HYNIX SEMICONDUCTOR INC.
8-BIT SINGLE-CHIP MICROCONTROLLERS

HMS81C4x60

User's Manual (Ver. 1.1)

---

hynix
Semiconductor

Version 1.1

Published by MCU Application Team
Heung-il Bae(hibae@hynix.com), Byoung-jin Lim(bjinlim@hynix.com)

Additional information of this manual may be served by Hynix Semiconductor offices in Korea or Distributors and Representatives listed at address directory.

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

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

HMS81C4x60

CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER FOR TELEVISION

1. OVERVIEW

1.1 Description

The HMS81C4x60 is an advanced CMOS 8-bit microcontroller with 60K bytes of ROM. This is one of the HMS800 family. This is a powerful microcontroller which provides a high flexibility and cost effective solution to many TV applications. The HMS81C4x60 provides following standard features: 60K bytes of ROM, 1024 bytes of RAM, 8/16-bit timer/counter, on-chip PLL oscillator and clock circuitry. In addition, there are other package types, HMS81C4360(32PDIP), HMS81C4360SK(32SKDIP), HMS81C4460(42SDIP).
This document is explained for the base of HMS81C4x60, the eliminated functions are same as below.

Device name	ROM Size	EPROM Size	RAM Size	I/O	Package
HMS81C4260	60K bytes	-	1024bytes	31	52SDIP
HMS87C4260		60K bytes	1024bytes	31	52SDIP

1.2 Features

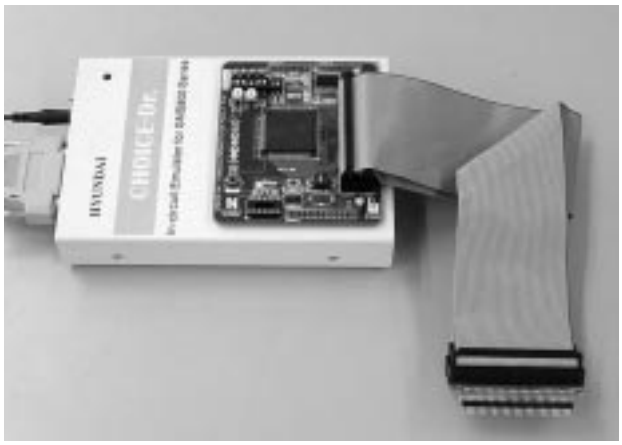
- **60K Bytes of On-chip Program Memory**
- **1024 Bytes of On-chip Data RAM**
- **Minimum Instruction Cycle Time**
  - 256ns (NOP operation)
- **PLL Oscillator for OSD and System Clock**
  - External 4MHz Crystal Input
- **31 Programmable I/O pins**
  - 26 Input/Output and 5 Input pins
- **I^2C Bus Interface**
  - Multimaster (2 Pairs interface pins)
- **A/D Converter**
  - 8-bit $\times$ 5 ch
- **Pulse Width Modulation**
  - 14-bit $\times$ 1 ch
  - 8-bit $\times$ 5 ch
- **Timer**
  - Timer/Counter : 8-bit $\times$ 4 ch(16-bit $\times$ 2 ch)
  - Basic interval timer
  - Watch Dog Timer
- **Number of Interrupt Source**
  - 16 Interrupts
  - 3 External Interrupts
- **On Screen Display**
  - 512 character fonts pattern
  - Character Size : 1.0, 1.5, 2.0 times
  - Character Pixel size : 12×10, 12×12, 12×14, 12×16, 16×18
  - Display Capability : 48 Characters $\times$ 16 Lines
  - Character, Background color : 512 colors, 8 pallet
  - Special functions : Rounding, Outline, Shadow, Underline, Double scanned line OSD
- **Buzzer Driving Port**
  - 500Hz ~ 250KHz @4MHz (Duty 50%)
- **Vertical Blanking Interveral Information capture for EIA-608(Closed Caption) or VPS, etc**

1.3 Development Tools

Note: *There are several setting switches in the Emulator. User should read carefully and do setting properly before developing the program. Otherwise, the Emulator may not work properly.*

The HMS87C4x60 is supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr.^{TM} and EPROM programmers. There are two different type programmers such as single type and gang type. For more detail, refer to EPROM Programming chapter. Macro assembler operates under the MS-Windows 95/98^{TM}.

Please contact sales part of Hynix Semiconductor.



1.4 Ordering Information

	Device name	ROM Size (bytes)	RAM size	Package
Mask ROM version	HMS81C4260	60K bytes	1024 bytes	52SDIP
OTP ROM version	HMS87C4260	60K bytes EPROM (OTP)	1024 bytes	52SDIP
Mask ROM version	HMS81C4360SK	60K bytes	1024 bytes	32SKDIP
OTP ROM version	HMS87C4360SK	60K bytes EPROM (OTP)	1024 bytes	32SKDIP
Mask ROM version	HMS81C4360	60K bytes	1024 bytes	32PDIP
OTP ROM version	HMS87C4360	60K bytes EPROM (OTP)	1024 bytes	32PDIP
Mask ROM version	HMS81C4460	60K bytes	1024 bytes	42SDIP
OTP ROM version	HMS87C4460	60K bytes EPROM (OTP)	1024 bytes	42SDIP

2. BLOCK DIAGRAM

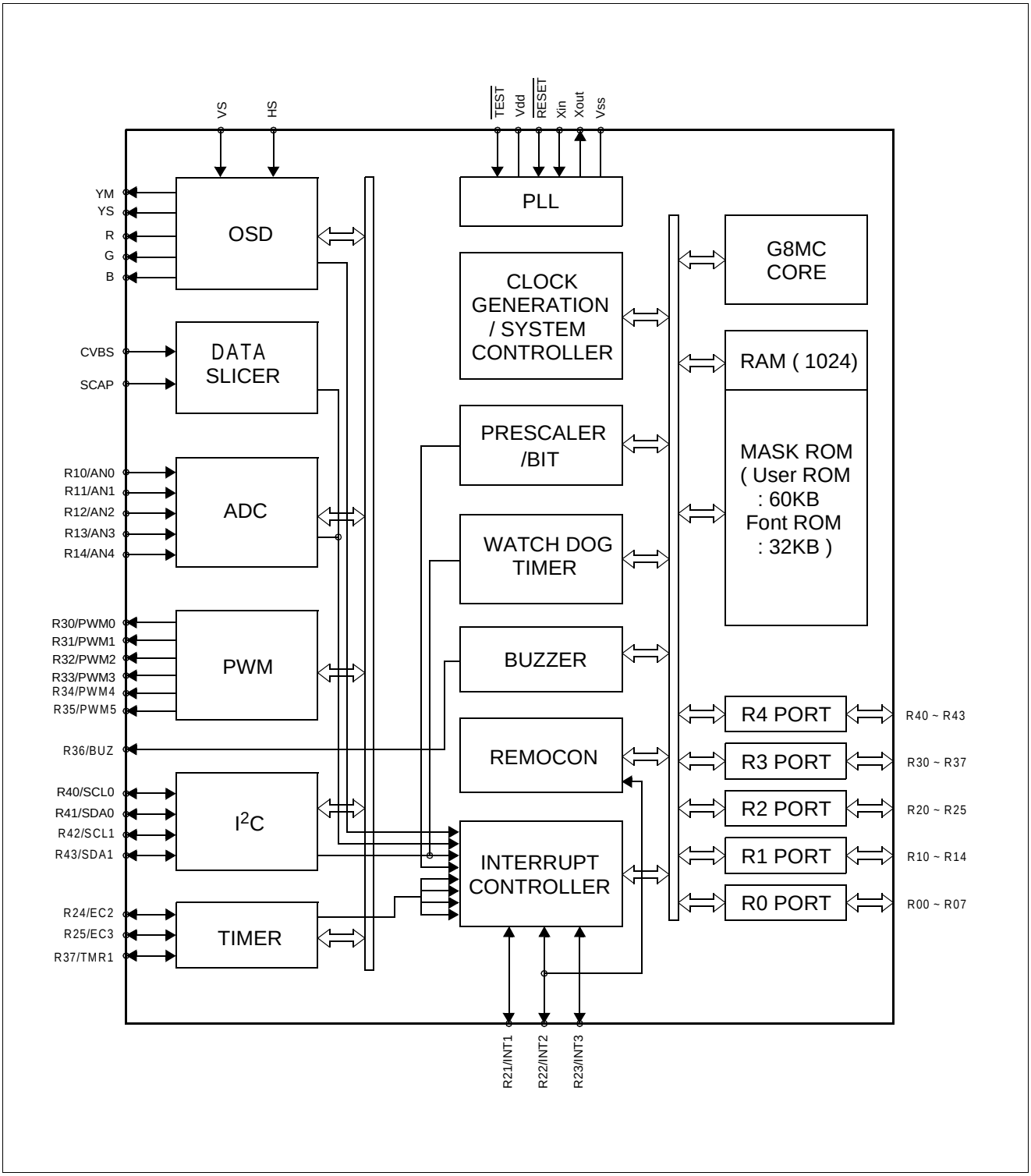


Figure 2-1 Block Diagram

3. PIN ASSIGNMENT

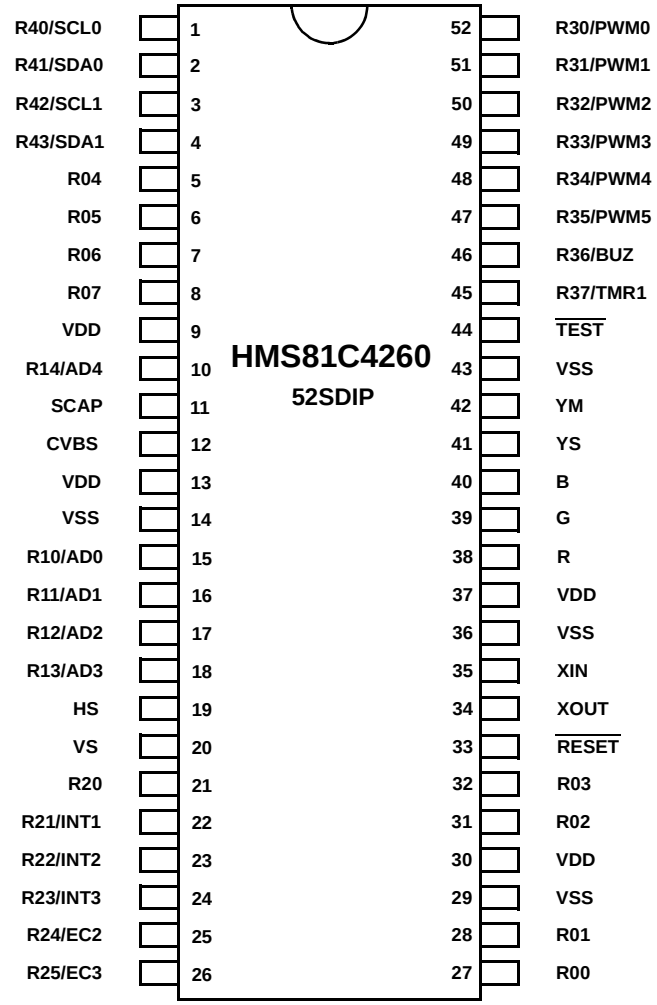


Figure 3-1 52SDIP

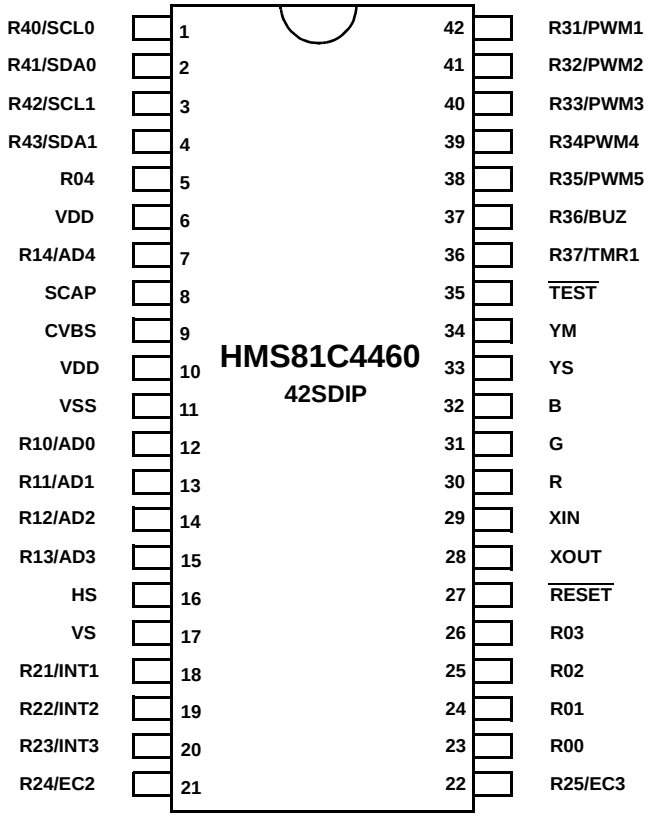


Figure 3-2 42SDIP

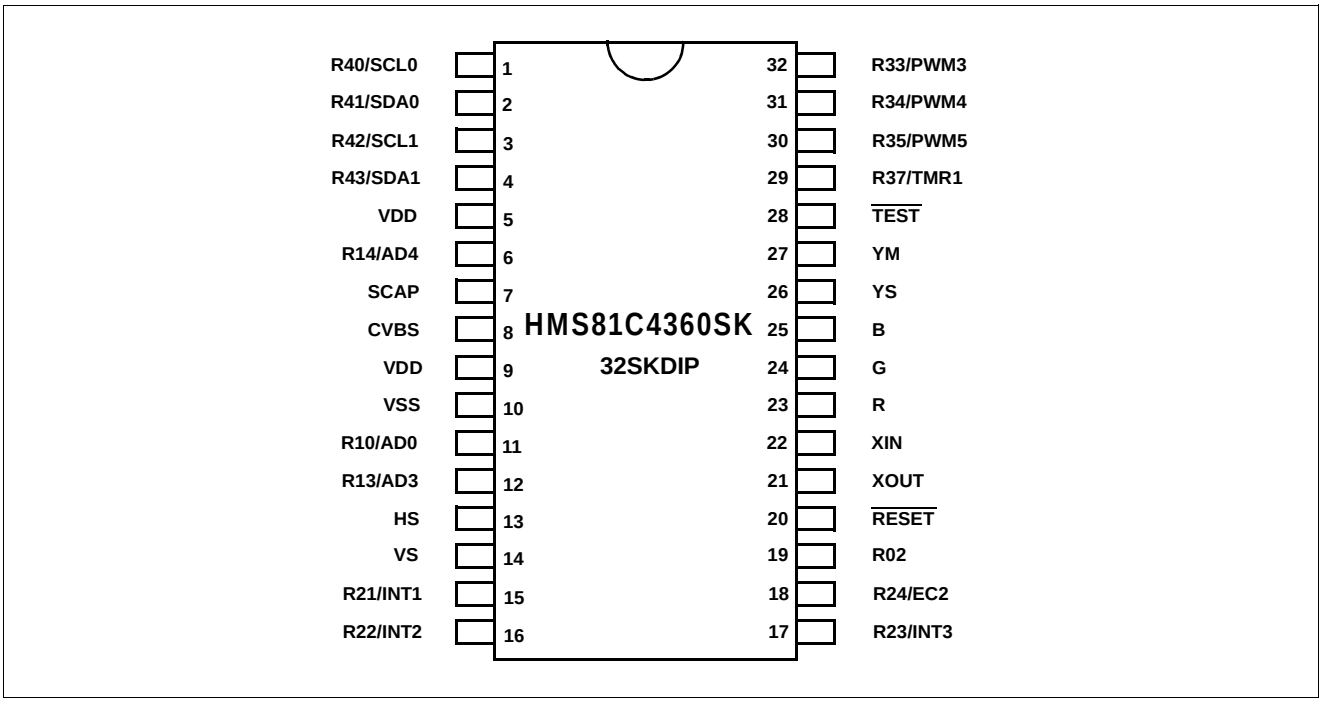


Figure 3-3 32SKDIP

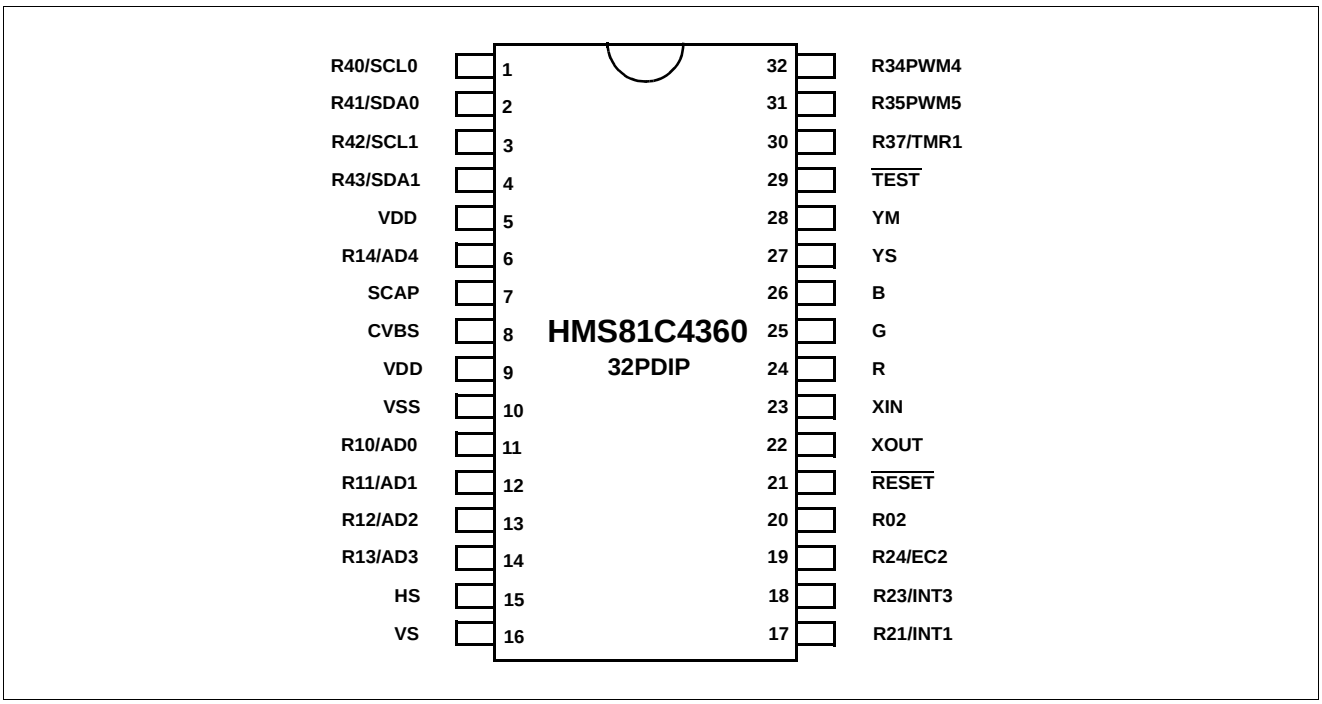
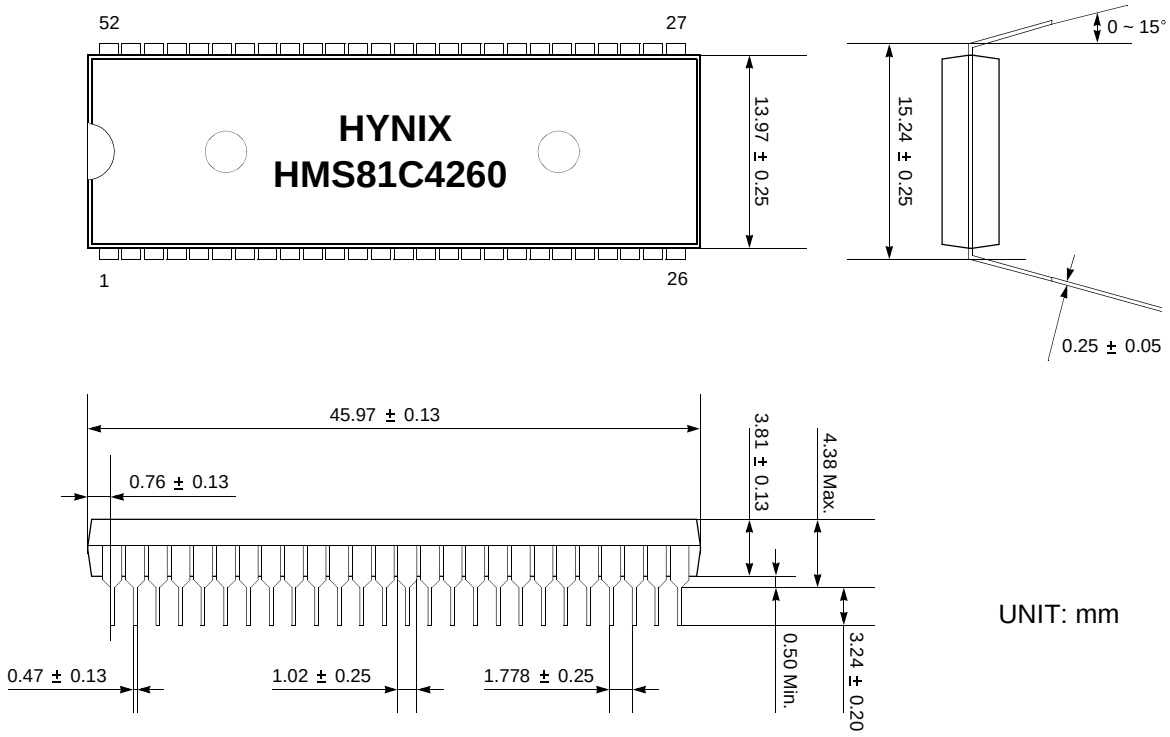
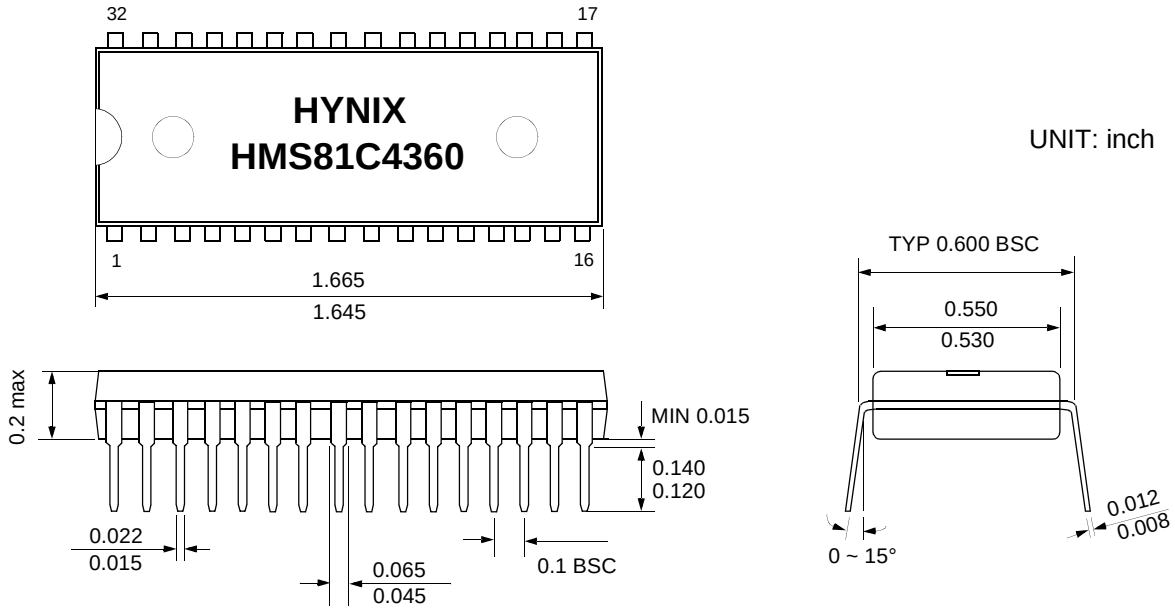


Figure 3-4 32PDIP

4. PACKAGE DIAGRAM



UNIT: mm



UNIT: inch

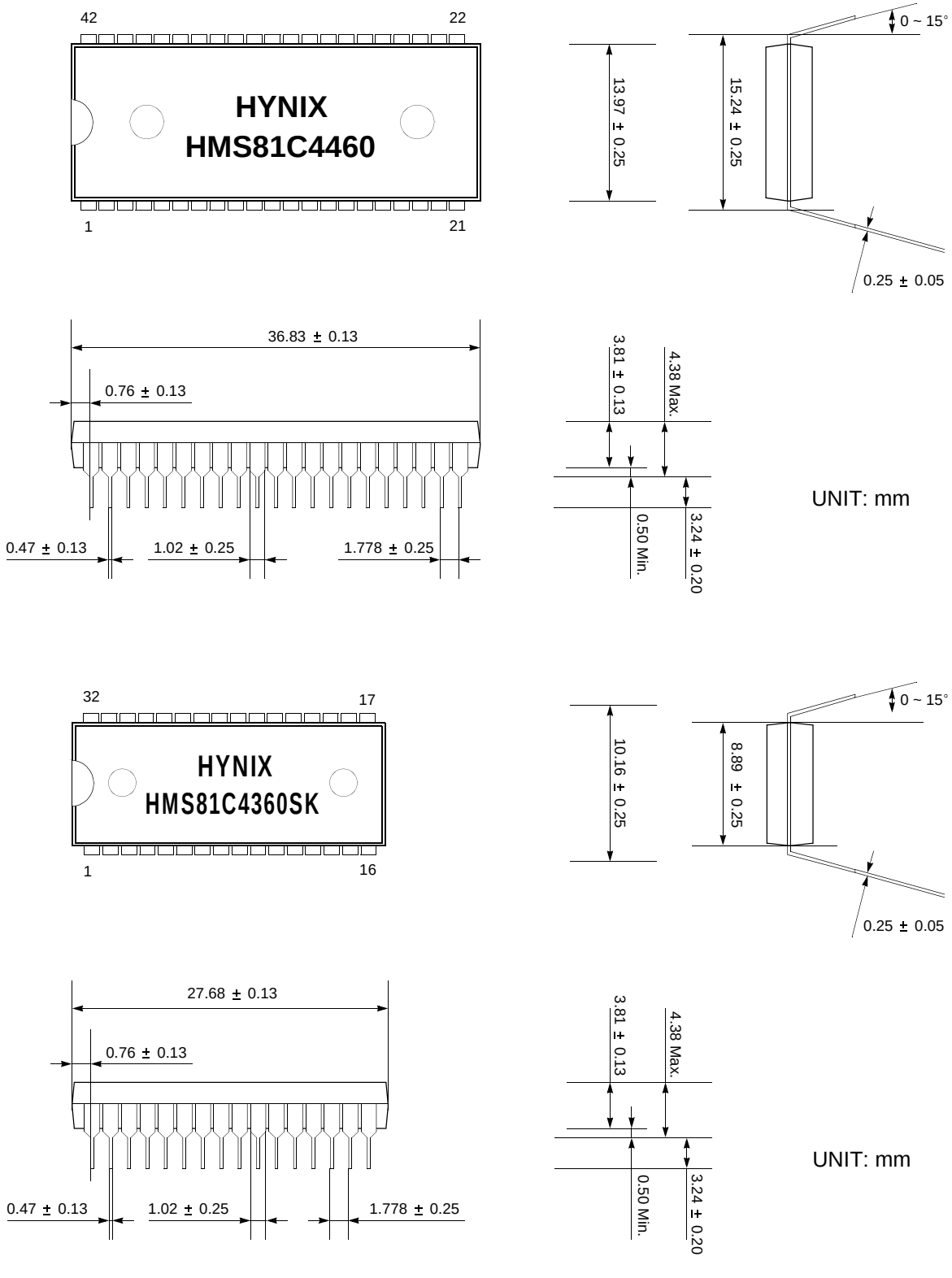


Figure 4-1 Package Diagram

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.

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

R10~R14: R1 is a 5-bit read only port. R1 pins 1 or 0 written to the Port Direction Register can be used as inputs.

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

Port pin	Alternate function
R10	AD0 (A/D converter input 0)
R11	AD1 (A/D converter input 1)
R12	AD2 (A/D converter input 2)
R13	AD3 (A/D converter input 3)
R14	AD4 (A/D converter input 4)

R20~R25: R2 is a 6-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
R21	INT1 (External interrupt input 1)
R22	INT2 (External interrupt input 2)
R23	INT3 (External interrupt input 3)
R24	EC2 (Event counter input 2)
R25	EC3 (Event counter input 3)

R30~R37: R3 is 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.

In addition, R3 serves the functions of the various following special features.

Port pin	Alternate function
R30	PWM0 (Pulse Width Modulation output 0)
R31	PWM1 (Pulse Width Modulation output 1)
R32	PWM2 (Pulse Width Modulation output 2)
R33	PWM3 (Pulse Width Modulation output 3)
R34	PWM4 (Pulse Width Modulation output 4)
R35	PWM5 (Pulse Width Modulation output 5) with 14bit resolution
R36	BUZ (Buzzer output)
R37	TMR1 (Timer Interrupt 1)

R40~R43: R4 is a 4-bit open drain I/O port. Each pins 1 or 0 written to the their 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	SCL0 (I^2C Clock 0)
R41	SDA0 (I^2C Data0)
R42	SCL1 (I^2C Clock 1)
R43	SDA1 (I^2C Data 1)

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

YM,YS: YM,YS are CMOS output port. Each pins controls Background, Edge control.

HS,VS: HS,VS are CMOS input port. Each pins Vertical Sync. input and Horizaltal Sync. inputs.

CVBS: CVBS is a CVBS(Composit Video in) signal input pin.

