HMS81C43xx/GMS87C4060

CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER FOR TELEVISION

1. OVERVIEW

1.1 Description

The HMS81C43xx/GMS87C4060 is an advanced CMOS 8-bit microcontroller with 8~32K(60K) bytes of ROM. The device is one of GMS800 family. The HYNIX's HMS81C43xx/GMS87C4060 is a powerful microcontroller which provides a highly flexible and cost effective solution to many TV applications. The HMS81C43xx/GMS87C4060 provides the following standard features: 8~32K(60K) bytes of ROM, 256~512/1,536 bytes of RAM, 8-bit timer/counter .

Device name	ROM Size	RAM Size	Package
HMS81C43xx	8~32K bytes Mask ROM	256~512 bytes	32 PDIP
GMS87C4060	60K bytes EPROM	1,536 bytes	52 SDIP

1.2 GMS87C4060 (OTP) Features

- **60K Bytes On-chip Program Memory**
- **1,536 Bytes of On-chip Data RAM (Included 256 bytes stack memory)**
- **Instruction Cycle Time (ex:NOP)**
  - **0.5us at 8MHz**
- **40 Programmable I/O pins**
  - **33 Input/Ooutput and 7 Output pins**
- **Serial I/O : 8bit x 1ch**
- **I^2C Bus interface**
  - **Multimaster (2 Pairs interface pins)**
- **A/D Converter : 8bit x 6ch (TBD LSB)**
- **Pulse Width Modulation**
  - **14bit x 1ch**
  - **8bit x 6ch**
- **Timer**
  - **Timer/Counter : 8bit x 4ch (16bit x 2ch)**
  - **Basic interval timer : 8bit x 1ch**
  - **Watch Dog Timer**
- **Number of Interrupt sources : 18**
- **On Screen Display**
  - **Number of characters : 512 (6 characters are reserved for IC test)**
  - **Character size : 12 dots(X) x 16 dots(Y)**
  - **Character display size : Large, Medium, Small**
  - **DIsplay capability : 24Characters x 16 Lines**
  - **Character, Back ground color : 16kinds**
  - **Special functions : Rounding, Outline, Sprite, Shadow, Half Tone Background,...**
- **Buzzer Driving port**
  - **500Hz ~ 250kHz @8MHz (Duty 50%)**
- **Operating Range : 4.5V to 5.5V**

1.3 HMS81C43xx Family Features

- **On-chip Program Memry and Data memry**
  HMS81C4332 (32K ROM, 512 RAM)
  HMS81C4324 (24K ROM, 256 RAM)
  HMS81C4316 (16K ROM, 256 RAM)
  HMS81C4308 (8K ROM, 256 RAM)
  (Included 64/256 bytes stack memory)
- **Instruction Cycle Time (ex:NOP)**
  - **1.0us at 4MHz**
- **21 Programmable I/O pins**
  - **19 Input/Output and 3 Output pins**
- **I^2C Bus interface**
  - **Multimaster (2 Pairs interface pins)**

- **A/D Converter : 8bit x 6ch (TBD LSB)**
- **Pulse Width Modulation**
  - **14bit x 1ch**
  - **8bit x 2ch**
- **Timer**
  - **Timer/Counter : 8bit x 4ch (16bit x 2ch)**
  - **Basic interval timer : 8bit x 1ch**
  - **Watch Dog Timer**
- **Number of Interrupt sources : 18**
- **On Screen Display**
  - **Number of characters : 256**
  - **Character size : 12 dots(X) x 16 dots(Y)**
  - **Character display size : Large, Medium, Small**
  - **DIsplay capability : 24Characters x 16 Lines**
  - **Character, Back ground color : 8 kinds**
  - **Special functions : Rounding, Outline, Shadow, Half tone Background...**
- **Buzzer Driving port**
  - **250Hz ~ 125kHz @4MHz (Duty 50%)**
- **Operating Range : 4.5V to 5.5V**
-

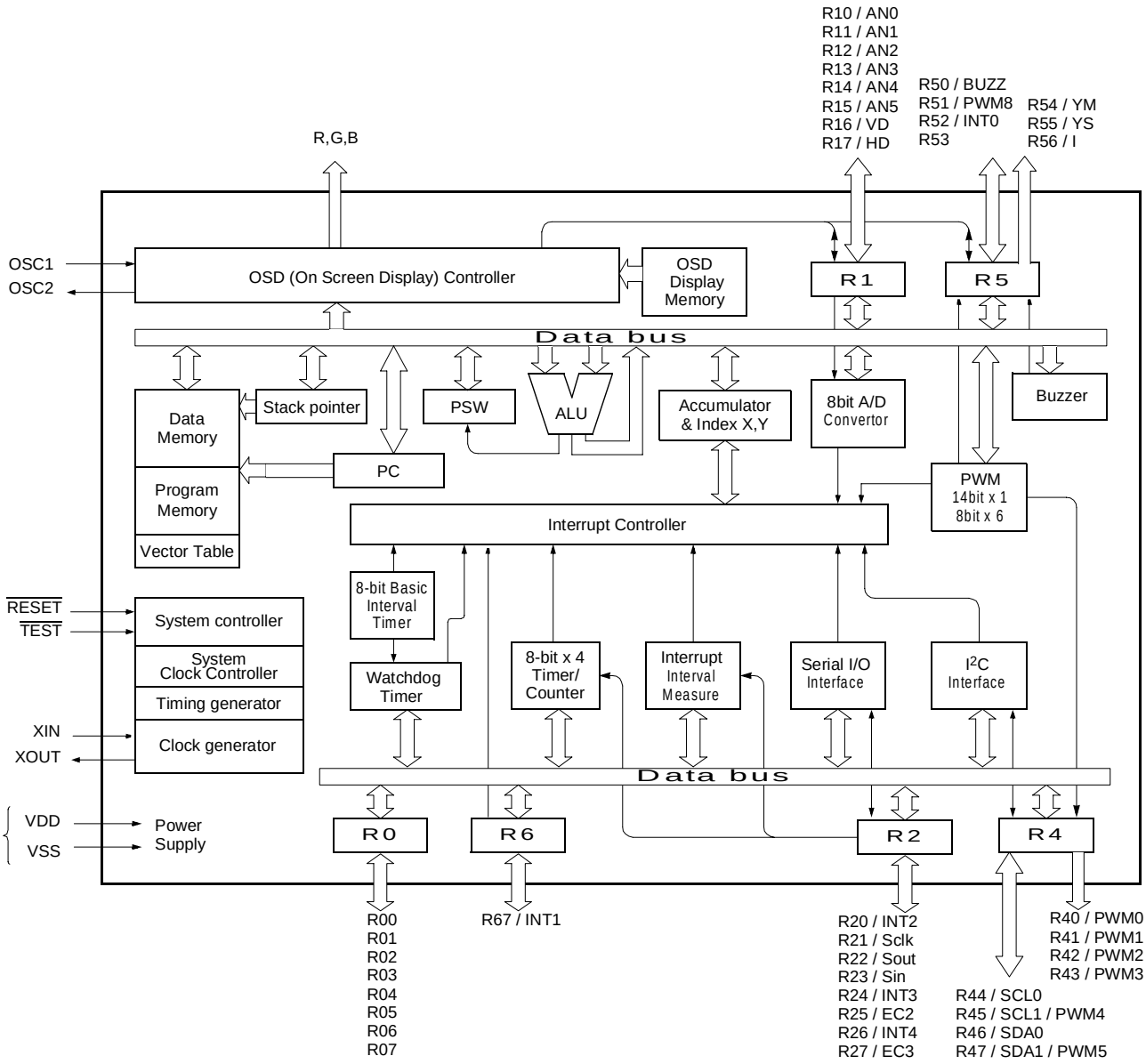
1.4 Development Tools

The HMS81C43xx/GMS87C4060 is supported by a full-featured macro assembler / linker , OSD font editor, an in-circuit emulator CHOICE-Dr™.

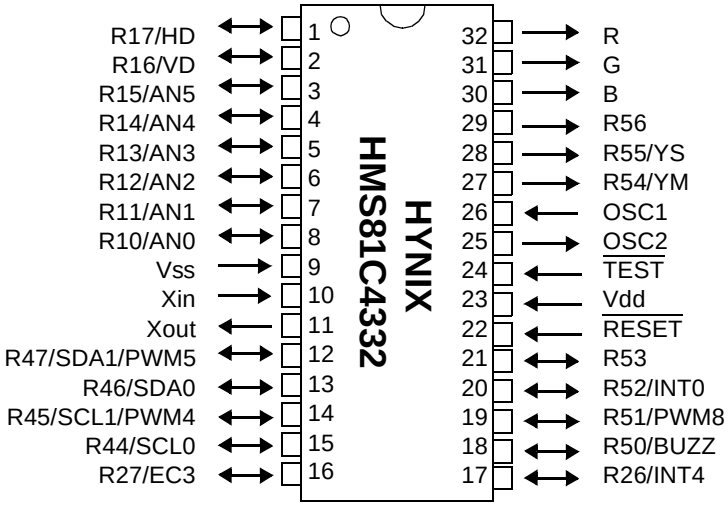
In Circuit Emulators	CHOICE-Dr. (with EVA81C4xxx board)

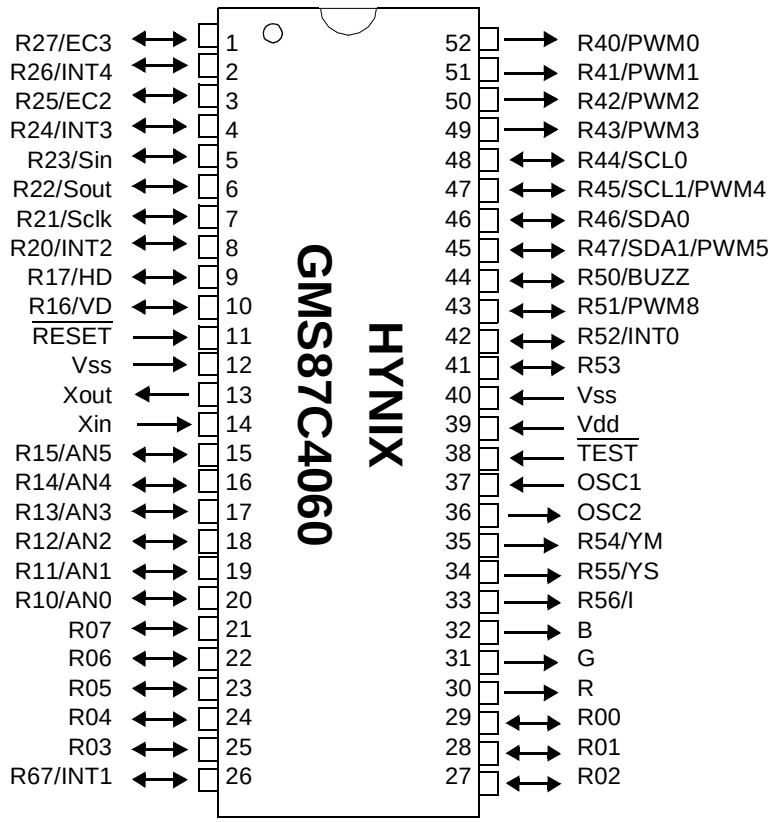
Assembler / Linker	HYNIX's Macro Assembler / Linker
Font Editor	MS-Windows GUI version
Debugger	MS-Windows GUI version

2. BLOCK DIAGRAM



3. PIN ASSIGNMENT





4. PACKAGE DIAGRAM

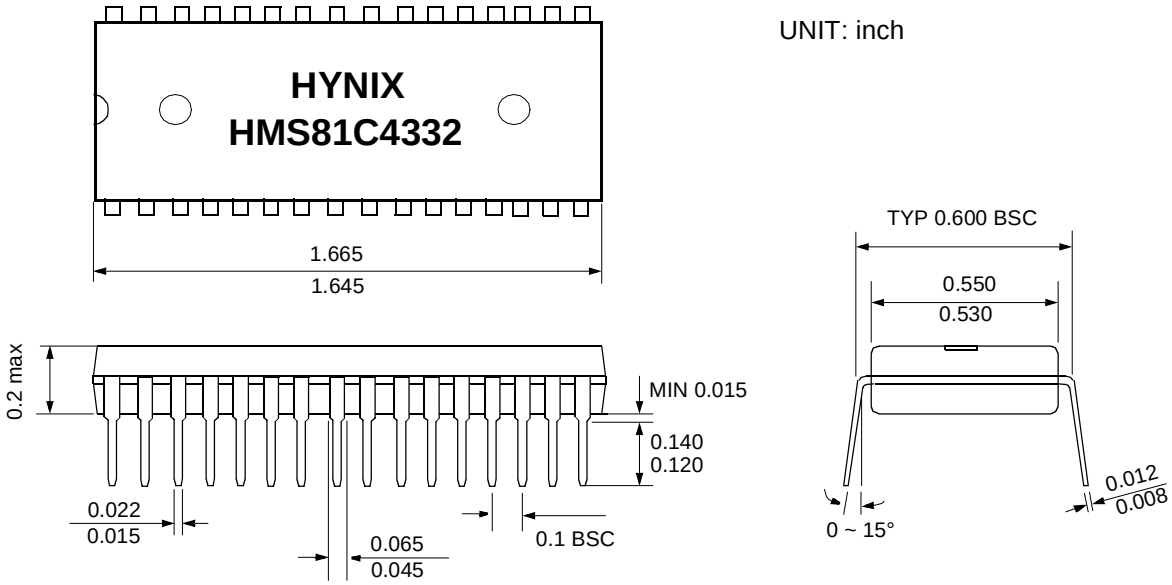
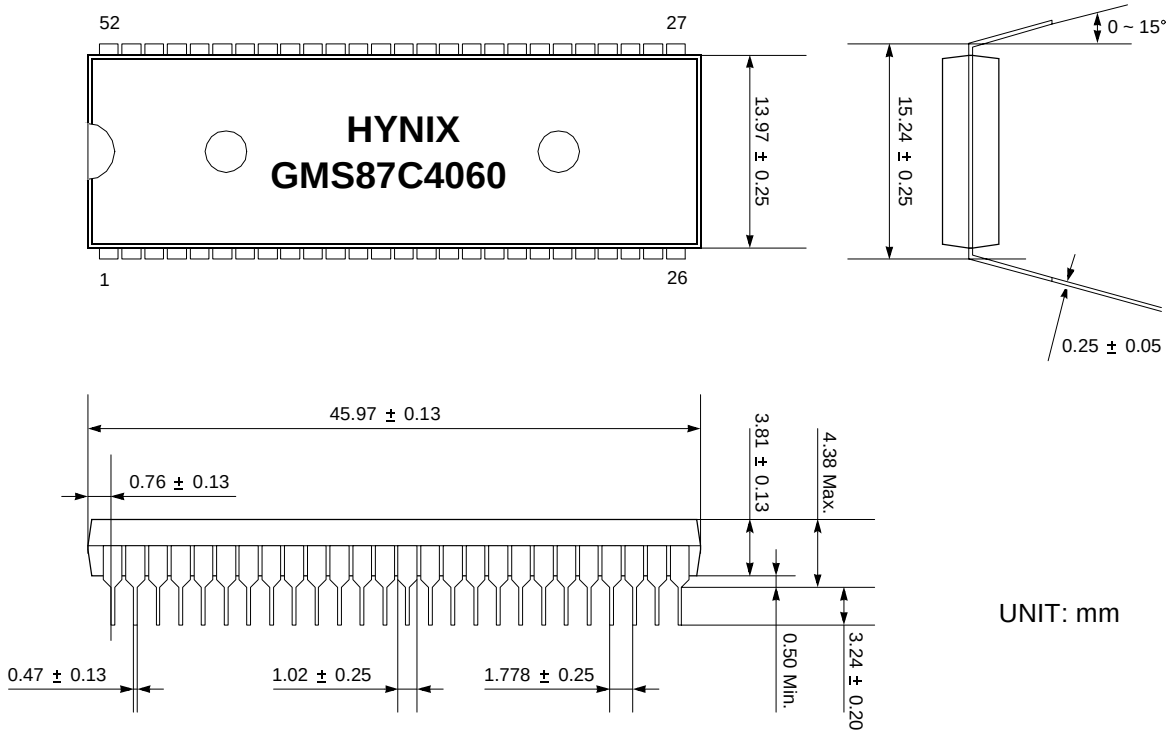


Figure 4-1 52pin Shrink DIP Package Diagram



5. PIN FUNCTION

V_{DD}: Supply voltage.

V_{SS}: Circuit ground.

$\overline{\text{TEST}}$: Used for shipping inspection of the IC. For normal operation, it should not be connected .

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

OSC1: Input to the internal On Screen Display operating circuit.

OSC2: Output from the inverting OSC1 amplifier.

R00~R07: R0 is an 8-bit CMOS bidirectional I/O port. R0 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

R10~R17: R1 is an 8-bit CMOS bidirectional I/O port. R1 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

In addition, R1 serves the functions of the various following special features.

Port pin	Alternate function
R10	AN0 (A/D converter input 0)
R11	AN1 (A/D converter input 1)
R12	AN2 (A/D converter input 2)
R13	AN3 (A/D converter input 3)
R14	AN4 (A/D converter input 4)
R15	AN5 (A/D converter input 5)
R16	VD (Vertical Sync. input)
R17	HD (Horisontal Sync. input)

R20~R27: R2 is a 8-bit CMOS bidirectional I/O port. Each pins 1 or 0 written to the their Port Direction Register can be used as outputs or inputs.

In addition, R2 serves the functions of the various following special features.

Port pin	Alternate function
R20	INT2 (External interrupt input 2)
R21	Sclk (Serial communication clock)
R22	Sout (Serial communication data out)
R23	Sin (Serial communication data in)
R24	INT3 (External interrupt input 3)
R25	EC2 (Event counter input 2)
R26	INT4 (External interrupt input 4)
R27	EC3 (Event counter input 3)

R40~R47: R40~R43 are 8-bit NMOS open drain output and R45~R47 are bidirectional CMOS Input / NMOS open drain output port. R4 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

In addition, R4 serves the functions of the various following special features.

Port pin	Alternate function
R40	PWM0 (Pulse Width Modulation output 0)
R41	PWM1 (Pulse Width Modulation output 1)
R42	PWM2 (Pulse Width Modulation output 2)
R43	PWM3 (Pulse Width Modulation output 3)
R44	SCL0 (I^2C Clock 0)
R45	SCL1 (I^2C Clock 1) PWM4 (Pulse Width Modulation output 4)
R46	SDA0 (I^2C Data 0)
R47	SDA1 (I^2C Data 1) PWM5 (Pulse Width Modulation output 5)

R50~R56: R50~R53 are 4-bit CMOS bidirectional I/O and R54~R56 are CMOS output port. R5 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

In addition, R5 serves the functions of the various following special features.

Port pin	Alternate function
R50	BUZZ (Buzzer output)
R51	PWM8 (Pulse Width Modulation output 8)
R52	INT0 (External interrupt input 0)
R54	YM (Back ground)
R55	YS (Edge)
R56	I (Intencity)

R67: R67 is an 1-bit CMOS bidirectional I/O port. R67 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

In addition, R67 serves the functions of the various following special features.

Port pin	Alternate function
R67	INT1 (External interrupt input 1)

R,G,B: R,G,B CMOS output port. Each pins controls Red, Green,. Blue color control.

PIN NAME	Pin No.	In/Out	Function	
V_{DD}	39	-	Supply voltage	
V_{SS}	12, 40	-	Circuit ground	
$\overline{TEST}$	38	I	For test purposes. Should not be connected. (N.C.)	
$\overline{RESET}$	11	I	Reset signal input	
X_{IN}	14	I	Main oscillation input	
X_{OUT}	13	O	Main oscillation output	
OSC1	37	I	On screen display functions	On screen display oscillation input
OSC2	36	O		On screen display osc. output
R17/HD	9	I/O		Horisontal Sync. input
R16/VD	10	I/O		Vertical Sync. input
R	30	O		Red signal output
G	31	O		Green signal output
B	32	O		Blue signal output
R56/I	33	O		Intencity signal output
R55/YS	34	O		Edge signal output
R54/YM	35	O		Background signal output
R40/PWM0	52	O	PWM functions	8bit PWM
R41/PWM1	51	O		8bit PWM
R42/PWM2	50	O		8bit PWM
R43/PWM3	49	O		8bit PWM
R45/SCL1/ PWM4	47	I/O		Include I^2C Serial clock 1 (SCL1)
R47/SDA1/ PWM5	45	I/O		Include I^2C Serial data 1 (SDA1)
R51/PWM8	43	I/O		14bit PWM
R44/SCL0	48	I/O	I^2C functions	I^2C Serial clock 0
R46/SDA0	46	I/O		I^2C Serial data 0
R23/Sin	5	I/O	SCI functions	Serial data input
R22/Sout	6	I/O		Serial data output
R21/Sclk	7	I/O		Serial communication clock
R27/EC3	1	I/O	Timer event functions	Event counter input 3
R25/EC2	3	I/O		Event counter input 2
R50/Buzzer	44	I/O	Buzzer function	500Hz ~ 250KHz @8MHz

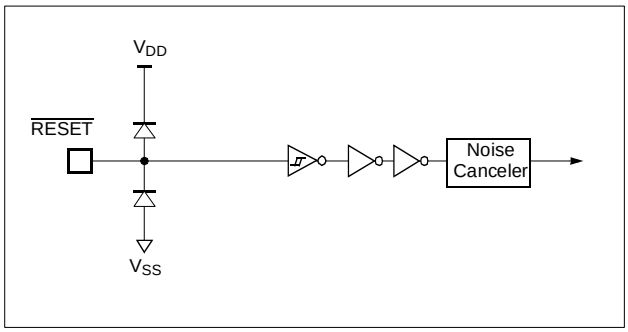
Table 5-1 Port Function Description

PIN NAME	Pin No.	In/Out	Function	
R52/INT0	42	I/O	External interrupt functions	External interrupt input 0
R67/INT1	26	I/O		External interrupt input 1
R20/INT2	8	I/O		External interrupt input 2
R24/INT3	4	I/O		External interrupt input 3
R26/INT4	2	I/O		External interrupt input 4
R10/AN0	20	I/O	A/D conversion functions	Analog input 0
R11/AN1	19	I/O		Analog input 1
R12/AN2	18	I/O		Analog input 2
R13/AN3	17	I/O		Analog input 3
R14/AN4	16	I/O		Analog input 4
R15/AN5	15	I/O		Analog input 5
R00	29	I/O	Digital I/O functions	
R01	28	I/O		
R02	27	I/O		
R03	25	I/O		
R04	24	I/O		
R05	23	I/O		
R06	22	I/O		
R07	21	I/O		
R53	41	I/O		

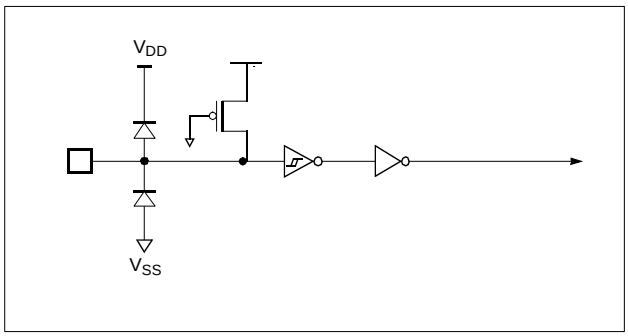
Table 5-1 Port Function Description

6. PORT STRUCTURES

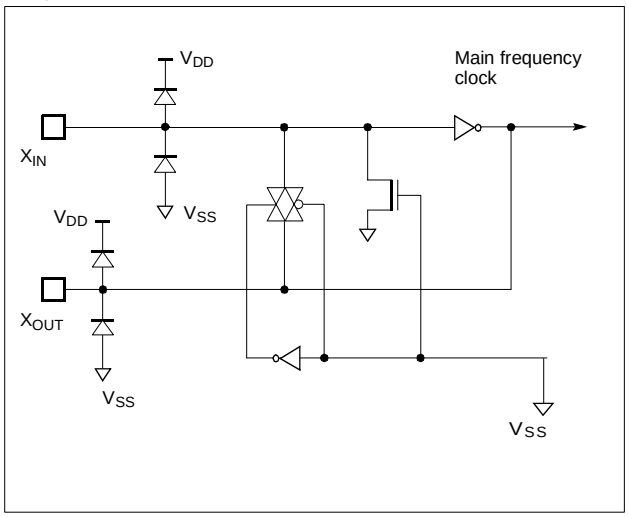
$\overline{\text{RESET}}$



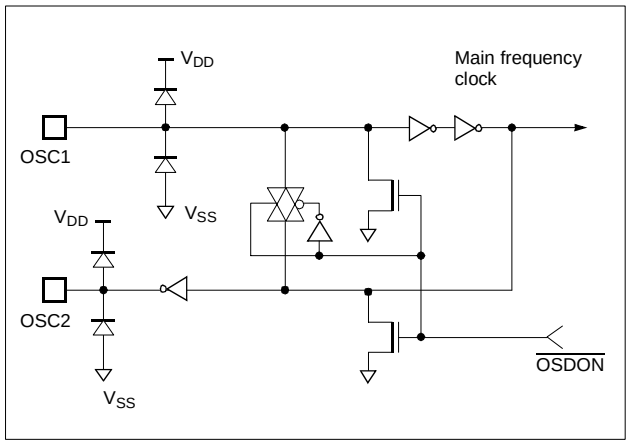
$\overline{\text{TEST}}$



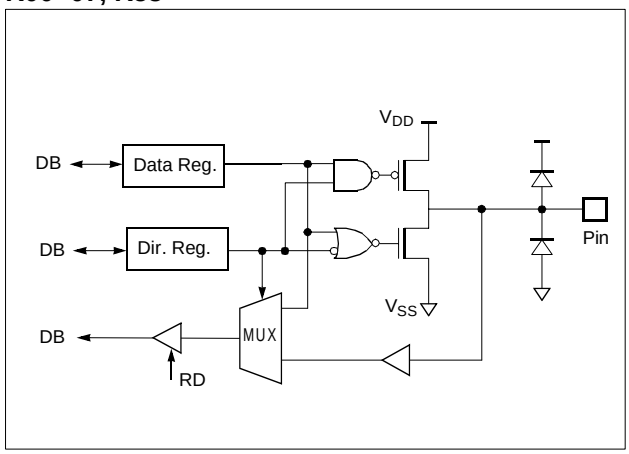
X_{IN}, X_{OUT}



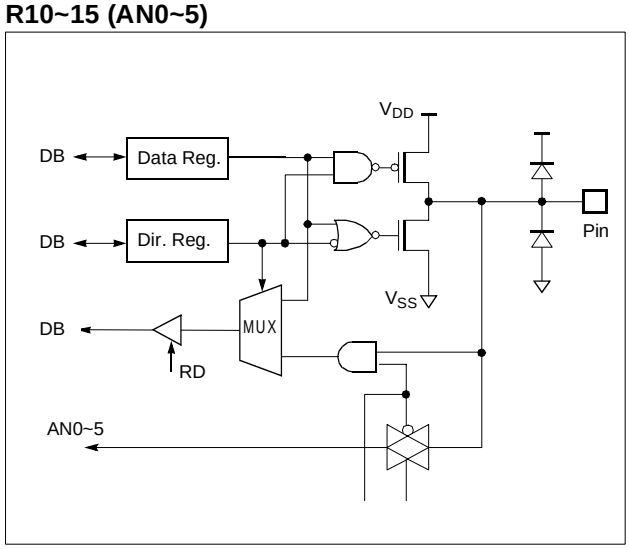
OSC1, OSC2



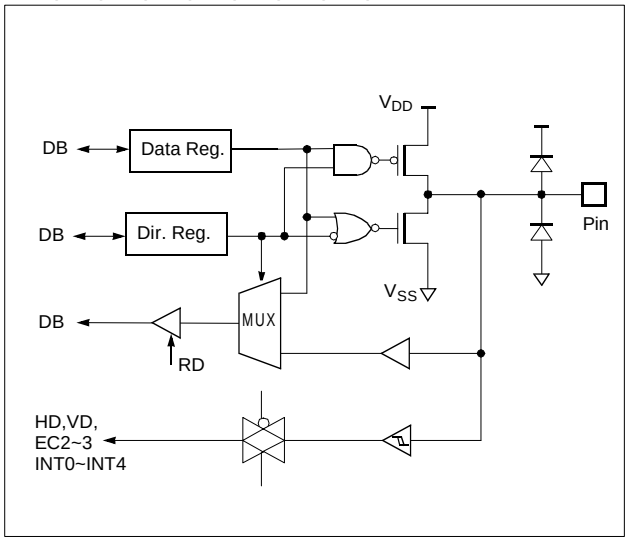
R00~07, R53



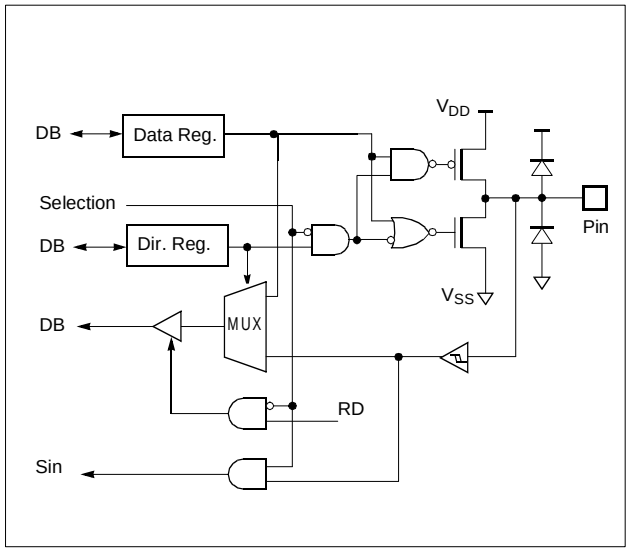
R10~15 (AN0~5)



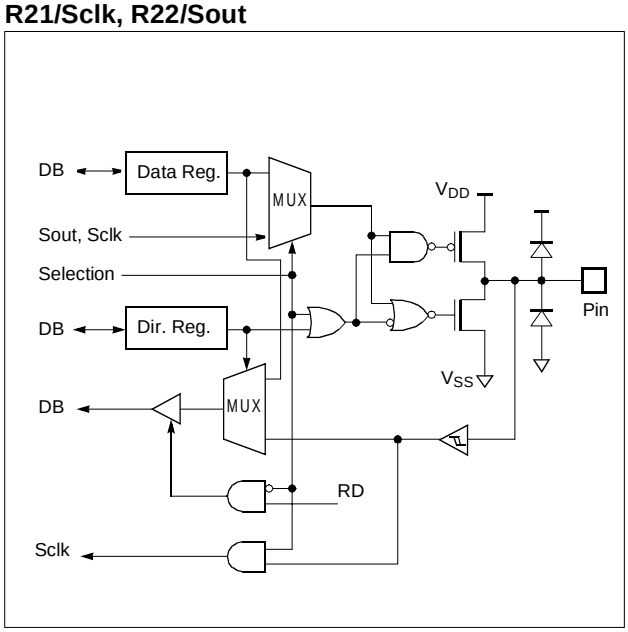
R16, 17, 20, 24, 25, 26, 27, 52, 67



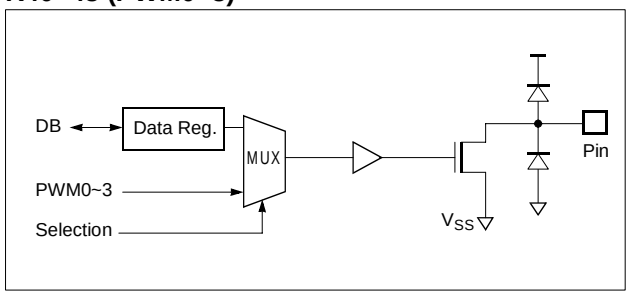
R23/Sin



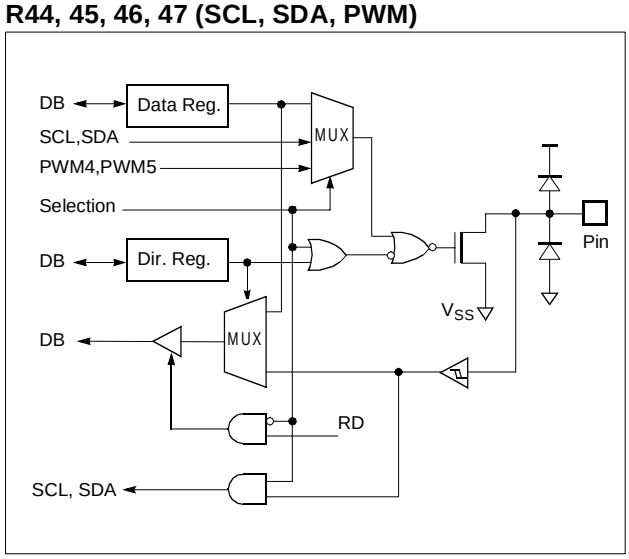
R21/Sclk, R22/Sout



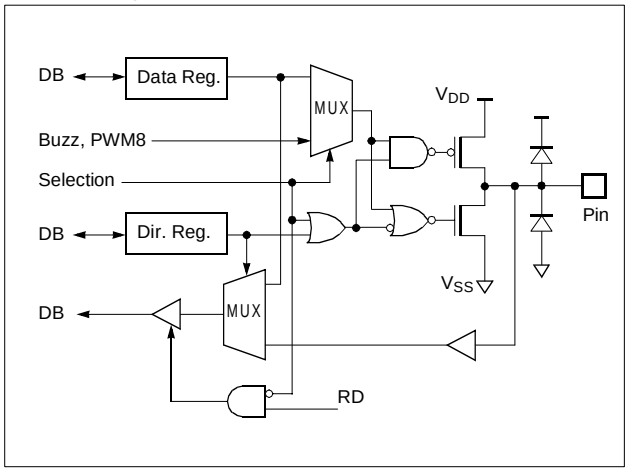
R40~43 (PWM0~3)



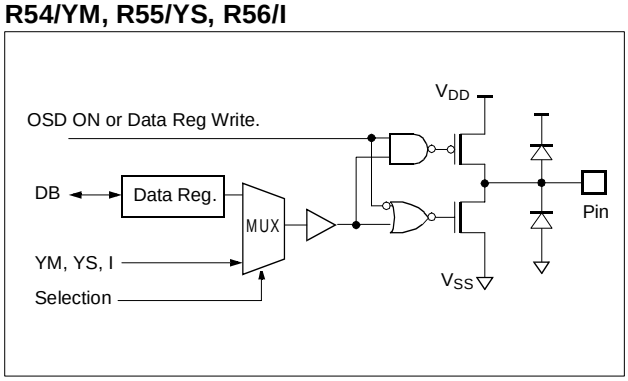
R44, 45, 46, 47 (SCL, SDA, PWM)



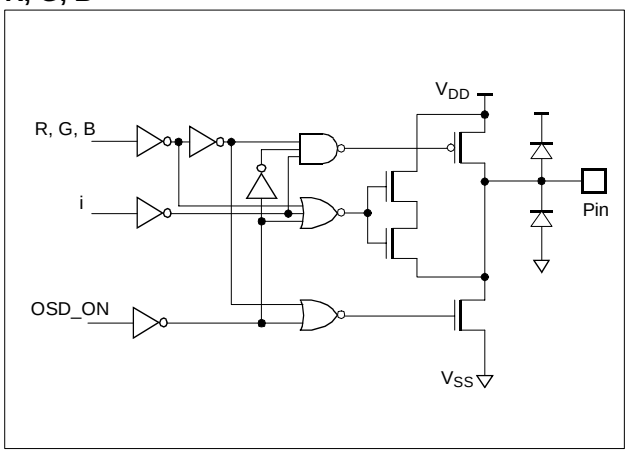
R50/BUZZ, R51/PWM8



R54/YM, R55/YS, R56/I



R, G, B



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 ..... 100 mA

Maximum current into V_{DD} pin ..... 80 mA

Maximum current sunk by (I_{OL} per I/O Pin) ..... 20 mA

Maximum output current sourced by (I_{OH} per I/O Pin) ..... 8 mA

Maximum current (ΣI_{OL}) ..... 80 mA

Maximum current (ΣI_{OH}) ..... 50 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 f_{OSC}=16MHz	4.5	5.5	V
Operating Frequency	f_{XIN}	V_{DD}=4.5~5.5V	4	8	MHz
On Screen Display Operating Frequency	f_{OSC}	V_{DD}=4.5~5.5V	8	16	MHz
Operating Temperature	T_{OPR}		-10	70	°C

7.3 DC Electrical Characteristics - HMS81C43xx

(T_A=-10~70°C, V_{DD}=4.5~5.5V),

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
High level input voltage	V_{IH1}	$\overline{TEST}$, $\overline{RESET}$, Xin, OSC1, R17~16, R27~20, R47~44, R52, R67	0.8 V_{DD}	-	V_{DD}	V
	V_{IH2}	R0, R15~10, R53~50	0.7 V_{DD}	-	V_{DD}	V
Low level input voltage	V_{IL1}	$\overline{TEST}$, $\overline{RESET}$, Xin, OSC1, R17~16, R27~20, R47~44, R52, R67	0	-	0.12 V_{DD}	V
	V_{IL2}	R0, R15~10, R53~50	0	-	0.3 V_{DD}	V
High level output voltage	V_{OH}	I_{OH} = -5mA R0, R1, R2, R5, R67	V_{DD} - 1	-	-	V
		I_{OH} = -1.2mA R,G,B	V_{DD} - 1	-	-	V
Low level output voltage	V_{OL}	I_{OL} = 5mA R0, R1, R2, R4, R5, R67, R, G, B	-	-	1.0	V
Supply current in ACTIVE mode	I_{DD}	V_{DD}	-	-	30	mA

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
pull-up lekage current	I_{RUP}	V_{DD} = 5.5v, V_{PIN} = 0.4V $\overline{TEST}$	-1.5		-400	μA
High input leakage current	I_{IZH}	V_{DD} = 5.5V, V_{PIN} = V_{DD} All input, I/O pins except X_{IN}, OSC1, R47~40	-5	-	5	μA
Low input leakage current	I_{IZL}	V_{DD} = 5.5V, V_{PIN} = 0V All input, I/O pins except X_{IN}, OSC1, R47~44	-5	-	5	μA
Open drain leakage current	I_{LOZ}	V_{DD} = 5.5V, V_{OH} = V_{DD}, N-ch Tr. off R47~40	-	-	10	μA
RAM data retention voltage	V_{RAM}	V_{DD}	1.2	-	-	V
I^2C port impedance (I/O Transistor off)	R_{BS}	V_{DD} = 4.5V, V_{SCL0} = V_{SCL1} = 2.25V V_{SDA0} = V_{SDA1} = 2.25V SCL0:SCl1 (R44:R45) SDA0:SDA1 (R46:R47)	-	-	120	Ω
Hysterisis	Vt+ ~ Vt-	$\overline{TEST}$, $\overline{RESET}$, Xin, OSC1, R17~16, R27~20, R47~44, R52, R67	1.0	-	-	V

7.4 A/D Comparator Characteristics

(T_A=-10~70°C, V_{DD}=5.0V)

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}	AN0~AN5	V_{SS}	-	V_{DD}	V
Accuracy	N_{FS}	-	-	-	TBΔ	LSB

7.5 AC Characteristics

(T_A=-10~70°C, V_{DD}=5V±10%, V_{SS}=0V)

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Operating Frequency	f_{XIN}	X_{IN}	4	-	8	MHz
	f_{OSC}	OSC	8	-	16	MHz
External Clock Pulse Width	t_{MCPW}	X_{IN}	62.5	-	125	nS
	t_{SCPW}	S_{CLK}	0.5	-		µS
External Clock Transition Time	t_{MRCP},t_{MFCP}	X_{IN}	-	-	20	nS
	t_{SRCP},t_{SFCP}	S_{CLK}	-	-	20	nS
Oscillation Stabilizing Time	t_{ST}	X_{IN}, X_{OUT}	-	-	20	mS
Interrupt Pulse Width	t_{IW}	INT0~4	2	-	-	t_{SYS}[1]
$\overline{\text{RESET}}$ Input Width	t_{RST}	$\overline{\text{RESET}}$	8	-	-	t_{SYS}[1]
Event Counter Input Pulse Width	t_{ECW}	EC2, EC3	2	-	-	t_{SYS}[1]
Event Counter Transition Time	t_{REC},t_{FEC}	EC2, EC3	-	-	20	nS

1. t_{SYS} is one of $2/f_{XIN}$ main clock operation mode,

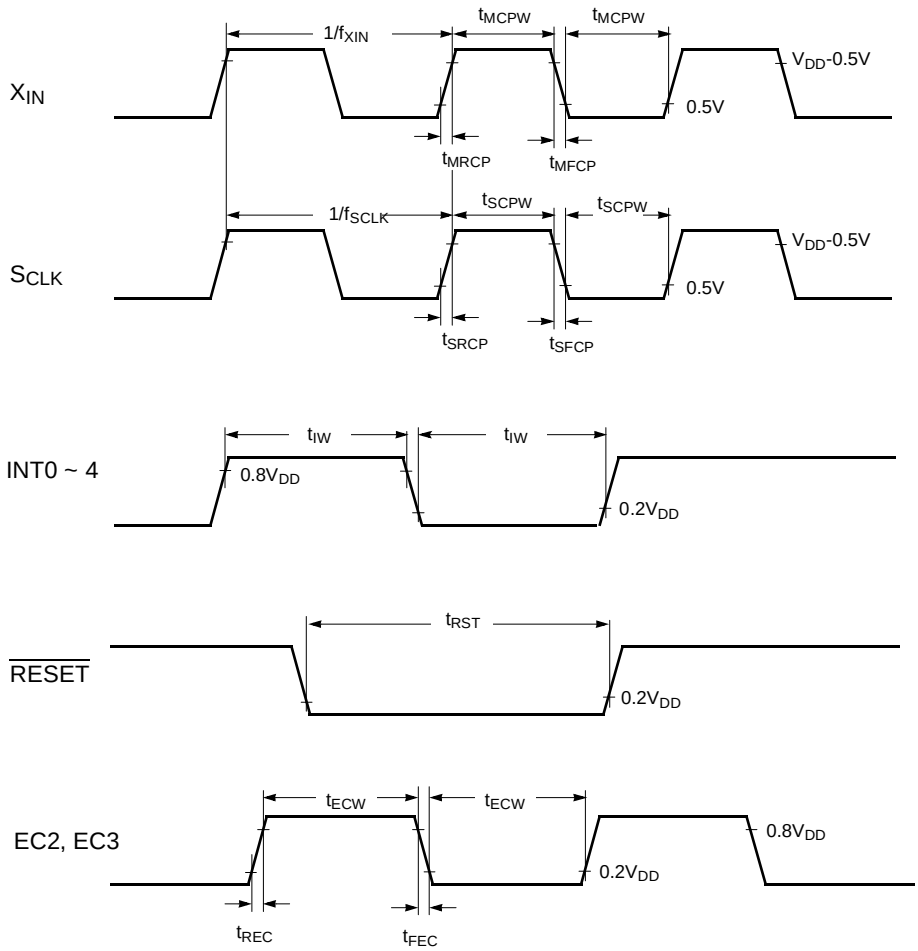


Figure 7-1 Timing Chart

7.6 Typical Characteristics

This data will generate after evaluation.
Not available at this time.

