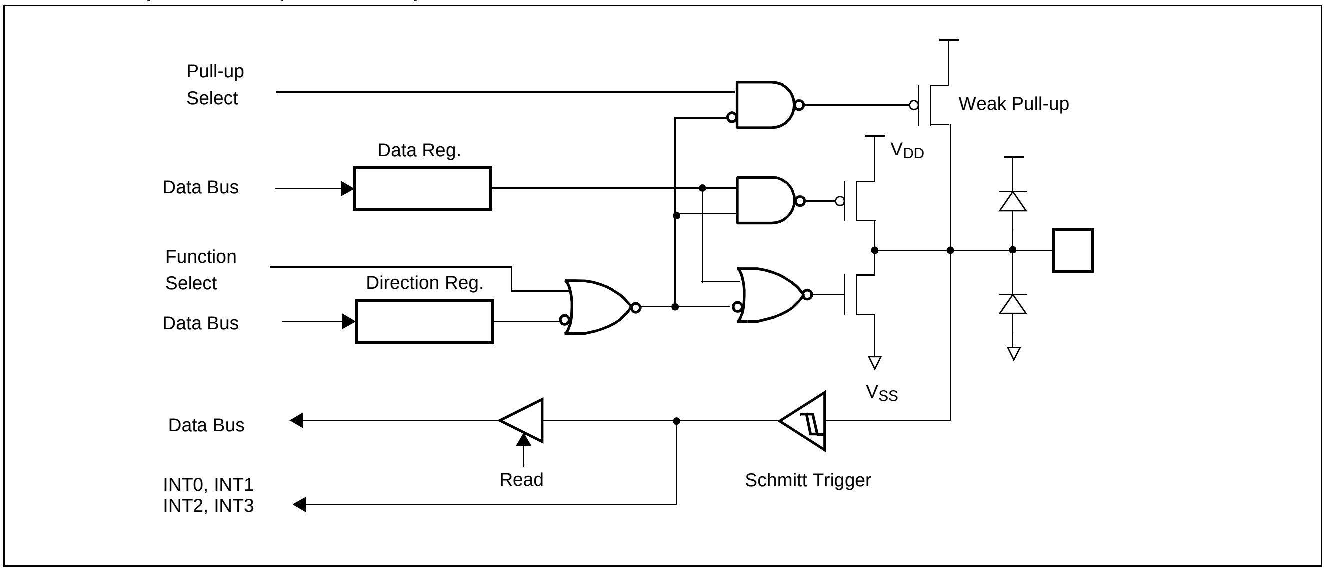
- **RB0 / AN0 / AVref**



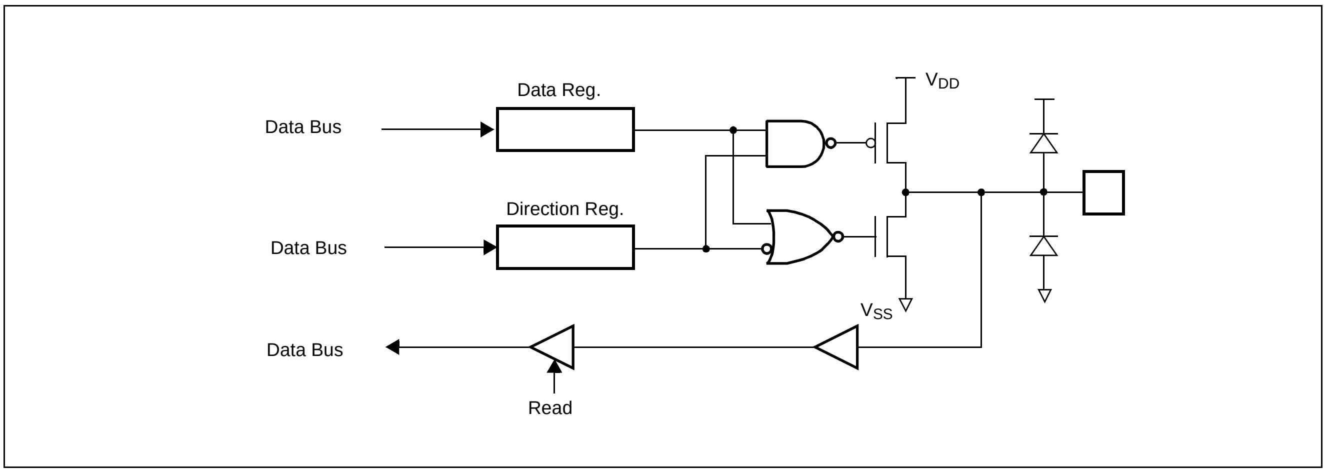
• **RB1/BUZ, RB4/PWM0/COMP0, RB5/PWM1/COMP1, RB7/TMR2OV, RC6/SOUT**



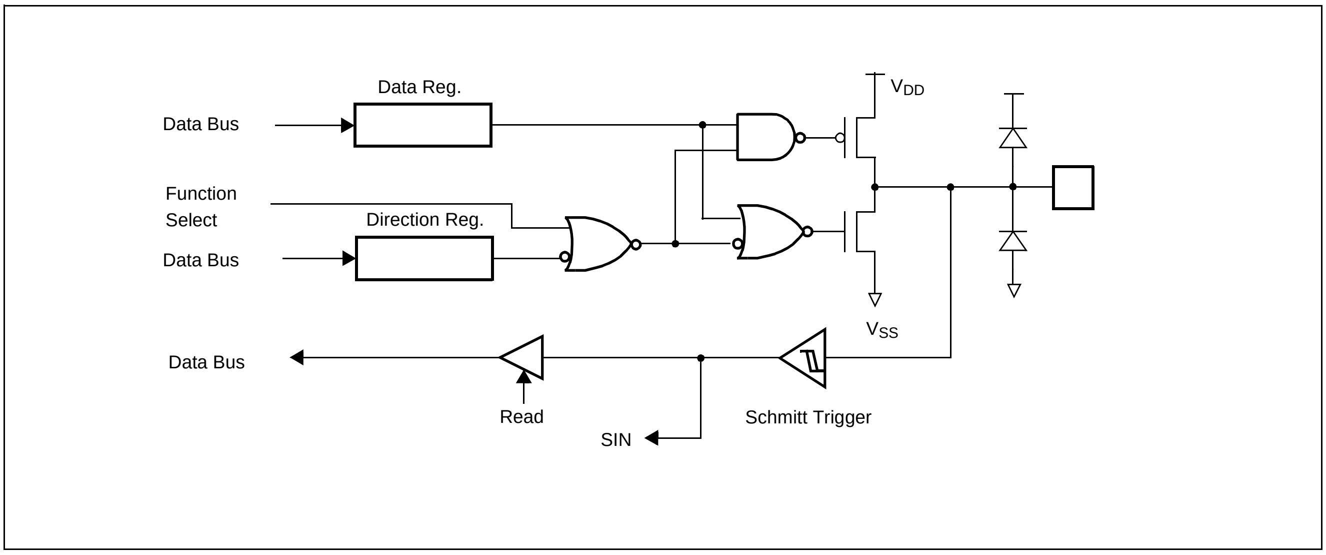
• **RB2/INT0, RB3/INT1, RD0/INT2, RD1/INT3**



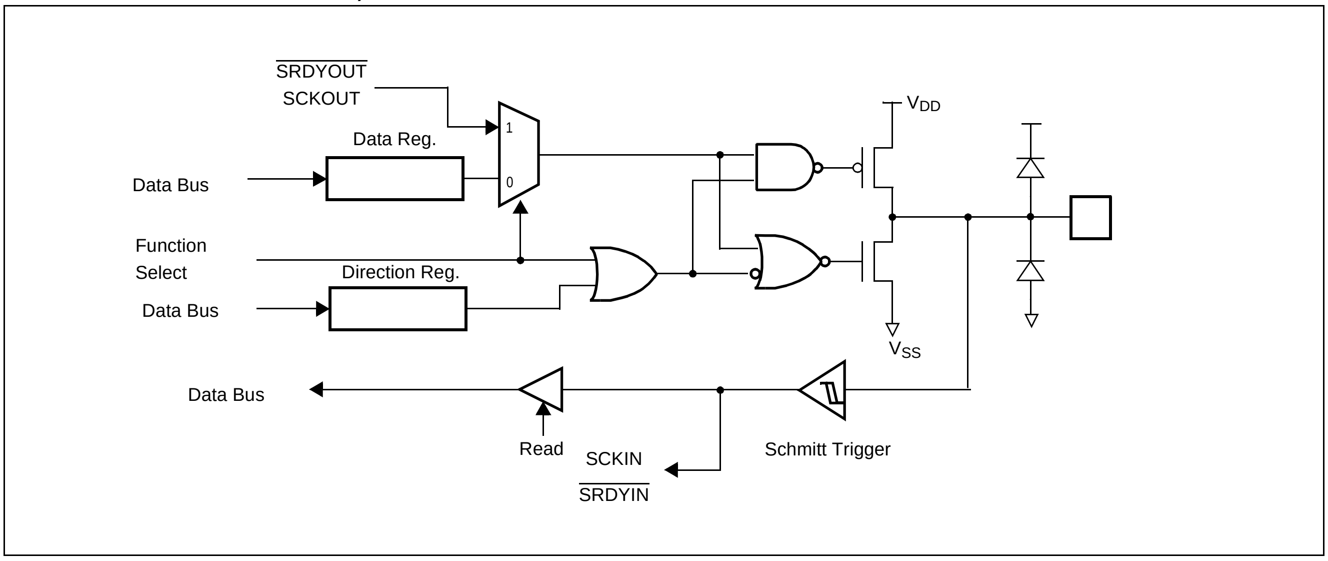
• **RD2**



- **RC5/SIN**



- **RC3 / $\overline{\text{SRDYIN}}$ / $\overline{\text{SRDYOUT}}$, RC4 / SCKIN / SCKOUT**



7. ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

Supply voltage .... -0.3 to +6.0 V

Storage Temperature ....-40 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 ....200 mA

Maximum current into V_{DD} pin ....150 mA

Maximum current sunk by (I_{OL} per I/O Pin) ....25 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}=8MHz	4.5	5.5	V
		f_{XIN}=4.2MHz	2.5	5.5	V
Operating Frequency	f_{XIN}	V_{DD}=4.5~5.5V	1	8	MHz
		V_{DD}=2.5~5.5V	1	4.2	kHz
Operating Temperature	T_{OPR}		-20	85	°C

7.3 A/D Converter Characteristics

(T_A=25°C, V_{SS}=0V, V_{DD}=5.12V @f_{XIN} =8MHz, V_{DD}=3.072V @f_{XIN} =4MHz)

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}	AVREFS=0	V_{SS}	-	V_{DD}	V
		AVREFS=1	V_{SS}	-	V_{REF}	
Analog Power Supply Input Voltage Range	V_{REF}	AVREFS=1	3	-	V_{DD}	V
Overall Accuracy	N_{ACC}		-	±1.0	±1.5	LSB
Non-Linearity Error	N_{NLE}		-	±1.0	±1.5	LSB
Differential Non-Linearity Error	N_{DNLE}		-	±1.0	±1.5	LSB
Zero Offset Error	N_{ZOE}		-	±0.5	±1.5	LSB
Full Scale Error	N_{FSE}		-	±0.25	±0.5	LSB
Gain Error	N_{NLE}		-	±1.0	±1.5	LSB
Conversion Time	T_{CONV}	f_{XIN}=8MHz	-	-	10	µS
		f_{XIN}=4MHz	-	-	20	
AV_{REF} Input Current	I_{REF}	AVREFS=1	-	0.5	1.0	mA

7.4 DC Electrical Characteristics

(T_A=-20~85°C, V_{DD}=2.5~5.5V, V_{SS}=0V),

Parameter	Symbol	Pin	Condition	Specifications			Unit
				Min.	Typ.	Max.	
Input High Voltage	V_{IH1}	X_{IN}, $\overline{RESET}$		0.9 V_{DD}	-	V_{DD}	V
	V_{IH2}	Hysteresis Input[1]		0.8 V_{DD}	-	V_{DD}	
	V_{IH3}	Normal Input		0.7 V_{DD}	-	V_{DD}	
Input Low Voltage	V_{IL1}	X_{IN}, $\overline{RESET}$		0	-	0.1 V_{DD}	V
	V_{IL2}	Hysteresis Input[1]		0	-	0.2 V_{DD}	
	V_{IL3}	Normal Input		0	-	0.3 V_{DD}	
Output High Voltage	V_{OH}	All Output Port	V_{DD}=5V, I_{OH}=-5mA	V_{DD} -1	-	-	V
Output Low Voltage	V_{OL}	All Output Port	V_{DD}=5V, I_{OL}=10mA	-	-	1	V
Input Pull-up Current	I_P	RB2, RB3, RD0, RD1	V_{DD}=5V	-550	-420	-200	μA
Input High Leakage Current	I_{IH1}	All Pins (except X_{IN})	V_{DD}=5V	-	-	5	μA
	I_{IH2}	X_{IN}	V_{DD}=5V	-	-	15	μA
Input Low Leakage Current	I_{IL1}	All Pins (except X_{IN})	V_{DD}=5V	-5	-	-	μA
	I_{IL2}	X_{IN}	V_{DD}=5V	-15	-	-	μA
Hysteresis	$\| V_T \|$	Hysteresis Input[1]	V_{DD}=5V	0.5	-	-	V
PFD Voltage	V_{PFD1}	V_{DD}	PFDM=0	2.5	3.0	3.5	V
	V_{PFD2}	V_{DD}	PFDM =1	2.0	2.5	3.0	
Internal RC WDT Period	T_{RCWDT}		V_{DD}=5V	40		120	μS
			V_{DD}=3V	95		280	
Operating Current	I_{DD}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz	-	5	6	mA
			V_{DD}=3.0V, f_{XIN}=4MHz	-	2	3	
Wake-up Timer Mode Current	I_{WKUP}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz	-	1	2	mA
			V_{DD}=3.0V, f_{XIN}=4MHz	-	0.5	1	
RCWDT Mode Current at STOP Mode	I_{RCWDT}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz	-	-	200	μA
			V_{DD}=3.0V, f_{XIN}=4MHz	-	-	100	
Stop Mode Current	I_{STOP}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz	-	0.5	3	μA
			V_{DD}=3.0V, f_{XIN}=4MHz	-	0.2	1	

1. Hysteresis Input: RA0, RB2, RB3, RB6, RC3, RC4, RC5, RD0, RD1

7.5 AC Characteristics

(T_A=-20~+85°C, V_{DD}=5V±10%, V_{SS}=0V)

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Operating Frequency	f_{CP}	X_{IN}	1	-	8	MHz
External Clock Pulse Width	t_{CPW}	X_{IN}	80	-	-	nS
External Clock Transition Time	t_{RCP},t_{FCP}	X_{IN}	-	-	20	nS
Oscillation Stabilizing Time	t_{ST}	X_{IN}, X_{OUT}	-	-	20	mS
External Input Pulse Width	t_{EPW}	INT0, INT1, INT2, INT3 EC0, EC1	2	-	-	t_{SYS}
External Input Pulse Transition Time	t_{REP},t_{FEP}	INT0, INT1, INT2, INT3 EC0, EC1	-	-	20	nS
$\overline{\text{RESET}}$ Input Width	t_{RST}	$\overline{\text{RESET}}$	8	-	-	t_{SYS}

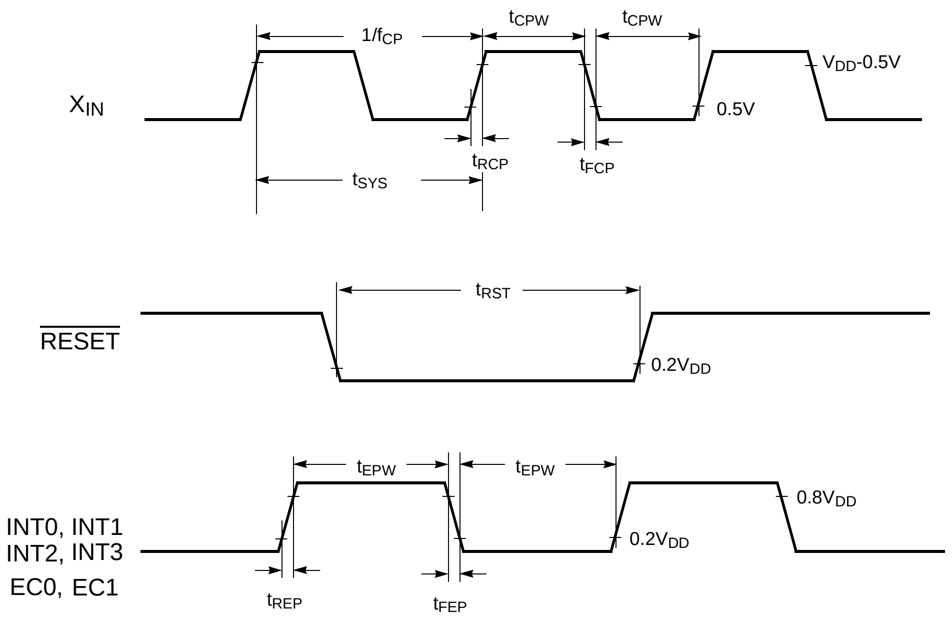


Figure 7-1 Timing Chart

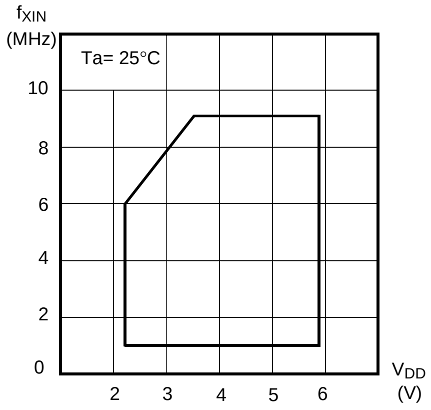
7.6 Typical Characteristics

This 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.

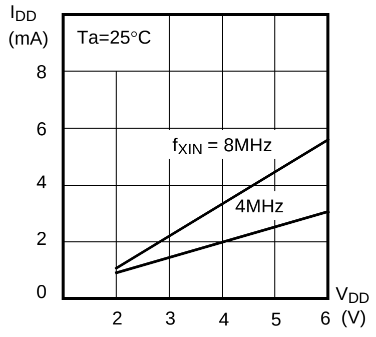
The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. “Typical” represents the mean of the distribution while “max” or “min” represents (mean + 3σ) and (mean – 3σ) respectively where σ is standard deviation

Operating Area



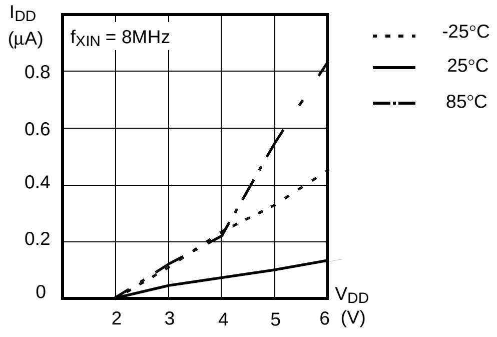
Normal Operation

I_{DD}–V_{DD}



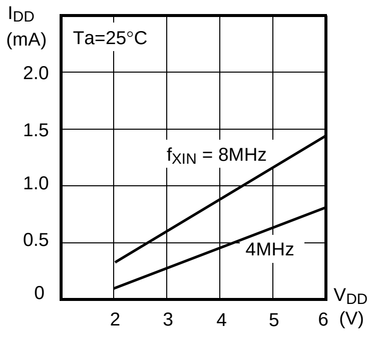
STOP Mode

I_{STOP}–V_{DD}



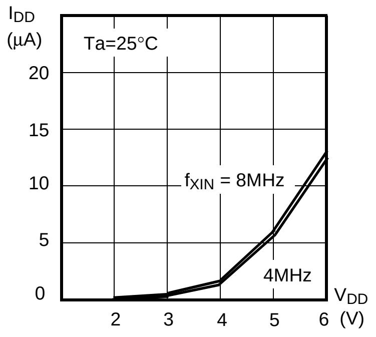
Wake-up Timer Mode

I_{WKUP}–V_{DD}

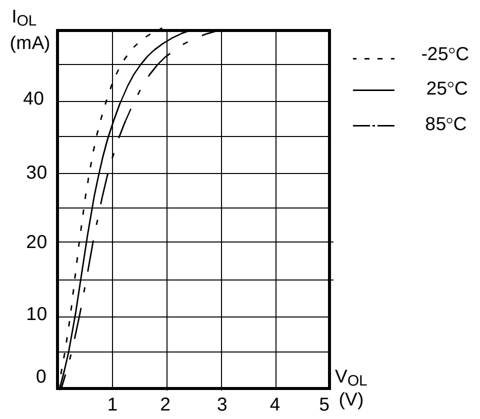


RC-WDT in Stop Mode

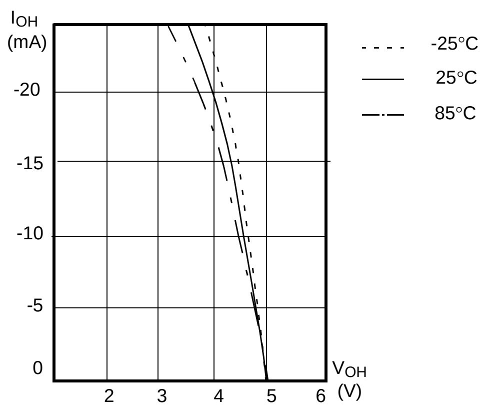
I_{RCWDT}–V_{DD}



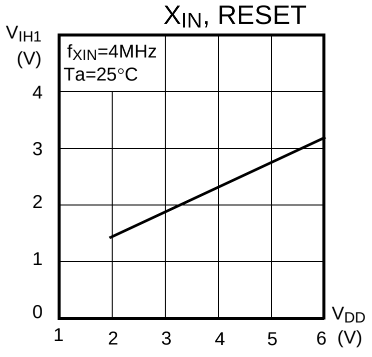
I_{OL}–V_{OL}, V_{DD}=5V



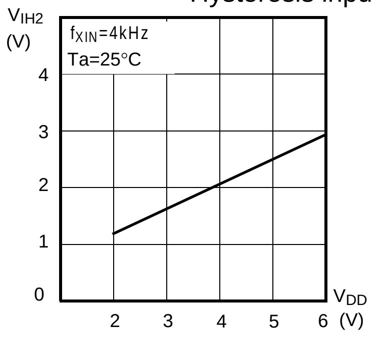
I_{OH}–V_{OH}, V_{DD}=5V



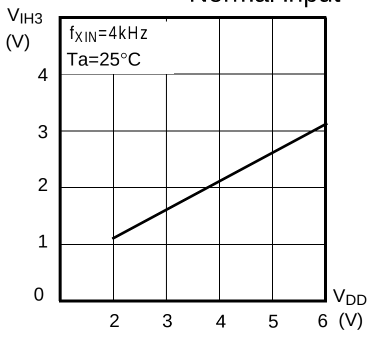
V_{DD}–V_{IH1} X_{IN}, $\overline{RESET}$



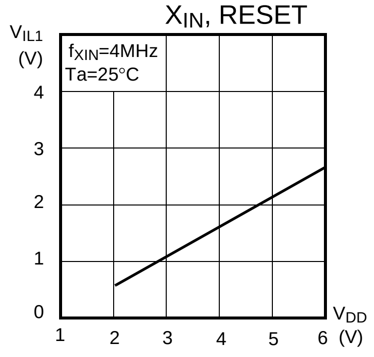
V_{DD}–V_{IH2} Hysteresis input



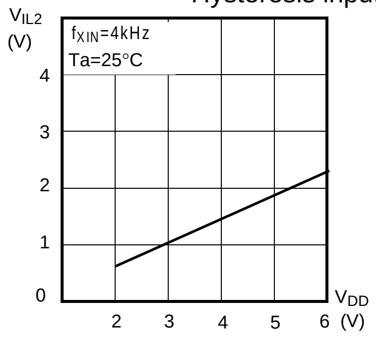
V_{DD}–V_{IH3} Normal input



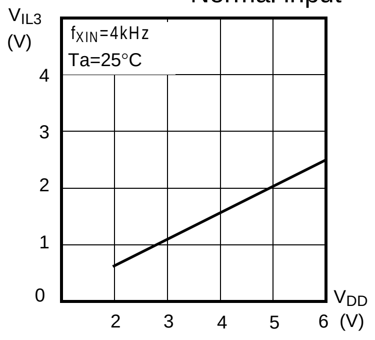
V_{DD}–V_{IL1} X_{IN}, $\overline{RESET}$



V_{DD}–V_{IL2} Hysteresis input



V_{DD}–V_{IL3} Normal input



8. MEMORY ORGANIZATION

The GMS87C1404 and GMS87C1408 have separate address spaces for Program memory and Data Memory. Program memory can only be read, not written to. It can be up to 4K /8K bytes of Program memory. Data memory can be read and written to up to 192 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.

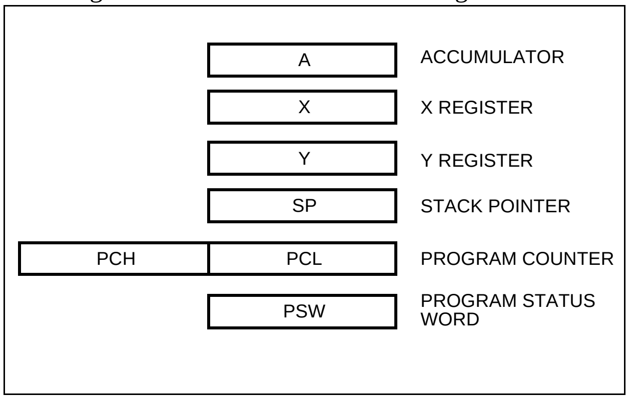


Figure 8-1 Configuration of Registers

Accumulator: The Accumulator is the 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.

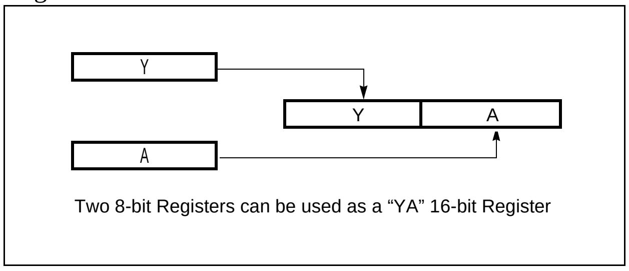


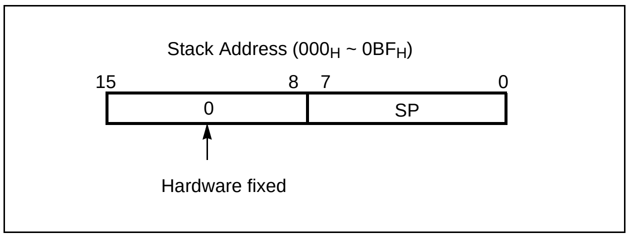
Figure 8-2 Configuration of YA 16-bit Register

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 00_H to BF_H 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 "BF_H" is used.



Note: *The Stack Pointer must be initialized by software because its value is undefined after RESET.*
Example: To initialize the SP
LDX #0BFH
TXSP ; SP ← BF_H

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 (PC_H:$0FF_H$, PC_L:$0FE_H$).

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.

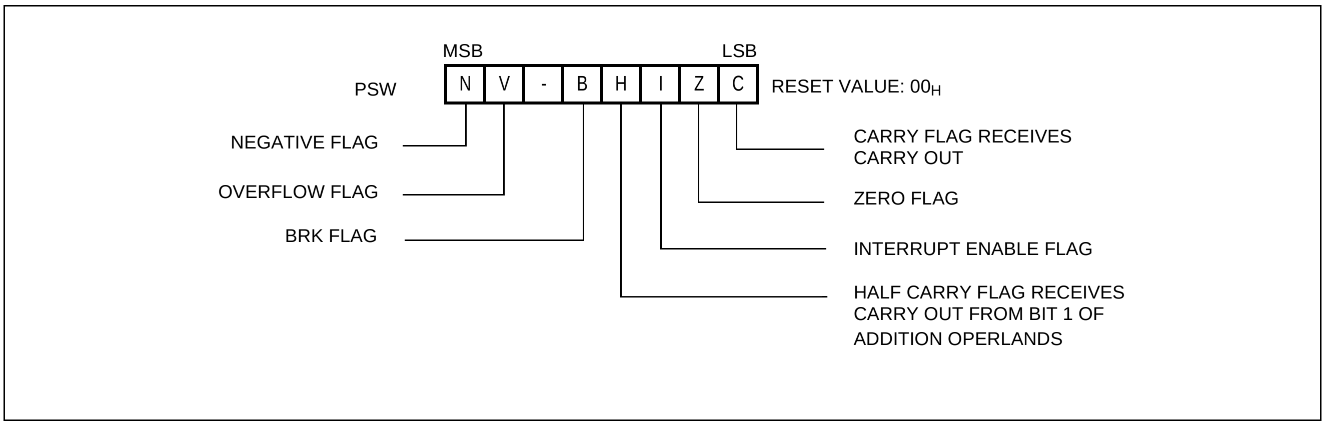


Figure 8-3 PSW (Program Status Word) Register

[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.

[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(7F$_H$) or -128(80$_H$). 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/8K 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.

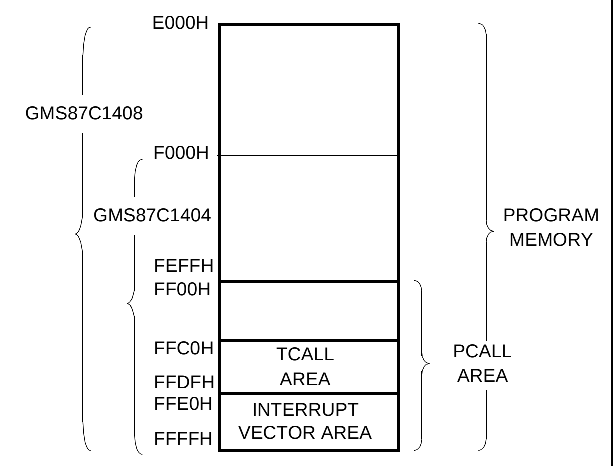


Figure 8-4 Program Memory Map

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.