PIN NAME	Pin No.	In/Out	Function
V_{DD}	9,13,30, 37	-	Supply voltage
V_{SS}	14,29, 36,43	-	Circuit ground

Table 5-1 Port Function Description

PIN NAME	Pin No.	In/Out	Function	
$\overline{TEST}$	44	I	TEST signal input (internal pull up resister)	
$\overline{RESET}$	33	I	Reset signal input	
X_{IN}	35	I	Main oscillation input	
X_{OUT}	34	O	Main oscillation output	
HS	19	I		Horisontal Sync. input
VS	20	I		Vertical Sync. input
R	38	O		Red signal output
G	39	O		Green signal output
B	40	O		Blue signal output
YS	41	O		Edge signal output
YM	42	O		Background signal output
R30/PWM0	52	I/O	PWM functions	8bit PWM (pull up)
R31/PWM1	51	I/O		8bit PWM (pull up)
R32/PWM2	50	I/O		8bit PWM (pull up)
R33/PWM3	49	I/O		8bit PWM (pull up)
R34/PWM4	48	I/O		8bit PWM
R35/PWM5	47	I/O		14bit PWM
R36/BUZ	46	I/O		Buzzer (pull up)
R37/TMR1	45	I/O		Timer Interrupt 1
R40/SCL0	1	I/O	I^2C functions (open drain)	I^2C Serial clock 0
R41/SDA0	2	I/O		I^2C Serial data 0
R42/SCL1	3	I/O		I^2C Serial clock 1
R43/SDA1	4	I/O		I^2C Serial data 1
R20	21	I/O	External interrupt functions	(pull up)
R21/INT1	22	I/O		External interrupt input 1
R22/INT2	23	I/O		External interrupt input 2 (pull up)
R23/INT3	24	I/O		External interrupt input 3
R24/EC2	25	I/O		Event counter input 2
R25/EC3	26	I/O		Event counter input 3 (pull up)
SCAP	11	I		Data slicer comparation reference voltage
R10/AD0	15	I	A/D conversion functions	Analog input 0
R11/AD1	16	I		Analog input 1
R12/AD2	17	I		Analog input 2
R13/AD3	18	I		Analog input 3
R14/AD4	10	I		Analog input 4
CVBS	12	I		Composit video input

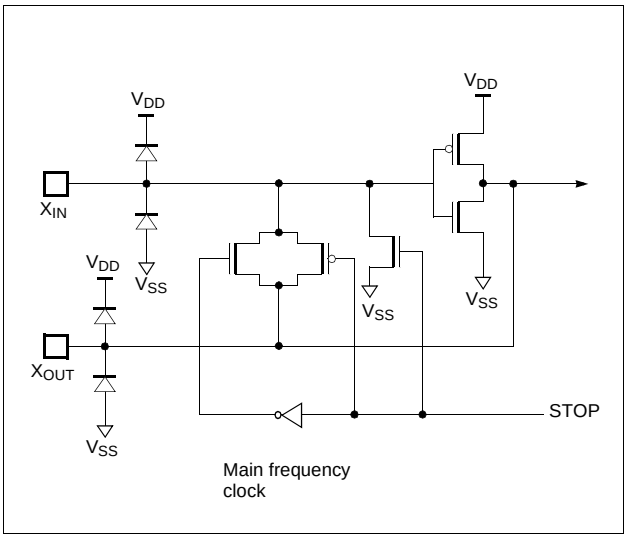
Table 5-1 Port Function Description

PIN NAME	Pin No.	In/Out	Function	
R00	27	I/O	Digital I/O functions	(normal I/O, pull up)
R01	28	I/O		(normal I/O, pull up)
R02	31	I/O		(normal I/O)
R03	32	I/O		(normal I/O, pull up)
R04	5	I/O		(open drain, pull up)
R05	6	I/O		(open drain, pull up)
R06	7	I/O		(open drain, pull up)
R07	8	I/O		(open drain, pull up)

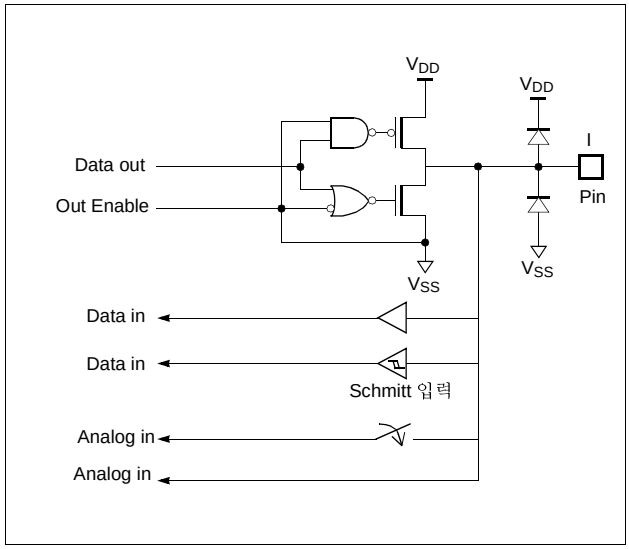
Table 5-1 Port Function Description

6. PORT STRUCTURES

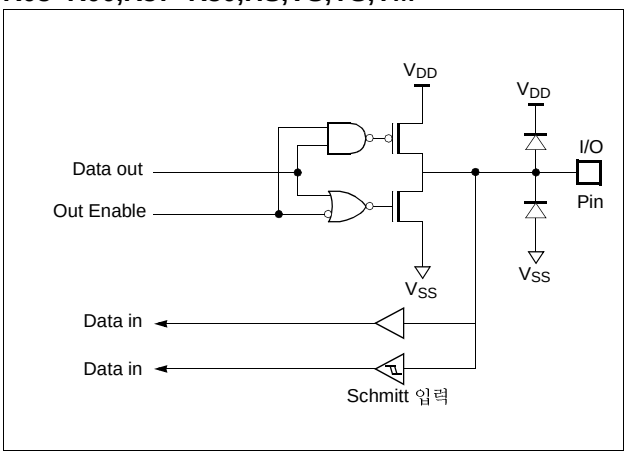
X_{IN}, X_{OUT}



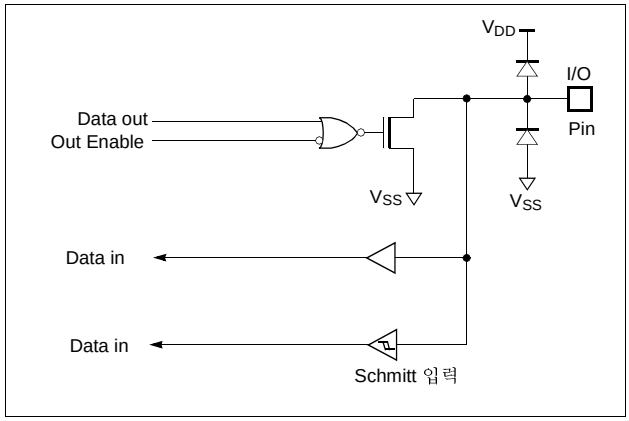
R14~10, CVBS



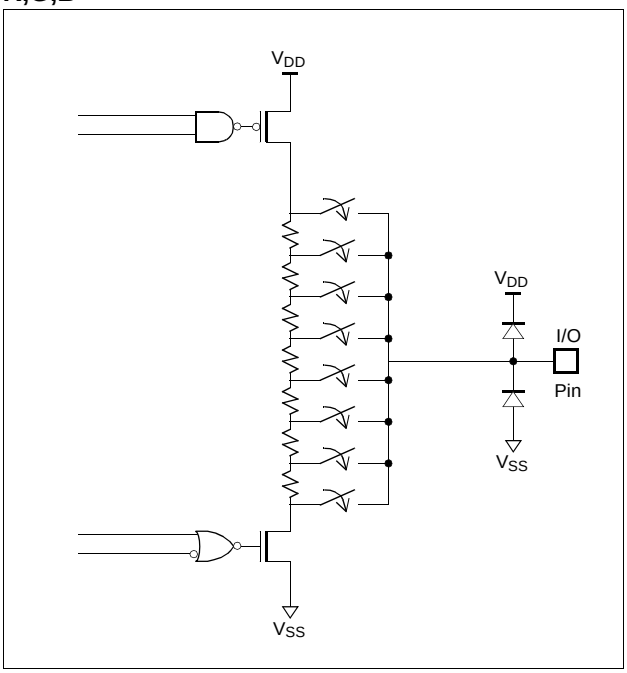
R03~R00,R37~R30,HS,VS,YS,YM



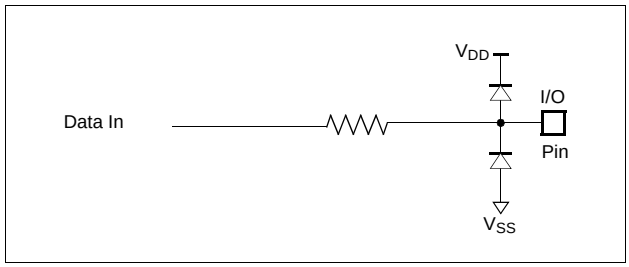
R07~R04, R43~R40, $\overline{\text{TEST}}$



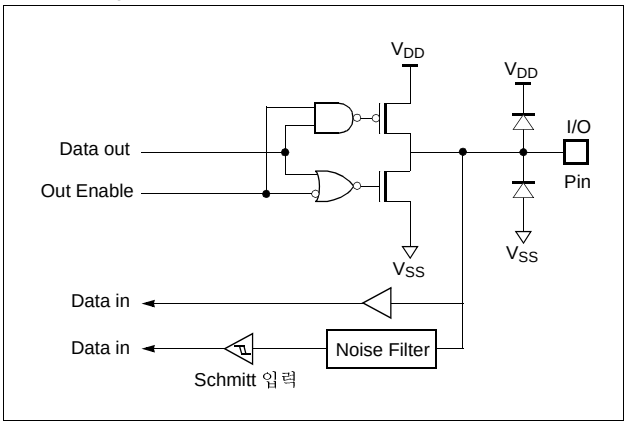
R,G,B



SCAP



R25~R20, $\overline{\text{RESET}}$



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 Vss pin..... 160 mA

Maximum current into V_{DD} pin ..... 160 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}) ..... 100 mA

Maximum current (ΣI_{OH})..... 80 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}	V_{DD}=4.5~5.5V	4.5	5.5	V
Operating Frequency	f_{XIN}	f_{XIN}=4MHz	-	4.0(typical)	MHz
Operating Temperature	T_{OPR}		-10	70	°C

7.3 DC Electrical Characteristics

(T_A=-10~70°C, V_{DD}=4.5~5.5V),

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
High level input voltage	V_{IH}	$\overline{TEST}$, $\overline{RESET}$, Xin, R0, R1, R2, R3, HS, VS	0.8 V_{DD}	-	V_{DD}	V
Low level input voltage	V_{IL}	$\overline{TEST}$, $\overline{RESET}$, Xin, R0, R1, R2, R3,R4 HS, VS	0	-	0.12 V_{DD}	V
High level output voltage	V_{OH}	I_{OH} = -5mA R0, R1, R2, R3, YS, YM	V_{DD} - 1	-	-	V
Low level output voltage	V_{OL}	I_{OL} = 5mA R0, R1, R2, R4	-	-	1.0	v
Supply current in ACTIVE mode	I_{DD}	V_{DD}	-	40	80	mA
pull-up lekage current	I_{RUP}	V_{DD} = 5.5v, V_{PIN} = 0.4V $\overline{TEST}$, R00, R01, R03, R04, R05, R06, R07, R20, R22, R25, R30, R31, R32, R33 R36	-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}	-5	-	5	μA

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Low input leakage current	I_{IZL}	V_{DD} = 5.5V, V_{PIN} = 0V All input, I/O pins except X_{IN}, OSC1	-5	-	5	μA
RAM data retention voltage	V_{RAM}	V_{DD}	1.2	-	-	V
Hysterisis	Vt+ ~ Vt-	$\overline{TEST}$, $\overline{RESET}$, Xin, HS, VS, R07 ~ R00, R21, R23, R24, R25, R37 ~ R30	1.0	-	-	V
Comparator operating range	V_{rCVBS}	V_{DD} = 5V CVBS pin	1.2	-	3.5	V
Comparator resolution	V_{aCVBS}	V_{DD} = 5V CVBS pin	-	-	0.08	V
RGB DAC Resolution 1	RGB_{R1}	V_{DD} = 5V No in/out current in R,G,B pin	-	-	5	%
RGB DAC Output voltage	V_{RGB}	RGB DAC On No in/out current in R,G,B pin				V
		Level 0		$3/40V_{dd}$		
		Level 1		$5/40V_{dd}$		
		Level 2		$8/40V_{dd}$		
		Level 3		$12/40V_{dd}$		
		Level 4		$17/40V_{dd}$		
		Level 5		$23/40V_{dd}$		
		Level 6		$30/40V_{dd}$		
		Level 7		$38/40V_{dd}$		
RGB V_{oh}	V_{ohrgb}	V_{DD} = 5V RGB DAC On Level 7 I_{OH} = -3mA	3.1	3.5	3.9	V
RGB V_{ol}	V_{olrgb}	V_{DD} = 5V RGB DAC On Level 0 I_{OL} = 3mA	0.4	0.6	0.8	V

7.4 AC Characteristics

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

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Crystal oscillator Frequency	f_{XIN}	X_{IN}	3	4	5	MHz
External Clock Pulse Width	t_{MCPW}	X_{IN}	180	-	350	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

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Oscillation Stabilizing Time	t_{ST}	X_{IN}, X_{OUT}	-	-	20	mS
Interrupt Pulse Width	t_{IW}	INT1~3	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 $1/f_{XIN}$ main clock operation mode,

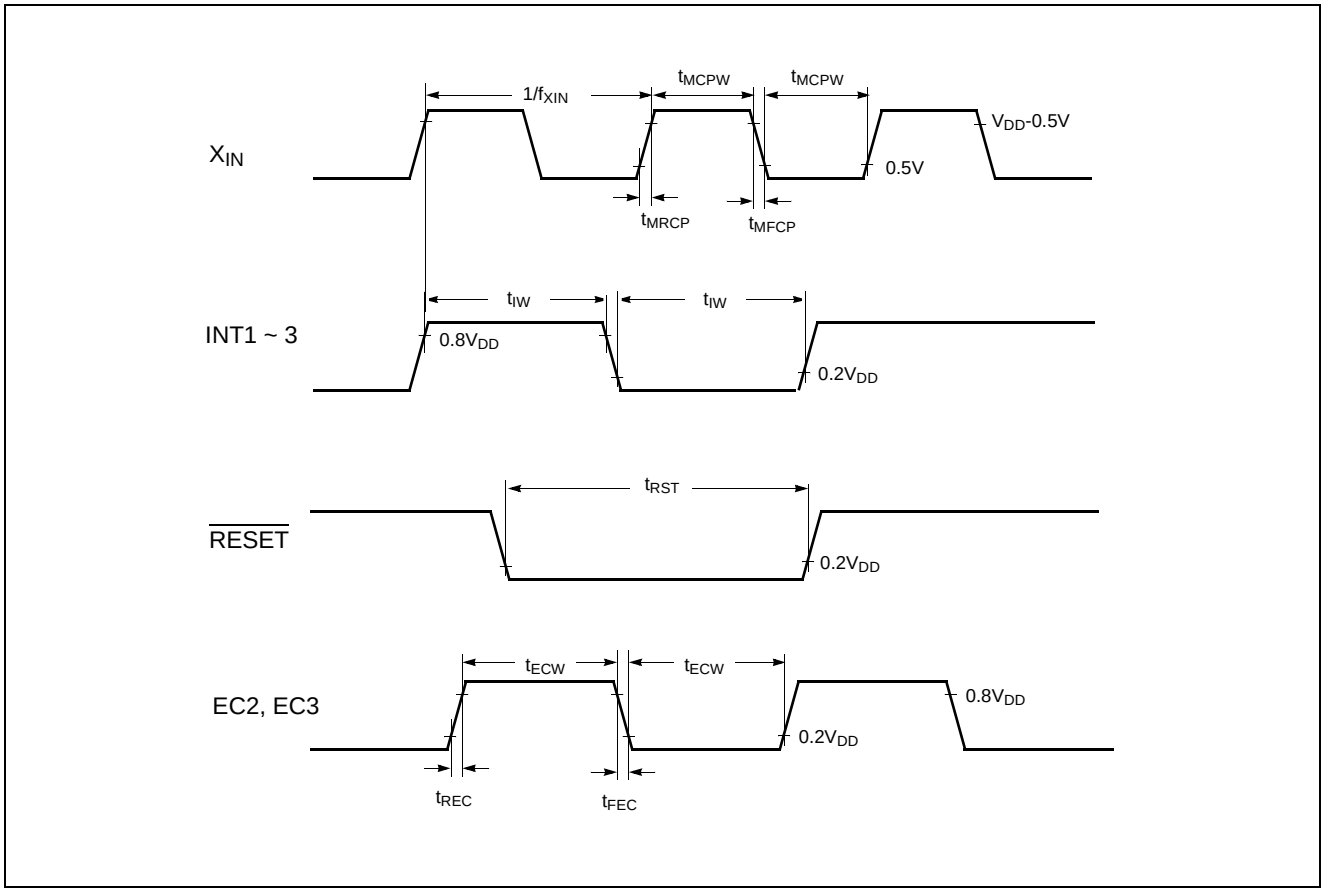


Figure 7-1 Timing Chart

7.5 A/D Converter Characteristics

(TA=25°C, V_{DD}=5V, V_{SS}=0V)

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AN}	-	V_{SS}-0.3	-	V_{DD}+0.3	V
Overall Accuracy	CAIN	-	-	±1.5	±2.5	LSB
Non Linearity Error	NNLE	-	-	±1.5	±2.5	
Differential Non Linearity Error	NDNLE	-	-	±1.5	±2.5	
Zero Offset Error	NZOE	-	-	±0.5	±2.0	
Full Scale Error	NFSE	-	-	±0.75	±1.0	
Gain Error	NGE	-	-	±1.5	±2.0	
Conversion Time	TCONV	f_{MAIN}=4MHz	-	-	15	μS

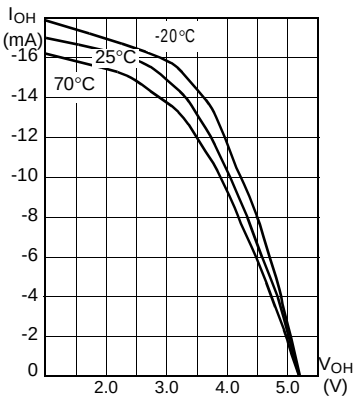
7.6 Typical Characteristics

These graphs and tables are for design guidance only and are not tested or guaranteed.

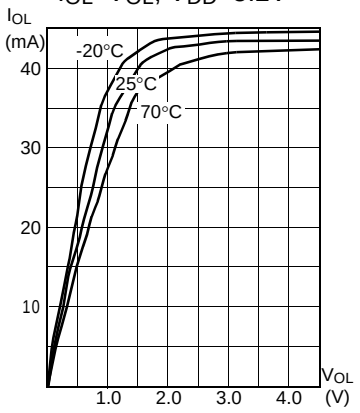
In some graphs or tables, the datas presented are outside specified operating range (e.g. outside specified V_{DD} range). This is for information only and devices are guaranteed to operate properly only within the specified range.

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

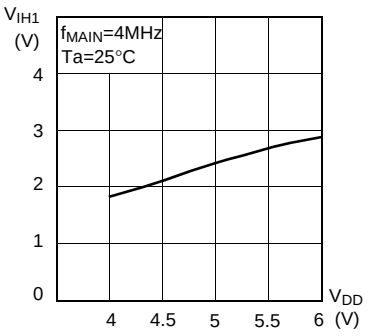
I_{OH}–V_{OH}, V_{DD}=5.2V



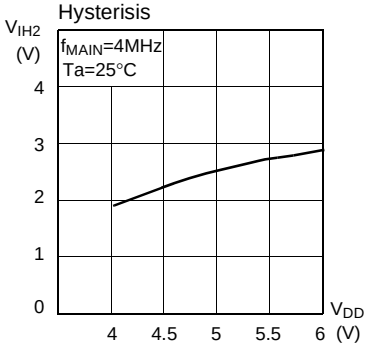
I_{OL}–V_{OL}, V_{DD}=5.2V



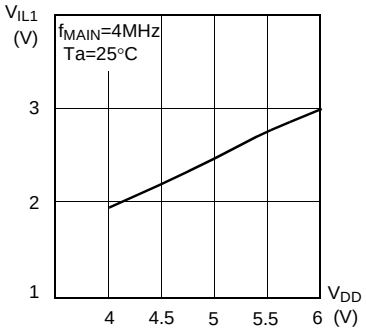
V_{DD}–V_{IH}



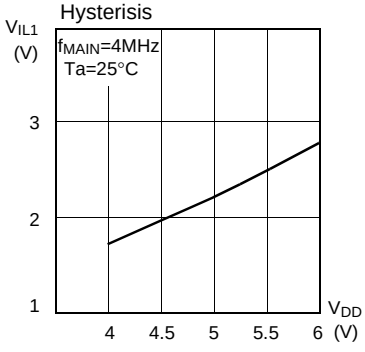
V_{DD}–V_{IH}
Hysterisis



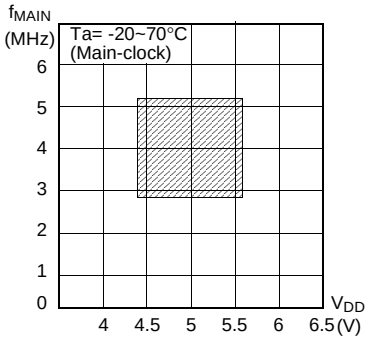
V_{DD}–V_{IL}



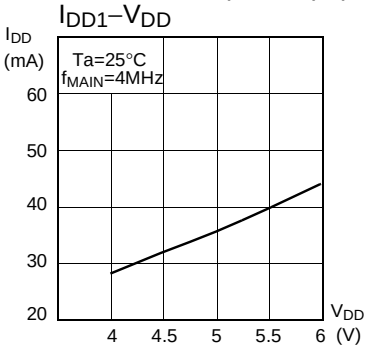
V_{DD}–V_{IL}
Hysterisis



Operating Area



Normal Mode (Main opr.)
I_{DD1}–V_{DD}



8. MEMORY ORGANIZATION

The GMS81C4x60 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 60K bytes of Program memory. Data memory can be read and written to up to 1024 bytes including the stack area. Font memory has prepared 32K 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.

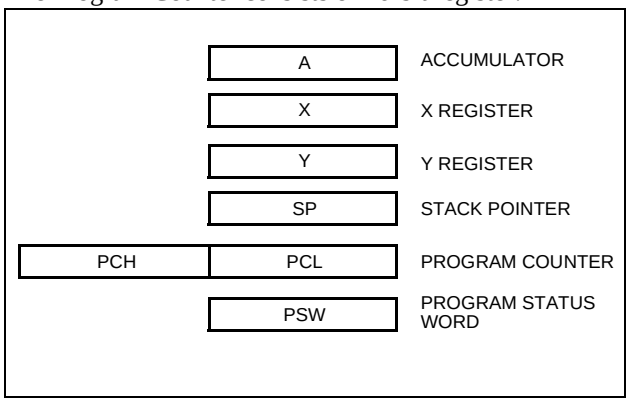


Figure 8-1 Configuration of Registers

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.

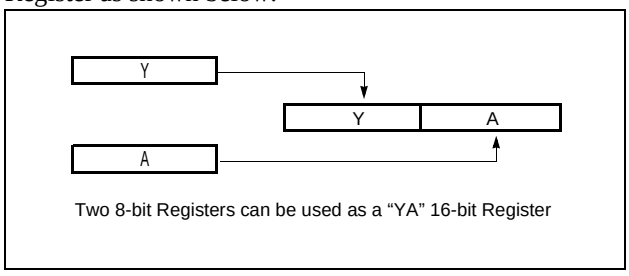


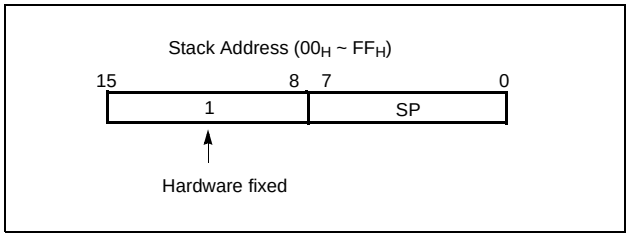
Figure 8-2 Configuration of YA 16-bit Register

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 00_H to FF_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

```
LDX    #0FFH
TXSP                ; SP ← FFH
```

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

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

[Carry flag C]

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

[Zero flag Z]

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

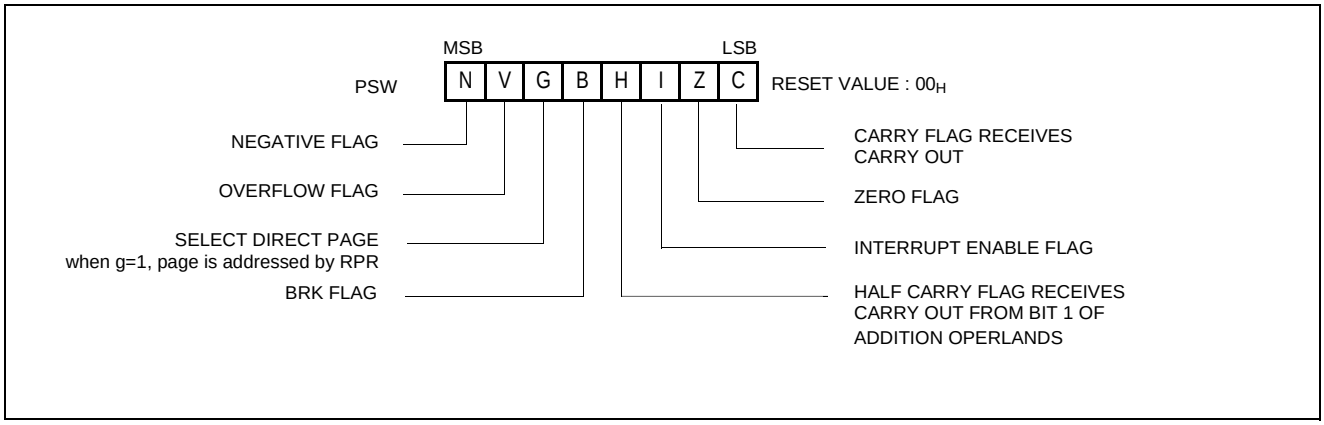


Figure 8-3 PSW (Program Status Word) Register

[Interrupt disable flag I]

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

[Half carry flag H]

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

[Break flag B]

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

[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 RPR register (address $0F3_H$). 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.

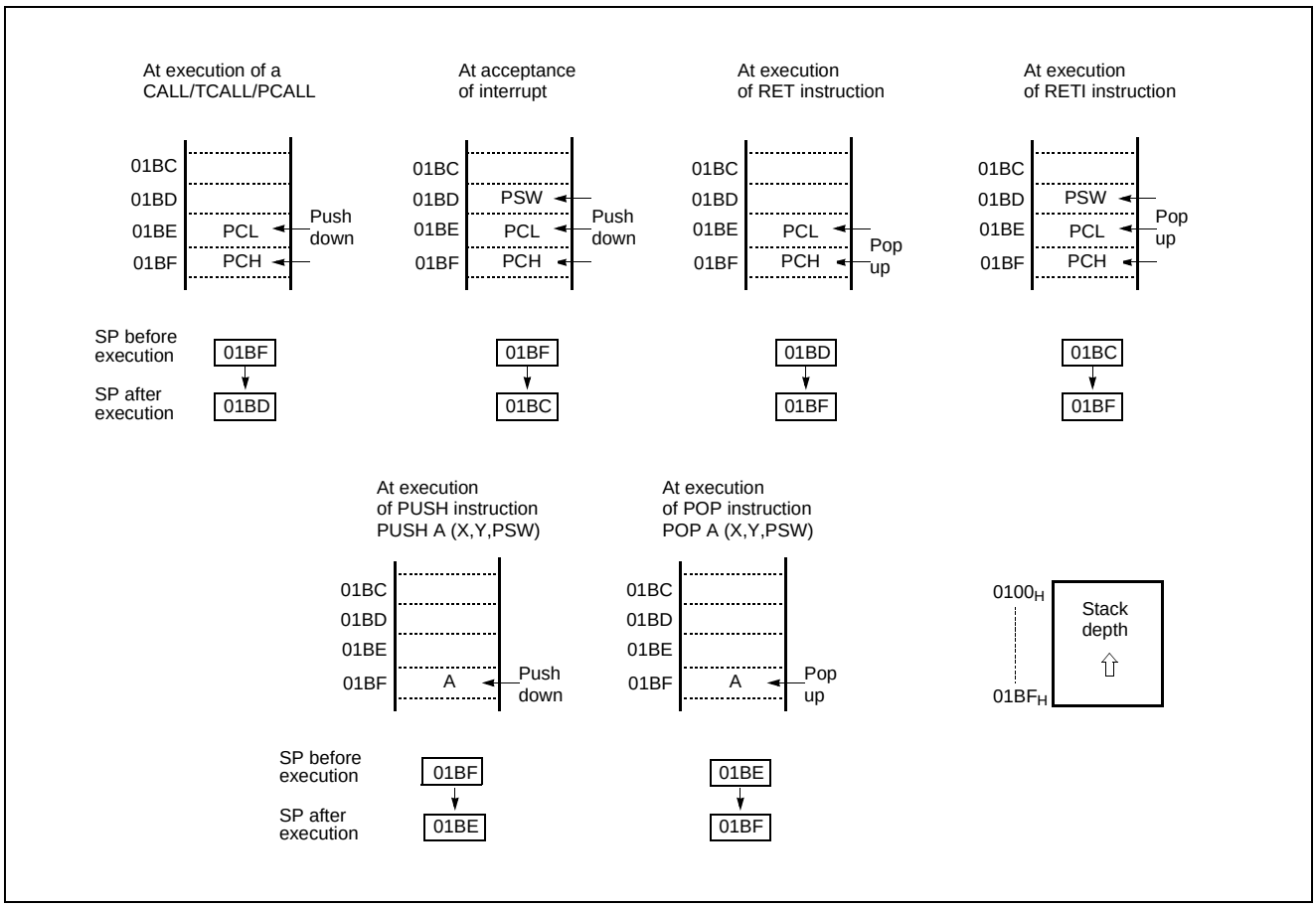


Figure 8-4 Stack Operation

8.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but this device has 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.