8. MEMORY ORGANIZATION

The GMS81C43xx/GMS87C4060 has separate address spaces for Program memory, Data Memory and Display memory. Program memory can only be read, not written to. It can be up to 32K/60K bytes of Program memory. Data memory can be read and written to up to 256~512/1,536 bytes including the stack area. Font memory has prepared 16K bytes for OSD.

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.

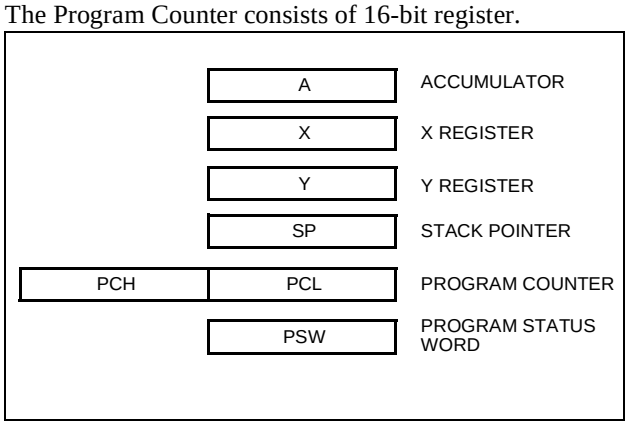


Figure 8-1 Configuration of Registers

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.

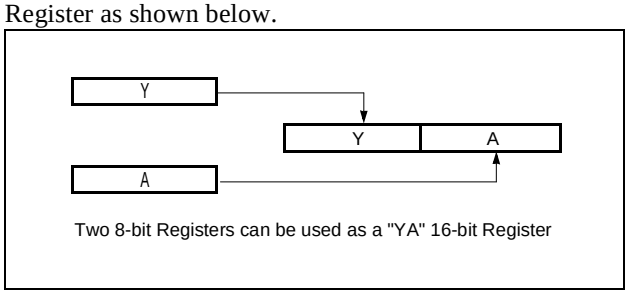


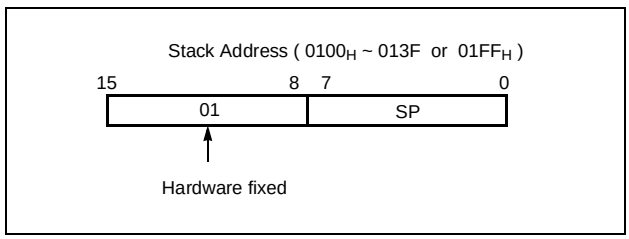
Figure 8-2 Configuration of YA 16-bit Register

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 0100_H to $01FF_H$ 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 "FF_H" is used.



Caution:

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

Example: To initialize the SP for 81C4324 (256 RAM)

```
LDX    #03FH
TXSP                ; SP ← 3FH
```

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.

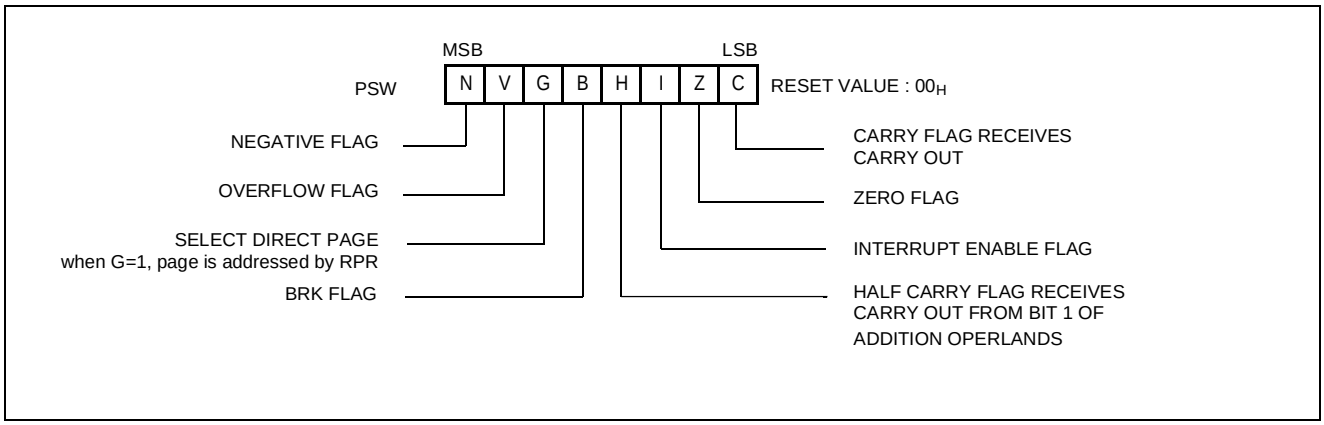


Figure 8-3 PSW (Program Status Word) Register

[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.

[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 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.

[Direct page flag G]

This flag assigns RAM page for direct addressing mode. In the direct addressing mode, addressing area is from zero page 00_H to $0FF_H$ when this flag is "0". If it is set to "1", addressing area is assigned by DPGR register (address $0F8_H$). It is set by SETG instruction and cleared by CLRG.

[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.

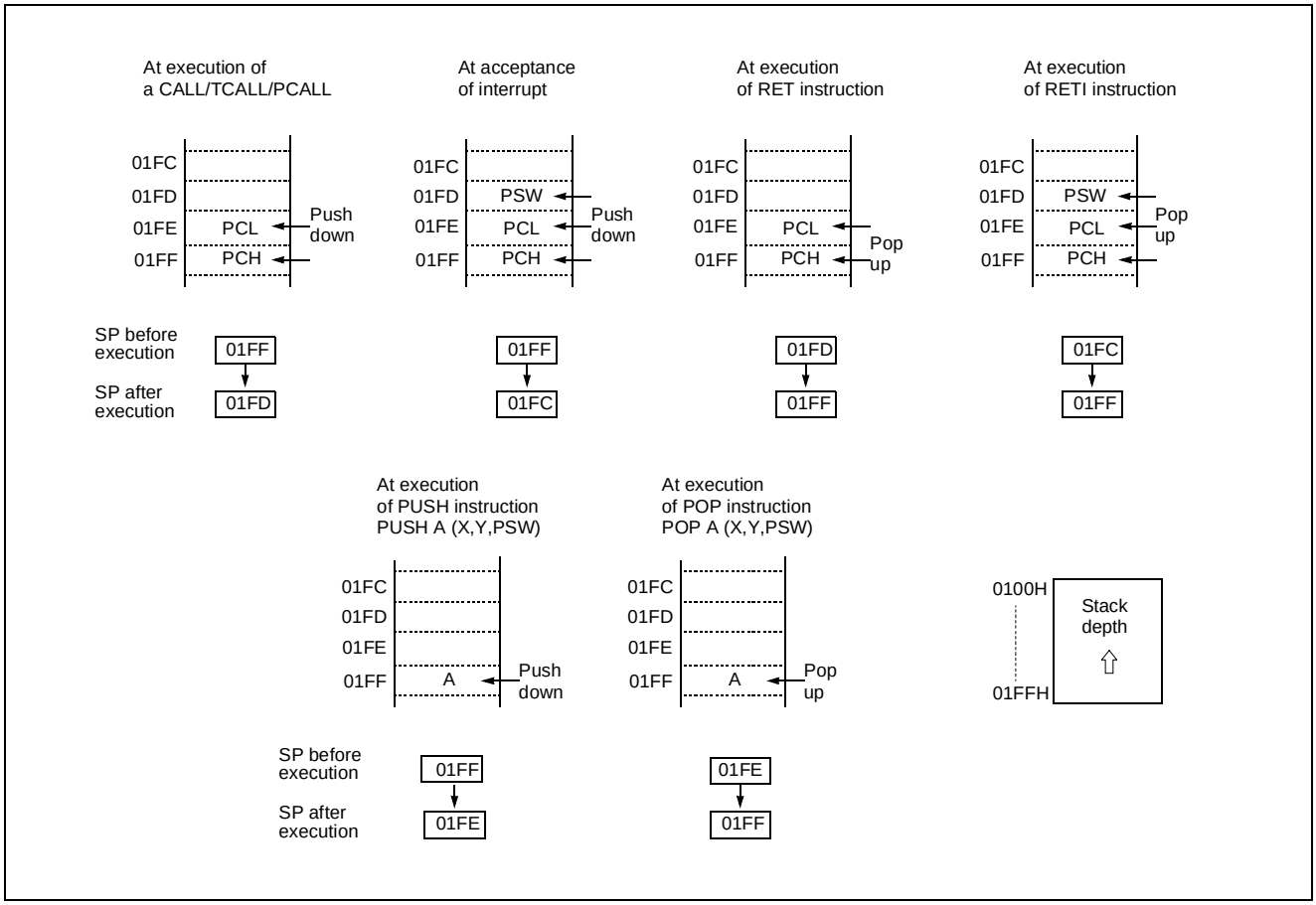


Figure 8-4 Stack Operation

8.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but HMS81C43xx/GMS87C4060 has 8~32K/ 60K bytes program memory space only physically implemented. Accessing a location above $FFFF_H$ will cause a wrap-around to 0000_H.

Figure 8-5 , 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 8-6 .

As shown in Figure 8-5 , each area is assigned a fixed location in Program Memory. Program Memory area contains the user program.

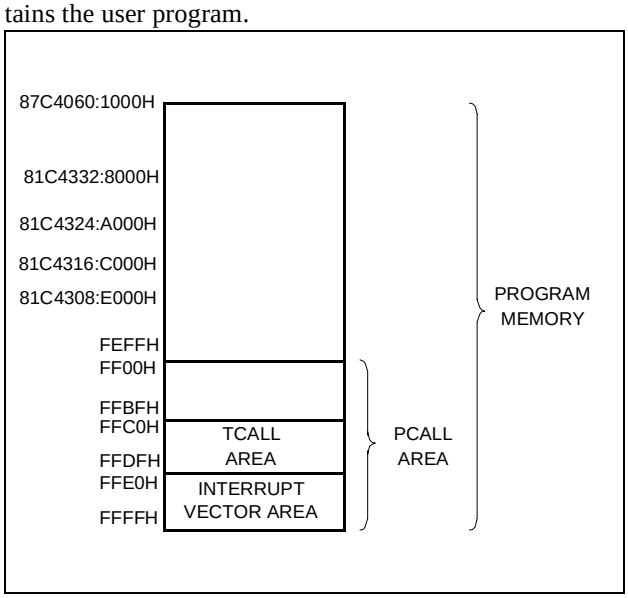


Figure 8-5 Program Memory Map

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 8-7 .

Example: Usage of TCALL

```
        LDA    #5
        TCALL  0FH          ;1BYTE INSTRUCTION
        :                   ;INSTEAD OF 2 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 $0FFFC_H$. The interrupt service locations spaces 2-byte interval: $0FFF8_H$ and $0FFF9_H$ for External Interrupt 1, $0FFFC_H$ and $0FFFD_H$ for External Interrupt 0, etc.

Any area from $0FF00_H$ to $0FFFF_H$, if it is not going to be used, its service location is available as general purpose Program Memory.

Address	Vector Area Memory
$0FFE0_H$	I^2C Bus Interface Interrupt Vector Area
E2	Serial I/O Interrupt Vector Area
E4	Basic Interval Timer Interrupt Vector Area
E6	Watchdog Timer Interrupt Vector Area
E8	External Interrupt 3/4 Vector Area
EA	Timer/Counter 3 Interrupt Vector Area
EC	Timer/Counter 1 Interrupt Vector Area
EE	V-Sync Interrupt Vector Area
F0	1 Frame Timer Interrupt Vector Area
F2	Timer/Counter 2 Interrupt Vector Area
F4	Timer/Counter 0 Interrupt Vector Area
F6	External Interrupt 2 Vector Area
F8	External Interrupt 1 Vector Area
FA	On Screen Display Interrupt Vector Area
FC	External Interrupt 0 Vector Area
FE	RESET Vector Area

NOTE:
"-" means reserved area.

Figure 8-6 Interrupt Vector Area

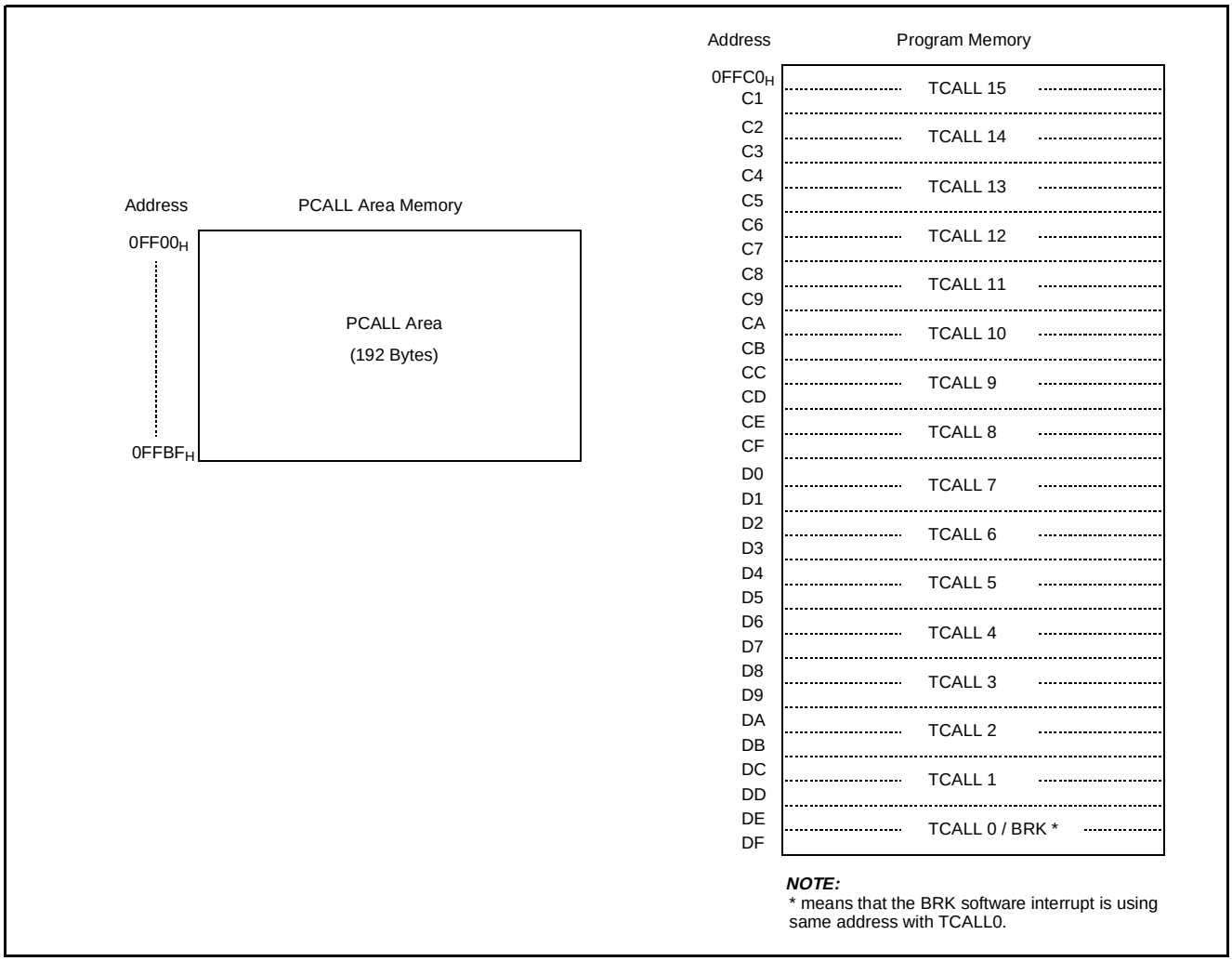
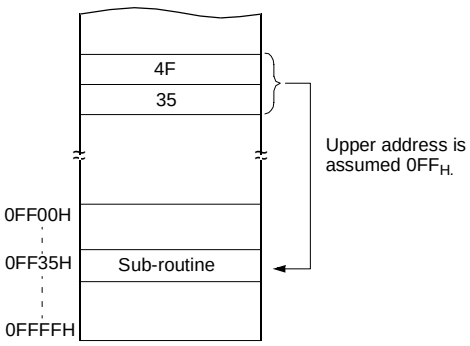


Figure 8-7 PCALL and TCALL Memory Area

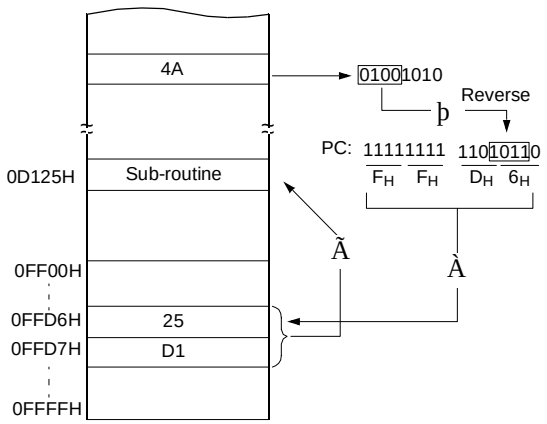
PCALL→ rel

4F35 PCALL 35H



TCALL→ n

4A TCALL 4



Example: The usage software example of Vector address and the initialize part in HMS81C4324.

```
          ORG     0FFE0H

          DW      I2C                   ; I2C
          DW      SERIAL                ; Serial I/O
          DW      BIT                   ; Basic interval timer
          DW      WATCHDOG              ; Watch dog timer
          DW      INT3_4                ; Interrupt 3/4
          DW      TIMER3                ; Timer 3
          DW      TIMER1                ; Timer 1
          DW      VSYNC                 ; Vertical Sync.
          DW      One_Frame             ; 1 Frame interrupt
          DW      TIMER2                ; Timer 2
          DW      TIMER0                ; Timer 0
          DW      INT2                  ; Interrupt 2
          DW      INT1                  ; Interrupt 1
          DW      OSD                   ; On Screen Display
          DW      INT0                  ; Interrupt 0
          DW      RESET                 ; Reset


          ORG     0A000H                ; Start Address of 24 kbytes Program Memory

;*****************************************
;         MAIN    PROGRAM               *
;*****************************************
;
;
RESET:    DI                            ; Disable All Interrupts
          LDX     #0
          LDA     #0                    ; RAM Page0 Clear(!0000H->!00BFH)
RAM_CLR:  STA     {X}+
          CMPX    #0C0H
          BNE     RAM_CLR
;
          CALL    OSD_Stop              ; Stop OSD display
;
          LDX     #03FH                 ; Stack Pointer Initialize
          TXSP

          LDM     R0, #0                ; Normal Port 0
          LDM     R0DD,#1000_0010B      ; Normal Port Direction
          :
          :
          :
          :
```

8.3 Data Memory

Figure 8-8 shows the internal Data Memory space available. Data Memory is divided into four groups, a user RAM, control registers, Stack, and OSD memory.

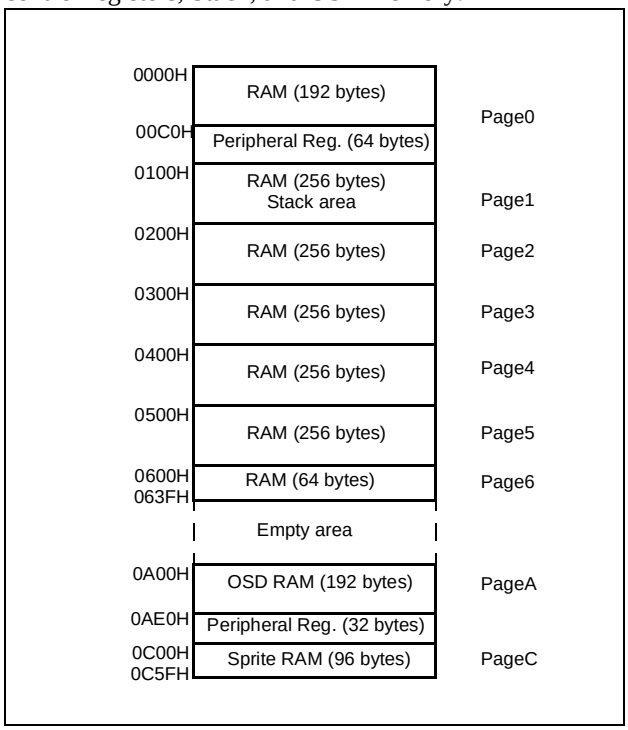


Figure 8-8 Data Memory Map

User Memory

The GMS87C4060 has 1,536 × 8 bits for the user memory (RAM). HMS81C4332 has 512 x 8 bit for the user memory (RAM) addressed 000H ~ 23FH except Peripheral Reg. (64 bytes) . HMS81C4308/16/24 has 256 x 8 bit RAM addressed 000H ~ 13FH except peripheral register (0C0h~0FFh)

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 basic control registers are in address range of $00C0_H$ to $00FF_H$. And OSD control registers are assigned within $0AE0_H$ ~ $0AFF_H$.

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    CLCTLR,#09H ;Divide ratio ÷8
```

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. Refer to Figure 8-4 on page 20.

Address	Symbol	R/W	Reset Value	Addressing mode
00C0H	R0	R/W	????????	byte, bit[1]
00C1H	R0DD	W	00000000	byte[2]
00C2H	R1	R/W	????????	byte, bit
00C3H	R1DD	W	00000000	byte
00C4H	R2	R/W	????????	byte, bit
00C5H	R2DD	W	00000000	byte
00C8H	R4	R/W	????????	byte, bit
00C9H	R4DD	W	0000----	byte
00CAH	R5	R/W	????????	byte, bit
00CBH	R5DD	W	----0000	byte
00CCH	R6	R/W	?-------	byte, bit
00CDH	R6DD	W	0-------	byte
00CEH	FUNC1	W	-0000000	byte
00CFH	FUNC2	W	---00000	byte

Table 8-1Control registers

0D0H	TM0	R/W	-0000000	byte
0D1H	TM2	R/W	-0000000	byte
0D2H	TDR0	R/W	????????	byte, bit
0D3H	TDR1	R/W	????????	byte, bit
0D4H	TDR2	R/W	????????	byte, bit
0D5H	TDR3	R/W	????????	byte, bit
0D6H	BITR	R	????????	byte
	CKCTLR	W	--010111	byte
0D7H	WDTR	W	-0111111	byte
0D8H	ICAR	R/W	00000000	byte, bit
0D9H	ICDR	R/W	11111111	byte, bit
0DAH	ICSR	R/W	0001000-	byte, bit
0DBH	ICCR1	R/W	00000000	byte, bit
0DCH	ICCR2	R/W	00000000	byte, bit
0DEH	SIOM	R/W	-0000001	byte, bit
0DFH	SIOR	R/W	????????	byte, bit
0E0H	PWMR0	W	????????	byte
0E1H	PWMR1	W	????????	byte
0E2H	PWMR2	W	????????	byte
0E3H	PWMR3	W	????????	byte
0E4H	PWMR4	W	????????	byte
0E5H	PWMR5	W	????????	byte
0E8H	PWM8H	R/W	????????	byte, bit
0E9H	PWM8L	R/W	--??????	byte, bit
0EAH	PWMCR1	R/W	00000000	byte, bit
0EBH	PWMCR2	R/W	--0-0000	byte, bit
0EEH	BUR	W	????????	byte
0EFH	AIPS	W	--000000	byte
0F0H	ADCM	R/W	--011101	byte, bit
0F1H	ADR	R	????????	byte
0F2H	IEDS	W	--000000	byte
0F3H	IMOD	R/W	--000000	byte, bit
0F4H	IENL	R/W	0000000-	byte, bit
0F5H	IRQL	R/W	0000000-	byte, bit
0F6H	IENH	R/W	00000000	byte, bit
0F7H	IRQH	R/W	00000000	byte, bit
0F9H	IDCR	R/W	0000-000	byte, bit
0FAH	IDFS	R	1----001	byte
0FBH	IDR	R	????????	byte
0FCH	DPGR	R/W	----0000	byte, bit
0FDH	TMR	W	????????	byte
0AE0H	OSDcon1	R/W	00000000	byte, bit
0AE1H	OSDcon2	R/W	-0000000	byte, bit
0AE2H	OSDPOL	W	????????	byte
0AE3H	FDWSET	W	01111010	byte
0AE4H	EDGEcol	W	10000111	byte
0AE5H	OSDLN	R	---00000	byte
0AE6H	LHPOS	W	????????	byte
0AE8H	SPVPOS	W	????????	byte
0AE9H	SPHPOS	W	????????	byte
0AF0H	L1ATTR	W	????????	byte
0AF1H	L1VPOS	W	????????	byte
0AF3H	L2ATTR	W	????????	byte
0AF4H	L2VPOS	W	????????	byte

Table 8-1Control 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.

8.4 Addressing Mode

The GMS800 series uses six addressing modes;

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

(1) Register Addressing

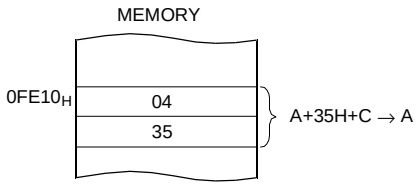
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

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

Example:

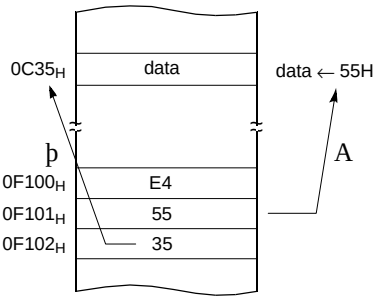
```
FE10: 0435      ADC          #35H
```



When G-flag is 1, then RAM address is defined by 16-bit address which is composed of 8-bit direct page accessable register (DPGR) and 8-bit immediate data.

Example: G=1, DPGR=0CH

```
F100: E45535   LDM          35H,#55H
```

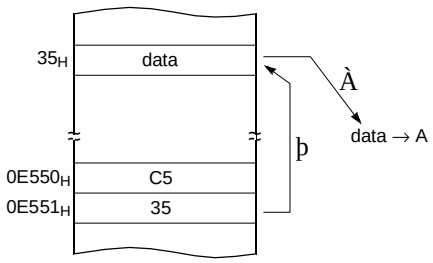


(3) Direct Page Addressing → dp

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

Example; G=0

```
E551: C535      LDA          35H;A ←RAM[35H]
```



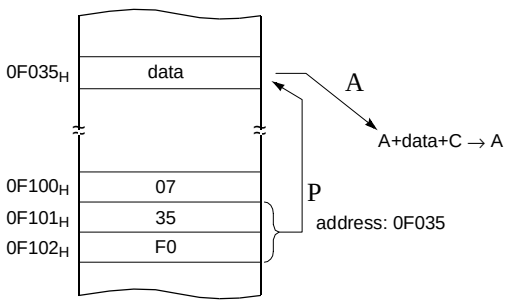
(4) Absolute Addressing → !abs

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;

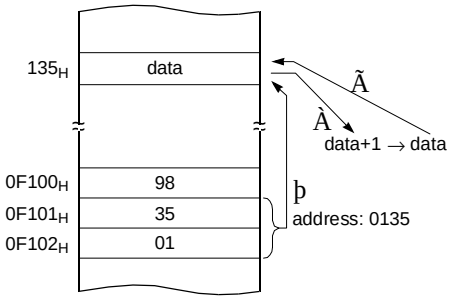
```
F100: 0735F0 ADC!0F035H    ;A ←ROM[0F035H]
```



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

Example; Addressing accesses the address 0135_H regardless of G-flag and DPGR.

```
F100: 983501 INC!0135H     ;A ←ROM[135H]
```



(5) Indexed Addressing

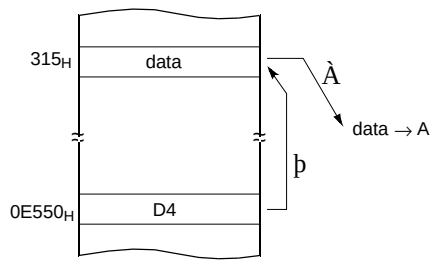
X indexed direct page (no offset) → {X}

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

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

Example; $X=15_H$, G=1, $DPGR=03_H$

```
E550: D4        LDA          {X};ACC←RAM[X].
```



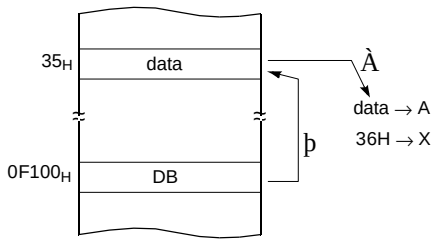
X indexed direct page, auto increment→ {X}+

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; G=0, $X=35_H$

```
F100: DB        LDA          {X}+
```



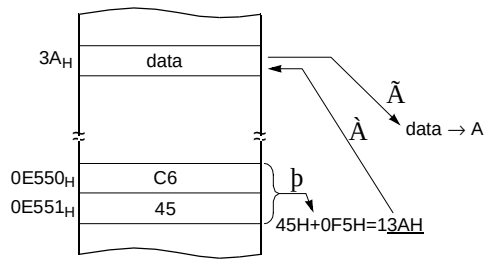
X indexed direct page (8 bit offset) → dp+X

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; G=0, $X=0F5_H$

```
E550: 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.

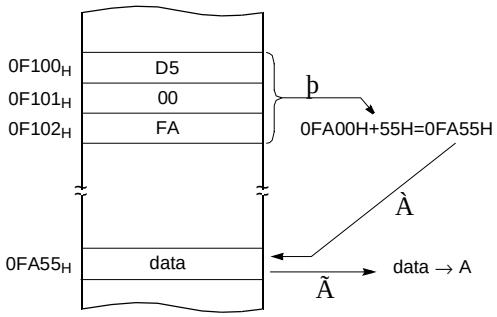
This is same with above (2). Use Y register instead of X.

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

```
F100: D500FA   LDA          !0FA00H+Y
```



(6) Indirect Addressing

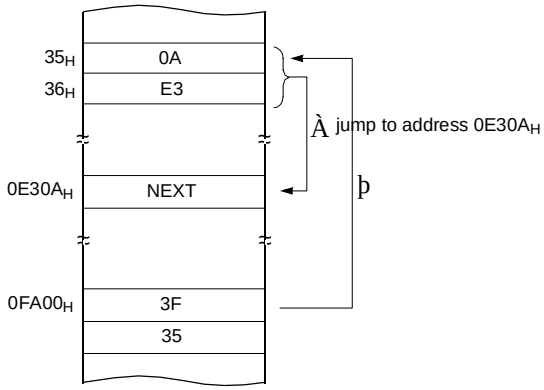
Direct page indirect → [dp]

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; G=0

```
FA00: 3F35     JMP          [35H]
```



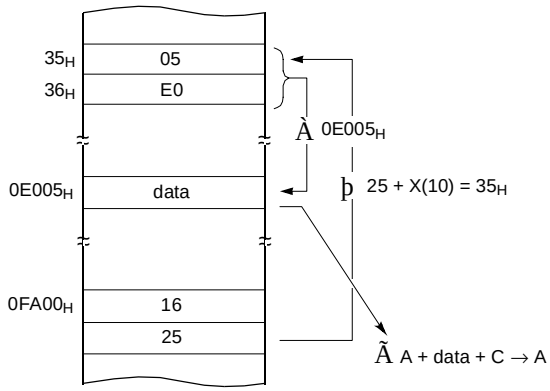
X indexed indirect → [dp+X]

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; G=0, X=10_H

```
FA00: 1625     ADC          [25H+X]
```



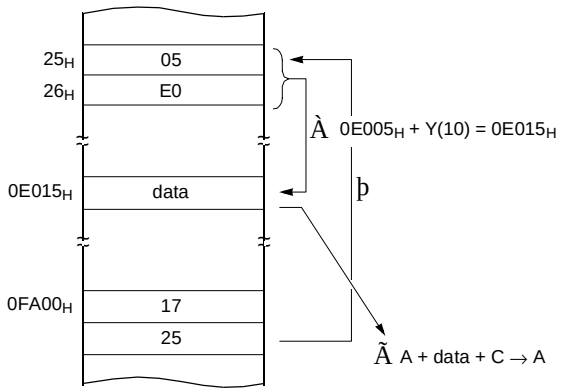
Y indexed indirect → [dp]+Y

Processes momory 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; G=0, Y=10_H

```
FA00: 1725      ADC          [25H]+Y
```



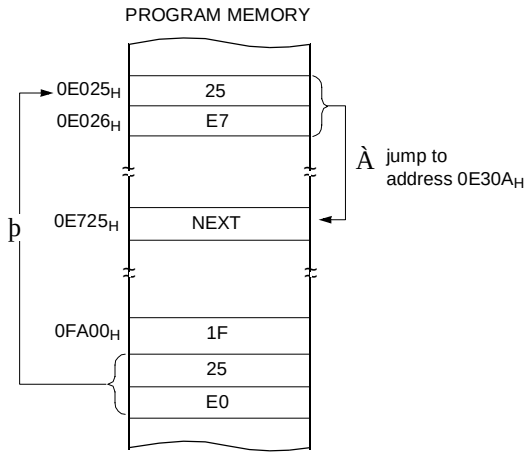
Absolute indirect → [!abs]

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

JMP

Example; G=0