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 .

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     ②    ①
         RET
;
;TABLE CALL ADD. AREA
;
         ORG    0FFC0H         ;TCALL ADDRESS AREA
         DW     FUNC_A
         DW     FUNC_B
```

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 External Interrupt 1, $0FFFA_H$ 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.

Address	Vector Area Memory
$0FFE0_H$	-
E2	-
E4	Serial Peripheral Interface Interrupt Vector Area
E6	Basic Interval Interrupt Vector Area
E8	Watchdog Timer Interrupt Vector Area
EA	A/D Converter Interrupt Vector Area
EC	Timer/Counter 3 Interrupt Vector Area
EE	Timer/Counter 2 Interrupt Vector Area
F0	External Interrupt 3 Vector Area
F2	External Interrupt 2 Vector Area
F4	Timer/Counter 1 Interrupt Vector Area
F6	Timer/Counter 0 Interrupt Vector Area
F8	External Interrupt 1 Vector Area
FA	External Interrupt 0 Vector Area
FC	-
FE	RESET Vector Area

NOTE:
"-" means reserved area.

Figure 8-5 Interrupt Vector Area

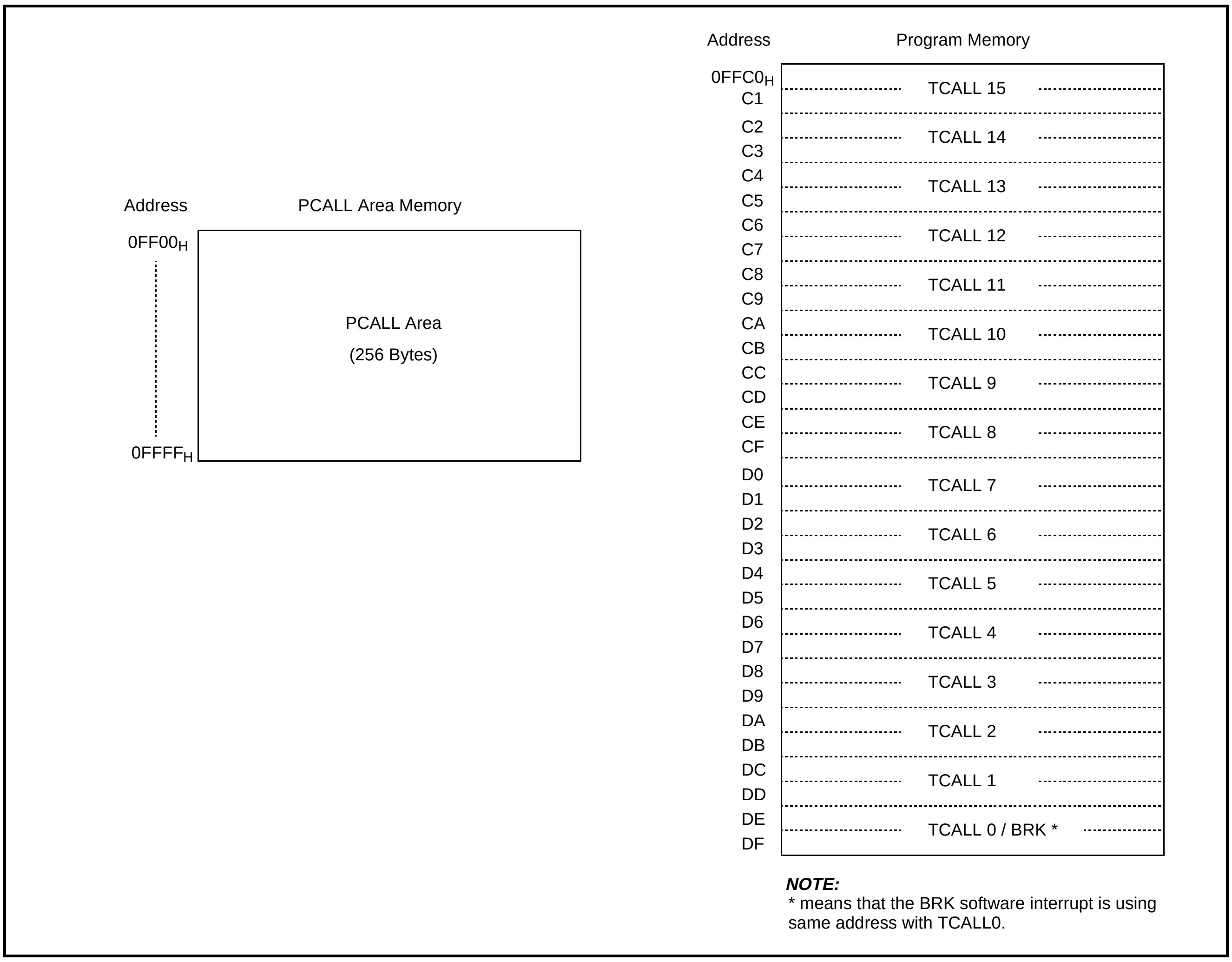
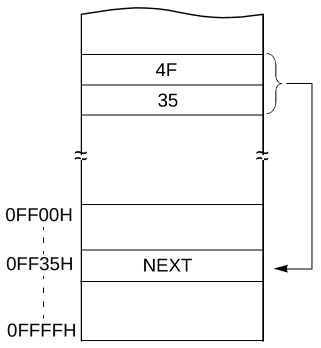


Figure 8-6 PCALL and TCALL Memory Area

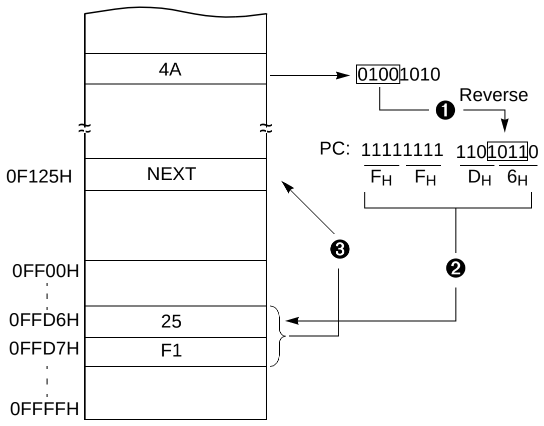
PCALL→ rel

```
4F35     PCALL  35H
```



TCALL→ n

```
4A       TCALL  4
```



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

```
        ORG     0FFE0H

        DW      NOT_USED            ; (0FFEO)
        DW      NOT_USED            ; (0FFE2)
        DW      SPI_INT             ; (0FFE4) Serial Peripheral Interface
        DW      BIT_INT             ; (0FFE6) Basic Interval Timer
        DW      WDT_INT             ; (0FFE8) Watchdog Timer
        DW      AD_INT              ; (0FFEA) A/D
        DW      TMR3_INT            ; (0FFEC) Timer-3
        DW      TMR2_INT            ; (0FFEE) Timer-2
        DW      INT3                ; (0FFF0) Int.3
        DW      INT2                ; (0FFF2) Int.2
        DW      TMR1_INT            ; (0FFF4) Timer-1
        DW      TMR0_INT            ; (0FFF6) Timer-0
        DW      INT1                ; (0FFF8) Int.1
        DW      INT0                ; (0FFFA) Int.0
        DW      NOT_USED            ; (0FFFC)
        DW      RESET               ; (0FFFE) Reset


        ORG     0F000H

;*********************************************
;           MAIN     PROGRAM        *
;********************************************
;
RESET:  DI                          ;Disable All Interrupts
        LDX     #0
RAM_CLR:LDA     #0                  ;RAM Clear(!0000H->!00BFH)
        STA     {X}+
        CMPX    #0C0H
        BNE     RAM_CLR
;
        LDX     #0BFH               ;Stack Pointer Initialize
        TXSP
;
        CALL    INITIAL             ;
;
        LDM     RA, #0              ;Normal Port A
        LDM     RAIO,#1000_0010B    ;Normal Port Direction
        LDM     RB, #0              ;Normal Port B
        LDM     RBIO,#1000_0010B    ;Normal Port Direction
        :
        :
        LDM     PFDR,#0             ;Enable Power Fail Detector
        :
        :
```

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.

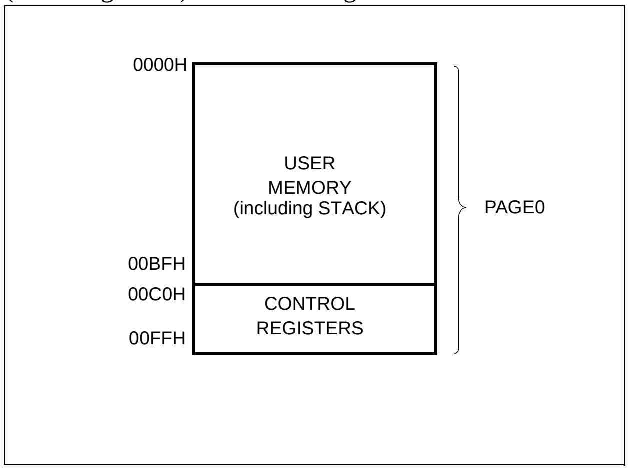


Figure 8-7 Data Memory Map

User Memory

The GMS87C1404 and GMS87C1408 has 192 $\times$ 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 $0C0_H$ to $0FF_H$.

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 CKCTLR

```
          LDM    CKCTLR,#09H ;Divide ratio ÷16
```

Address	Symbol	R/W	RESET Value	Addressing mode
0C0H	RA	R/W	Undefined	byte, bit[1]
0C1H	RAIO	R/W	0000_0000	byte[2]
0C2H	RB	R/W	Undefined	byte, bit
0C3H	RBIO	R/W	00000000	byte
0C4H	RC	R/W	Undefined	byte, bit
0C5H	RCIO	R/W	-000_0---	byte
0C6H	RD	R/W	Undefined	byte, bit
0C7H	RDIO	W	----_-000	byte
0CAH	RAFUNC	W	0000_0000	byte
0CBH	RBFUNC	W	0000_0000	byte
0CCH	PUPSEL	W	----_0000	byte
0CDH	RDFUNC	W	----_--00	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	PWM0HR	W	----_0000	byte
0D6H	TM2	R/W	--00_0000	byte, bit
0D7H	T2	R	0000_0000	byte
0D7H	TDR2	W	1111_1111	byte
0D7H	CDR2	R	0000_0000	byte
0D8H	TM3	R/W	0000_0000	byte, bit
0D9H	TDR3	W	1111_1111	byte
0D9H	T3PPR	W	1111_1111	byte
0DAH	T3	R	0000_0000	byte
0DAH	CDR3	R	0000_0000	byte
0DAH	T3PDR	R/W	0000_0000	byte, bit
0DBH	PWM1HR	W	----_0000	byte
0DEH	BUR	W	1111_1111	byte
0E0H	SIOM	R/W	0000_0001	byte, bit
0E1H	SIOR	R/W	Undefined	byte, bit
0E2H	IENH	R/W	0000_0000	byte, bit
0E3H	IENL	R/W	0000_----	byte, bit
0E4H	IRQH	R/W	0000_0000	byte, bit
0E5H	IRQL	R/W	0000_----	byte, bit
0E6H	IEDS	R/W	0000_0000	byte, bit
0EAH	ADCM	R/W	--00_0001	byte, bit
0EBH	ADCR	R	Undefined	byte
0ECH	BITR	R	0000_0000	byte
0ECH	CKCTLR	W	-001_0111	byte
0EDH	WDTR	R	0000_0000	byte
0EDH	WDTR	W	0111_1111	byte
0EFH	PFDR	R/W	----_-100	byte, bit

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
D7H	T2	CDR2	-	TDR2	-
D9H	-			TDR3	T3PPR
DAH	T3	CDR3	T3PDR	-	T3PDR
ECH	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	RA	RA Port Data Register							
C1H	RAIO	RA Port Direction Register							
C2H	RB	RB Port Data Register							
C3H	RBIO	RB Port Direction Register							
C4H	RC	RC Port Data Register							
C5H	RCIO	RC Port Direction Register							
C6H	RD	RD Port Data Register							
C7H	RDIO	RD Port Direction Register							
CAH	RAFUNC	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0
CBH	RBFUNC	TMR2OV	EC1I	PWM1O	PWM0O	INT1I	INT0I	BUZO	AVREFS
CCH	PUPSEL	-	-	-	-	PUPSEL3	PUPSEL2	PUPSEL1	PUPSEL0
CDH	RDFUNC	-	-	-	-	-	-	INT3I	INT2I
D0H	TM0	-	-	CAP0	T0CK2	T0CK1	T0CK0	T0CN	T0ST
D1H	T0/TDR0/ CDR0	Timer0 Register / Timer0 Data Register / Capture0 Data Register							
D2H	TM1	POL	16BIT	PWM0E	CAP1	T1CK1	T1CK0	T1CN	T1ST
D3H	TDR1/ T1PPR	Timer1 Data Register / PWM0 Period Register							
D4H	T1/CDR1/ T1PDR	Timer1 Register / Capture1 Data Register / PWM0 Duty Register							
D5H	PWM0HR	PWM0 High Register							
D6H	TM2	-	-	CAP2	T2CK2	T2CK1	T2CK0	T2CN	T2ST
D7H	T2/TDR2/ CDR2	Timer2 Register / Timer2 Data Register / Capture2 Data Register							
D8H	TM3	POL	16BIT	PWM1E	CAP3	T3CK1	T3CK0	T3CN	T3ST
D9H	TDR3/ T3PPR	Timer3 Data Register / PWM1 Period Register							
DAH	T3/CDR3/ T3PDR	Timer3 Register / Capture3 Data Register / PWM1Duty Register							
DBH	PWM1HR	PWM1 High Register							
DEH	BUR	BUCK1	BUCK0	BUR5	BUR4	BUR3	BUR2	BUR1	BUR0
E0H	SIOM	POL	SRDY	SM1	SM0	SCK1	SCK0	SIOST	SIOSF
E1H	SIOR	SPI DATA REGISTER							
E2H	IENH	INT0E	INT1E	T0E	T1E	INT2E	INT3E	T2E	T3E
E3H	IENL	ADE	WDTE	BITE	SPIE	-	-	-	-
E4H	IRQH	INT0IF	INT1IF	T0IF	T1IF	INT2IF	INT3IF	T2IF	T3IF
E5H	IRQL	ADIF	WDTIF	BITIF	SPIF	-	-	-	-
E6H	IEDS	IED3H	IED3L	IED2H	IED2L	IED1H	IED1L	IED0H	IED0L

Table 8-3 Control Registers of GMS87C1408 and GMS87C1404

▨	*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”.*

EAH	ADCM	-	-	ADEN	ADS2	ADS1	ADS0	ADST	ADSF
EBH	ADCR	ADC Result Data Register							
ECH	BITR[1]	Basic Interval Timer Data Register							
ECH	CKCTLR[1]	-	WAKEUP	RCWDT	WDTON	BTCL	BTS2	BTS1	BTS0
EDH	WDTR	WDTCL	7-bit Watchdog Counter Register						
EFH	PFDR[2]	-	-	-	-	-	PFDIS	PFDM	PFDS

Table 8-3 Control Registers of GMS87C1408 and GMS87C1404

▨ *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 ECH 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 GMS87C1404 and GMS87C1408 uses six addressing modes;

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

(1) Register Addressing

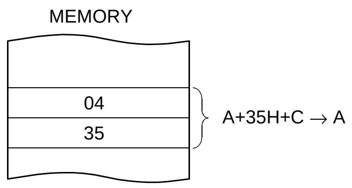
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

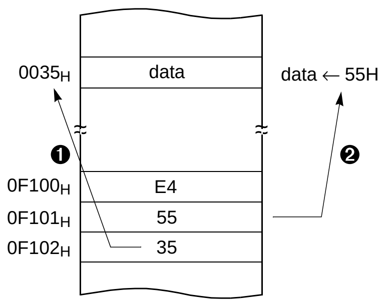
In this mode, second byte (operand) is accessed as a data immediately.

Example:

```
0435     ADC    #35H
```



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

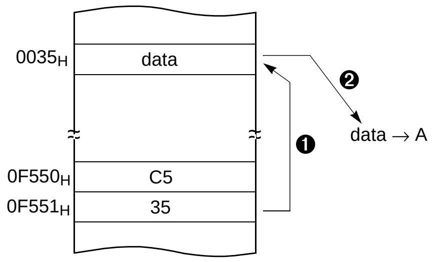


(3) Direct Page Addressing → dp

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

Example;

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



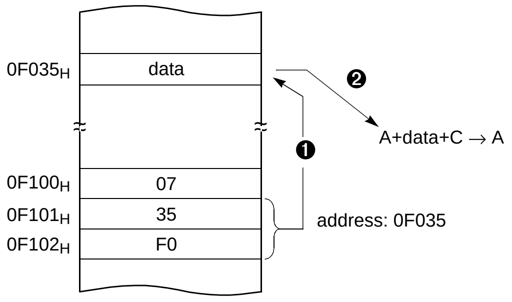
(4) Absolute Addressing → !abs

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;

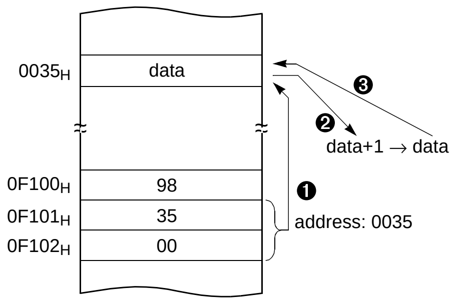
```
0735F0   ADC    !0F035H     ;A ←ROM[0F035H]
```



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

Example; Addressing accesses the address 0135_H.

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



(5) Indexed Addressing

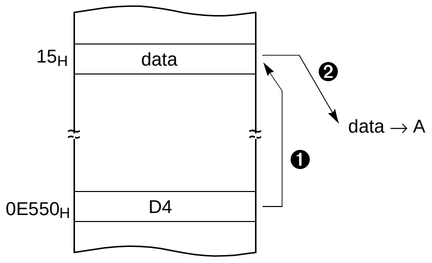
X indexed direct page (no offset) → {X}

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

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

Example; $X=15_H$

```
D4       LDA    {X}         ;ACC←RAM[X].
```



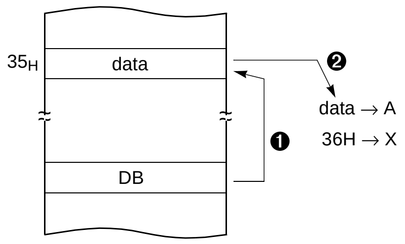
X indexed direct page, auto increment→ {X}+

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; $X=35_H$

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



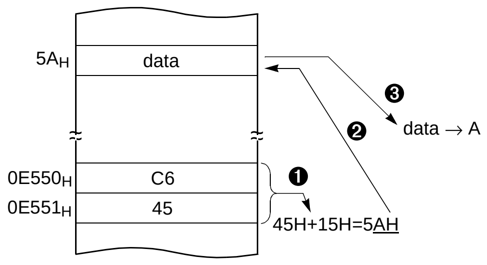
X indexed direct page (8 bit offset) → dp+X

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; $X=015_H$

```
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.

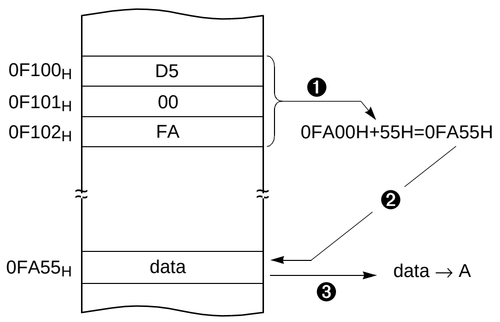
This is same with above (2). Use Y register instead of X.

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=55_H

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



(6) Indirect Addressing

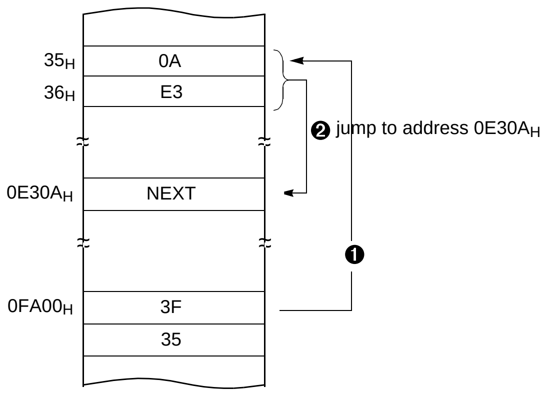
Direct page indirect → [dp]

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;

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



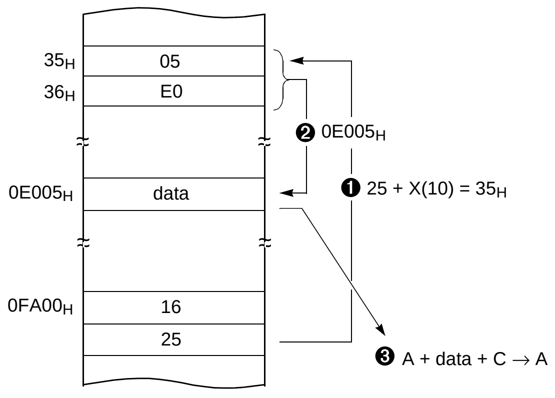
X indexed indirect → [dp+X]

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=10_H

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



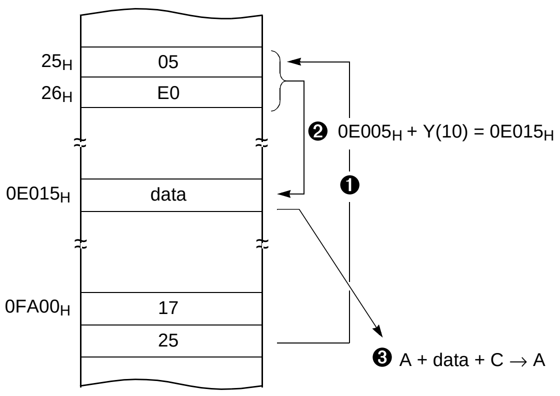
Y indexed indirect → [dp]+Y

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=10_H

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



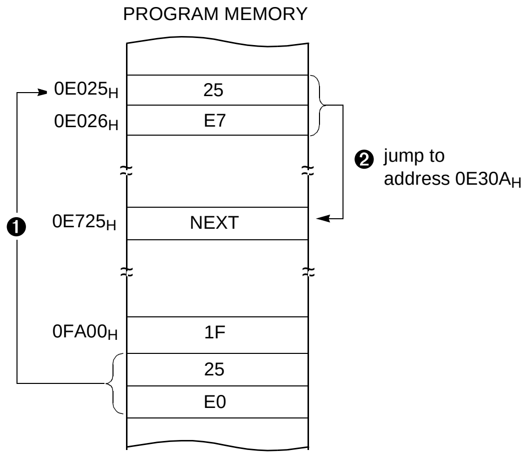
Absolute indirect → [!abs]

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

JMP

Example;

```
1F25E0   JMP    [!0C025H]
```



9. I/O PORTS

The GMS87C1408 and GMS87C1404 has four ports, RA, RB, RC and RD. These ports 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. A “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 RA as output ports and the odd numbered bits as input ports, write “55_H” to address $C1_H$ (RA 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.

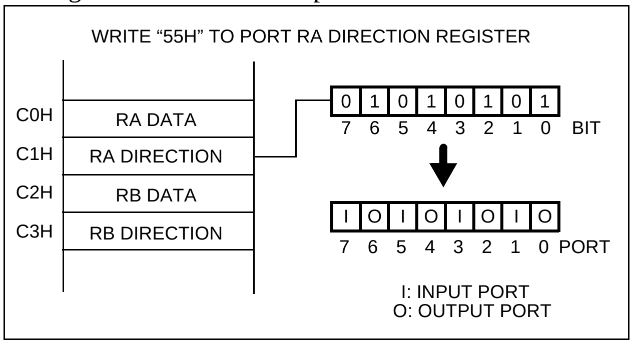


Figure 9-1 Example of port I/O assignment

9.1 RA and RAIO registers

RA is an 8-bit bidirectional I/O port (address $C0_H$). Each port can be set individually as input and output through the RAIO register (address $C1_H$).

RA7~RA1 ports are multiplexed with Analog Input Port (AN7~AN1) and RA0 port is multiplexed with Event Counter Input Port (EC0).

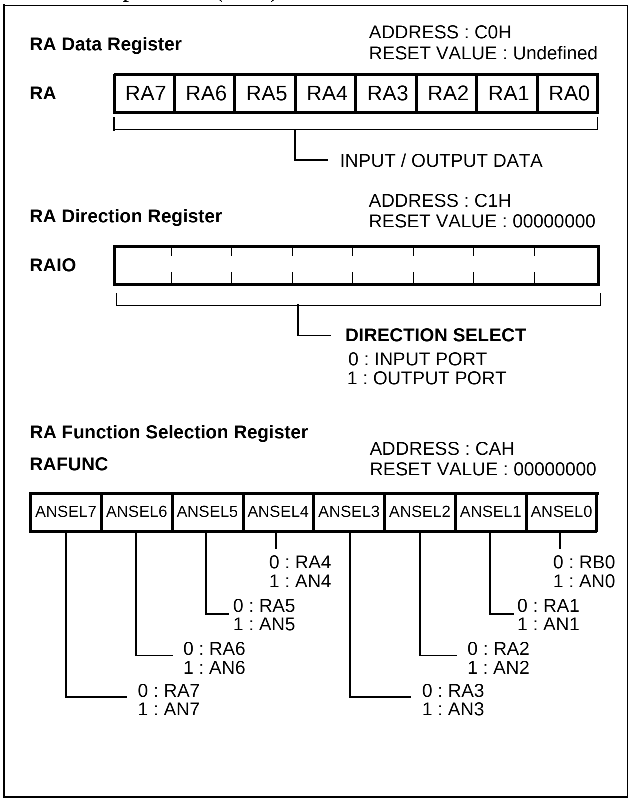


Figure 9-2 Registers of Port RA

The control register RAFUNC (address CA_H) 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 RAFUNC.Regardless of the direction register RAIO, RAFUNC is selected to use as alternate functions, port pin can be used as a corresponding alternate features (RA0/EC0 is controlled by RBFUNC)

PORT	RAFUNC.7~0	Description
RA7/AN7	0	RA7 (Normal I/O Port)
	1	AN7 (ADS2~0=111)
RA6/AN6	0	RA6 (Normal I/O Port)
	1	AN6 (ADS2~0=110)
RA5/AN5	0	RA5 (Normal I/O Port)
	1	AN5 (ADS2~0=101)
RA4/AN4	0	RA4 (Normal I/O Port)
	1	AN4 (ADS2~0=100)
RA3/AN3	0	RA3 (Normal I/O Port)
	1	AN3 (ADS2~0=011)
RA2/AN2	0	RA2 (Normal I/O Port)
	1	AN2 (ADS2~0=010)
RA1/AN1	0	RA1 (Normal I/O Port)
	1	AN1 (ADS2~0=001)
RA0/EC0[1]		RA0 (Normal I/O Port)
		EC0 (T0CK2~0=111)

1. This port is not an Analog Input port, but Event Counter clock source input port. ECO is controlled by setting TOCK2~0 = 111. The bit RAFUNC.0 (ANSEL0) controls the RB0/AN0/AVref port (Refer to Port RB).

9.2 RB and RBIO registers

RB is a 5-bit bidirectional I/O port (address $C2_H$). Each pin can be set individually as input and output through the RBIO register (address $C3_H$). In addition, Port RB is multiplexed with various special features. The control register RBFUNC (address CB_H) 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 External interrupt or Timer compare output, write "1" to the corresponding bit of RBFUNC.