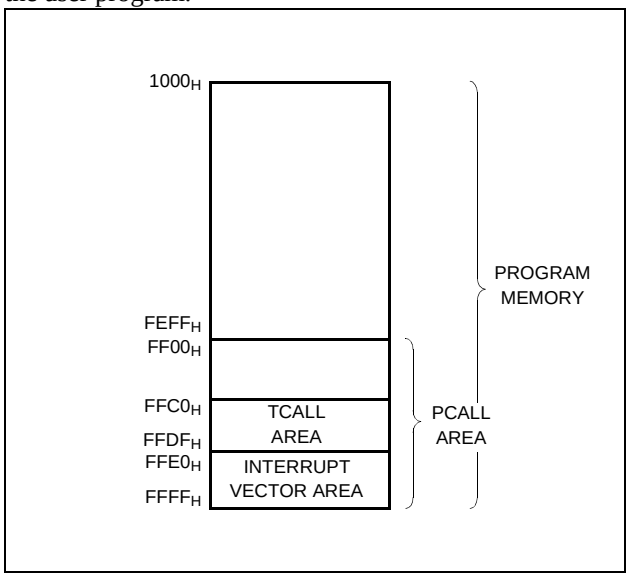


Figure 8-5 Program Memory Map

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  15          ;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 1, for example, is assigned to location $0FFF8_H$. The interrupt service locations spaces 2-byte interval: $0FFF6_H$ and $0FFF7_H$ for External Interrupt 2, $0FFE8_H$ and $0FFE9_H$ for External Interrupt 3, 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
E2	-
E4	Basic Interval Timer Interrupt Vector
E6	Watchdog Timer Interrupt Vector
E8	External Interrupt 3/4 Vector
EA	Timer/Counter 3 Interrupt Vector
EC	Timer/Counter 1 Interrupt Vector
EE	V-Sync Interrupt Vector
F0	Slicer Interrupt Vector
F2	Timer/Counter 2 Interrupt Vector
F4	Timer/Counter 0 Interrupt Vector
F6	External Interrupt 2 Vector
F8	External Interrupt 1 Vector
FA	On Screen Display Interrupt Vector
FC	-
FE	RESET Vector

NOTE:
"-" means reserved area.

Figure 8-6 Interrupt Vector Area

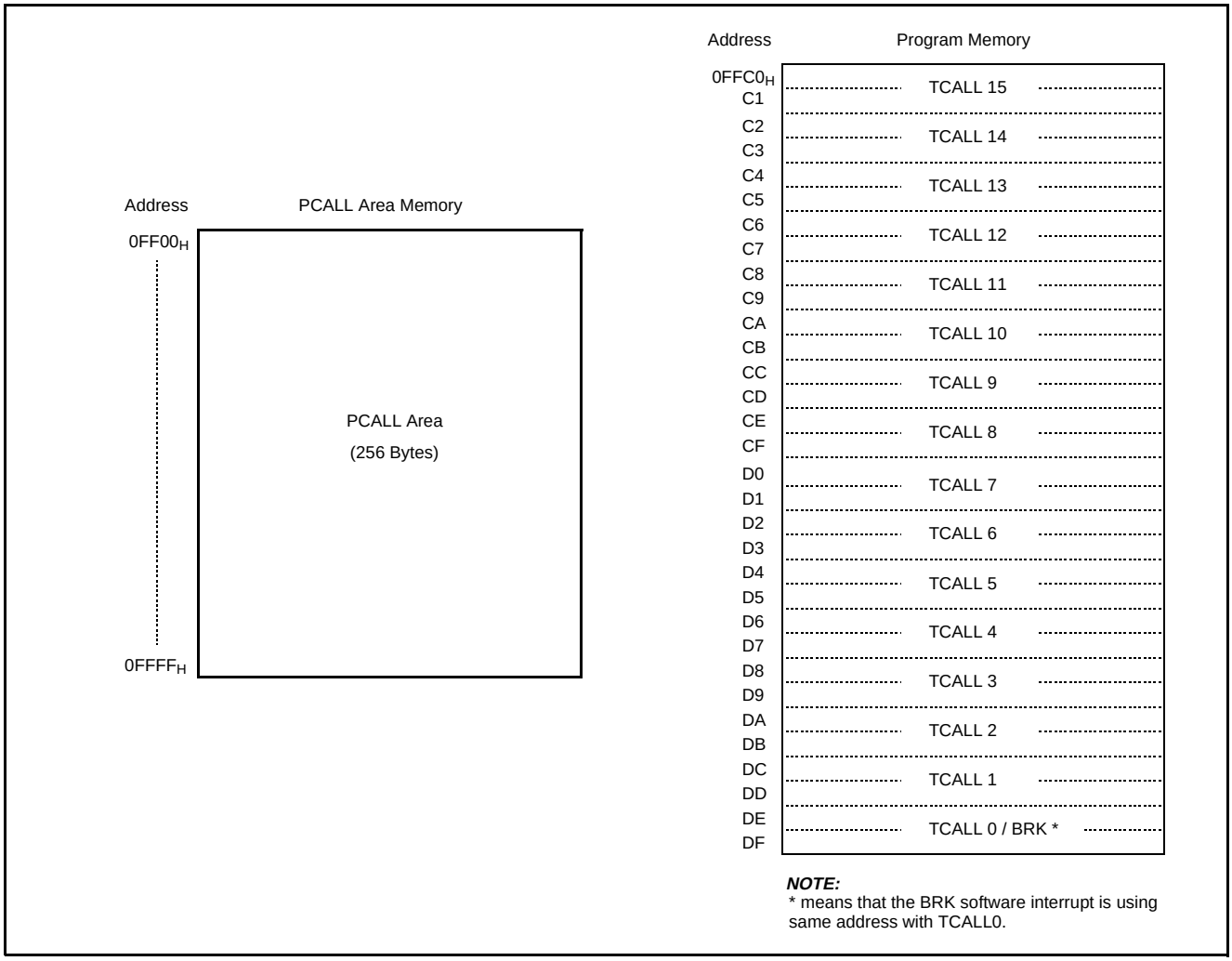
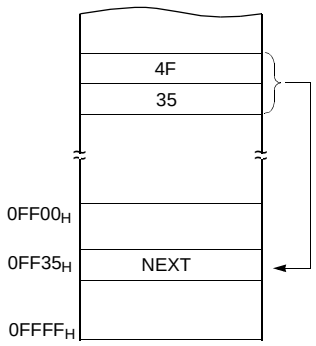


Figure 8-7 PCALL and TCALL Memory Area

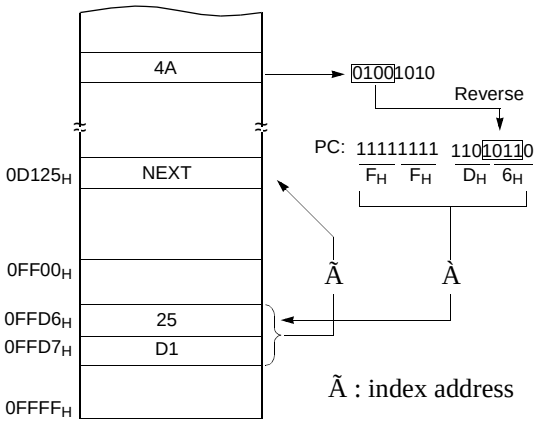
PCALL→ rel

4F35 PCALL 35_H



TCALL→ n

4A TCALL 4



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

```
           ORG     0FFE0H

           DW      I2C_INT
           DW      NOT_USED
           DW      BIT_INT
           DW      WDT_INT
           DW      IR_INT
           DW      TIMER3
           DW      TIMER1
           DW      VSYNC_INT
           DW      SLICE_INT
           DW      T2_INT
           DW      T0_INT
           DW      EXT2_INT
           DW      EXT1_INT
           DW      OSD_INT
           DW      NOT_USED
           DW      RESET

           ORG     0F000H

;*****************************************
;              MAIN     PROGRAM            *
;*****************************************
;
RESET:     DI                              ;Disable All Interrupts
           CLRG
           LDX     #0
RAM_CLR:   LDA     #0                      ;RAM Clear(!0000H->!00BFH)
           STA     {X}+
           CMPX    #0C0H
           BNE     RAM_CLR
;
           LDX     #0FFH                   ;Stack Pointer Initialize
           TXSP
;
           LDM     PLLC,#0000_0101b        ;16MHz system clock
;
           LDM     R0, #0FFh               ;Normal Port 0
           LDM     R0DIR,#0FFh             ;Normal Port Direction
           :
           :
           LDM     TM0,#0000_0000B         ;timer stop
           :
           :
           CALL    VRAM_CLR                ;Clear VRAM
           :
           :
```

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.

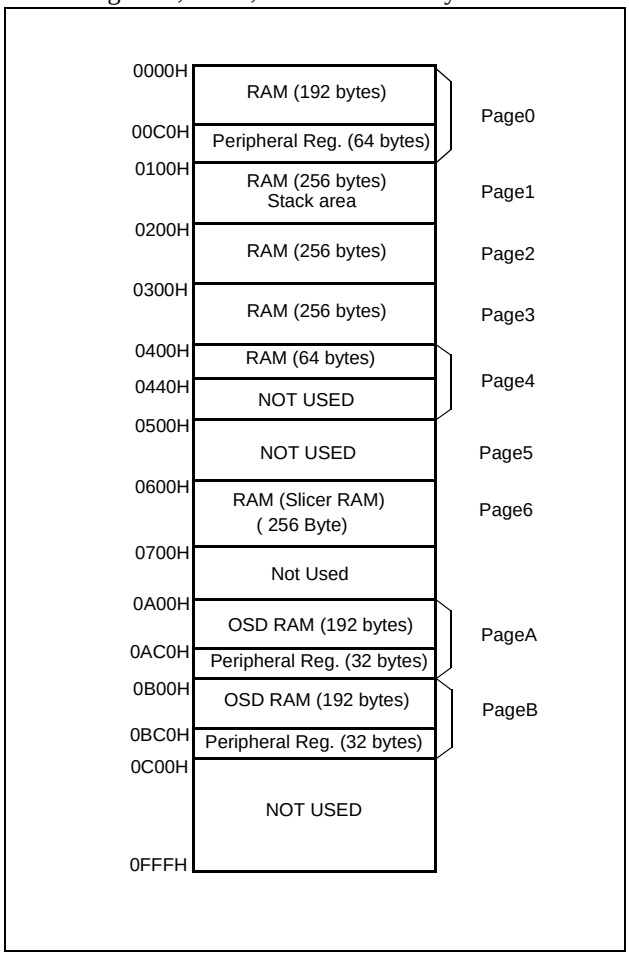


Figure 8-8 Data Memory Map

User Memory

The GMS81C4x60 has $1,024 \times 8$ bits for the user memory (RAM) except Peripheral Reg. (64 bytes) .

Control Registers

The control registers are used by the CPU and Peripheral function blocks for controlling the desired operation of the device. Therefore these registers contain control and status bits for the interrupt system, the timer/ counters, analog to digital converters and I/O ports. The control registers are in address range of $0C0_H$ to $0FF_H$.

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

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

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

Example; To write at CKCTLR

```
LDM    CKCTLR,#05H ;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 22.

Address	Symbol	R/W	Reset Value	Addressing mode
00C0H	R0	R/W	????????	byte, bit[1]
00C1H	R0DD	W	00000000	byte[2]
00C2H	R1	R	????????	byte, bit
00C3H	R1DD	W	---00000	byte
00C4H	R2	R/W	????????	byte, bit
00C5H	R2DD	W	--000000	byte
00C6H	R3	R/W	????????	byte, bit
00C7H	R3DD	W	00000000	byte
00C8H	R4	R/W	????????	byte, bit
00C9H	R4DD	W	----0000	byte
00CAH	reserved	-	-	-
00CBH	reserved	-	-	-
00CCH	reserved	-	-	-
00CDH	reserved	-	-	-
00CEH	FUNC	W	0000000-	byte
00CFH	PLLC	W	-0000000	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
0D6H	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	ICCR	R/W	00000000	byte, bit
0DCH	reserved	-	-	-
0DEH	reserved	-	-	-
0DFH	reserved	-	-	-
0E0H	PWMR0	W	????????	byte
0E1H	PWMR1	W	????????	byte
0E2H	PWMR2	W	????????	byte
0E3H	PWMR3	W	????????	byte
0E4H	PWMR4	W	????????	byte
0E5H	PWMR5H	R/W	????????	byte
0E6H	PWMR5L	R/W	--??????	byte, bit
0E7H	reserved	-	-	-
0E8H	reserved	-	-	-
0E9H	reserved	-	-	-
0EAH	PWMCR1	R/W	00000000	byte, bit
0EBH	PWMCR2	R/W	-----000	byte, bit
0ECH	reserved	-	-	-
0EDH	reserved	-	-	-
0EEH	reserved	-	-	-
0EFH	AIPS	W	--000000	byte
0F0H	ADCM	R/W	????????	byte, bit
0F1H	ADR	R	????????	byte
0F2H	IEDS	W	--000000	byte
0F3H	IMOD	R/W	--000000	byte, bit
0F4H	IENL	R/W	00000000	byte, bit
0F5H	IRQL	R/W	00000000	byte, bit
0F6H	IENH	R/W	00000000	byte, bit
0F7H	IRQH	R/W	00000000	byte, bit
0F8H	reversed	-	-	-
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
0FEH	reserved	-	-	-
0FFH	reserved	-	-	-

Table 8-1Control registers

0AD0	RED0	W	????????	byte, bit-
0AD1	RED1	W	????????	byte, bit
0AD2	RED2	W	????????	byte, bit
0AD3	GREEN0	W	????????	byte, bit
0AD4	GREEN1	W	????????	byte, bit
0AD5	GREEN2	W	????????	byte, bit
0AD6	BLUE0	W	????????	byte, bit
0AD7	BLUE1	W	????????	byte, bit
0AD8	BLUE2	W	????????	byte, bit
0AD9	reserved	-	-	-
0ADA	reserved	-	-	-
0ADB	reserved	-	-	-
0ADC	reserved	-	-	-
0ADD	reserved	-	-	-
0ADE	reserved	-	-	-
0ADF	reserved	-	-	-
0AE0H	OSDCON1	R/W	00000000	byte, bit
0AE1H	OSDCON2	R/W	00000000	byte, bit
0AE2H	OSDCON3	W	00000000	byte, bit
0AE3H	FDWSET	W	01111010	byte
0AE4H	EDGECOL	W	10000111	byte
0AE5H	CHEDCL	W	????????	byte
0AE6H	OSDLN	R	---00000	byte
0AE7H	LHPOS	W	????????	byte
0AE8H	DLLMOD	W	00000000	byte
0AE9H	DLLTST	R	--000000	byte
0AEAH	L1ATTR	W	??????-?	byte, bit
0AEBH	L1EATR	W	---?????	byte, bit
0AECH	L1VPOS	W	????????	byte
0AEDH	L2ATTR	W	????????	byte, bit
0AEEH	L2EATR	W	---?????	byte, bit
0AEFH	L2VPOS	W	????????	byte, bit
0AF0H	WINSH	W	????????	byte
0AF1H	WINSY	W	????????	byte
0AF2H	WINEH	W	????????	byte
0AF3H	WINEY	W	????????	byte
0AF4H	VCNT	R	????????	byte
0AF5H	HCNT	R	????????	byte
0AF9H	CULTAD	W	????????	byte
0BE0H	SLCON	R/W	00000000	byte, bit
0BE1H	SLINF0	W	00000000	byte, bit
0BE2H	SLINF1	W	00000000	byte, bit
0BE3H	RIKST	W	????????	byte
0BE4H	RIKED	W	????????	byte
0BE7H	SNCST	W	????????	byte
0BE8H	SNCED	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 GMS81C4x60 uses six addressing modes;

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

(1) Register Addressing

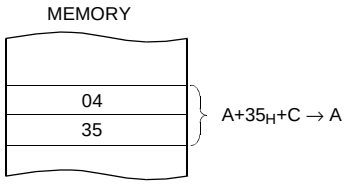
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

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

Example:

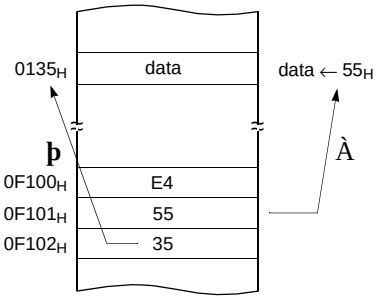
```
FE0435   ADC    #35H
```



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

Example: G=1, RPR=01_H

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

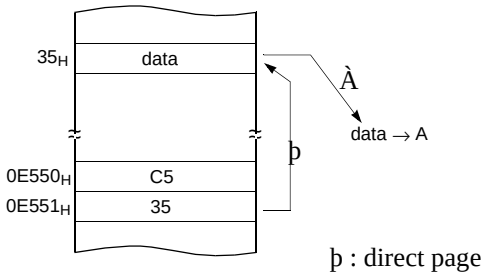


(3) Direct Page Addressing → dp

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

Example; G=0

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



þ : direct page

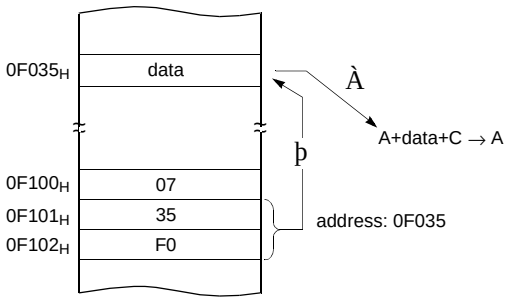
(4) Absolute Addressing → !abs

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;

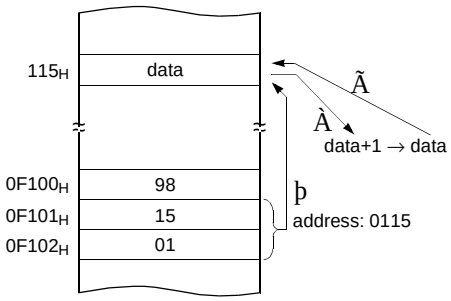
```
F100: 0735F0  ADC   !0F035H  ;A ←ROM[0F035H]
```



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 RPR.

```
F100: 981501  INC  !0115H  ;A ←ROM[115H]
```



(5) Indexed Addressing

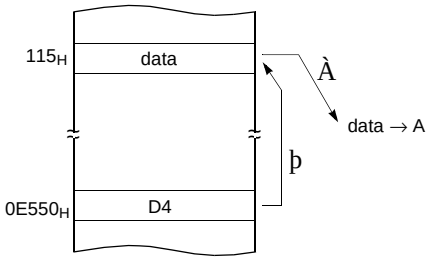
X indexed direct page (no offset) → {X}

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, $RPR=01_H$

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



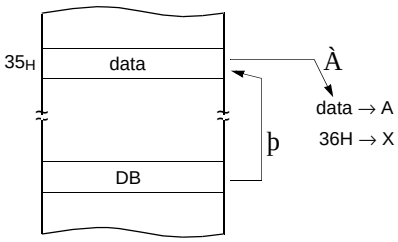
X indexed direct page, auto increment→ {X}+

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}+
```



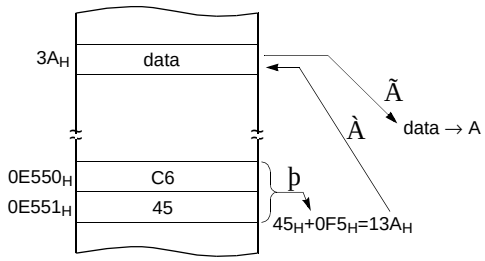
X indexed direct page (8 bit offset) → dp+X

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.

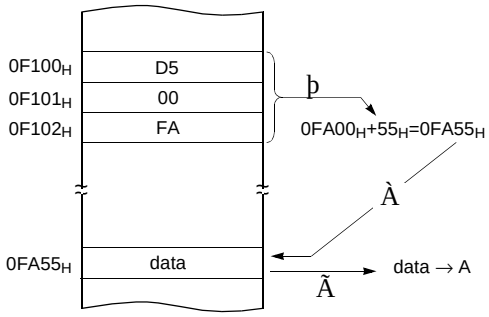
This is same with above (2). Use Y register instead of X.

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

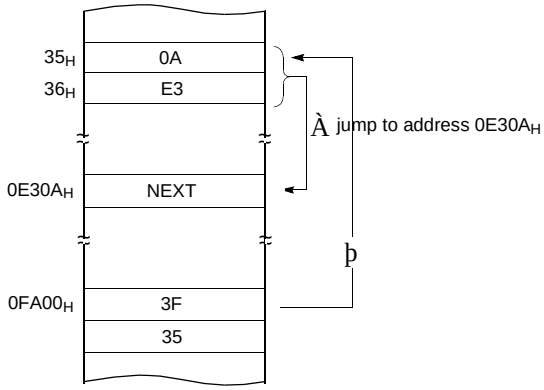
Direct page indirect → [dp]

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]
```



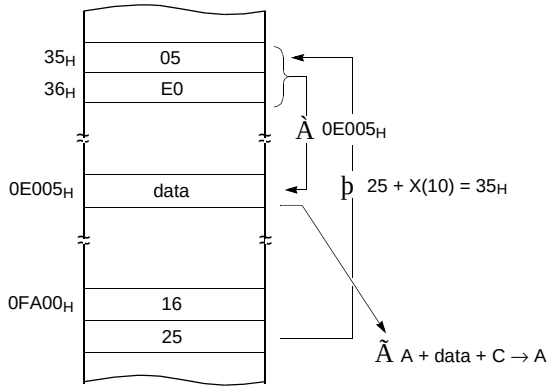
X indexed indirect → [dp+X]

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]
```



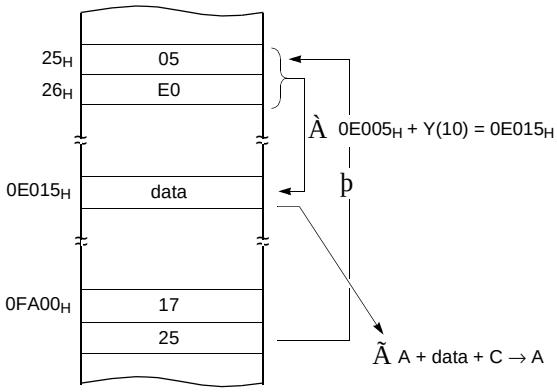
Y indexed indirect → [dp]+Y

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

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

Example; G=0, Y=10_H

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



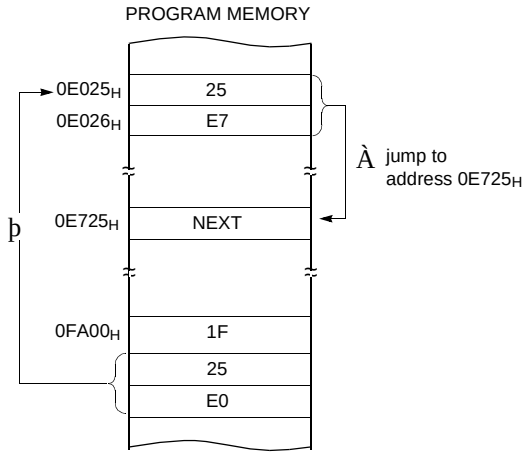
Absolute indirect → [!abs]

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

JMP

Example; G=0

```
FA00: 1F25E0  JMP  [!0E025H]
```



9. I/O PORTS

The HMS81C4x60 has 5 ports (R0, R1, R2, R3 and R4) and OSD ports (R,G,B,YS,YM). These ports pins may be multiplexed with an alternatefunction for the peripheral features on the device. In general, in an 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 (R0, R1, R2, R3, R4) are represented as a D-Type 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 HMS81C4x60 have been written to zero by reset function. On the other hand, its initial status is input.

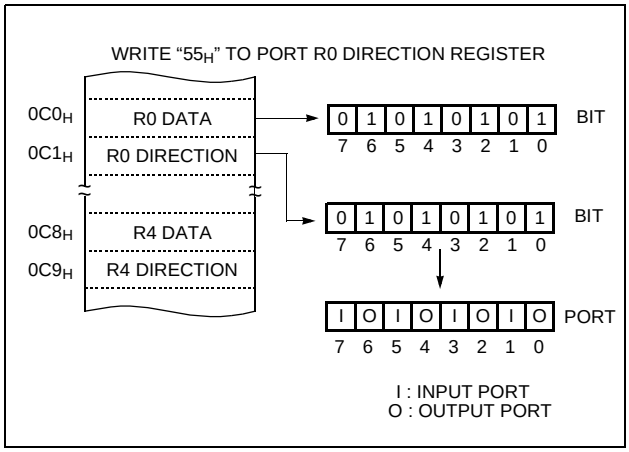


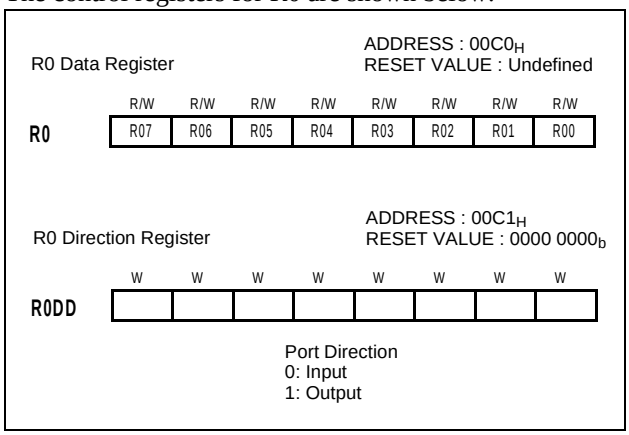
Figure 9-1 Example of port I/O assignment

9.2 I/O Ports Configuration

R0 Ports

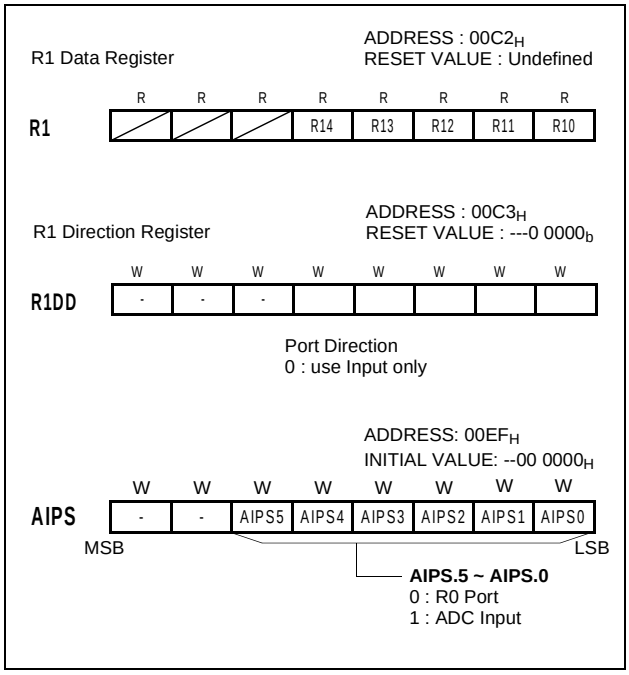
R07 ~ R04 is an open drain bidirectional I/O port and R03 ~ R00 is a CMOS bidirectional I/O port(address $0C0_H$). Each I/O pin can independently used as an input or an output through the R0DD register (address $0C1_H$).

The control registers for R0 are shown below.



R1 Ports

R1 is a 5-bit CMOS input port only(address $0C2_H$). Each pin can independently used as an input through the R1DD register (address $0C3_H$). User can use R0DD register when its bit is 0 only. The control registers for R1 are shown below.



R1 port also can use the value bit5 ~ bit0 of AIPS register to secondary function register. 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)

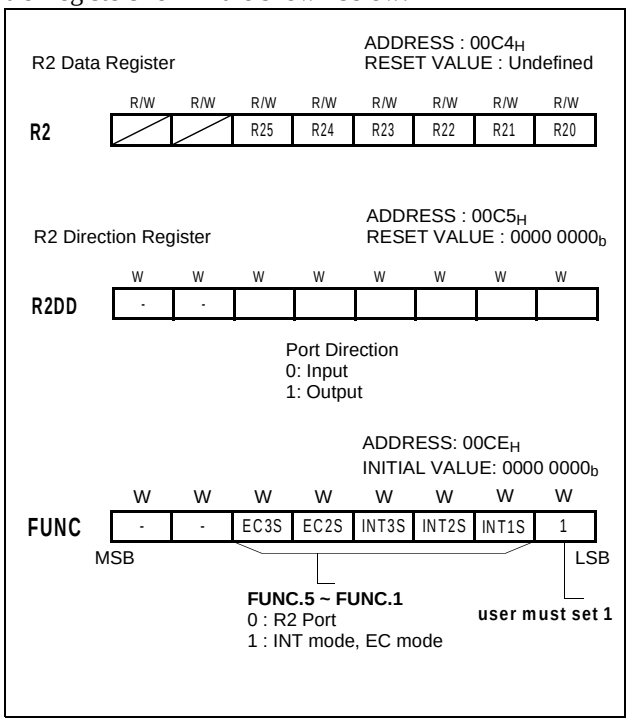
Port R1 is multiplexed with various special features.The control registers controls the selection of alternate function. After reset, this value is “0”, port may be used as normal input port. The way to select alternate function such as comparator input will be shown in each peripheral section.

In addition, R1 port is used as key scan function which operate with normal input port.

Input or output is configured automatically by each function register (KSMR) regardless of R1DD.

R2 Port

R2 is a 6-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.