```
FA00: 1F25E0   JMP          [!0C025H]
```



9. I/O PORTS

The GMS87C4060 has digital ports (R0, R1, R2, R4, R5 and R6) and OSD ports (R,G,B), but HMS81C43xx has R1, R2, R4, R5 and OSD ports (R,G,B)

These ports pins may be multiplexed with an alternate function for the peripheral features on the device. In general, in a initial reset state, R ports are used as a general purpose digital port.

9.1 Registers for Port

Port Data Registers

The Port Data Registers in I/O buffer in each R ports are represented as a Type D flip-flop, which will clock in a value from the internal bus in response to a "write to data register" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read data register" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to "read data register" signal from the CPU. Some instructions that read a port activating the "read register" signal, and others activating the "read pin" signal

Port Direction Registers

All pins have data direction registers which can define these ports as output or input. A "1" in the port direction register configure the corresponding port pin as output. Conversely, write "0" to the corresponding bit to specify it as input pin. For example, to use the even numbered bit of R0 as output ports and the odd numbered bits as input ports, write "55_H" to address $0C1_H$ (R0 port direction register) during initial setting as shown in Figure 9-1 .

All the port direction registers in the HMS81C43xx/ GMS87C4060 have 0 written to them by reset function. On the other hand, its initial status is input.

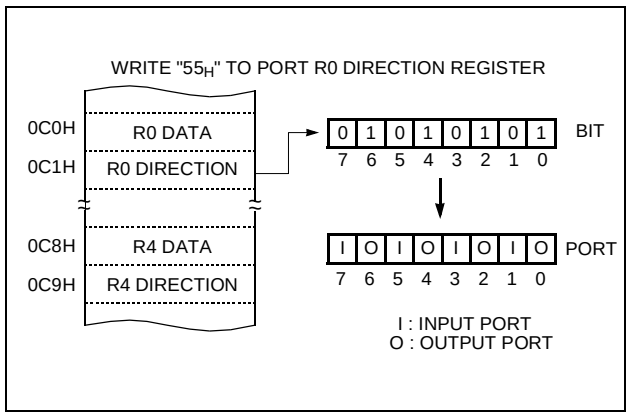


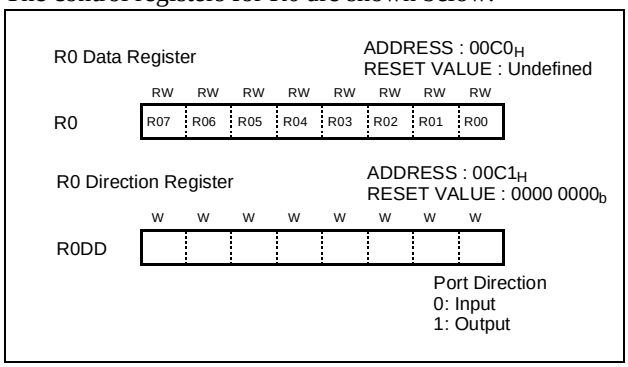
Figure 9-1 Example of port I/O assignment

9.2 I/O Ports Configuration

R0 Ports

R0 is an 8-bit CMOS bidirectional I/O port (address $0C0_H$). Each I/O pin can independently used as an input or an output through the R0DD register (address $0C1_H$).

The control registers for R0 are shown below.



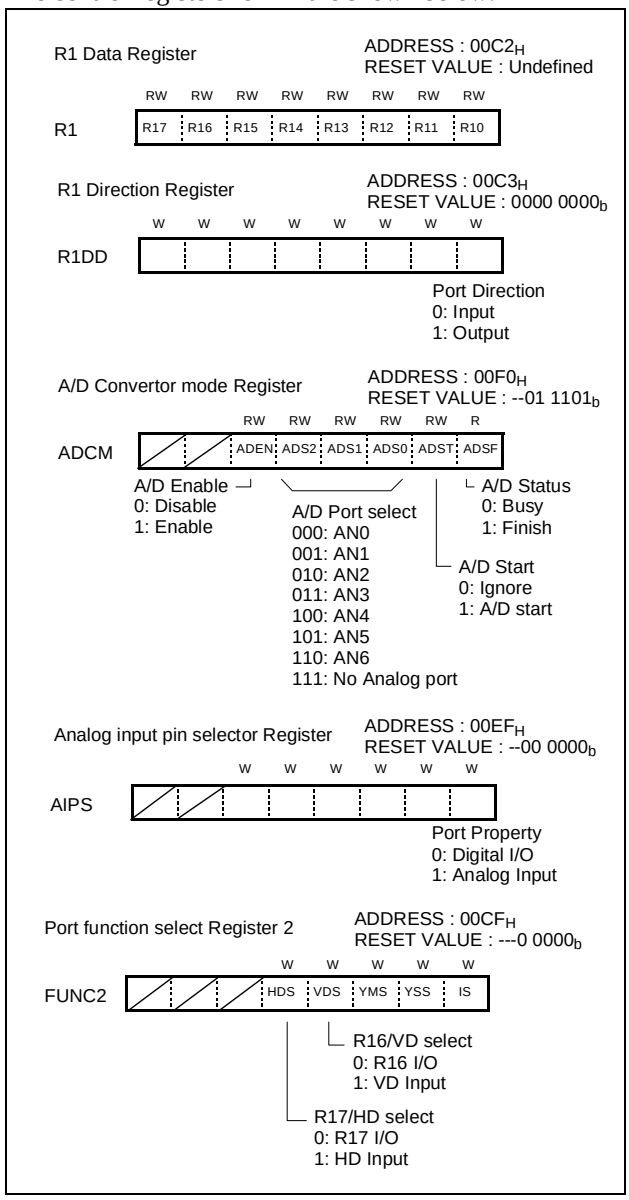
In addition, Port R0 is only digital I/O. After reset, R0DD value is "0", R0 acts as normal digital input port.

R1 Ports

R1 is an 8-bit CMOS bidirectional I/O port (address $0C2_H$). Each I/O pin can independently used as an input or an output through the R1DD register (address $0C3_H$).
R1 port have secondary functions as following table.

Port Pin	Alternate Function
R10	AN0 (A/D input 0)
R11	AN1 (A/D input 1)
R12	AN2 (A/D input 2)
R13	AN3 (A/D input 3)
R14	AN4 (A/D input 4)
R15	AN5 (A/D input 5)
R16	VD (Vertical Sync. input)
R17	HD (Horizontal Sync. input)

The control registers for R1 are shown below.

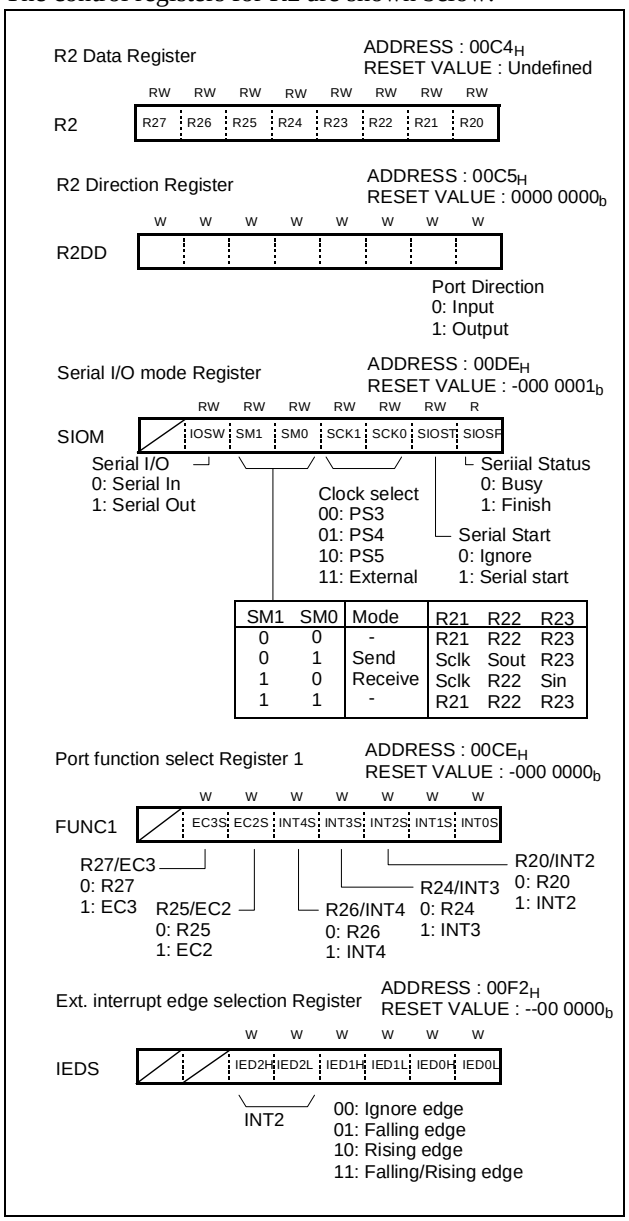


Port R1 is multiplexed with various special features.The control registers controls the selection of alternate function. After reset, R1 port acts as normal digital input port. The way to select alternate function such as A/D input or HD,VD will be shown in each peripheral section.

R2 Port

R2 is an 8-bit CMOS bidirectional I/O port (address $0C4_H$). Each I/O pin can independently used as an input or an output through the R2DD register (address $00C5_H$).

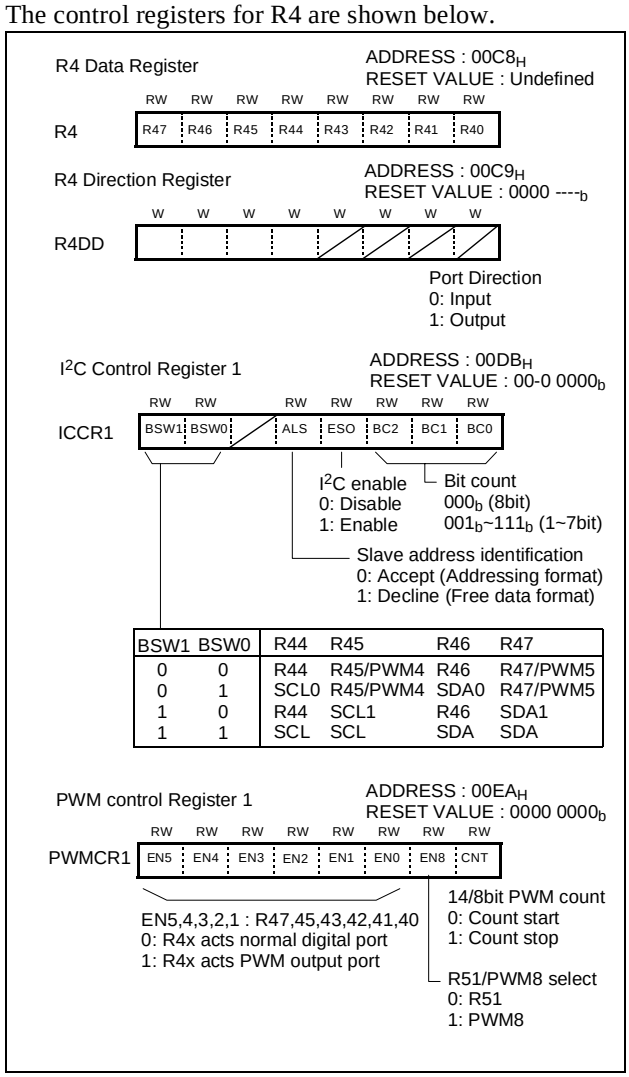
The control registers for R2 are shown below.



R4 Port

R4 is consrutced with 4-bit Open drain Output port and 4-bit CMOS bidirectional I/O port (address $0C8_H$). Each I/O pin can independently used as an input or an output through the R4DD register (address $0C9_H$).

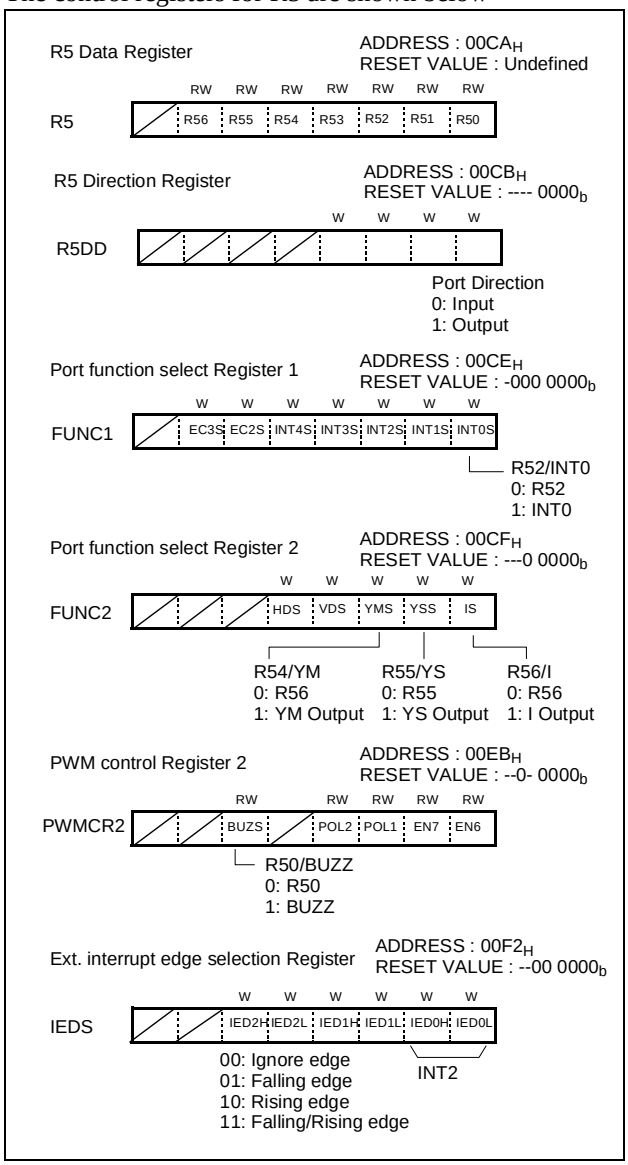
The control registers for R4 are shown below.



R5 Port

R5 is an 7-bit port (address $0CA_H$). Each I/O pin can independently used as an input or an output through the R5DD register (address $0CB_H$).

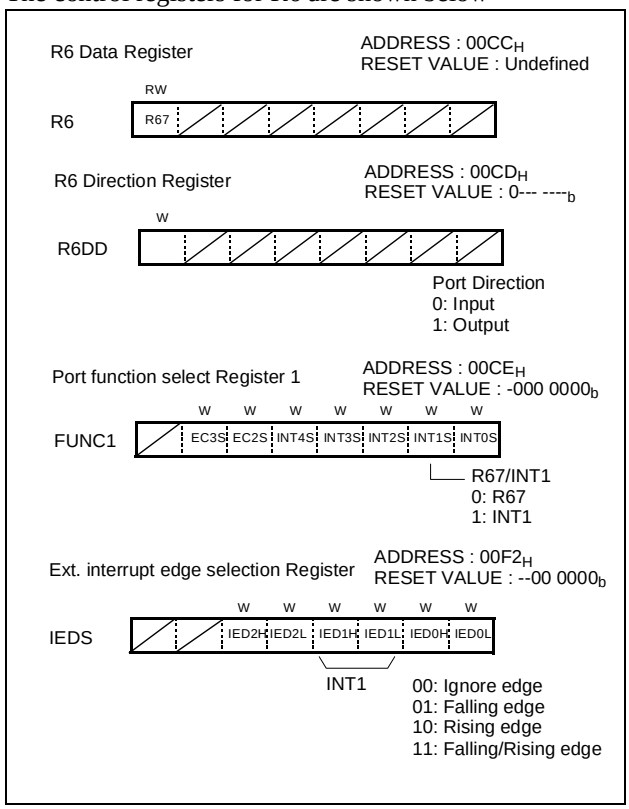
The control registers for R5 are shown below



R6 Port

R6 is an 1-bit CMOS bidirectional I/O port (address $0CC_H$). Each I/O pin can independently used as an input or an output through the R6DD register (address $0CD_H$).

The control registers for R6 are shown below



10. CLOCK GENERATOR

As shown in Figure 10-1 , the clock generator produces the basic clock pulses which provide the system clock to be supplied to the CPU and the peripheral hardware. It contains two oscillators: a main-frequency clock oscillator and a sub-frequency clock oscillator. The system clock can also be obtained from the external oscillator.

The clock generator produces the system clocks forming clock pulse, which are supplied to the CPU and the peripheral hardware.

Main clock	Minimum instruction cycle time (ex:NOP ; f_{ex} 4clock is needed)
3.6MHz	1,111nS
4MHz	1,000nS
8MHz	500nS

To the peripheral block, the clock among the not-divided original clocks, divided by 2, 4,..., up to 1024 can be provided. Peripheral clock is enabled or disabled by bit 4 of the peripheral clock enable register (ENPCK).

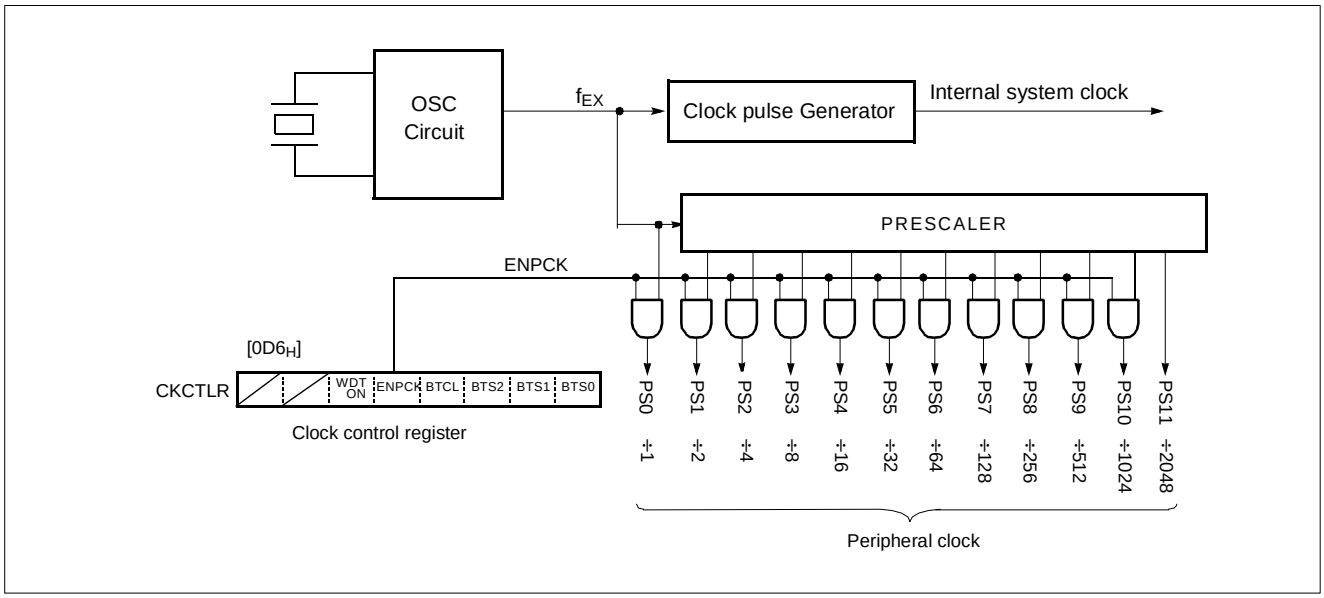
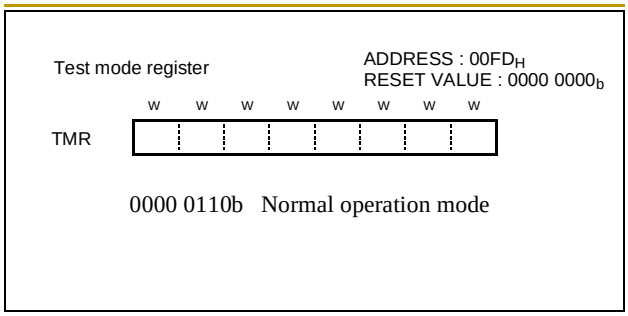
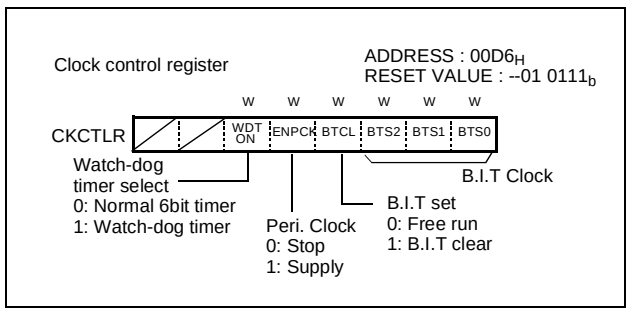


Figure 10-1 Block Diagram of Clock Generator

Note: *On the initial reset, all peripherals are run because peripheral clock is supplied to each function block. If you want to see more details, see Clock Control Register (CKCTLR).*





11. TIMER

11.1 Basic Interval Timer

The HMS81C43xx/GMS87C4060 has one 8-bit Basic Interval Timer that is free-run and can not be stopped. Block diagram is shown in Figure 11-1 .

The Basic Interval Timer generates the time base for watchdog timer counting, and etc. It also provides a Basic interval timer interrupt (BITIF). As the count overflow from FF_H to 00_H, this overflow causes the interrupt to be generated. The Basic Interval Timer is controlled by the clock control register (CKCTLR) shown in Figure 11-2 .

Source clock can be selected by lower 3 bits of CKCTLR.

BITR and CKCTLR are located at same address, and address $00D6_H$ is read as a BITR and written to CKCTLR..

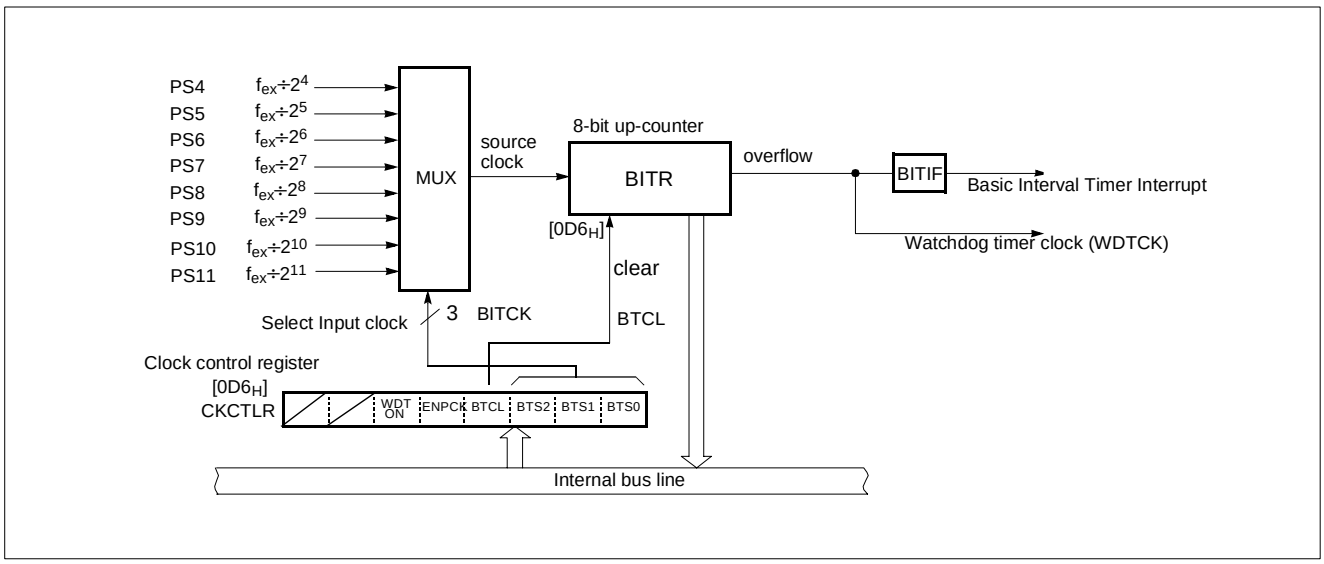


Figure 11-1 Block Diagram of Basic Interval Timer

BTS2~0	Source clock		Interrupt (overflow) Period
	Pre-Scaler output		At f_{ex}=8MHz
000	PS4	$f_{ex}÷2^4$	0.512 mS
001	PS5	$f_{ex}÷2^5$	1.024
010	PS6	$f_{ex}÷2^6$	2.048
011	PS7	$f_{ex}÷2^7$	4.096
100	PS8	$f_{ex}÷2^8$	8.192
101	PS9	$f_{ex}÷2^9$	16.384
110	PS10	$f_{ex}÷2^{10}$	32.768
111	PS11	$f_{ex}÷2^{11}$	65.536

Table 11-1 Basic Interval Timer Interrupt Time

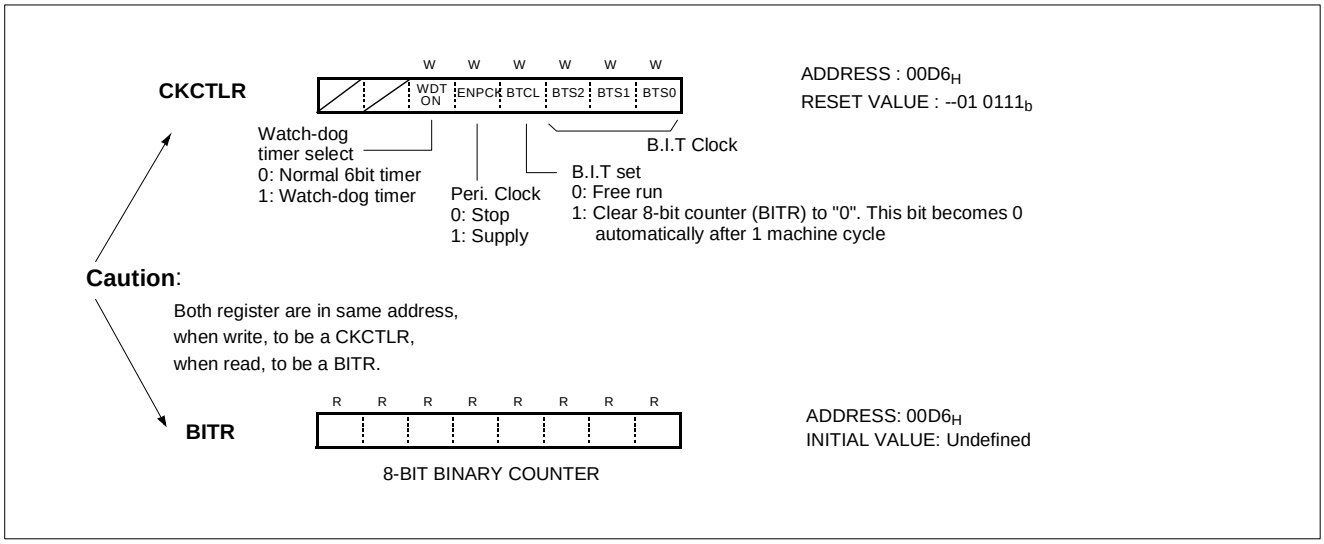


Figure 11-2 BITR: Basic Interval Timer Mode Register

11.2 Timer 0, 1

Timer 0, 1 consists of prescaler, multiplexer, 8-bit compare data register, 8-bit count register, Control register, and Comparator as shown in Figure 11-3 .

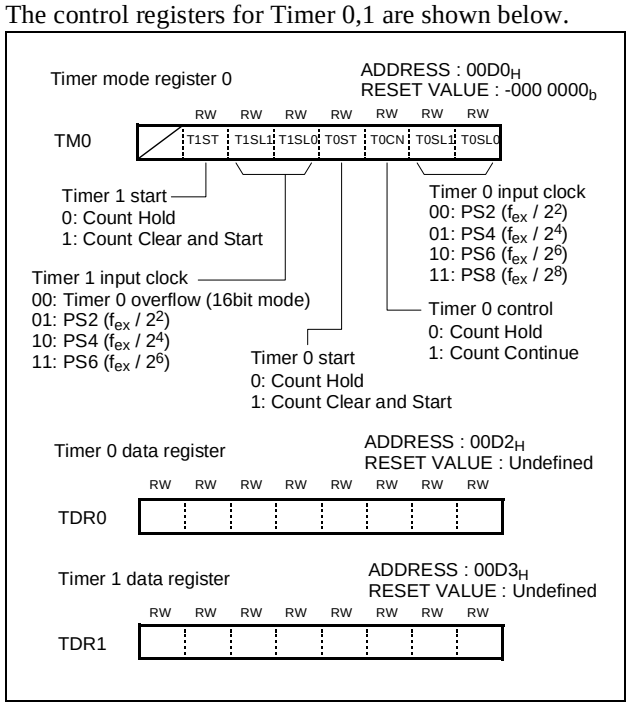
These Timers can run separated 8bit timer or combined 16bit timer. These timers are operated by internal clock.

The contents of TDR1 are compared with the contents of up-counter T1. If a match is found, a timer/counter 1 interrupt (T1IF) is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.

Note: *You can read Timer 0, Timer 1 value from TDR0 or TDR1. But if you write data to TDR0 or TDR1, it changes Timer 0 or Timer 1 modulo data, not Timer value.*

The content of TDR0, TDR1 must be initialized (by software) with the value between 01_H and FF_H,not to 00_H. Or not, Timer 0 or Timer 1 can not count up forever.

The control registers for Timer 0,1 are shown below.



.

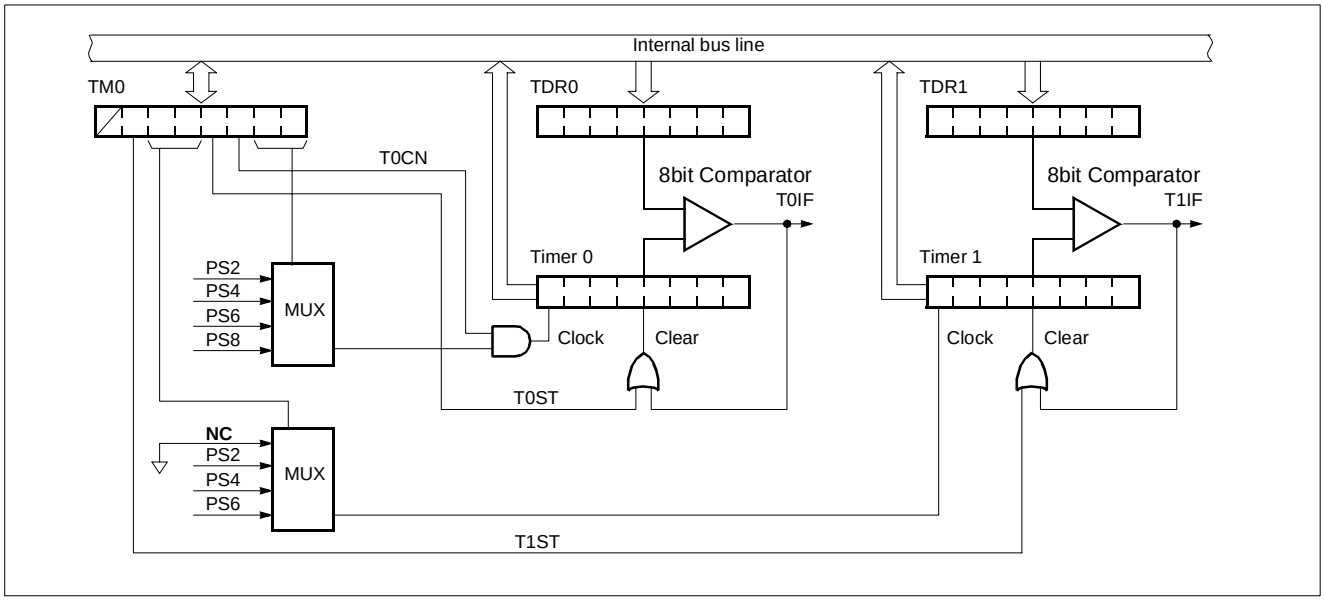


Figure 11-3 Simplified Block Diagram of 8bit Timer0, 1

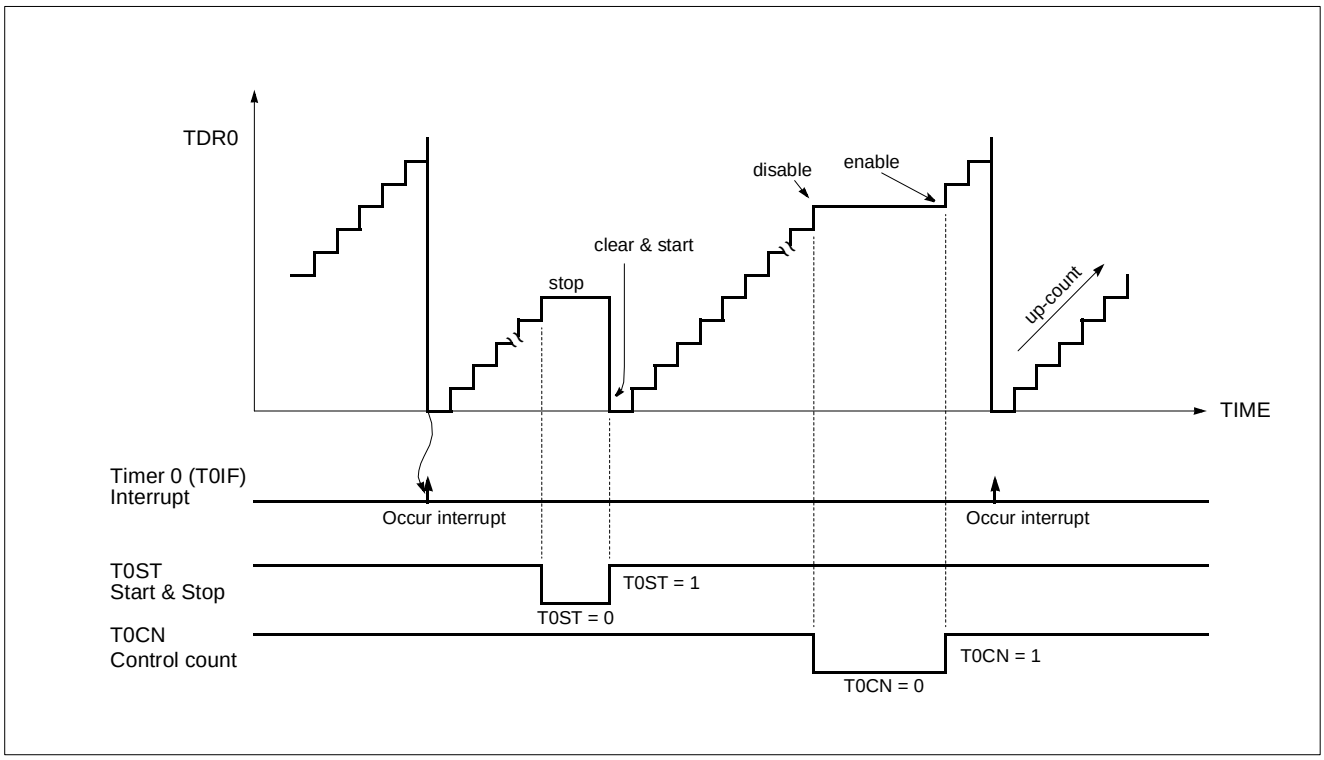


Figure 11-4 Count Example of Timer

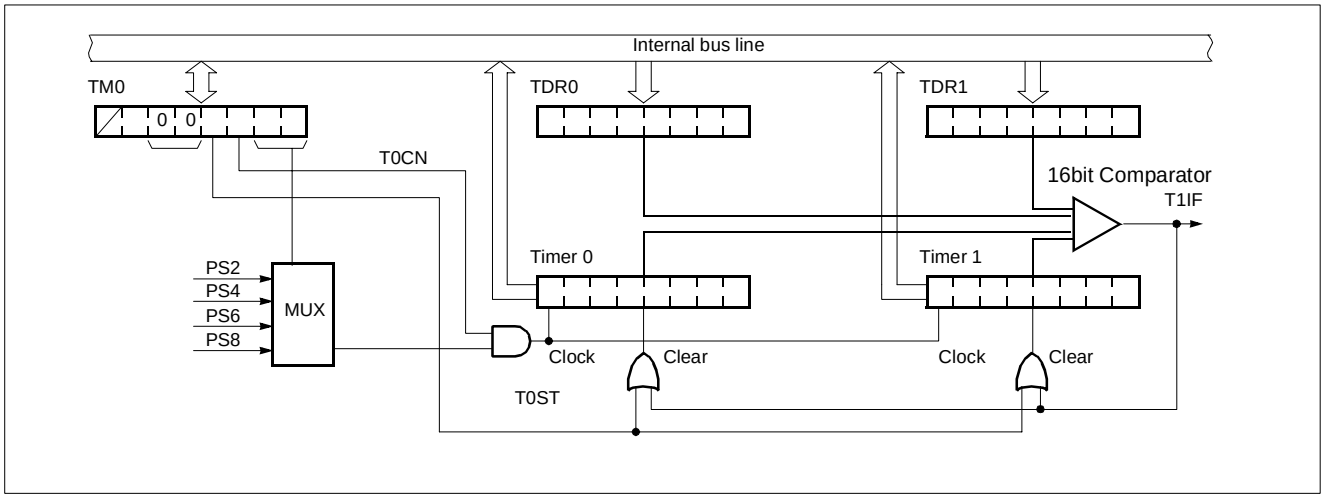


Figure 11-5 Simplified Block Diagram of 16bit Timer0, 1

11.3 Timer / Event Counter 2, 3

Timer 2, 3 consists of prescaler, multiplexer, 8-bit compare data register, 8-bit count register, Control register, and Comparator as shown in Figure 11-5 .

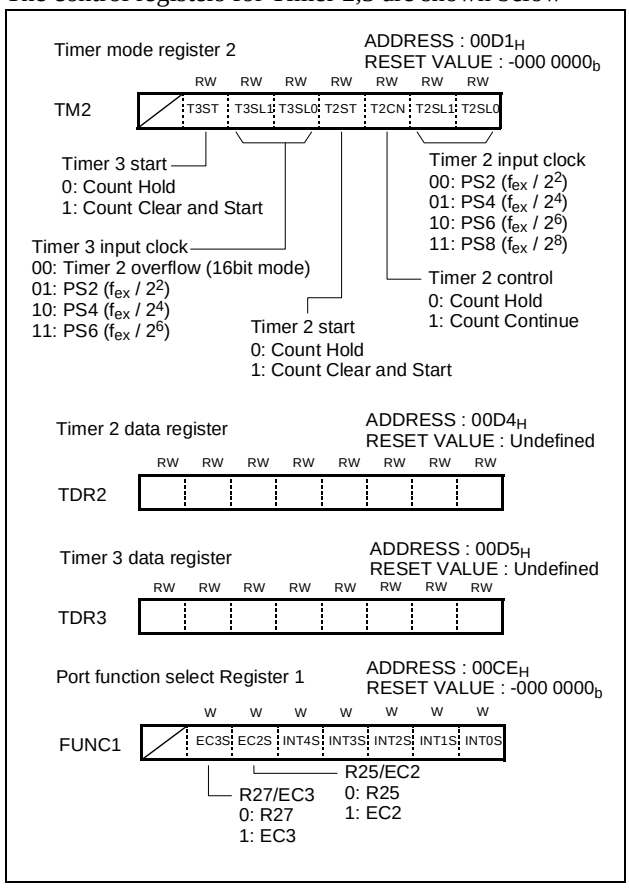
These Timers have two operating modes. One is the timer mode which is operated by internal clock, other is event counter mode which is operated by external clock from pin R25/EC2, R27/EC3.

These Timers can run separated 8bit timer or combined 16bit timer.

Note: *You can read Timer 2, Timer 3 value from TDR2 or TDR3. But if you write data to TDR2 or TDR3, it changes Timer 2 or Timer 3 modulo data, not Timer value.*

The content of TDR2, TDR3 must be initialized (by software) with the value between 01_H and FF_H,not to 00_H. Or not, Timer 2 or Timer 3 can not count up forever.

The control registers for Timer 2,3 are shown below



.

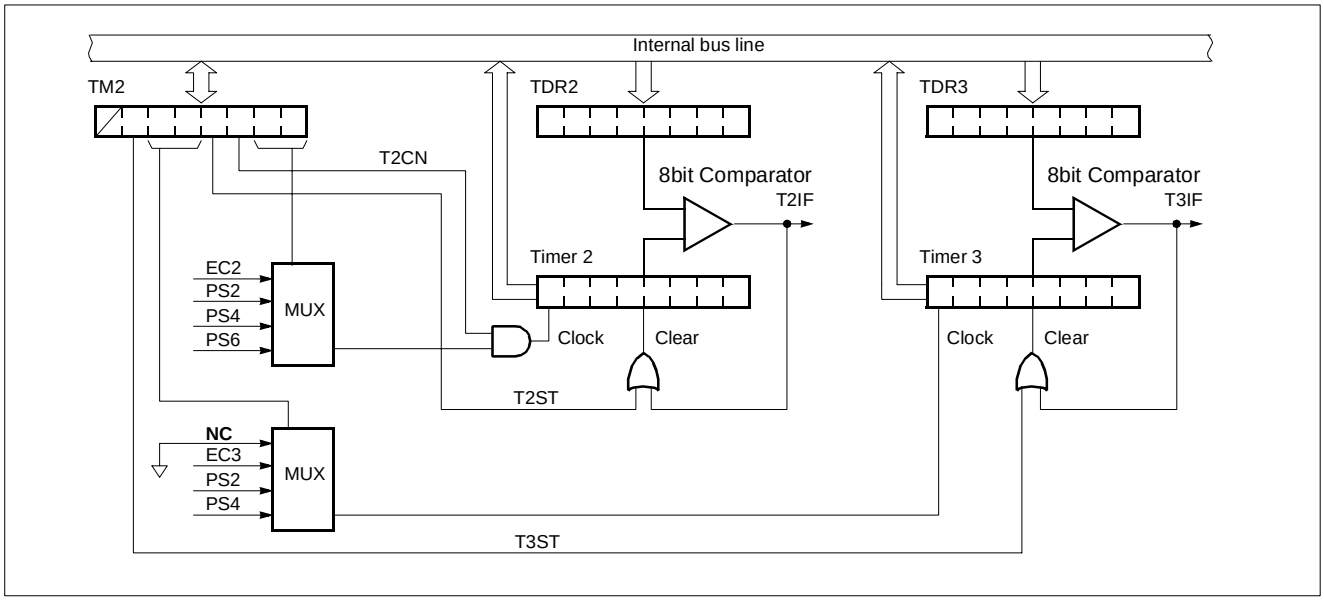


Figure 11-6 Simplified Block Diagram of 8bit Timer/Event Counter 2,3

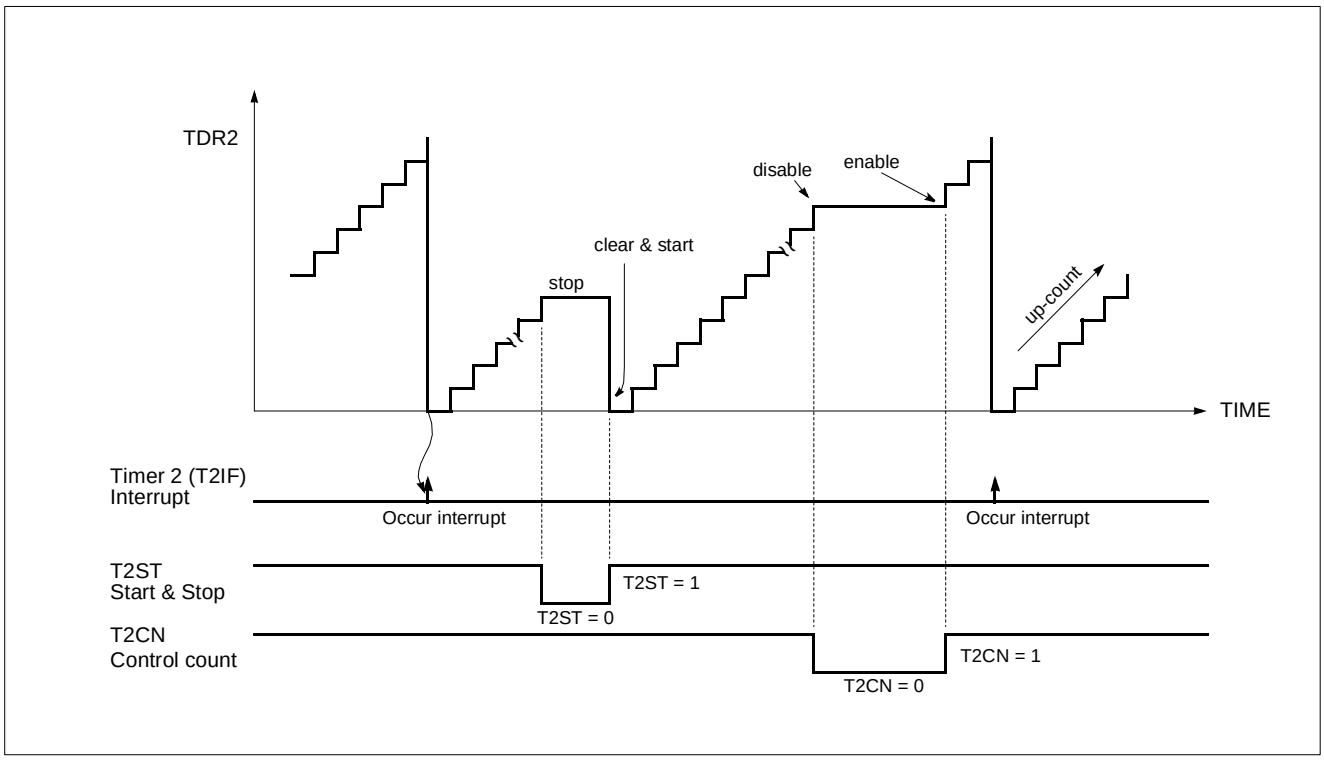


Figure 11-7 Count Example of Timer / Event counter

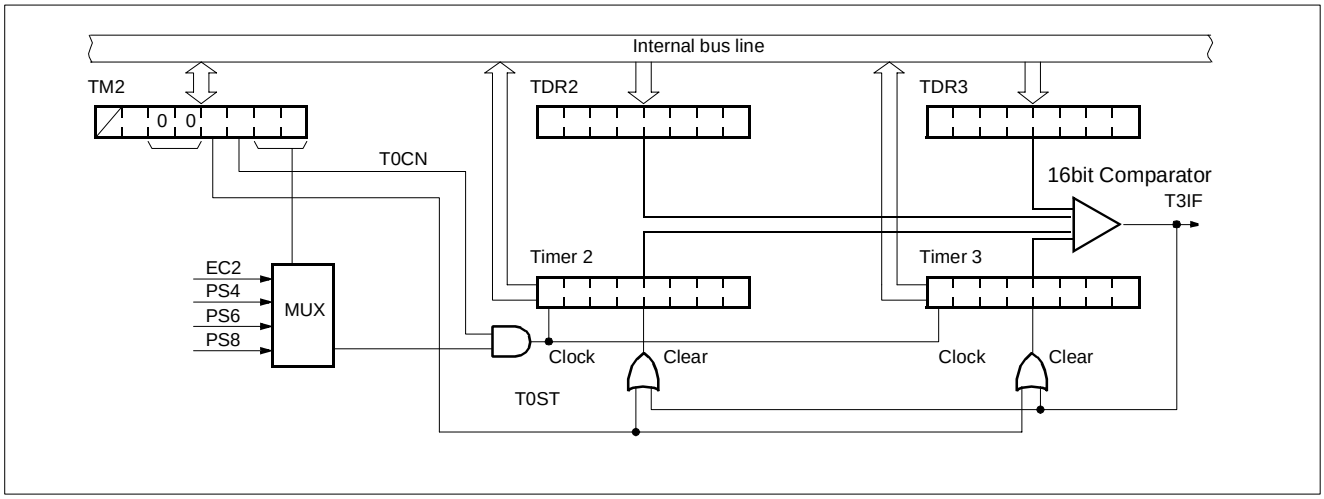


Figure 11-8 Simplified Block Diagram of 16bit Timer/Event Counter 2,3

Timer Mode

In the timer mode, the internal clock is used for counting up. Thus, you can think of it as counting internal clock input. The contents of TDRn (n=0~3) are compared with the contents of up-counter, Timer n. If match is found, a timer n interrupt (TnIF) is generated and the up-counter is cleared to 0. Counting up is resumed after the up-counter is cleared.

As the value of TDRn is changeable by software, time interval is set as you want.

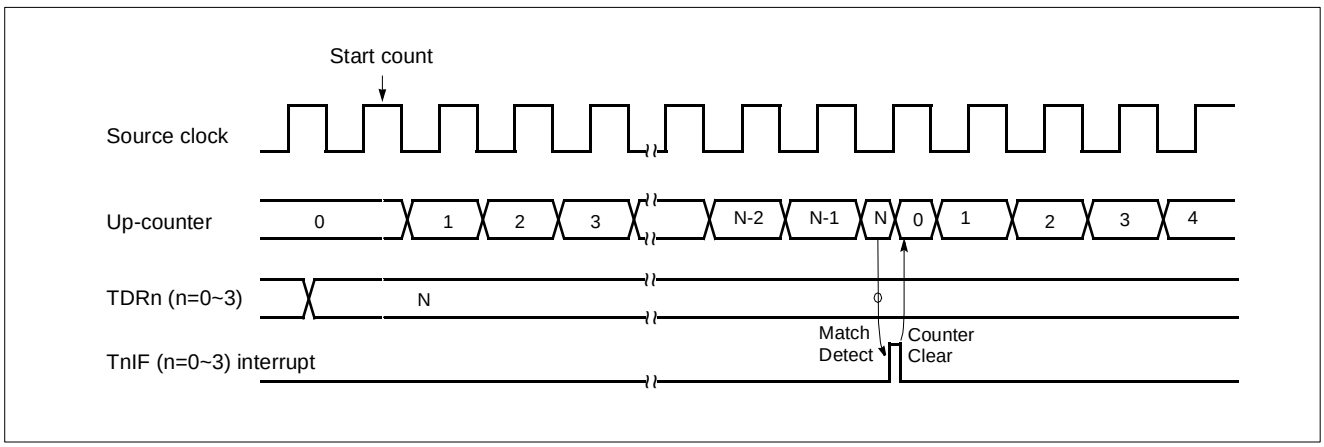


Figure 11-9 Timer Mode Timing Chart

Event counter Mode

In event timer mode, counting up is started by an external trigger. This trigger means falling edge of the ECn (n=2~3) pin input. Source clock is used as an internal clock selected with TM2. The contents of TDRn are compared with the contents of the up-counter. If a match is found, an TnIF interrupt is generated, and the counter is cleared to 00_H. The counter is restarted by the falling edge of the ECn pin input.

The maximum frequency applied to the ECn pin is $f_{ex}/2$ [Hz] in main clock mode.

In order to use event counter function, the bit EC2S, EC3 of the Port Function Select Register1 FUNC1(address $0CE_H$) is required to be set to "1".

After reset, the value of TDRn is undefined, it should be

initialized to between 01_H~FF_H, not to 00_H.

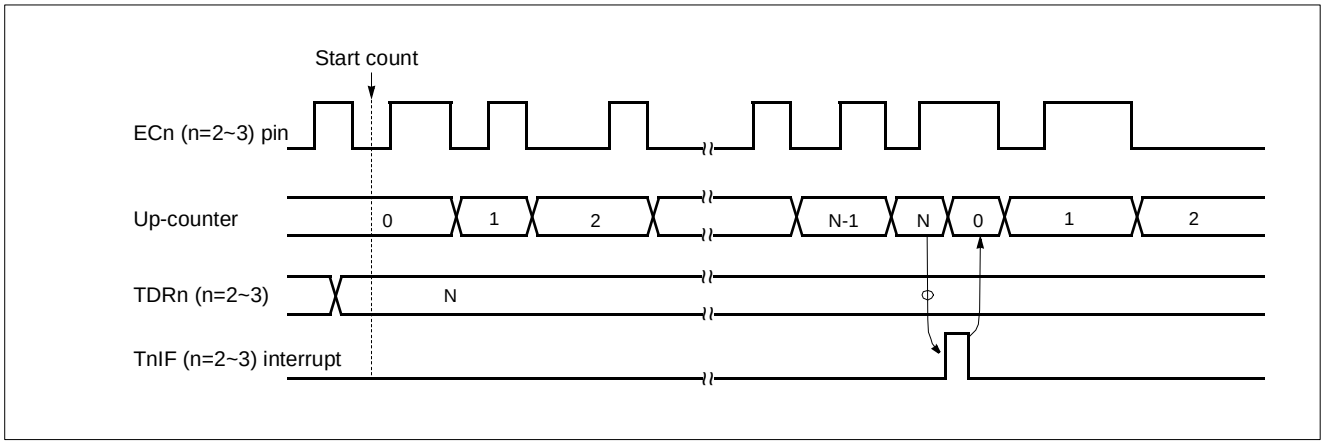


Figure 11-10 Event Counter Mode Timing Chart

The interval period of Timer is calculated as below equation.

$$Period = \frac{1}{f_{ex}} \times Prescaler\ \text{ratio} \times TDRn$$

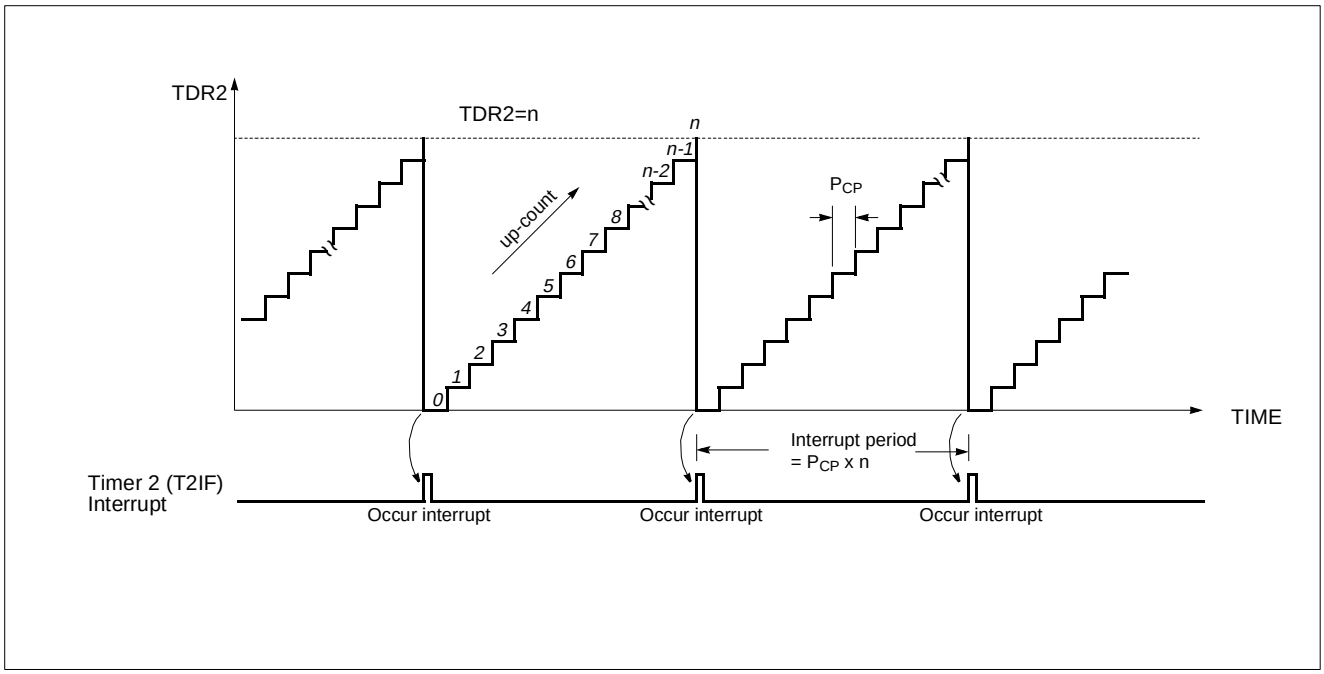


Figure 11-11 Count Example of Timer / Event counter

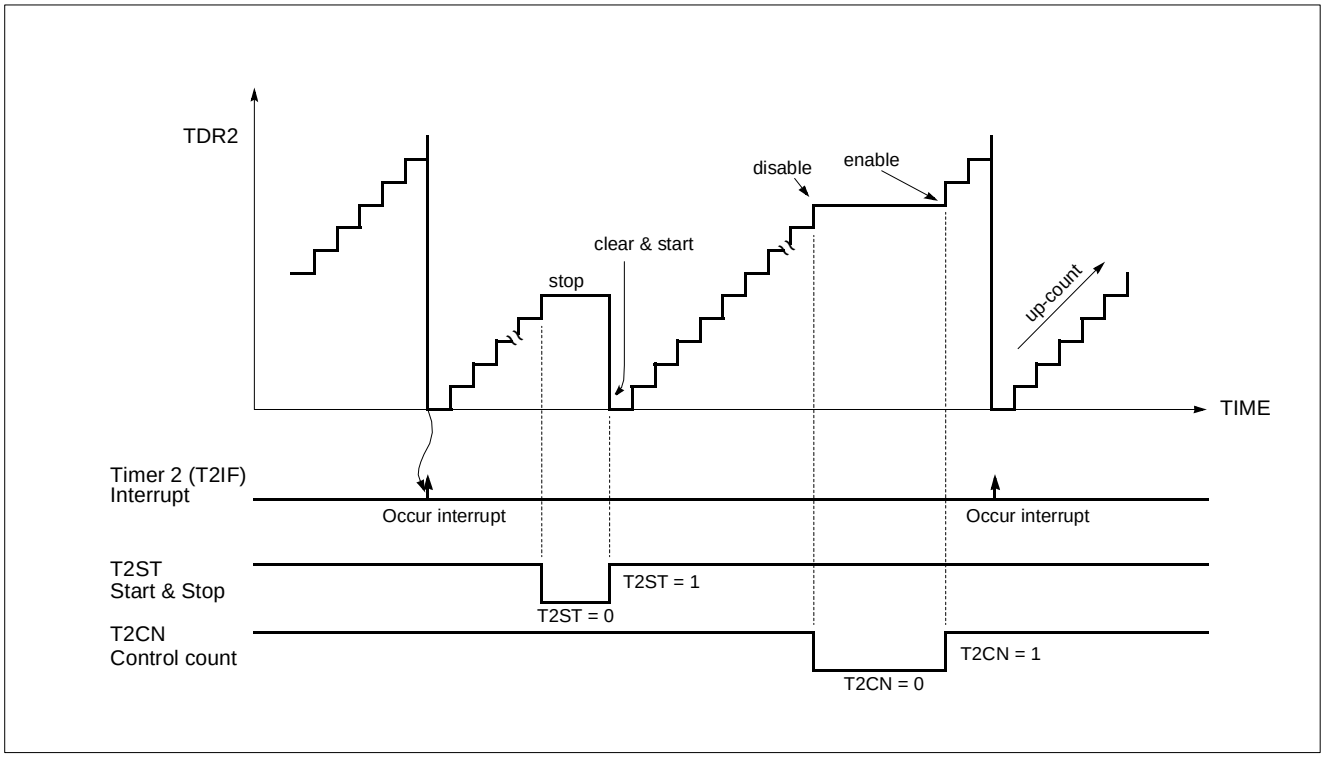


Figure 11-12 Count Operation of Timer / Event counter

12. A/D Converter

The A/D convertor circuit is shown in Figure 12-1 .

The A/D convertor circuit consists of the comparator and control register AIPS($00EF_H$), ADCM($00F0_H$), ADR($00F1_H$). The AIPS register select normal port or analog input. The ADCM register control A/D converter's activity. The ADR register stores A/D converted 8bit result. The more details are shown Figure 12-2 .

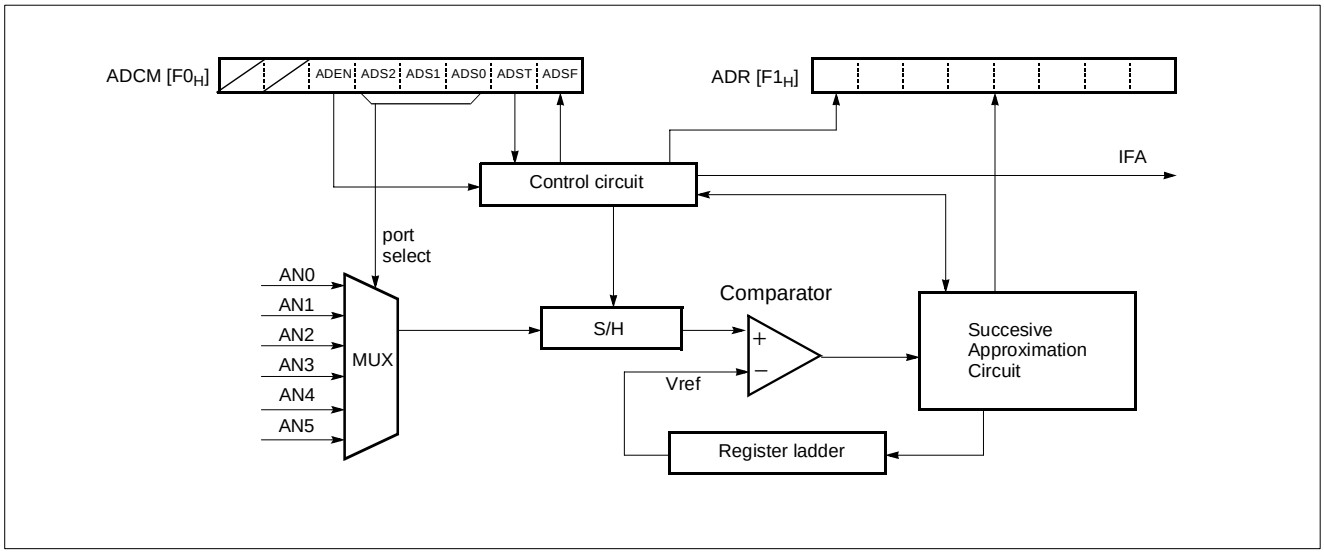


Figure 12-1 Block Diagram of A/D convertor circuit

Control

The HMS81C43xx/GMS87C4060 contains a A/D converter module which has six analog inputs.

1. First of all, you have to select analog input pin by set the ADCM and AIPS.

2. Set ADEN (A/D enable bit : ADCM bit5).

3. Set ADST (A/D start bit : ADCM bit1). We recommend you do not set ADEN and ADST at once, it makes worse A/D converted result.

4. ADST bit will be cleared automatically 1cycle after you set this.

Example:

```
    :
; Set AIPS, change ? to what you want
;         0 : digital port
;         1 : analog port
    LDM   AIPS,#00??????b
; Set ADEN, xxx is analog port number
    LDM   ADCM,#001xxx00b
    ; or "SET1 ADEN"
; Set ADST, xxx is analog port number
    LDM   ADCM,#001xxx10b
    ; or "SET1 ADST"
    :
    :
```

5. After A/D conversion is completed, ADSF bit and interrupt flag IFA will be set. (A/D conversion takes 36 machine cycle : 9uS when f_{ex}=8MHz).

Note: *Make sure AIPS bits, if you using a port which is set digital input by AIPS, analog voltage will be flow into MCU internal logic not A/D converter. Sometimes device or port is damaged permanently.*

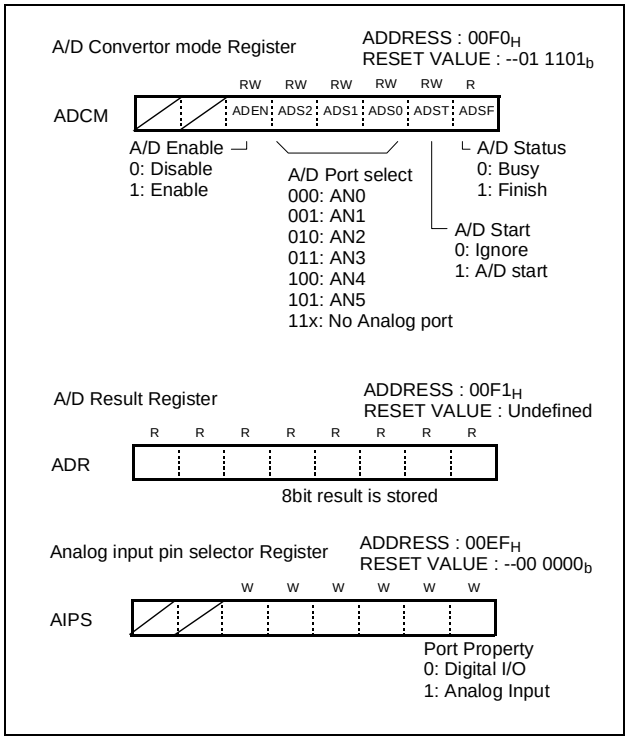


Figure 12-2 A/D convertor Registers

13. Serial I/O

The Serial I/O circuit is shown in Figure 13-1 .

The Serial I/O circuit consists of the octal counter, SIOR(DF_H), SIOM(DE_H). The SIOR register stores received data or data which will be transfered. The SIOM register controls serial communication mode, speed, start, etc.

The more details about registers are shown Figure 13-2 .

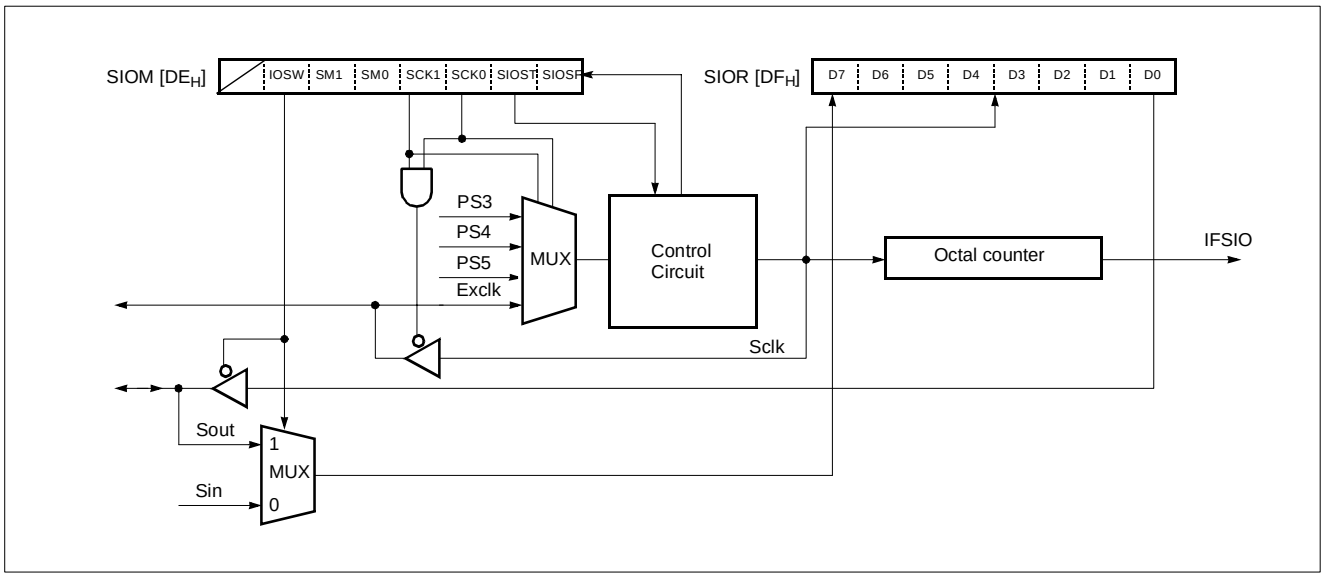


Figure 13-1 Block Diagram of Serial I/O circuit

Control

The HMS81C43xx/GMS87C4060 contains a Synchronous type Serial I/O module.

1. You have to select serial I/O pins by set the SM1~0.

SM1	SM0	Function	Port select		
			R21	R22	R23
0	0	-	R21	R22	R23
0	1	Send	Sclk	Sout	R23
1	0	Receive	Sclk	R22	Sin
1	1	-	R21	R22	R23

Note: *Sout pin can handle serial data output or serial data input. You can input serial data to Sout pin when IOSW bit is 1. But Sin pin is dedicated serial data input pin.*

2. You have to select serial communication clock by set the SCK1~0.

SCK1	SCK0	Selected Clock	Ex: Frequency (f_{ex}=8MHz)
0	0	PS3	1uS
0	1	PS4	2uS
1	0	PS5	4uS
1	1	External clock	User define

3. If you want to send data, write it to SIOR. Or not skip this.

4. Start serial communication by set SIOST(Serial I/O start, SIOM bit1).

5. After serial communication is completed, SIOSF bit and interrupt flag IFSIO will be set.

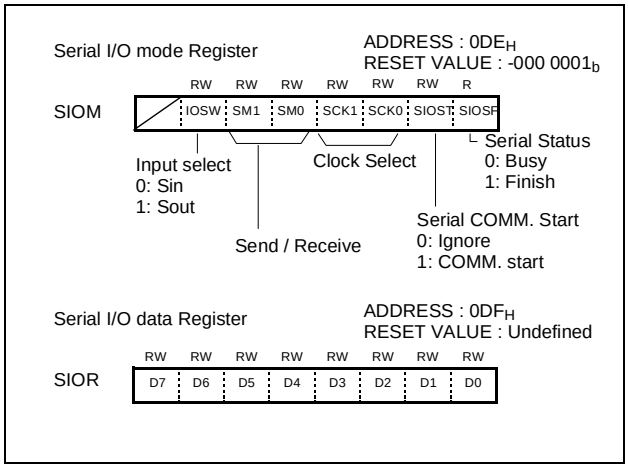


Figure 13-2 Serial I/O Registers

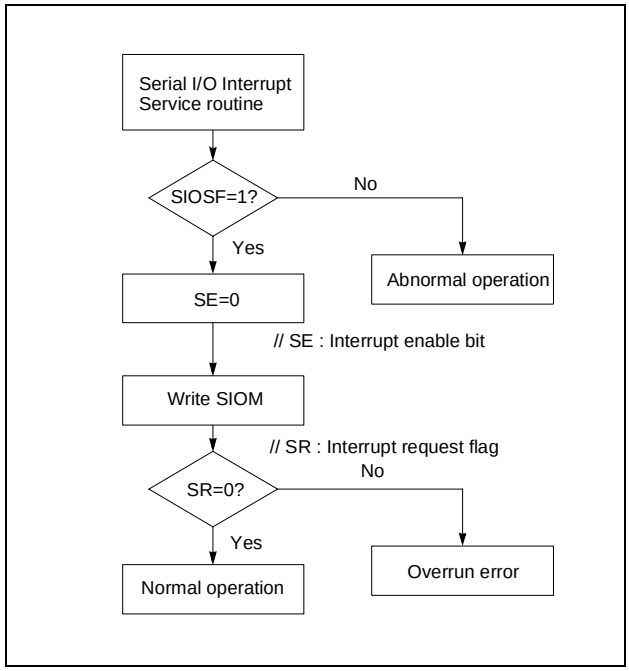


Figure 13-3 Example for serial I/O check by S/W

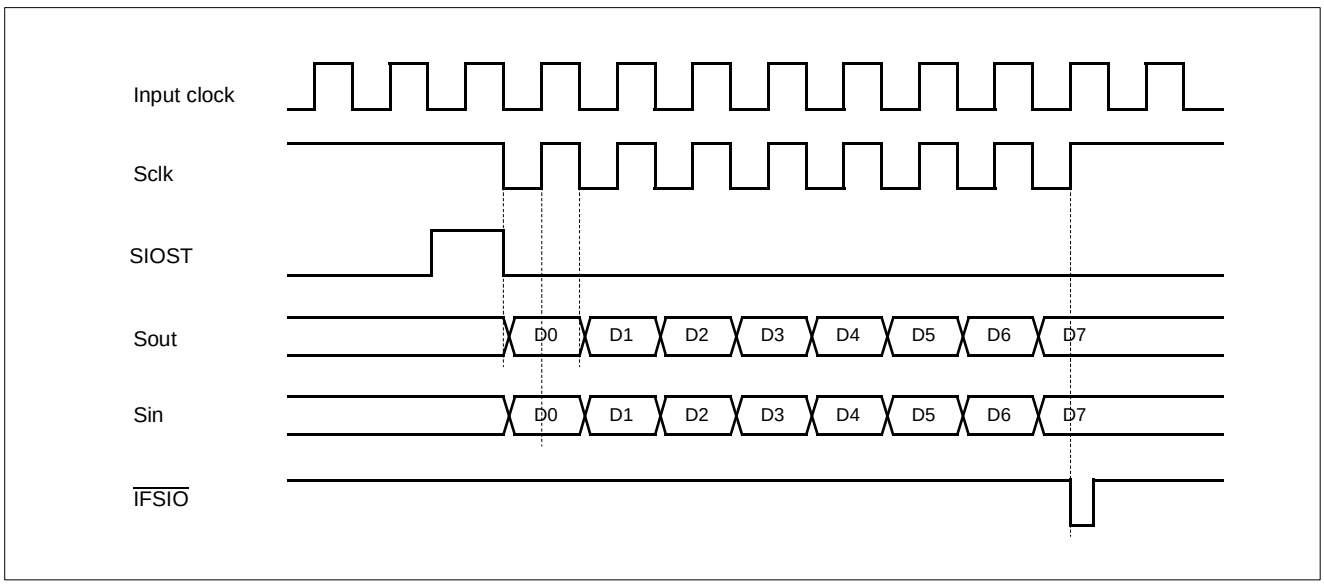


Figure 13-4 Serial I/O Timing Chart

14. Pulse Width Modulation (PWM)

The PWM circuit is shown in Figure 14-1 , Figure .

Example (f_{ex}=8MHz)	14bit PWM	8bit PWM
Resolution	0.5uS	4uS
Input Clock	2MHz	250KHz
Frame cycle	8,192uS	1,024uS

The PWM circuit consists of the counter, comparator, Data register.

The PWM control registers are PWMR7~0, PWMCR2~1, PWM8H, PWM8L.

The more details about registers are shown Figure 14-2 .

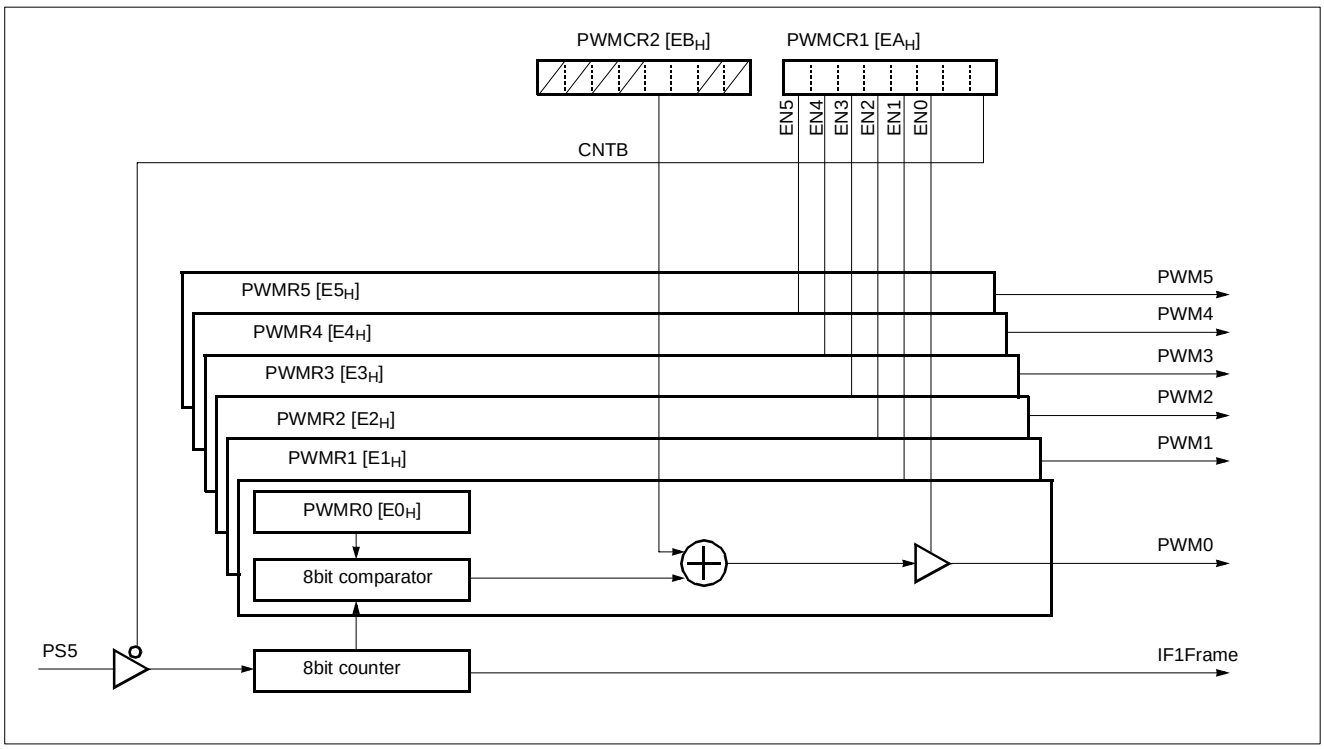


Figure 14-1 8bit register (PWM7~0) circuit

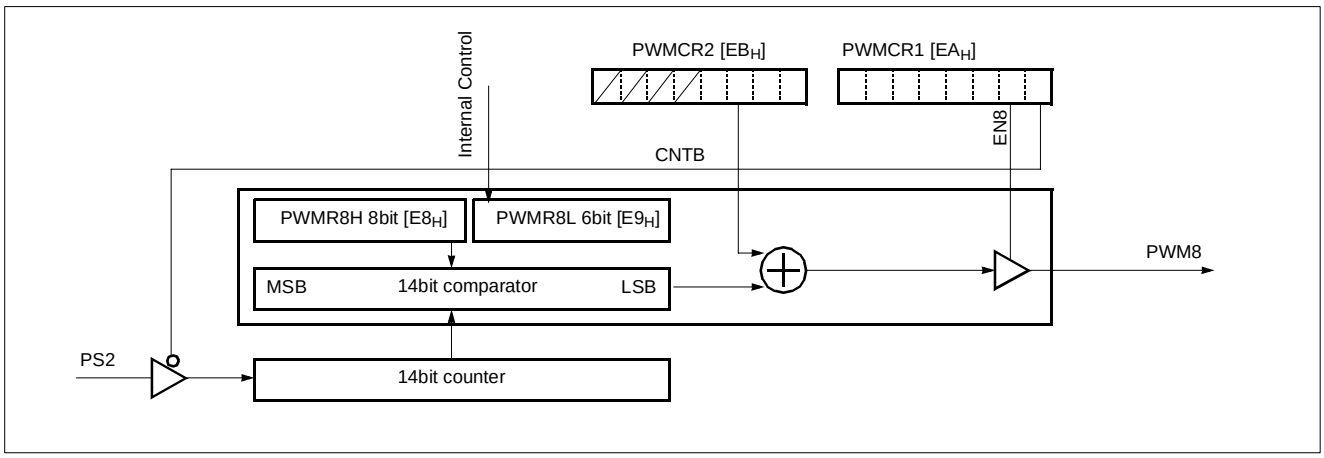


Figure 14-2 14bit register (PWM8) circuit

8bit PWM Control

The HMS81C43xx/GMS87C4060 contains a one 14bit PWM and six 8bit PWM module.

1. 8bit PWM0~5 is wholy same internal circuit, but PWM0~5 output port is NMOS open drain.

2. Al l PWM polarity has the same by POL2's value.

3. Calulate Frame cycle and Pulse width is as following.
   PWM Frame Cycle = $2^{13}/ f_{ex}$ (Sec)
   PWM Width = (PWMRn+1) * $2^5 / f_{ex}$ (n=0~5)
   Pulse Duty (%) = (PWMRn +1) / 256 *100(%) (n=0~5)

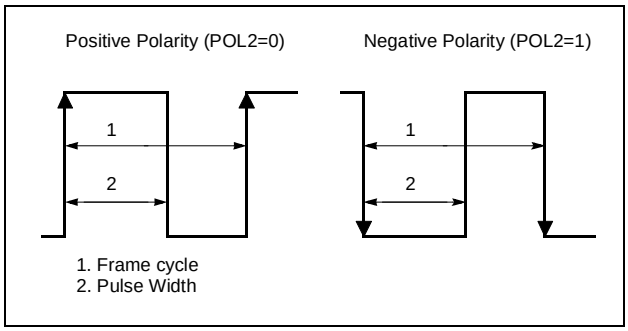


Figure 14-3 Wave form example for 8bit PWM

4. PWM output is enabled during ENn(n=0~5) bit (See PWMCR1~2) contains 1.

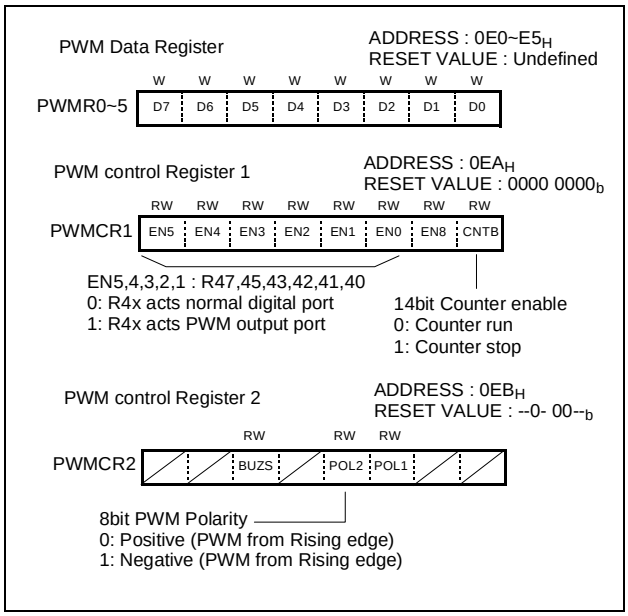


Figure 14-4 8bit PWM Registers

5. CNTB controls all PWM counter enable.
   If CNTB=0, Counter is enabled.

14bit PWM Control

1. 14bit PWM's operation concept is not the same as 8bit PWM.
   1 PWM frame contains 64 sub PWMs.
   PWM8H : Set sub PWM's basic Pulse Width.
   PWM8L : Number of sub PWM which is added 1 clock.

2. PWM polarity is selected by POL1's value.
   If POL1=0, Positive Polarity.

3. Calulate Frame cycle and Pulse width is as following.
   Main PWM Frame Cycle = $2^{16}/ f_{ex}$ (Sec).
   Sub PWM Frame Cycle = Main Frame Cycle / 64.

4. Table 14-1, "PWM8L and Sub frame matching table," on page 48 show PWM8L function.

Bit value	Sub frame number which is added 1 clock	Pulse count
if Bit0=1	32	1
if Bit1=1	16, 48	2
if Bit2=1	8, 24, 40, 56	4
if Bit3=1	4, 12, 20, 28, 36, 44, 52, 60	8
if Bit4=1	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54	16
if Bit5=1	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63	32

Table 14-1 PWM8L and Sub frame matching table

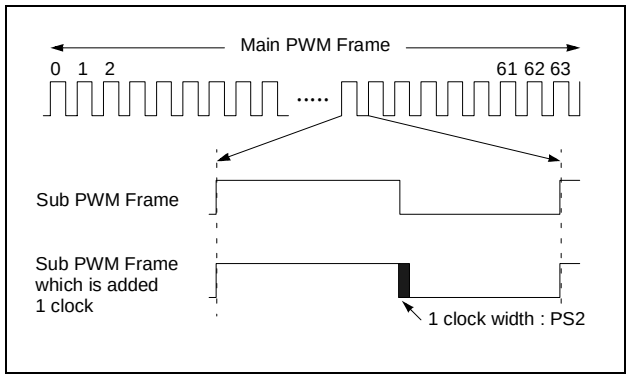


Figure 14-5 Wave form example for 14bit PWM

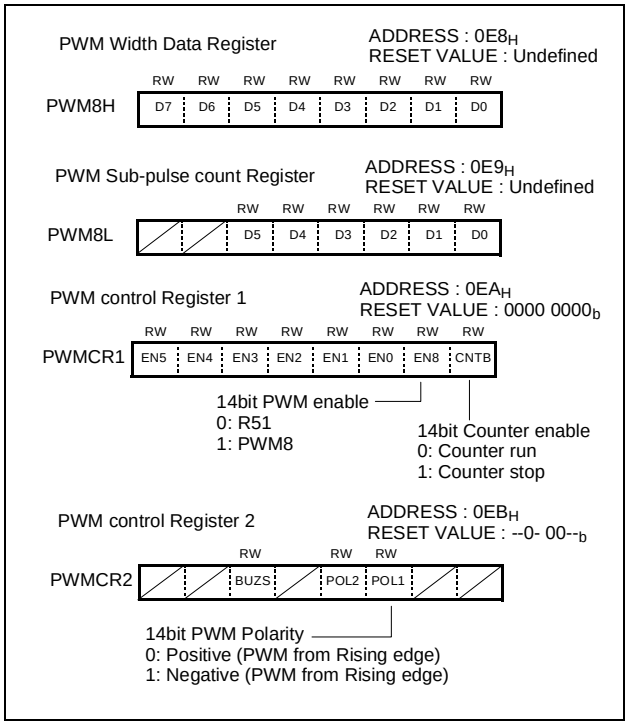


Figure 14-6 14bit PWM Registers

5. PWM output is enabled during EN8 bit contains 1.

6. CNTB controls PWM counter enable.
   If CNTB=0, Counter is enabled.

15. Interrupt interval measurement circuit

The Interrupt interval measurement circuit is shown in Figure 15-1 .

The Interrupt interval measurement circuit consists of the input multiplexer, sampling clock multiplexer, Edge detector, 8bit counter, measured result storing register, FIFO (9bit, 6level) interrupt, Control register, etc.

The more details about registers are shown Figure 15-2 .

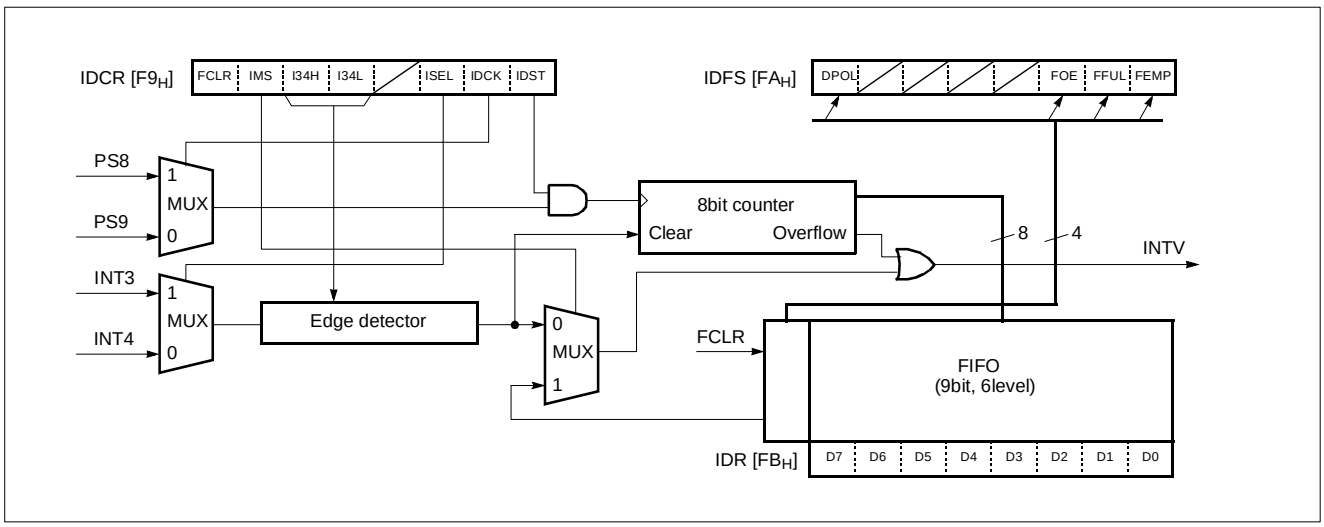


Figure 15-1 Block Diagram of Interrupt interval measurement circuit

Control

The HMS81C43xx/GMS87C4060 contains a Interrupt interval measurement module.

1. Select interrupt input pin what you want to measure by set the FUNC1 [$00CE_H$].

2. Set IDCR [$00F9_H$] : FIFO clear, interrupt mode, interrupt edge select, external interrupt select between INT3 and INT4, sampling clock select.

3. Set IDCR [$00F9_H$] : set IDST to start measuring.

4. Counter value is stored to IDR [$00FB_H$] when selected edge is detected. After data was written, timer is cleard automatically and it counts continue.

5. You can select interrupt occuring point by set Interrupt Mode Select bit (IMS), every edge what you selected or FIFO 4 level is filled.

6. If input signal's interval is larger than maximum counter value ($0FF_H$), counter occurring an interrupt and count again from 00_H.

7. See Figure 15-4 FIFO operating mechanism.

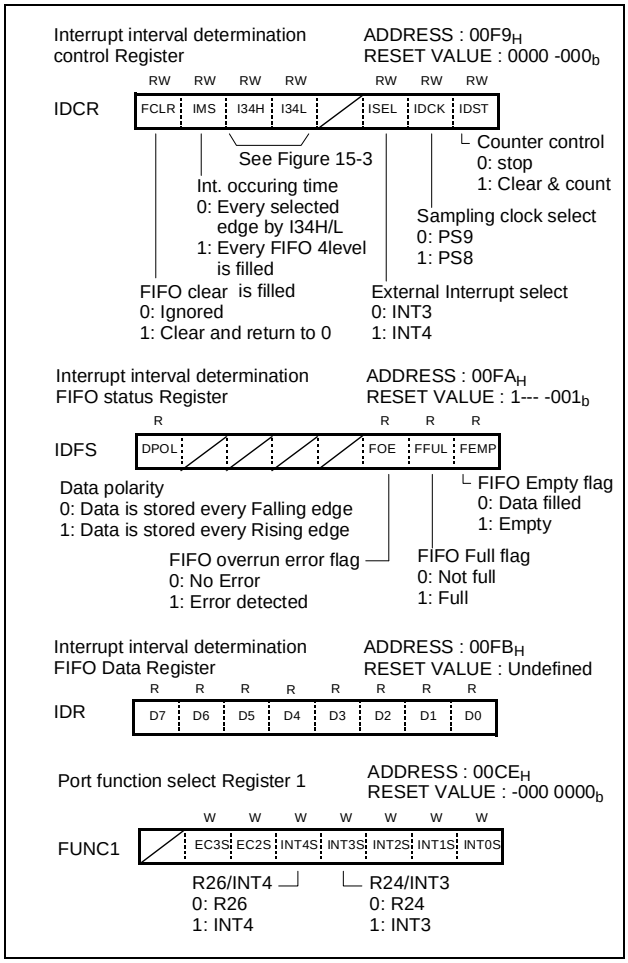
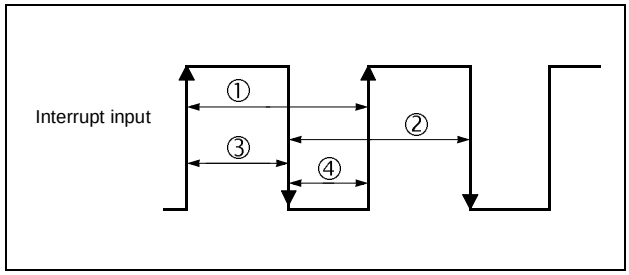


Figure 15-2 Int. interval measurement Registers



Item	Symbol	I34H	I34L	Detecting edge
Frame Cycle	①	1	0	Rising edge
	②	0	1	Falling edge
Pulse width	③	1	1	Both edge
	④	1	1	Both edge

Figure 15-3 Setting for measurement

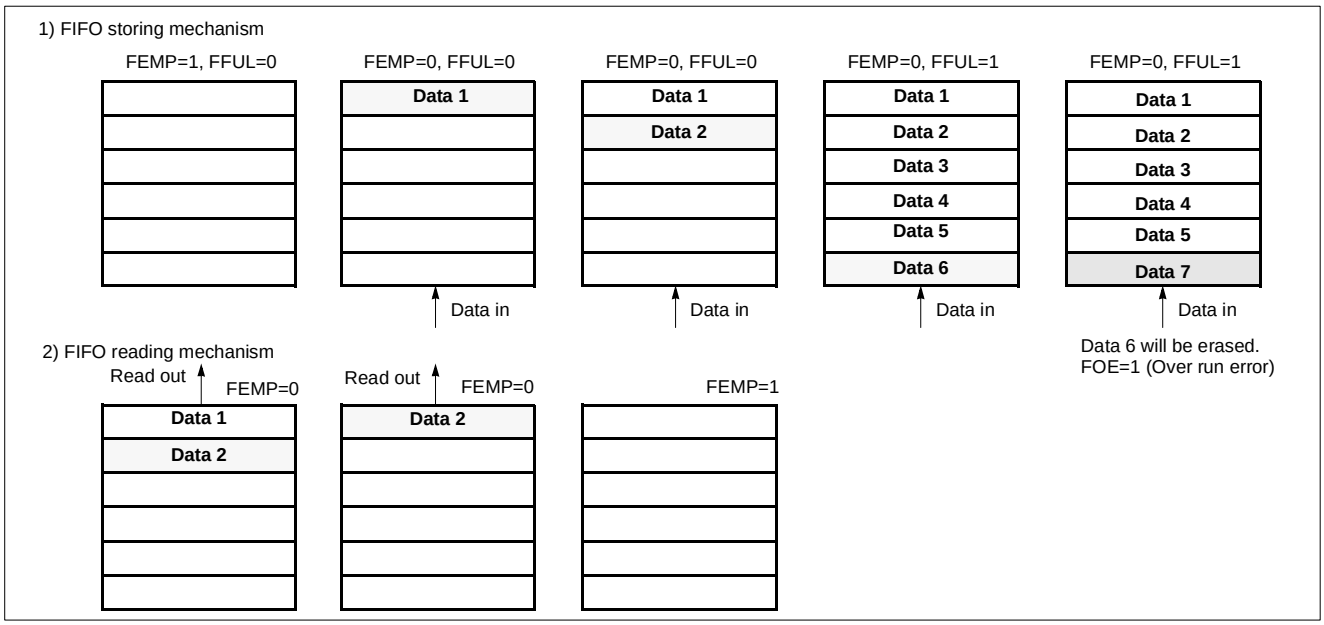


Figure 15-4 Example for FIFO operating mechanism

16. Buzzer driver

The Buzzer driver circuit is shown in Figure 16-1 .

The Buzzer driver circuit consists of the 6bit counter, 6bit comparator, Buzzer data register BUR($00EE_H$). The BUR register controls source clock and output frequency.

The more details about registers are shown Figure 16-2 .

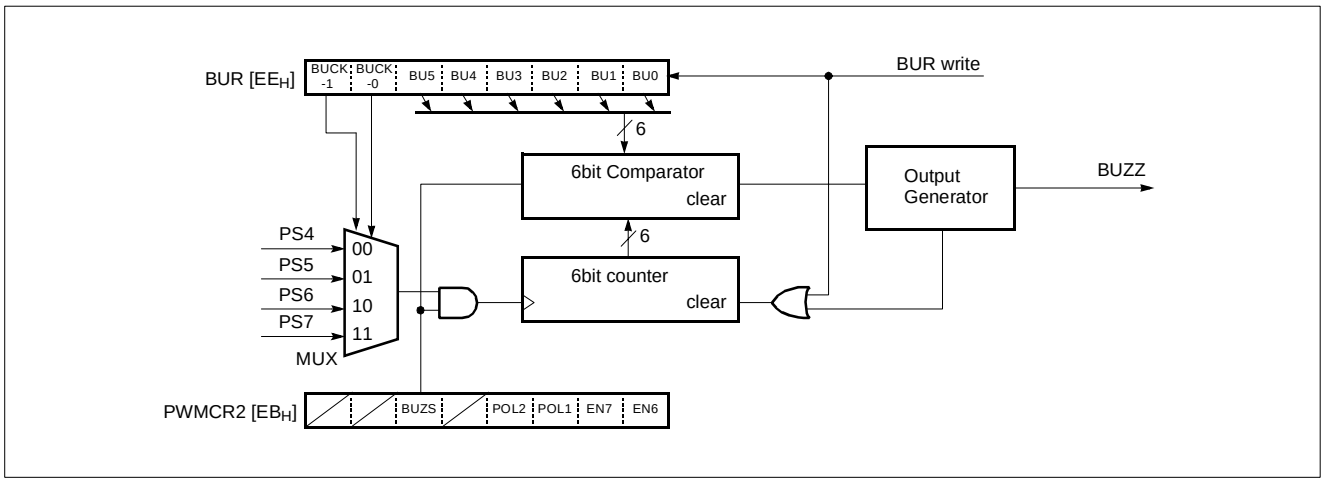


Figure 16-1 Block Diagram of Buzzer driver circuit

Control

The HMS81C43xx/GMS87C4060 contains a Buzzer driver module.

1. Select an input clock among PS4~7 by set the BUCK1~0 of BUR.

BUCK1	BUCK0	Clock source
0	0	PS4
0	1	PS5
1	0	PS6
1	1	PS7

2. Select output frequency by change the BU5~0.
   Output frequency = 1 / (PSx * BUy *2) Hz.
   x=4~7, y=5~0
   See example Table 16-1 and Table 16-2.

Note: *Do not select 00_H to BU5~0. It means counter stop.*

3. Set BUZS bit for output enable.

4. Output waveform is rectagle clock which has 50% duty.

5. You can use this clock for the other purposes.

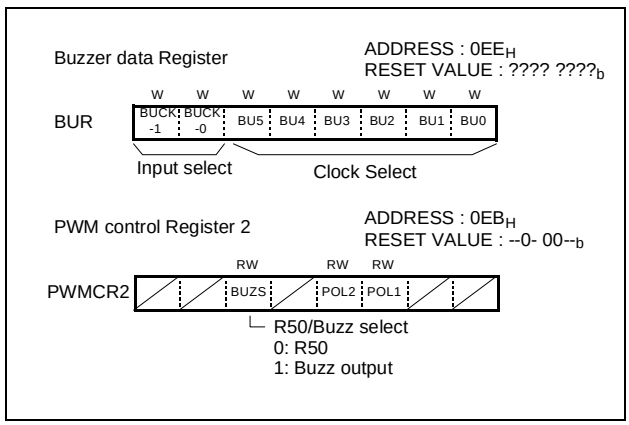


Figure 16-2 Buzzer driver Registers

BUR5~0		Output frequency (KHz)			
Dec	Hex	PS4	PS5	PS6	PS7
1	01	250	125	62.5	31.25
2	02	125	62.5	31.25	15.625
3	03	83.333	42.666	20.833	10.436
4	04	62.5	31.25	15.625	7.813
5	05	50	25	12.5	6.25
6	06	41.666	20.888	10.461	5.208
7	07	35.714	17.858	8.928	4.464
8	08	31.25	15.625	7.813	3.907
9	09	27.728	13.888	6.944	3.472
10	0A	25	12.5	6.25	3.125
11	0B	22.728	11.364	5.682	2.841
12	0C	20.834	10.417	5.209	2.604
13	0D	19.23	9.615	4.808	2.404
14	0E	17.858	8.929	4.484	2.242
15	0F	16.666	8.333	4.166	2.083
16	10	15.626	7.813	3.906	1.953
17	11	14.706	7.353	3.676	1.838
18	12	13.888	6.944	3.472	1.736
19	13	13.158	6.579	3.289	1.644
20	14	12.5	6.25	3.125	1.562
21	15	11.904	5.952	2.976	1.438
22	16	11.364	5.682	2.841	1.420
23	17	10.87	5.435	2.718	1.359
24	18	10.416	5.208	2.604	1.302
25	19	10	5	2.5	1.25
26	1A	9.616	4.808	2.404	1.202
27	1B	9.26	4.63	2.315	1.158
28	1C	8.928	4.464	2.232	1.116
29	1D	8.62	4.31	2.155	1.078
30	1E	8.334	4.167	2.084	1.042
31	1F	8.064	4.032	2.016	1.008
32	20	7.812	3.906	1.953	0.976
33	21	7.576	3.788	1.894	0.947
34	22	7.352	3.676	1.838	0.919
35	23	7.124	3.571	1.786	0.893
36	24	6.944	3.472	1.736	0.868
37	25	6.756	3.378	1.689	0.845
38	26	6.578	3.289	1.645	0.822
39	27	6.41	3.205	1.602	0.801
40	28	6.25	3.125	1.563	0.781
41	29	6.098	3.049	1.524	0.762
42	2A	5.952	2.976	1.488	0.744
43	2B	5.814	2.907	1.453	0.727
44	2C	5.682	2.841	1.421	0.710
45	2D	5.556	2.778	1.389	0.694
46	2E	5.434	2.717	1.359	0.679
47	2F	5.32	2.66	1.33	0.665
48	30	5.208	2.604	1.302	0.651
49	31	5.102	2.551	1.276	0.638
50	32	5	2.5	1.25	0.625
51	33	4.902	2.451	1.225	0.613
52	34	4.808	2.404	1.202	0.601
53	35	4.716	2.358	1.179	0.590
54	36	4.63	2.315	1.157	0.579
55	37	4.546	2.273	1.136	0.568
56	38	4.464	2.232	1.116	0.558
57	39	4.386	2.193	1.096	0.548
58	3A	4.31	2.155	1.078	0.539
59	3B	4.238	2.119	1.059	0.530
60	3C	4.166	2.083	1.042	0.521
61	3D	4.098	2.049	1.025	0.512
62	3E	4.032	2.016	1.008	0.504
63	3F	3.968	1.984	0.992	0.496

Table 16-1 . Example for f_{ex}=8MHz

BUR5~0		Output frequency (KHz)			
Dec	Hex	PS4	PS5	PS6	PS7
1	01	375	187.5	93.75	46.875
2	02	187.5	93.75	46.875	23.438
3	03	125	62.5	31.35	15.625
4	04	93.75	46.875	23.436	11.719
5	05	75	37.5	18.75	9.375
6	06	62.5	31.25	15.625	7.813
7	07	53.572	26.786	13.393	6.696
8	08	46.875	23.436	11.719	5.895
9	09	41.666	20.833	10.417	5.208
10	0A	37.5	18.75	9.375	4.688
11	0B	34.09	17.045	8.523	4.261
12	0C	31.25	15.625	7.813	3.906
13	0D	28.846	14.423	7.211	3.606
14	0E	26.786	13.393	6.696	3.348
15	0F	25	12.5	6.25	3.125
16	10	23.436	11.719	5.859	2.930
17	11	22.058	11.029	5.515	2.757
18	12	20.833	10.417	5.208	2.604
19	13	19.736	9.868	4.934	2.467
20	14	18.75	9.375	4.688	2.344
21	15	17.858	8.929	4.464	2.232
22	16	17.045	8.523	4.261	2.131
23	17	16.304	8.152	4.076	2.038
24	18	15.625	7.813	3.906	1.953
25	19	15	7.5	3.75	1.875
26	1A	14.424	7.212	3.606	1.803
27	1B	13.888	6.944	3.472	1.736
28	1C	13.393	6.696	3.348	1.674
29	1D	12.932	6.466	3.233	1.616
30	1E	12.5	6.25	3.125	1.563
31	1F	12.096	6.048	3.024	1.512
32	20	11.718	5.859	2.930	1.465
33	21	11.364	5.682	2.841	1.420
34	22	11.03	5.515	2.757	1.379
35	23	10.714	5.357	2.679	1.339
36	24	10.416	5.208	2.604	1.302
37	25	10.136	5.068	2.534	1.267
38	26	9.868	4.934	2.467	1.234
39	27	9.616	4.808	2.404	1.202
40	28	9.375	4.688	2.344	1.172
41	29	9.146	4.573	2.287	1.143
42	2A	8.929	4.464	2.232	1.116
43	2B	8.72	4.36	2.18	1.09
44	2C	8.523	4.261	2.131	1.065
45	2D	8.334	4.167	2.083	1.042
46	2E	8.152	4.076	2.038	1.019
47	2F	7.978	3.989	1.995	0.997
48	30	7.813	3.906	1.953	0.977
49	31	7.654	3.827	1.913	0.957
50	32	7.5	3.75	1.875	0.938
51	33	7.352	3.676	1.838	0.919
52	34	7.212	3.606	1.802	0.901
53	35	7.076	3.538	1.769	0.884
54	36	6.944	3.472	1.736	0.868
55	37	6.818	3.409	1.705	0.852
56	38	6.696	3.348	1.674	0.837
57	39	6.578	3.289	1.645	0.822
58	3A	6.466	3.233	1.616	0.808
59	3B	6.356	3.178	1.589	0.795
60	3C	6.25	3.125	1.563	0.781
61	3D	6.148	3.074	1.537	0.768
62	3E	6.048	3.024	1.512	0.756
63	3F	5.952	2.976	1.488	0.744

Table 16-2 . Example for f_{ex}=12MHz

17. On Screen Display (OSD)

The On Screen Display circuit is shown in Figure 17-1 .