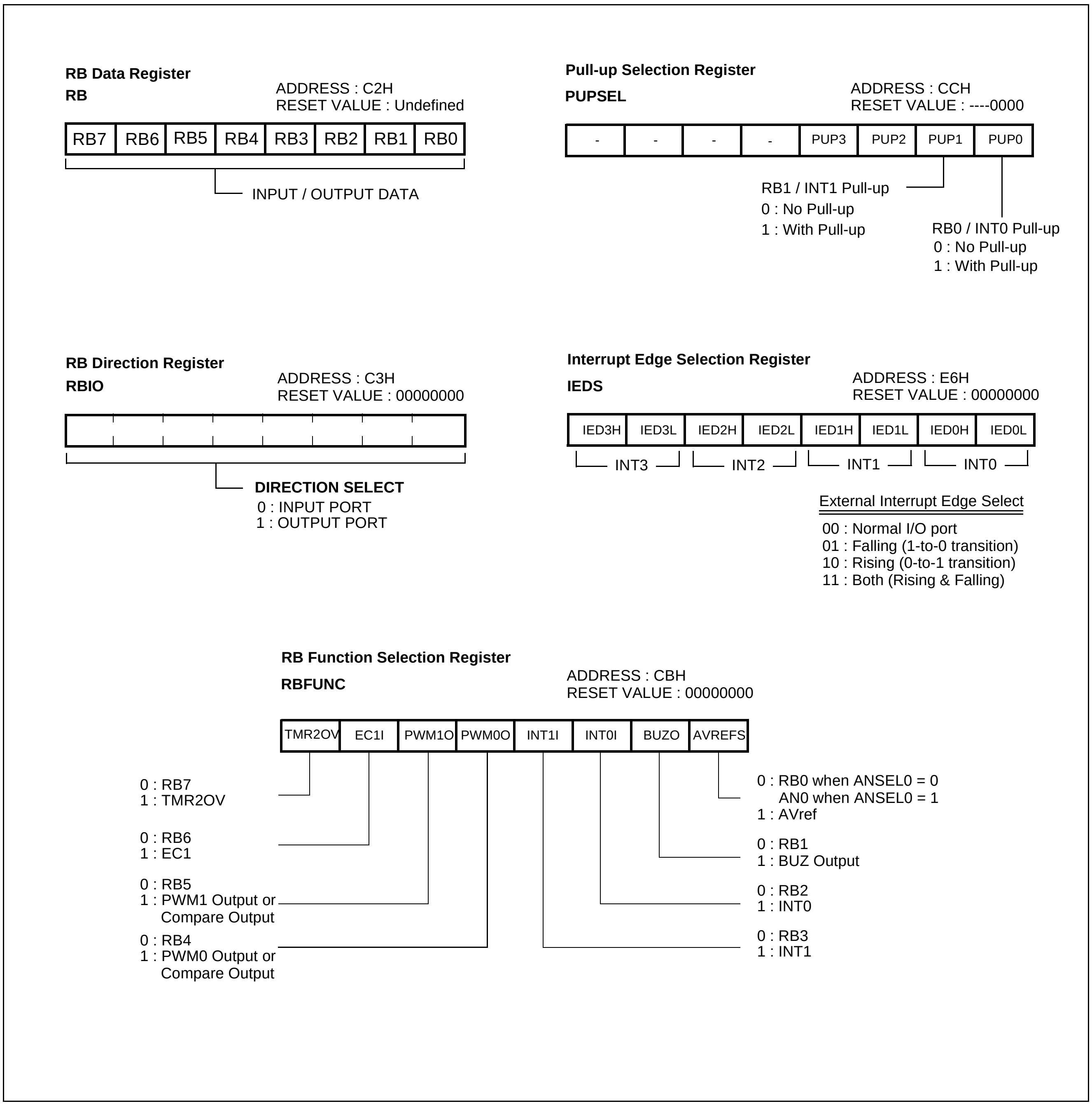


Figure 9-3 Registers of Port RB

Regardless of the direction register RBIO, RBFUNC is selected to use as alternate functions, port pin can be used as a corresponding alternate features.

PORT	RBFUNC.4~0	Description
RB7/ TMR2OV	0	RB7 (Normal I/O Port)
	1	Timer2 Overflow Output
RB6/EC1	0	RB6 (Normal I/O Port)
	1	Event Counter 1 Input
RB5/ PWM1/ COMP1	0	RB5 (Normal I/O Port)
	1	PWM1 Output / Timer3 Compare Output
RB4/ PWM0/ COMP0	0	RB4 (Normal I/O Port)
	1	PWM0 Output / Timer1 Compare Output
RB3/INT1	0	RB3 (Normal I/O Port)
	1	External Interrupt Input 1
RB2/INT0	0	RB2 (Normal I/O Port)
	1	External Interrupt Input 0
RB1/BUZ	0	RB1 (Normal I/O Port)
	1	Buzzer Output
RB0/AN0/ AVref	0[1]	RB0 (Normal I/O Port)/ AN0 (ANSEL0=1)
	1[2]	External Analog Reference Voltage

1. When ANSEL0 = “0”, this port is defined for normal I/O port (RB0).
When ANSEL0 = “1” and ADS2~0 = “000”, this port can be used Analog Input Port (AN0).
2. When this bit set to “1”, this port defined for AVref, so it can not be used Analog Input Port AN0 and Normal I/O Port RB0.

9.3 RC and RCIO registers

RC is an 4-bit bidirectional I/O port (address $C4_H$). Each pin can be set individually as input and output through the RCIO register (address $C5_H$).

In addition, Port RC is multiplexed with Serial Peripheral Interface (SPI).

The control register SIOM (address $E0_H$) controls to select Serial Peripheral Interface function.

After reset, the RCIO register value is “0”, port may be used as general I/O ports. To select Serial Peripheral Interface function, write “1” to the corresponding bit of SIOM.

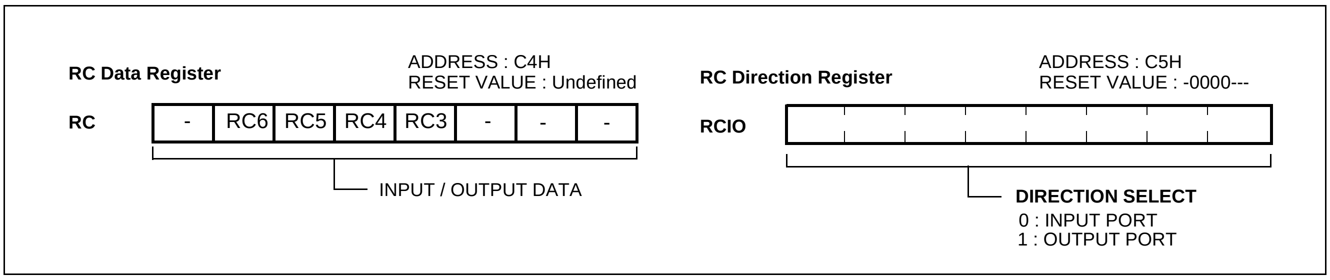


Figure 9-4 Registers of Port RC

PORT	Function	SIOM			Description
		SRDY	SM [1:0]	SCK [1:0]	
RC6/ SOUT	RC6	X	X:0	X:X	RC6 (Normal I/O Port)
	SOUT	X	X:1	X:X	SPI Serial Data Output
RC5/ SIN	RC5	X	0:X	X:X	RC5 (Normal I/O Port)
	SIN	X	1:X	X:X	SPI Serial Data Input
RC4/ SCK	RC4	X	0:0	X:X	RC4 (Normal I/O Port)
	SCKO	X	0:0	00, 01, 10	SPI Synchronous Clock Output
	SCKI	X	0:0	1:1	SPI Synchronous Clock Input
RC3/ $\overline{\text{SRDY}}$	RC3	0	X:X	X:X	RC3 (Normal I/O Port)
	$\overline{\text{SRDYIN}}$	1	X:X	00, 01, 10	SPI Ready Input (Master Mode)
	$\overline{\text{SRDYOUT}}$	1	X:X	1:1	SPI Ready Output (Slave Mode)

Table 9-1 Serial Communication Functions in RC Port

9.4 RD and RDIO registers

RD is a 3-bit bidirectional I/O port (address $C6_H$). Each pin can be set individually as input and output through the RDIO register (address $C7_H$).

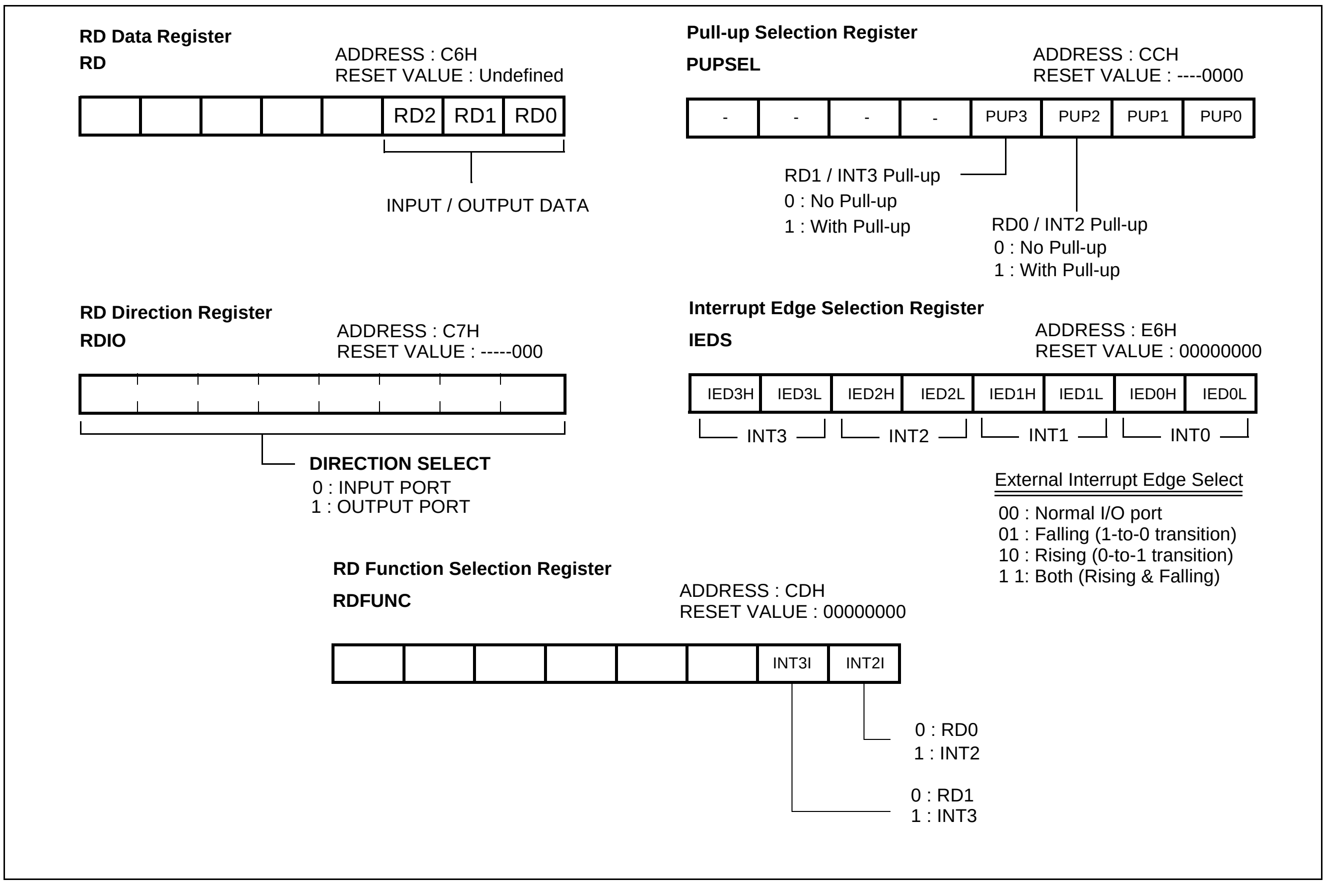


Figure 9-5 Registers of Port RD

In addition, Port RD is multiplexed with external interrupt input function. The control register RDFUNC (address CD_H) controls to select alternate function. After reset, this value is “0”, port may be used as general I/O ports. To select alternate function, write “1” to the corresponding bit of RDFUNC.

Regardless of the direction register RDIO, RDFUNC is selected to use as external interrupt input function, port pin can be used as a interrupt input feature.

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 Xin and Xout pins. External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the Xin pin and open the Xout pin.

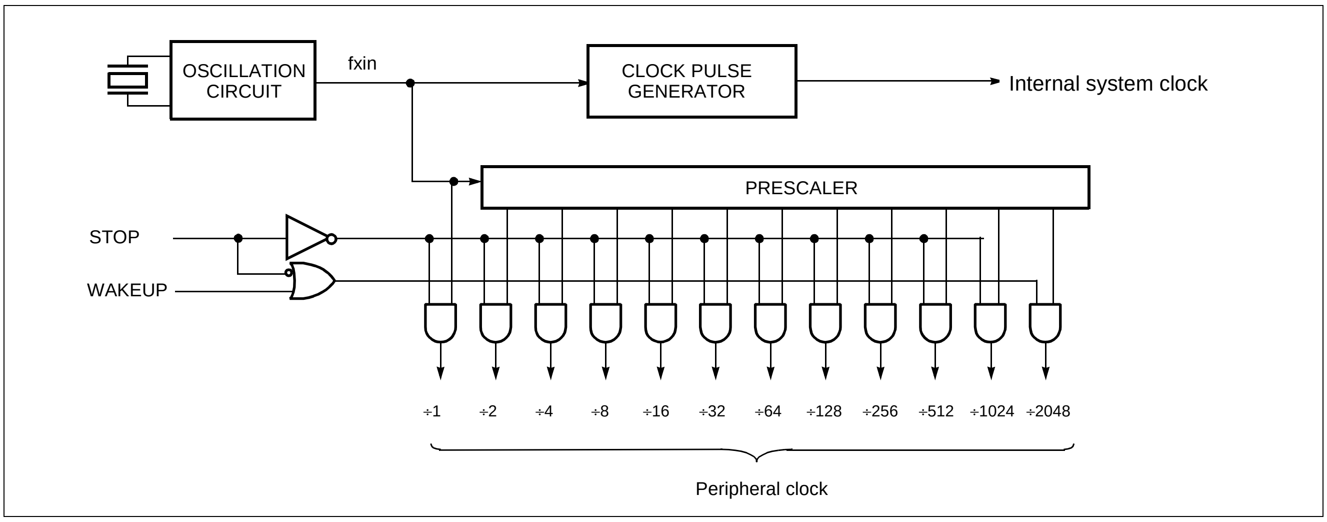
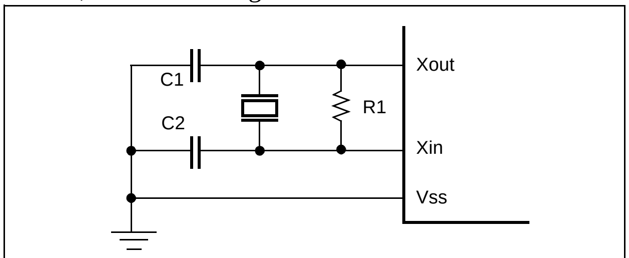


Figure 10-1 Block Diagram of Clock Pulse Generator

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 .



Recommended: C1, C2 = 30pF±10pF for Crystals
R1 = 1MΩ

Figure 10-2 Oscillator Connections

To drive the device from an external clock source, Xout should be left unconnected while Xin 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.

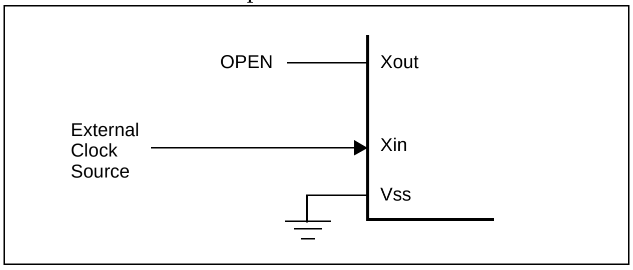


Figure 10-3 External Clock Connections

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 capacities.*
- 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 Vss. Do not ground to any ground pattern where high current is present.
- Do not fetch signals from the oscillator.

11. Basic Interval Timer

The GMS87C1408 and GMS87C1404 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 BITF 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 WAKEUP of CKCTLR, it goes into the wake-up timer mode. In this mode, all of the block is halted except the oscillator, prescaler (only fxin÷2048) and Timer0.

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 EC_H). Address EC_H is read as BITR, written to CKCTLR. Therefore, the CKCTLR can not be accessed by bit manipulation instruction.*

.

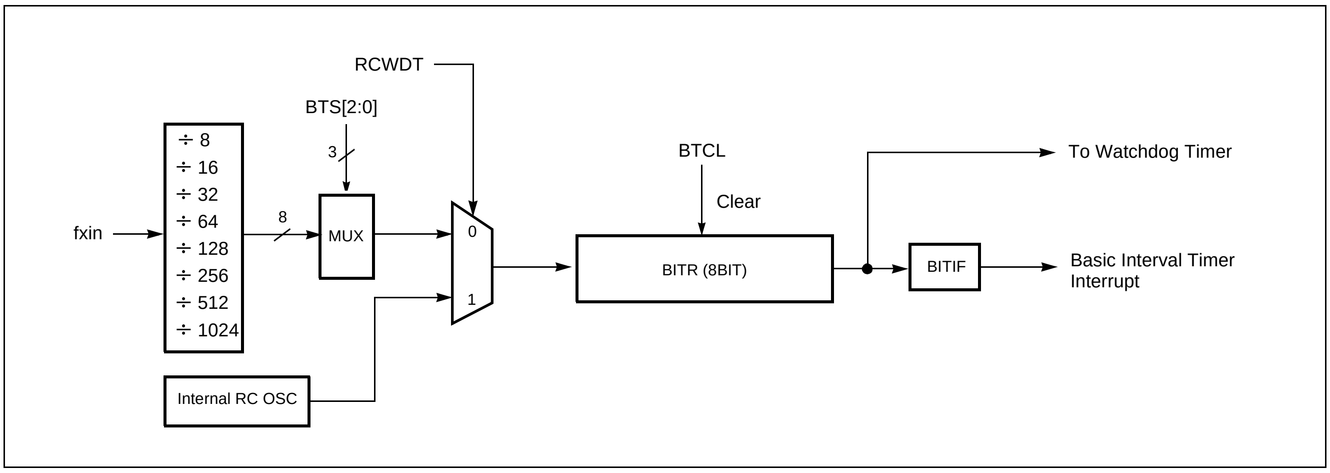


Figure 11-1 Block Diagram of Basic Interval Timer

Clock Control Register

CKCTLR	-	WAKEUP	RCWDT	WDTON	BTCL	BTS2	BTS1	BTS0

ADDRESS : ECH
RESET VALUE : -0010111
Bit Manipulation Not Available

Basic Interval Timer Clock Selection

000 : fxin ÷ 8
001 : fxin ÷ 16
010 : fxin ÷ 32
011 : fxin ÷ 64
100 : fxin ÷ 128
101 : fxin ÷ 256
110 : fxin ÷ 512
111 : fxin ÷ 1024

Symbol	Function Description
WAKEUP	1 : Enables Wake-up Timer 0 : Disables Wake-up Timer
RCWDT	1 : Enables Internal RC Watchdog Timer 0 : Disables Internal RC Watchdog Time
WDTON	1 : Enables Watchdog Timer 0 : Operates as a 7-bit Timer
BTCL	1 : BITR is cleared and BTCL becomes “0” automatically after one machine cycle, and BITR continue to count-up

Figure 11-2 CKCTLR: Clock Control Register

12. TIMER / COUNTER

The GMS87C1408 and GMS87C1404 has four 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 the two 8-bit Timer/Counter or one 16-bit Timer/Counter by combining them. Also Timer 2 and Timer 3 are same. In this document, explain Timer 0 and Timer 1 because Timer2 and Timer3 same with Timer 0 and Timer 1.

In the "timer" function, the register is increased every internal clock input. Thus, one can think of it as counting internal clock input. Since a least clock consists of 2 and 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) or EC1(Timer 2).

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 and Timer 3 are 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 .

Timer 0(2) Mode Register

TM0(2)	-	-	CAPx	TxCK2	TxCK1	TxCK0	TxCN	TxST

ADDRESS : D0H (D6H for TM2)
RESET VALUE : --000000

CAP0 CAP2	<u>Capture mode selection bit</u>. 0 : Disables Capture 1 : Enables Capture	T0CN T2CN	<u>Continue control bit</u> 0 : Stop counting 1 : Start counting continuously
T0CK[2:0] T2CK[2:0]	<u>Input clock selection</u> 000 : fxin ÷ 2, 100 : fxin ÷ 128 001 : fxin ÷ 4, 101 : fxin ÷ 512 010 : fxin ÷ 8, 110 : fxin ÷ 2048 011 : fxin ÷ 32, 111 : External Event (EC0)	T0ST T2ST	<u>Start control bit</u> 0 : Stop counting 1 : Counter register is cleared and start again

Timer 1(3) Mode Register

TM1(3)	POL	16BIT	PWMxE	CAPx	TxCK1	TxCK0	TxCN	TxST

ADDRESS : D2H (D8H for TM3)
RESET VALUE : 00000000

POL	<u>PWM Output Polarity</u> 0 : Duty active low 1 : Duty active high	T1CK[2:0] T3CK[2:0]	<u>Input clock selection</u> 00 : fxin 10 : fxin ÷ 8 01 : fxin ÷ 2 11 : using the Timer 0 clock
16BIT	<u>16-bit mode selection</u> 0 : 8-bit mode 1 : 16-bit mode	T1CN T3CN	<u>Continue control bit</u> 0 : Stop counting 1 : Start counting continuously
PWM0E PWM1E	<u>PWM enable bit</u> 0 : Disables PWM 1 : Enables PWM	T1ST T3ST	<u>Start control bit</u> 0 : Stop counting 1 : Counter register is cleared and start again
CAP1 CAP3	<u>Capture mode selection bit</u>. 0 : Disables Capture 1 : Enables Capture		

Figure 12-1 Timer Mode Register (TMx, x = 0~3)

16BIT	CAP0	CAP1	PWME	T0CK[2:0]	T1CK[1:0]	PWMO	TIMER 0	TIMER1
0	0	0	0	XXX	XX	0	8-bit Timer	8-bit Timer
0	0	1	0	111	XX	0	8-bit Event Counter	8-bit Capture
0	1	0	0	XXX	XX	1	8-bit Capture	8-bit Compare output
0	X[1]	0	1	XXX	XX	1	8-bit Timer/Counter	10-bit PWM
1	0	0	0	XXX	11	0	16-bit Timer	
1	0	0	0	111	11	0	16-bit Event Counter	
1	1	X	0	XXX	11	0	16-bit Capture	
1	0	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 GMS87C1408 and GMS87C1404 has four 8-bit Timer/Counters, Timer 0, Timer 1, Timer 2 and Timer 3, 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 16BIT of TM1 should be cleared to “0”(Table 12-1).

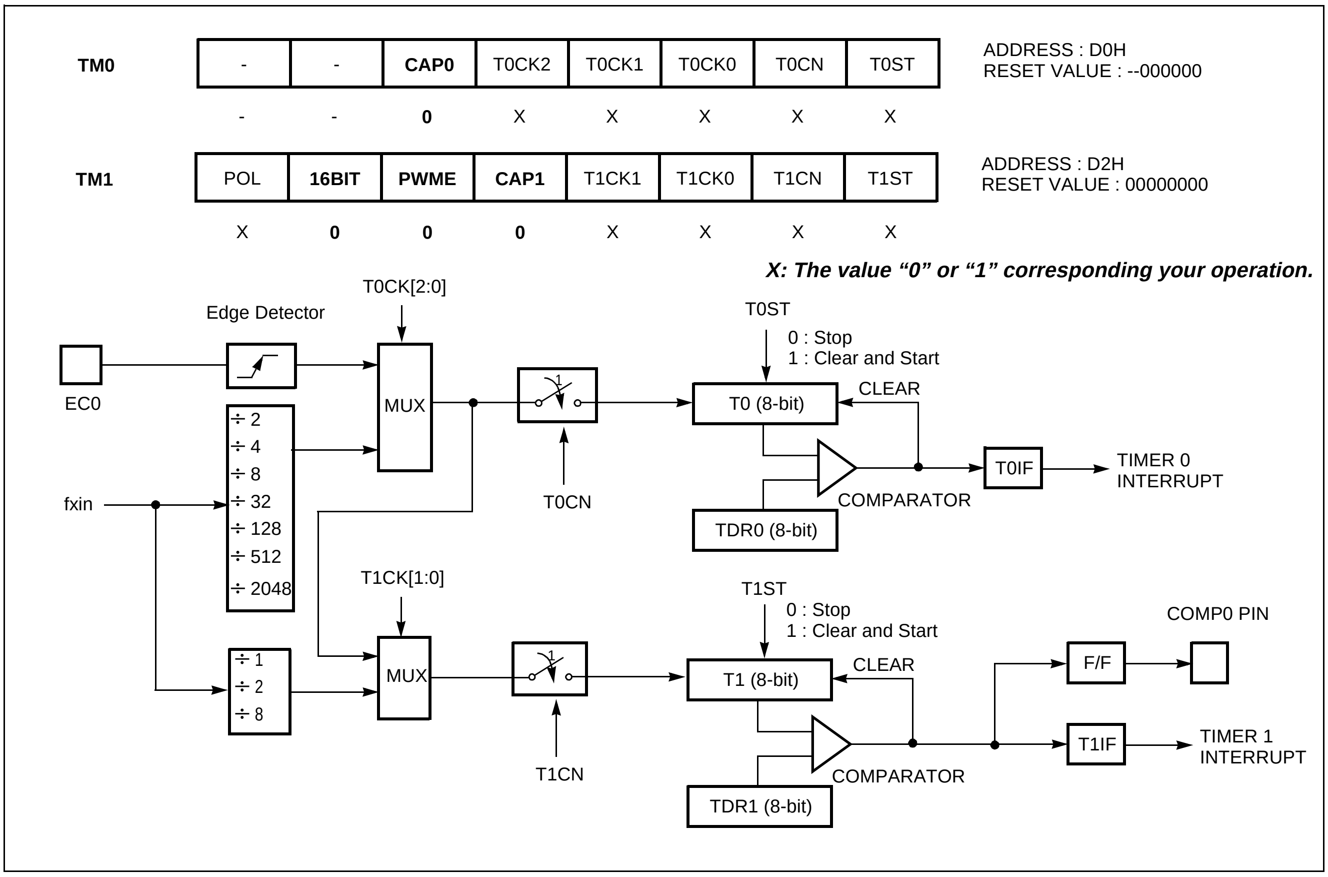


Figure 12-2 8-bit Timer / Counter Mode

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 RA0 of the RA Direction Register RAIO is set to “0”. The Timer 0 can be used as a counter by pin EC0 input, but Timer 1 can not.

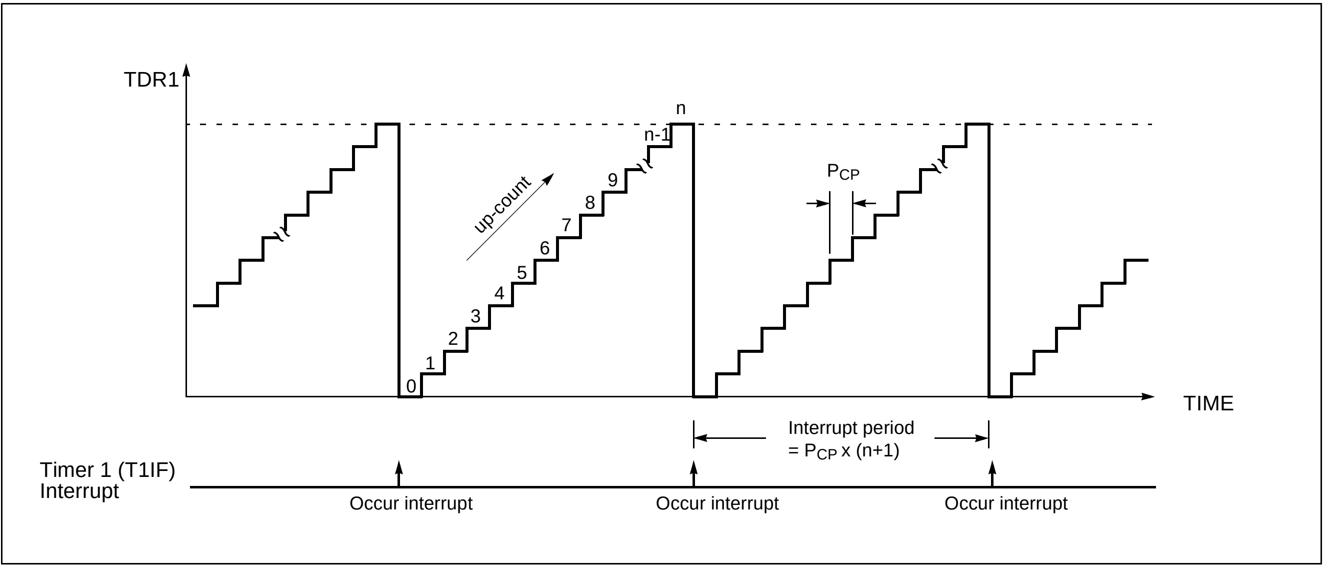


Figure 12-3 Counting Example of Timer Data Registers

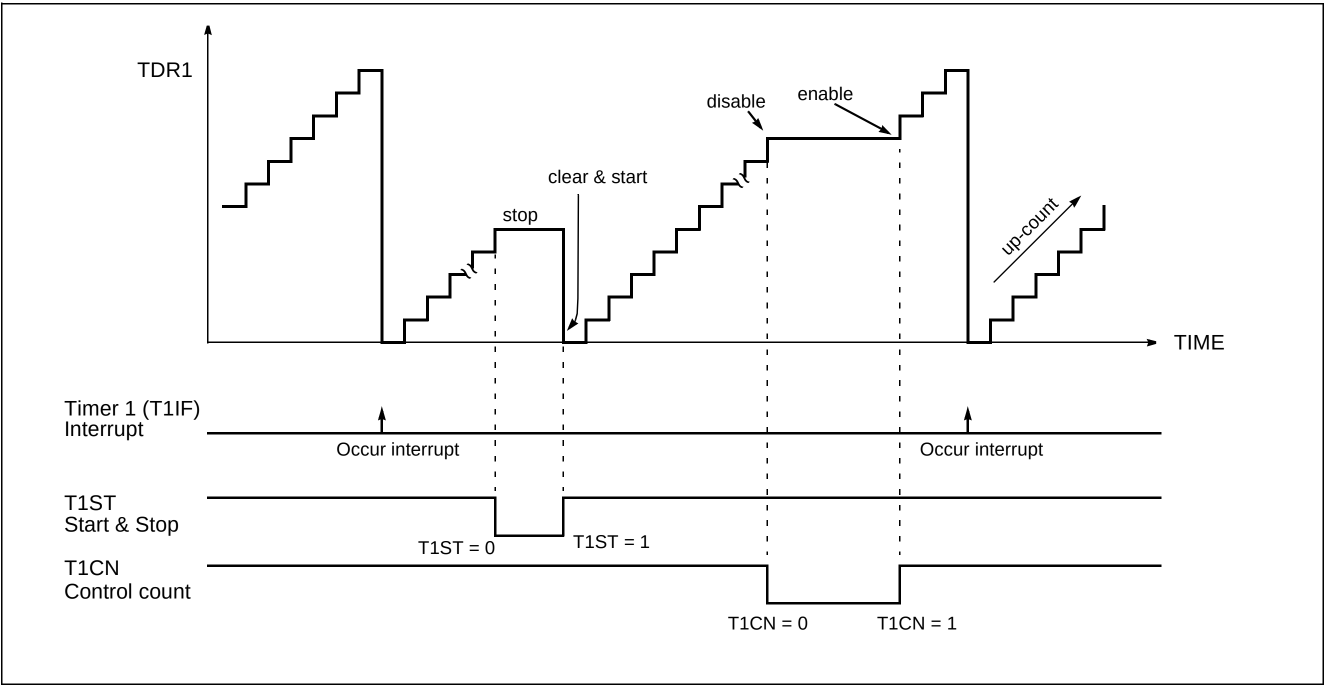


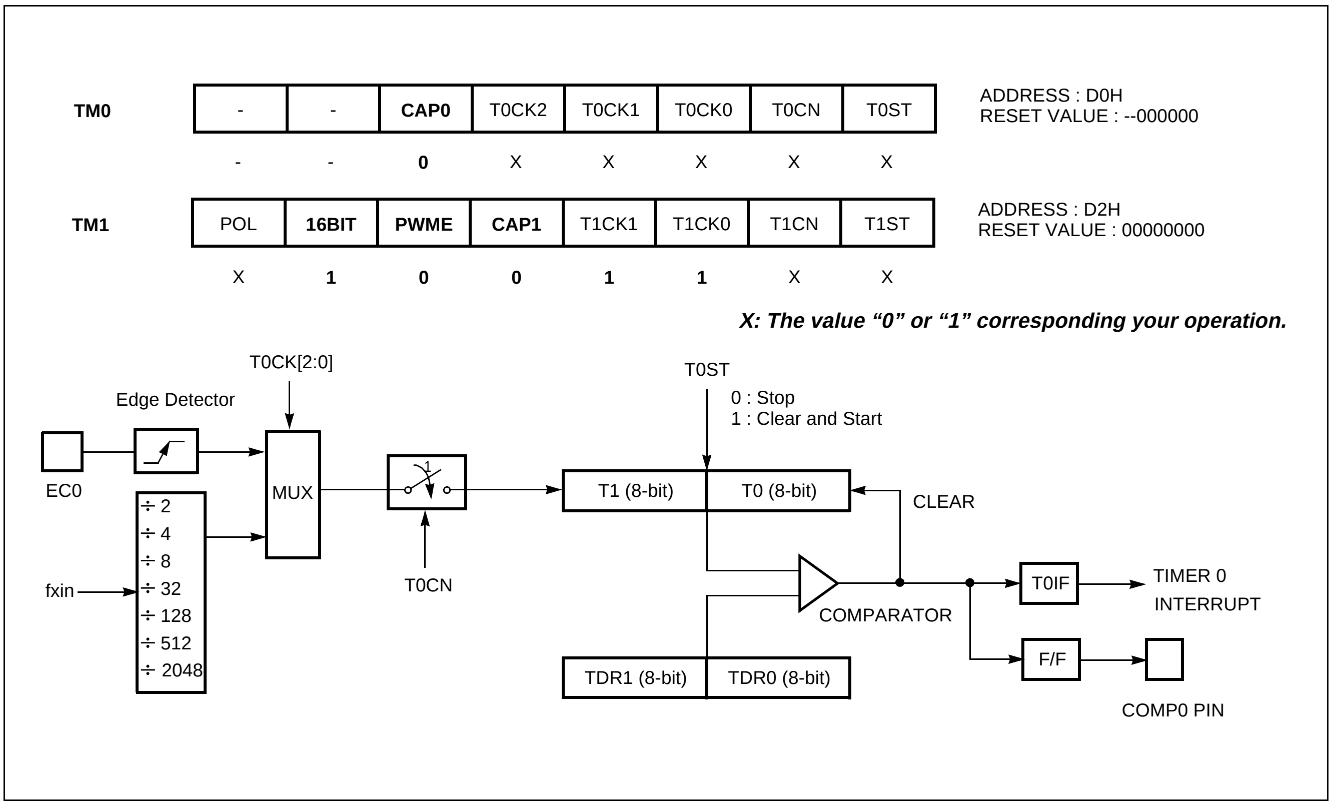
Figure 12-4 Timer Count Operation

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 T0SL0.