R2 port also use the value bit5 ~ bit1 of FUNC register to secondary function register. R2 port have secondary func-

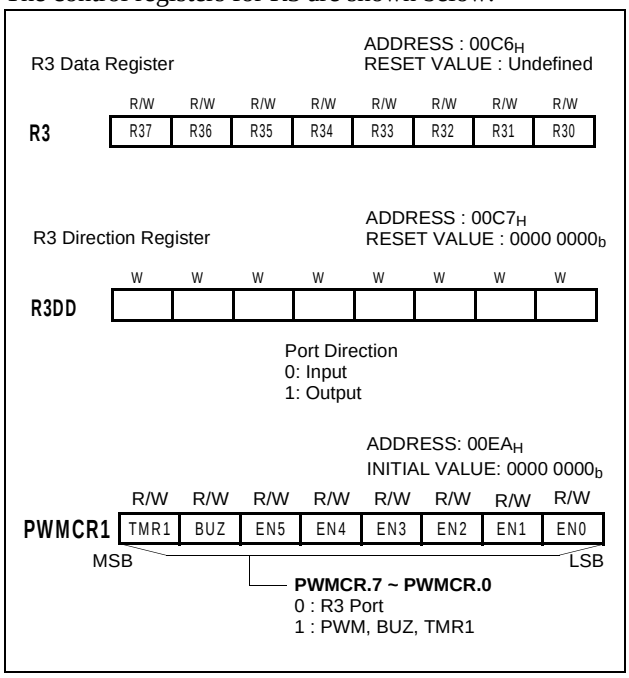
tions as following table.

Port Pin	Alternate Function
R21	INT1 (External Interrupt 1)
R22	INT2 (External Interrupt 2)
R23	INT3 (External Interrupt 3)
R24	EC2 (Event Counter 2)
R25	EC3 (Event Counter 3)

R3 Port

R3 is a 8-bit CMOS bidirectional output port (address $0C6_H$). Each I/O pin can independently used as an input or an output through the R3DD register (address $0C7_H$).

The control registers for R3 are shown below.



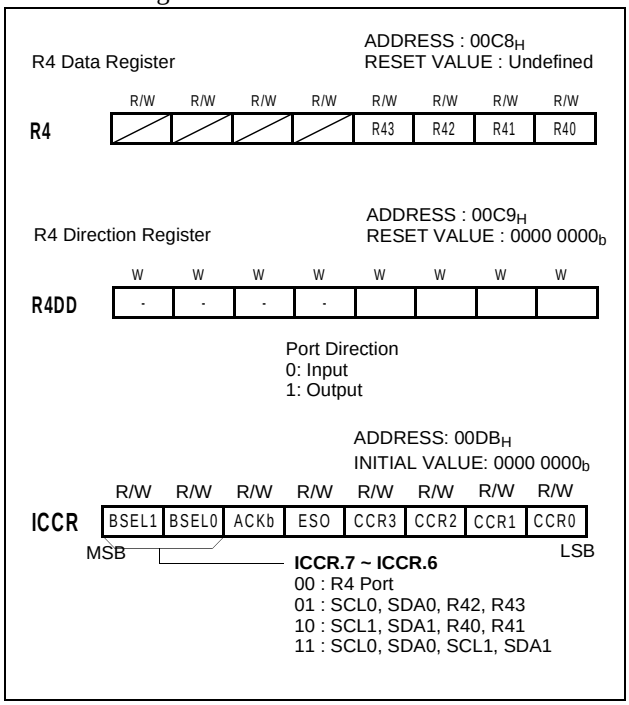
R3 port also use the value bit7 ~ bit0 of PWMCR1 register to secondary function register. R3 port have secondary functions as following table.

R30	PWM0 (Pulse Width Modulation 0)
R31	PWM1 (Pulse Width Modulation 1)
R32	PWM2 (Pulse Width Modulation 2)
R33	PWM3 (Pulse Width Modulation 3)
R34	PWM4 (Pulse Width Modulation 4)
R35	PWM5 (Pulse Width Modulation 5 - 14bit)
R36	BUZ (Buzzer Output)
R37	TMR1 (Timer Interrup 1)

R4 Port

R4 is a 4-bit open drain and 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.



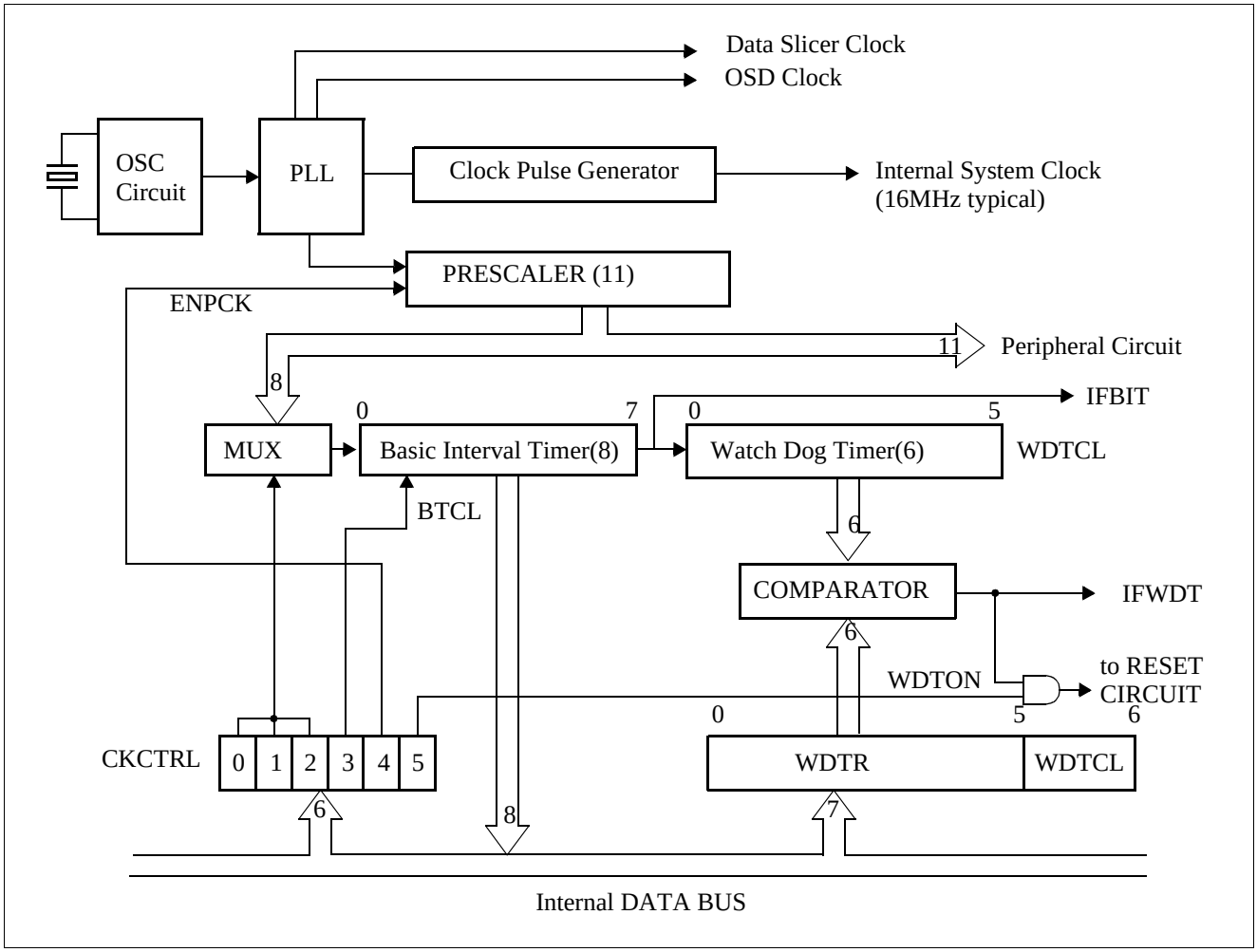
R4 port also use the value bit7 ~ bit6 of ICCR register to secondary function register. R4 port have secondary functions as following table.

R40	SCL0 (Serial Clock 0)
R41	SDA0 (Serial Data 0)
R42	SCL1 (Serial Clock 1)
R43	SDA1 (Serial Data 1)

10. CLOCK GENERATOR

As shown in Figure 10-1, the clock generation Circuit consist PLL that generate multiplicated frequency of Crystal clock, Generation Circuit which create CPU clock, Prescaler which generate input clock of Basic Interval Timer and variable hardware clock, Basic Interval timer which is generate standard time, Watch Dog Timer which is protect Software Overflow.

See “12.1 BASIC INTERVAL TIMER” on page for details.



10.1 Clock Generation Circuit

The clock signal come from crystal oscillator or ceramic via Xin and Xout or from external clock via Xin is supplied to Clock Pulse Generator and Prescaler.

Internal System Clock for CPU is made by Clock Pulse Generator, and several peripherial clock is divided by prescaler.

Clock Generation circuit of Crystal Oscillator or Ceramic Resonator is shown as below.

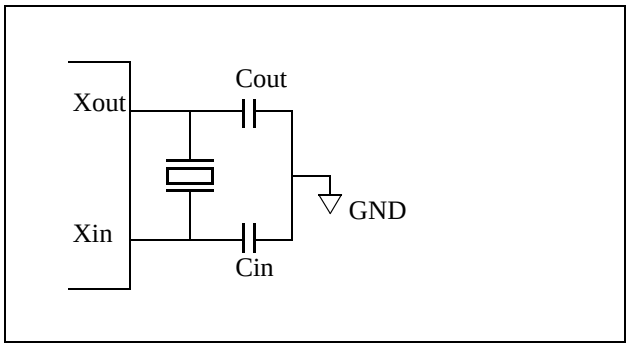


Figure 10-1 Cristal Oscillator or Ceramic Resonator

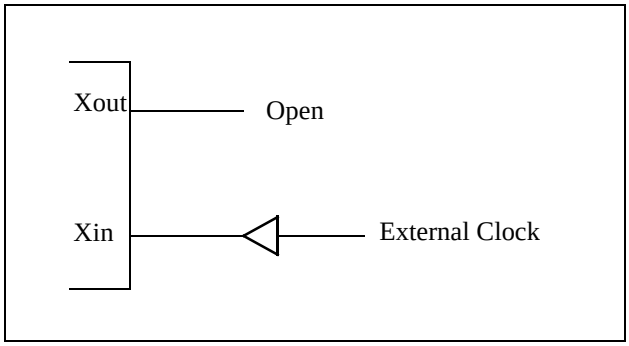
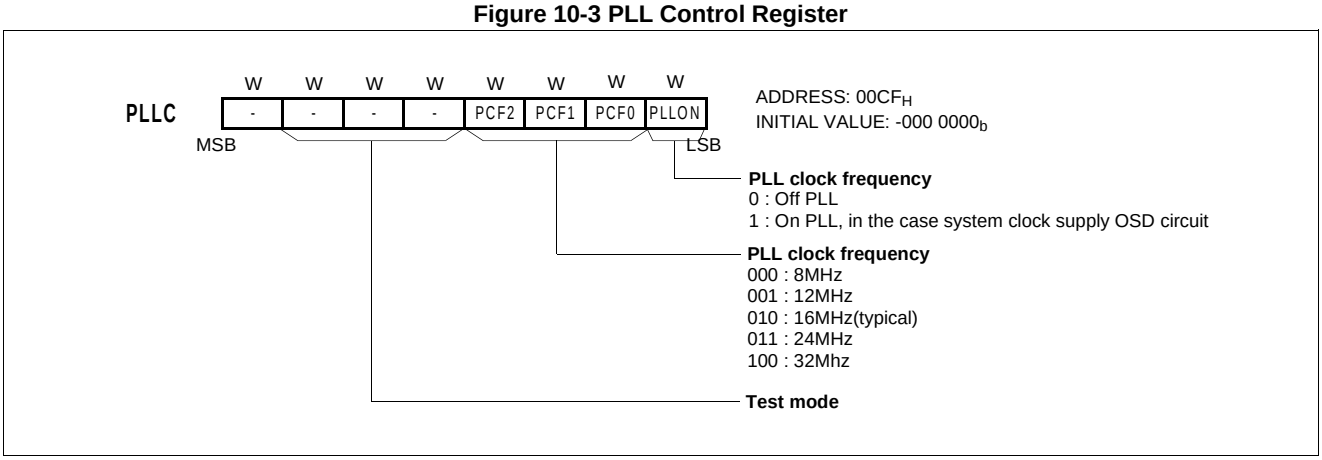


Figure 10-2 External Clock

10.2 Phase Locked Loop

PLL(Phase Locked Loop) from OSC 4MHz clock circuit generate Internal System clock, Timer clock(PS0), Data Slicer Clock, OSD clock, etc.

Figure 10-3 PLL Control Register



10.3 PRESCALER

Prescaler consistor of 11-bit binary counter, and input clock which is supplied by oscillation circuit. Frequency divided by prescaler is used as a source clock for peripherial hardwares.

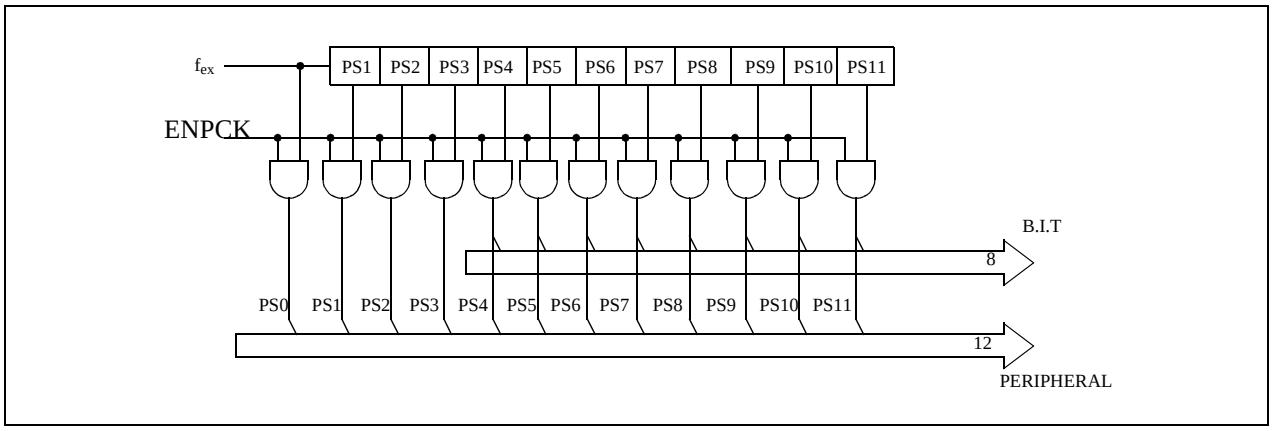


Figure 10-4 Prescaler

Peripheral Clock supplied from prescaler can be stopped by ENPCK. Peripheral clock is determined by CKCTLR Register.(However, PS11 cannot be stopped by ENPCK)

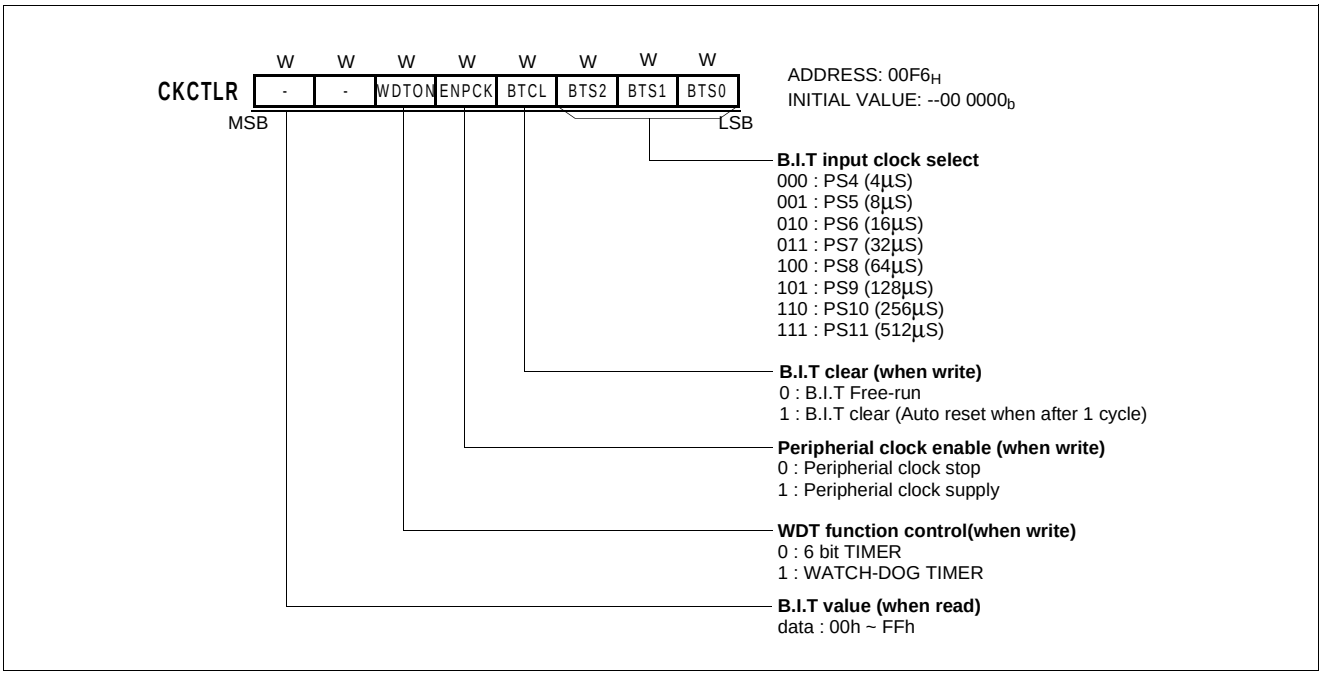


Figure 10-5 Clock Control Register

11. INTERRUPTS

The HMS81C4x60 interrupt circuits consist of Interrupt enable register (IENH, IENL), Interrupt request flags of IRQH and 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 11-2.

Below table shows the Interrupt priority

Reset/Interrupt	Symbol	Priority
Hardware Reset	RESET	-
reserved	-	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
Slicer Interrupt	Slicer	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
reserved	-	14
I^2C Interrupt	I2C	15

The External Interrupts can 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 is set by a overflow in the timer register.

The interrupts are controlled by the interrupt master enable flag I-flag (bit 2 of PSW), that is 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 occured 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	- OSD INT1 INT2 Timer 0 Timer 2 Slicer VSync Timer 1 Timer 3 INTV(INT3/4) WDT BIT - I2C Not used

Table 11-1 Bit function

Interrupt Mode Register

ADDRESS : $00F3_H$
RESET VALUE : Undefined

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
IMOD			M1	M0	IP3	IP2	IP1	IP0

Figure 11-1 Interrupt Mode Register

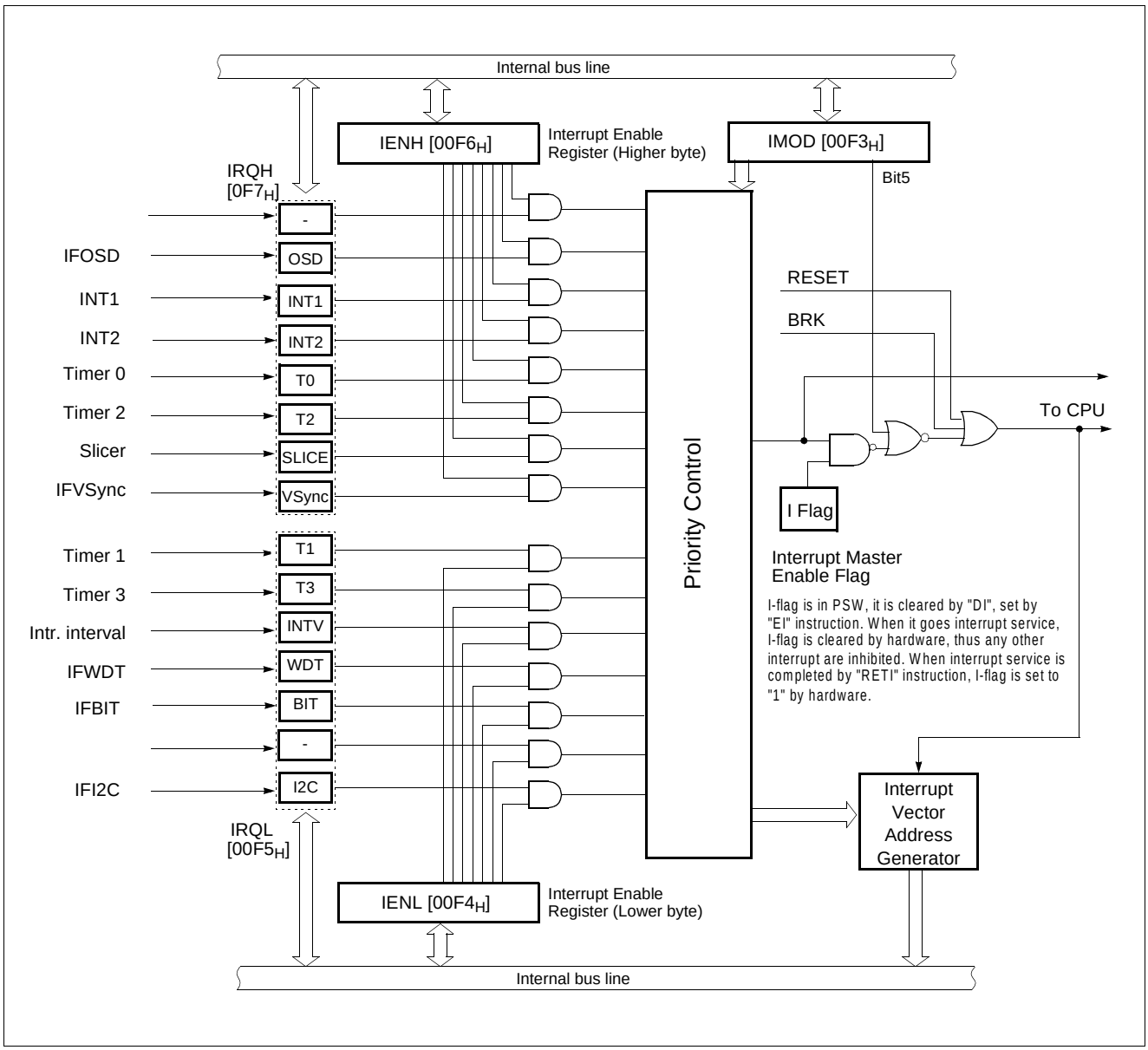


Figure 11-2 Block Diagram of Interrupt

Interrupt request flag registers are shown in Figure 11-3. Interrupt request is generated when suitable bit is set, and suitable request flag of accepted interrup is clear when interrupt processing cycle. Suitable bit is set when interrupt request is occured, but no accepted request flag is set to hold when the interrupt is accepted. Also, interrupt request flag register(IRQH, IRQL) is the register of read or write. So, request flag can be changed by program.

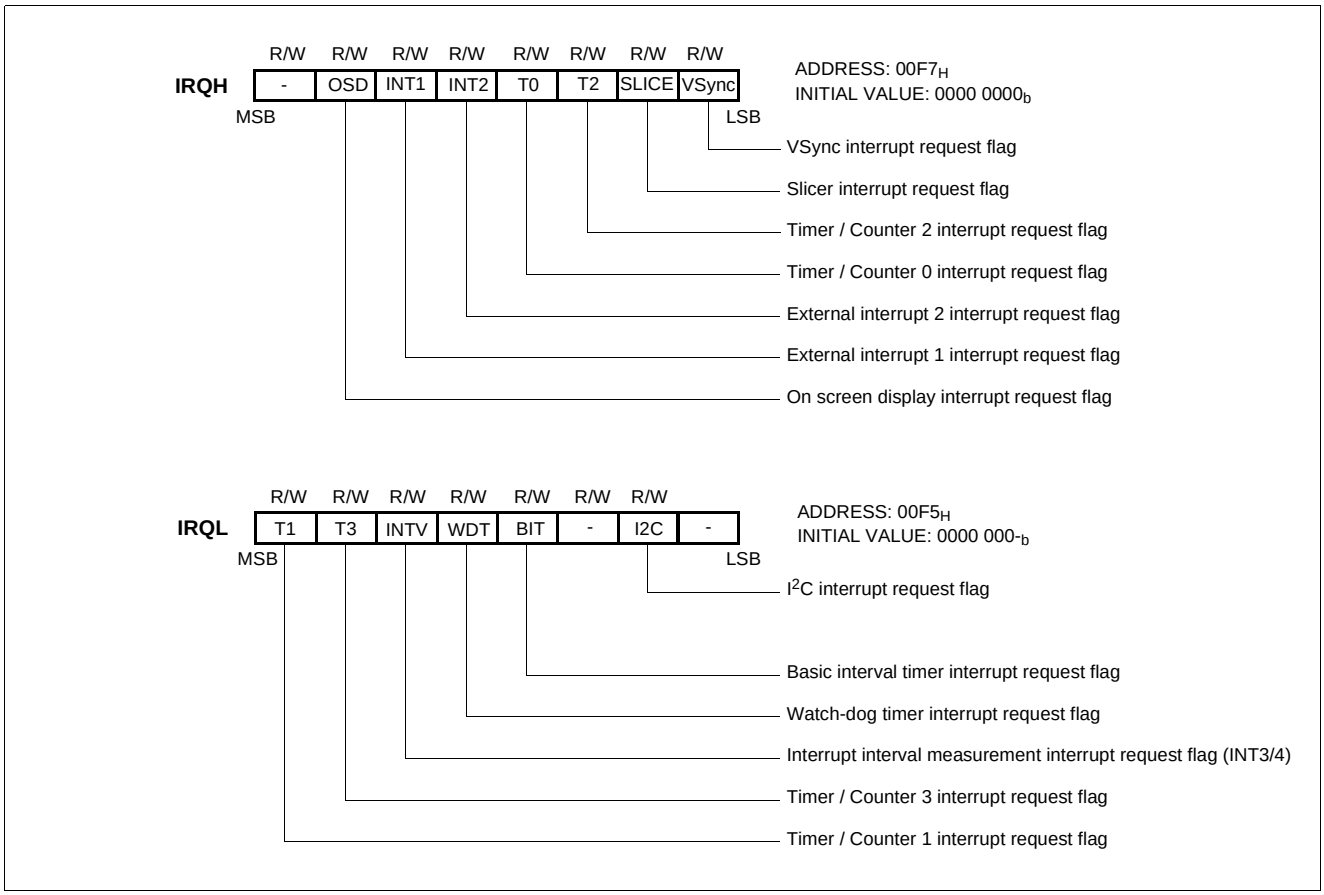


Figure 11-3 Interrupt Request Flag Registers

Interrupt enable flag registers are shown in Figure 11-4. 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.

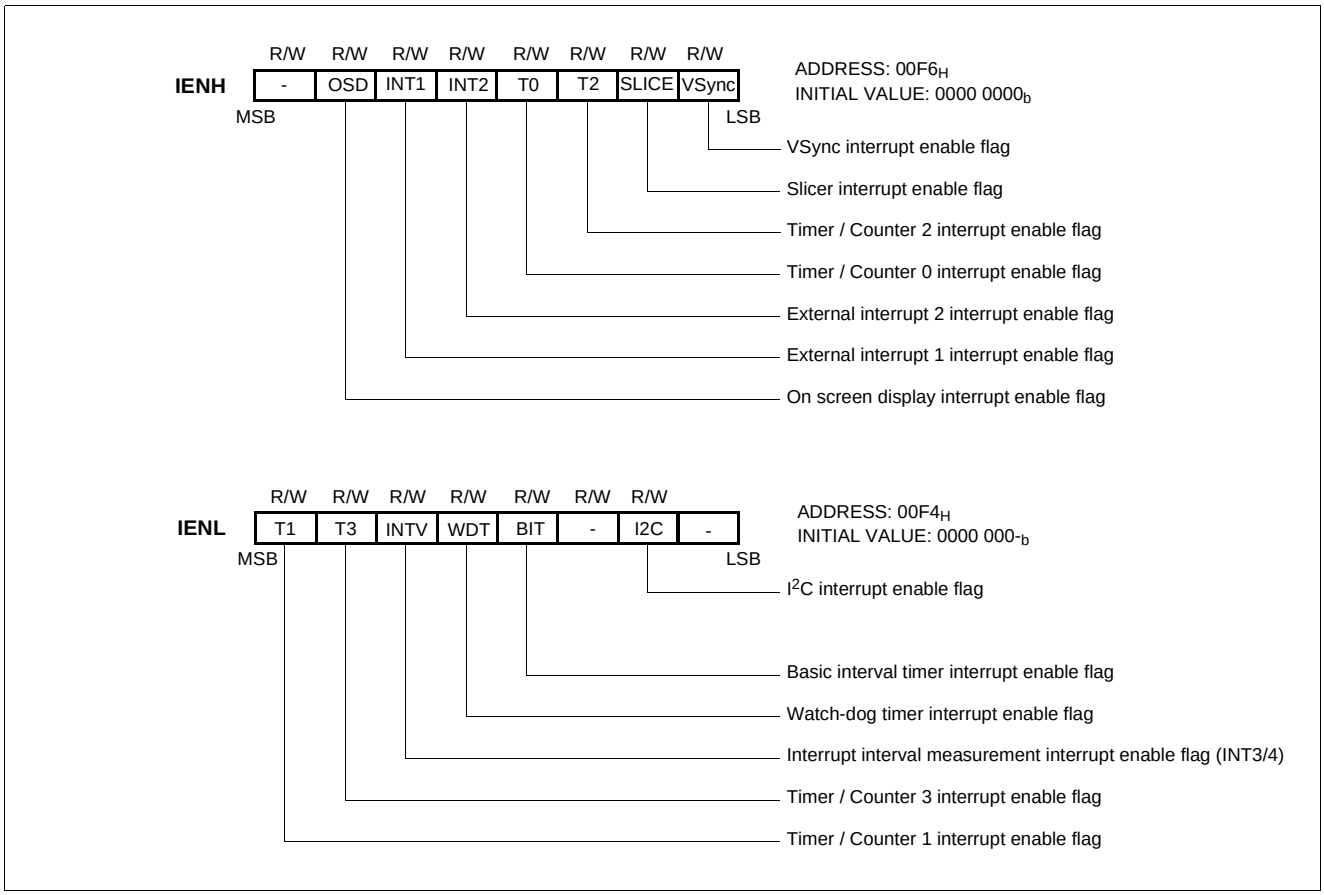


Figure 11-4 Interrupt Enable Flag Regesters

11.1 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.