The HMS81C43xx/GMS87C4060 can support 512 OSD chacters, but the last 6 characters (number 506 ~ 511, $1FA_H$ ~ $1FF_H$) are reserved for IC test and its pattern is fixed by manufacturer. So you can use 506 characters for your own.

The OSD circuit consists of the Position attribute register, Line register, Full screen screen control register, sprite control register, sprite position reigster, I/O polarity register, sprite RAM, font ROM, VRAM, etc. The more details about registers are shown Figure 17-2.

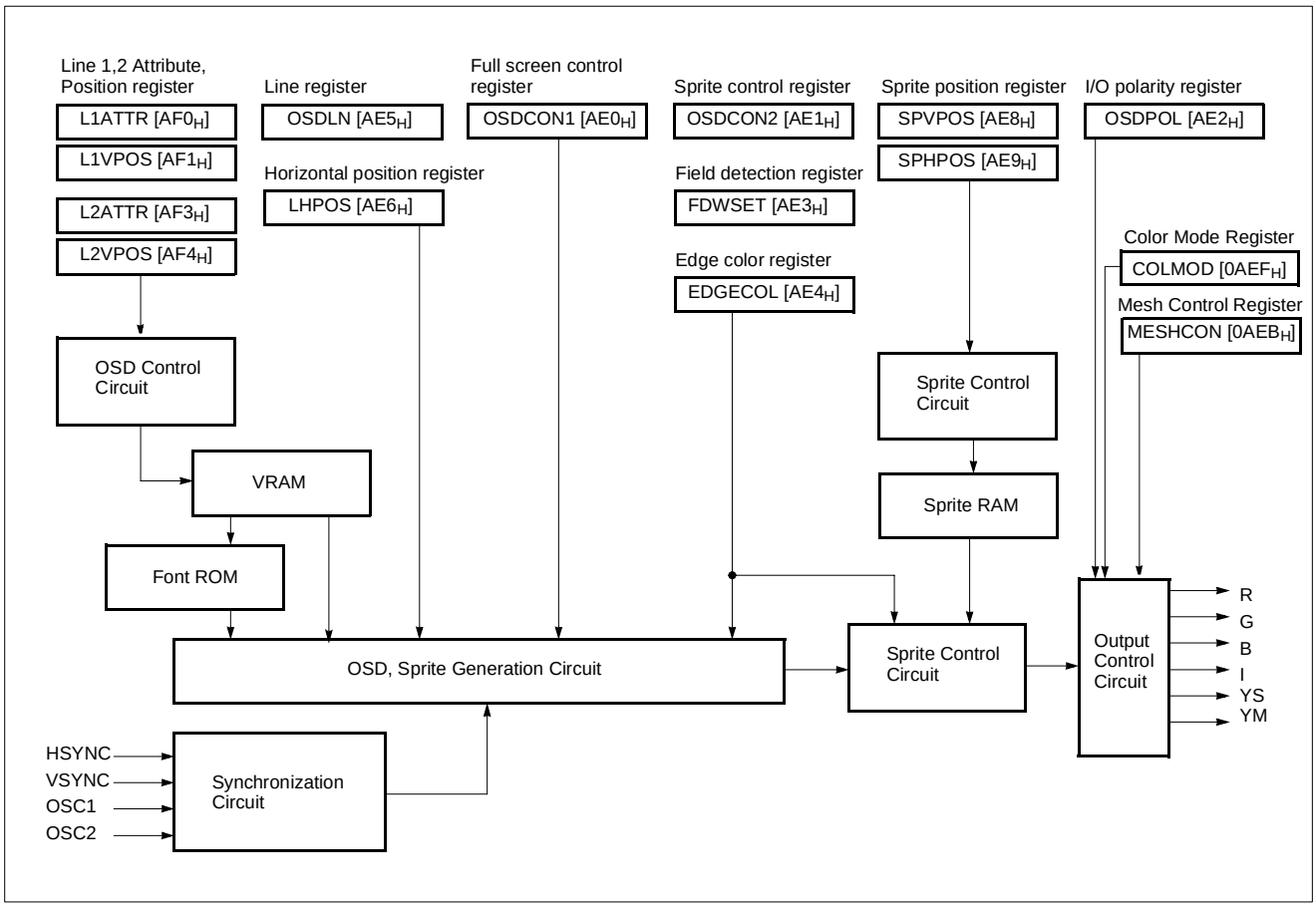


Figure 17-1 Block Diagram of On Screen Display circuit

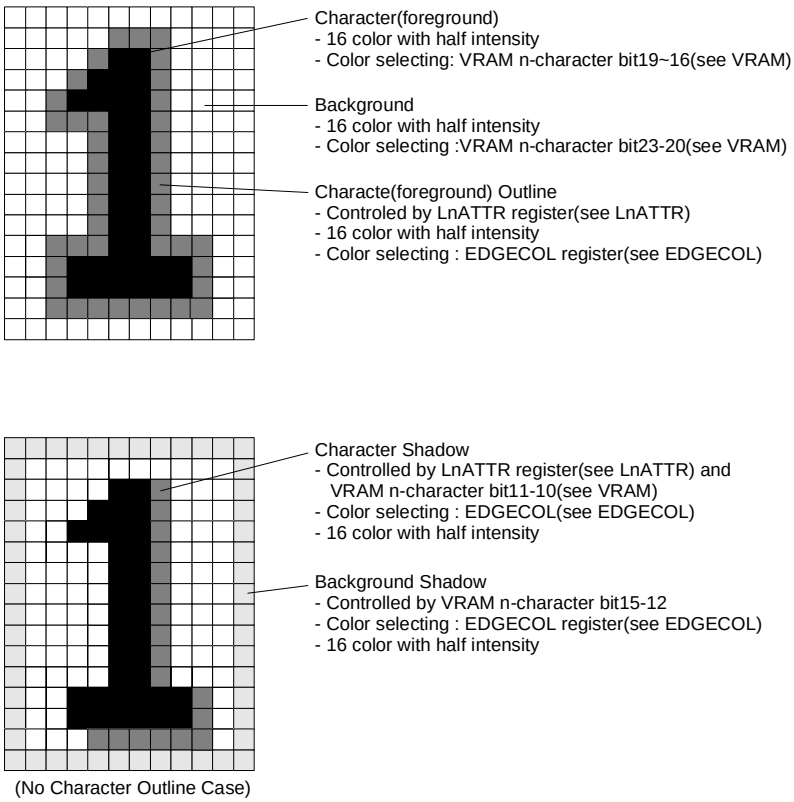


Figure 17-2 OSD Character Font Example

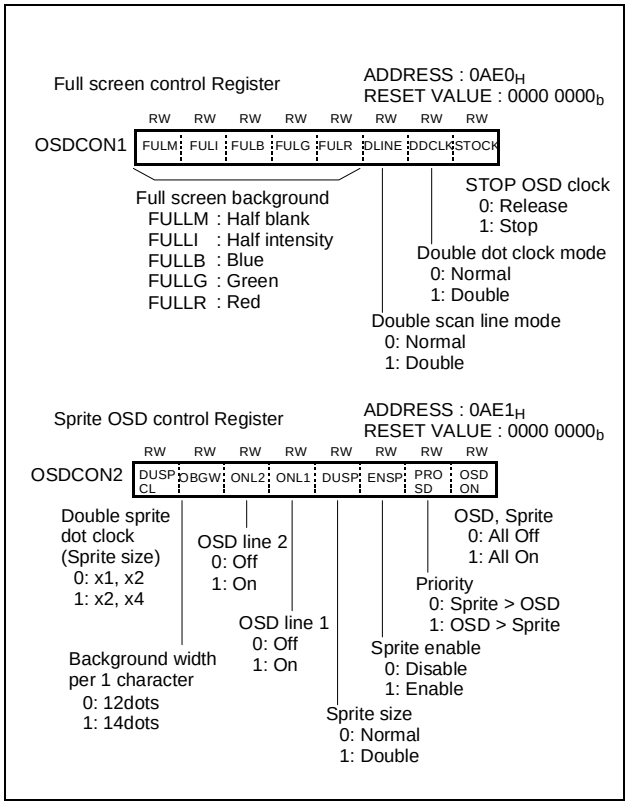


Figure 17-3 OSD Control Registers - 1

OSDCON1

bit 0: STOCK

It controls OSD LC oscillation. If oscillation is stoped, IC's power consumption is decreased.

bit 1: DDCLK

If you set this bit to 1, OSD input clock is doubled from LC oscillation. It makes OSD horisontal image size as doubled.

bit 2: DLINE

If you set this bit to 1, OSD vertical scan counter input clock is doubled from normal state. It makes OSD vertical image size as doubled.

bit 7~3: FULLM, I, B, G, R

It controls back ground color as below.

M	I	B	G	R	Color
0	0	0	0	0	Transparent (Normal TV)
0	0	0	0	1	RED

Table 17-1 Full Screen Back ground color selection

M	I	B	G	R	Color
0	0	0	1	0	GREEN
0	0	0	1	1	RED+GREEN
0	0	1	0	0	BLUE
0	0	1	0	1	BLUE+RED
0	0	1	1	0	BLUE+GREEN
0	0	1	1	1	RED+GREEN+BLUE (WHITE)
0	1	0	0	0	BLACK
0	1	0	0	1	Half intensity RED
0	1	0	1	0	Half intensity GREEN
0	1	0	1	1	Half intencity RED+GREEN
0	1	1	0	0	Half intensity BLUE
0	1	1	0	1	Half intensity GREEN+BLUE
0	1	1	1	1	Half intensity WHITE
1	0	0	0	0	Half BLANK

Table 17-1 Full Screen Back ground color selection

OSDCON2

bit 0: OSDON

It controls OSD, Sprite, Full screen background at once. It does not affect anything to Vsync interrupt and OSD interrupt, etc.

bit 1: PROSD

It controls screen output priority between sprite and OSD. If its value is 1, OSD hide sprite pattern in overapped area.

bit 2: ENSP

It enables sprite display.

bit 3: DUSP

It doubles sprite's horizontal & vertical size during this value is 1.

bit 4: ONL1

It enables OSD line 1 display. If it is enabled, OSD interrupt is activated.

bit 5: ONL2

It enables OSD line 2 display. If it is enabled, OSD interrupt is activated.

bit 6: OBGW

It controls character's width. Default width is 12dots. If its value is set, 2 dots (background color) are added both left

and right side of character.

bit 7: DUSPCL

It controls sprite's dot clock and scan line speed. It does not affect to OSD. Sprite size is controlled as below.

DUSPCL	DUSP	Size	
0	0	Normal	12x16
0	1	x 2	24x32
1	0	Not used	-
1	1	x 4	48x64

Table 17-2 Sprite pattern size

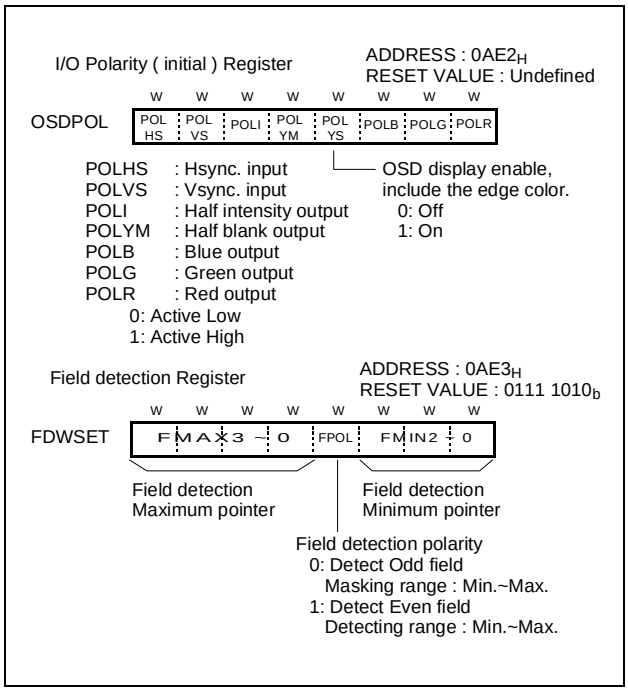


Figure 17-4 OSD Registers - 2

OSDPOL

bit7~0 : POL HS, VS, I, YM, YS, B, G, R

It controls HS, VS, I, YM, YS, B, G, R port's polarity. If its value is 1, polarity is active high.

FDWSET

FDWSET (Field Detection Window Seting) register detects the begin of VSync(Vertical Sync.) signal and distinguishs its current field is Even field or Odd field.

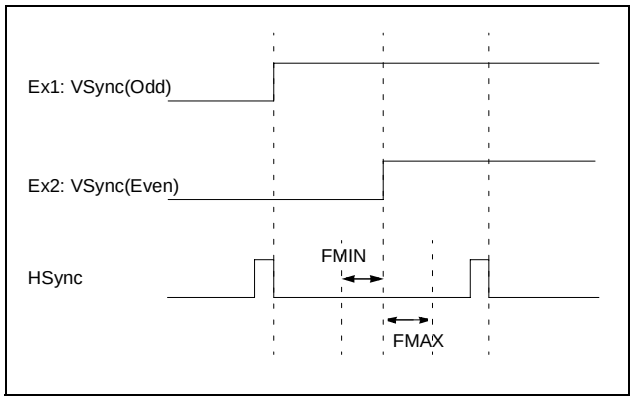


Figure 17-5 FDWSET detection region

The region of FMIN[2:0] ~ FMAX[3:0] is field detection window.

FMAX[3:0] can divide the region between HSync(Horizontal Sync.) by 16. You can assume there is 4 bit horizontal counter, for example HCOUNT[3:0] which count 0~15.

If the start of VSync is detected at the window, next field is even. Else if VSync is detected another region of the window, next field is odd.

It means start of VSync is detected during FMIN[2:0] < HCOUNT[3:0] < FMAX[3:0] and FPOL value is 0, it distinguish odd field.

And, start of VSync is detected during FMIN[2:0] < HCOUNT[3:0] < FMAX[3:0] and FPOL value is 1, it distinguish even field.

FMIN[2:0], FMAX[3:0] are compared with the horizontal counter in OSD block.

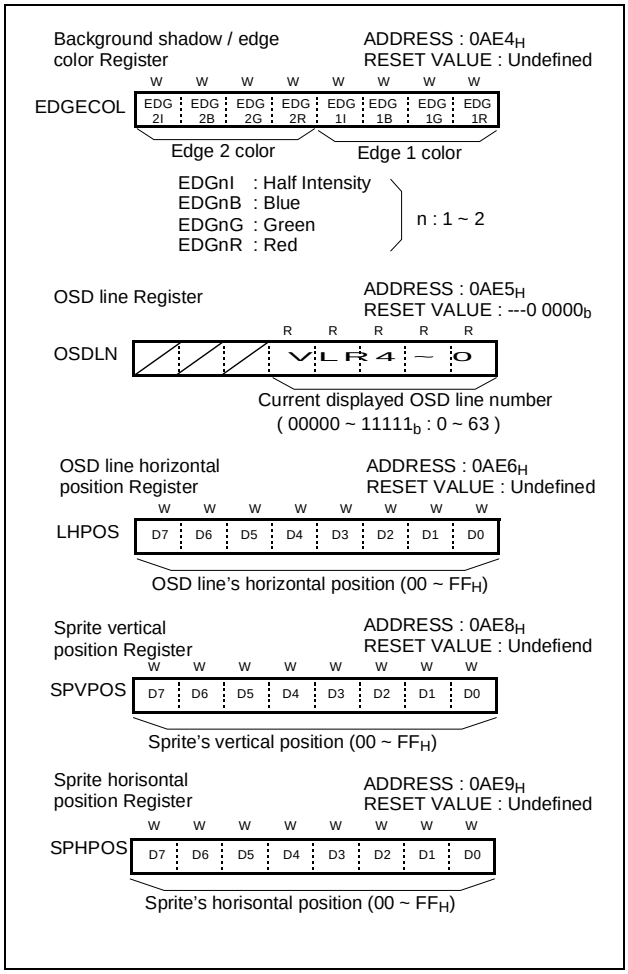


Figure 17-6 OSD Registers - 3

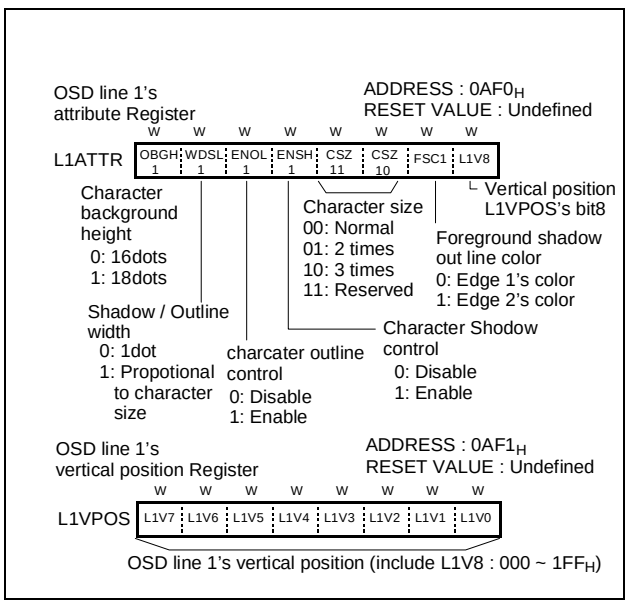


Figure 17-7 OSD Registers - 4

L1ATTR

bit 0 : LIV8

It is equivalent with L1VPOS's bit 8. See more details in L1VPOS.

bit 1: FSC1

It selects character outline and shadow color. If it is 1, it select EDGE2 color in EDGECOL register. Or not, it select EDGE1 color. According to EDGECOL register and this bit character and shadow colors are selected simulteneously

bit 3~2: CSZ11~CSZ10

It controls OSD character's size (x1, x2, x3). You can use this register and DDCLK, DLINE bit, horizontal / vertical size can be controlled (x2, x4, x6).

bit 4: ENSH1

It enables line 1's character(foreground) shadow.

bit 5: ENOL1

It enables line 1's character(foreground) outline.

bit 6: WDSL1

It shows thickness of line 1's shadow and outline.

WDSL	ENOL	ENSH	outline, shadow
0	0	0	No outline, No shadow
0	0	1	Thin shadow
0	1	0	Thin outline
0	1	1	Thin outline Thick shadow
1	0	0	No outline, No shadow
1	0	1	Thick shadow
1	1	0	Thick outline
1	1	1	Thick outline Thick shadow

Table 17-3 Character Outline, Shadow table

bit 7: OBGH1

It controls character's height. Default height is 16dots. If its value is set, 2 dots (background color) are added both top and bottom side of character.

L1VPOS

It shows OSD line 1's vertical position in 9bit format (LIV8 + L1VPOS, 000 ~ 1FF$_H$).

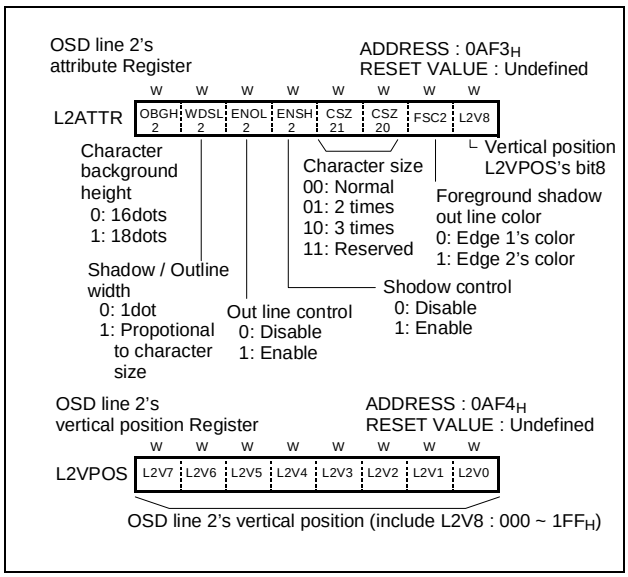


Figure 17-8 OSD Registers - 5

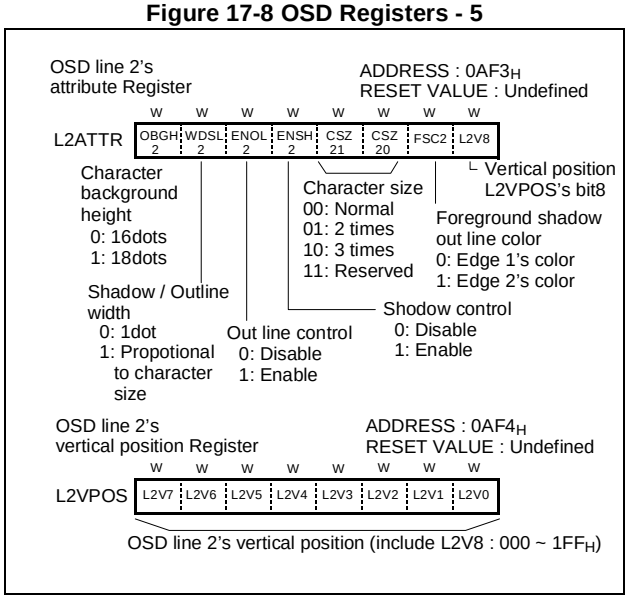


Figure 17-9 OSD Register - 6

L2ATTR

It controls OSD line 2's attributes. Its function is the same as L1ATTR.

L2VPOS

It shows OSD line 2's vertical position. Its function is the same as L1VPOS.

COLMOD

It controls OSD output mode-RGB direct half intencity.

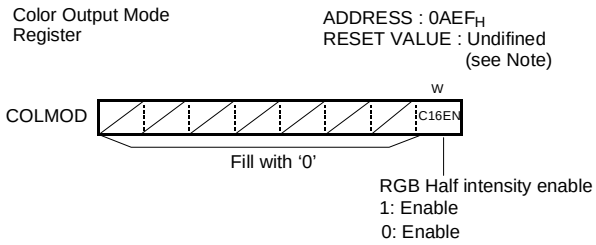


Figure 17-10 OSD Register - 7

bit 0: C16EN

It enables RGB port half intencity output. When this bit is set, RGB port generates half intencity output. Half intencity output is 3.5V voltage level output of RGB port. When you use this bit, you must fill all the other bit with '0'.

Note: *When you do not use RGB direct half intncsity output , please initialize this register as 00h.*

MESHCON

It controls OSD mesh mode color.

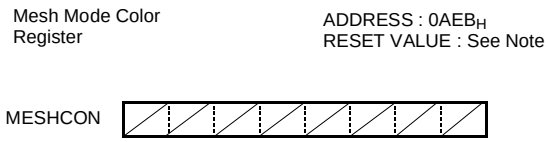


Figure 17-11 OSD Register - 8

Note: *Please initialize this register as 00h. Though this register is for mesh mode color, it is not used currently.*

VRAM

VRAM contains 1 OSD line, 24 character's attributes.

Each character's attribute is constructed with 3 bytes, it contains color data for background, shadow, outline, character and character number (000$_H$ ~ 1FF$_H$, 512 characters

), etc.

Line No.	Character No.	Address (bit 23~0) Hexa decimal		
1	1	A40	A20	A00
	2	A41	A21	A01
	3	A42	A22	A02
	:	:	:	:
	22	A55	A35	A15
	23	A56	A36	A16
	24	A57	A37	A17
2	1	AC0	AA0	A80
	2	AC1	AA1	A81
	3	AC2	AA2	A82
	:	:	:	:
	22	AD5	AB5	A95
	23	AD6	AB6	A96
	24	AD7	AB7	A97

Table 17-4 VRAM memory map

Bit No.	Name	Function
15	BSR	Enable right side background shadow. **cf**. If BSL=1 and BSCUL=1 and LnATTR.ENSHn=1, character's right bottom shadow is shifted to right side by 1dot unit. It acts continued until current character's right side chacter's BSR is set to 1.
14	BSL	Enable left side background shadow.
13	BSD	Enable bottom side background shadow.
12	BSU	Enable top side background shadow.
11	BSCDR	**Select color of right and bottom side shadow of the background** 0: Edge1, 1: Edge2 color

Table 17-5 VRAM (bit15~0) function

Bit No.	Name	Function
10	BSCUL	Select color of left and top side shadow of the background 0: Edge1, 1: Edge2 color
9	ENRND	Enable character's rounding
8~0	CG8~0	Character font number (among 000 ~ $1FF_H$)

Table 17-5 VRAM (bit15~0) function

Note: *if (BSL = 1) & (BSCUL = 0) & (LnATTR,ENSHn = 1), then the right bottom shadow of font character is shifted to 1 dot right side. This shadow effect will continue until that (BSR) of adjacent character attribution become (BSR = 1).*

Bit No. & Name				Output (Polarity : Through)						Character color
19	18	17	16							
I	B	G	R	YM	YS	I	B	G	R	
0	0	0	0	0	0	0	0	0	0	Clear
0	0	0	1	0	1	0	0	0	1	Red
0	0	1	0	0	1	0	0	1	0	Green
0	0	1	1	0	1	0	0	1	1	Yellow
0	1	0	0	0	1	0	1	0	0	Blue
0	1	0	1	0	1	0	1	0	1	Magenta
0	1	1	0	0	1	0	1	1	0	Cyan
0	1	1	1	0	1	0	1	1	1	White
1	0	0	0	0	1	0	0	0	0	Black
1	0	0	1	0	1	0	0	0	1	Half-I,Red
1	0	1	0	0	1	0	0	1	0	Half-I,Green
1	0	1	1	0	1	0	0	1	1	Half-I, Yellow
1	1	0	0	0	1	0	1	0	0	Half-I,Blue
1	1	0	1	0	1	0	1	0	1	Half-I, Magenta
1	1	1	0	0	1	0	1	1	0	Half-I,Cyan
1	1	1	1	0	1	0	1	1	1	Half-I,White

Table 17-6 VRAM (bit19~16) function

Bit No. & Name				Output (Polarity : Through)						Back ground color
23	22	21	20							
I	B	G	R	YM	YS	I	B	G	R	
0	0	0	0	0	0	0	0	0	0	Clear
0	0	0	1	0	1	0	0	0	1	Red
0	0	1	0	0	1	0	0	1	0	Green
0	0	1	1	0	1	0	0	1	1	Yellow
0	1	0	0	0	1	0	1	0	0	Blue
0	1	0	1	0	1	0	1	0	1	Magenta
0	1	1	0	0	1	0	1	1	0	Cyan
0	1	1	1	0	1	0	1	1	1	White
1	0	0	0	1	0	0	0	0	0	Half blanking
1	0	0	1	0	1	1	0	0	1	Half-I,Green
1	0	1	0	0	1	1	0	1	0	Half-I, Yellow
1	0	1	1	0	1	1	0	1	1	Half-I,Blue
1	1	0	0	0	1	1	1	0	0	Half-I, Magenta
1	1	0	1	0	1	1	1	0	1	Half-I,Cyan
1	1	1	0	0	1	1	1	1	0	Half-I,White
1	1	1	1	0	1	0	1	1	1	Black

Table 17-7 VRAM (bit 23 ~ 20) function

Font ROM

The HMS81C43xx/GMS87C4060 OSD character size is fixed as 12dots (Horisontal) * 16dots (Vertical).

1. Each horisontal data (12dots) needs 2byte ROM.

2. One character is constructed with 16 horisontal data to vertically. As a result, one character needs 32bytes (2 * 16 bytes).

3. HMS81C4332/GMS87C4060 contains 256/512 characters.
Total Font ROM memory size is calulated as 16,384bytes (32bytes / character * 512 character)

4. Font ROM memory is located from 10000_H ~ $13FFF_H$, this memory can not be accessed by user program.

Character code	Address range	
	Upper 4bit	Lower 8bit
000_H	12000_H ~ $1200F_H$	10000_H ~ $1000F_H$
001_H	12010_H ~ $1201F_H$	10010_H ~ $1001F_H$
002_H	12020_H ~ $1202F_H$	10020_H ~ $1002F_H$
:	:	:
xyz_H	(12000H + xyz0H) ~ (12000H + xyzFH)	(10000H + xyz0H) ~ (10000H + xyzFH)
:	:	:
$1FD_H$	$13FD0_H$ ~ $13FDF_H$	$11FD0_H$ ~ $11FDF_H$
$1FE_H$	$13FE0_H$ ~ $13FEF_H$	$11FE0_H$ ~ $11FEF_H$
$1FF_H$	$13FF0_H$ ~ $13FFF_H$	$11FF0_H$ ~ $11FFF_H$

Table 17-8 Font ROM memory map

5. A character's address and dot position in font ROM is described in Figure 17-12 .

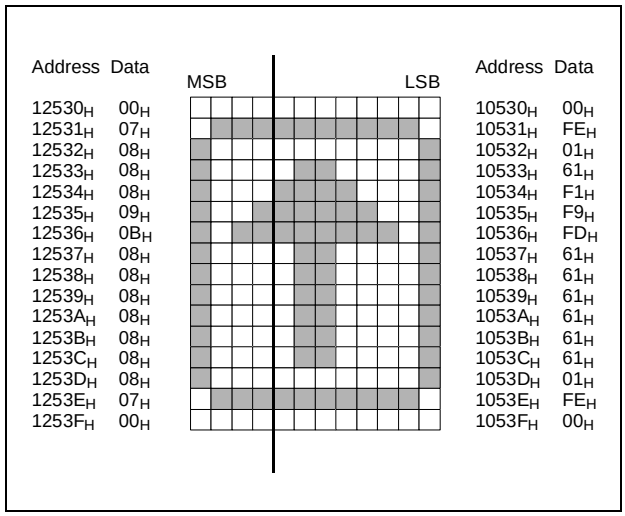


Figure 17-12 Example for a character (53_H)

Sprite RAM

The HMS81C43xx/GMS87C4060 contains a 32bytes (12dot * 16dot) sprite RAM.

1. In view point, sprite is similar to character font but it is not using font ROM.

2. You can selct color by dot unit.

3. Using above 1 and 2, you can make any of patterns what you want by software. For example, arrow cursor or some-

thing.

4. Sprite position is controlled by sprite position register SPVPOS[$0AE8_H$] and SPHPOS[$0AE9_H$].

5. Sprite RAM is located $0C00 \sim 0CF5_H$. One sprite RAM byte contains 2 dot's color data. See more details in Table 17-9 ~ Table 17-11.

Column number	Row number		
	MSB	~	LSB
00_H	$0C05_H$	~	$0C00_H$
01_H	$0C15_H$	~	$0C10_H$
02_H	$0C25_H$	~	$0C20_H$
:	:	~	:
$0n_H$ (n=0~F)	$0Cn5_H$	~	$0Cn0_H$
:	:	~	:
$0E_H$	$0C05_H$	~	$0C00_H$
$0F_H$	$0C05_H$	~	$0C00_H$

Table 17-9 Sprite RAM address map

	Odd dot color				Even dot color			
bit No.	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	-	B	G	R	-	B	G	R

Table 17-10 A sprite RAM's contents

B	G	R	Color
0	0	0	Clear
0	0	1	Red
0	1	0	Green
0	1	1	Yellow
1	0	0	Blue
1	0	1	Black
1	1	0	Cyan
1	1	1	White

Table 17-11 Sprite RAM Color Table

Test Font

HMS81C43xx use first OSD font as test purpose(see Fig17-13). When you design OSD characte font, you incert following font to Font ROM 00h. If you like to use this font originally, please contact us

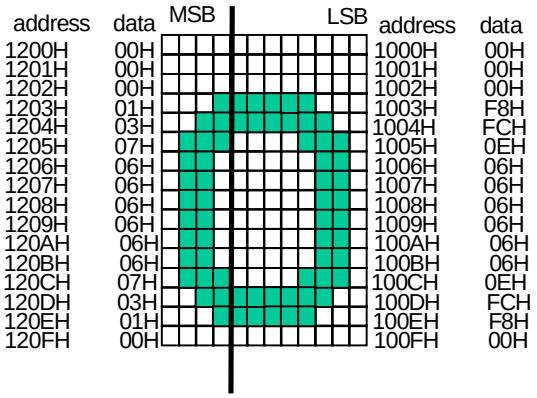


Figure 17-13 Test Font Pattern

18. I²C Bus Interface

The I²C Bus interface circuit is shown in Figure 18-1 .