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



TM0	-	-	**CAP0**	T0CK2	T0CK1	T0CK0	T0CN	T0ST	ADDRESS : D0H RESET VALUE : --000000
	-	-	**0**	X	X	X	X	X	
TM1	POL	**16BIT**	**PWME**	**CAP1**	T1CK1	T1CK0	T1CN	T1ST	ADDRESS : D2H RESET VALUE : 00000000
	X	**1**	**0**	**0**	**1**	**1**	X	X	

X: The value “0” or “1” corresponding your operation.

Figure 12-5 16-bit Timer / Counter Mode

12.3 8-bit Compare Output (16-bit)

The GMS87C1408 and GMS87C1404 has a function of Timer Compare Output. To pulse out, the timer match can goes to port pin(COMP0) 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, RB4/COMP0/PWM.

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 PWMO of RB function register (RBFUNC) should be set to “1”, and the bit PWME 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 CAP0 of timer mode register TM0 (bit CAP1 of timer mode register TM1 for Timer 1) 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 INTx pin causes the current value in the Timer x register (T0,T1), to be captured into registers CDRx (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 INTx pin generate an interrupt.

Note: *The CDRx, TDRx and Tx are in same address. In the capture mode, reading operation is read the CDRx, not Tx because path is opened to the CDRx, and TDRx is only for writing operation.*

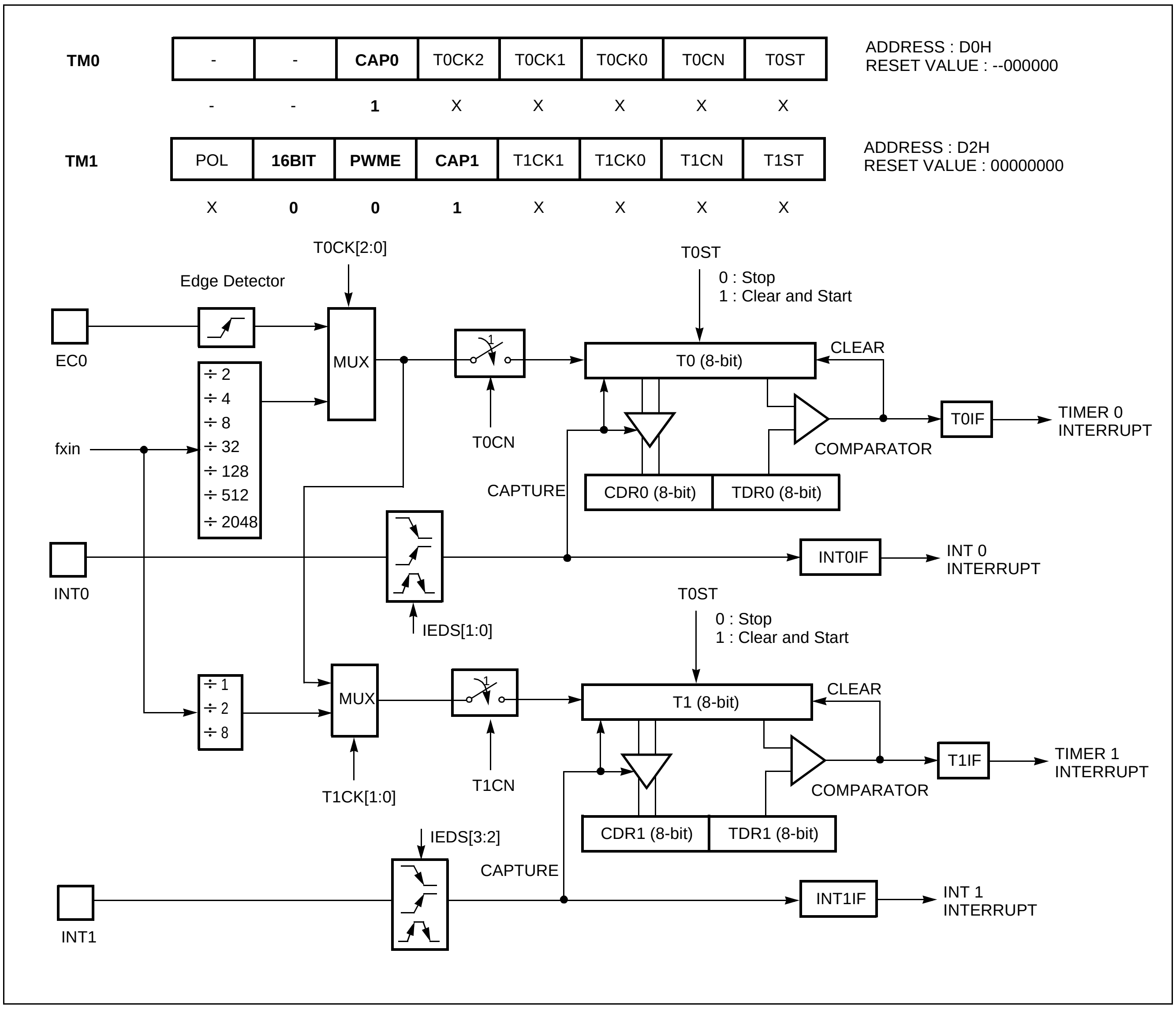


Figure 12-6 8-bit Capture Mode

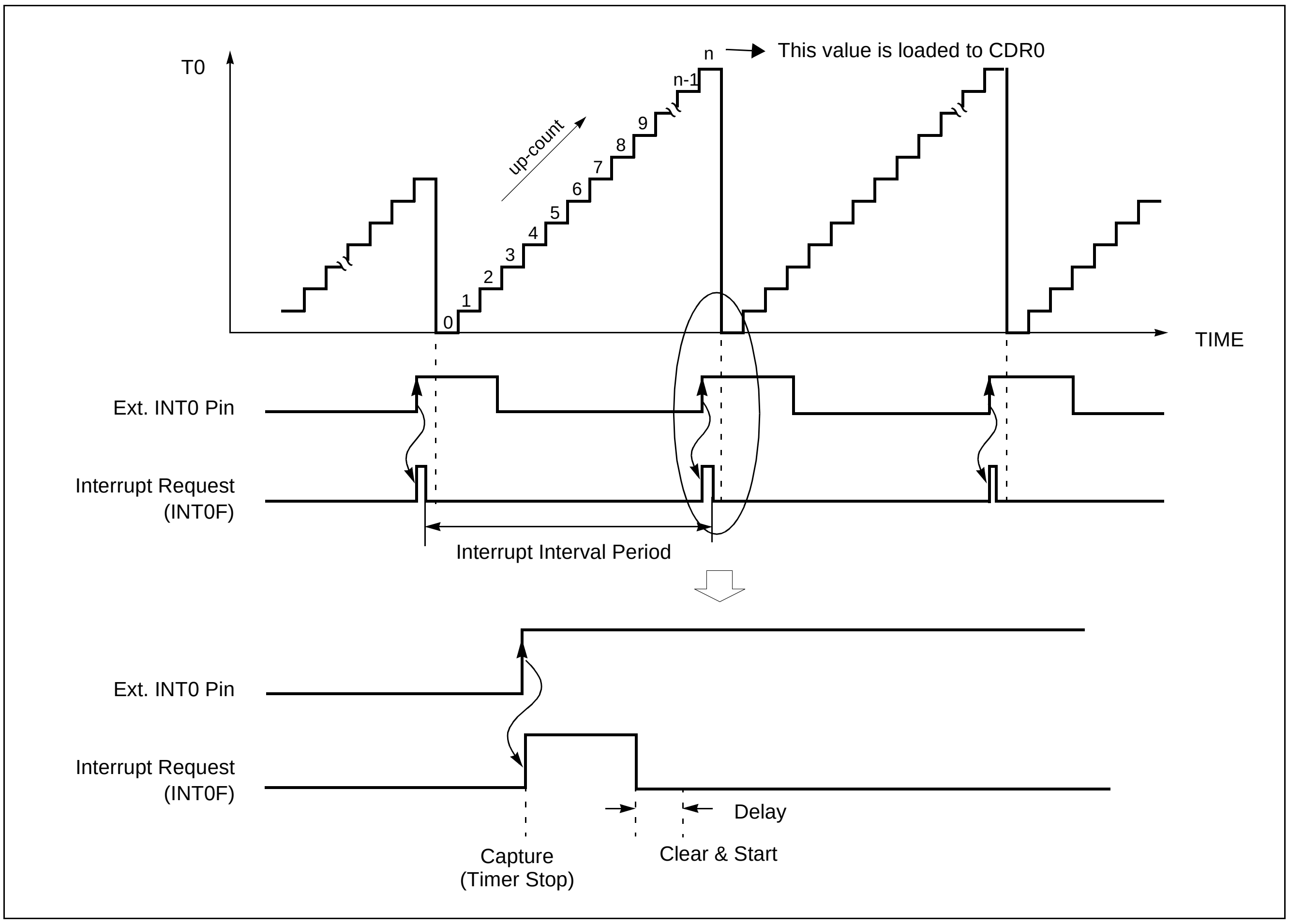

T0
n
This value is loaded to CDR0
n-1
up-count
9
8
7
6
5
4
3
2
1
0
TIME
Ext. INT0 Pin
Interrupt Request (INT0F)
Interrupt Interval Period
Ext. INT0 Pin
Interrupt Request (INT0F)
Delay
Capture (Timer Stop)
Clear & Start

Figure 12-7 Input Capture Operation

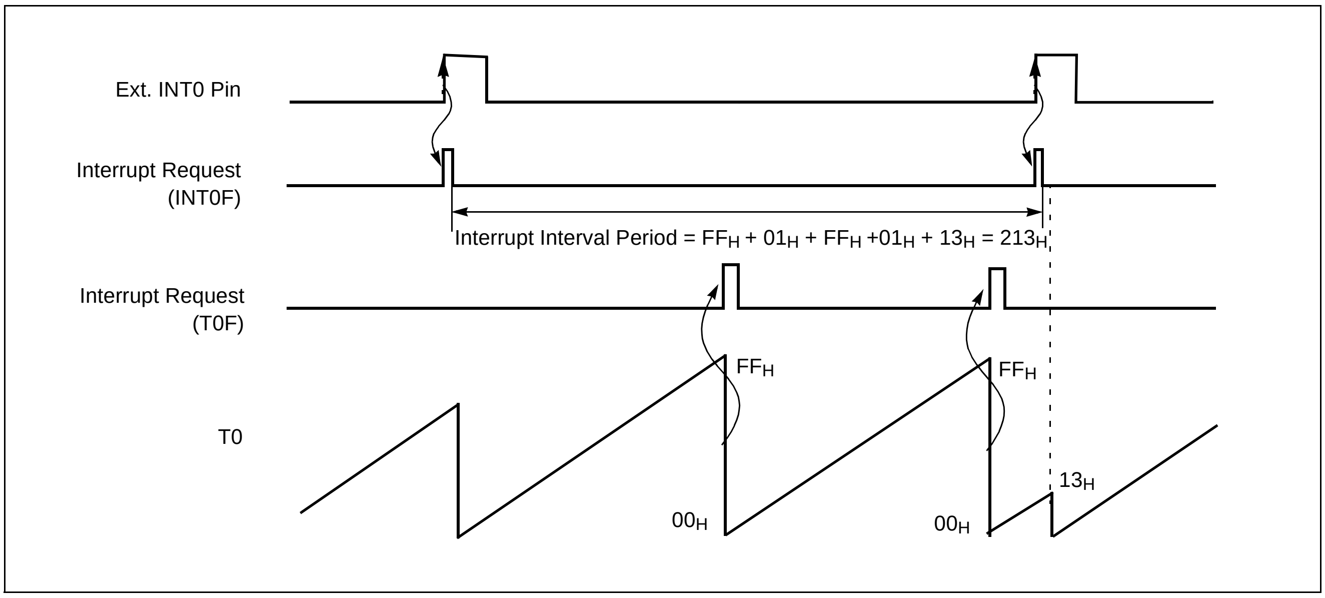

Ext. INT0 Pin
Interrupt Request (INT0F)
Interrupt Interval Period = $FF_H + 01_H + FF_H + 01_H + 13_H = 213_H$
Interrupt Request (T0F)
T0
FF_H
FF_H
13_H
00_H
00_H

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.

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 16BIT of TM1 should be set to “1” respectively.

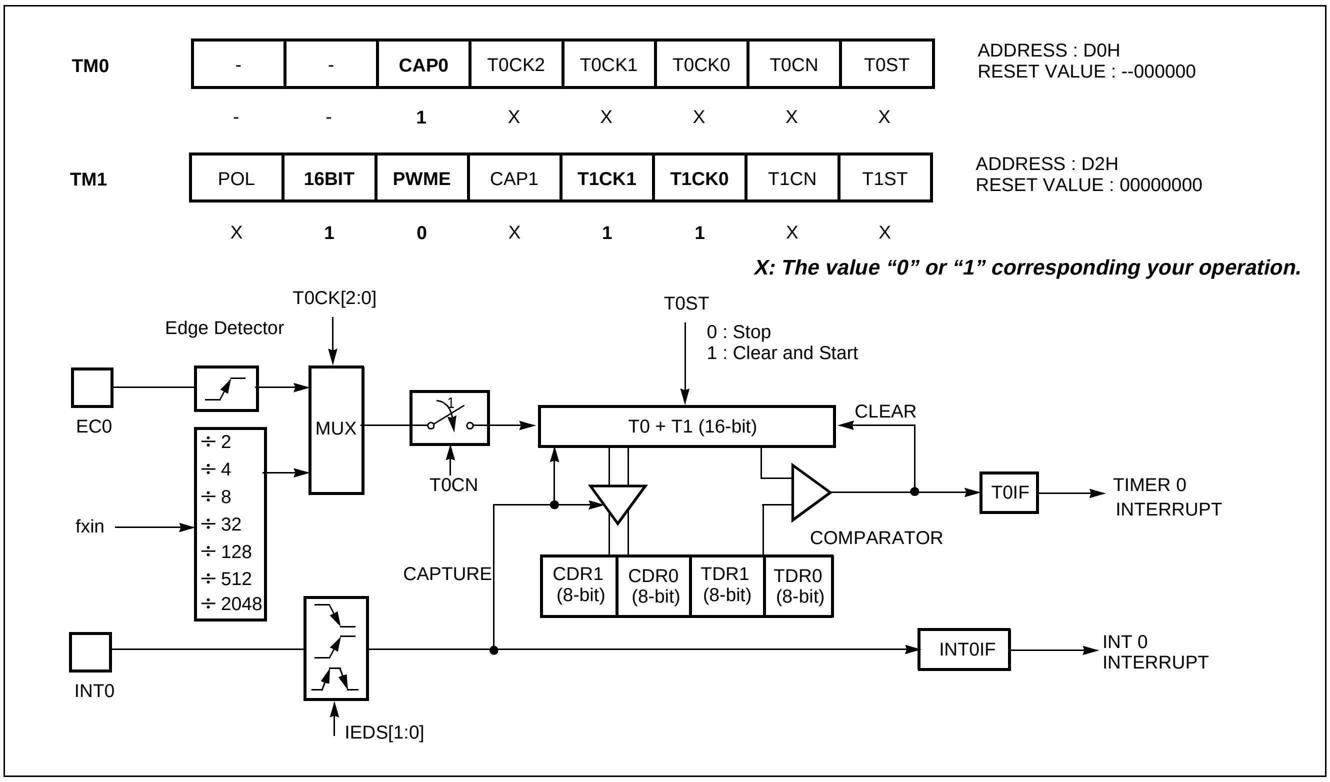


Figure 12-9 16-bit Capture Mode

12.6 PWM Mode

The GMS87C1408 and GMS87C1404 has a two high speed PWM (Pulse Width Modulation) functions which shared with Timer1 (Timer 3). In this document, it will be explained only PWM0.

In PWM mode, pin RB4/COMP0/PWM0 outputs up to a 10-bit resolution PWM output. This pin should be configure as a PWM output by setting “1” bit PWM0O in RBFUNC register. (PWM1 output by setting “1” bit PWM1O in RBFUNC)

The period of the PWM output is determined by the T1PPR (PWM0 Period Register) and PWM0HR[3:2] (bit3,2 of PWM0 High Register) and the duty of the PWM output is determined by the T1PDR (PWM0 Duty Register) and PWM0HR[1:0] (bit1,0 of PWM0 High Register).

The user writes the lower 8-bit period value to the T1PPR and the higher 2-bit period value to the PWM0HR[3:2]. And writes duty value to the T1PDR and the PWM0HR[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 = [PWM0HR[3:2]T1PPR] X Source Clock