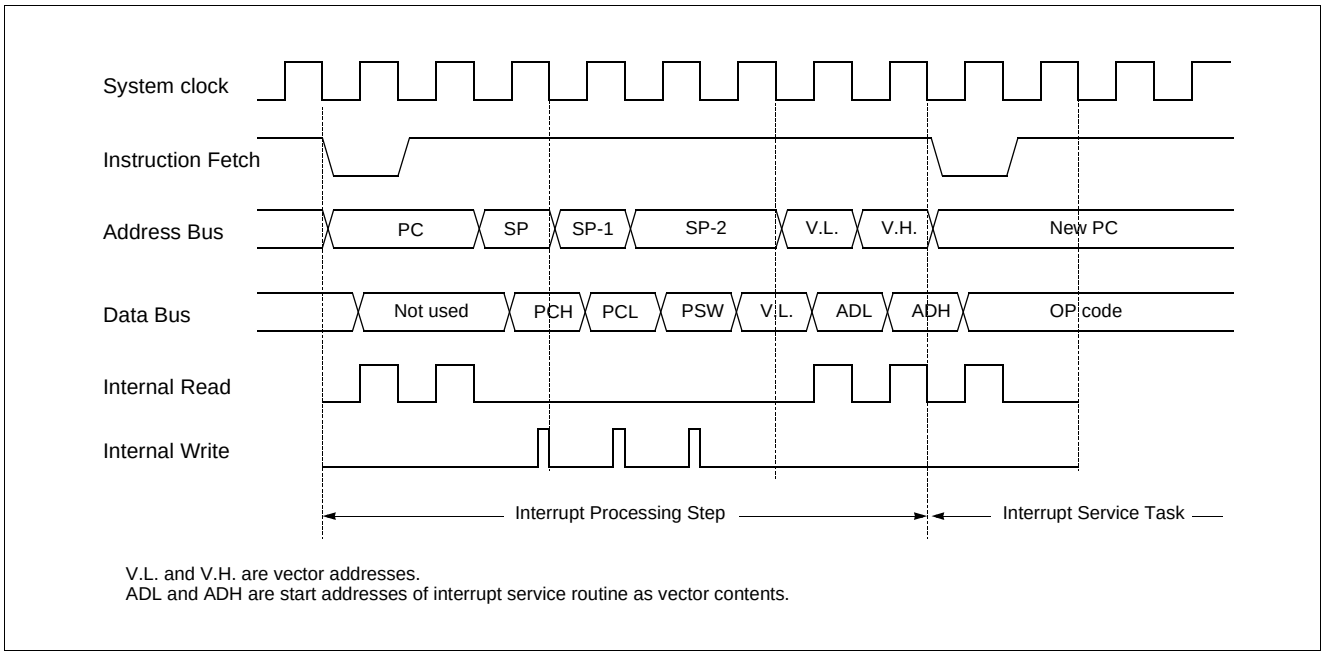
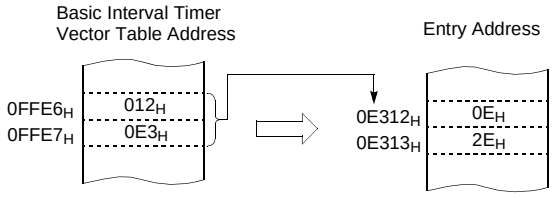


Figure 11-5 Interrupt Service routine Entering Timing



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

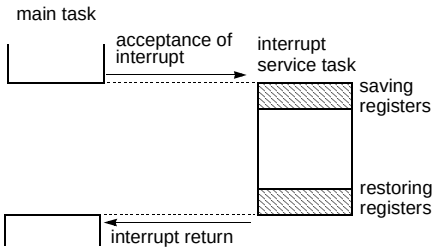
During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using 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;



11.2 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 11-6.

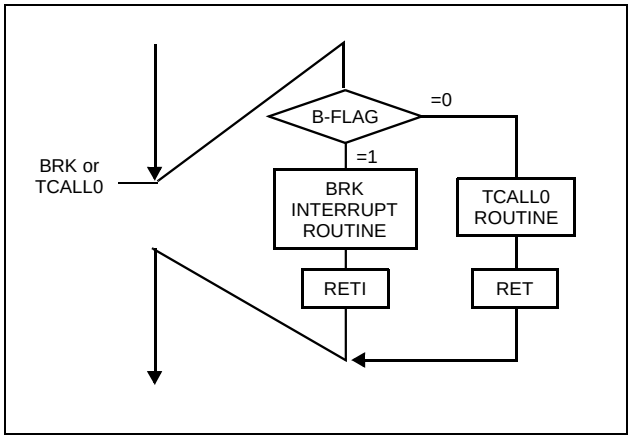


Figure 11-6 Execution of BRK/TCALL0

11.3 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.

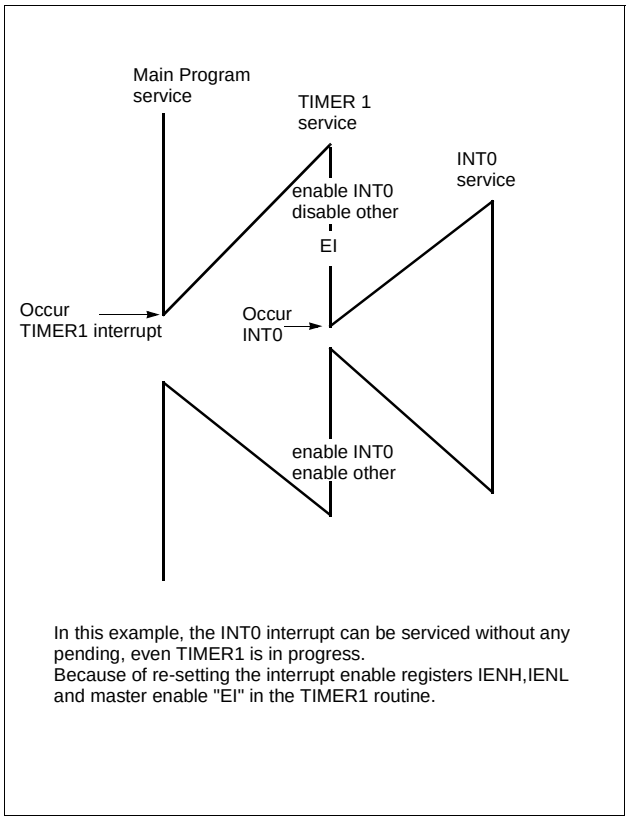


Figure 11-7 Execution of Multi Interrupt

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,#20H    ;Enable INT1 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
```

11.4 External Interrupt

The external interrupt on INT1, INT2... pins are edge triggered depending the edge selection register.

Refer to “6. PORT STRUCTURES” on page 12.

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

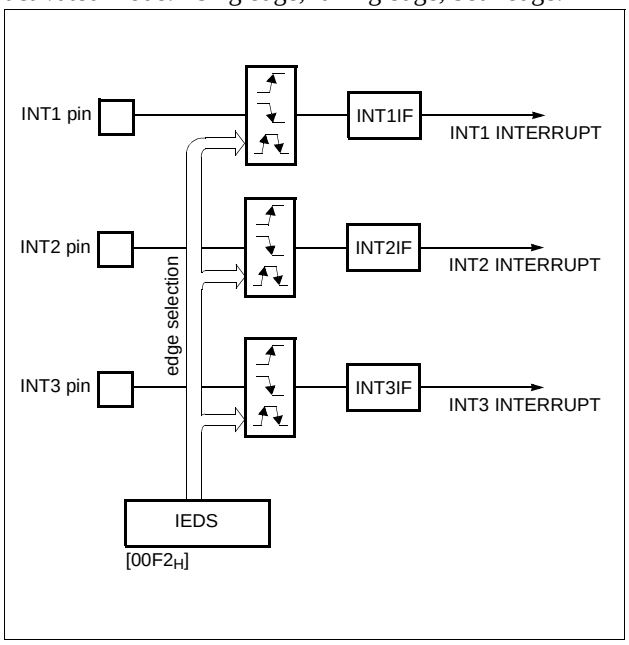


Figure 11-8 External Interrupt Block Diagram

INT1, INT2 and INT3 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 INT1, INT2 and INT3 edge are latched into INT1IF, INT2IF and INT3IF at every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. 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

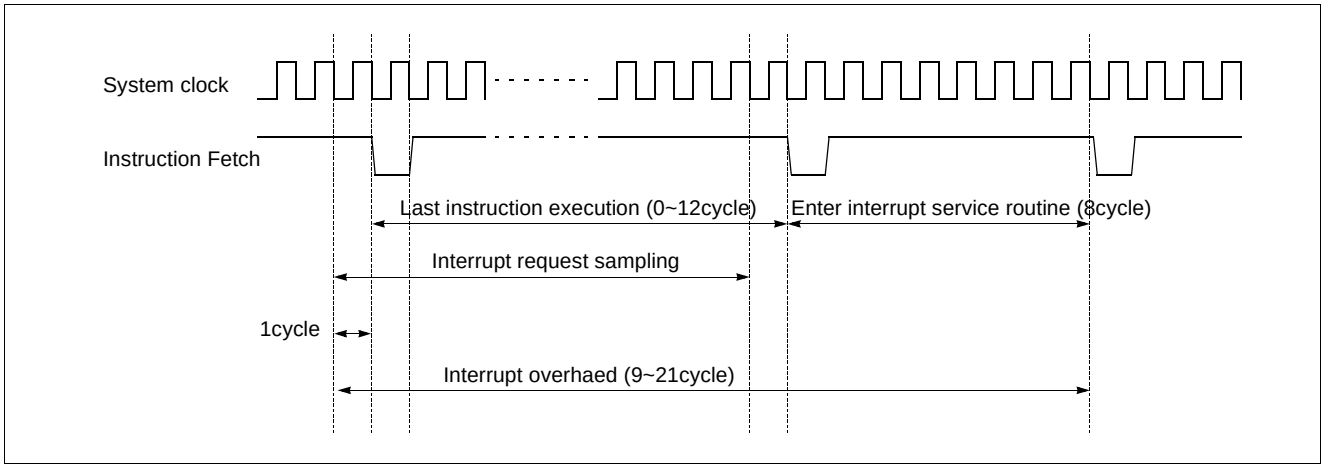


Figure 11-9 Interrupt Response Timing Diagram (Interrupt overhead)

12. TIMER

12.1 Basic Interval Timer

The HMS81C4x60 has one 8-bit Basic Interval Timer that is free-run and can not be stopped. Block diagram is shown in Figure 12-1.

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 12-2.

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..

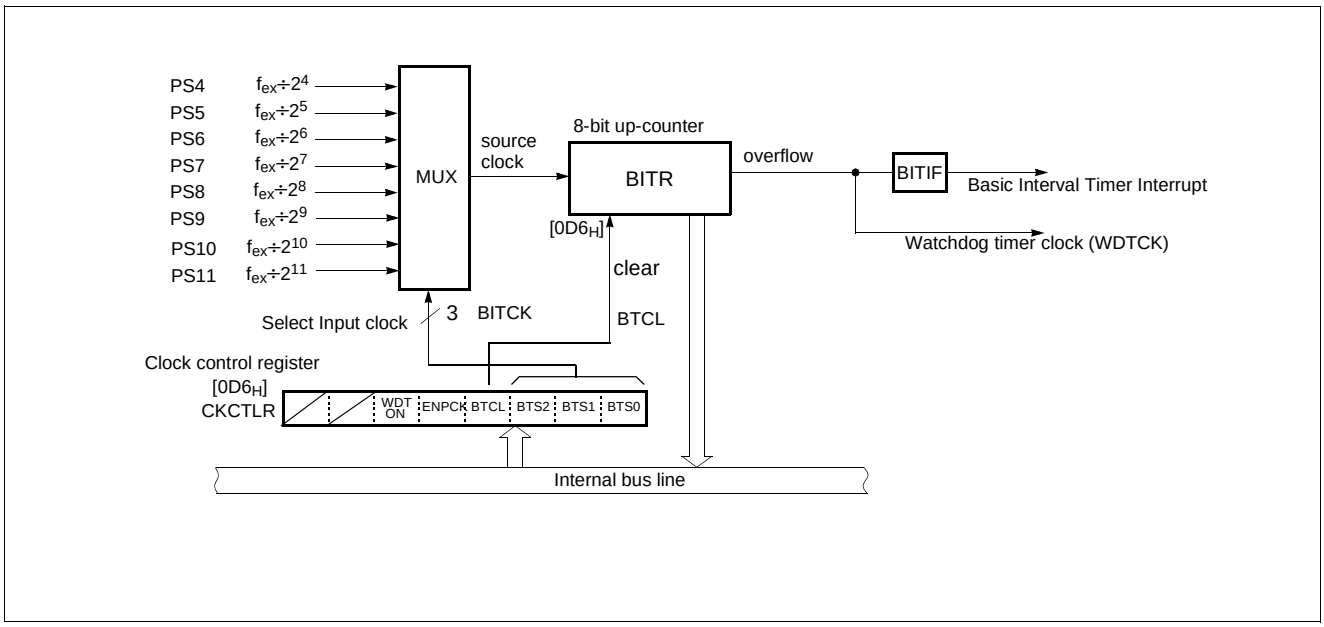


Figure 12-1 Block Diagram of Basic Interval Timer

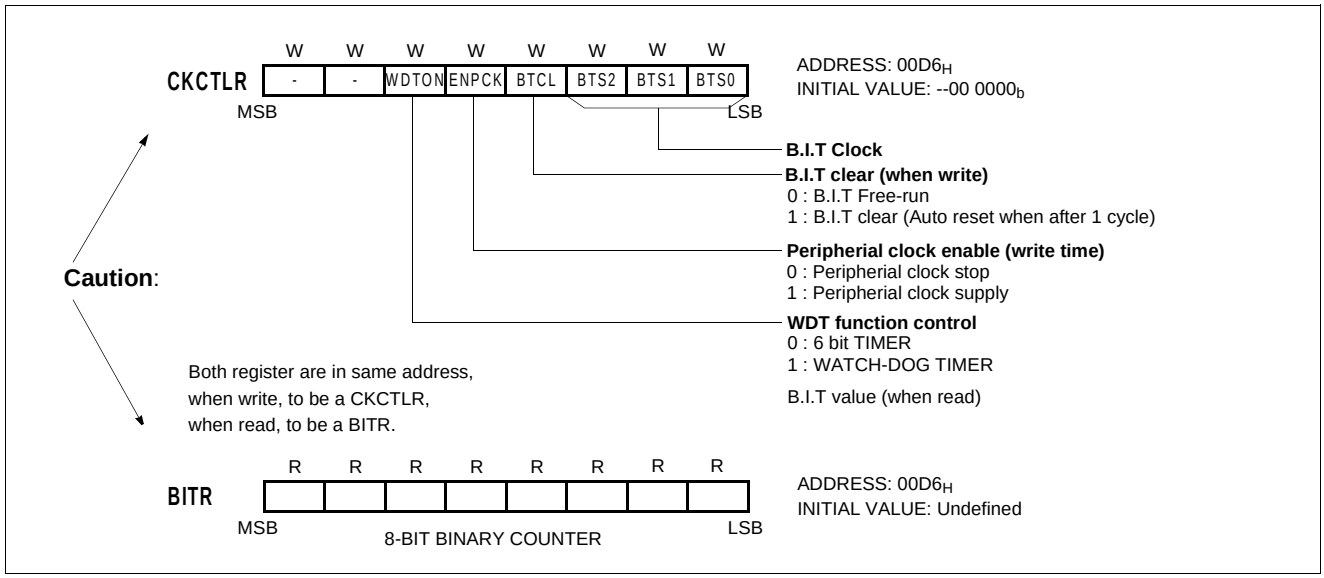


Figure 12-2 BITR Basic Interval Timer Mode Register

12.2 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 12-3 and Figure 12-4.

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.

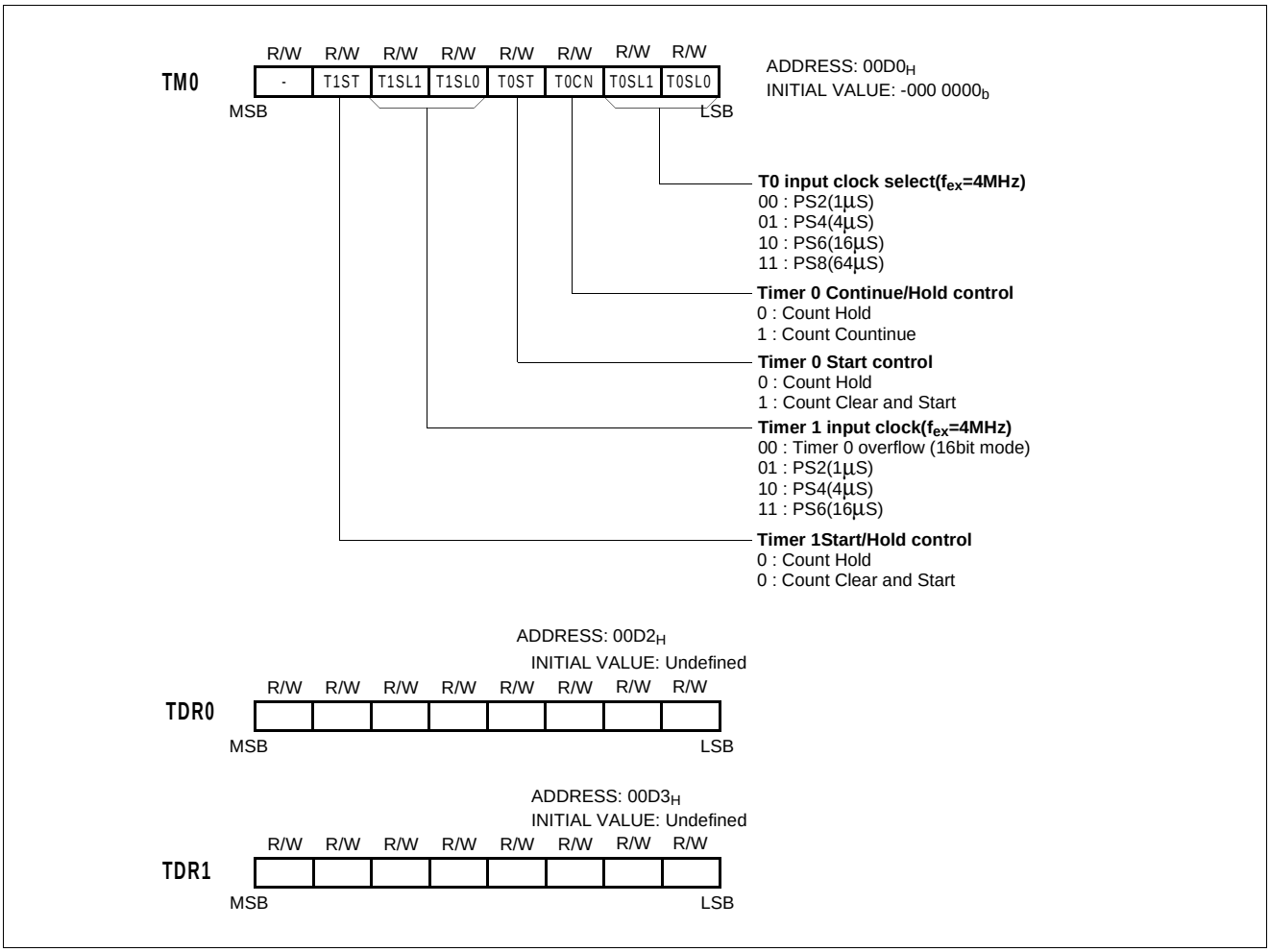


Figure 12-3 Timer / Event Count 0,1