The multi-master I²C Bus interface is a serial communications circuit, conforming to the Phlips I²C Bus data transfer format. This interface, offering both arbitration lost detection and a synchronous functions, is useful for the multi-master serial communications.

This multi-master I²C Bus interface circuit consists of the I²C address register, the I²C data shift register, the I²C clock control register, the I²C control register, the I²C status register and other control circuits.

The more details about registers are shown Figure 18-2 ~ Figure 18-6 .

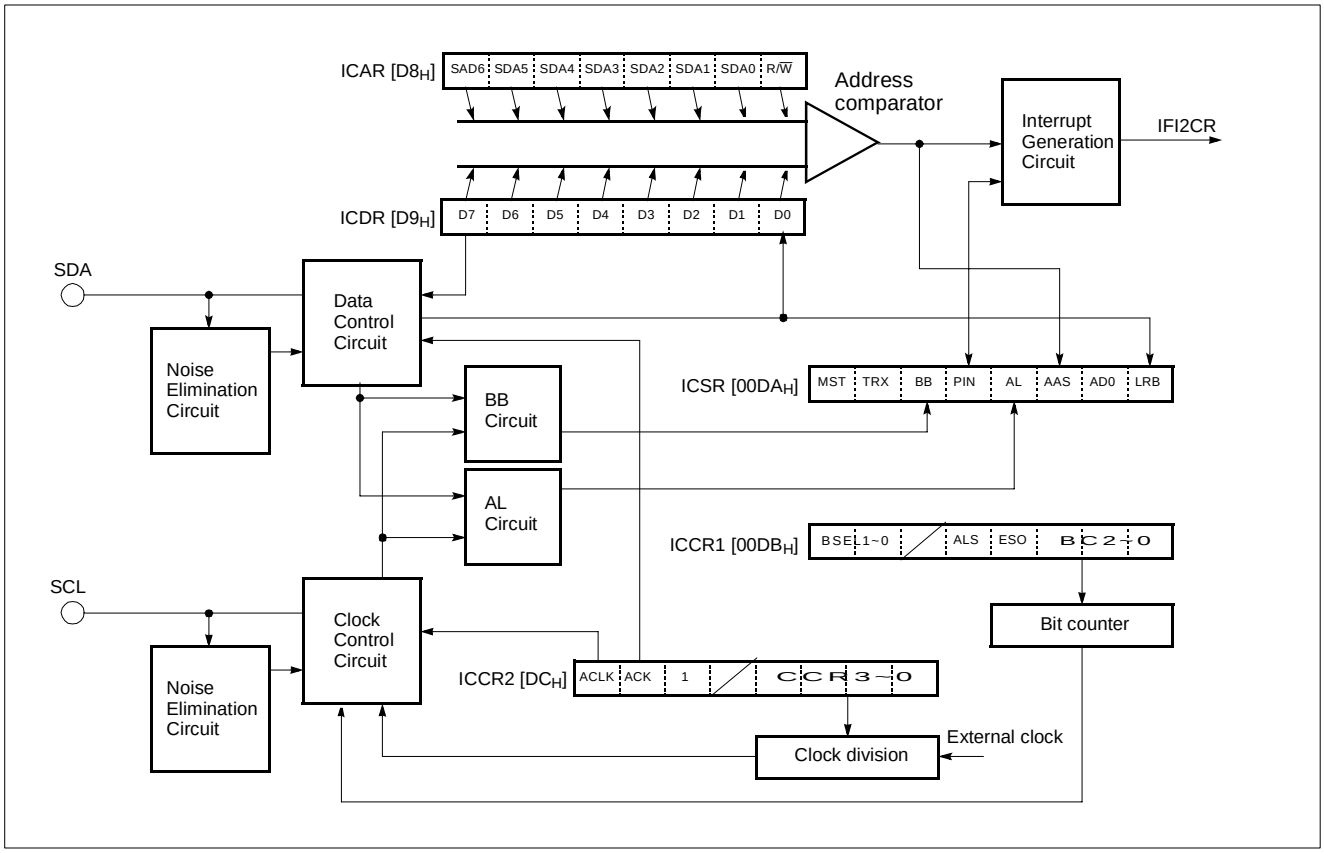


Figure 18-1 Block Diagram of multi-master I²C circuit

Control

The HMS81C43xx/GMS87C4060 contains two I²C Bus interface modules. It supports multi-master function, so it contains arbitration lost detection, synchronization function,etc.

ITEM	Function
Format	Philips I²C standard 7bit addressing format
Communication mode	Master transmitter Master receiver Slave transmitter Slave receiver
SCL clock frequency	66.6KHz ~ 500KHz (f_{ex}=12MHz) 44.4KHz ~ 333.3KHz (f_{ex}=8MHz)

I²C address register

It contains slave address (7bit) which is used during slave mode and Read/$\overline{\text{Write}}$ bit.

Bit 7 ~ 0 : Slave address 6~0

Note: *Bit 7~0 (SAD6~0) store slave address. The address data transmitted from the master is compared with the contents of these bits.*

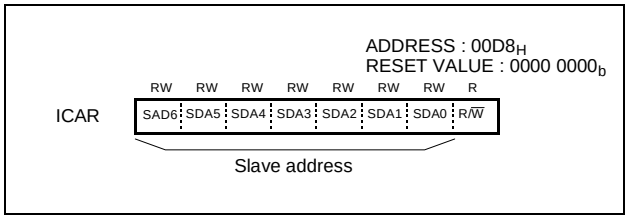


Figure 18-2 I²C address Register

I²C data shift register [ICDR]

The I²C data shift register is an 8bit shift register to store received data and write transmit data.

When transmit data is written into this register, it is transfered to the outside from bit7 in synchronization with the SCL clock, and each time one-bit data is output, the data of this register are shifted one bit to the left. When data is received, it is input to this register from bit0 in synchronization with the SCL clock, and each time one-bit data is input, the data of this register are shifted one bit to the left.

The I²C data shift register is in a write enable status only when the ESO bit of the I²C control register (address $00DC_H$) is "1". The bit counter is reset by a write instruction to the I²C data shift register. Reading data from the I²C data shift register is always enabled regardless of the ESO bit value.

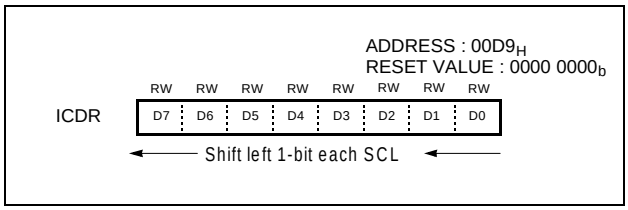


Figure 18-3 Data shift register

I²C status register

The I²C status register controls the I²C Bus interface status. The low-order 4bits are read only bits and the high-order 4bits can be read out and written to.

The more details about its bits are shown Table 18-1.

Bit No.	Name	Function
7 6	MST TRX	00: Slave / Receiver mode 01: Slave / Transmitter mode 10: Master / Receiver mode 11: Master / Transmitter mode **MST is cleared when** - After reset. - After the arbitration lost is occured and 1 byte data transmission is finished. - After stop condition is detected. - When start condition is disabled by start condition duplication preventation function. **TRX is cleared when** - After reset. - When arbitration lost or stop condition is occured . - When MST is '0', and start condition or ACK non-return mode is detected.
5	BB	BB(Bus busy)bit is 1 during bus is busy. This bit can be written by S/W. its value is '1' by start condition, and cleared by stop condition.
4	PIN	PIN(Pending Interrupt Not)bit is interrupt request bit. If I²C interrupt request is issued, its value is 0. **PIN is cleared when** - After 1 byte trasmission / receive is finished. **PIN is set when** - After reset. - After write instruction is excuted into I²C data shift register ICDR. - When PIN bit low, the output of SCL is pulled down, So if you want to release SCL, you must perform write instruction CDR.
3	AL	Arbitration lost detection flag. If arbitration lost is detected, AL=1, or 0.
2	AAS	Slave address comparison flag. It shows compared result with received address data and I²C address register (ICAR). It is 1, when two of data is same.

Table 18-1 Bit function

Bit No.	Name	Function
1	AD0	General call detection flag. If general call is detected, AD0=1, or not 0. * General call : If received address is all '0' . it is called general call.
0	LRB	Last received bit. it is used for receive confirmation. If ACK is returned, LRB=0, or not 1.

Table 18-1 Bit function

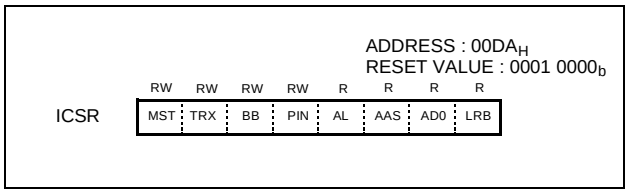


Figure 18-4 I²C status Register

I²C control register 1

It controls communication data format.

Bit No.	Name	Function
7 6	BSEL1 BSEL0	I²C connection control. 00: No connection 01: SCL1, SDA1 10: SCL2, SDA2 11: SCL1, SDA1, SCL2, SDA2
4	ALS	Data format selection. 0: Addressing format 1: Free data format
3	ESO	I²C Bus interface use enable flag 0: Disabled 1: Enabled
2	BC2	Bit counter. 000_b: 8bit 001_b~111_b: 1~7bit
1	BC1	
0	BC0	

Table 18-2 Bit function

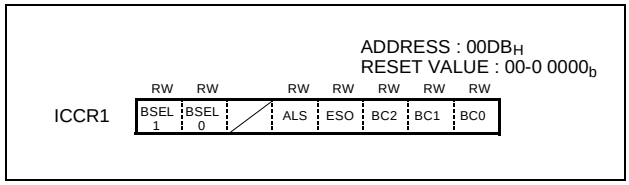


Figure 18-5 I²C control Register 1

I²C control register 2

It controls SCL mode, SCL frequency, etc.

It contains 8bit data to transmit to external device when tr-asmitter mode, or received 8bit data from external device when receive mode.

Bit No.	Name	Function
7	ACLK	Select acknowledge clock (ACK) mode. 0: No acknowledge clock mode. acknowledge clock is not generated after data was transmismitted. 1: acknowledge clock mode. acknowledge clock is generated after data was transmismitted.
6	ACK	If acknowledge clock is returned, this bit is 0. Or not 1.
5	1 (fixed)	Not used.

Table 18-3 Bit function

Bit No.	Name	Function		
		SCL Frequency selection SCL frequency = f_{ex} / (12 * CCR)		
		Value	f_{ex} = 12MHz	f_{ex} = 8MHz
3 2 1 0	CCR3 CCR2 CCR1 CCR0	0000	Not allowed	Not allowed
		0001	Not allowed	Not allowed
		0010	500.0KHz	333.3KHz
		0011	333.3KHz	222.2KHz
		0100	250.0KHz	166.6KHz
		0101	200.0KHz	133.3KHz
		0110	166.6KHz	111.1KHz
		0111	142.9KHz	95.2KHz
		1000	125.0KHz	83.3KHz
		1001	111.1KHz	74.1KHz
		1010	100.0KHz	66.6KHz
		1011	90.0KHz	60.6KHz
		1100	83.3KHz	55.5KHz
		1101	76.4KHz	51.3KHz
		1110	71.4KHz	47.6KHz
		1111	66.6KHz	44.4KHz

Table 18-3 Bit function

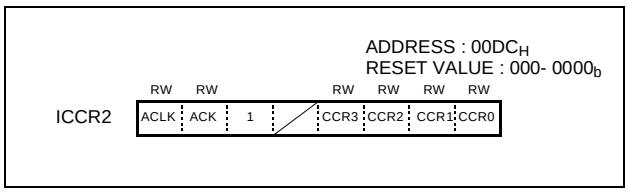


Figure 18-6 I²C control Register 2

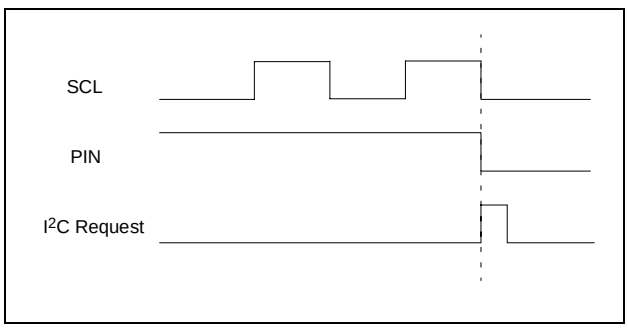


Figure 18-7 Interrupt request signal generation timing

START condition generation

When the ESO bit of the I²C control register ($00DB_H$) is "1", writing to the I²C status register will generate START condition. Refer to Figure 18-8 for the START condition generation timing diagram.

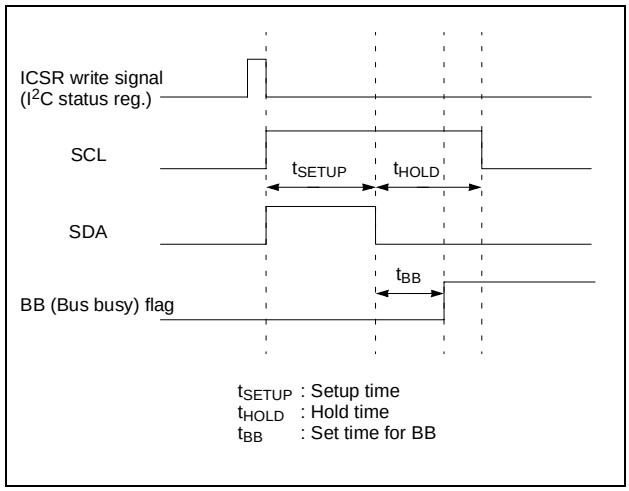


Figure 18-8 START condition generation timing

RESTART condition generation

RESTART condition's setting sequence is as followings.

1. Write 020_H to I²C status register (ICSR, $00DA_H$)

2. Write slave address to I²C data shift register (ICDR, $00D9_H$)

3. Write $0F0_H$ to I²C status register (ICSR, $00DA_H$)

STOP condition generation

Writing 'C0h' to ICSR will generate a stop condition,

when ESO(ICCR bit3) is '1'

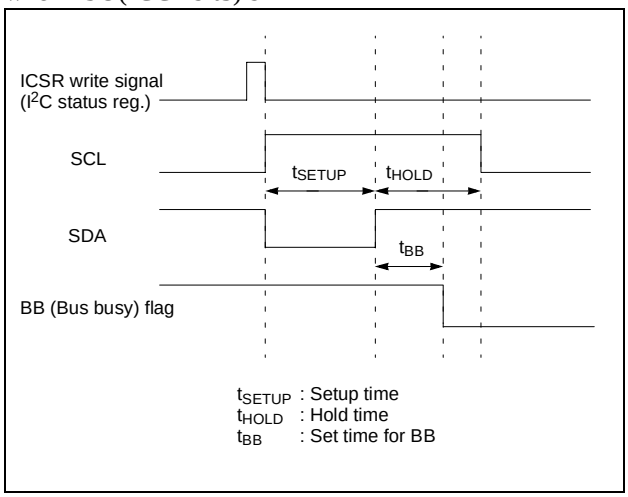


Figure 18-9 STOP condition generating timing diagram

START / STOP condition generation time is shown Table 18-4.

ITEM	Timing SPEC.
Setup time (t_{SETUP})	3.3uS (n=20cycles)
Hold time (t_{HOLD})	3.3uS (n=20cycles)
Set/Reset time for BB flag (t_{BB})	3.0uS (n=18cycles)

Table 18-4 Example time (f_{ex}=12MHz)

START / STOP condition detect

START / STOP condition is detected when Table 18-4 is satisfied.

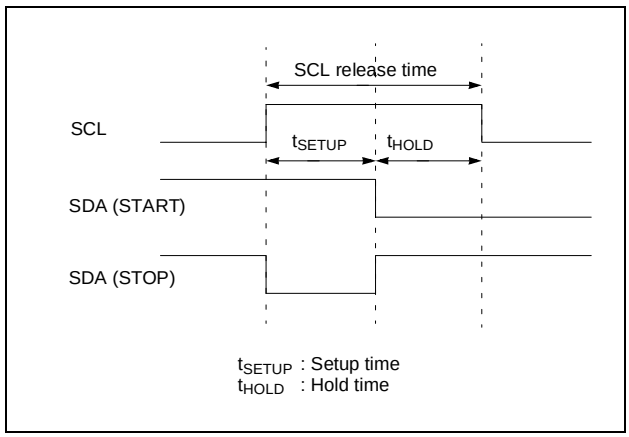


Figure 18-10 START / STOP condition detection timing

START / STOP detection time is showed Table 18-5.

ITEM	Timing SPEC.
SCL release time	> 2.0uS (n=12cycles)
Setup time	> 1.0uS (n=6cycles)
Hold time	> 1.0uS (n=6cycles)

Table 18-5 Example time (f_{ex}=12MHz)

Address data communication

The first transmitted data from master is compared with I^2C address register (ICAR, $00D8_H$). At this time R/$\overline{W}$ is not compared but it determines next data operation. i.e, transmitting or receiving data

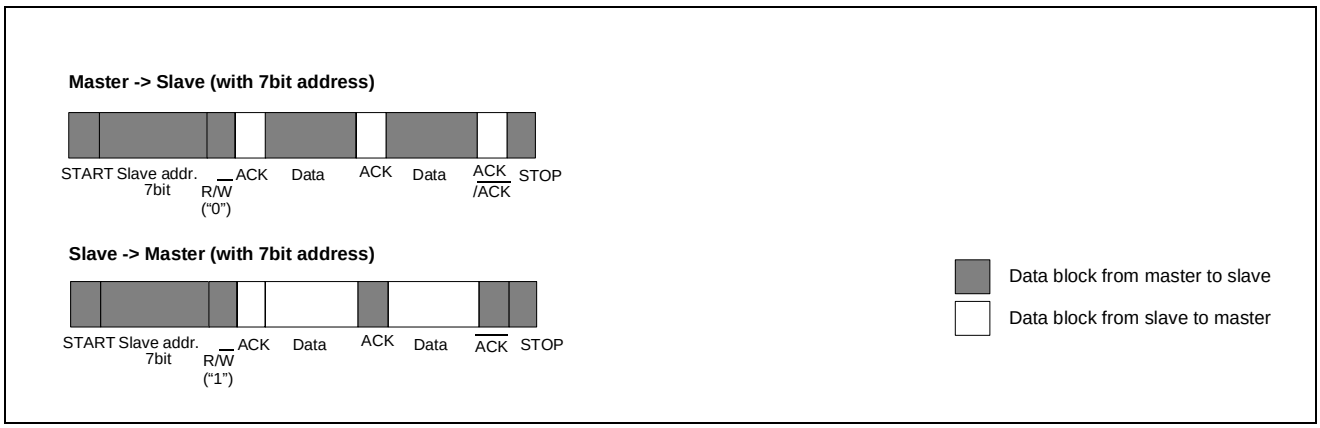


Figure 18-11 Address data communication format

19. INTERRUPTS

The HMS81C43xx/GMS87C4060 interrupt circuits consist of Interrupt enable register (IENH, IENL), Interrupt request flags of IRQH, IRQL, Priority circuit and Master enable flag ("I" flag of PSW). 16 interrupt sources are provided. The configuration of interrupt circuit is shown in Figure 19-2 .

Below table shows the Interrupt priority

Reset/Interrupt	Symbol	Priority
Hardware Reset	RESET	-
External Interrupt 0	INT0	1
OSD Interrupt	OSD	2
External Interrupt 1	INT1	3
External Interrupt 2	INT2	4
Timer/Counter 0	Timer 0	5
Timer/Counter 2	Timer 2	6
1 Frame Interrupt	1Frame	7
VSync Interrupt	VSync	8
Timer/Counter 1	Timer 1	9
Timer/Counter 3	Timer 3	10
Interrupt interval measure	INTV(INT3/4)	11
Watchdog Timer	WDT	12
Basic Interval Timer	BIT	13
Serial I/O Interrupt	SIO	14
I^2C Interrupt	I2C	15

The External Interrupts can each be transition-activated (1-to-0 or 0-to-1 transition).
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/Counter Interrupts are generated by TnIF(n=0~3), which is set by a match in their respective timer/counter register.

The Basic Interval Timer Interrupt is generated by BITIF which are set by a overflow in the timer register.

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 Mode Register

It controls interrupt priority. It takes only one specified interrupt.

Of course, interrupt's priority is fixed by H/W, but sometimes user want to get specified interrupt even if higher priority interrupt was occured. Higher priority interrupt is processed the next time.

It contains 2bit data to enable priority selection and 4bit data to select specified interrupt.

Bit No.	Name	Value	Function
5,4	IM1~0	00 01 1X	Mode 0: H/W priority Mode 1: S/W priority Interrupt is disabled, even if IE is set.
3~0	IP3~0	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	INT0 OSD INT1 INT2 Timer 0 Timer 2 1Frame VSync Timer 1 Timer 3 INTV(INT3/4) WDT BIT SIO I2C Not used

Table 19-1 Bit function

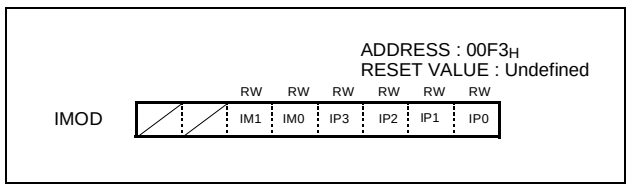


Figure 19-1 Interrupt Mode Register

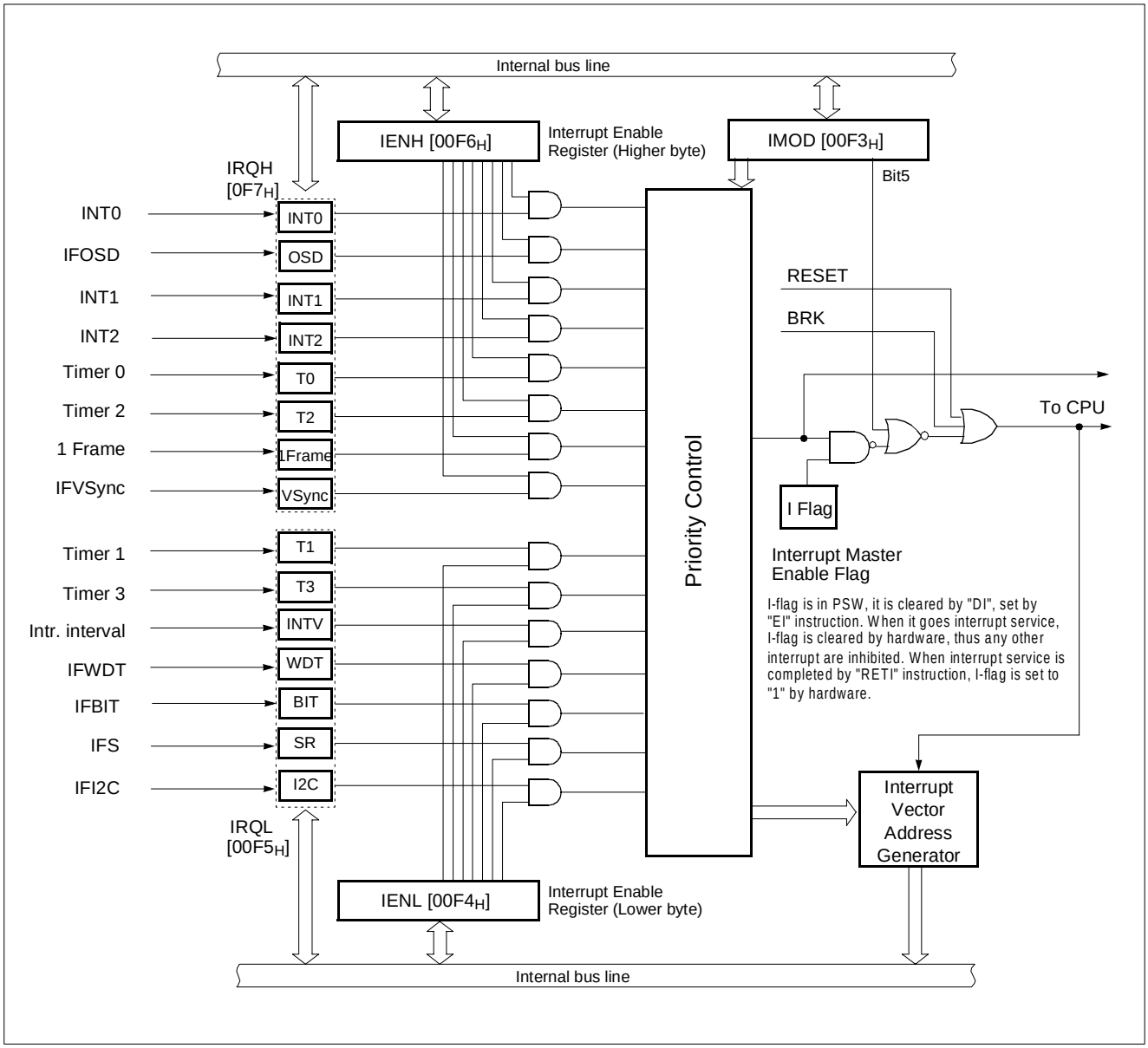


Figure 19-2 Block Diagram of Interrupt

Interrupt enable registers are shown in Figure 19-4 . 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.

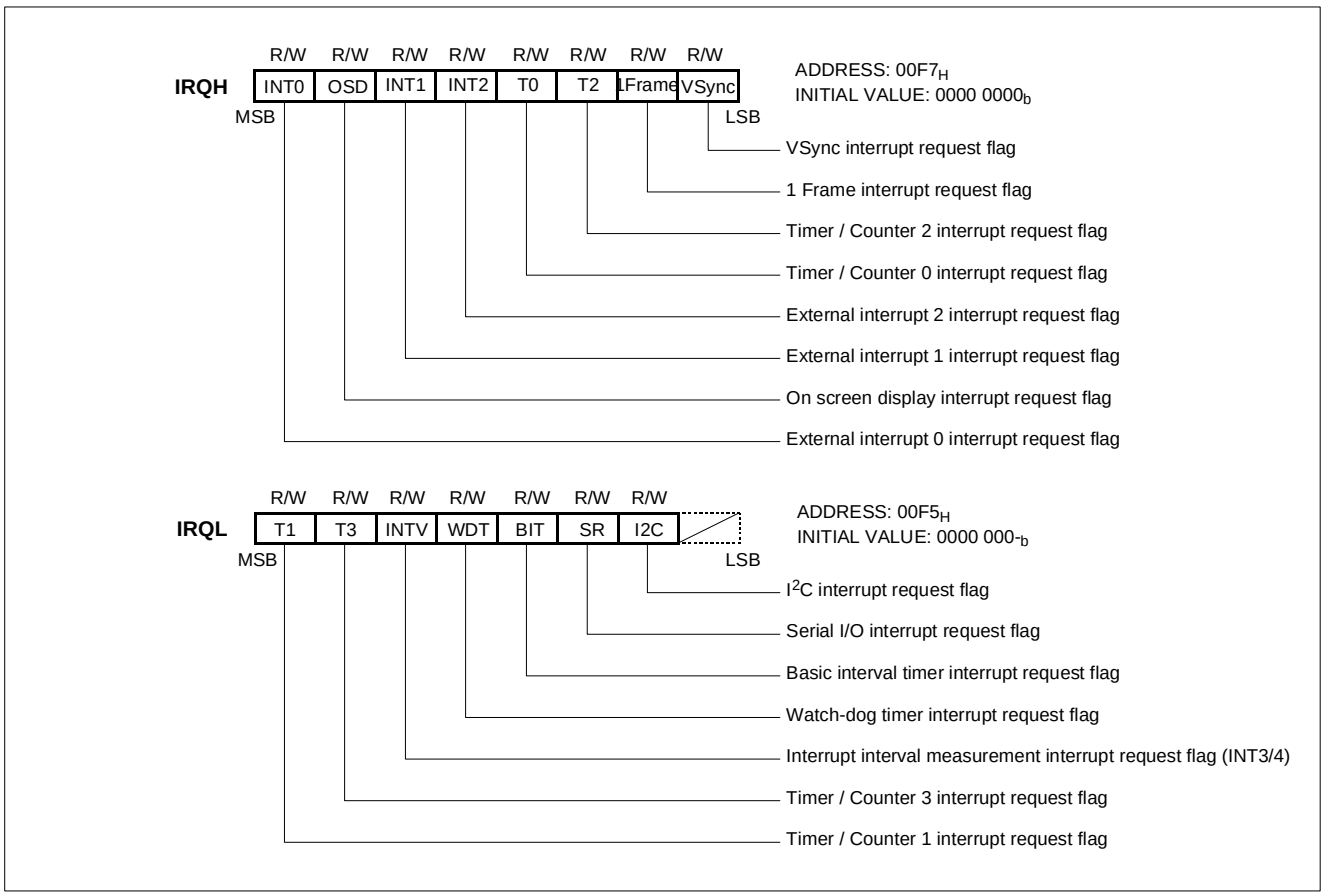


Figure 19-3 Interrupt Request Flags

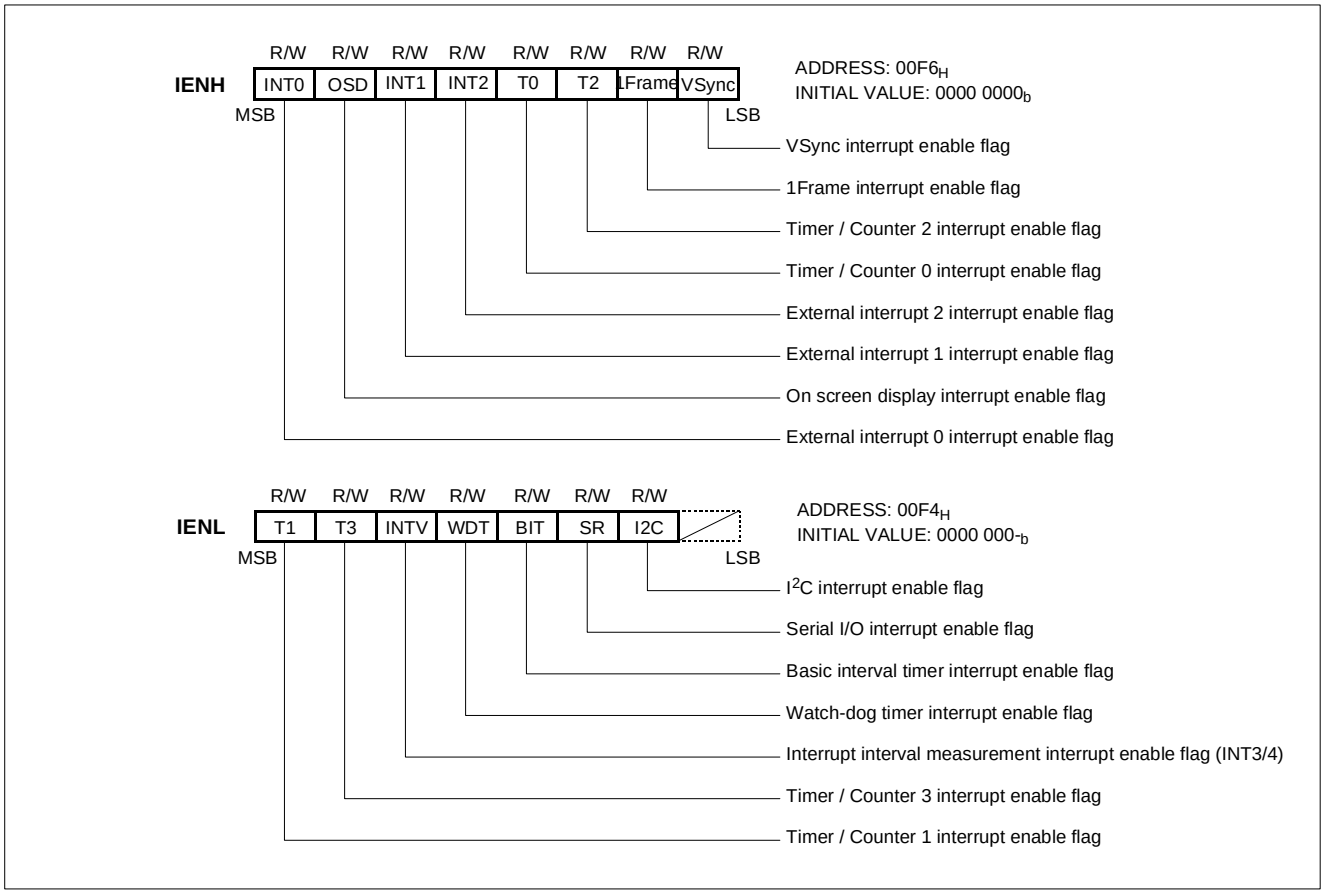


Figure 19-4 Interrupt Enable Flags

19.1 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_{ex} (2 µs at f_{MAIN}=4MHz) after the completion of the current instruction execution. The interrupt service task terminates 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 decrements 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.

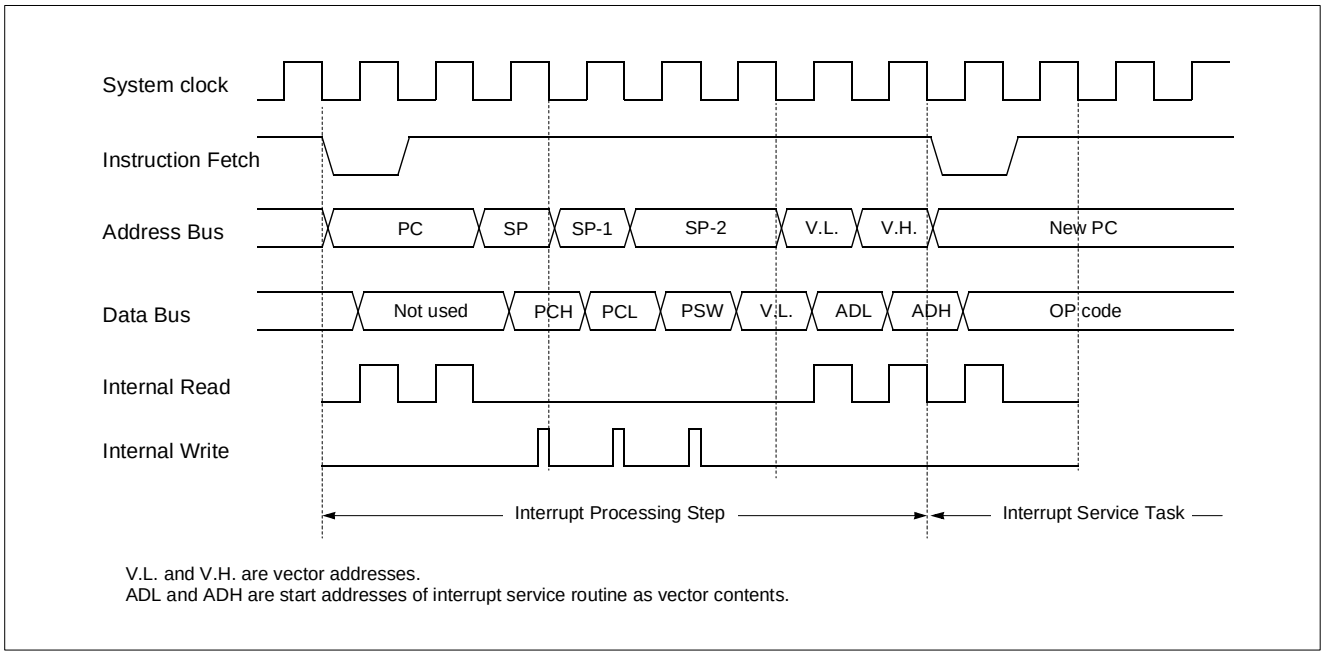
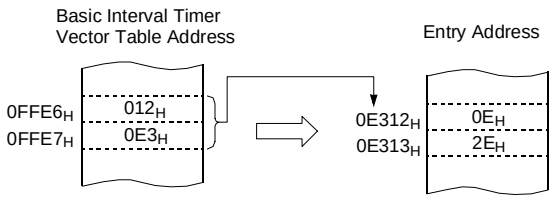


Figure 19-5 Interrupt Service routine Entering Timing



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

A maskable interrupt is not accepted until the I-flag is set to "1" even if a maskable interrupt of higher priority than that of the current interrupt being serviced.

When nested interrupt service is necessary, the I-flag is set to "1" 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

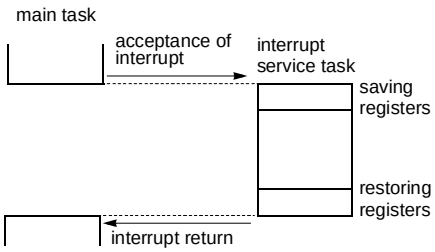
During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using 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.
        LDA     DPGR    ;SAVE DPGR
                        ; Direct page
                        ; accessable reg.
        PUSH    A       ;
        :
interrupt processing
        :
        POP     A
        STA     DPGR    ;RESTORE DPGR
        POP     X       ;RESTORE X REG.
        POP     A       ;RESTORE ACC.
        RETI            ;RETURN
```