PWM Duty = [PWM0HR[1:0]T1PDR] X 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 “00_H”, 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(3) 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  PWM0HR,#00H
          LDM  RBFUNC,#0001_1100B
          LDM  TM1,#1010_1011B
```

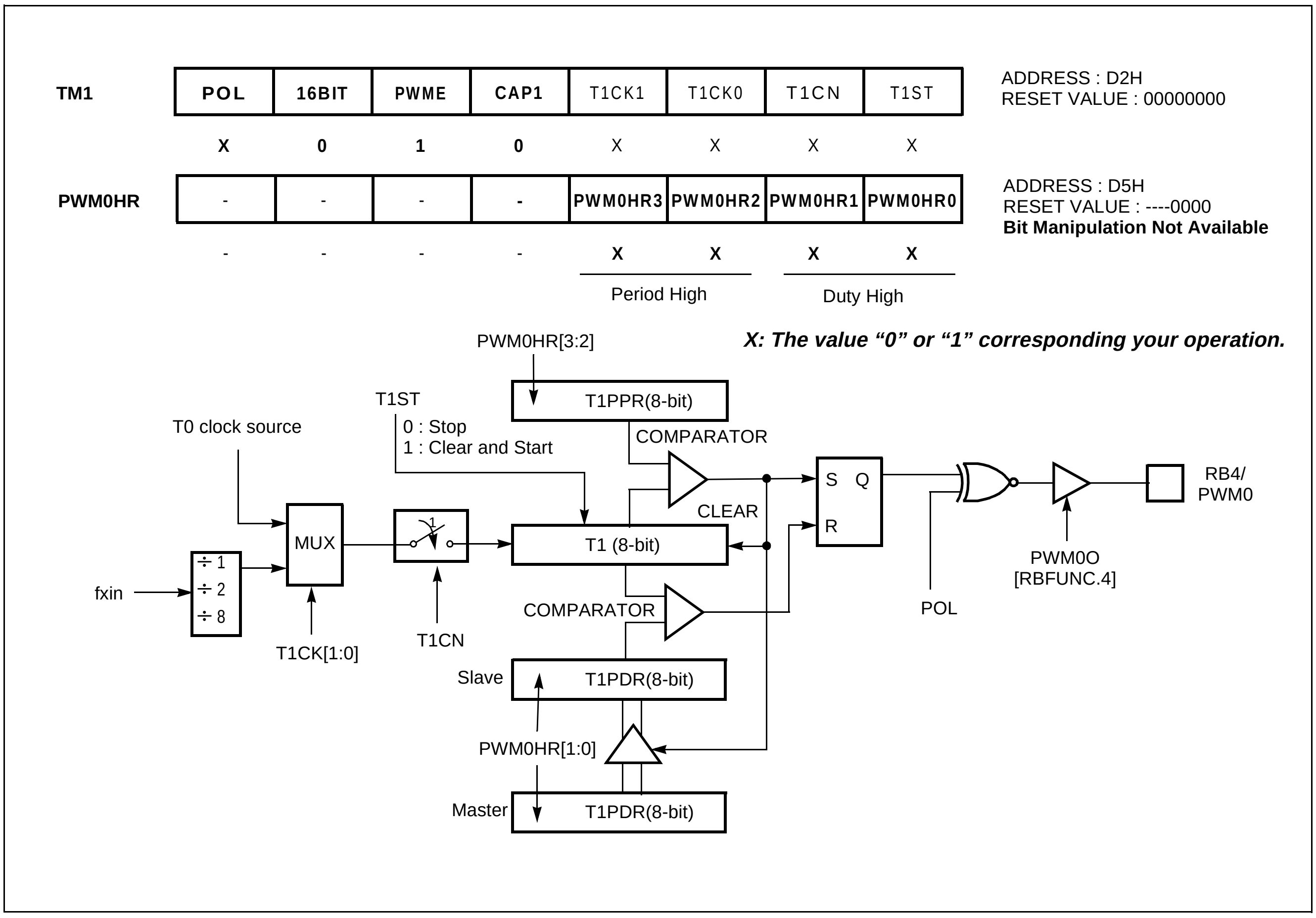


Figure 12-10 PWM Mode

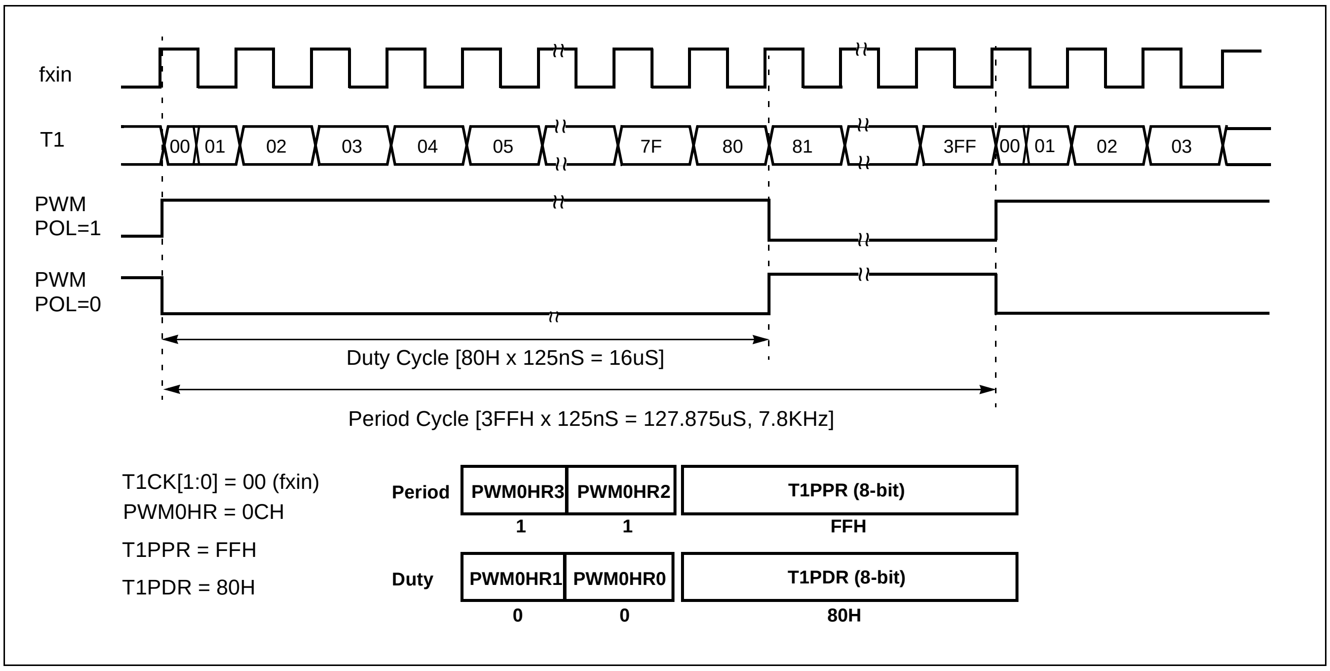

fxin
T1
00 01 02 03 04 05 7F 80 81 3FF 00 01 02 03
PWM POL=1
PWM POL=0
Duty Cycle [80H x 125nS = 16uS]
Period Cycle [3FFH x 125nS = 127.875uS, 7.8KHz]
T1CK[1:0] = 00 (fxin)
PWM0HR = 0CH
T1PPR = FFH
T1PDR = 80H
Period PWM0HR3 PWM0HR2 T1PPR (8-bit)
1 1 FFH
Duty PWM0HR1 PWM0HR0 T1PDR (8-bit)
0 0 80H

Figure 12-11 Example of PWM at 8MHz

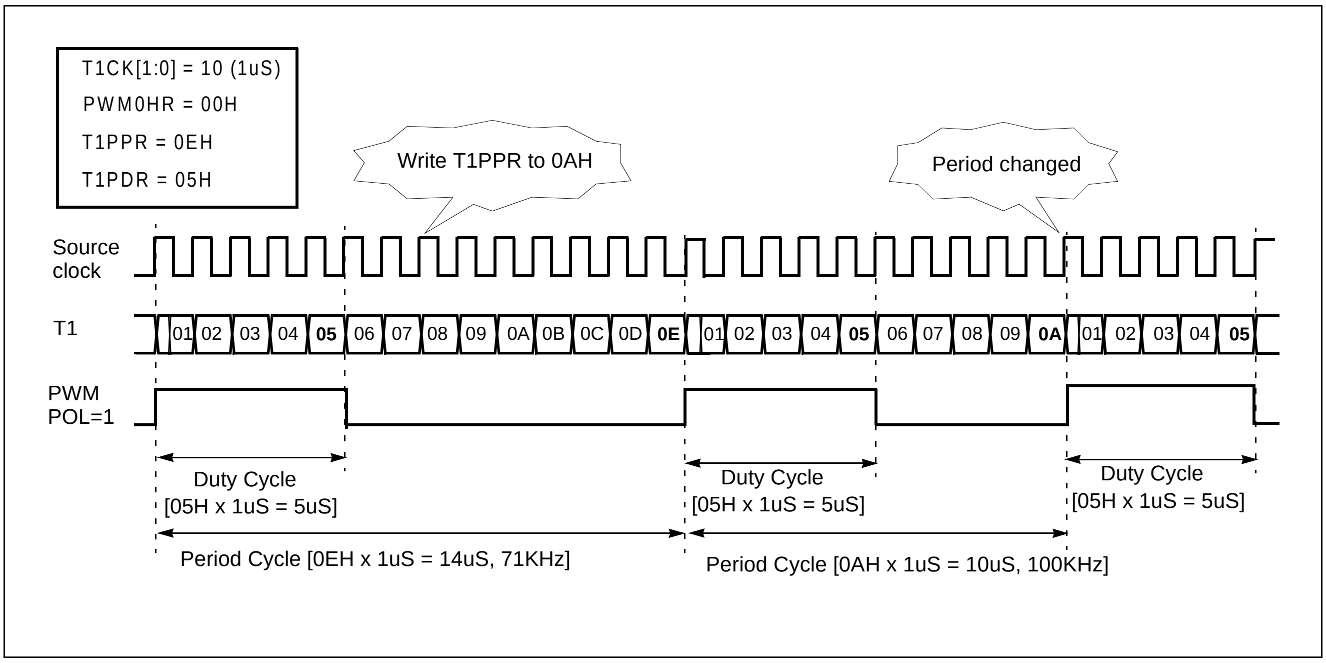

T1CK[1:0] = 10 (1uS)
PWM0HR = 00H
T1PPR = 0EH
T1PDR = 05H
Write T1PPR to 0AH
Period changed
Source clock
T1
01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 01 02 03 04 05 06 07 08 09 0A 01 02 03 04 05
PWM POL=1
Duty Cycle [05H x 1uS = 5uS]
Duty Cycle [05H x 1uS = 5uS]
Duty Cycle [05H x 1uS = 5uS]
Period Cycle [0EH x 1uS = 14uS, 71KHz]
Period Cycle [0AH x 1uS = 10uS, 100KHz]

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

13. Serial Peripheral Interface

The Serial Peripheral Interface (SPI) module is a serial interface useful for communicating with other peripheral of microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc.

SPI Mode Control Register

SIOM	POL	SRDY	SM1	SM0	SCK1	SCK0	SIOST	SIOSF

ADDRESS : E0H
RESET VALUE : 00000001

POL	Serial Clock Polarity Selection bit. 0 : Data Transmission at falling edge (Received data latch at rising edge) 1 : Data Transmission at rising edge (Received data latch at falling edge)
SRDY	Serial Ready Enable bit 0 : Disable (RC3) 1 : Enable (SRDYIN / $\overline{\text{SRDYOUT}}$)
SM[1:0]	Serial Operation Mode Selection bits 00 : Normal Port (RC4, RC5, RC6) 01 : Transmit Mode (SCK, RC5, SOUT) 10 : Receive Mode (SCK, SIN, RC6) 11 : Transmit & Receive Mode (SCK, SIN, SOUT)
SCK[1:0]	Serial Clock Selection bits 00 : fxin ÷ 4 01 : fxin ÷ 16 10 : TMR2OV (Overflow of Timer 2) 11 : External Clock
SIOST	Serial Transmit Start bit 0 : Disable 1 : Start (After one SCK, becomes “0”)
SIOSF	Serial Transmit Status bit 0 : During Transmission 1 : Finished

SPI Data Register

SIOR

ADDRESS : E1H
RESET VALUE : Undefined

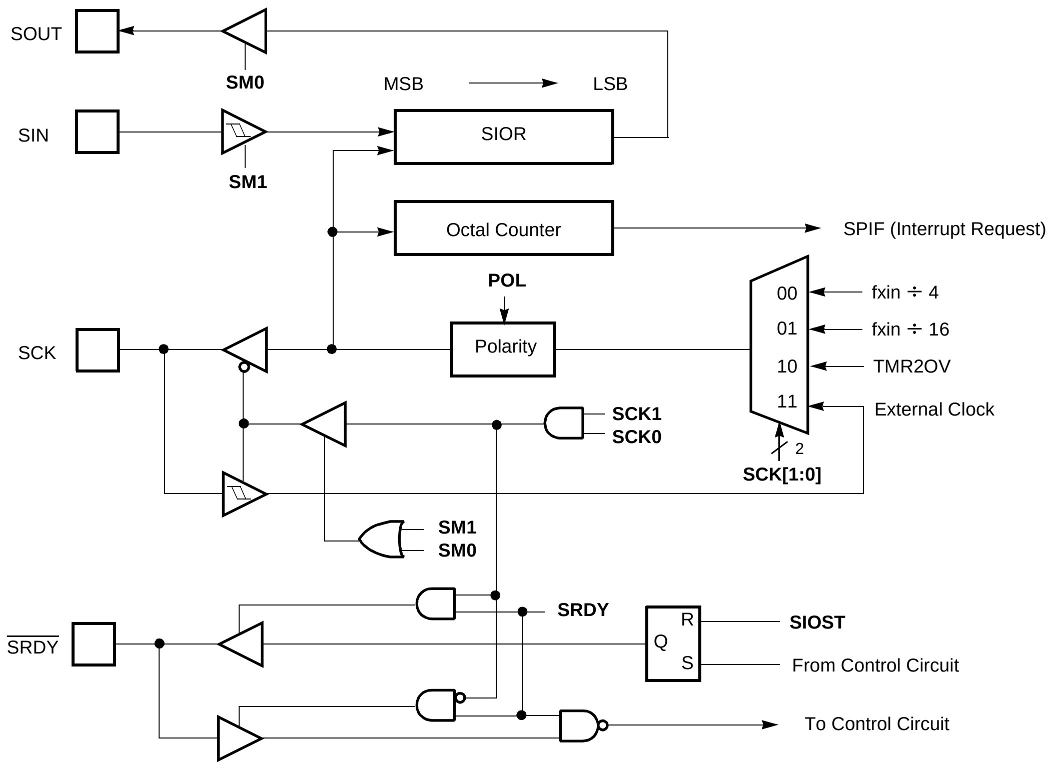


Figure 13-1 SPI Registers and Block Diagram

The SPI allows 8-bits of data to be synchronously transmitted and received. To accomplish communication, typically three pins are used:

- Serial Data In	RC5/SIN
- Serial Data Out	RC6/SOUT
- Serial Clock	RC4/SCK

Additonarlly a fourth pin may be used when in a master or a slave mode of operation:

- Serial Transfer Ready	RC3/SRDYIN/SRDYOUT

The serial data transfer operation mode is decided by setting the SM1 and SM0 of SPI Mode Control Register, and the transfer clock rate is decided by setting the SCK1 and SCK0 of SPI Mode Control Register as shown in Figure 13-1 . And the polarity of transfer clock is selected by setting the POL.

The bit SRDY is used for master / slave selection. If this bit is set to “1” and SCK[1:0] is set to “11”, the controller is performed to slave controller. As it were, the port RC3 is served for $\overline{\text{SRDYOUT}}$.

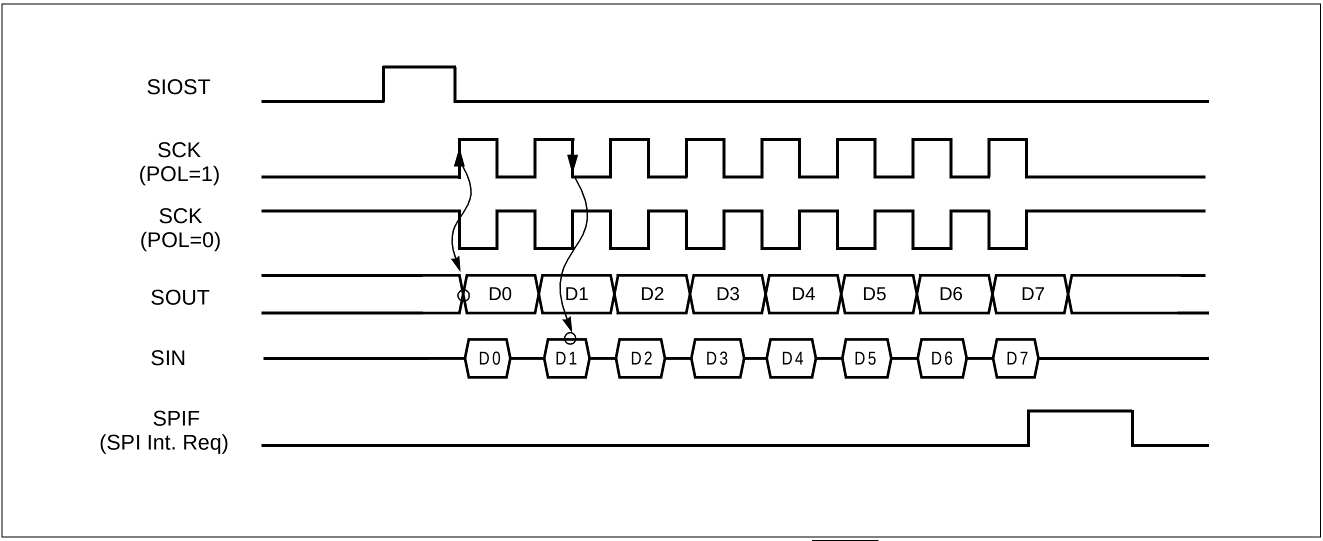


Figure 13-2 SPI Timing Diagram (without $\overline{\text{SRDY}}$ control)

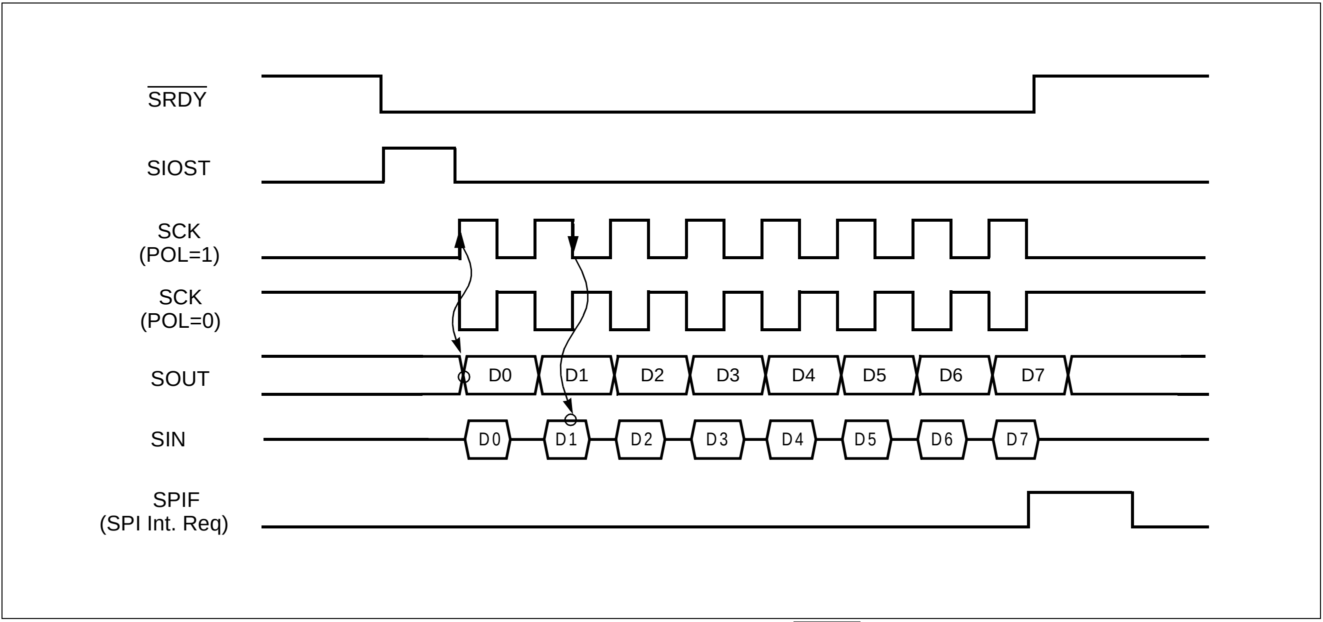


Figure 13-3 SPI Timing Diagram (with $\overline{\text{SRDY}}$ control)

14. Buzzer Output function

The buzzer driver consists of 6-bit binary counter, the buzzer register BUR and the clock selector. It generates square-wave which is very wide range frequency (480 Hz~250 KHz at fxin = 4 MHz) by user programmable counter.

Pin RB1 is assigned for output port of Buzzer driver by setting the bit BUZO of RBFUNC to “1”.

The 6-bit buzzer counter is cleared and start the counting by writing signal to the register BUR. It is increased from 00H until it matches 6-bit register BUR.

Also, it is cleared by counter overflow and count up to output the square wave pulse of duty 50%.

The bit 0 to 5 of BUR determines output frequency for buzzer driving. Frequency calculation is following as shown below.

$$f_{BUZ}(Hz) = \frac{\text{Oscillator Frequency}}{2 \times \text{Prescaler Ratio} \times (BUR + 1)}$$

The bits BUCK1, BUCK0 of BUR selects the source clock from prescaler output.

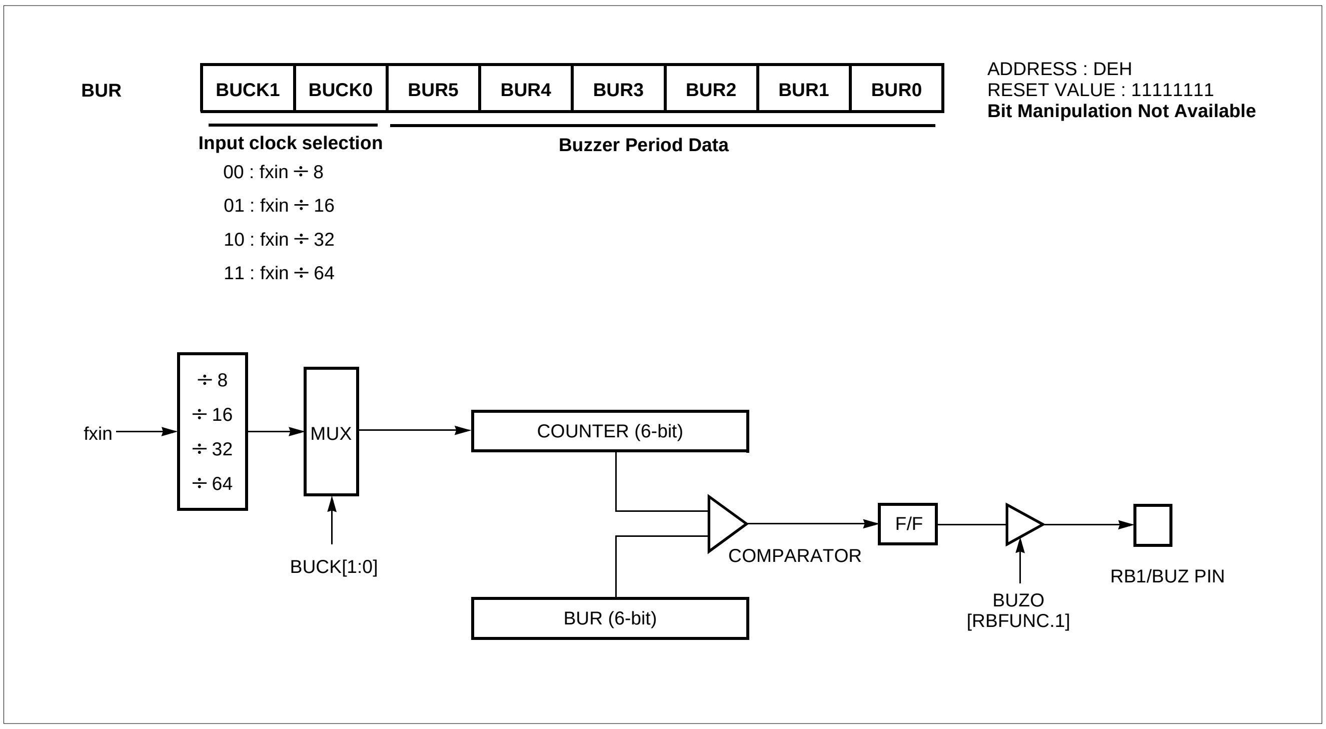


Figure 14-1 Buzzer Driver

15. 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 analog reference voltage is selected to V_{DD} or AVref by setting of the bit AVREFS in RBFUNC register. If external analog reference AVref is selected, the bit ANSEL0 should not be set to “1”, because this pin is used to an analog reference of A/D converter.

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 15-2 , controls the operation of the A/D converter module. The port pins can be configure as analog inputs or digital I/O.

To use analog inputs, each port is assigned analog input port by setting the bit ANSEL[7:0] in RAFUNC register. And selected the corresponding channel to be converted by setting ADS[2:0].

The processing of conversion is 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 15-1 . 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 uS (at fxin=8 MHz).

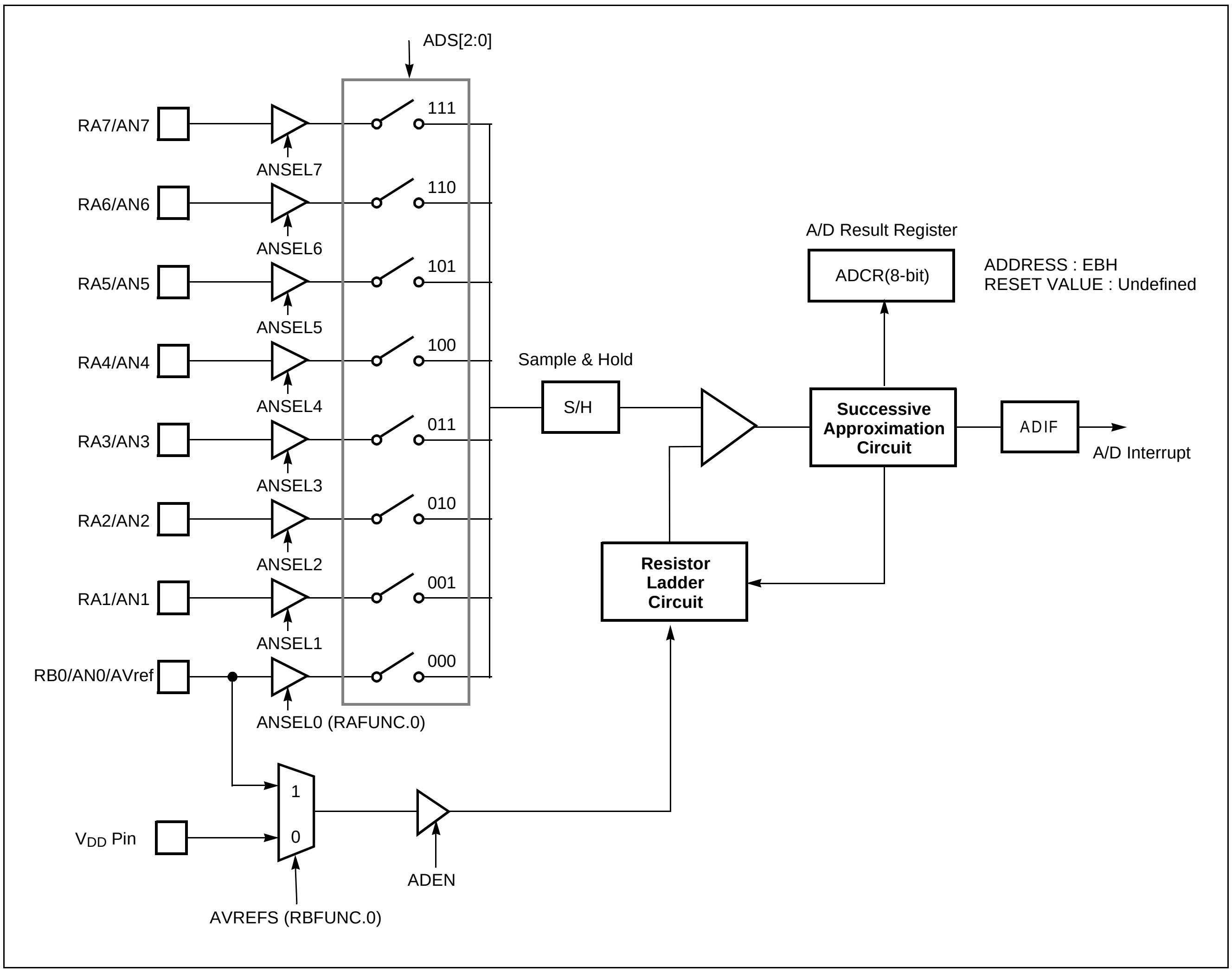


Figure 15-1 A/D Converter Block Diagram

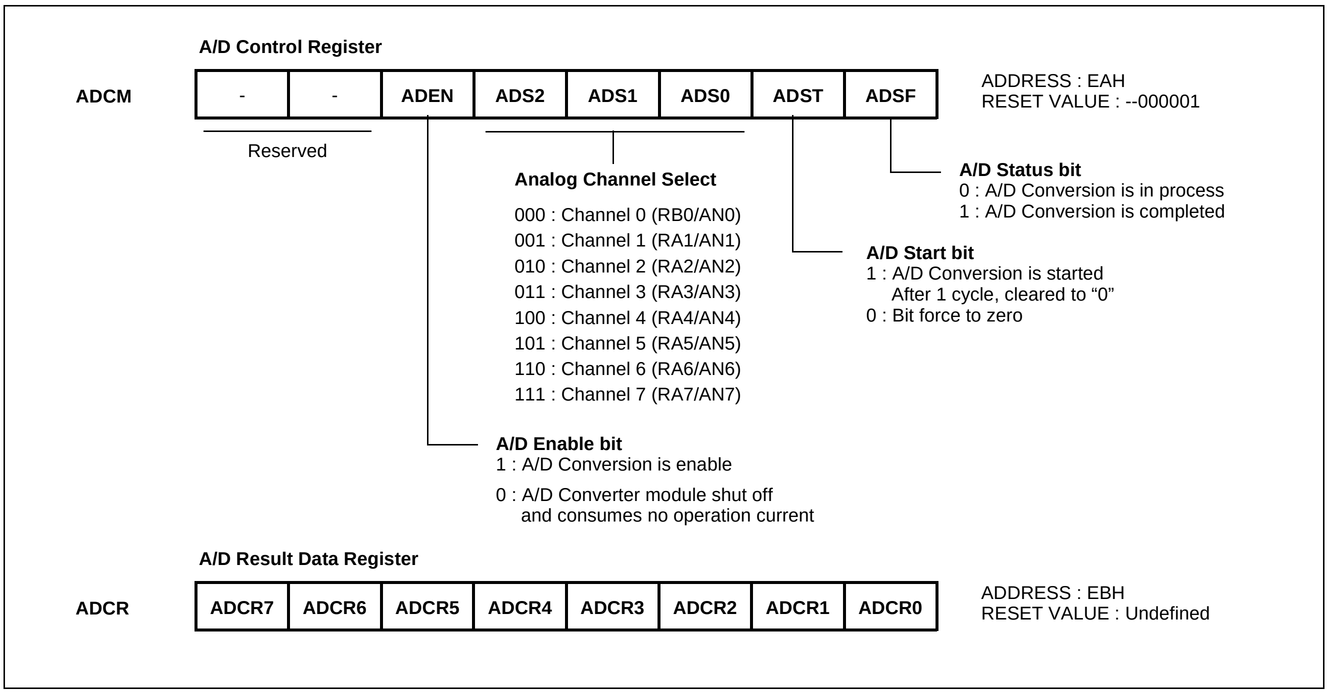

A/D Control Register
ADCM
-
-
ADEN
ADS2
ADS1
ADS0
ADST
ADSF
ADDRESS : EAH
RESET VALUE : --000001
Reserved
Analog Channel Select
000 : Channel 0 (RB0/AN0)
001 : Channel 1 (RA1/AN1)
010 : Channel 2 (RA2/AN2)
011 : Channel 3 (RA3/AN3)
100 : Channel 4 (RA4/AN4)
101 : Channel 5 (RA5/AN5)
110 : Channel 6 (RA6/AN6)
111 : Channel 7 (RA7/AN7)
A/D Status bit
0 : A/D Conversion is in process
1 : A/D Conversion is completed
A/D Start bit
1 : A/D Conversion is started
After 1 cycle, cleared to "0"
0 : Bit force to zero
A/D Enable bit
1 : A/D Conversion is enable
0 : A/D Converter module shut off and consumes no operation current
A/D Result Data Register
ADCR
ADCR7
ADCR6
ADCR5
ADCR4
ADCR3
ADCR2
ADCR1
ADCR0
ADDRESS : EBH
RESET VALUE : Undefined

Figure 15-2 A/D Converter Registers

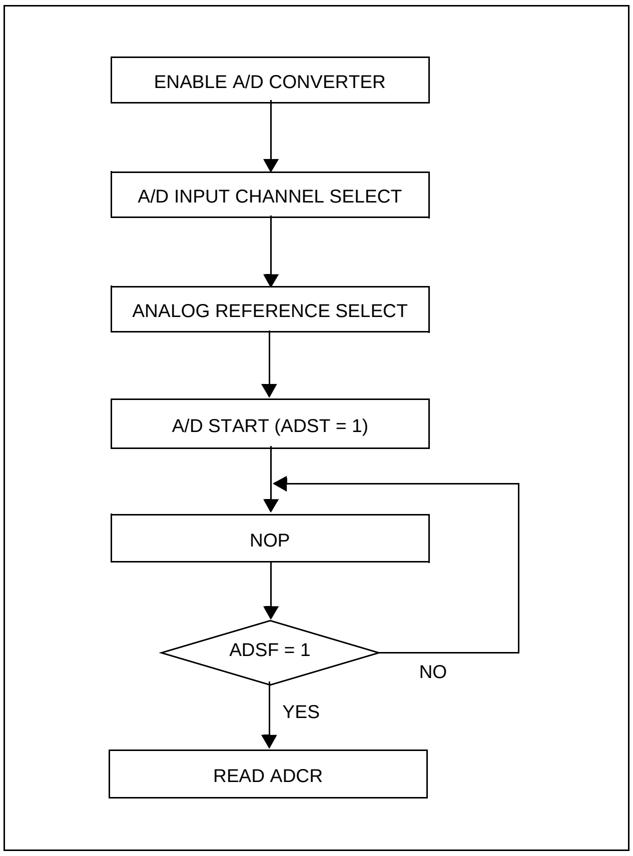


Figure 15-3 A/D Converter Operation Flow

A/D Converter Cautions

(1) Input range of AN0 to AN7

The input voltage of AN0 to AN7 should be within the specification range. In particular, if a voltage above VDD (or AVref) or below V_{SS} is input (even if within the absolute maximum rating range), the conversion value for that channel can not be indeterminate. 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 AVref(or VDD)and AN0 to AN7. 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 15-4 in order to reduce noise.

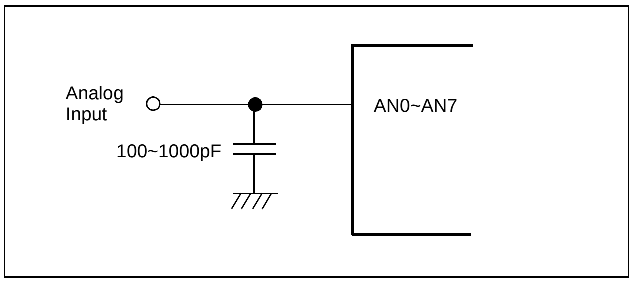


Figure 15-4 Analog Input Pin Connecting Capacitor

(3) Pins AN0/RB0 and AN1/RA1 to AN7/RA7

The analog input pins AN0 to AN7 also function as input/output port (PORT RA and RB0) pins. When A/D conversion is performed with any of pins AN0 to AN7 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) AVref pin input impedance

A series resistor string of approximately 10KΩ is connected between the AVref 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 AVref pin and the V_{SS} pin, and there will be a large reference voltage error.

16. INTERRUPTS

The GMS87C1408 and GMS87C1404 interrupt circuits consist of Interrupt enable register (IENH, IENL), Interrupt request flags of IRQH, IRQL, Interrupt Edge Selection Register (IEDS), priority circuit and Master enable flag("I" flag of PSW). The configuration of interrupt circuit is shown in Figure 16-1 and Interrupt priority is shown in Table 16-1 .

The External Interrupts INT0, INT1, INT2 and INT3 can each be transition-activated (1-to-0, 0-to-1 and both transition).

The flags that actually generate these interrupts are bit INT0IF, INT1IF, INT2IF and INT3IF in Register IRQH. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated.

The Timer 0, Timer 1, Timer 2 and Timer 3 Interrupts are generated by T0IF, T1IF, T2IF and T3IF, which are set by a match in their respective timer/counter register. The AD converter Interrupt is generated by ADIF 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).

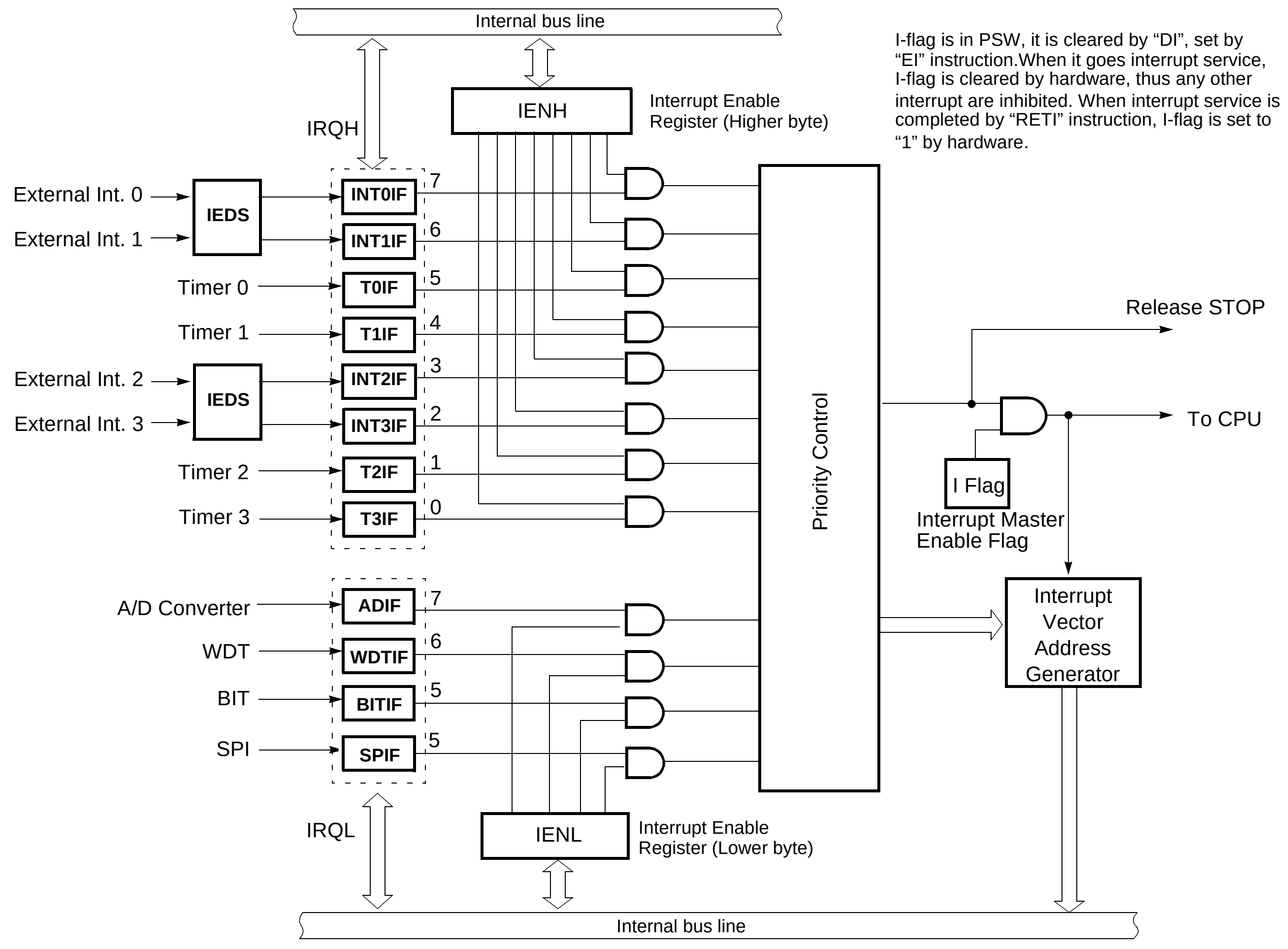

Internal bus line
IRQH
IENH
Interrupt Enable Register (Higher byte)
I-flag is in PSW, it is cleared by "DI", set by "EI" instruction.When it goes interrupt service, I-flag is cleared by hardware, thus any other interrupt are inhibited. When interrupt service is completed by "RETI" instruction, I-flag is set to "1" by hardware.
External Int. 0
External Int. 1
IEDS
INT0IF
INT1IF
Timer 0
T0IF
Timer 1
T1IF
External Int. 2
External Int. 3
INT2IF
INT3IF
Timer 2
T2IF
Timer 3
T3IF
7
6
5
4
3
2
1
0
Priority Control
Release STOP
To CPU
I Flag
Interrupt Master Enable Flag
A/D Converter
ADIF
WDT
WDTIF
BIT
BITIF
SPI
SPIF
Interrupt Vector Address Generator
IRQL
IENL
Interrupt Enable Register (Lower byte)
Internal bus line

Figure 16-1 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, IENL) and the interrupt request flags (in IRQH, IRQL) except Power-on reset and software BRK interrupt.

Interrupt enable registers are shown in Figure 16-2 . 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	-	$FFFE_H$
External Interrupt 0	INT0	1	$FFFA_H$
External Interrupt 1	INT1	2	$FFF8_H$
Timer 0	Timer 0	3	$FFF6_H$
Timer 1	Timer 1	4	$FFF4_H$
External Interrupt 2	INT2	5	$FFF2_H$
External Interrupt 3	INT3	6	$FFF0_H$
Timer 2	Timer 2	7	$FFEE_H$
Timer 3	Timer 3	8	$FFEC_H$
A/D Converter	A/D C	9	$FFEA_H$
Watch Dog Timer	WDT	10	$FFE8_H$
Basic Interval Timer	BIT	11	$FFE6_H$
Serial Interface	SPI	12	$FFE6_H$

Table 16-1 Interrupt Priority

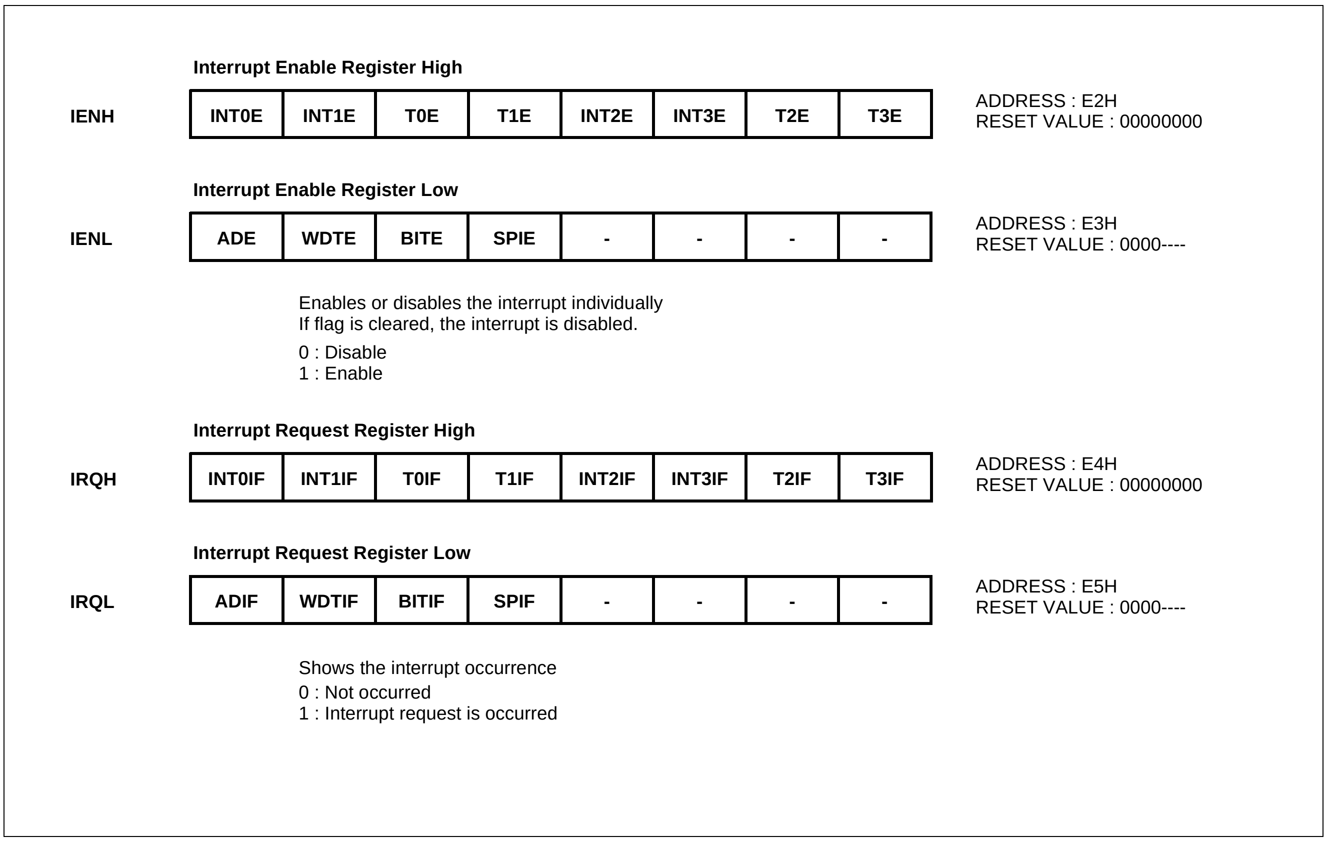


Figure 16-2 Interrupt Enable Registers and Interrupt Request Registers

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.

16.1 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 μs 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.

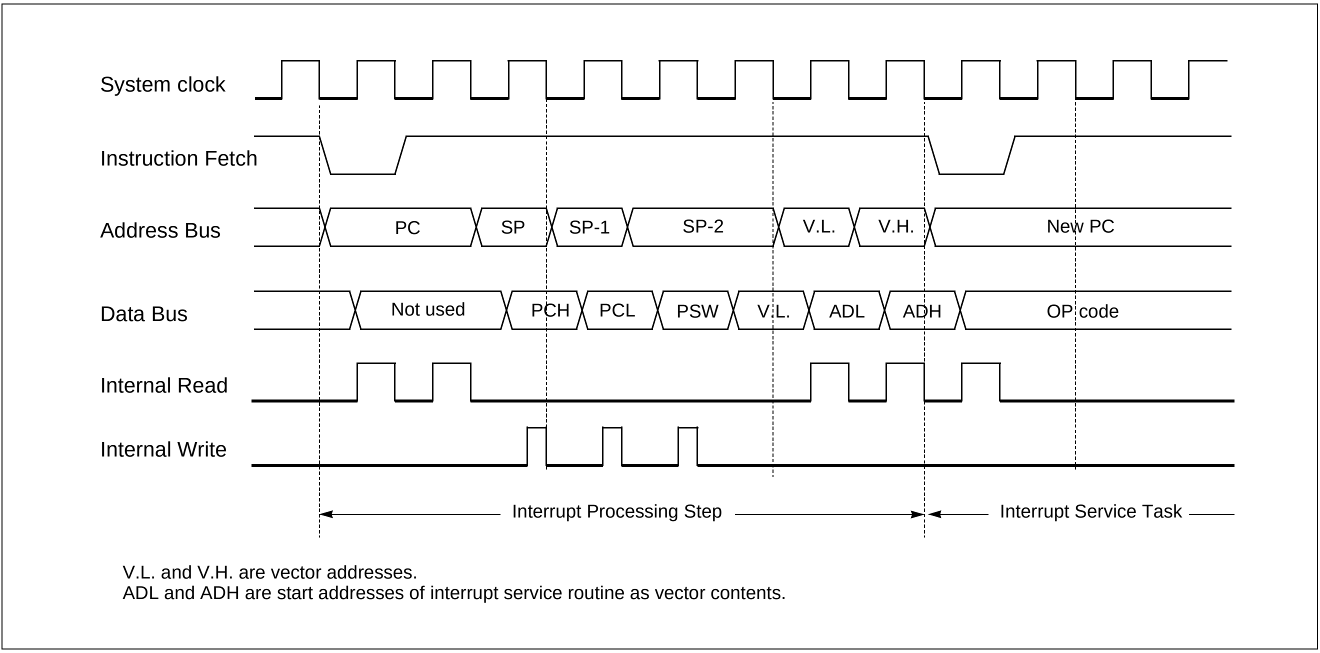
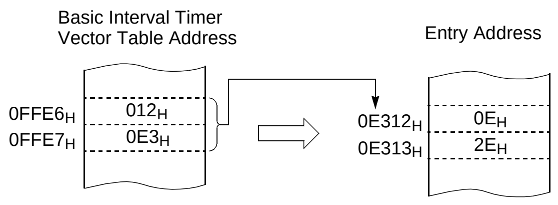


Figure 16-3 Timing chart of Interrupt Acceptance and Interrupt Return Instruction



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

A 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.

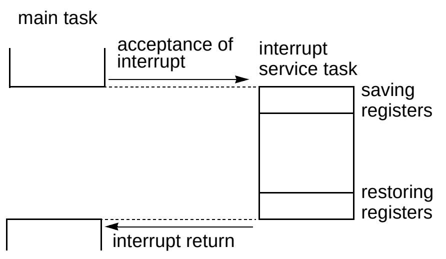
Example: Register save using push and pop instructions