```
(Example) TIMER0 1mS TIME INTERVAL INTERRUPT
          :
          :
TDR_CNT:  LDM     TDR0,#249
          LDM     TDR1,#0
          LDM     TM0,#0011_1101b         ; 4uSEC PRESCALER FOR T0
          :
          :
```

.

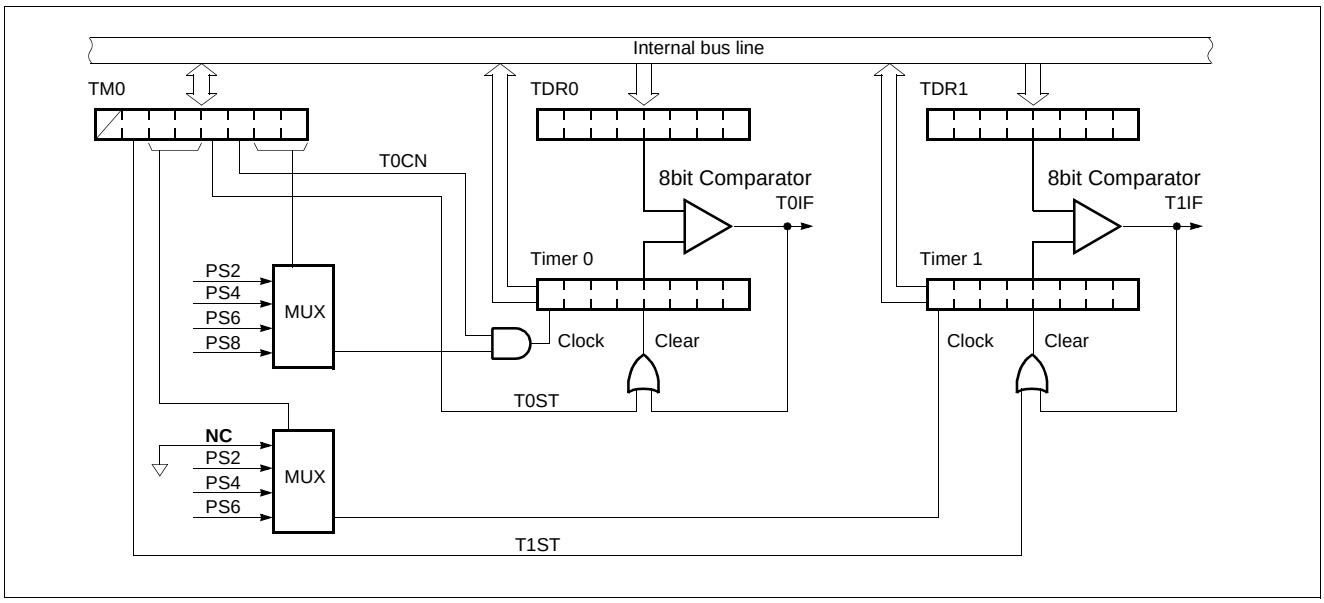


Figure 12-4 Simplified Block Diagram of 8bit Timer0, 1

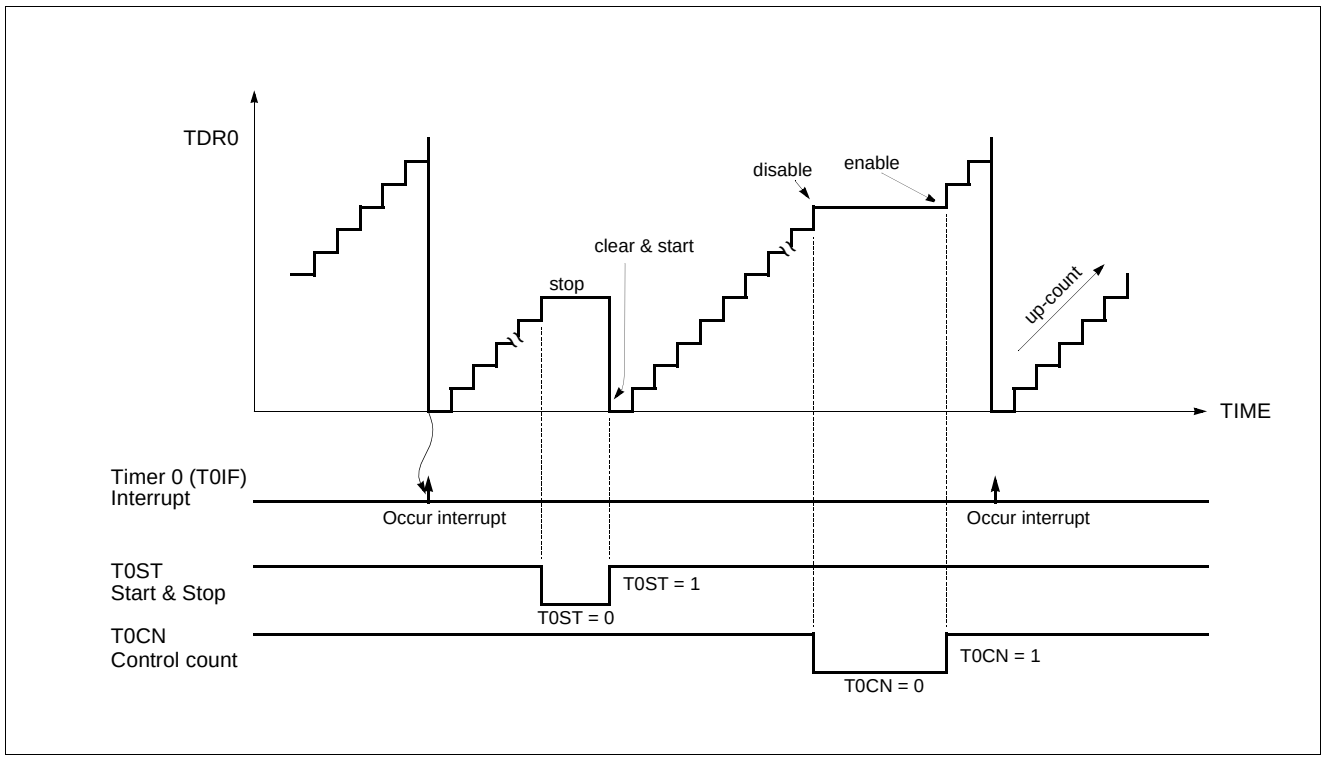


Figure 12-5 Count Example of Timer

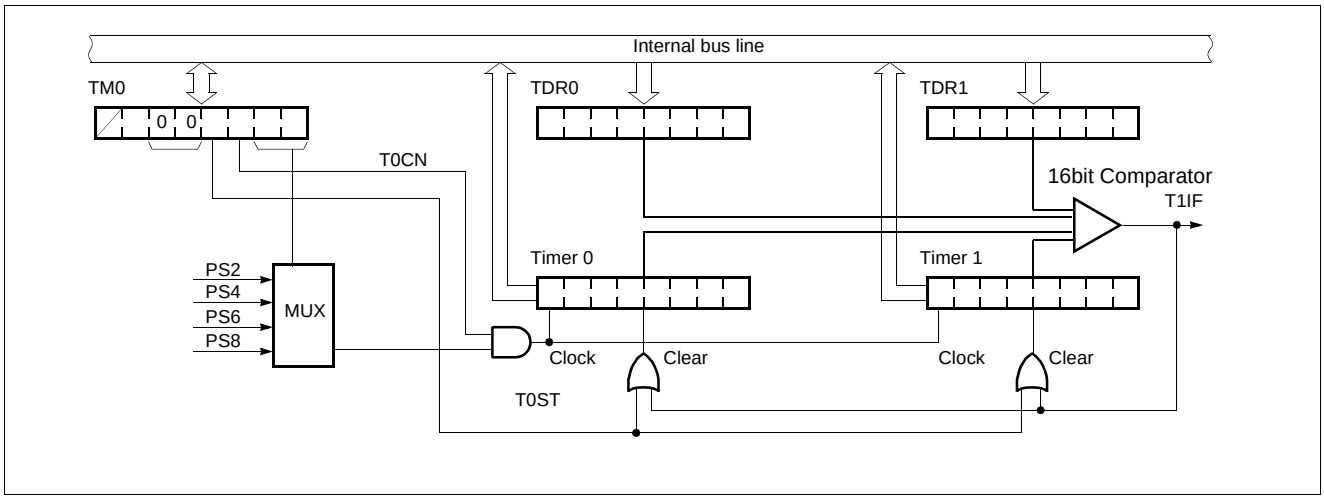


Figure 12-6 Simplified Block Diagram of 16bit Timer0, 1

12.3 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 12-7 and Figure 12-8.

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 R24/EC2, R25/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

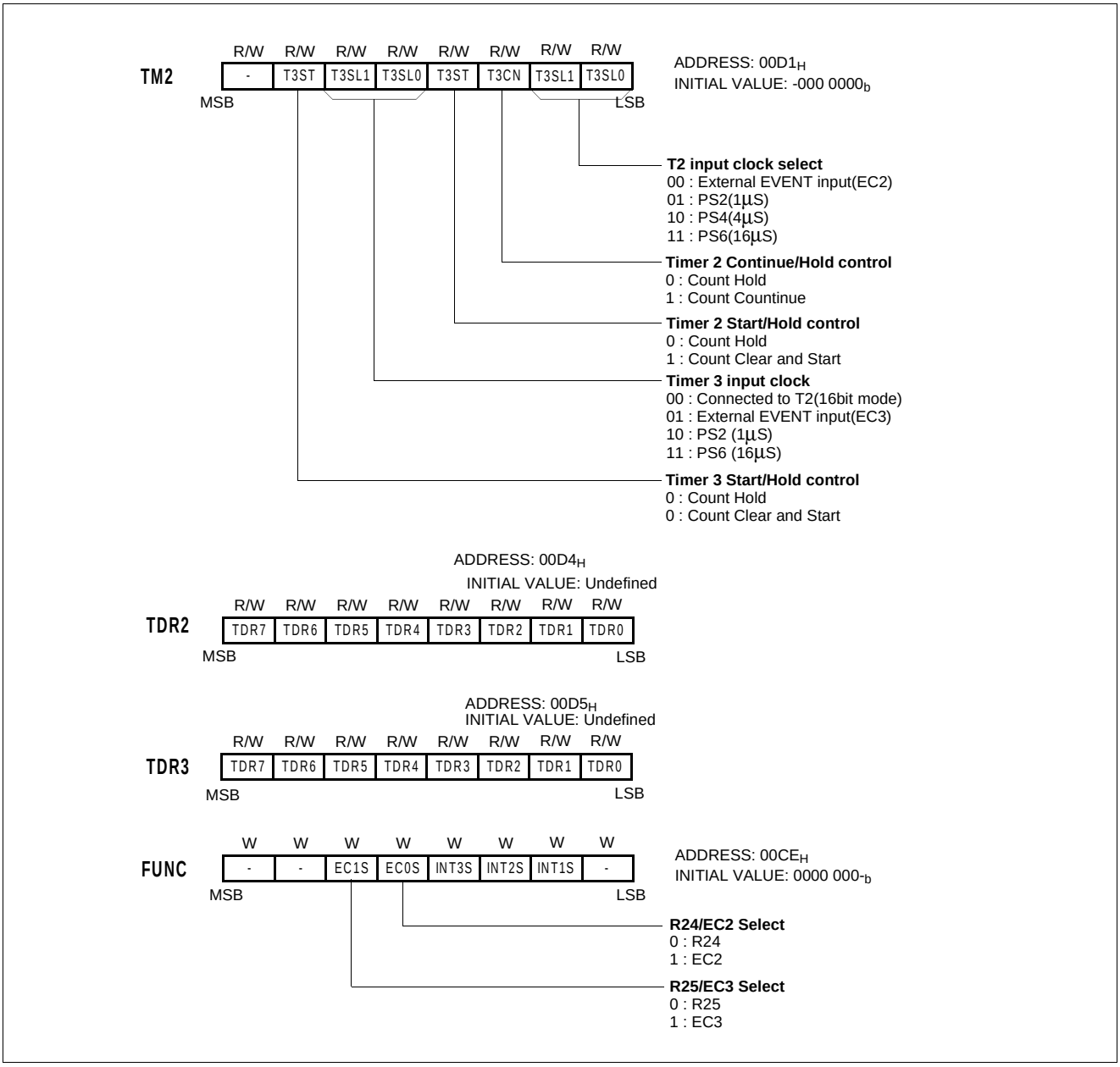


Figure 12-7 Timer / Event Count 2,3

.

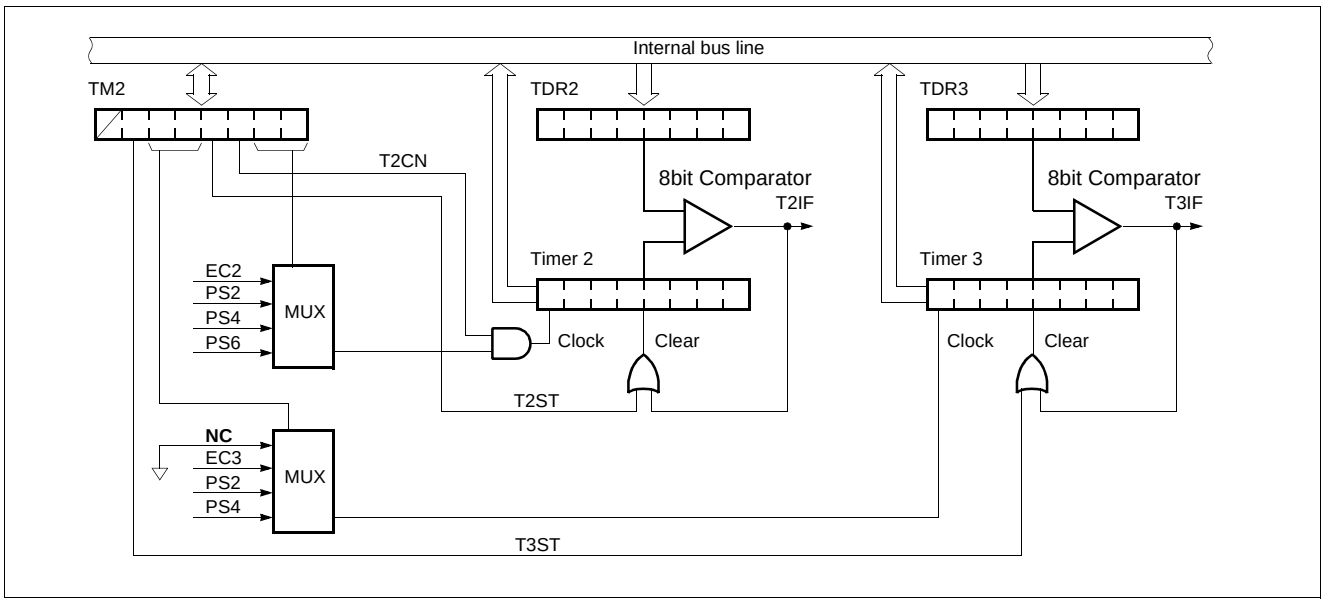


Figure 12-8 Simplified Block Diagram of 8bit Timer/Event Counter 2,3

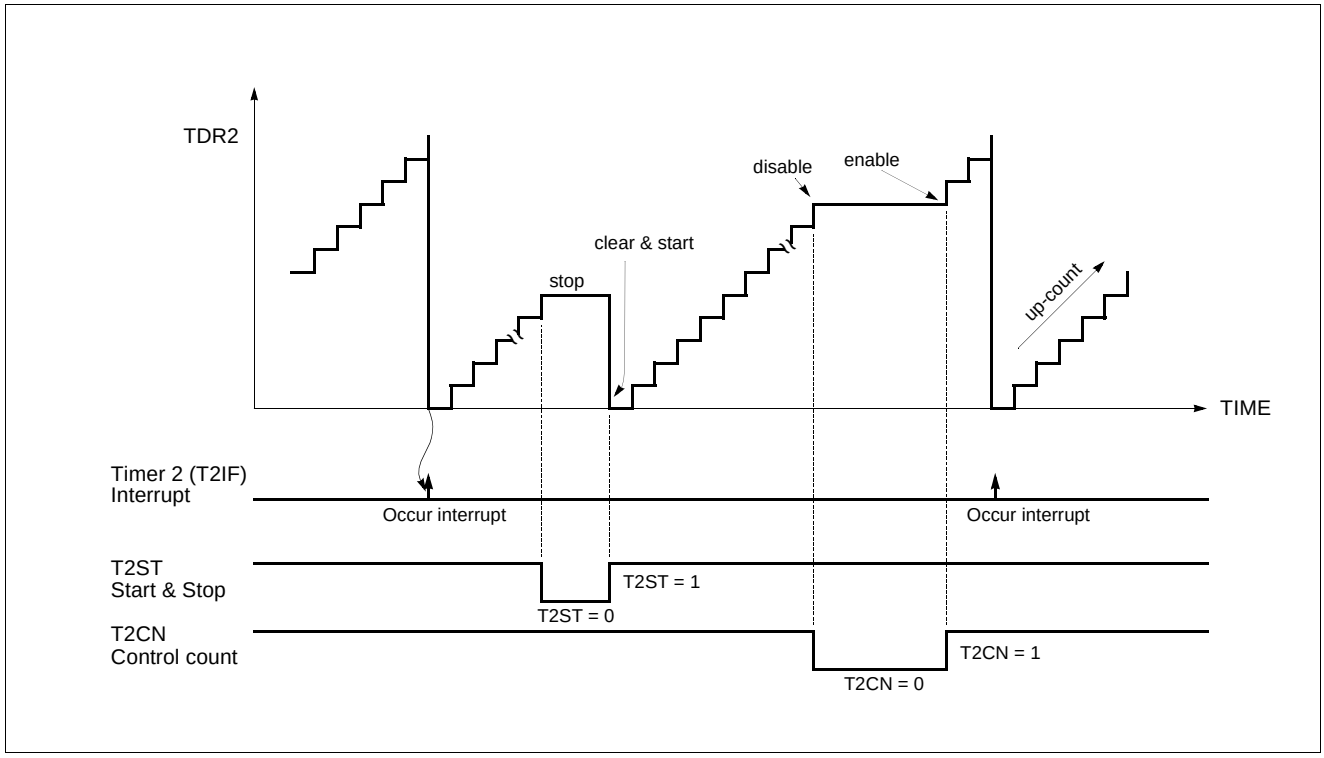


Figure 12-9 Count Example of Timer / Event counter

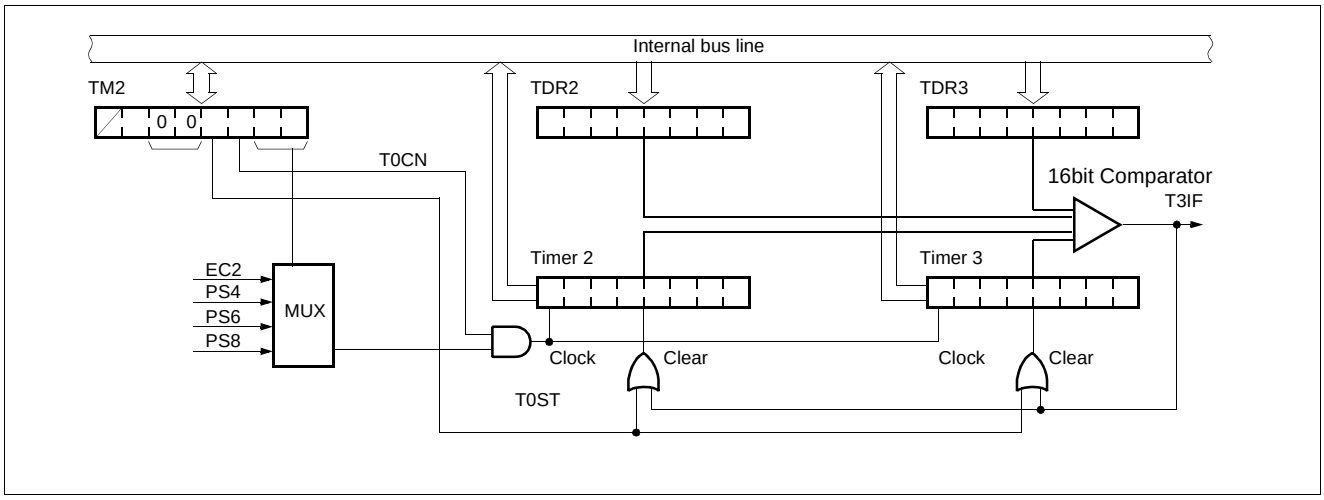


Figure 12-10 Simplified Block Diagram of 16bit Timer/Event Counter 2,3

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.

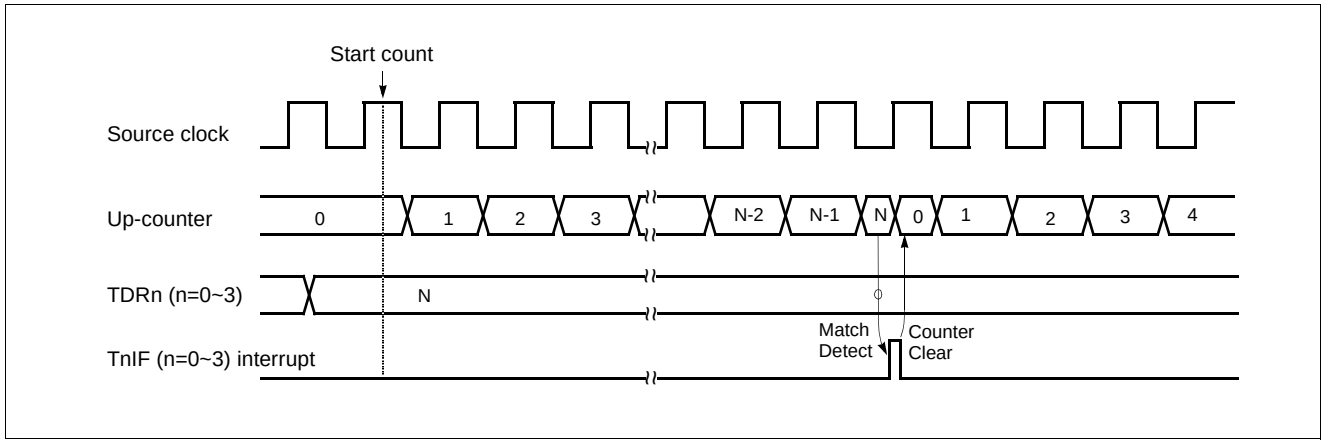


Figure 12-11 Timer Mode Timing Chart

Event Counter Mode

In event timer mode, counting up is started by an external trigger. This trigger means falling edge of the ECn (n=0~1) 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 EC0S, EC1S of the Port Function Select Register FUNC(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.

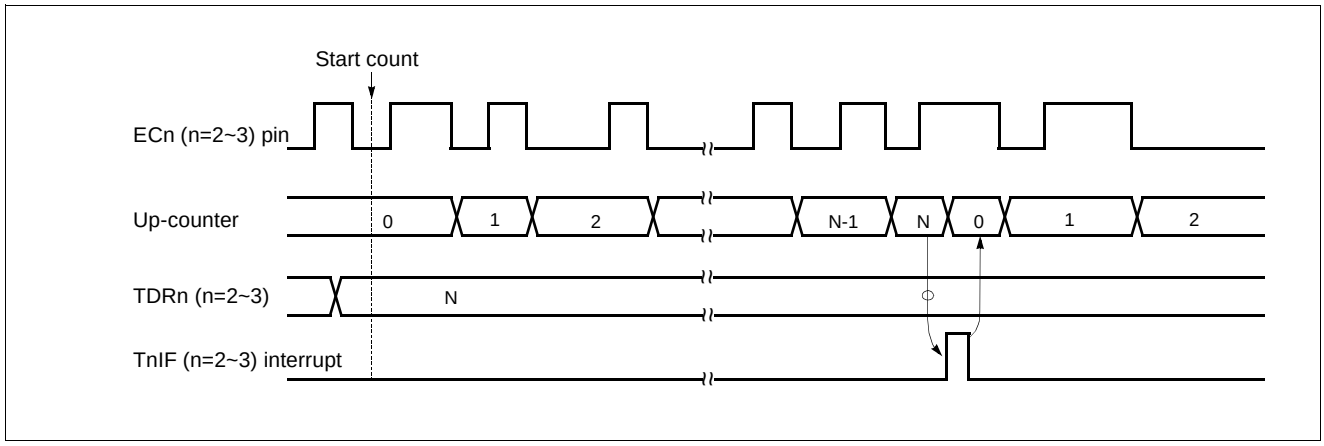


Figure 12-12 Event Counter Mode Timing Chart

The interval period of Timer is calculated as below equation.

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

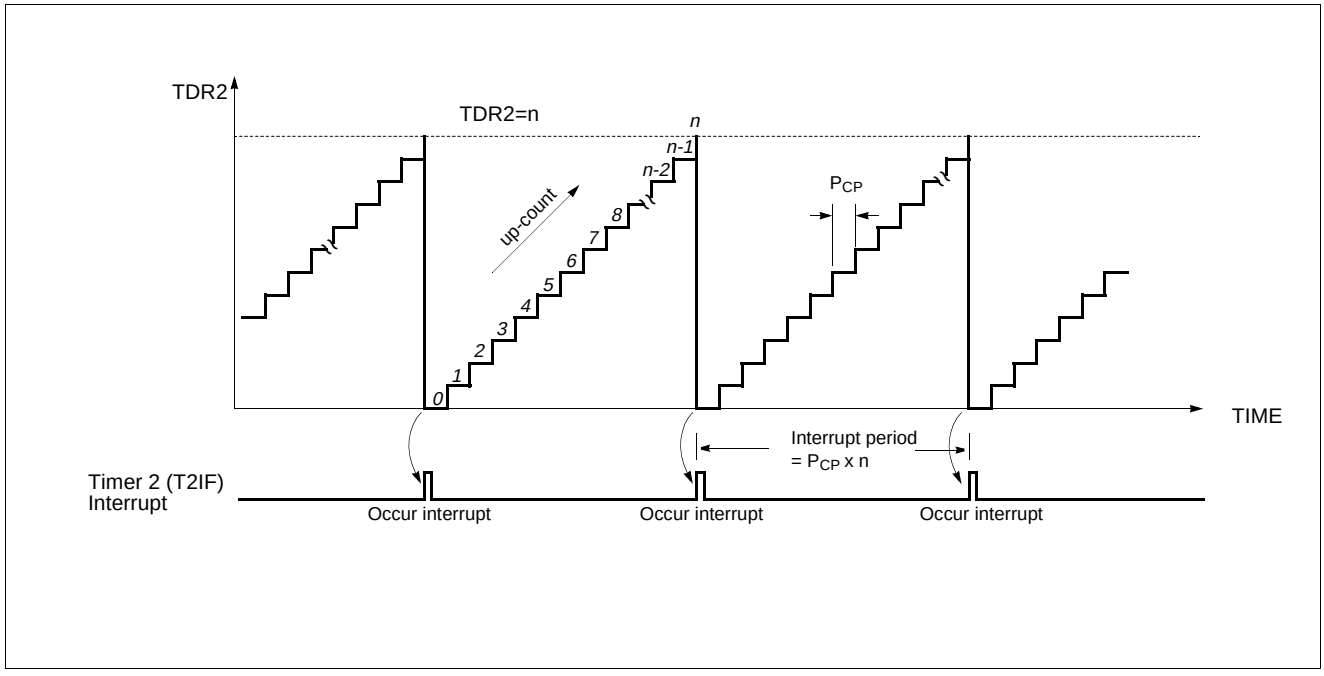


Figure 12-13 Count Example of Timer / Event counter

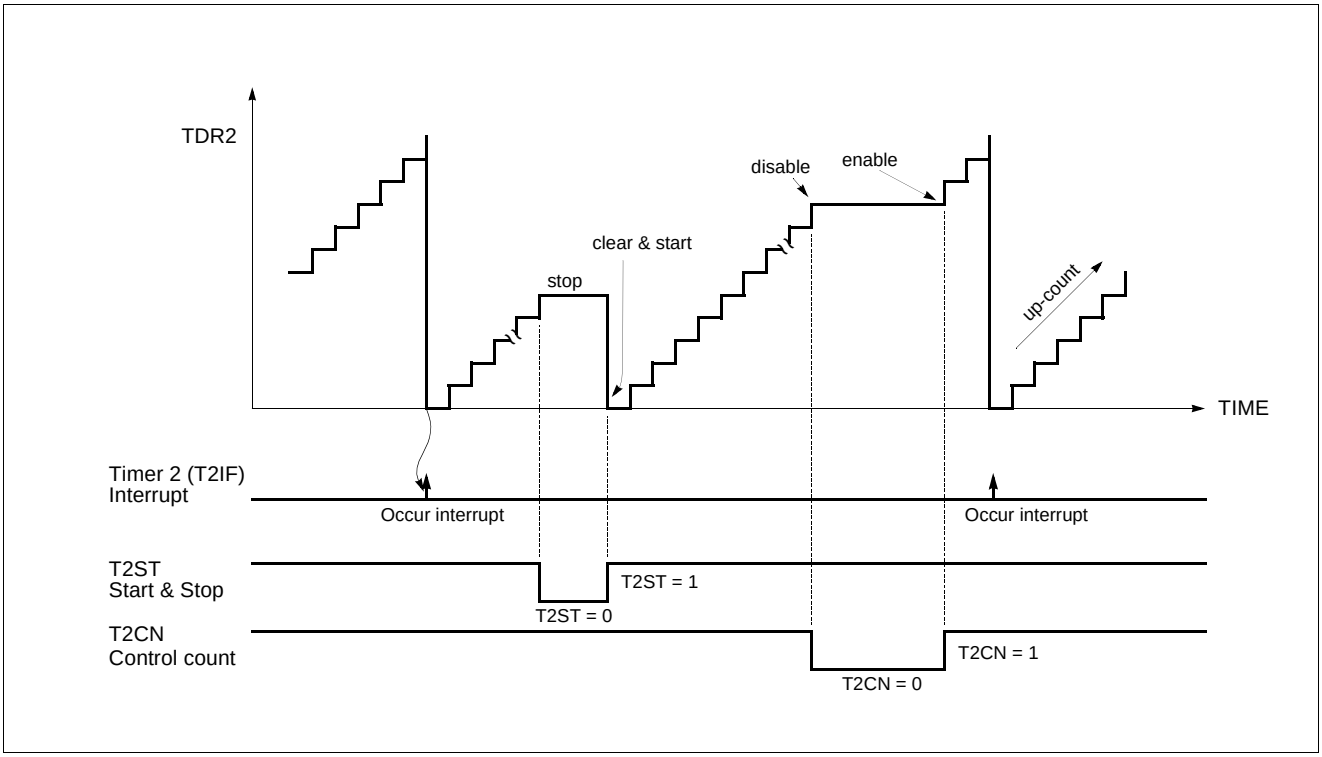


Figure 12-14 Count Operation of Timer / Event counter

13. A/D Converter

The A/D converter circuit is shown in Figure 13-1.

The A/D converter 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 13-2.

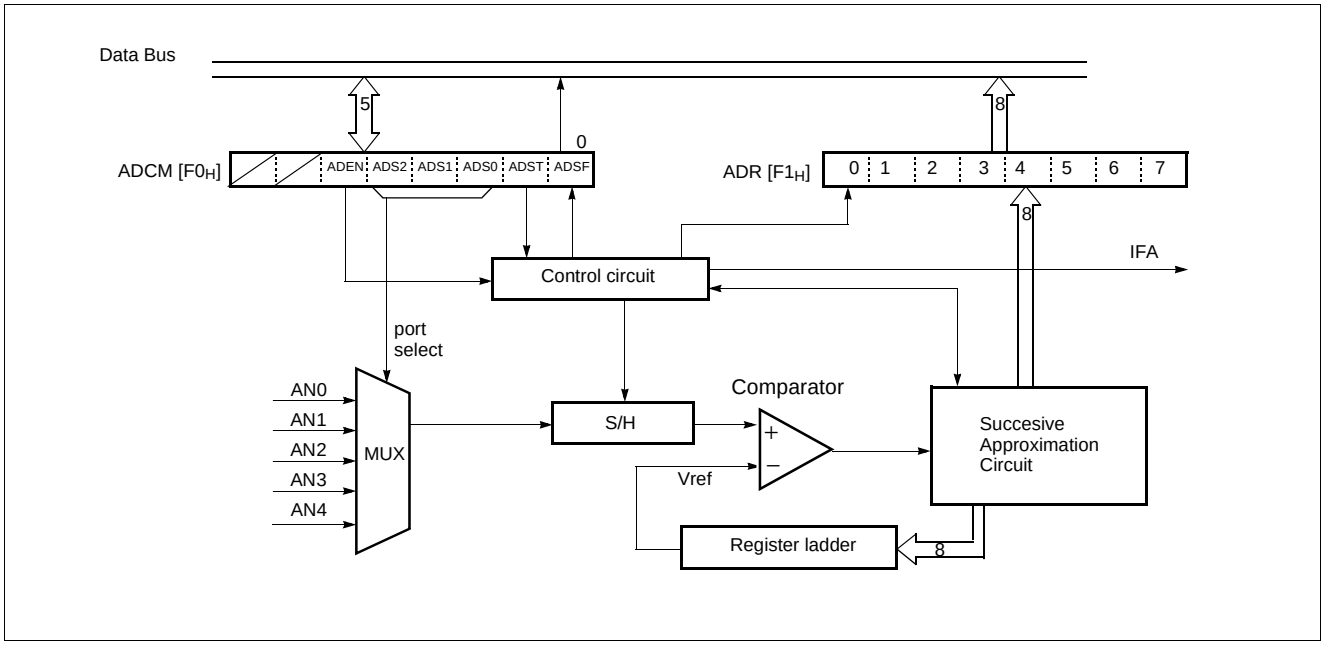


Figure 13-1 Block Diagram of A/D convertor circuit

Control

The HMS81C4x60 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 1 cycle automatically after you set this.

[Example]

```
;Set AIPS, change ? to what you want
;    0 : digital port
;    1 : analog port
LDM   AIPS,#0000_1000b
; Set ADEN, xxx is analog port number
LDM   ADCM,#0010_1100b
; or "SET1 ADEN"
; Set ADST, xxx is analog port number
LDM   ADCM,#0010_11110b
BBC  ADCM.ADSF,$
LDA  ADR
; 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 : 18uS when f_{ex}=4MHz).

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.*

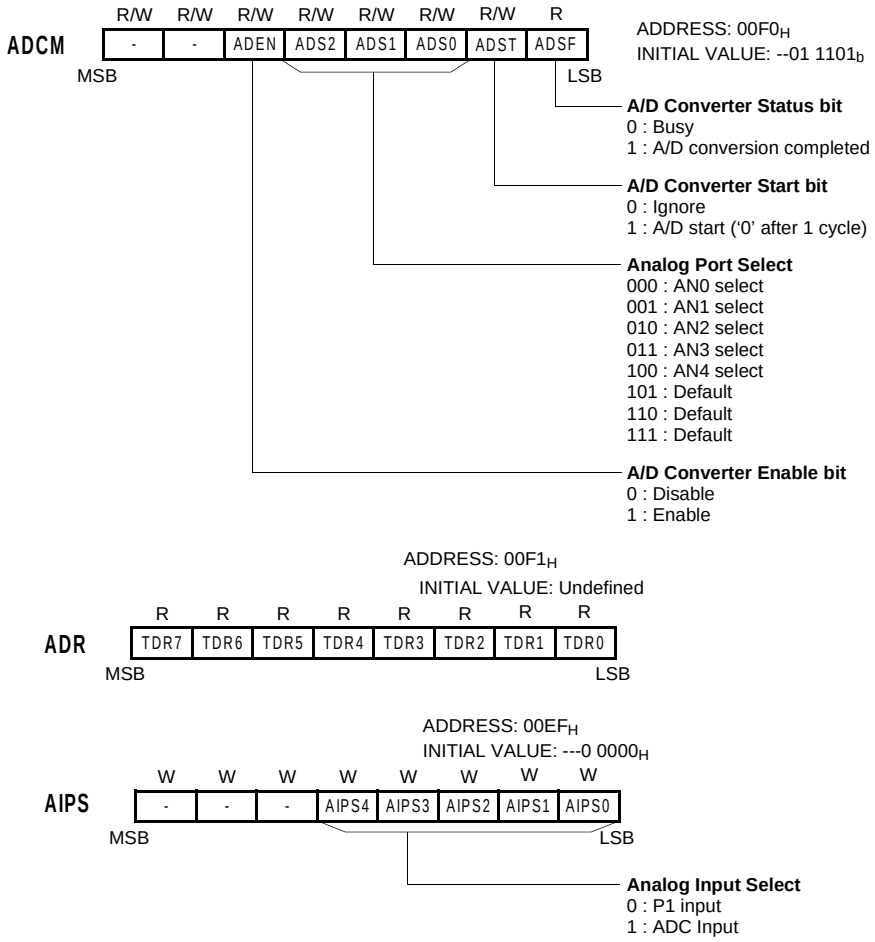


Figure 13-2 A/D convertor Registers

ADS2	ADS1	ADS0	Function	PORT select				
				R14/AN4	R13/AN3	R12/AN2	R11/AN1	R10/AN0
0	0	0	AN0	R14	R13	R12	R11	**AN0**
0	0	1	AN1	R14	R13	R12	**AN1**	R10
0	1	0	AN2	R14	R13	**AN2**	R11	R10
0	1	1	AN3	R14	**AN3**	R12	R11	R10
1	0	0	AN4	**AN4**	R13	R12	R11	R10

Figure 13-3 A/D Conversion Data Register

14. Pulse Width Modulation (PWM)

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

Example (f_{ex}=4MHz)	14bit PWM	8bit PWM
Resolution	14 bits	8 bits
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 PWMR4~0, PWMCR2~1, PWM5H, PWM5L.