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



19.2 BRK Interrupt

Software interrupt can be invoked by BRK instruction, which is 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 19-6 .

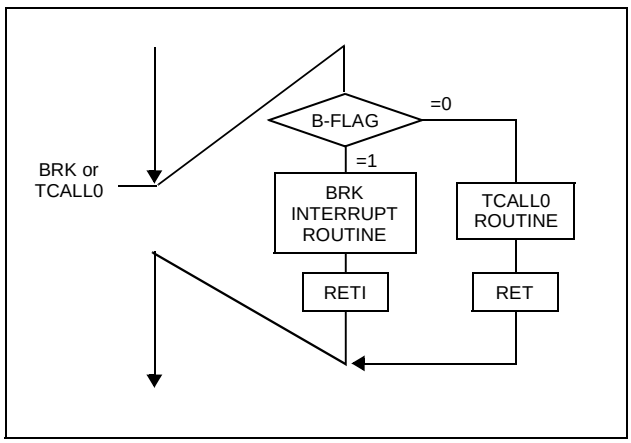


Figure 19-6 Execution of BRK/TCALL0

19.3 Multi Interrupt

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

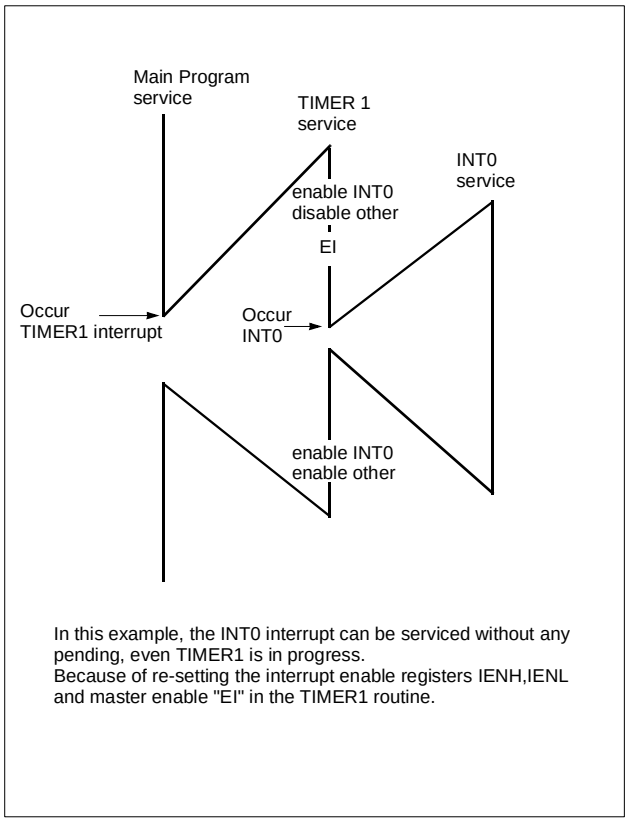


Figure 19-7 Execution of Multi Interrupt

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 set I-flag in interrupt routine, some further interrupt can be serviced even if certain interrupt is in progress.

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,#FFH     ;Enable all interrupts
         LDM    IENL,#FEH
         POP    Y
         POP    X
         POP    A
         RETI
```

19.4 External Interrupt

The external interrupt on INT0, INT1... pins are edge triggered depending the edge selection register.

Refer to “6. PORT STRUCTURES” on page 10.

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

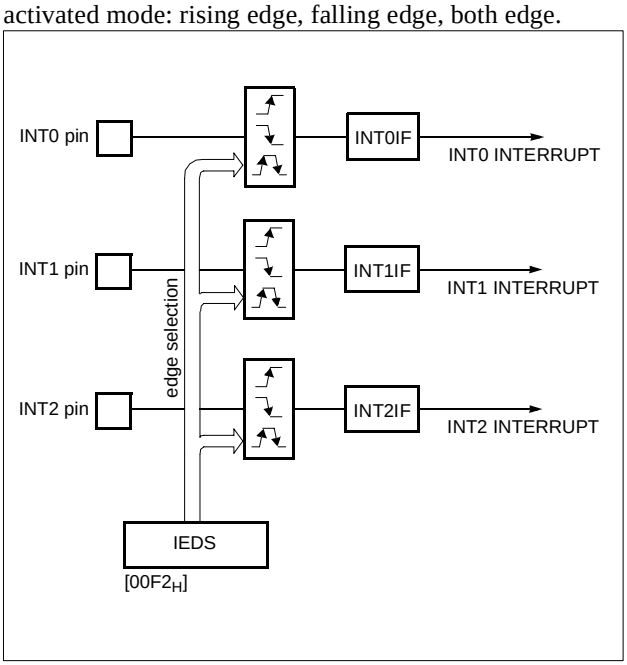


Figure 19-8 External Interrupt Block Diagram

INT0, INT1 and INT2 are multiplexed with general I/O ports. To use external interrupt pin, the bit of port function register FUNC1 should be set to "1" correspondingly.

Response Time

The INT0, INT1 and INT2 edge are latched into INT0IF, INT1IF and INT2IF 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. For example, the DIV instruction takes twelve machine cycles. Thus, a minimum of twelve complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine

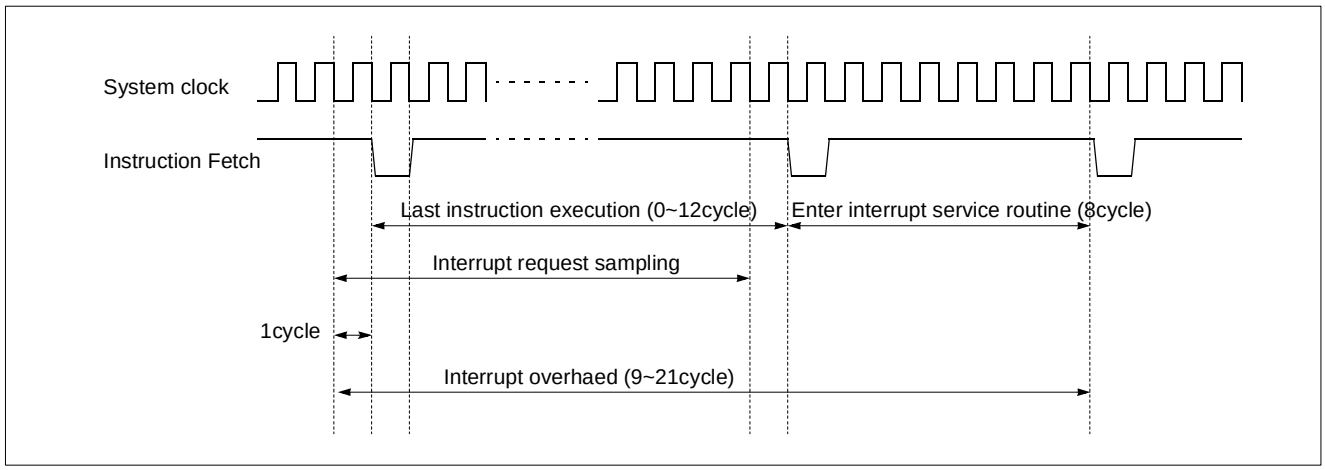


Figure 19-9 Interrupt Response Timing Diagram (Interrupt overhead)

20. WATCHDOG TIMER

The watchdog timer rapidly detects the CPU malfunction such as endless looping caused by noise or the like, and resumes the CPU to the normal state.
The watchdog timer signal for detecting malfunction can be selected either a reset CPU or a interrupt request.

When the watchdog timer is not being used for malfunction detection, it can be used as a timer to generate an interrupt at fixed intervals.

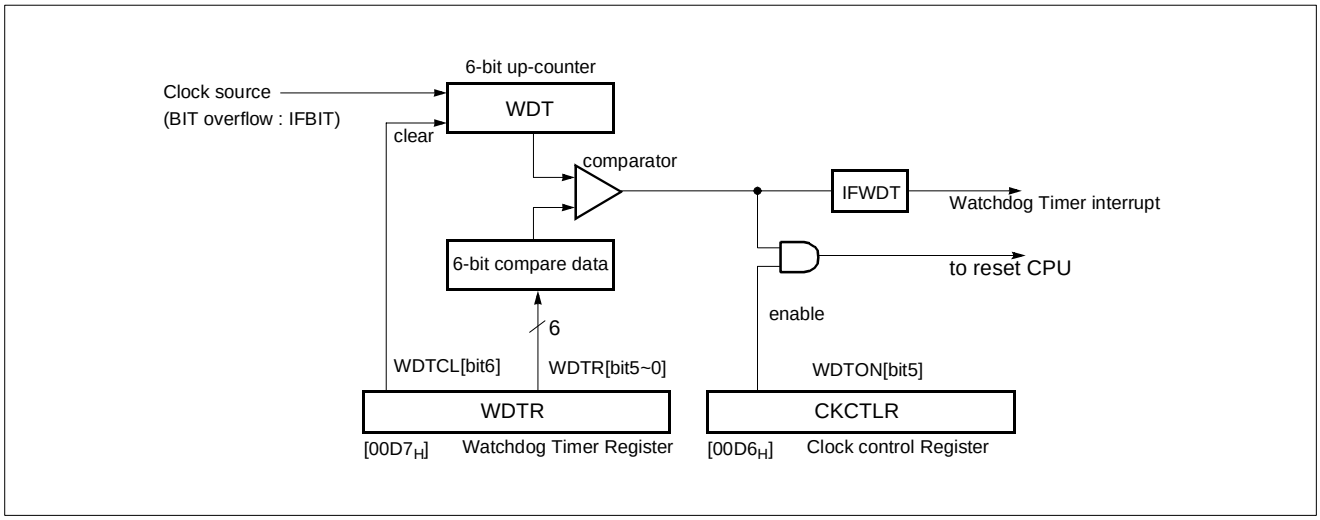


Figure 20-1 Block Diagram of Watchdog Timer

Watchdog Timer Control

Figure 20-2 shows the watchdog timer control register. The watchdog timer is automatically disabled after reset.

The CPU malfunction is detected as setting the detection time, selecting output, and clearing the binary counter. Repeatedly clearing the binary counter within the setting detection time.

If the malfunction occurs for any cause, the watchdog timer output will become active at the rising overflow from the binary counters unless the binary counter are cleared. At this time, when WDTON=1 a reset is generated, which drives the $\overline{\text{RESET}}$ pin low to reset the internal hardware. When WDTON=0, a watchdog timer interrupt (IFWDT) is generated.

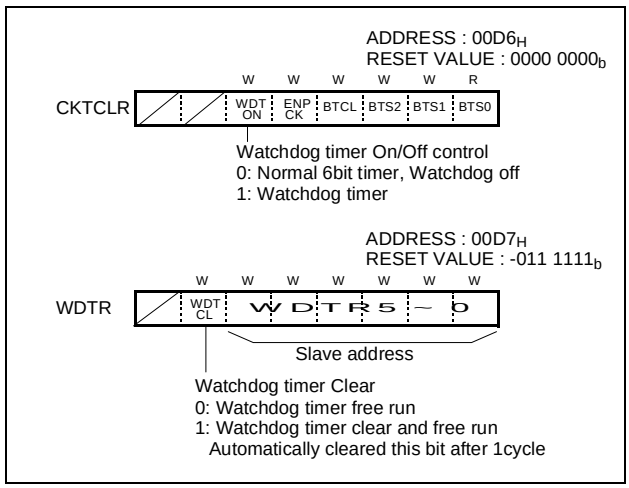


Figure 20-2 Watchdog timer register

Example: Sets the watchdog timer detection time

```
        LDM   WDTR,#01??????b      ;Clear Counter and set value(??????b)
                                   ;You have to set WDTR first, for prevent unpredictable interrupt
                                   ;when you set WDTON bit.
        LDM   CKCTLR,#00111???b    ;Select clock source(???b) and WDTON=1
Within WDT
detection time
        LDM   WDTR,#01??????b      ;Clear counter
        :
        :
        :
Within WDT
detection time
        LDM   WDTR,#01??????b      ;Clear counter
        :
        :
        :
        LDM   WDTR,#01??????b      ;Clear counter
```

Enable and Disable Watchdog

Watchdog timer is enabled by setting WDTON (bit 5 in CKTCLR) to "1". WDTON is initialized to "0" during reset, WDTON should be set to "1" to operate after reset is released.

Example: Enables watchdog timer reset

```
        :
LDM     CKTCLR,#001?????b ;WDTON←1
        :
        :
```

The watchdog timer is disabled by clearing bit 5 (WDTON) of CKTCLR.

Watchdog Timer Interrupt

The watchdog timer can also be used as a simple 6-bit timer by clearing bit 5 (WDTON) of CKTCLR. The interval of watchdog timer interrupt is decided by Basic Interval Timer.

Interval equation is shown as below.

$$T = WDTR \times Interval\ of\ BIT$$

The stack pointer (SP) should be initialized before using the watchdog timer output as an interrupt source.

Example: 6-bit timer interrupt setting up.