```
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;



16.2 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 16-4 .

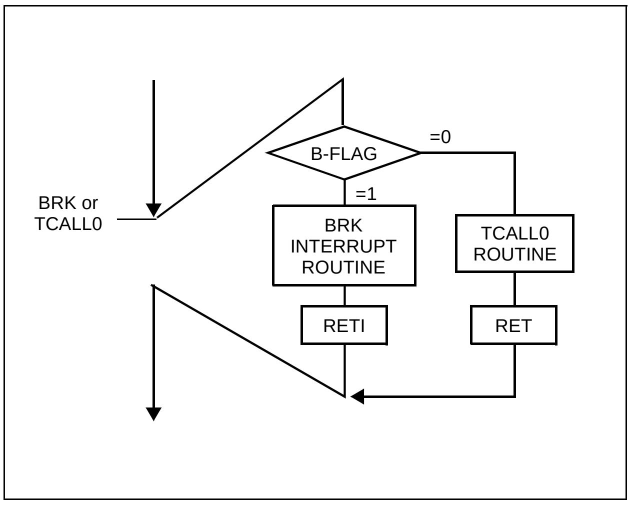
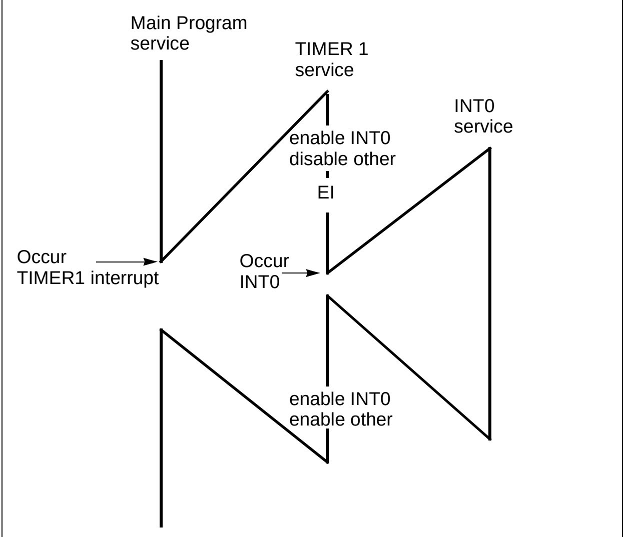


Figure 16-4 Execution of BRK/TCALL0

16.3 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.



In this example, the INT0 interrupt can be serviced without any pending, even TIMER1 is in progress.
Because of re-setting the interrupt enable registers IENH,IENL and master enable "EI" in the TIMER1 routine.

Figure 16-5 Execution of Multi Interrupt

Example: Even though Timer1 interrupt is in progress, INT0 interrupt serviced without any suspend.

```
TIMER1: PUSH   A
        PUSH   X
        PUSH   Y
        LDM    IENH,#80H    ;Enable INT0 only
        LDM    IENL,#0      ;Disable other
        EI                  ;Enable Interrupt
        :
        :
        :


        :
        :
        :
        LDM    IENH,#0FFH   ;Enable all interrupts
        LDM    IENL,#0F0H
        POP    Y
        POP    X
        POP    A
        RETI
```

16.4 External Interrupt

The external interrupt on INT0, INT1, INT2 and INT3 pins are edge triggered depending on the edge selection register IEDS (address $0E6_H$) as shown in Figure 16-6 .

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

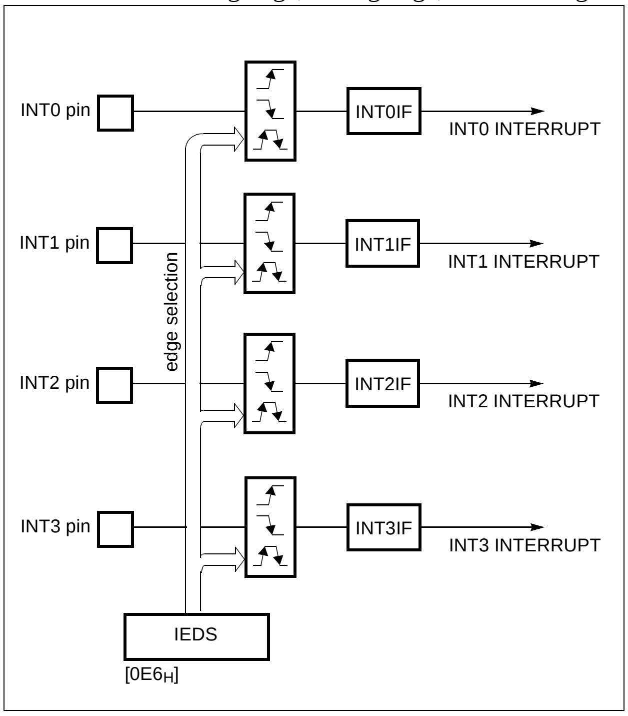
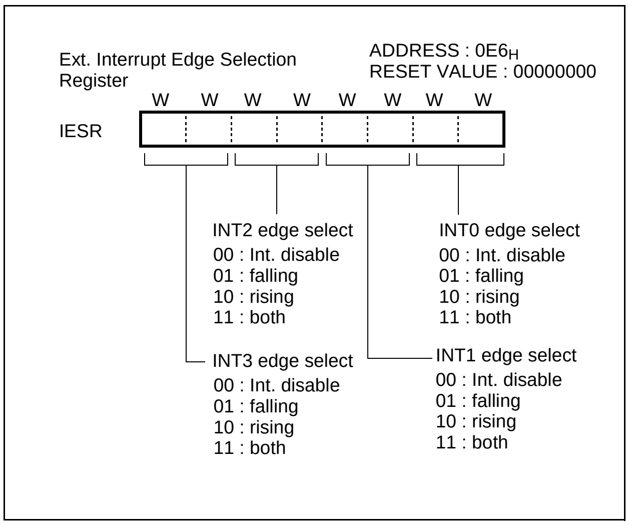


Figure 16-6 External Interrupt Block Diagram



Example: To use as an INT0 and INT2

```
            :
            :
;**** Set port as an input port RB2,RD0
            LDM    RBIO,#1111_1011B
            LDM    RDIO,#1111_1110B
            ;
;**** Set port as an interrupt port
            LDM    RBFUNC,#04H
            LDM    RDFUNC,#01H
            ;
;**** Set Falling-edge Detection
            LDM    IEDS,#0001_0001B
            :
            :
            :
```

Response Time

The INT0, INT1,INT2 and INT3 edge are latched into INT0IF, INT1IF, INT2IF and INT3IF 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.

shows interrupt response timings.

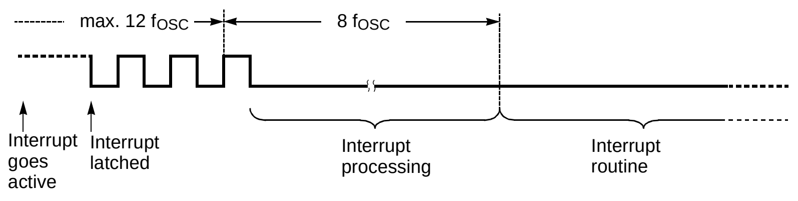


Figure 16-7 Interrupt Response Timing Diagram

17. 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 oscillator which does not require any external components. This RC oscillator is separate from the external oscillator of the Xin pin. It means that the watchdog timer will run, even if the clock on the Xin pin of the device has been stopped, for example, by entering the STOP mode.

The other type is a prescaled system clock.

The watchdog timer consists of 7-bit binary counter and the watchdog timer data register. 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 reset the CPU in accordance with the bit WDTON.

Note: *Because the watchdog timer counter is enabled after clearing Basic Interval Timer, after the bit WDTON set to "1", maximum error of timer is depend on prescaler ratio of Basic Interval Timer.*

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

The RC oscillated watchdog timer is activated by setting the bit RCWDT as shown below.

```
  :
LDM    CKCTLR,#3FH   ; enable the RC-osc WDT
LDM    WDTR,#0FFH    ; set the WDT period
STOP                 ; enter the STOP mode
NOP
NOP                  ; RC-osc WDT running
  :
```

The RC oscillation period is vary with temperature, V_{DD} and process variations from part to part (approximately, 40~120uS). The following equation shows the RC oscillated watchdog timer time-out.

$$T_{RCWDT} = CLK_{RC} \times 2^8 \times [WDTR.6\sim0] + (CLK_{RC} \times 2^8)/2$$

$$\text{where, } CLK_{RC} = 40\sim120uS$$

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.

$$T_{WDT} = [WDTR.6\sim0] \times \text{Interval of BIT}$$

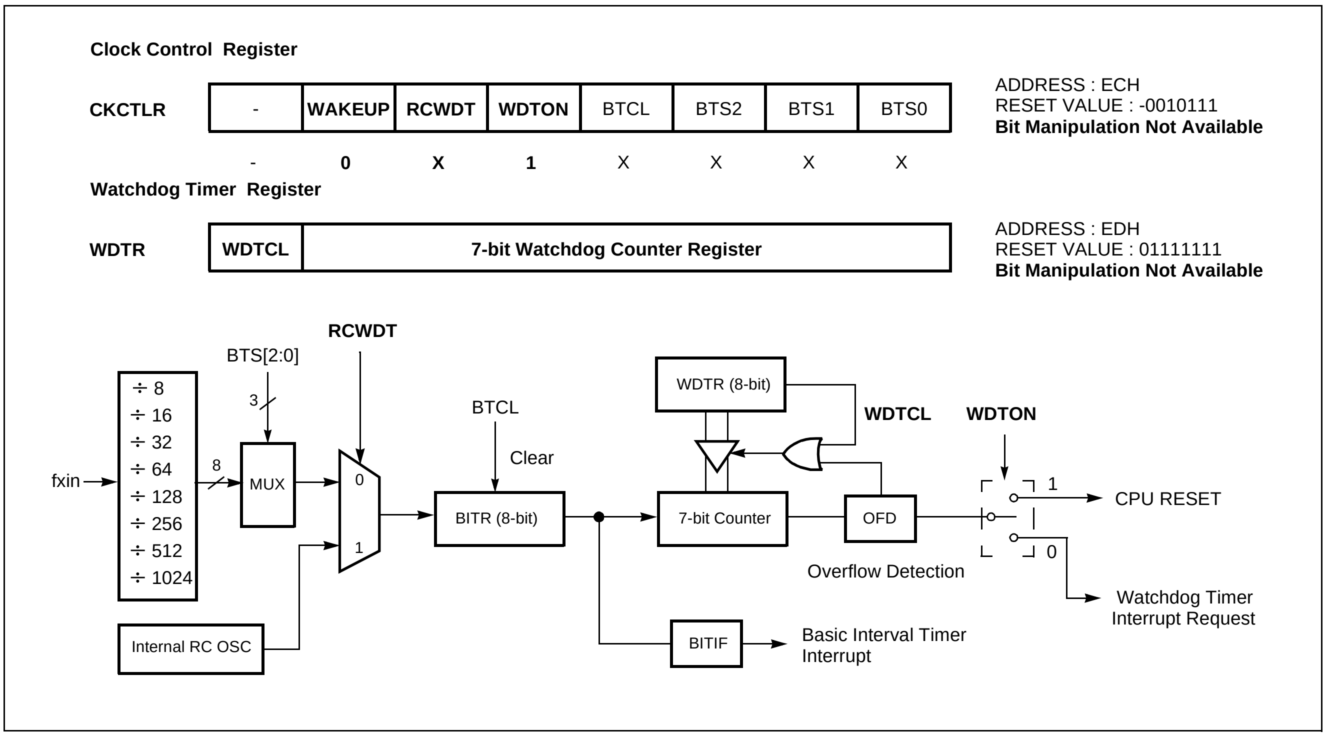


Figure 17-1 Block Diagram of Watchdog Timer

18. Power Saving Mode

For applications where power consumption is a critical factor, device provides two kinds of power saving functions, STOP mode and Wake-up Timer mode.

The power saving function is activated by execution of STOP instruction after setting the corresponding status (WAKEUP) of CKCTLR.

Table 18-1 shows the status of each Power Saving Mode.

Peripheral	STOP	Wake-up Timer
RAM	Retain	Retain
Control Registers	Retain	Retain
I/O Ports	Retain	Retain
CPU	Stop	Stop
Timer0, Timer2	Stop	Operation
Oscillation	Stop	Oscillation
Prescaler	Stop	÷ 2048 only
Entering Condition [WAKEUP]	0	1
Release Sources	RESET, RCWDT, INT0~3, EC0~1, SPI	RESET, RCWDT, INT0~3, EC0~1, SPI, TIMER0, TIMER2

Table 18-1 Power Saving Mode

18.1 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 direction 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 clearing the bit WAKEUP of CKCTLR to "0". (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example "set1" or "clr1" instruction, it may be undesired operation)

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       CKCTLR,#0000_1110B
           STOP
           NOP
           NOP
```

In the STOP operation, the dissipation of the power 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 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. 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. 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 18-1)

By reset, exit from Stop mode is shown in Figure 18-3 .When exit from Stop mode by external interrupt, enough oscillation stabilization time is required to normal operation. Figure 18-2 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..

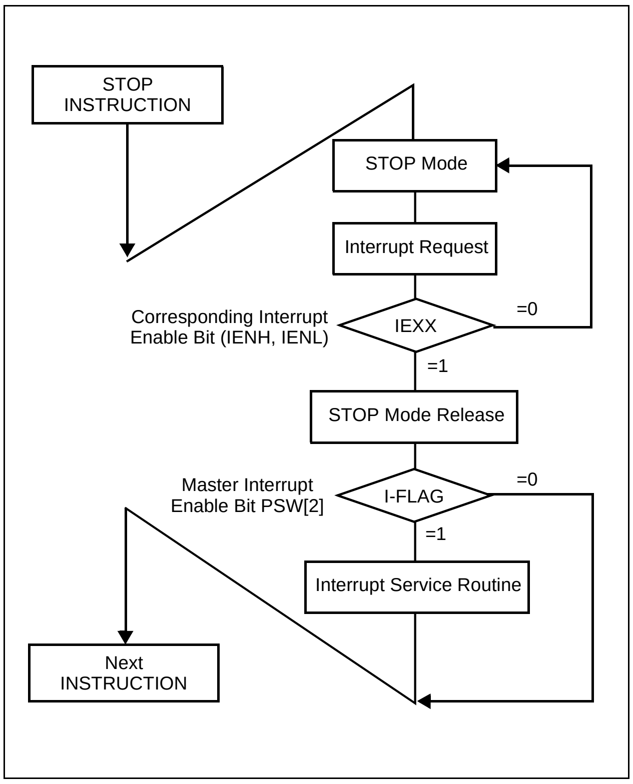

STOP INSTRUCTION
STOP Mode
Interrupt Request
Corresponding Interrupt Enable Bit (IENH, IENL)
IEXX
=0
=1
STOP Mode Release
Master Interrupt Enable Bit PSW[2]
I-FLAG
=0
=1
Interrupt Service Routine
Next INSTRUCTION

Figure 18-1 STOP Releasing Flow by Interrupts

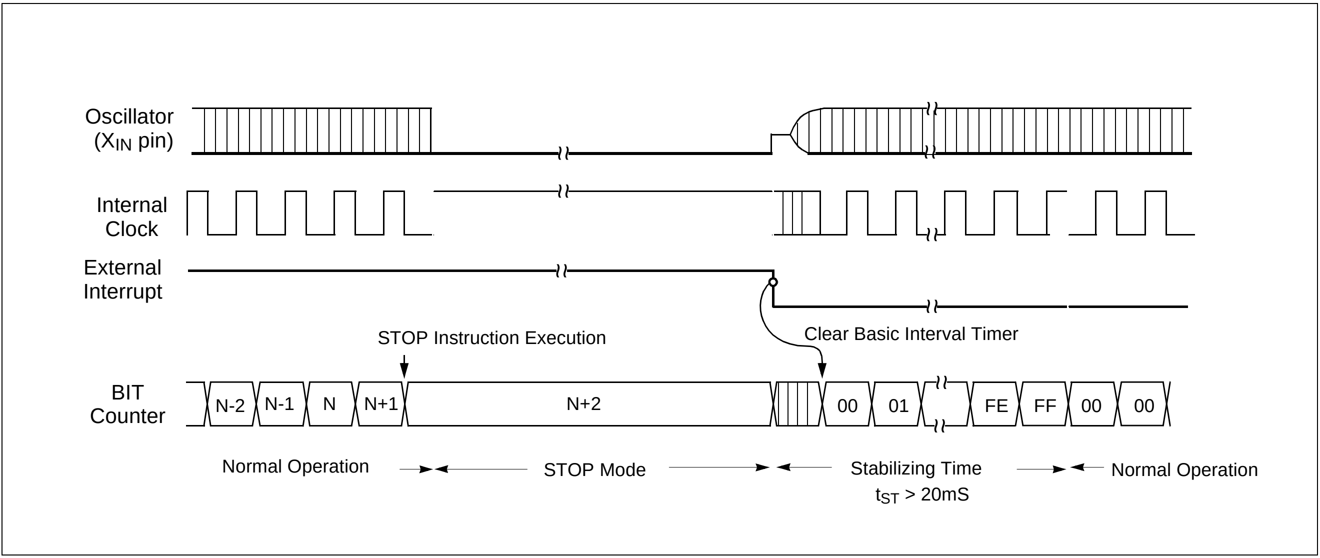


Figure 18-2 Timing of STOP Mode Release by External Interrupt

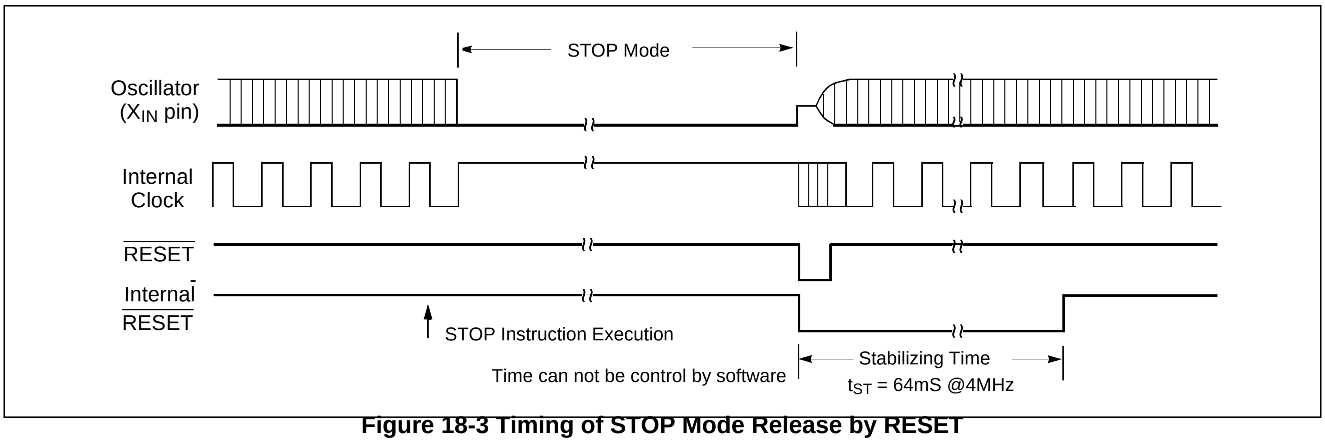

STOP Mode
Oscillator (X_{IN} pin)
Internal Clock
RESET
Internal RESET
STOP Instruction Execution
Time can not be control by software
Stabilizing Time
t_{ST} = 64mS @4MHz

Figure 18-3 Timing of STOP Mode Release by RESET

18.2 STOP Mode using Internal RCWDT

In the STOP mode using Internal RC-Oscillated Watchdog Timer, the on-chip oscillator is stopped. But internal RC oscillation circuit is oscillated in this mode. 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 Internal RC-Oscillated Watchdog Timer mode is activated by execution of STOP instruction after setting the bit RCWDT of CKCTLR to "1". (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example "set1" or "clr1" instruction, it may be undesired operation)

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