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

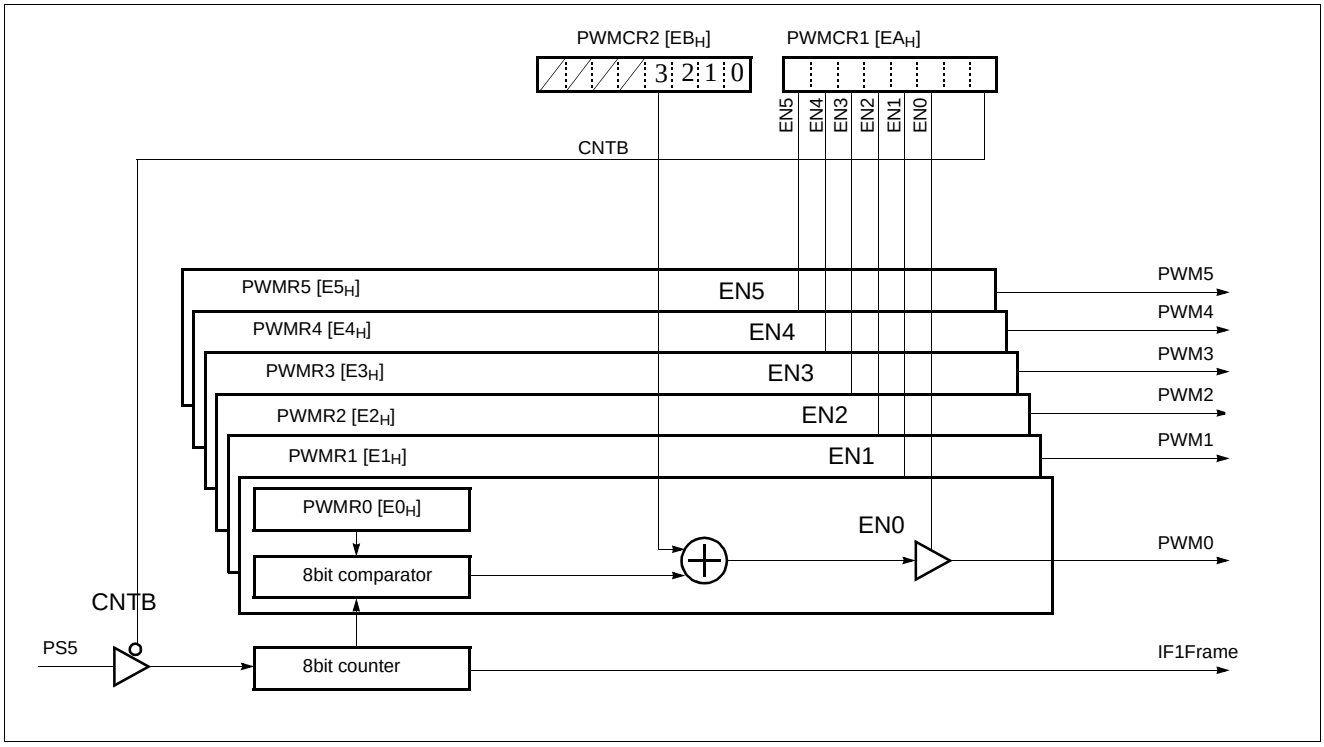


Figure 14-1 8bit register (PWM7~0) circuit

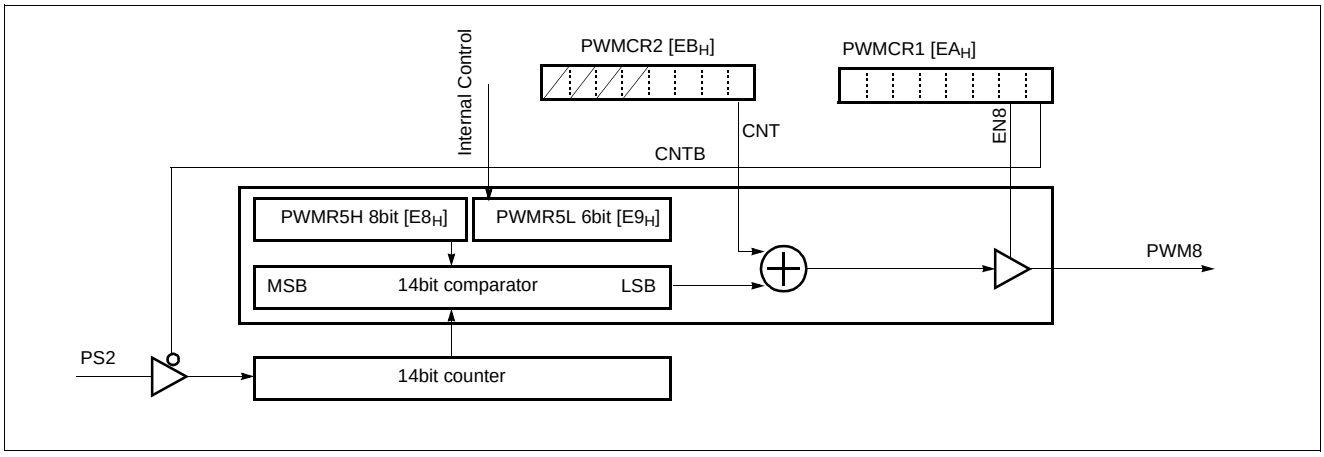


Figure 14-2 14bit register (PWM8) circuit

8bit PWM Control

The HMS81C4x60 contains a one 14bit PWM and five 8bit PWM module.

1. 8bit PWM0~5 is wholy same internal circuit, but PWM0~5 output port is CMOS bidirectional I/O pin.

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)

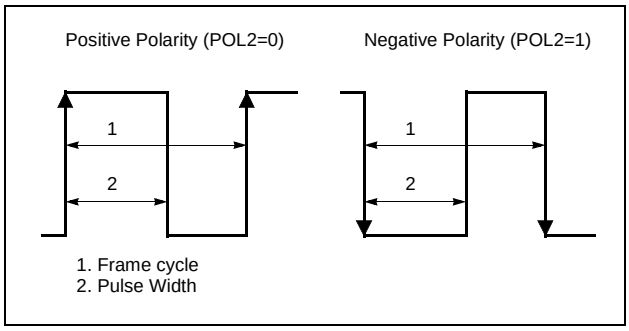


Figure 14-3 Wave form example for 8bit PWM

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

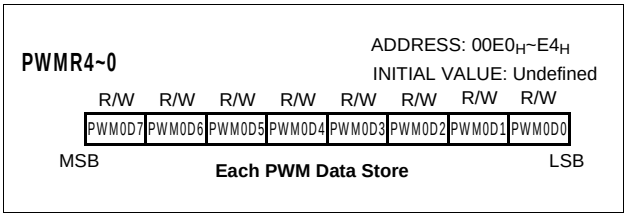


Figure 14-4 8bit PWM Registers

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

14bit PWM Control

1. 14bit PWM's operation concept is not the same as 8bit PWM.
1 PWM frame contains 64 sub PWMs.
PWM5H : Set sub PWM's basic Pulse Width.
PWM5L : 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, "PWM5L and Sub frame matching table," on page 58 show PWM5L 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 PWM5L and Sub frame matching table

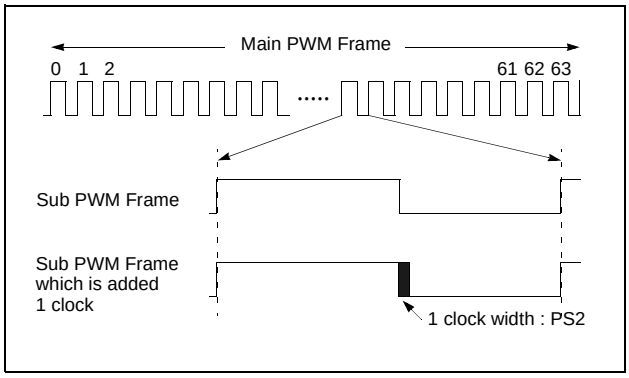


Figure 14-5 Wave form example for 14bit PWM

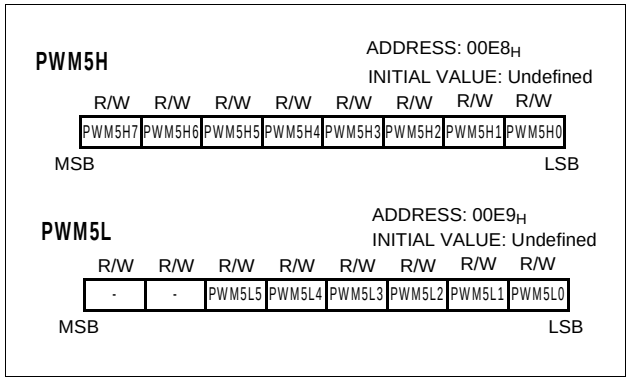


Figure 14-6 PWM5H, PWM5L Register

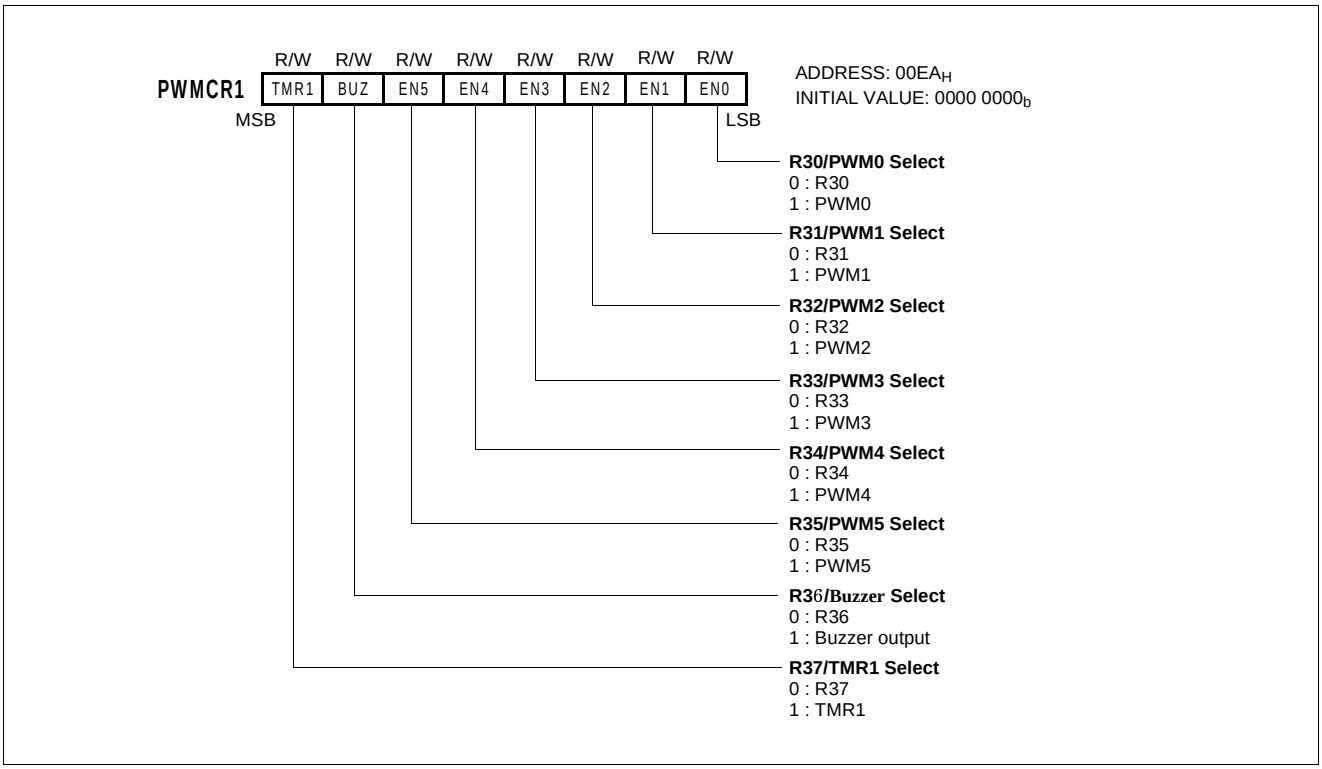


Figure 14-7 PWM Control Register 1

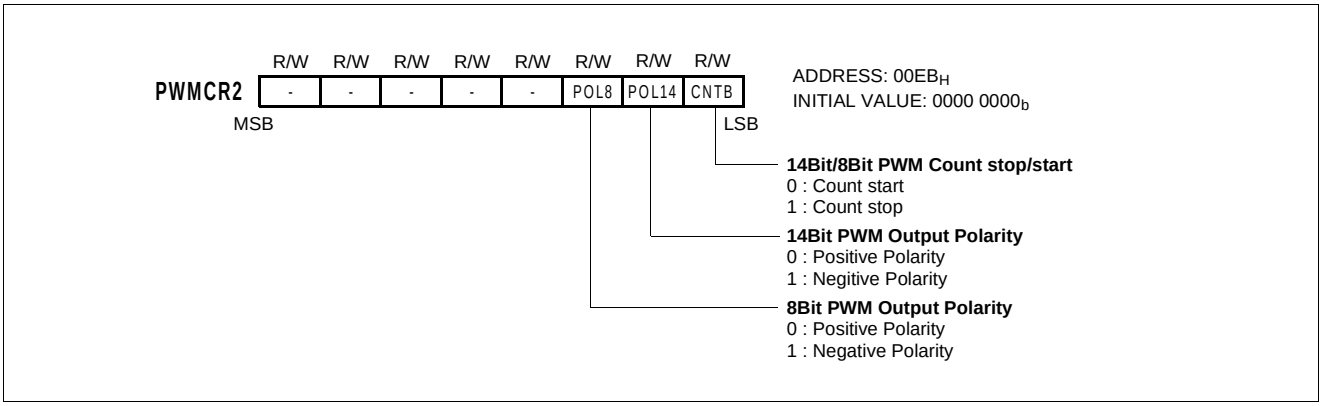


Figure 14-8 PWM Control Register 2

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 (9 bit, 6 level) interrupt, Control register, etc.

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

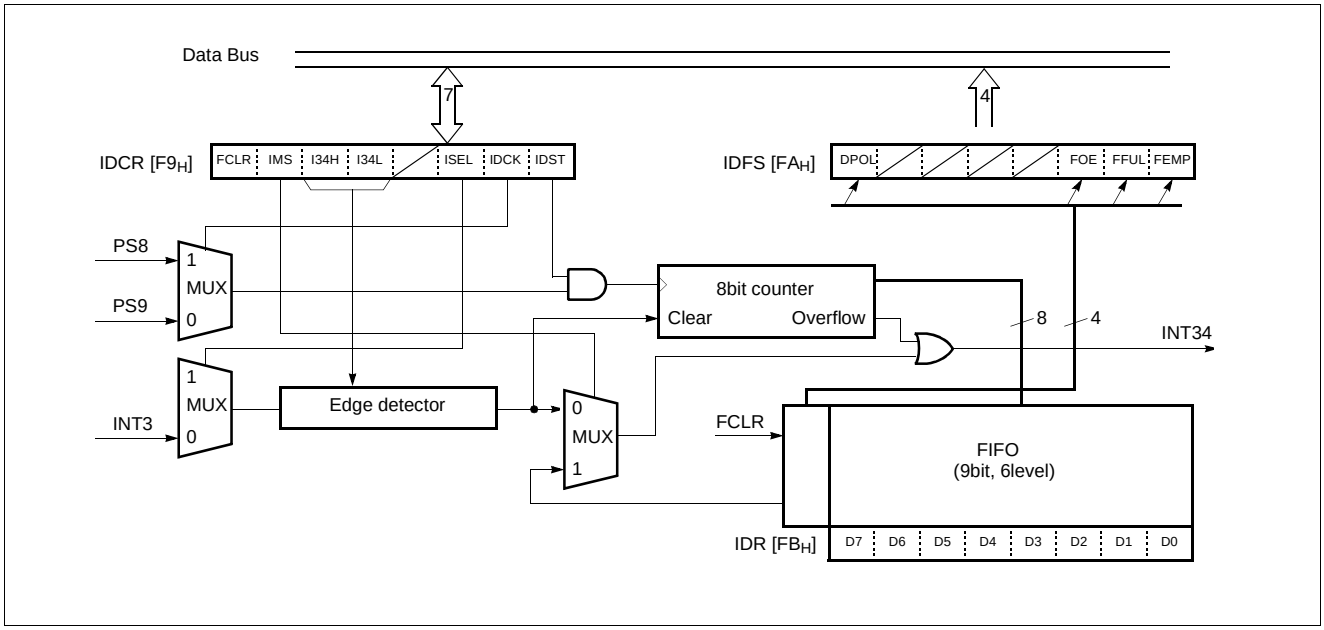


Figure 15-1 Block Diagram of Interrupt interval measurement circuit

Control

The HMS81C4x60 contains a Interrupt interval measurement module.

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

2. Set IDCR [$00F9_H$] : FIFO clear, interrupt mode select, interrupt edge select, external interrupt INT3 select, sampling clock select, COUNT start/stop 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 15-7 FIFO operating mechanism.

```
[Example]
        ;Set INT3 for remote control pulse
         reception

        LDM    FUNC,#0000_1001b;INT3 SET
        LDM    IDCR,#1001_0001b ;64uSec PCS
        :
        :
```

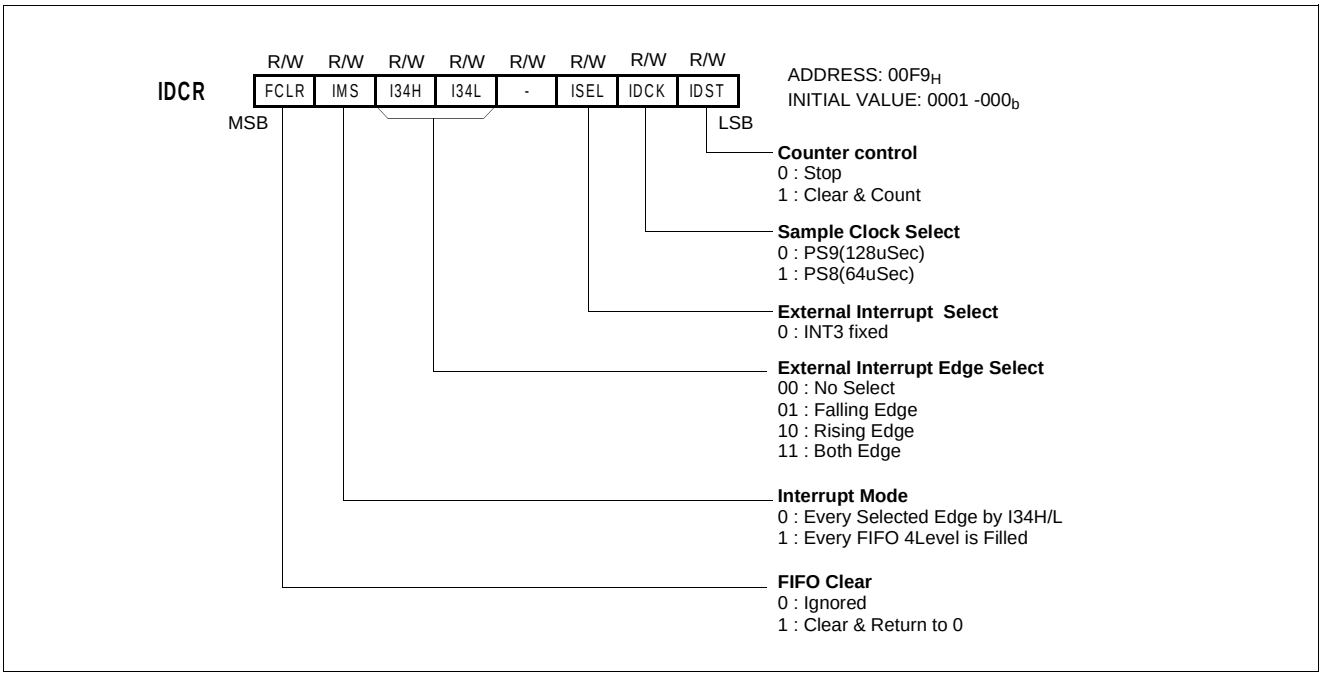


Figure 15-2 Int. interval determination control register

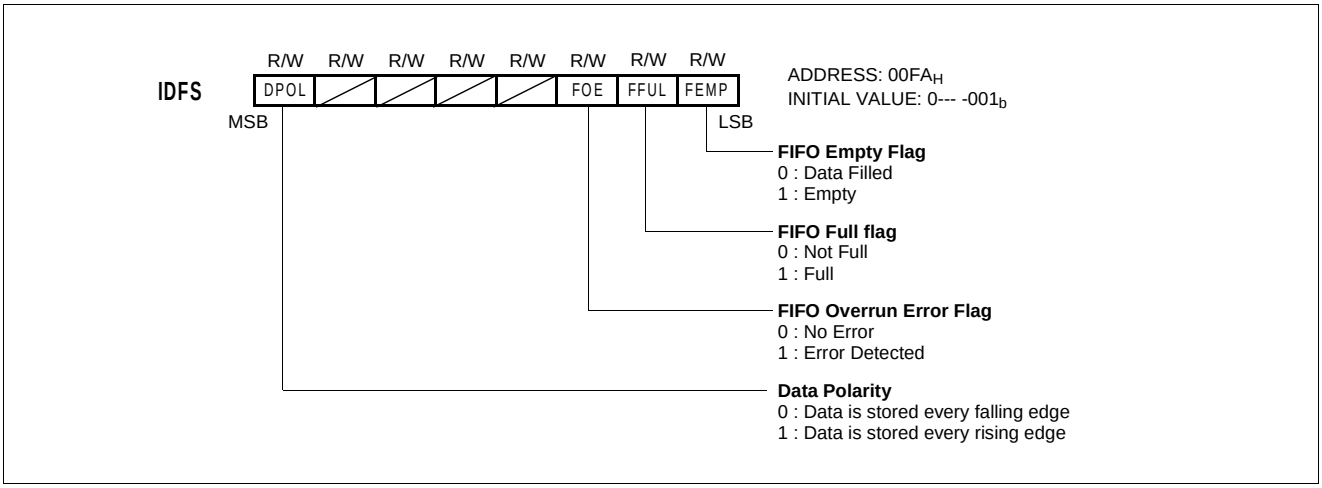


Figure 15-3 Port function select register

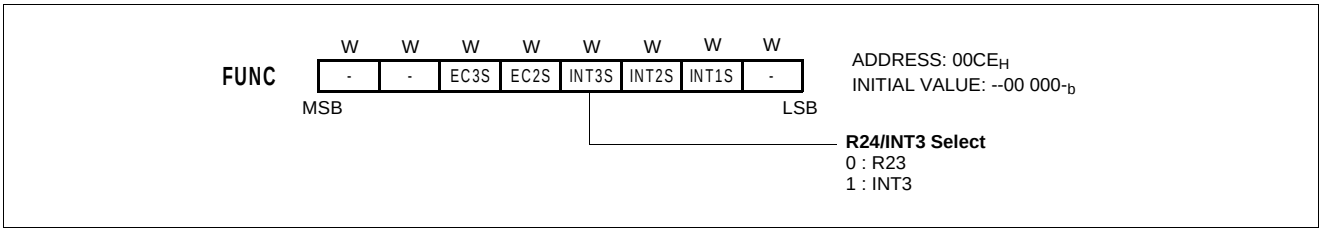
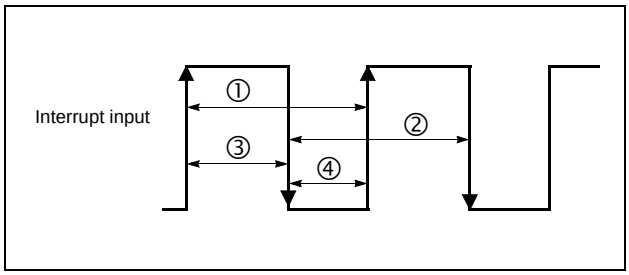


Figure 15-4 Port function select register



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 15-5 Setting for measurement

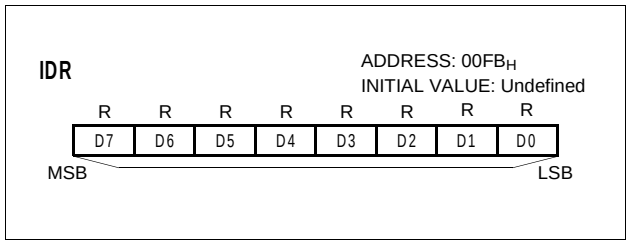


Figure 15-6 INT. interval determination FIFO data register

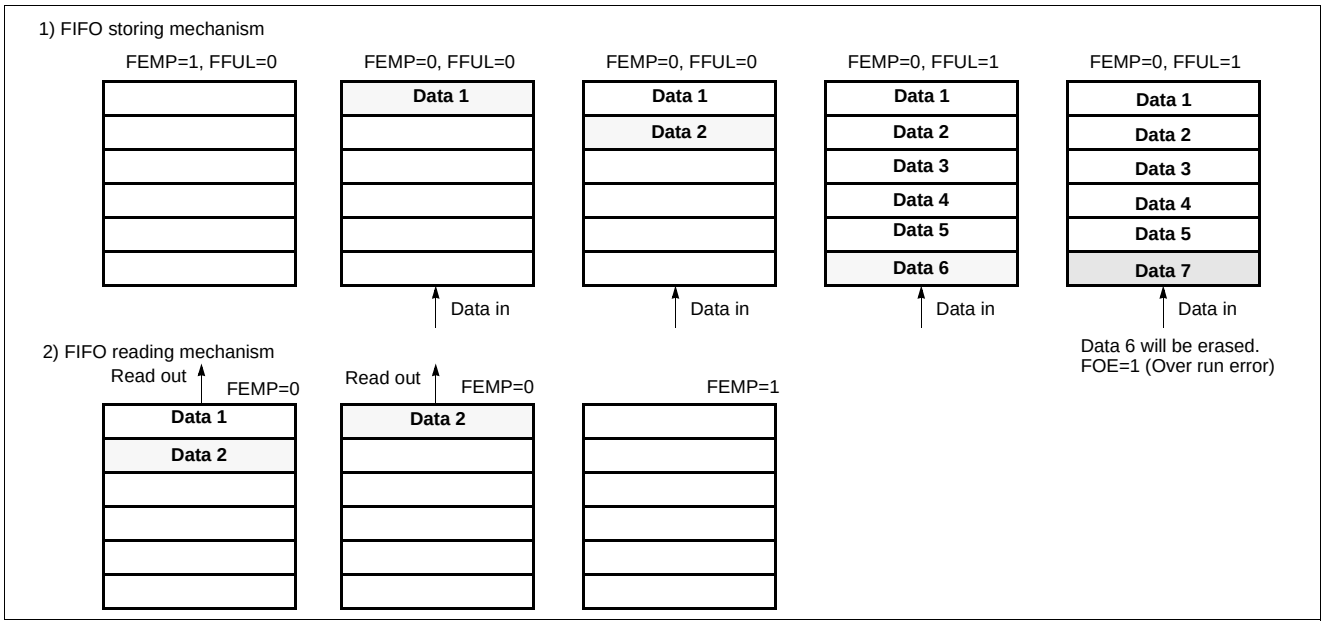


Figure 15-7 Example for FIFO operating mechanism

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 .

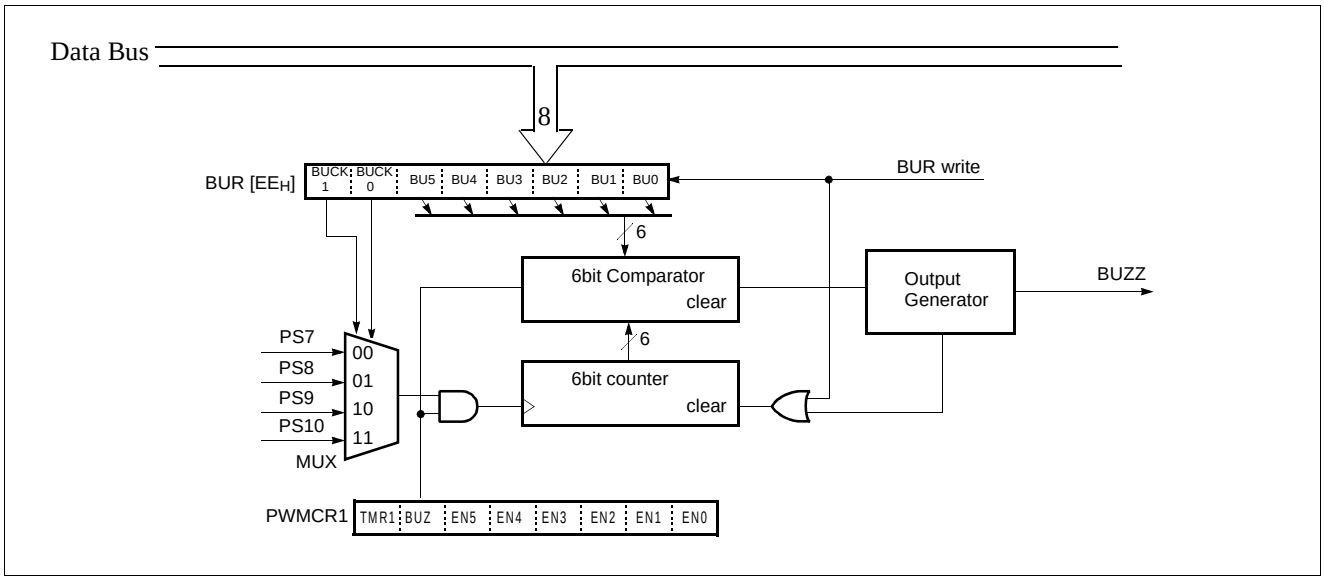


Figure 16-1 Block Diagram of Buzzer driver circuit

Control

The HMS81C4x60 contains a Buzzer driver module.

1. Select an input clock among PS7~PS10 by set the BUCK1~0 of BUR.

BUCK1	BUCK0	Clock source
0	0	PS7
0	1	PS8
1	0	PS9
1	1	PS10

2. Select output frequency by change the BU5~0.
   Output frequency = 1 / (PSx × BUy × 2) Hz.
   x=7~10, y=5~0
   See example Table 16-1.

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

3. Set BUZ bit for output enable.

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

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

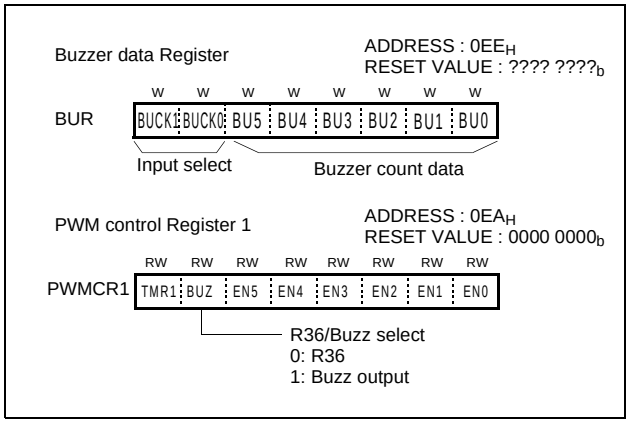


Figure 16-2 Buzzer driver Registers

BUR5~0		Output frequency (KHz)			
Dec	Hex	PS7 (32μS)	PS8 (64μS)	PS9 (128μS)	PS10 (256μS)
1	01	31.25	62.50	125.0	250.0
2	02	15.625	31.25	62.5	125.0
3	03	10.436	20.872	41.744	83.488
4	04	7.813	15.626	31.252	62.504
5	05	6.25	12.50	25.0	50.0
6	06	5.208	10.416	20.832	41.664
7	07	4.464	8.928	17.858	35.716
8	08	3.907	8.814	17.628	35.256
9	09	3.472	6.942	13.884	27.768
10	0A	3.125	6.25	12.5	25.0
11	0B	2.841	5.682	12.364	24.728
12	0C	2.604	5.208	10.416	20.832
13	0D	2.404	4.808	9.616	19.232
14	0E	2.242	4.484	8.968	17.936
15	0F	2.083	4.166	8.332	16.664
16	10	1.953	3.906	7.812	15.624
17	11	1.838	3.676	7.342	14.684
18	12	1.736	3.472	6.944	13.888
19	13	1.644	3.288	6.576	13.152
20	14	1.562	3.124	6.248	12.496
21	15	1.438	2.876	5.752	11.504
22	16	1.420	2.840	5.680	11.360
23	17	1.359	2.718	5.436	10.872
24	18	1.302	2.604	5.208	10.416
25	19	1.25	2.50	5.0	10.0
26	1A	1.202	2.404	4.808	9.616
27	1B	1.158	2.316	4.632	9.262
28	1C	1.116	2.232	4.464	8.928
29	1D	1.078	2.156	4.302	8.604
30	1E	1.042	2.084	4.168	8.336
31	1F	1.008	2.016	4.032	8.064
32	20	0.976	1.952	3.904	7.808
33	21	0.947	1.894	3.788	7.576
34	22	0.919	1.838	3.676	7.352
35	23	0.893	1.786	3.552	7.104
36	24	0.868	1.736	3.472	6.944
37	25	0.845	1.690	3.380	6.760
38	26	0.822	1.644	3.288	6.576
39	27	0.801	1.602	3.204	6.408
40	28	0.781	1.562	3.124	6.248
41	29	0.762	1.524	3.048	6.096
42	2A	0.744	1.488	2.978	5.956
43	2B	0.727	1.454	2.908	5.816
44	2C	0.710	1.420	2.840	5.680
45	2D	0.694	1.388	2.776	5.552
46	2E	0.679	1.358	2.706	5.412
47	2F	0.665	1.33	2.66	5.320
48	30	0.651	1.302	2.604	5.208
49	31	0.638	1.276	2.542	5.184
50	32	0.625	1.25	2.5	5.0
51	33	0.613	1.226	2.452	4.904
52	34	0.601	1.202	2.404	4.808
53	35	0.590	1.18	2.36	4.720
54	36	0.579	1.158	2.316	4.632
55	37	0.568	1.136	2.272	4.544
56	38	0.558	1.116	2.232	4.464
57	39	0.548	1.096	2.192	4.384
58	3A	0.539	1.078	2.156	4.312
59	3B	0.530	1.06	2.12	4.24
60	3C	0.521	1.042	2.084	4.168
61	3D	0.512	1.024	2.048	4.096
62	3E	0.504	1.008	2.016	4.032
63	3F	0.496	0.992	1.984	3.968

Table 16-1 . Example for f_{ex}=4MHz

17. On Screen Display (OSD)

The HMS81C4x60 can support 512 OSD chacters and font size is used 12×10, 12×12, 12×14, 12×16, 16×18. It can support 48 character columns and 2 line buffers respectively and also support full screen OSD when use interrupt. Each characters have bit plane of 24bit and support attribute with OSD line and full screen OSD respectively.