```
LDX     #03FH
TXSP                    ;SP ← 3F
LDM     CKTCLR,#000?????b ;WDTON←0
LDM     WDTR,#01??????b   ;WDTCL←0
:
:
```

Refer table and see BIT timer ().

CKCTLR BTS2~0	BIT input clock	Watchdog timer input clock	IFWDT cycle
000_b	PS4 (2uS)	512uS	32,256uS
001_b	PS5 (4uS)	1,024uS	64,512uS
010_b	PS6 (8uS)	2,048uS	129,024uS
011_b	PS7 (16uS)	4,096uS	258,048uS
100_b	PS8 (32uS)	8,192uS	516,096uS
101_b	PS9 (64uS)	16,384uS	1,032,192uS
110_b	PS10 (128uS)	32,768uS	2,064,384uS
111_b	PS11 (256uS)	65,536uS	4,128,768uS

Table 20-1 Watchdog timer MAX. cycle (Ex:f_{ex}=8MHz)

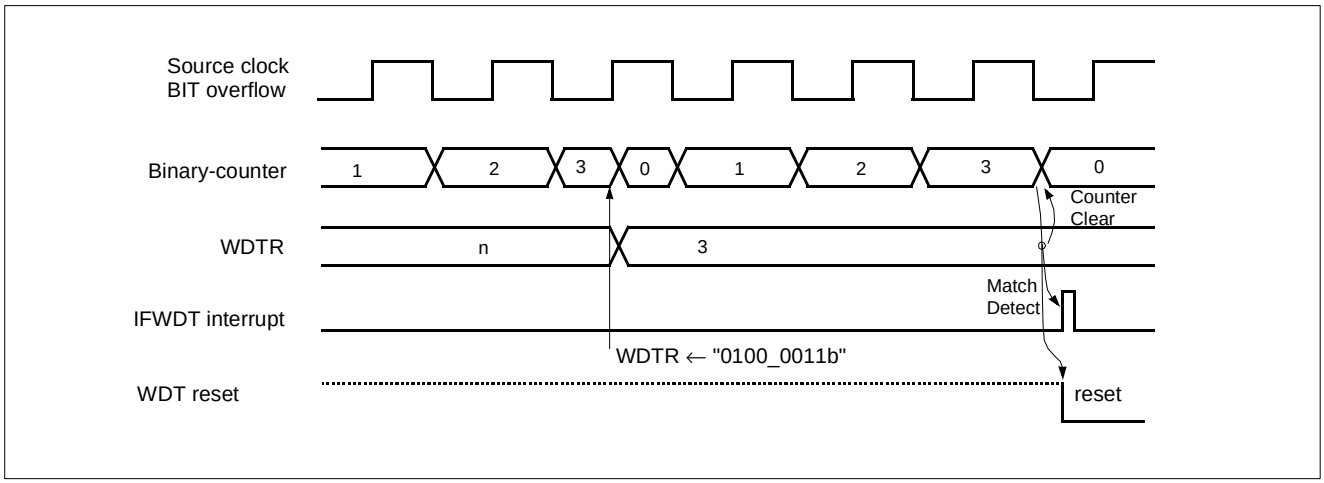


Figure 20-3 Watchdog timer Timing

Minimizing Current Consumption

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 unfirmed voltage level 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. See Figure 20-4 .

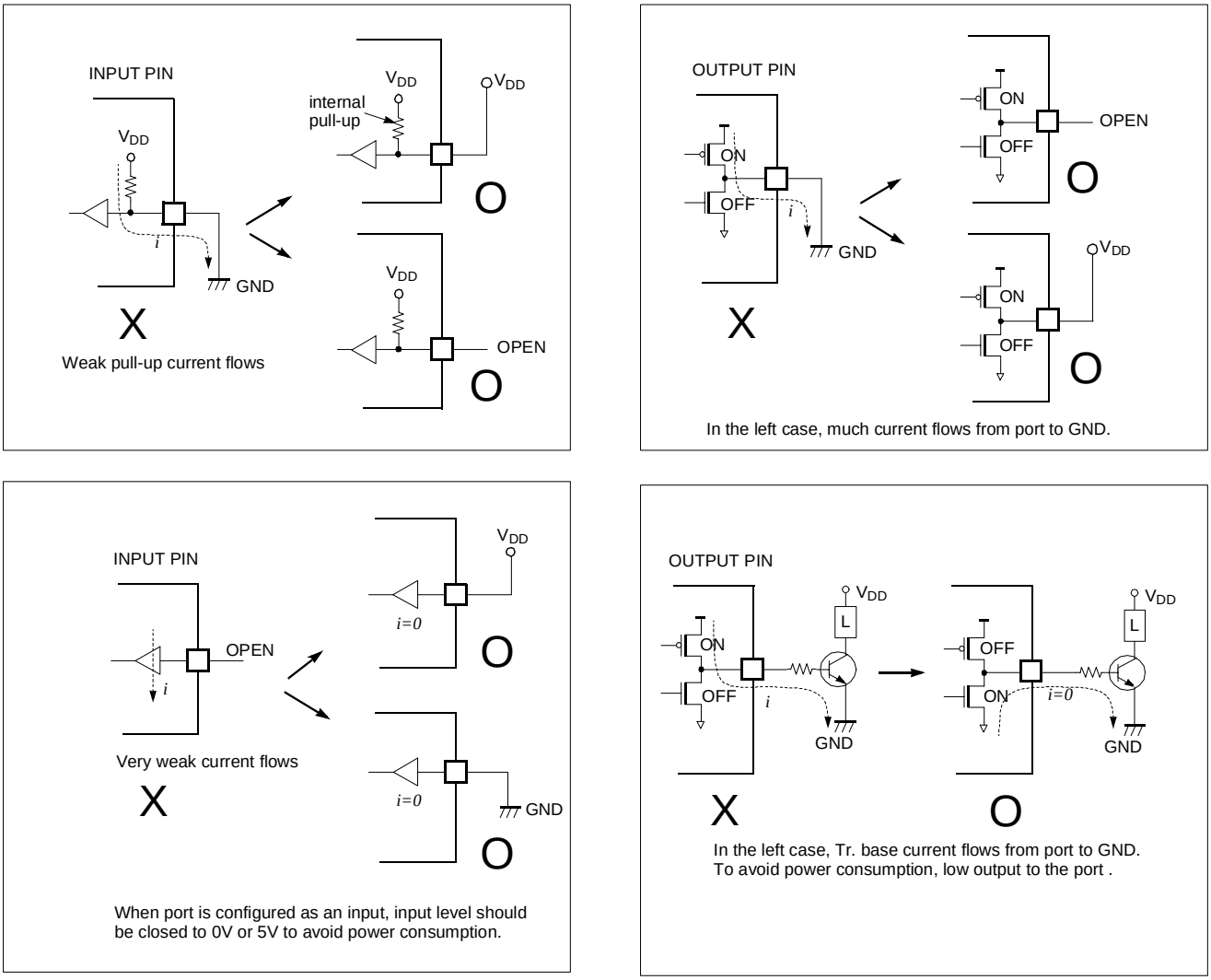


Figure 20-4 Application example of Port under Power Consumption

21. OSCILLATOR CIRCUIT

The HMS81C43xx/GMS87C4060 has two oscillation circuits internally. X_{IN} and X_{OUT} are input and output for main frequency and OSC1 and OSC2 are input and output for OSD(On Screen display) frequency, respectively, of a inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 21-1 .

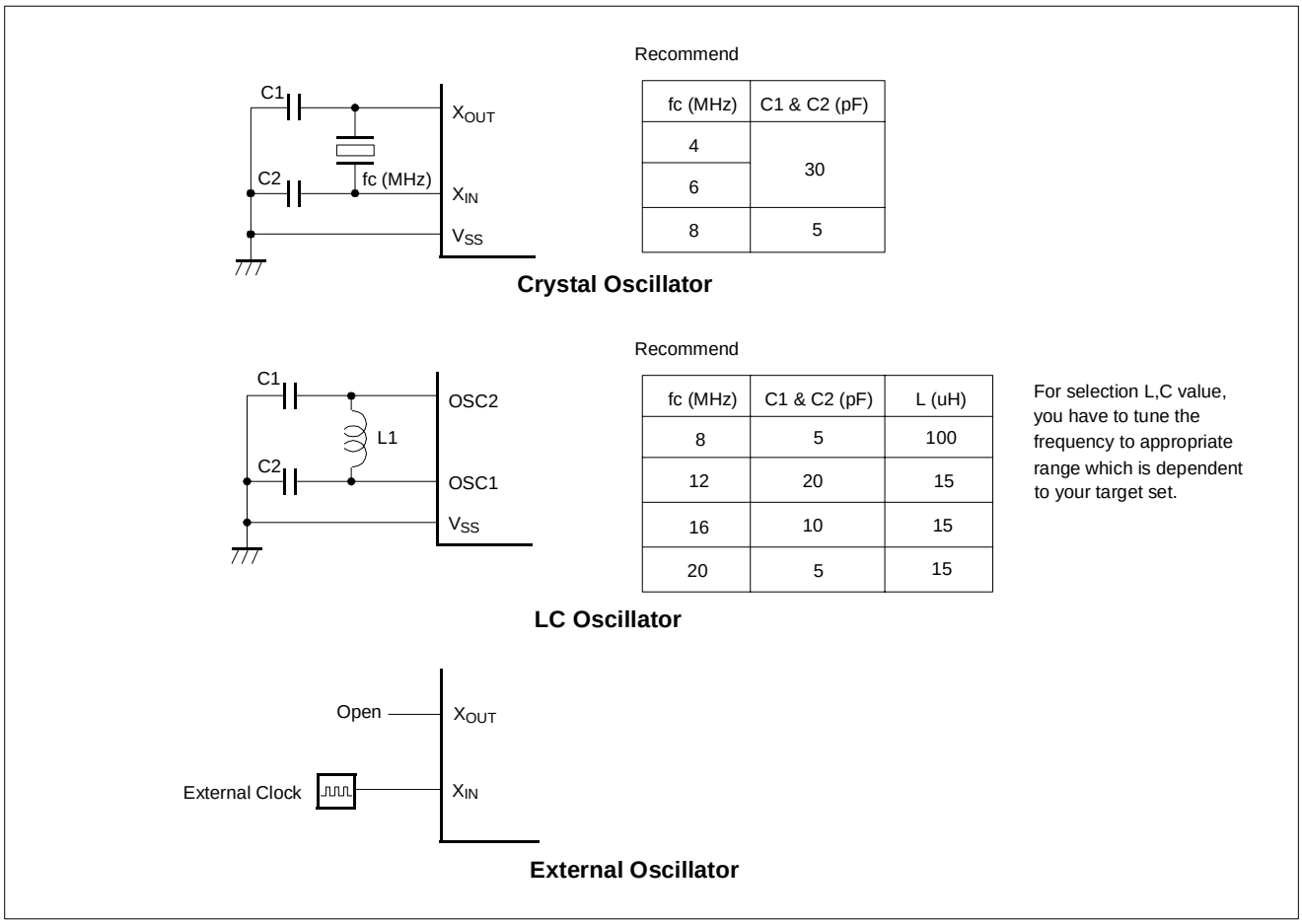


Figure 21-1 Oscillation Circuit

Oscillation components have their own characteristics, so user should consult the component manufacturers for appropriate values of external components.

In addition, see Figure 21-2 for the layout of the crystal.

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

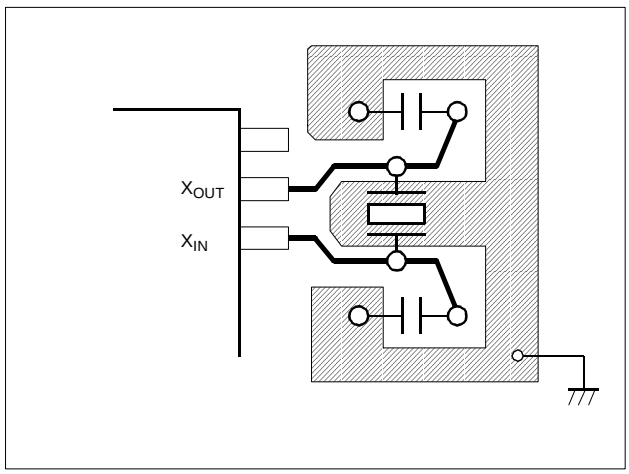


Figure 21-2 Layout example of Oscillator PCB circuit

22. RESET

The HMS81C43xx/GMS87C4060 have two types of reset generation procedures; one is an external reset input, other is a watch-dog timer reset. Table 22-1 shows on-chip hardware initialization by reset action.

On-chip Hardware		Initial Value
Program counter	PC	($FFFF_H$) - ($FFFE_H$)
RAM page register	DPGR	00_H
G-flag of PSW	G	0

On-chip Hardware	Initial Value
Peripheral clock	Off
Watchdog timer	Disable
Control registers	Refer to Table 8-1 on page 22

Table 22-1 Initializing Internal Status by Reset Action

22.1 External Reset Input

The reset input is the $\overline{RESET}$ 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, within the operating voltage range and oscillation stable, a reset is applied and the internal state is initialized. After reset, 64ms (at 4 MHz) add with 7 oscillator periods are required to start execution as shown in Figure 22-2 .

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

When the $\overline{RESET}$ pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses $FFFE_H$ - $FFFF_H$.

A connecting for simple power-on-reset is shown in Figure 22-1 .

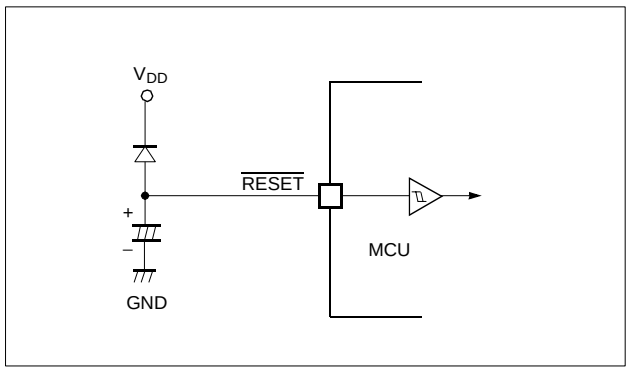


Figure 22-1 Simple Power-on-Reset Circuit

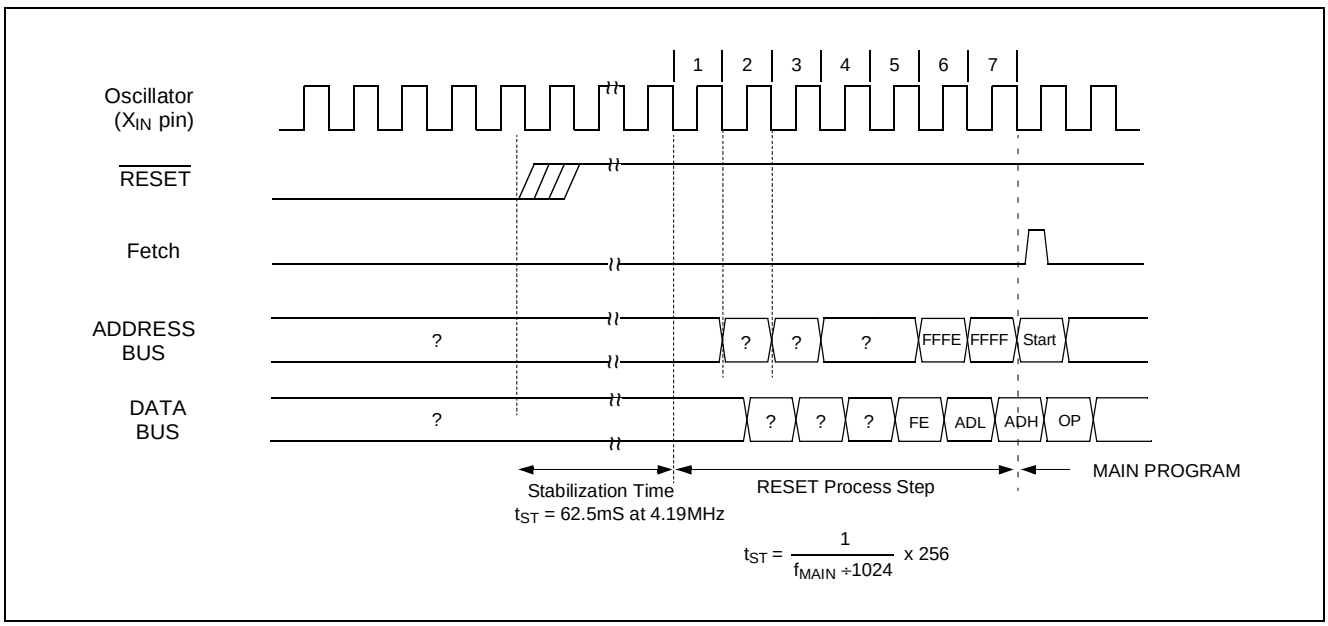


Figure 22-2 Timing Diagram after RESET

22.2 Watchdog Timer Reset

Refer to “20. WATCHDOG TIMER” on page 77.

23. OTP Programming

23.1 GMS87C4060 OTP Programming

You can burn out GMS87C4060 OTP through the general Gang programmer using Intel 27010/C010 mode. In Develeopment tool package auxiliary, GMS87C4060-to-27010/C010 conversion socket is included. GMS87C4060 have two ROM memory areas. One is Program ROM memory and the other is Font ROM memory. Program ROM area is from 1000h to FFFFh Font ROM area is from 10000h to 13FFFh. When you acquire new OTP, actually, the OTP is not fully blank. The OPT have six test pattern in the OSD Font ROM memory(see figure23-1). The test pattern are written at 11FA0h ~ 11FFFh and 13FA0h ~ 13FFFh.

Note: *DO NOT write any data in this area(11FA0h ~ 11FFFh, 13FA0h ~ 13FFFh)*

Blank Check

If you run blank check function of ROM writer, ROM writer inform blank error because of test pattern. To avoid this situation, you must run the blank check function sepereteley. For example, check OTP address rage of 1000h ~ 11F9Fh at first. And then check OPT address range of 12000h ~ 13F9Fh. If you have ROM writer without partial blank check function, please do not run blank check function.

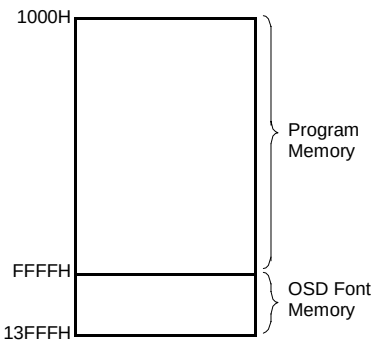


Figure 23-1 GMS87C4060 OTP Memory Map

Program Writing

There are two kind of OTP file. One is program OTP file(***.OTP) and the other is font OTP file(***.FNT). You can make each file through ASMLINKER.exe and OSDFONT.exe respectively. All OTP file is Motolora S-format. You can burn the program file and font file respectively or together. To burn program file and font file respectively, refer following procedure

1. Make program OTP file and font OTP file repectively.
2. Check whether six test pattern is included in font OTP file(see below Six Text Pattern)
3. Burn program OTP file(Set chip target address 1000h ~ FFFFh)
4. Burn font OTP file(Set chip target address 10000h ~13FFFh)

Note: *When you program the OTP file, DO NOT check the blank. Because there are already written data(Six test pattern / 11FA0h ~ 11FFFh, 13FA0h~13FFFh) It will occur blank error*

To burn program file and font file together, refer following procedure

1. Add program OTP file and font OTP file
2. Check whether six test pattern is included in font OTP file(see below Six Text Pattern)
3. Burn OTP file(Set chip target address 1000h ~ 13FFFh)

About other details, refer ROM wirter manual.

Six Test Pattern

When you make font file through OSDFONT.exe, please confirm whether six test pattern is included or not in character address 1FAh ~ 1FFh, To include six test patern, refer following procedure.

1. Make Font file and save it to your PC
2. Reopen the font file and save it to your HDD once again.
3. Then six test pattern will be included automatically. (Character address 1FAh ~ 1FFh)

23.2 .Device configuration data

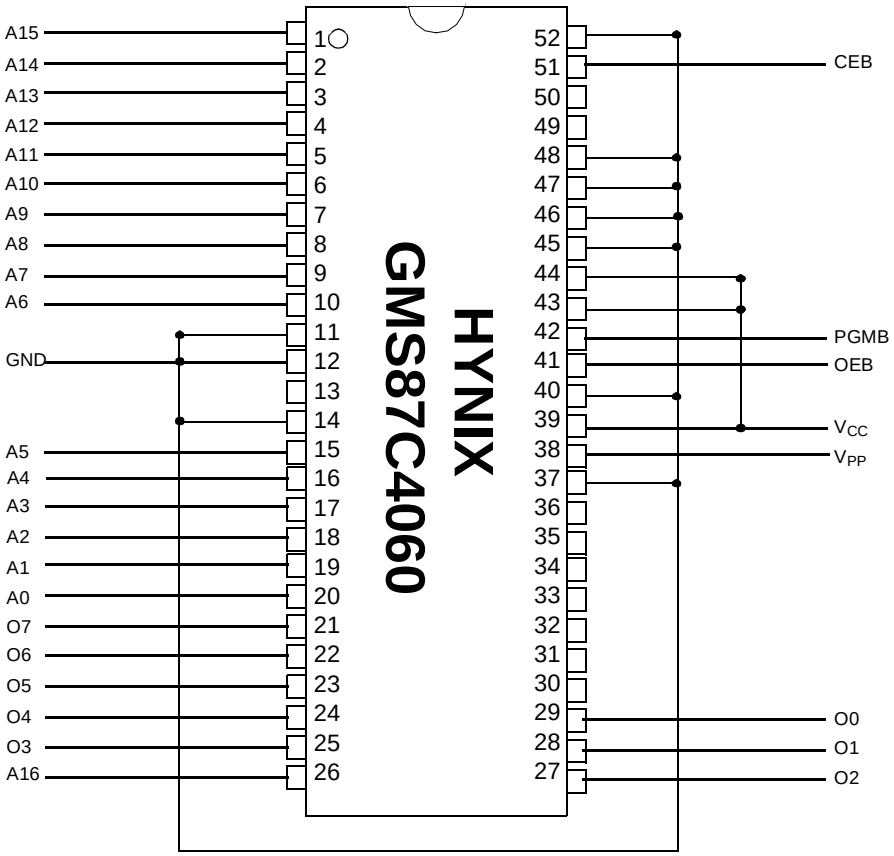


Figure 23-2 Figure Pin Configuration in OTP Programming Mode

	GMS87C4060				Intel 27010			
Mode	VPP	CEB	OEB	PGMB	VPP	CEB	OEB	PGMB
Program	12.75V	Low	High*2	Low*1	12.75V	Low	High	Low
Verify	12.75V	Low	Low	High	12.75V	Low	Low	High
Optional Verify	5V	Low	Low	X	5V	Low	Low	X
Gang Write *3	12.75V	Low	High	Low	12.75V	Low	High	Low
Gang Verify *4	12.75V, 5V	Low	Low	X	12.75V	Low	Low	X

Figure 23-3 Figure Mode Table

*1: Low = Input Low Voltage = V_{IL}(<0.8V)

*2: High = Input High Value = VIL(>2.0V)

*3: In Gang Write Mode, All OTPs are programmed simulaneously. So all signals of OTPs are in the same condition

*4: In Gang Verify mode, the VPP pin can be sset to both normal high(5V), aand 12.75V and chip slecection is possible using the CEB pin

SYMBOL	Parameter	Limits				Conditions
		Min	Typ	Max	Unit	
tAS	Address Setup Time	2			µs	
tOES	OEB Setup Time	2			µs	
tDS	Data Setup Time	2			µs	
tAH	Address Hold Time	0			µs	
tDH	Data Hold Time	2			µs	
tDFP	OEB High to Output Float Delay	0		130	ns	Note1
tVPS	Vpp Setup Time	2			µs	
tCES	CEB Setup Time	2			µs	
tPW	PGMB initial program pulse width	95	100	105	µs	Quick pulse programming
tOE	Data Valid from OEB			100	ns	
tACC	Address to output delay			150	ns	
tOH	output hold from addresses CEB or OEB whichever occurrs first	0		0	ns	
tCE	CEB to output delay			100	ns	
tCS	chip selection interval (@Gang verify)			100	ns	

*Note1: Output Float is defined as the point where data is no longer driven

Figure 23-4 Figure AC Programming Characteristics

Intel 27010		GMS87C4060	
Pin Name	Pin Number	Pin Name	Pin Number
VPP	1	TEST_N	38
A16	2	R67	26
A15	3	R27	1
A12	4	R24	4
A7	5	R17	9
A6	6	R16	10
A5	7	R15	15
A4	8	R14	16
A3	9	R13	17
A2	10	R12	18
A1	11	R11	19
A0	12	R10	20
O0	13	R00	29
O1	14	R01	28
O2	15	R02	27
GND	16	VSS	12, 40
O3	17	R03	25
O4	18	R04	24
O5	19	R05	23
O6	20	R06	22
O7	21	R07	21
CEB	22	R41	51
A10	23	R22	6
OEB	24	R53	41
A11	25	R23	5
A9	26	R21	7
A8	27	R20	8
A13	28	R25	3
A14	29	R26	2
N.C.	30		
PGMB	31	R52	42
VCC	32	VDD	39

Figure 23-5 Pin Mapping Table between Intel 27010/C010 and GMS87C4060

Pin Name	Pin Number	Connect to
RESET_N	11	GND
Xout	13	Not Connect
Xin	14	GND
R	30	Not Connect
G	31	Not Connect
B	32	Not Connect
R56	33	Not Connect
R55	34	Not Connect
R54	35	Not Connect
OSC2	36	Not Connect
OSC1	37	GND
R47	45	GND
R46	46	GND
R45	47	GND
R44	48	GND
R43	49	Not Connect
R42	50	Not Connect
R40	52	GND
R50	44	VDD
R51	43	VDD

Figure 23-6 Connection of Other Pins of GMS87C4060 in OTP Mode

23.3 Timing Chart

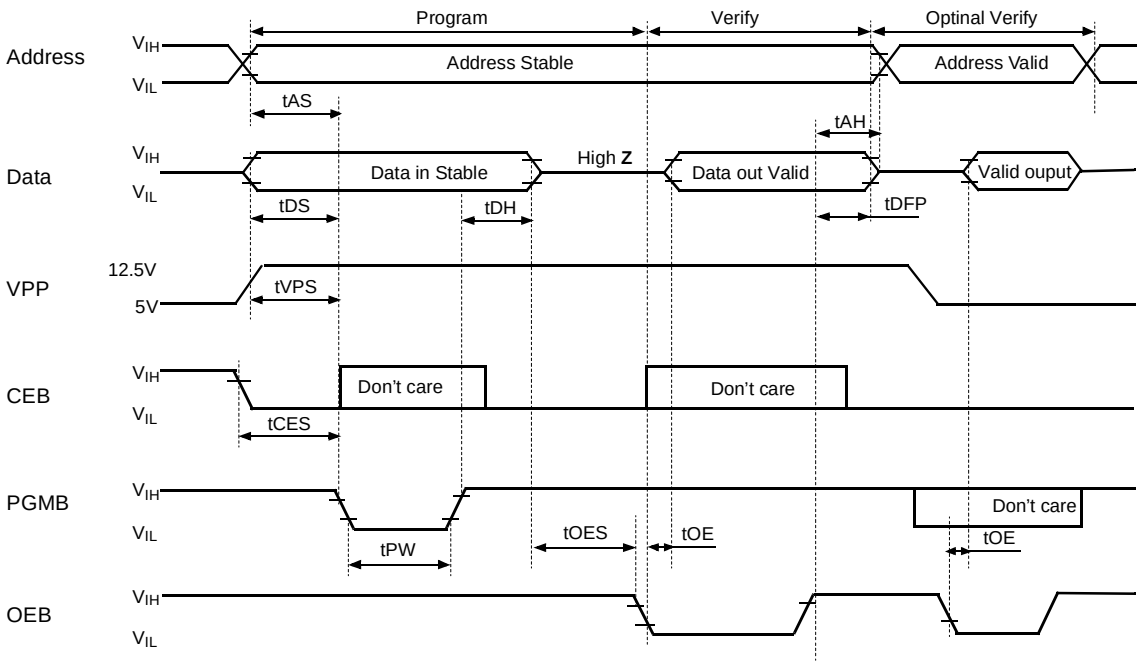
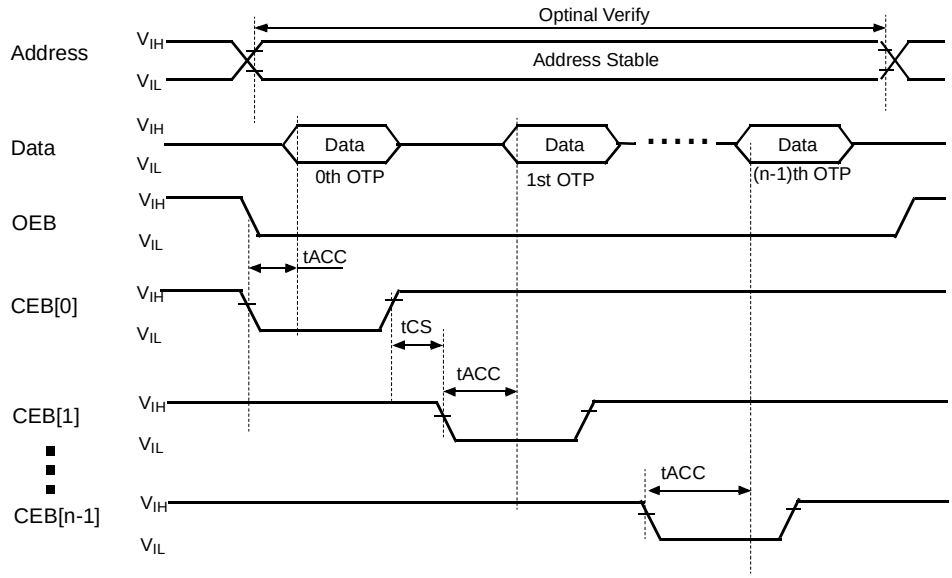


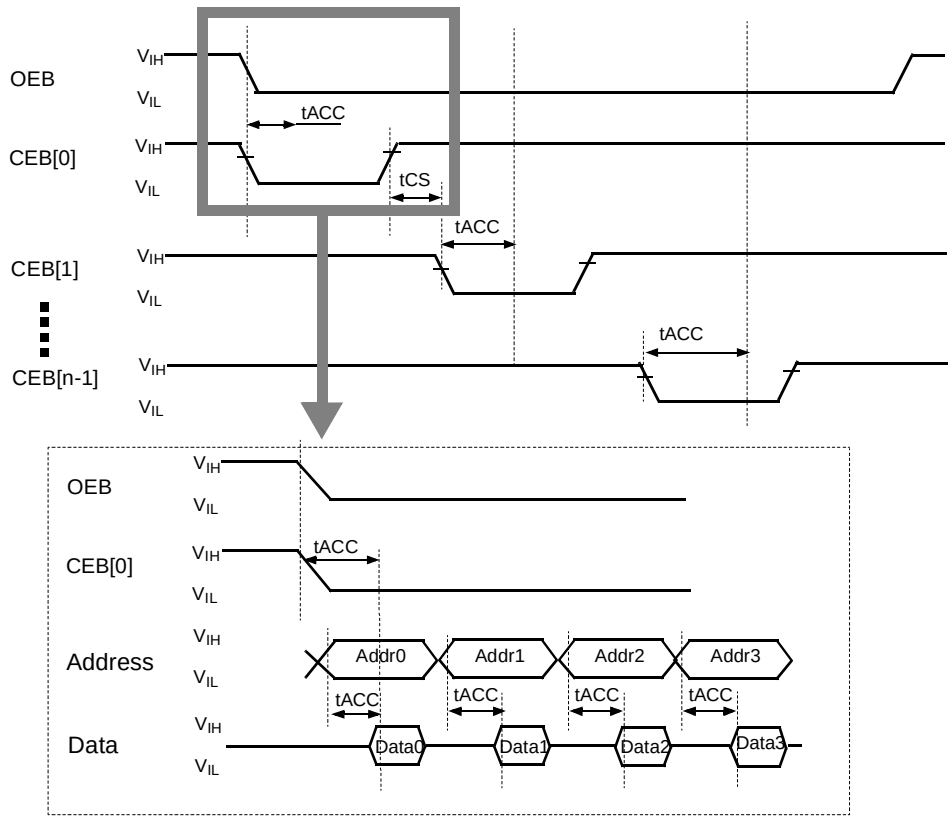
Figure 23-7 Figure Programming Timing Chart



1) When you verify the data in the same address of many OTPs.

When you select OTPs using CEB, and can verify the data inthe same address.

(PGMB : Don't care , Vpp : V_{IH} or 12.5V)



2) When you verify the data in s single OTP throughout the ROM address

Figure 23-8 AC Wave Form in Gang Verify Mode

24. Assemble mnemonics

24.1 Instruction Map

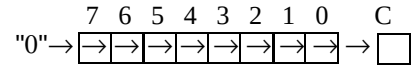
	00000 00	00001 01	00010 02	00011 03	00100 04	00101 05	00110 06	00111 07	01000 08	01001 09	01010 0A	01011 0B	01100 0C	01101 0D	01110 0E	01111 0F
000	NOP	SET1 dp.bit	BBS A.bit,rel	BBS dp.bit,rel	ADC #imm	ADC dp	ADC dp+X	ADC !abs	ASL A	ASL dp	TCALL 0	SETA1 .bit	BIT dp	POP A	PUSH A	BRK
001	CLRC	//	//	//	SBC #imm	SBC dp	SBC dp+X	SBC !abs	ROL A	ROL dp	TCALL 2	CLRA1 .bit	COM dp	POP X	PUSH X	BRA rel
010	CLRG	//	//	//	CMP #imm	CMP dp	CMP dp+X	CMP !abs	LSR A	LSR dp	TCALL 4	NOT1 M.bit	TST dp	POP Y	PUSH Y	PCALL Upage
011	DI	//	//	//	OR #imm	OR dp	OR dp+X	OR !abs	ROR A	ROR dp	TCALL 6	OR1 OR1B	CMPX dp	POP PSW	PUSH PSW	RET
100	CLRV	//	//	//	AND #imm	AND dp	AND dp+X	AND !abs	INC A	INC dp	TCALL 8	AND1 AND1B	CMPY dp	CBNE dp+X	TXSP	INC X
101	SETC	//	//	//	EOR #imm	EOR dp	EOR dp+X	EOR !abs	DEC A	DEC dp	TCALL 10	EOR1 EOR1B	DBNE dp	XMA dp+X	TSPX	DEC X
110	SETG	//	//	//	LDA #imm	LDA dp	LDA dp+X	LDA !abs	TXA	LDY dp	TCALL 12	LDC LDCB	LDX dp	LDX dp+Y	XCN	DAS
111	EI	//	//	//	LDM dp,#imm	STA dp	STA dp+X	STA !abs	TAX	STY dp	TCALL 14	STC M.bit	STX dp	STX dp+Y	XAS	

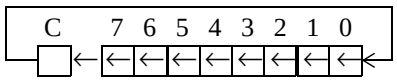
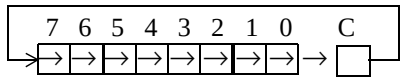
	10000 10	10001 11	10010 12	10011 13	10100 14	10101 15	10110 16	10111 17	11000 18	11001 19	11010 1A	11011 1B	11100 1C	11101 1D	11110 1E	11111 1F
000	BPL rel	CLR1 dp.bit	BBC A.bit,rel	BBC dp.bit,rel	ADC {X}	ADC !abs+Y	ADC [dp+X]	ADC [dp]+Y	ASL !abs	ASL dp+X	TCALL 1	JMP !abs	BIT !abs	ADDW dp	LDX #imm	JMP [!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

24.2 Alphabetic order table of instruction

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADC #imm	04	2	2	Add with carry.	NV - - H - ZC
2	ADC dp	05	2	3	$A \leftarrow A + (M) + C$	
3	ADC dp + X	06	2	4		
4	ADC !abs	07	3	4		
5	ADC !abs+Y	15	3	5		
6	ADC [dp+X]	16	2	6		
7	ADC [dp]+Y	17	2	6		
8	ADC {X}	14	1	3		
9	ADDW dp	1D	2	5	16-bits add without carry : YA ← YA + (dp+1)(dp)	NV - - H - ZC
10	AND #imm	84	2	2	Logical AND	N - - - - - Z -
11	AND dp	85	2	3	$A \leftarrow A \wedge (M)$	
12	AND dp + X	86	2	4		
13	AND !abs	87	3	4		
14	AND !abs+Y	95	3	5		
15	AND [dp+X]	96	2	6		
16	AND [dp] + Y	97	2	6		
17	AND {X}	94	1	3		
18	AND1 M.bit	8B	3	4	Bit AND C-flag : C ← C ^ (M.bit)	- - - - - - - C
19	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : C ← C ^ ~(M.bit)	- - - - - - - C
20	ASL A	08	1	2	Arithmetic shift left C 7 6 5 4 3 2 1 0 □ ← ← ← ← ← ← ← ← ← "0"	N - - - - - ZC
21	ASL dp	09	2	4		
22	ASL dp + X	19	2	5		
23	ASL !abs	18	3	5		
24	BBC A.bit,rel	y2	2	4/6	Branch if bit clear :	- - - - - - - -
25	BBC dp.bit,rel	y3	3	5/7	if(bit) = 0, then PC ← PC + rel	
26	BBS A.bit,rel	x2	2	4/6	Branch if bit clear :	- - - - - - - -
27	BBS dp.bit,rel	x3	3	5/7	if(bit) = 1, then PC ← PC + rel	
28	BCC rel	50	2	2/4	Branch if carry bit clear : if(C) = 0, then PC ← PC + rel	MM - - - - Z -
29	BCS rel	D0	2	2/4	Branch if carry bit set : If (C) =1, then PC ← PC + rel	- - - - - - - -
30	BEQ rel	F0	2	2/4	Branch if equal : if (Z) = 1, then PC ← PC + rel	- - - - - - - -
31	BIT dp	0C	2	4	Bit test A with memory :	MM - - - - Z -
32	BIT !abs	1C	3	5	$Z \leftarrow A \wedge M$, $N \leftarrow (M_7)$, $V \leftarrow (M_6)$	
33	BMI rel	90	2	2/4	Branch if munus : if (N) = 1, then PC ← PC + rel	- - - - - - - -
34	BNE rel	70	2	2/4	Branch if not equal : if (Z) = 0, then PC ← PC + rel	- - - - - - - -
35	BPL rel	10	2	2/4	Branch if not minus : if (N) = 0, then PC ← PC + rel	- - - - - - - -
36	BRA rel	2F	2	4	Branch always : PC ← PC + rel	- - - - - - - -
37	BRK	0F	1	8	Software interrupt: B ← "1", $M(SP) \leftarrow (PC_H)$, SP ← SP - 1, $M(s) \leftarrow (PC_L)$, SP ← S - 1, M(SP) ← PSW, SP ← SP - 1, $PC_L \leftarrow (0FFDE_H)$, $PC_H \leftarrow (0FFDF_H)$	- - - 1 - 0 - -
38	BVC rel	30	2	2/4	Branch if overflow bit clear : If (V) = 0, then PC ← PC + rel	- - - - - - - -
39	BVS rel	B0	2	2/4	Branch if overflow bit set : If (V) = 1, then PC ← PC + rel	- - - - - - - -
40	CALL !abs	3B	3	8	Subroutine call	- - - - - - - -
41	CALL [dp]	5F	2	8	$M(SP) \leftarrow (PC_H)$, SP ← SP-1, $M(SP) \leftarrow (PC_L)$, SP←SP-1 if !abs, PC ← abs ; if [dp], $PC_L \leftarrow (dp)$, $PC_H \leftarrow (dp+1)$	
42	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal ;	- - - - - - - -
43	CBNE dp + X, rel	8D	3	6/8	If $A \neq (M)$, then PC ← PC + rel.	
44	CLR1 dp.bit	y1	2	4	Clear bit : (M.bit) ← "0"	- - - - - - - -
45	CLR1A A.bit	2B	2	2	Clear A.bit : (A.bit) ← "0"	- - - - - - - -
46	CLRC	20	1	2	Clear C-flag : C ← "0"	- - - - - - - 0
47	CLRG	40	1	2	Clear G-flag : G ← "0"	- - 0 - - - - -

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
48	CLRV	80	1	2	Clear V-flag : V ← “0”	- 0 - - 0 - - -
49	CMP #imm	44	2	2	Compare accumulator contents with memory contents	
50	CMP dp	45	2	3	A - (M)	
51	CMP dp + X	46	2	4		
52	CMP !abs	47	3	4		N - - - - - ZC
53	CMP !abs + Y	55	3	5		
54	CMP [dp + X]	56	2	6		
55	CMP [dp] + Y	57	2	6		
56	CMP {X}	54	1	3		
57	CMPW dp	5D	2	4	Compare YA contents with memory pair contents : YA - (dp+1)(dp)	N - - - - - ZC
58	CMPX #imm	5E	2	2	Compare X contents with memory contents	
59	CMPX dp	6C	2	3	X - (M)	N - - - - - ZC
60	CMPX !abs	7C	3	4		
61	CMPY #imm	7E	2	2	Compare Y contents with memory contents	
62	CMPY dp	8C	2	3	Y - (M)	N - - - - - ZC
63	CMPY !abs	9C	3	4		
64	COM dp	2C	2	4	1’s complement : (dp) ← ~(dp)	N - - - - - Z -
65	DAA	DF	1	3	Decimal adjust for addition	N - - - - - ZC
66	DAS	CF	1	3	Decimal adjust for substraction	N - - - - - ZC
67	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal :	- - - - - - - -
68	DBNE Y,rel	7B	2	4/6	if (M) ≠ 0, then PC ← PC + rel.	
69	DEC A	A8	1	2	Decrement	
70	DEC dp	A9	2	4	M ← M - 1	
71	DEC dp + X	B9	2	5		
72	DEC !abs	B8	3	5		N - - - - - Z -
73	DEC X	AF	1	2		
74	DEC Y	BE	1	2		
75	DECW dp	BD	2	6	Decrement memory pair : (dp+1)(dp) ← {(dp+1)(dp)} - 1	N - - - - - Z -
76	DI	60	1	3	Disable interrupts : I ← “0”	- - - - - 0 - -
77	DIV	9B	1	12	Divide : YA / X ← Q:A, R:Y	NV - - H - Z -
78	EI	E0	1	3	Enable interrupts : I ← “1”	- - - - - 1 - -
79	EOR #imm	A4	2	2	Exclusive OR	
80	EOR dp	A5	2	3	A ← A ⊕ (M)	
81	EOR dp + X	A6	2	4		
82	EOR !abs	A7	3	4		
83	EOR !abs + Y	B5	3	5		N - - - - - Z -
84	EOR [dp + X]	96	2	6		
85	EOR [dp] + Y	97	2	6		
86	EOR {X}	94	1	3		
87	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : C ← C ⊕ (M.bit)	- - - - - - - C
88	EOR1B M.bit	AB	3	5	Bit exclusive-OR C-flag and NOT : C ← C ⊕ ~(M.bit)	- - - - - - - C
89	INC A	88	1	2	Increment	N - - - - - ZC
90	INC dp	89	2	4	(M) ← (M) + 1	
91	INC dp + X	99	2	5		
92	INC !abs	98	3	5		N - - - - - Z -
93	INC X	8F	1	2		
94	INC Y	9E	1	2		
95	INCW dp	9D	2	6	Increment memory pair : (dp+1)(dp) ← {(dp+1)(dp)} + 1	N - - - - - Z -
96	JMP !abs	1B	3	3	Unconditional jump	
97	JMP [!abs]	1F	3	5	PC ← jump address	- - - - - - - -
98	JMP [dp]	3F	2	4		

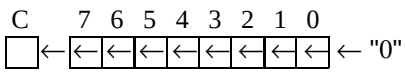
NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
99	LDA #imm	C4	2	2	Load accumulator	N - - - - - Z -
100	LDA dp	C5	2	3	A ← (M)	
101	LDA dp + X	C6	2	4		
102	LDA !abs	C7	3	4		
103	LDA !abs + Y	D5	3	5		
104	LDA [dp + X]	D6	2	6		
105	LDA [dp]+Y	D7	2	6		
106	LDA {X}	D4	1	3		
107	LDA {X}+	DB	1	4	X-register auto-increment : A ← (M), X ← X + 1	
108	LDC M.bit	CB	3	4	Load C-flag : C ← (M.bit)	- - - - - - - C
109	LDCB M.bit	CB	3	4	Load C-flag with NOT : C ← ~(M.bit)	- - - - - - - C
110	LDM dp,#imm	E4	3	5	Load memory with immediate data : (M) ← imm	- - - - - - - -
111	LDX #imm	1E	2	2	Load X-register	N - - - - - Z -
112	LDX dp	CC	2	3	X ← (M)	
113	LDX dp + Y	CD	2	4		
114	LDX !abs	DC	3	4		
115	LDY #imm	3E	2	2	Load X-register	N - - - - - Z -
116	LDY dp	C9	2	3	Y ← (M)	
117	LDY dp + Y	D9	2	4		
118	LDY !abs	D8	3	4		
119	LDYA dp	7D	2	5	Load YA : YA ← (dp+1)(dp)	N - - - - - Z -
120	LSR A	48	1	2	Logical shift right	N - - - - - ZC
121	LSR dp	49	2	4		
122	LSR dp + X	59	2	5		
123	LSR !abs	58	3	5		
124	MUL	5B	1	9	Multiply : YA ← Y x A	N - - - - - Z -
125	NOP	00,FF	1	2	No operation	- - - - - - - -
126	NOT1 M.bit	4B	3	5	Bit complement : (M.bit) ← ~(M.bit)	- - - - - - - -
127	OR #imm	64	2	2	Logical OR	N - - - - - Z -
128	OR dp	65	2	3	A ← A V (M)	
129	OR dp + X	66	2	4		
130	OR !abs	67	3	4		
131	OR !abs + Y	75	3	5		
132	OR [dp +X}	76	2	6		
133	OR [dp] + Y	77	2	6		
134	OR {X}	74	1	3		
135	OR1 M.bit	6B	3	5	Bit OR C-flag : C ← C V (M.bit)	- - - - - - - C
136	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : C ← C V ~(M.bit)	- - - - - - - C
137	PCALL	4F	2	6	U-page call : M(SP) ← (PC_H), SP ← SP -1, M(SP) ← (PC_L), SP ← SP -1, PC_L ← (upage), PC_H ←"OFF_H"	- - - - - - - -
138	POP A	0D	1	4	Pop from stack	- - - - - - - -
139	POP X	2D	1	4	SP ← SP + 1, Reg. ← M(SP)	
140	POP Y	4D	1	4		
141	POP PSW	6D	1	4		(restored)
142	PUSH A	0E	1	4	Push to stack	- - - - - - - -
143	PUSH X	2E	1	4	M(SP) ← Reg. SP ← SP - 1	
144	PUSH Y	4E	1	4		
145	PUSH PSW	6E	1	4		
146	RET	6F	1	5	Return from subroutine : SP ← SP+1, PC_L ← M(SP), SP ← SP+1, PC_H ← M(SP)	- - - - - - - -
147	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)

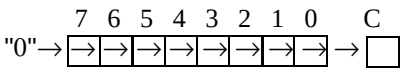
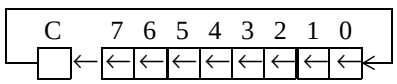
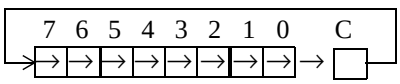
NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
148	ROL A	28	1	2	Rotate left through carry 	N - - - - - ZC
149	ROL dp	29	2	4		
150	ROL dp + X	39	2	5		
151	ROL !abs	38	3	5		
152	ROR A	68	1	2	Rotate right through carry 	N - - - - - ZC
153	ROR dp	69	2	4		
154	ROR dp + X	79	2	5		
155	ROR !abs	78	3	5		
156	SBC #imm	24	2	2	Substract with carry A ← A - (M) - ~(C)	NV - - HZC
157	SBC dp	25	2	3		
158	SBC dp + X	26	2	4		
159	SBC !abs	27	3	4		
160	SBC !abs + Y	35	3	5		
161	SBC [dp + X]	36	2	6		
162	SBC [dp] + Y	37	2	6		
163	SBC {X}	34	1	3		
164	SET1 dp.bit	x1	2	4	Set bit : (M.bit) ← “1”	- - - - - - - -
165	SETA1 A.bit	0B	2	2	Set A.bit : (A.bit) ← “1”	- - - - - - - -
166	SETC	A0	1	2	Set C-flag : C ← “1”	- - - - - - - 1
167	SETG	C0	1	2	Set G-flag : G ← “1”	- - 1 - - - - -
168	STA dp	E5	2	3	Store accumulator contents in memory (M) ← A	- - - - - - - -
169	STA dp + X	E6	2	4		
170	STA !abs	E7	3	4		
171	STA !abs + Y	F5	3	5		
172	STA [dp + X]	F6	2	6		
173	STA [dp] + Y	F7	2	6		
174	STA {X}	F4	1	3		
175	STA {X}+	FB	1	4	X-register auto-increment : (M) ← A, X ← X + 1	
176	STC M.bit	EB	3	6	Store C-flag : (M.bit) ← C	- - - - - - - -
177	STX dp	EC	2	4	Store X-register contents in memory (M) ← X	- - - - - - - -
178	STX dp + Y	ED	2	5		
179	STX !abs	FC	3	5		
180	STY dp	E9	2	4	Store Y-register contents in memory (M) ← Y	- - - - - - - -
181	STY dp + X	F9	2	5		
182	STY !abs	F8	3	5		
183	STYA dp	DD	2	5	Store YA : (dp+1)(dp) ← YA	- - - - - - - -
184	SUBW dp	3D	2	5	16-bits substract without carry : YA ← YA - (dp+1)(dp)	NV - - H - ZC
185	TAX	E8	1	2	Transfer accumulator contents to X-register : X ← A	N - - - - - Z -
186	TAY	9F	1	2	Transfer accumulator contents to Y-register : Y ← A	N - - - - - Z -
187	TCALL n	nA	1	8	Table call : M(SP) ← (PC_H), SP ← SP -1, M(SP) ← (PC_L), SP ← SP -1 PC_L ← (Table vector L), PC_H ← (Table vector H)	- - - - - - - -
188	TCLR1 !abs	5C	3	6	Test and clear bits with A : A - (M), (M) ← (M) ^ ~(A)	N - - - - - Z -
189	TSET1 !abs	3C	3	6	Test and set bits with A : A - (M), (M) ← (M) V (A)	N - - - - - Z -
190	TSPX	AE	1	2	Transfer stack-pointer contents to X-register : X ← SP	N - - - - - Z -
191	TST dp	4C	2	3	Test memory contents for negative or zero : (dp) - 00_H	N - - - - - Z -
192	TXA	C8	1	2	Transfer X-register contents to accumulator : A ← X	N - - - - - Z -
193	TXSP	8E	1	2	Transfer X-register contents to stack-pointer : SP ← X	N - - - - - Z -
194	TYA	BF	1	2	Transfer Y-register contents to accumulator : A ← Y	N - - - - - Z -
195	XAX	EE	1	4	Exchange X-register contents with accumulator : X fA	- - - - - - - -
196	XAY	DE	1	4	Exchange Y-register contents with accumulator : Y fA	- - - - - - - -

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
197	XCN	CE	1	5	Exchange nibbles within the accumulator: $A_7 \sim A_4$ f $A_3 \sim A_0$	N - - - - - Z -
198	XMA dp	BC	2	5	Exchange memory contents with accumulator (M) f A	N - - - - - Z -
199	XMA dp + X	AD	2	6		
200	XMA {X}	BB	1	5		
201	XYX	FE	1	4	Exchange X-register contents with Y-register : X f Y	- - - - - - - -

24.3 Instruction Table by Function

1. Arithmetic/Logic Operation

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

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
44	DIV	9B	1	12	Divide : YA / X ← Q:A, R:Y	NV - - H - Z -
45	EOR #imm	A4	2	2	Exclusive OR	
46	EOR dp	A5	2	3	A ← A ⊕ (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]	96	2	6		
51	EOR [dp] + Y	97	2	6		
52	EOR {X}	94	1	3		
53	INC A	88	1	2	Increment	N - - - - - ZC
54	INC dp	89	2	4	(M) ← (M) + 1	
55	INC dp + X	99	2	5		
56	INC !abs	98	3	5		N - - - - - Z -
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		
61	LSR dp + X	59	2	5		N - - - - - ZC
62	LSR !abs	58	3	5		
63	MUL	5B	1	9	Multiply : YA ← Y x A	N - - - - - Z -
64	OR #imm	64	2	2	Logical OR	
65	OR dp	65	2	3	A ← A V (M)	
66	OR dp + X	66	2	4		
67	OR !abs	67	3	4		
68	OR !abs + Y	75	3	5		N - - - - - Z -
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		
74	ROL dp + X	39	2	5		N - - - - - ZC
75	ROL !abs	38	3	5		
76	ROR A	68	1	2	Rotate right through carry	
77	ROR dp	69	2	4		
78	ROR dp + X	79	2	5		N - - - - - ZC
79	ROR !abs	78	3	5		
80	SBC #imm	24	2	2	Substract with carry	
81	SBC dp	25	2	3	A ← A - (M) - ~(C)	
82	SBC dp + X	26	2	4		
83	SBC !abs	27	3	4		
84	SBC !abs + Y	35	3	5		NV - - HZC
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: $A_7 \sim A_4$ f $A_3 \sim A_0$	N - - - - - Z -

2. Register / Memory Operation

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	LDA #imm	C4	2	2	Load accumulator	N - - - - - Z -
2	LDA dp	C5	2	3	A ← (M)	
3	LDA dp + X	C6	2	4		
4	LDA !abs	C7	3	4		
5	LDA !abs + Y	D5	3	5		
6	LDA [dp + X]	D6	2	6		
7	LDA [dp]+Y	D7	2	6		
8	LDA {X}	D4	1	3		
9	LDA {X}+	DB	1	4	X-register auto-increment : A ← (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	N - - - - - Z -
12	LDX dp	CC	2	3	X ← (M)	
13	LDX dp + Y	CD	2	4		
14	LDX !abs	DC	3	4		
15	LDY #imm	3E	2	2	Load X-register	N - - - - - Z -
16	LDY dp	C9	2	3	Y ← (M)	
17	LDY dp + Y	D9	2	4		
18	LDY !abs	D8	3	4		
19	STA dp	E5	2	3	Store accumulator contents in memory	- - - - - - - -
20	STA dp + X	E6	2	4	(M) ← A	
21	STA !abs	E7	3	4		
22	STA !abs + Y	F5	3	5		
23	STA [dp + X]	F6	2	6		
24	STA [dp] + Y	F7	2	6		
25	STA {X}	F4	1	3		
26	STA {X}+	FB	1	4	X-register auto-increment : (M) ← A, X ← X + 1	
27	STX dp	EC	2	4	Store X-register contents in memory	- - - - - - - -
28	STX dp + Y	ED	2	5	(M) ← X	
29	STX !abs	FC	3	5		
30	STY dp	E9	2	4	Store Y-register contents in memory	- - - - - - - -
31	STY dp + X	F9	2	5	(M) ← Y	
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 fA	- - - - - - - -
40	XAY	DE	1	4	Exchange Y-register contents with accumulator : Y fA	- - - - - - - -
41	XMA dp	BC	2	5	Exchange memory contents with accumulator	N - - - - - Z -
42	XMA dp + X	AD	2	6	(M) f A	
43	XMA {X}	BB	1	5		
44	XYX	FE	1	4	Exchange X-register contents with Y-register : X f Y	- - - - - - - -

3. 16-Bit Operation

NO.	MNENONIC	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 -

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
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 substract without carry : YA ← YA - (dp+1)(dp)	NV - - H - ZC

4. Bit Manipulation

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	AND1 M.bit	8B	3	4	Bit AND C-flag : C ← C ^ (M.bit)	- - - - - - - C
2	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : C ← C ^ ~(M.bit)	- - - - - - - C
3	BIT dp	0C	2	4	Bit test A with memory : Z ← A ^ M, N ← (M_7), V ← (M_6)	MM - - - - Z -
4	BIT !abs	1C	3	5		
5	CLR1 dp.bit	y1	2	4	Clear bit : (M.bit) ← “0”	- - - - - - - -
6	CLR1A 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 V (M.bit)	- - - - - - - C
16	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : C ← C V ~(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) ^ ~(A)	N - - - - - Z -
23	TSET1 !abs	3C	3	6	Test and set bits with A : A - (M), (M) ← (M) V (A)	N - - - - - Z -

5. Branch / Jump Operation

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BBC A.bit,rel	y2	2	4/6	Branch if bit clear : if(bit) = 0, then PC ← PC + rel	- - - - - - - -
2	BBC dp.bit,rel	y3	3	5/7		
3	BBS A.bit,rel	x2	2	4/6	Branch if bit clear : if(bit) = 1, then PC ← PC + rel	- - - - - - - -
4	BBS dp.bit,rel	x3	3	5/7		
5	BCC rel	50	2	2/4	Branch if carry bit clear : if(C) = 0, then PC ← PC + rel	MM - - - - Z -
6	BCS rel	D0	2	2/4	Branch if carry bit set : If (C) =1, then PC ← PC + rel	- - - - - - - -
7	BEQ rel	F0	2	2/4	Branch if equal : if (Z) = 1, then PC ← PC + rel	- - - - - - - -
8	BMI rel	90	2	2/4	Branch if munus : if (N) = 1, then PC ← PC + rel	- - - - - - - -
9	BNE rel	70	2	2/4	Branch if not equal : if (Z) = 0, then PC ← PC + rel	- - - - - - - -
10	BPL rel	10	2	2/4	Branch if not minus : if (N) = 0, then PC ← PC + rel	- - - - - - - -
11	BRA rel	2F	2	4	Branch always : PC ← PC + rel	- - - - - - - -
12	BVC rel	30	2	2/4	Branch if overflow bit clear : If (V) = 0, then PC ← PC + rel	- - - - - - - -
13	BVS rel	B0	2	2/4	Branch if overflow bit set : If (V) = 1, then PC ← PC + rel	- - - - - - - -

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
14	CALL !abs	3B	3	8	Subroutine call	
15	CALL [dp]	5F	2	8	$M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP-1$, $M(SP) \leftarrow (PC_L)$, $SP \leftarrow SP-1$ if !abs, $PC \leftarrow$ abs ; if [dp], $PC_L \leftarrow (dp)$, $PC_H \leftarrow (dp+1)$	- - - - - - - -
16	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal ;	
17	CBNE dp + X, rel	8D	3	6/8	If $A \neq (M)$, then $PC \leftarrow PC + rel$.	- - - - - - - -
18	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal :	
19	DBNE Y,rel	7B	2	4/6	if $(M) \neq 0$, then $PC \leftarrow PC + rel$.	- - - - - - - -
20	JMP !abs	1B	3	3	Unconditional jump	
21	JMP [!abs]	1F	3	5	$PC \leftarrow$ jump address	- - - - - - - -
22	JMP [dp]	3F	2	4		
23	PCALL	4F	2	6	U-page call : $M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP -1$, $M(SP) \leftarrow (PC_L)$, $SP \leftarrow SP -1$, $PC_L \leftarrow$ (upage), $PC_H \leftarrow$"OFF_H"	- - - - - - - -
24	TCALL n	nA	1	8	Table call : $M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP -1$, $M(SP) \leftarrow (PC_L)$, $SP \leftarrow SP -1$ $PC_L \leftarrow$ (Table vector L), $PC_H \leftarrow$ (Table vector H)	- - - - - - - -

6. Control Operation & etc.

NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BRK	0F	1	8	Software interrupt: $B \leftarrow$ "1", $M(SP) \leftarrow (PC_H)$, $SP \leftarrow SP - 1$, $M(s) \leftarrow (PC_L)$, $SP \leftarrow S - 1$, $M(SP) \leftarrow PSW$, $SP \leftarrow SP - 1$, $PC_L \leftarrow (0FFDE_H)$, $PC_H \leftarrow (0FFDF_H)$	- - - 1 - 0 - -
2	DI	60	1	3	Disable interrupts : $I \leftarrow$ "0"	- - - - - 0 - -
3	EI	E0	1	3	Enable interrupts : $I \leftarrow$ "1"	- - - - - 1 - -
4	NOP	FF	1	2	No operation	- - - - - - - -
5	POP A	0D	1	4	Pop from stack	
6	POP X	2D	1	4	$SP \leftarrow SP + 1$, Reg. $\leftarrow M(SP)$	- - - - - - - -
7	POP Y	4D	1	4		
8	POP PSW	6D	1	4		(restored)
9	PUSH A	0E	1	4	Push to stack	
10	PUSH X	2E	1	4	$M(SP) \leftarrow$ Reg. $SP \leftarrow SP - 1$	- - - - - - - -
11	PUSH Y	4E	1	4		
12	PUSH PSW	6E	1	4		
13	RET	6F	1	5	Return from subroutine : $SP \leftarrow SP+1$, $PC_L \leftarrow M(SP)$, $SP \leftarrow SP+1$, $PC_H \leftarrow M(SP)$	- - - - - - - -
14	RETI	7F	1	6	Return from interrupt : $SP \leftarrow SP+1$, $PSW \leftarrow M(SP)$, $SP \leftarrow SP+1$, $PC_L \leftarrow M(SP)$, $SP \leftarrow SP+1$, $PC_H \leftarrow M(SP)$	(restored)