```
Ex)        LDM  WDTR,#1111_1111B
           LDM  CKCTLR,#0010_1110B
           STOP
           NOP
           NOP
```

Release the STOP mode using internal RCWDT

The exit from STOP mode using Internal RC-Oscillated Watchdog Timer is hardware reset or external interrupt. 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.

If I-flag = 1, the normal interrupt response takes place. In this case, if the bit WDTON of CKCTLR is set to "0" and the bit WDTE of IENH is set to "1", the device will execute the watchdog timer interrupt service routine.(Figure 18-4) However, if the bit WDTON of CKCTLR is set to "1", the device will generate the internal RESET signal and execute the reset processing. (Figure 18-5)

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 18-1)

When exit from STOP mode using Internal RC-Oscillated Watchdog Timer by external interrupt, the oscillation stabilization time is required to normal operation. Figure 18-4 shows the timing diagram. When release the Internal RC-Oscillated Watchdog Timer 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 using internal RC-Oscillated Watchdog Timer is shown in Figure 18-5 .

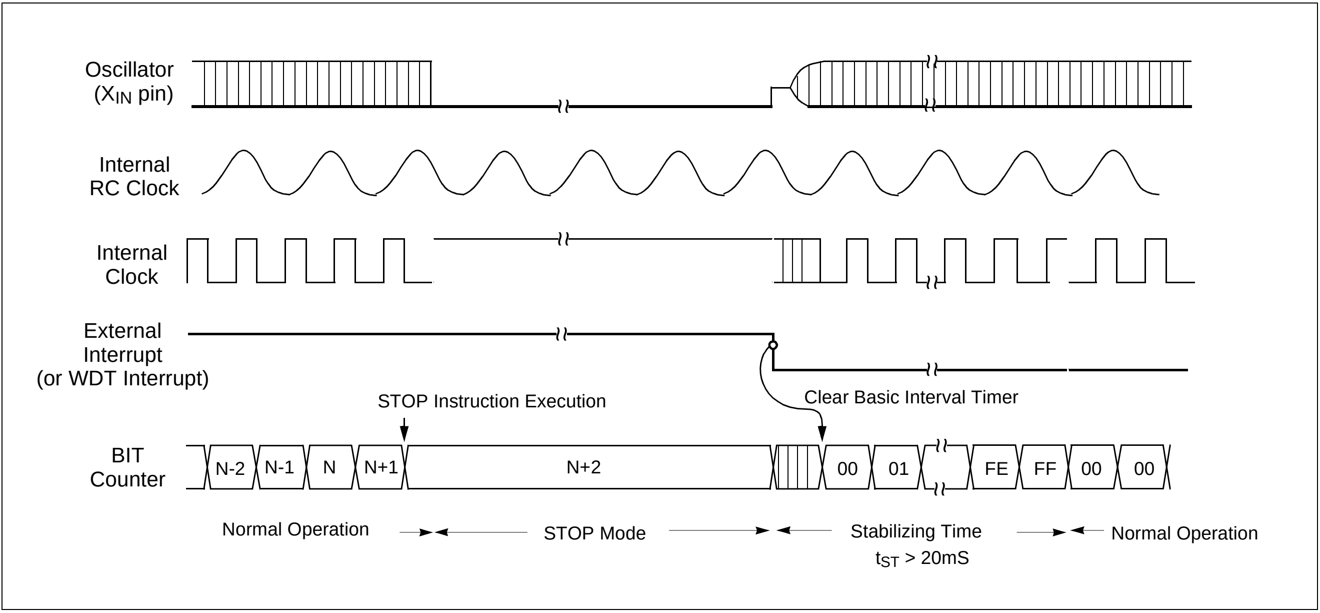


Figure 18-4 STOP Mode Releasing by External Interrupt or WDT Interrupt(using RCWDT)

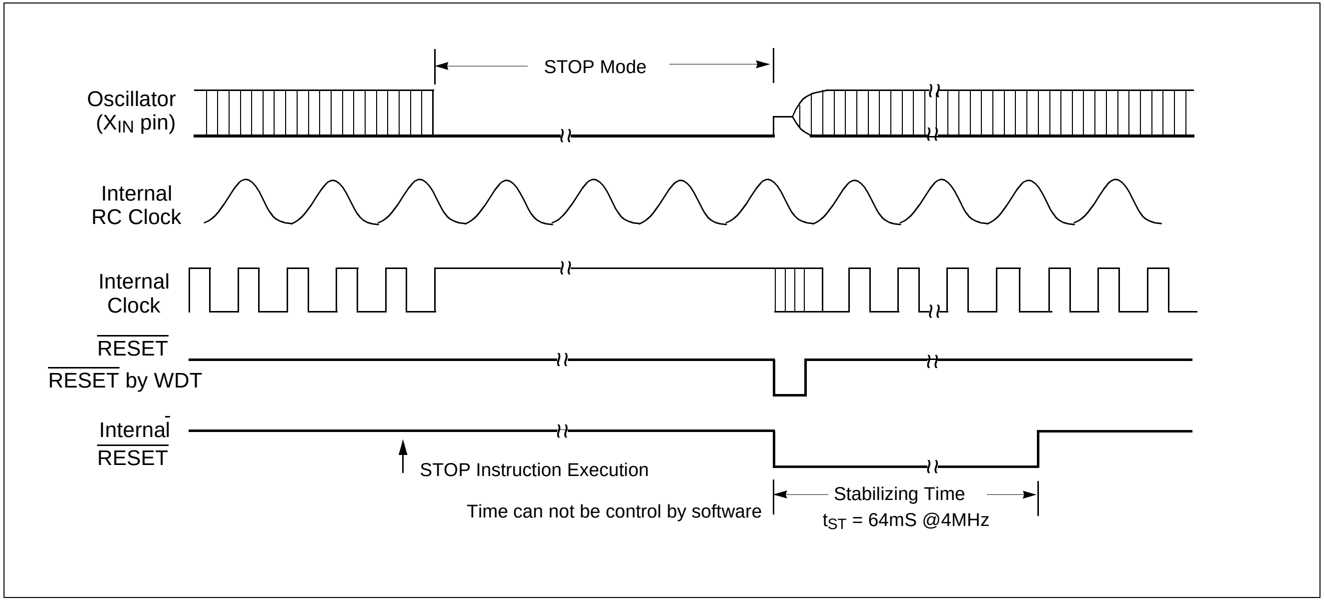


Figure 18-5 STOP Mode Releasing by RESET(using RCWDT)

18.3 Wake-up Timer Mode

In the Wake-up Timer mode, the on-chip oscillator is not stopped. Except the Prescaler(only 2048 devided ratio), Timer0 and Timer2, 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 Wake-up Timer mode is activated by execution of STOP instruction after setting the bit WAKEUP of CKCTLR to “1”. (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example “set1” or “clr1” instruction, it may be undesired operation)

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

```
Ex)      LDM  TDR0,#0FFH
         LDM  TM0,#0001_1011B
         LDM  CKCTLR,#0100_1110B
         STOP
         NOP
         NOP
```

In addition, the clock source of timer0 and timer2 should be selected to 2048 devided ratio. Otherwise, the wake-up function can not work. And the timer0 and timer2 can be operated as 16-bit timer with timer1 and timer3(refer to timer function). The period of wake-up function is varied by setting the timer data register0, TDR0 or timer data register2, TDR2.

Release the Wake-up Timer mode

The exit from Wake-up Timer mode is hardware reset, Timer0(Timer2) overflow or external interrupt. Reset redefines all the Control registers but does not change the on-chip RAM. External interrupts and Timer0(Timer2) overflow allow both on-chip RAM and Control registers to retain their values.

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 18-1)

When exit from Wake-up Timer mode by external interrupt or timer0(Timer2) overflow, the oscillation stabilizing time is not required to normal operation. Because this mode do not stop the on-chip oscillator shown as Figure 18-6 .

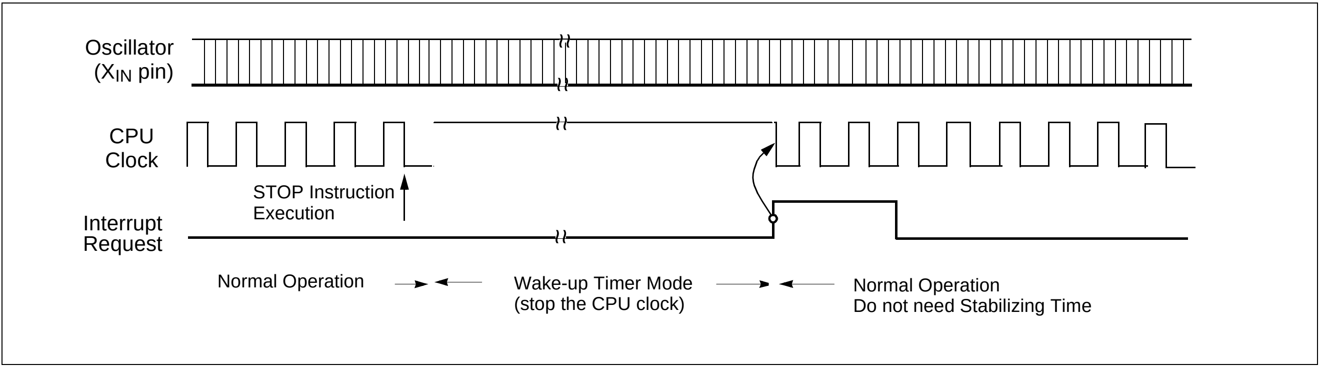


Figure 18-6 Wake-up Timer Mode Releasing by External Interrupt or Timer0(Timer2) Interrupt

18.4 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.

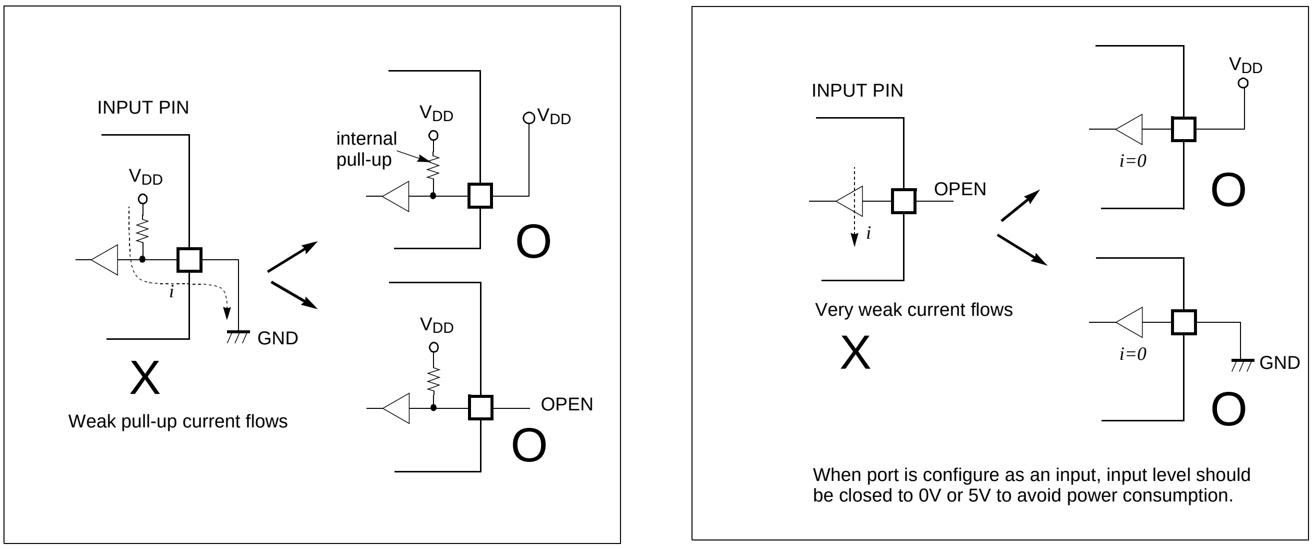
***Note:** In the STOP operation, the power dissipation associated with the oscillator and the internal hardware is lowered; however, the power dissipation associated with the pin interface (depending on the external circuitry and program) is not directly determined by the hardware operation of the STOP 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 conseider 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.

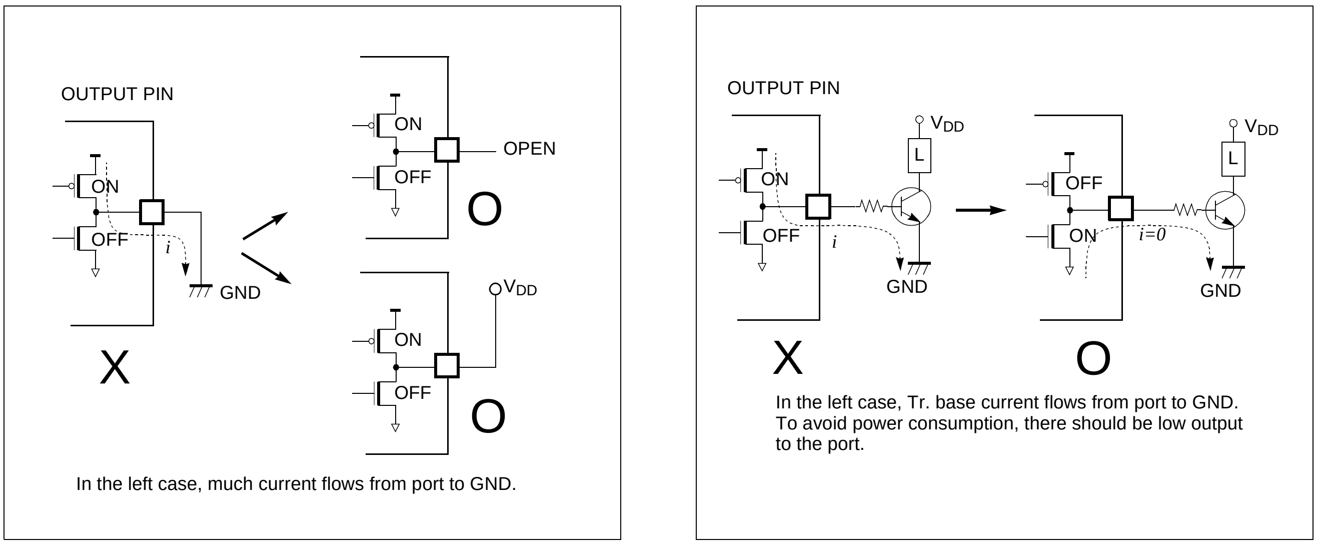


Weak pull-up current flows

Very weak current flows

When port is configure as an input, input level should be closed to 0V or 5V to avoid power consumption.

Figure 18-7 Application Example of Unused Input Port



In the left case, much current flows from port to GND.

In the left case, Tr. base current flows from port to GND. To avoid power consumption, there should be low output to the port.

Figure 18-8 Application Example of Unused Input Port

19. RESET

The reset input is the RESET pin, which is the input to a Schmitt Trigger. A reset in accomplished by holding the RESET pin low for at least 8 oscillator periods, while the oscillator running. After reset, 64ms (at 4 MHz) add with 7 oscillator periods are required to start execution as shown in Figure 19-1 .

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 .

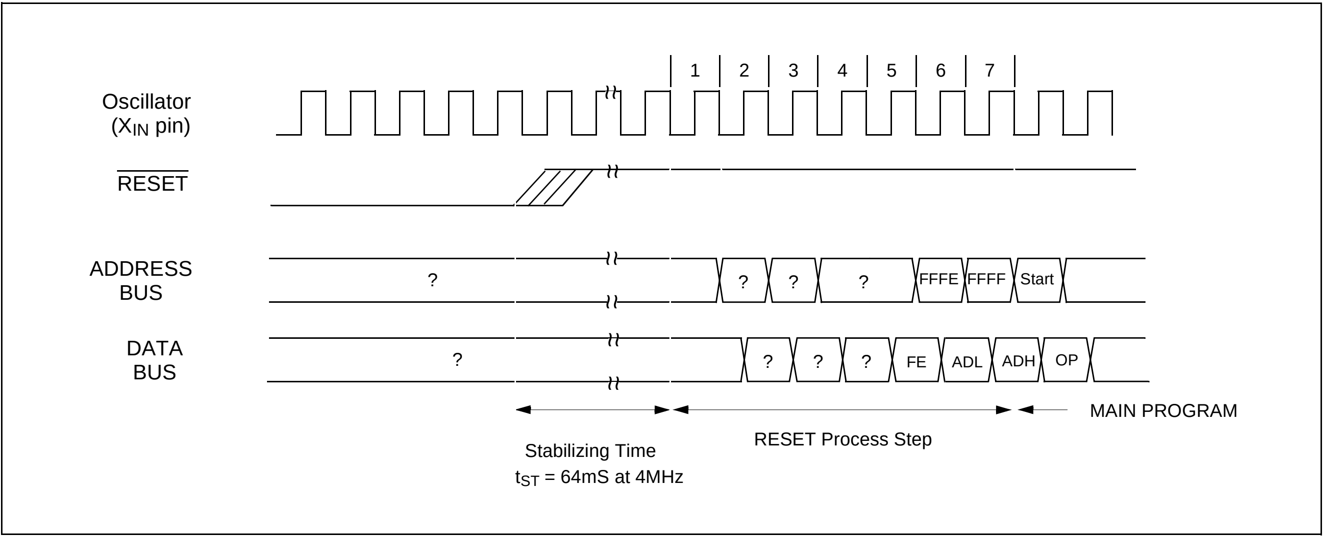


Figure 19-1 Timing Diagram after RESET

20. POWER FAIL PROCESSOR

The GMS87C1408 and GMS87C1404 has an on-chip power fail detection circuitry to immunize against power noise. A configuration register, PFDR, can enable (if clear/ programmed) or disable (if set) the Power-fail Detect circuitry. If V_{DD} falls below 2.5~3.5V(2.0~3.0V) range for longer than 50 nS, the Power fail situation may reset MCU according to PFR bit of PFDR.

As below PFDR register is not implemented on the in-circuit emulator, user can not experiment with it. Therefore, after final development of user program, this function may be experimented.

Note: *Power fail detect level is decided by setting the bit PFDLEVEL of CONFIG register (refer to Figure 21-1).*

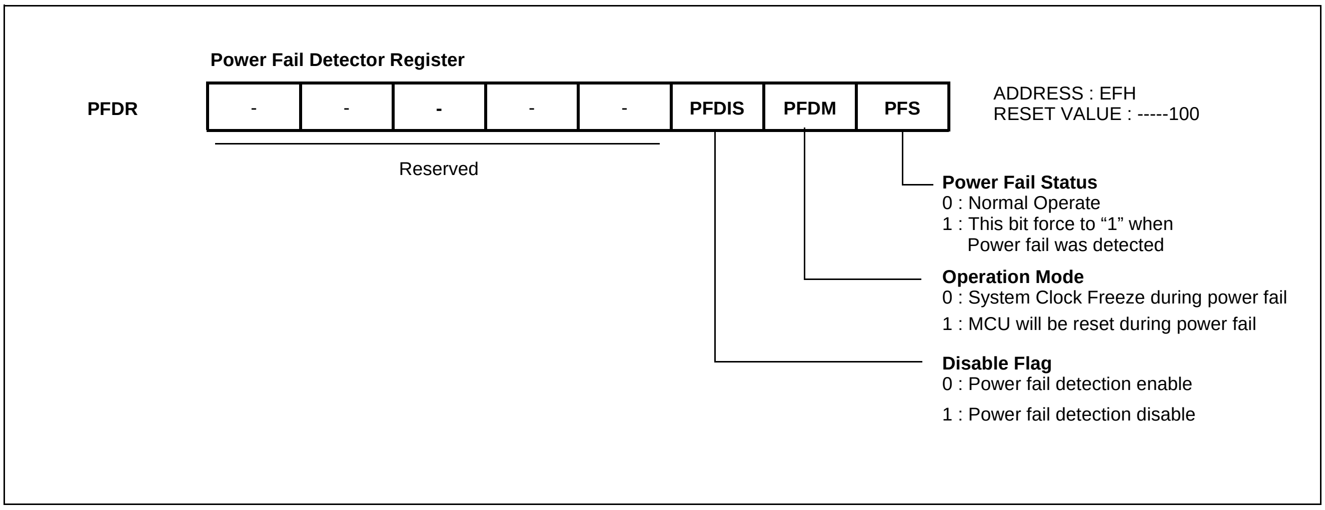

Power Fail Detector Register
PFDR
-
-
-
-
-
PFDIS
PFDM
PFS
ADDRESS : EFH
RESET VALUE : -----100
Reserved
Power Fail Status
0 : Normal Operate
1 : This bit force to “1” when Power fail was detected
Operation Mode
0 : System Clock Freeze during power fail
1 : MCU will be reset during power fail
Disable Flag
0 : Power fail detection enable
1 : Power fail detection disable

Figure 20-1 Power Fail Detector Register

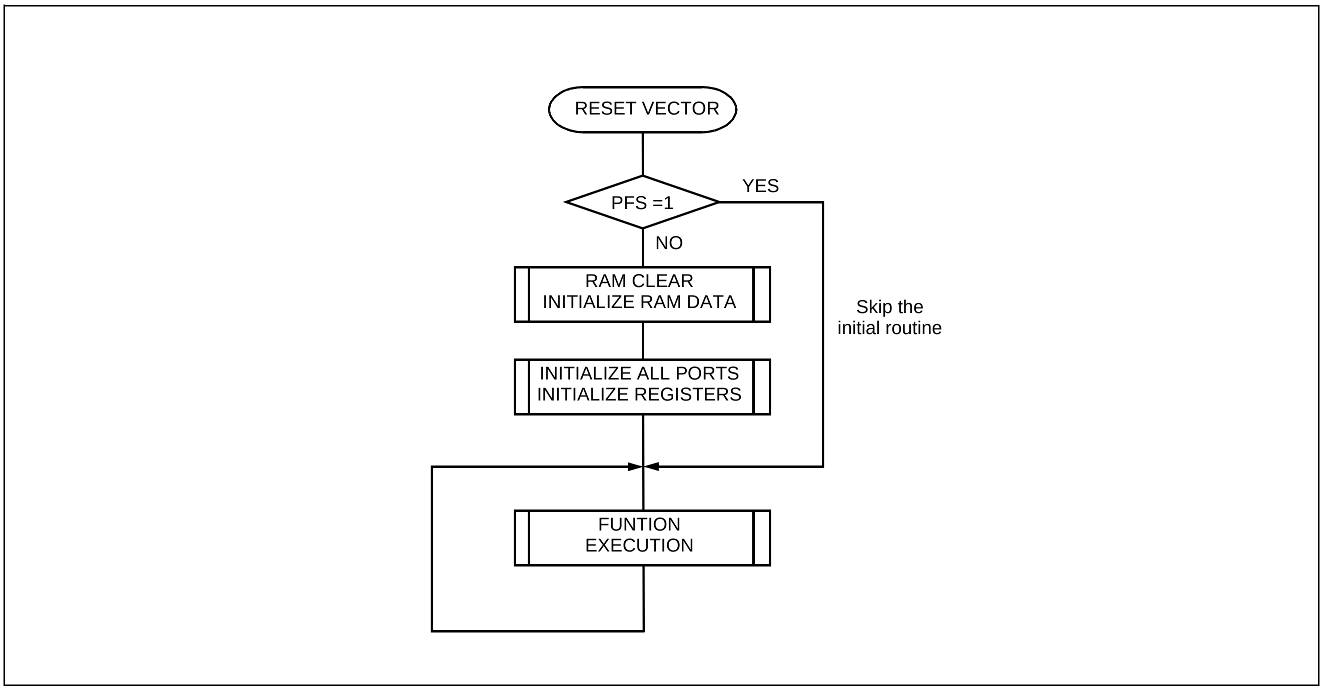


Figure 20-2 Example S/W of RESET by Power fail

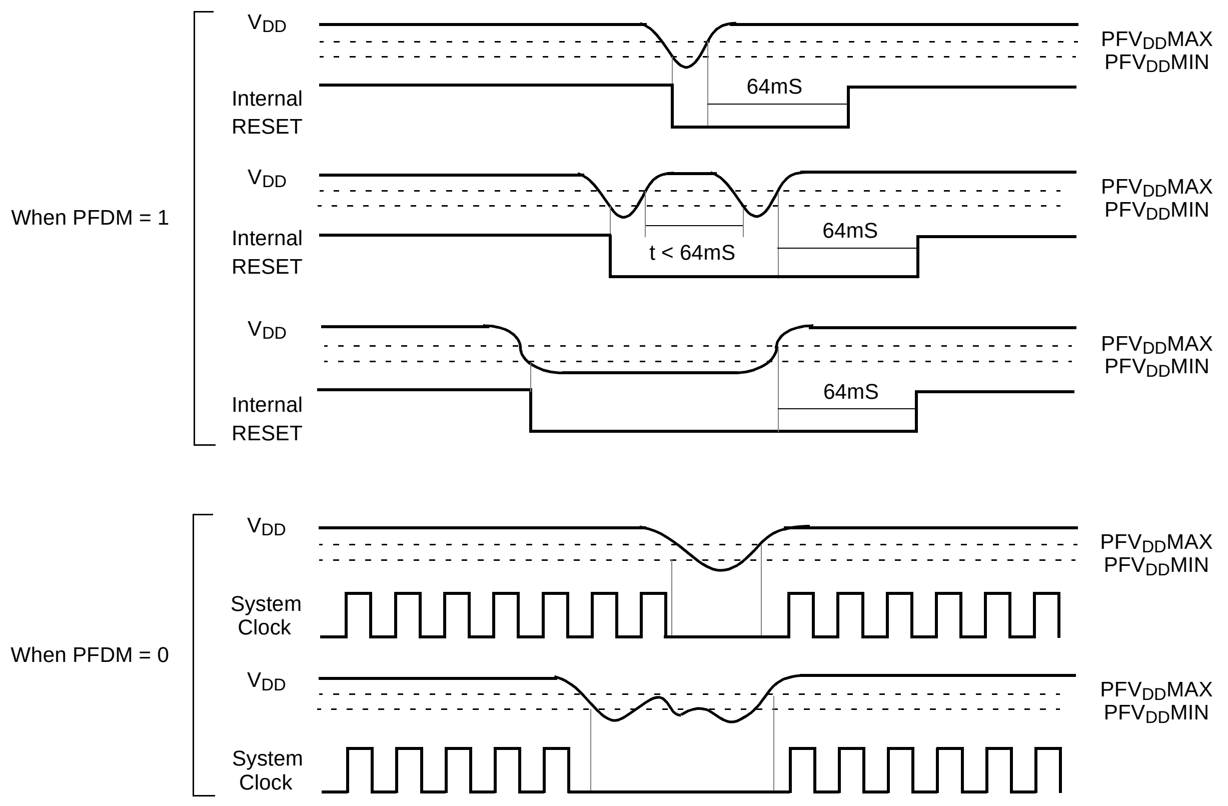


Figure 20-3 Power Fail Processor Situations

21. OTP PROGRAMMING

21.1 DEVICE CONFIGURATION AREA

The Device Configuration Area can be programmed or left unprogrammed to select device configuration such as security bit.

Ten memory locations ($0F50_H$ ~ $0FE0_H$) are designated as Customer ID recording locations where the user can store check-sum or other customer identification numbers.

This area is not accessible during normal execution but is readable and writable during program / verify.

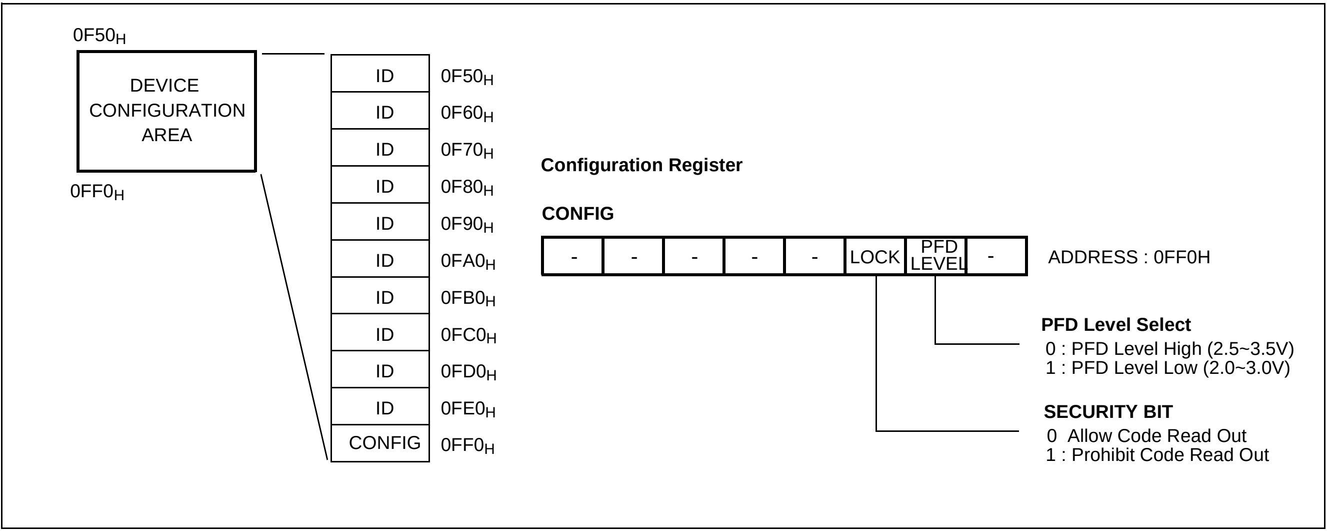


Figure 21-1 Device Configuration Area

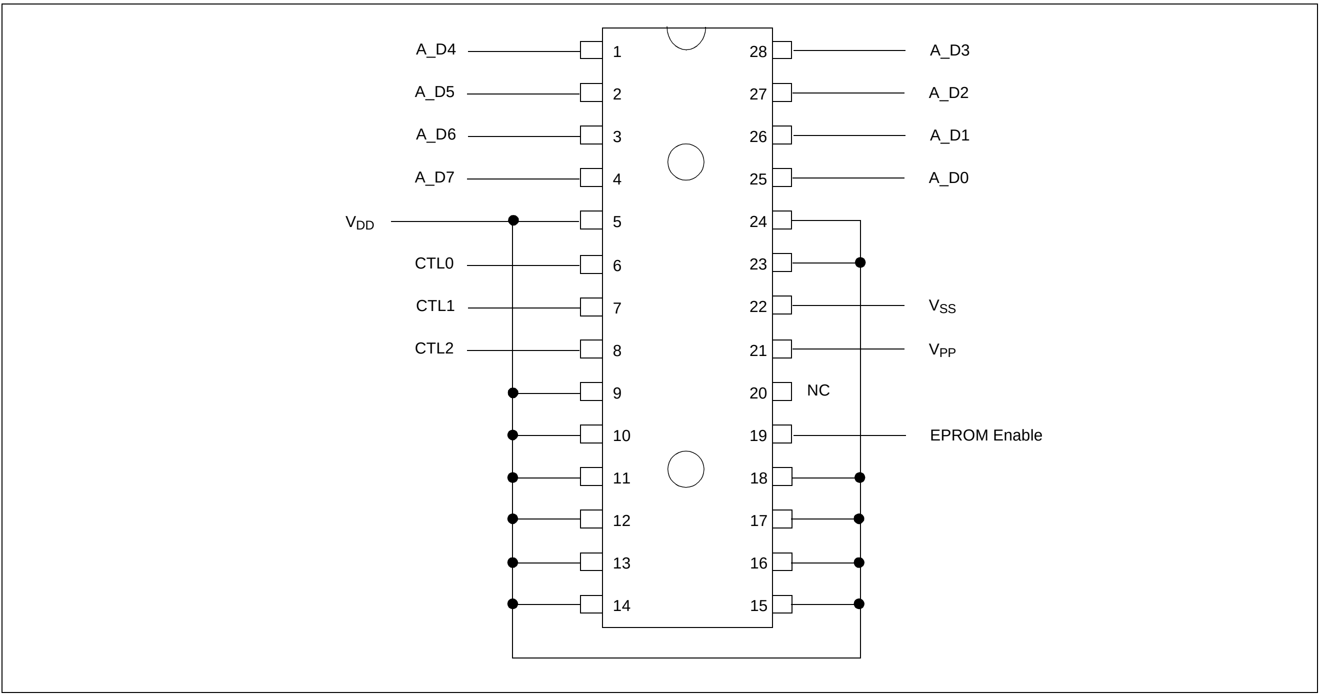


Figure 21-2 Pin Assignment

Pin No.	User Mode	EPROM MODE				
	Pin Name	Pin Name	Description			
1	RA4 (AN4)	A_D4	Address Input Data Input/Output	A12	A4	D4
2	RA5 (AN5)	A_D5		A13	A5	D5
3	RA6 (AN6)	A_D6		A14	A6	D6
4	RA7 (AN7)	A_D7		A15	A7	D7
5	V_{DD}	V_{DD}	Connect to V_{DD} (6.0V)			
6	RB0 (AVref/AN0)	CTL0	Read/Write Control Address/Data Control			
7	RB1 (INT0)	CTL1				
8	RB2 (INT1)	CTL2				
9~18	RB3~7, RC3~6, RD2	V_{DD}	Connect to V_{DD} (6.0V)			
19	X_{IN}	EPROM Enable	High Active, Latch Address in falling edge			
20	X_{OUT}	NC	No connection			
21	$\overline{\text{RESET}}$	V_{PP}	Programming Power (0V, 12.75V)			
22	V_{SS}	V_{SS}	Connect to V_{SS} (0V)			
23, 24	RC0, 1	V_{DD}	Connect to V_{DD} (6.0V)			
25	RA0 (EC0)	A_D0	Address Input Data Input/Output	A8	A0	D0
26	RA1 (AN1)	A_D1		A9	A1	D1
27	RA2 (AN2)	A_D2		A10	A2	D2
28	RA3 (AN3)	A_D3		A11	A3	D3

Table 21-1 Pin Description in EPROM Mode

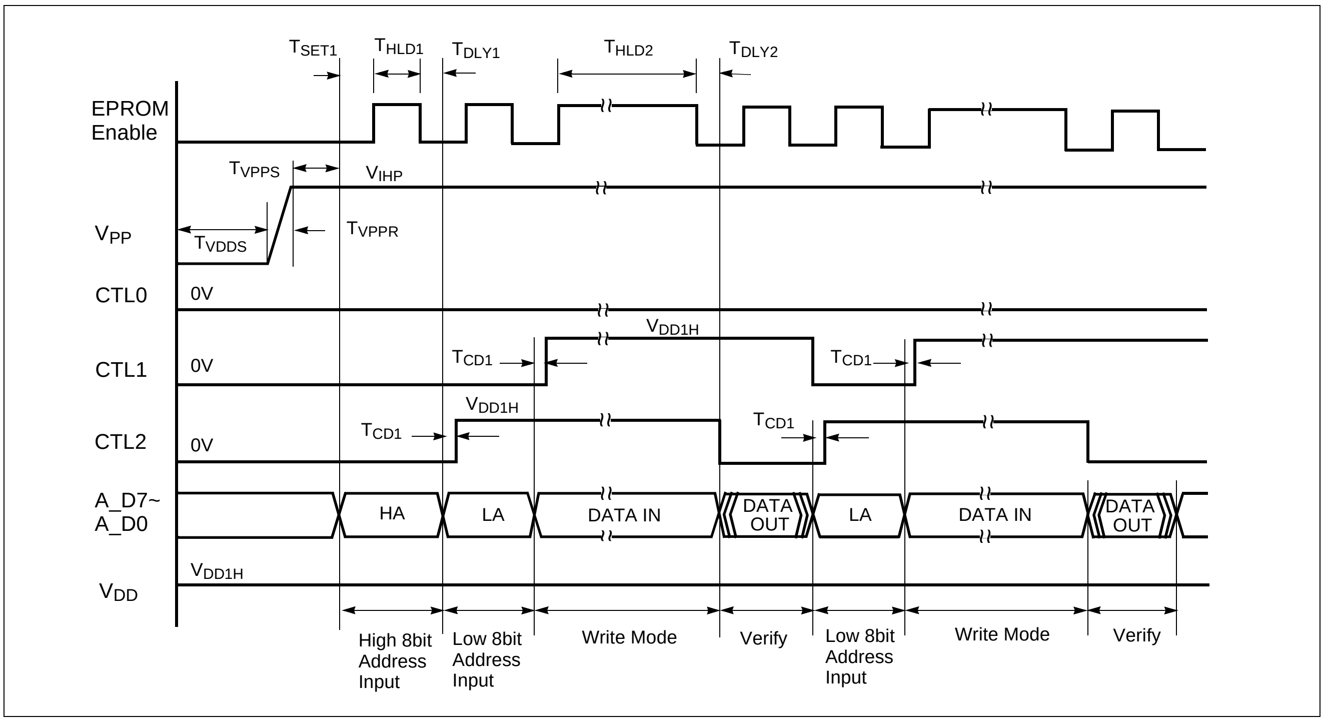


Figure 21-3 Timing Diagram in Program (Write & Verify) Mode

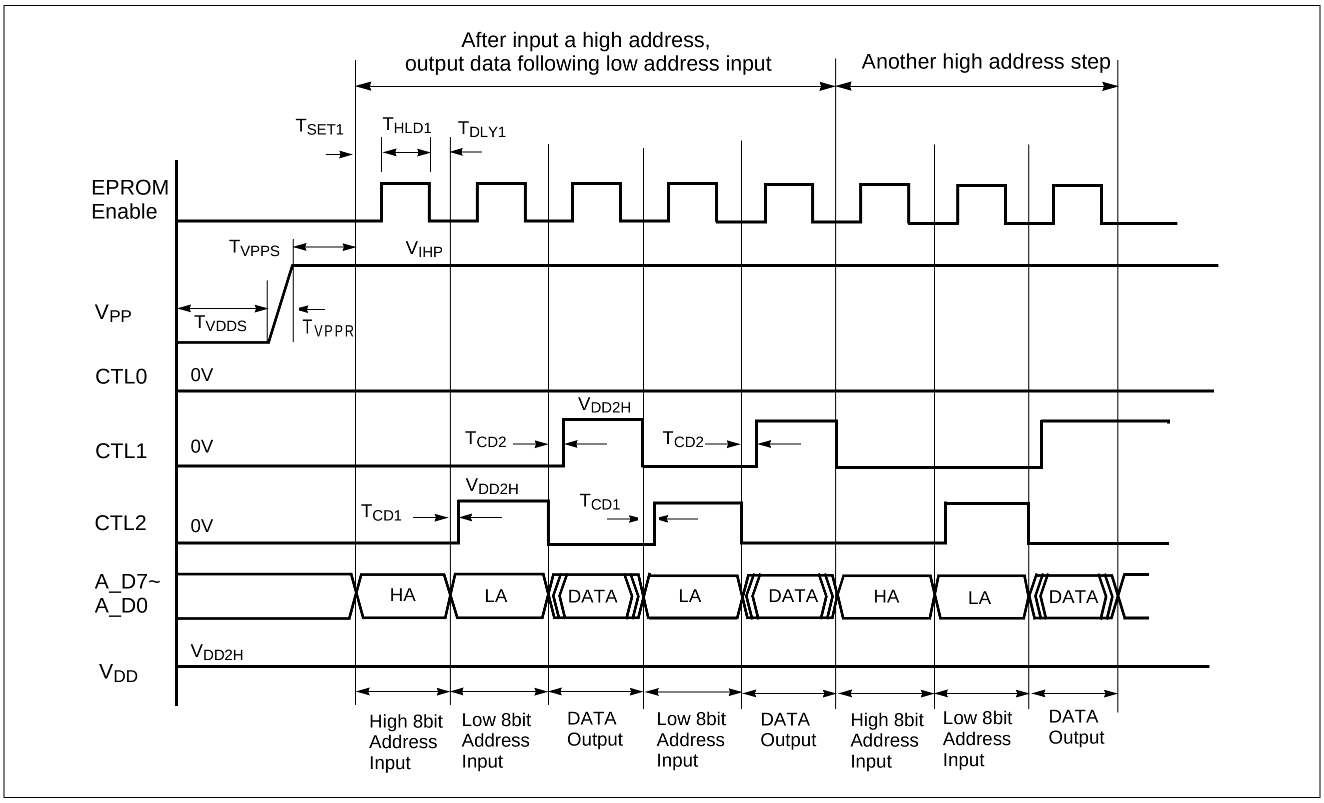


Figure 21-4 Timing Diagram in READ Mode

Parameter	Symbol	MIN	TYP	MAX	Unit
Programming Supply Current	I_{VPP}	-	-	50	mA
Supply Current in EPROM Mode	I_{VDDP}	-	-	20	mA
V_{PP} Level during Programming	V_{IHP}	11.5	12.0	12.5	V
V_{DD} Level in Program Mode	V_{DD1H}	5	6	6.5	V
V_{DD} Level in Read Mode	V_{DD2H}	-	2.7	-	V
CTL2~0 High Level in EPROM Mode	V_{IHC}	$0.8V_{DD}$	-	-	V
CTL2~0 Low Level in EPROM Mode	V_{ILC}	-	-	$0.2V_{DD}$	V
A_D7~A_D0 High Level in EPROM Mode	V_{IHAD}	$0.9V_{DD}$	-	-	V
A_D7~A_D0 Low Level in EPROM Mode	V_{ILAD}	-	-	$0.1V_{DD}$	V
V_{DD} Saturation Time	T_{VDDS}	1	-	-	mS
V_{PP} Setup Time	T_{VPPR}	-	-	1	mS
V_{PP} Saturation Time	T_{VPPS}	1	-	-	mS
EPROM Enable Setup Time after Data Input	T_{SET1}		200		nS
EPROM Enable Hold Time after T_{SET1}	T_{HLD1}		500		nS
EPROM Enable Delay Time after T_{HLD1}	T_{DLY1}		200		nS
EPROM Enable Hold Time in Write Mode	T_{HLD2}		100		nS
EPROM Enable Delay Time after T_{HLD2}	T_{DLY2}		200		nS
CTL2,1 Setup Time after Low Address input and Data input	T_{CD1}		100		nS
CTL1 Setup Time before Data output in Read and Verify Mode	T_{CD2}		100		nS

Table 21-2 AC/DC Requirements for Program/Read Mode

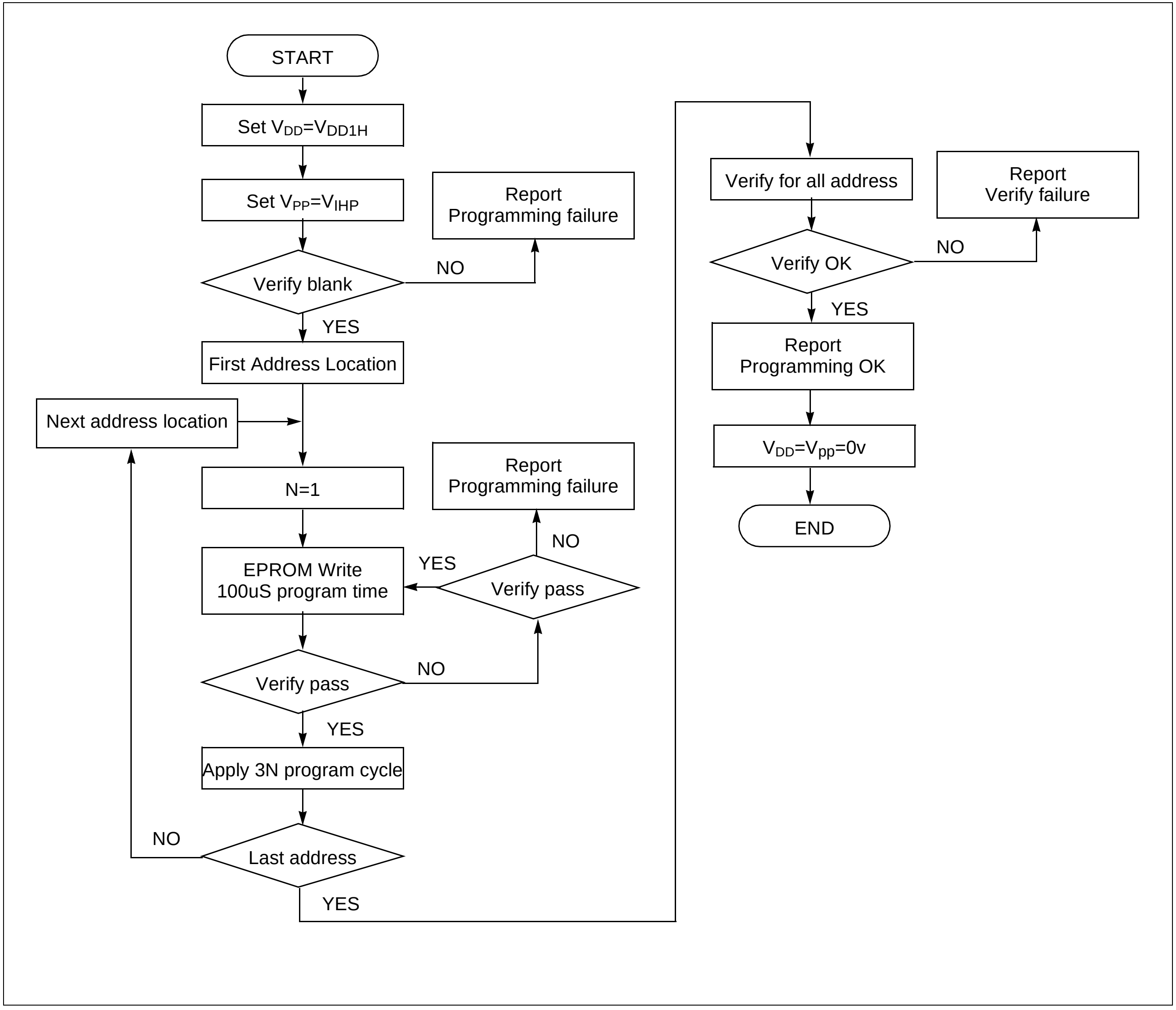


Figure 21-5 Programming Flow Chart

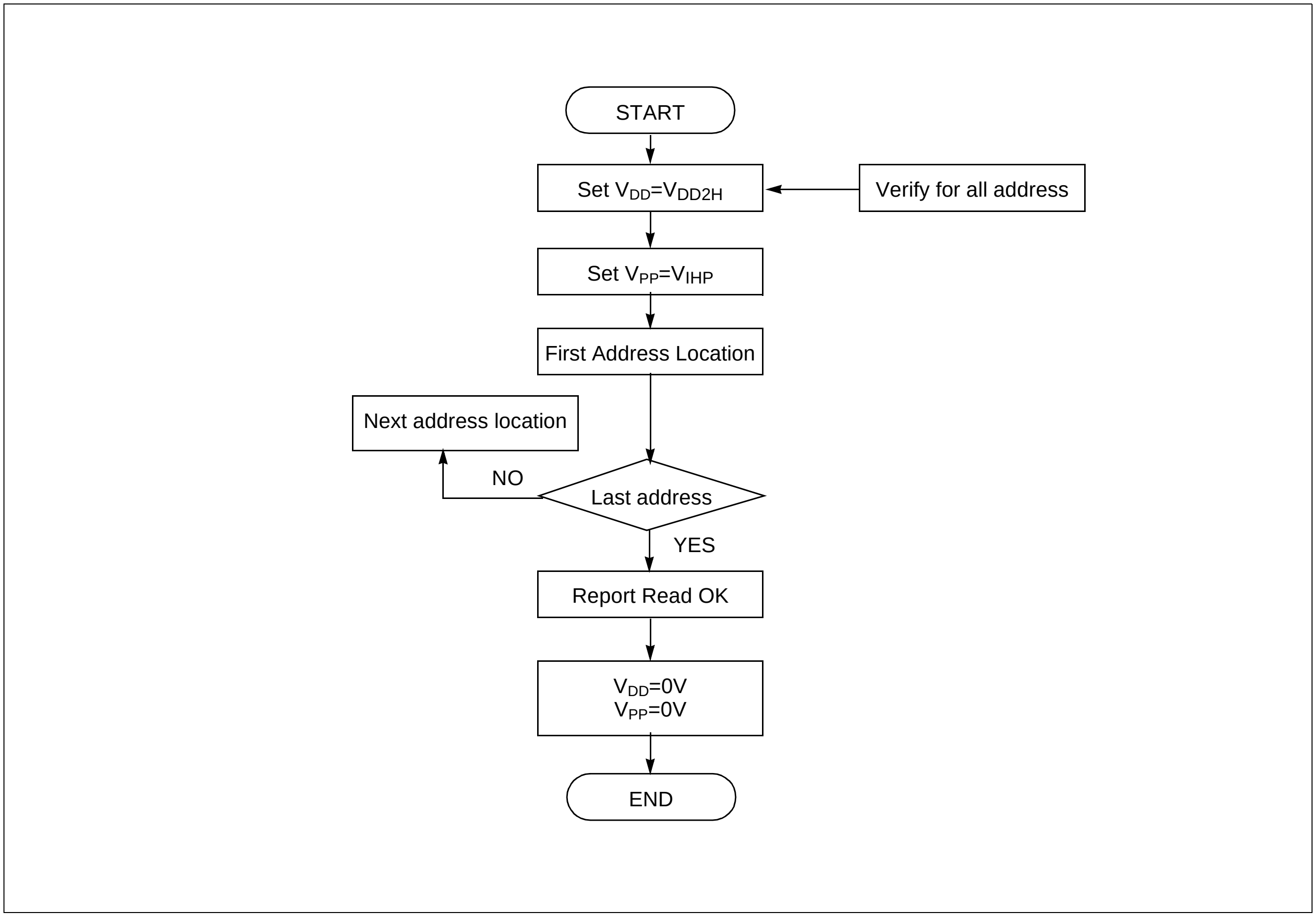


Figure 21-6 Reading Flow Chart

APPENDIX

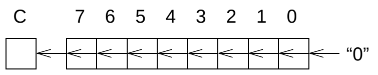
A. INSTRUCTION MAP

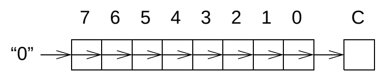
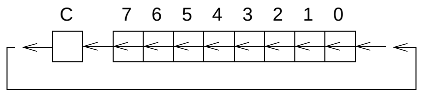
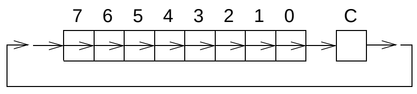
LOW HIGH	00000 00	00001 01	00010 02	00011 03	00100 04	00101 05	00110 06	00111 07	01000 08	01001 09	01010 0A	01011 0B	01100 0C	01101 0D	01110 0E	01111 0F
000	-	SET1 dp.bit	BBS A.bit,rel	BBS dp.bit,rel	ADC #imm	ADC dp	ADC dp+X	ADC !abs	ASL A	ASL dp	TCALL 0	SETA1 .bit	BIT dp	POP A	PUSH A	BRK
001	CLRC				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	XAX	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

B. INSTRUCTION SET

1. ARITHMETIC/ LOGIC OPERATION

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 ← (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	AND #imm	84	2	2	Logical AND	N-----Z-
10	AND dp	85	2	3	A ← (A) ∧ (M)	
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	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	N-----ZC
22	CMP dp	45	2	3	(A) - (M)	
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	N-----ZC
30	CMPX dp	6C	2	3	(X) - (M)	
31	CMPX !abs	7C	3	4		
32	CMPY #imm	7E	2	2	Compare Y contents with memory contents	N-----ZC
33	CMPY dp	8C	2	3	(Y) - (M)	
34	CMPY !abs	9C	3	4		
35	COM dp	2C	2	4	1’S Complement : (dp) ← ~(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	N-----Z-
39	DEC dp	A9	2	4	M ← (M) - 1	
40	DEC dp + X	B9	2	5		N-----Z-
41	DEC !abs	B8	3	5		
42	DEC X	AF	1	2		
43	DEC Y	BE	1	2		
44	DIV	9B	1	12	Divide : YA / X Q: A, R: Y	NV--H-Z-

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
45	EOR #imm	A4	2	2	Exclusive OR	
46	EOR dp	A5	2	3	$A \leftarrow (A) \oplus (M)$	
47	EOR dp + X	A6	2	4		
48	EOR !abs	A7	3	4		N-----Z-
49	EOR !abs + Y	B5	3	5		
50	EOR [dp + X]	B6	2	6		
51	EOR [dp] + Y	B7	2	6		
52	EOR { X }	B4	1	3		
53	INC A	88	1	2	Increment	N-----Z-
54	INC dp	89	2	4	$M \leftarrow (M) + 1$	
55	INC dp + X	99	2	5		N-----Z-
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	
60	LSR dp	49	2	4		N-----ZC
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	
65	OR dp	65	2	3	$A \leftarrow (A) \vee (M)$	
66	OR dp + X	66	2	4		
67	OR !abs	67	3	4		N-----Z-
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	
73	ROL dp	29	2	4		N-----ZC
74	ROL dp + X	39	2	5		
75	ROL !abs	38	3	5		
76	ROR A	68	1	2	Rotate right through carry	
77	ROR dp	69	2	4		N-----ZC
78	ROR dp + X	79	2	5		
79	ROR !abs	78	3	5		
80	SBC #imm	24	2	2	Subtract with carry	
81	SBC dp	25	2	3	$A \leftarrow (A) - (M) - \sim(C)$	
82	SBC dp + X	26	2	4		
83	SBC !abs	27	3	4		NV--HZC
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-

2. REGISTER / MEMORY OPERATION

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

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 ← (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) ← (dp+1) (dp) - 1	N-----Z-
4	INCW dp	9D	2	6	Increment memory pair (dp+1) (dp) ← (dp+1) (dp) + 1	N-----Z-
5	LDYA dp	7D	2	5	Load YA YA ← (dp +1) (dp)	N-----Z-
6	STYA dp	DD	2	5	Store YA (dp +1) (dp) ← YA	--------
7	SUBW dp	3D	2	5	16-Bits substact without carry YA ← (YA) - (dp +1) (dp)	NV--H-ZC

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 ← (C) $\wedge$ (M .bit)	-------C
2	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : C ← (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 ← (A) $\wedge$ (M) , N ← (M_7) , V ← (M_6)	
5	CLR1 dp.bit	y1	2	4	Clear bit : (M.bit) ← “0”	--------
6	CLRA1 A.bit	2B	2	2	Clear A bit : (A.bit)← “0”	--------
7	CLRC	20	1	2	Clear C-flag : C ← “0”	-------0
8	CLRG	40	1	2	Clear G-flag : G ← “0”	--0-----
9	CLRV	80	1	2	Clear V-flag : V ← “0”	-0--0---
10	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : C ← (C) ⊕ (M .bit)	-------C
11	EOR1B M.bit	AB	3	5	Bit exclusive-OR C-flag and NOT : C ← (C) ⊕ ~(M .bit)	-------C
12	LDC M.bit	CB	3	4	Load C-flag : C ← (M .bit)	-------C
13	LDCB M.bit	CB	3	4	Load C-flag with NOT : C ← ~(M .bit)	-------C
14	NOT1 M.bit	4B	3	5	Bit complement : (M .bit) ← ~(M .bit)	--------
15	OR1 M.bit	6B	3	5	Bit OR C-flag : C ← (C) $\vee$ (M .bit)	-------C
16	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : C ← (C) $\vee$ ~(M .bit)	-------C
17	SET1 dp.bit	x1	2	4	Set bit : (M.bit) ← “1”	--------
18	SETA1 A.bit	0B	2	2	Set A bit : (A.bit) ← “1”	--------
19	SETC	A0	1	2	Set C-flag : C ← “1”	-------1
20	SETG	C0	1	2	Set G-flag : G ← “1”	--1-----
21	STC M.bit	EB	3	6	Store C-flag : (M .bit) ← C	--------
22	TCLR1 !abs	5C	3	6	Test and clear bits with A : A - (M) , (M) ← (M) $\wedge$ ~(A)	N-----Z-
23	TSET1 !abs	3C	3	6	Test and set bits with A : A - (M) , (M) ← (M) $\vee$ (A)	N-----Z-

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 set :	- - - - - - - -
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	- - - - - - - -
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 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 upage	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 ← "$0FF_H$" .	- - - - - - - -
24	TCALL n	nA	1	8	Table call : (sp) ←(pc_H), sp ← sp - 1, M(sp) ← (pc_L),sp ← sp - 1, pc_L ← (Table vector L), pc_H ← (Table vector H)	- - - - - - - -

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 ← sp - 1, M(sp) ← (PSW), sp ← sp -1, pc_L ← ($0FFDE_H$) , pc_H ← ($0FFDF_H$) .	---1-0--
2	DI	60	1	3	Disable interrupts : I ← “0”	-----0--
3	EI	E0	1	3	Enable interrupts : I ← “1”	-----1--
4	NOP	FF	1	2	No operation	--------
5	POP A	0D	1	4	sp ← sp + 1, A ← M(sp)	--------
6	POP X	2D	1	4	sp ← sp + 1, X ← M(sp)	
7	POP Y	4D	1	4	sp ← sp + 1, Y ← M(sp)	
8	POP PSW	6D	1	4	sp ← sp + 1, PSW ← M(sp)	restored
9	PUSH A	0E	1	4	M(sp) ← A , sp ← sp - 1	--------
10	PUSH X	2E	1	4	M(sp) ← X , sp ← sp - 1	
11	PUSH Y	4E	1	4	M(sp) ← Y , sp ← sp - 1	
12	PUSH PSW	6E	1	4	M(sp) ← PSW , sp ← sp - 1	
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)	--------