OSD circuit consists of the Position attribute register, Line register, Full screen screen control register, I/O polarity register, font ROM, VRAM, etc. On Screen Display block diagram is shown in Figure 17-1 and the more details about display characters are shown in Figure 17-2.

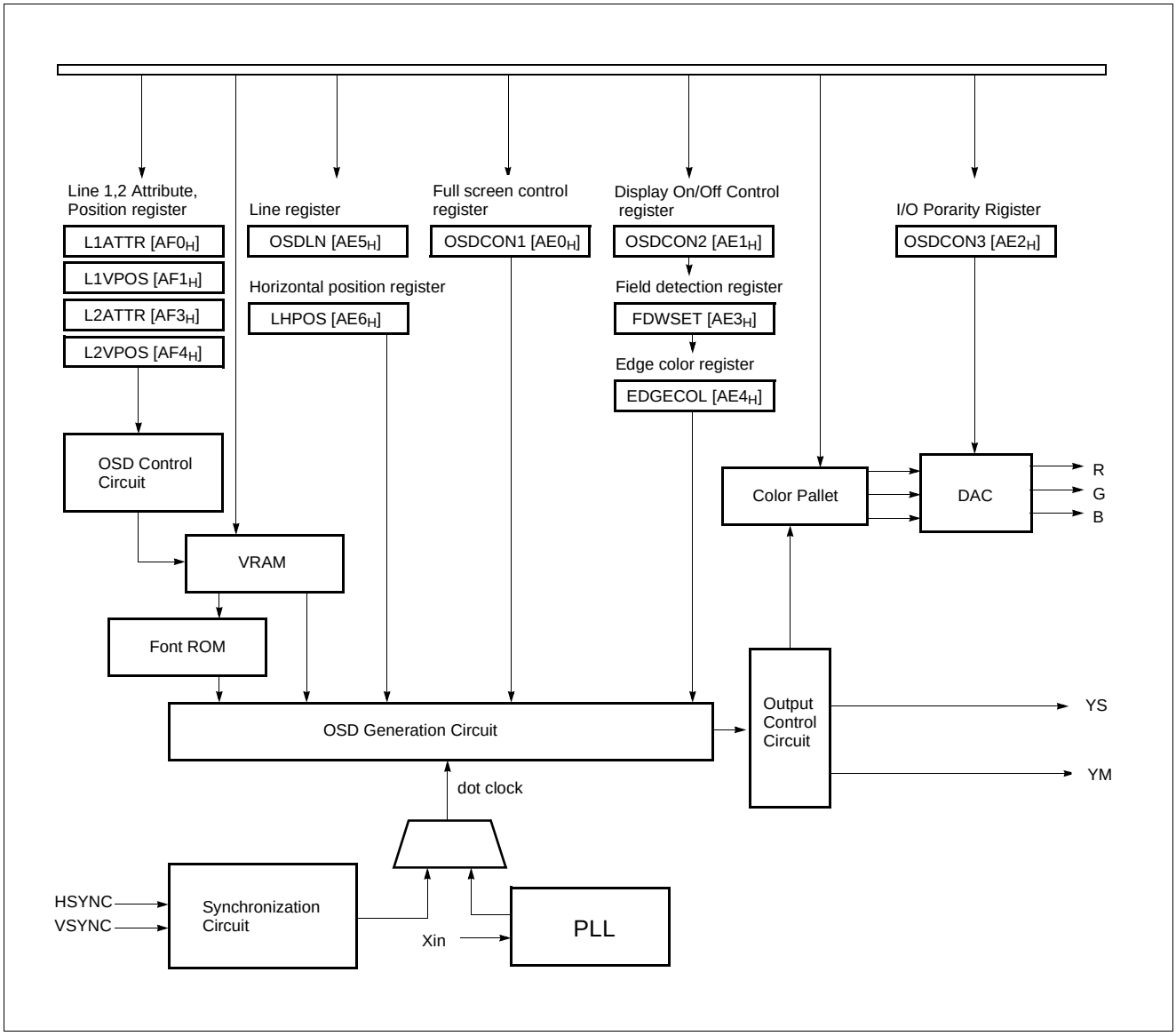
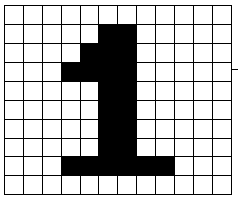
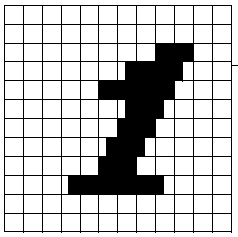


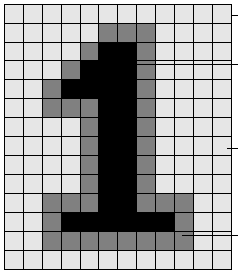
Figure 17-1 Block Diagram of On Screen Display circuit



[12 × 10 Character Font]



[12 × 12 Character Font]
- italic (only 12 × 12 mode can be supported)

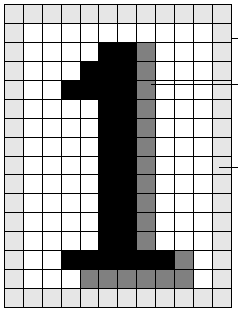


[12 × 14 Character Font]

Foreground Character
- 512 color (8 pallet)
- color selecting : VRAM n-character bit 19~16

Background
- 512 color (8 pallet)
- color selecting : VRAM n-character bit 23~20

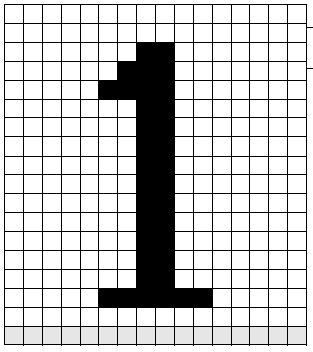
Foreground Character outline
- setting by LnATTR register
- color selecting : EDGECOL register



[12 × 16 Character Font]

Character shadow
- setting by LnATTR register and VRAM n-character bit 9
- color selecting : EDGECOL register

Background shadow color
- setting by VRAM n-character bit 15~12
- color selecting : EDGECOL register
- 512 color (8 pallet)



[16 × 18 Character Font]

OSD background shadow
- Character flash
- background underline

Figure 17-2 OSD Character Font Example

17.1 Feature of OSD

The Feature of OSD shown in below.

- Font pixel matrix

: 12×10, 12×12, 12×14, 12×16, 16×18 dots

- The number of font pattern

: 512 fonts

- Display ability

: 48Character × n lines (multilined by OSD interrupt)

- 8 foreground pallet of 512 colors for each character

- 8 background pallet of 512 colors for each character

- Full screen 8 background color

- Character size

: 3 fonts(2 times, 1.5 times, 1 times)

- Progressive scan line switch

- Attribute

: Outline, Shadow, Rounding

- RGB DAC

: 8 level each color

- Display clock frequency

: 12MHz ~ 64MHz

17.2 OSD Registers

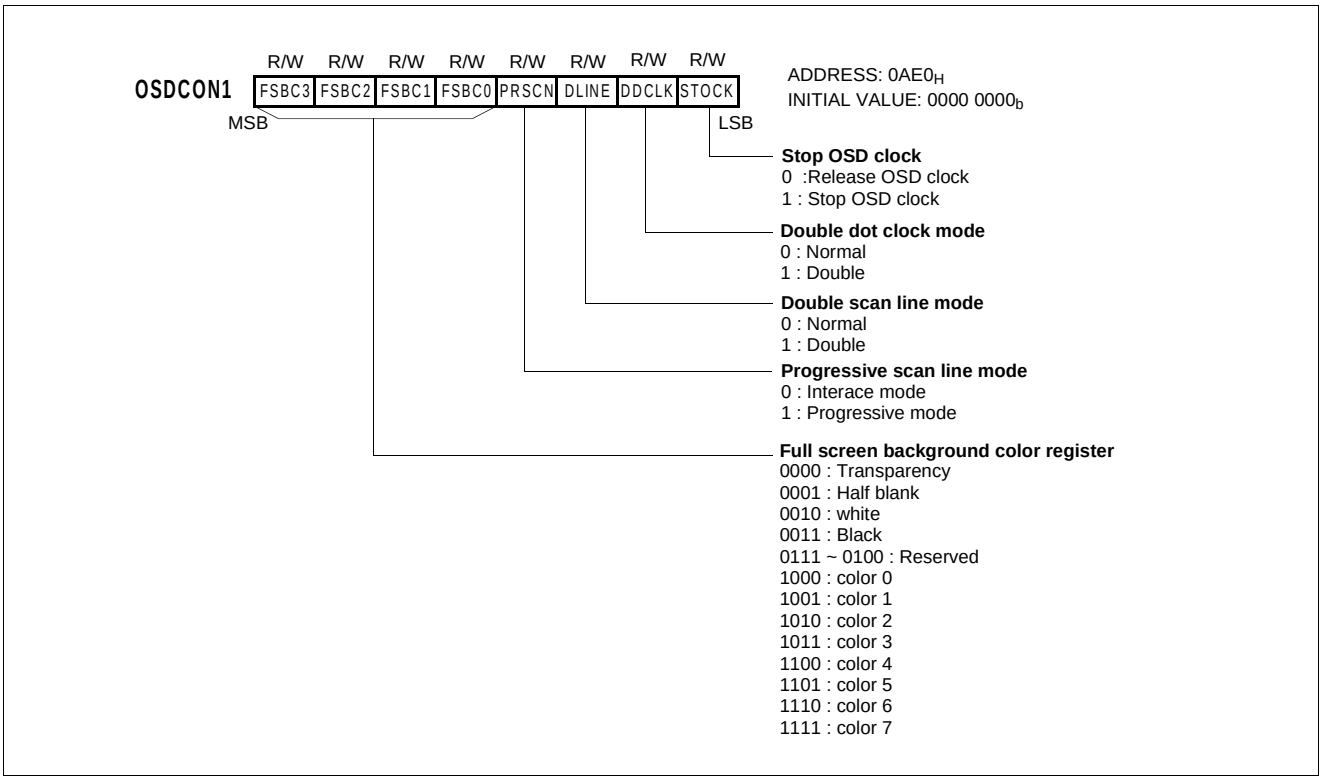


Figure 17-3 OSD Control Registers - 1

OSDCON1

bit 0: STOCK

It stop or start OSD clock. If oscillation is stoped, IC's power consumption is decreased.

bit 1: DDCLK

If you set this bit to 1, OSD input clock is divided by two , than 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 3: PRSCN

It control progressive scan line mode.

bit clear than interace mode and bit set than processive mode.

bit 7~4: FSBC3 ~ FSBC0

It controls full screen background color as figure shows.

NOTE:

Data slicer operate when OSDCON1.PRSCN(0AE0.3) bit of OSD register is cleared. Namely, it operate interace scan display mode.

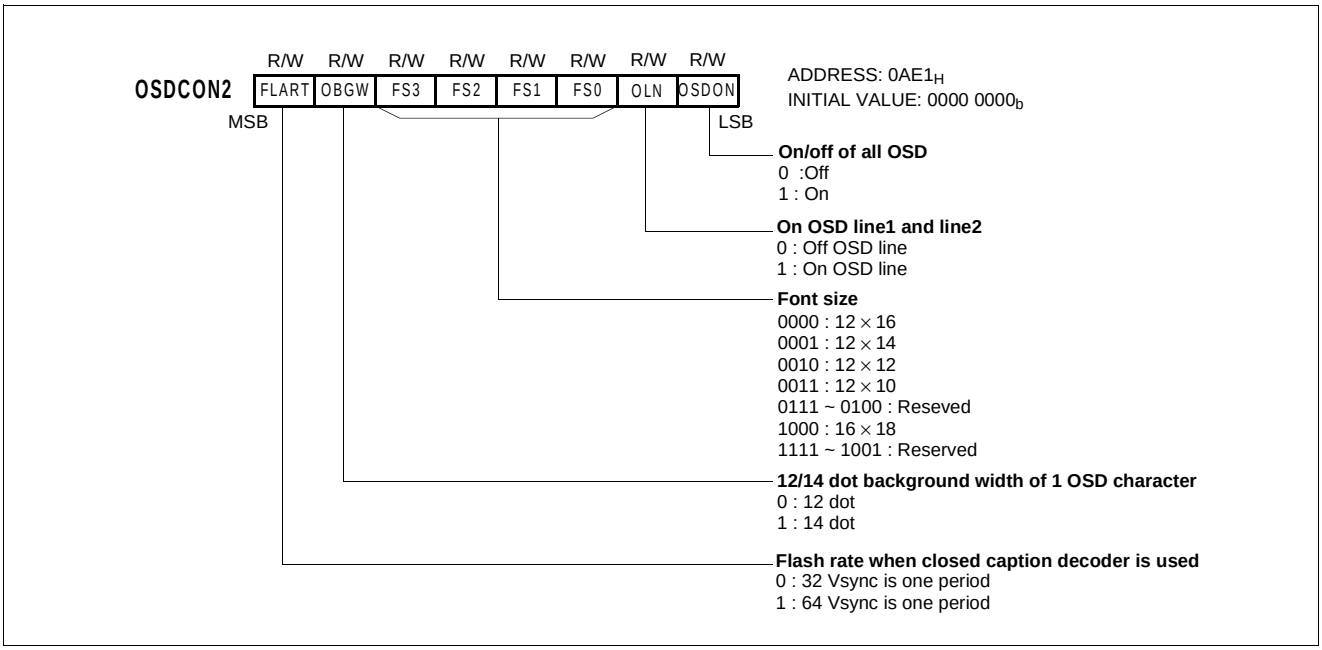


Figure 17-4 OSD Control Register - 2

OSDCON2

bit 0: OSDON

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

bit 1: ONL

It controls OSD line1 and line2 on/off. If its value is 1, OSD line is on.

bit 2 ~ 5: FS0 ~ FS3

It controls OSD font size.

bit 6: OBGW

It controls dot background 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: FLRAT

It controls OSD flash rate when closed caption decoder is used. Bit clear than 32 Vsync is one period and bit set than 64 Vsync is one period.

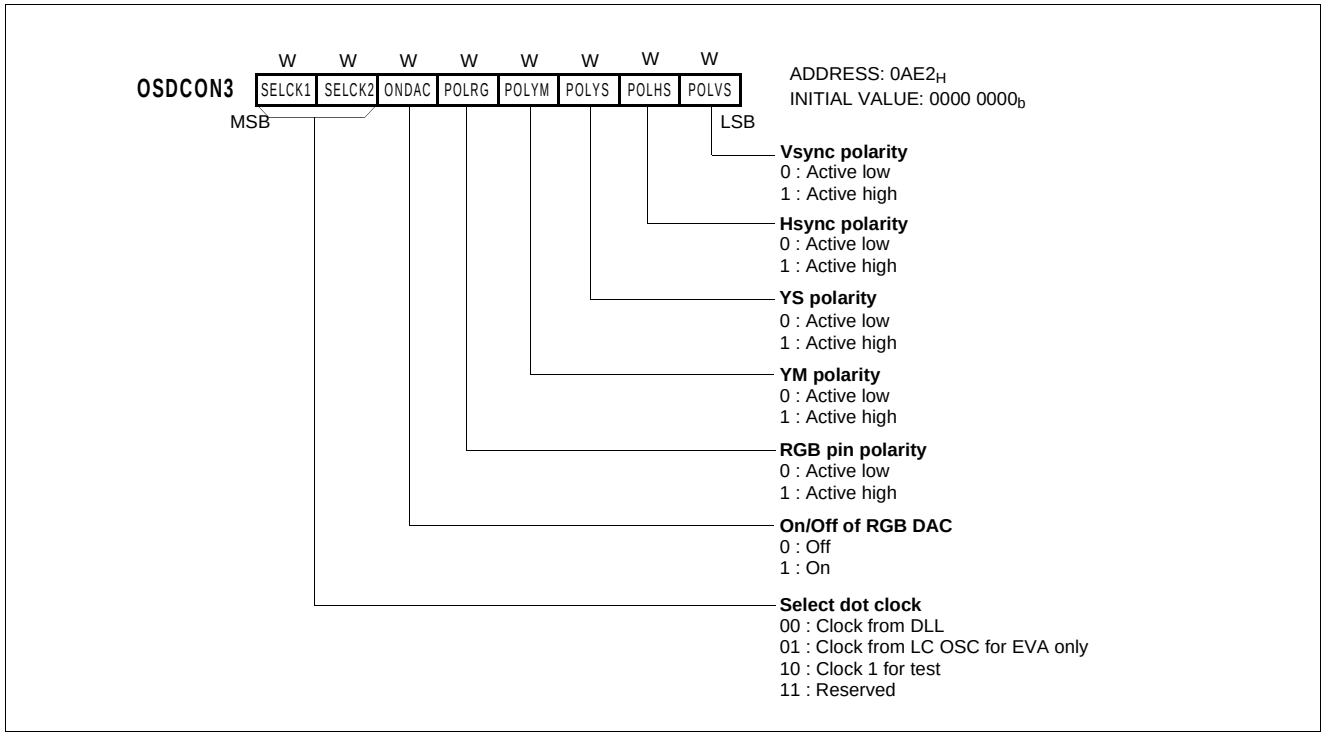
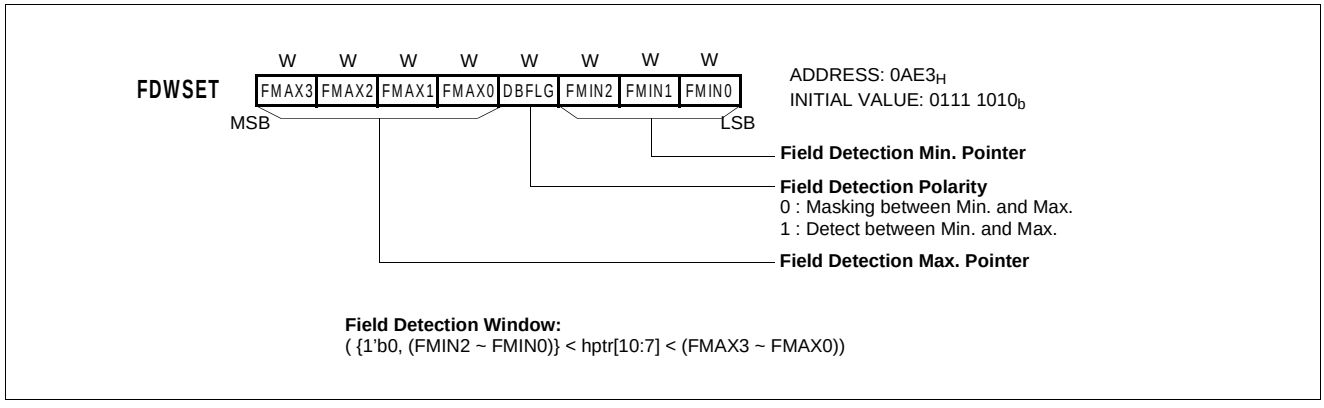


Figure 17-5 I/O Polarity(Initial) Register

OSDCON3

bit7~0 : SELCK1, SELCK0, ONDAC, POLRG, POLYM, POLYS, POLHS, POLVS

It controls Hsync/Vsync polarity, YS/YM polarity, RGB polarity, RGB DAC on/off and select dot clock.



FDWSET

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

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 windows. You can assume there is 4 bit horizontal counter, for example HCOUNT[3:0](hptr[10:7]) which count 0~15.

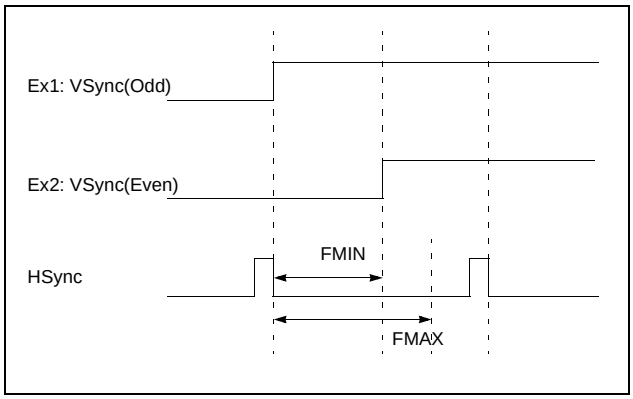


Figure 17-6 FDWSET detection region

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 DBFLG value is 0, it distinguish odd field.

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

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

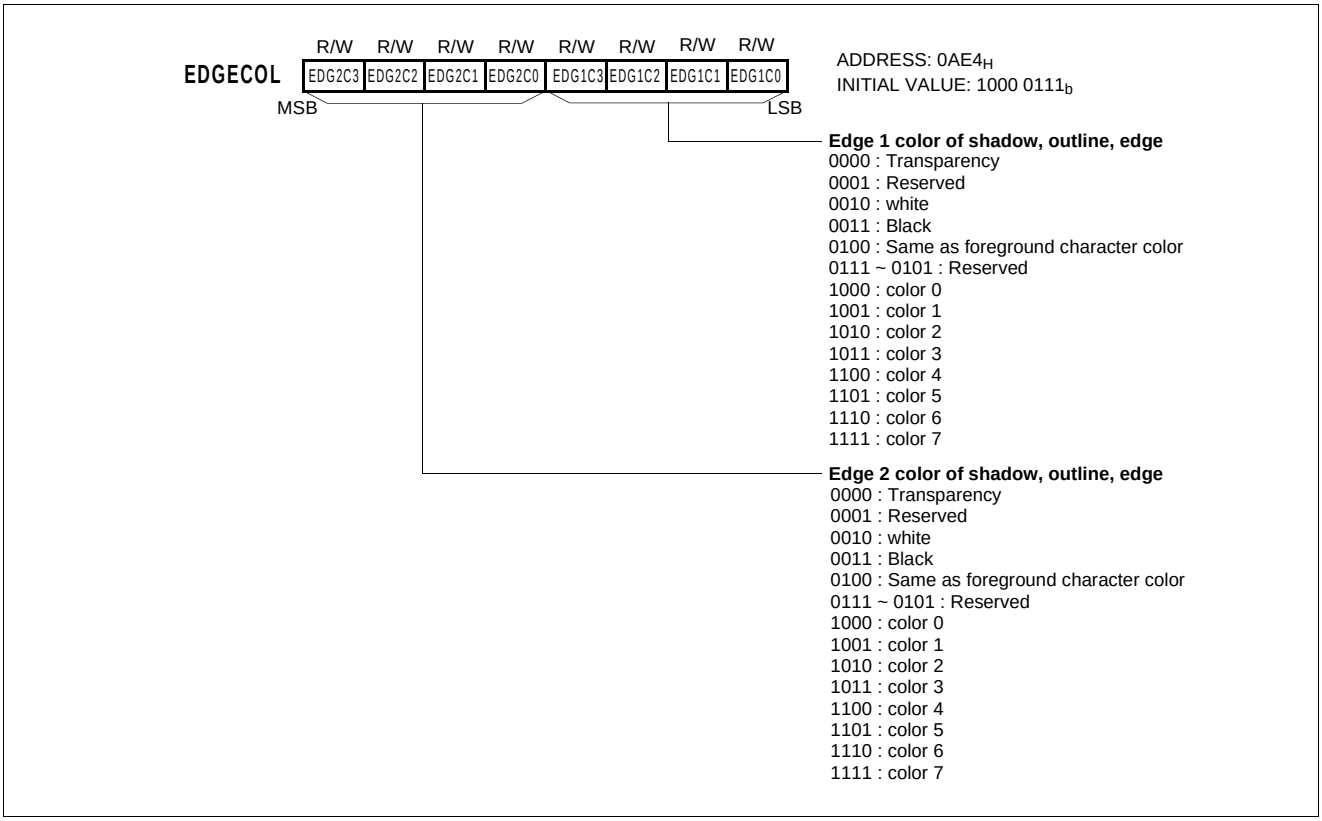


Figure 17-7 Character, Window color Register

EDGECOL

bit 7 ~ bit 0 : EDG1C0,EDG1C1,EDG1C2,EDG1C3

EDG2C0,EDG2C1,EDG2C2,EDG2C3

It control shadow color, outline color and edge color.

Low 4 bits controls edge 1 shadow, outline color and high 4 bits controls edge 2 shadow, outline color.

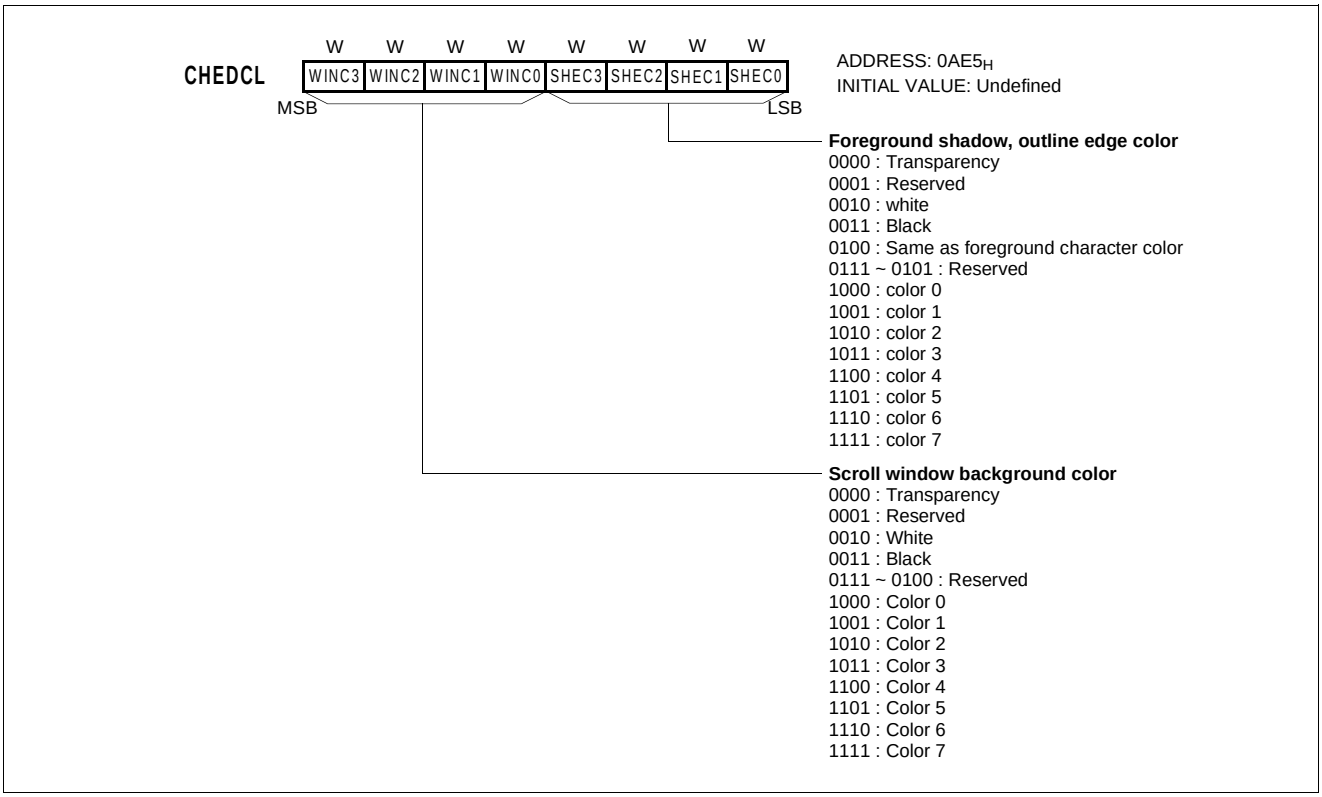


Figure 17-8 Scroll window color Register

CHEDCL

bit 7 ~ bit 0 : SHEC0,SHEC1,SHEC2,SHEC3

WINC0,WINC1,WINC2,WINC3

It controls foreground shadow and outline edge color and scroll window background color.

Low 4 bits controls scroll window background color and high 4 bits controls foreground shadow outline edge color.

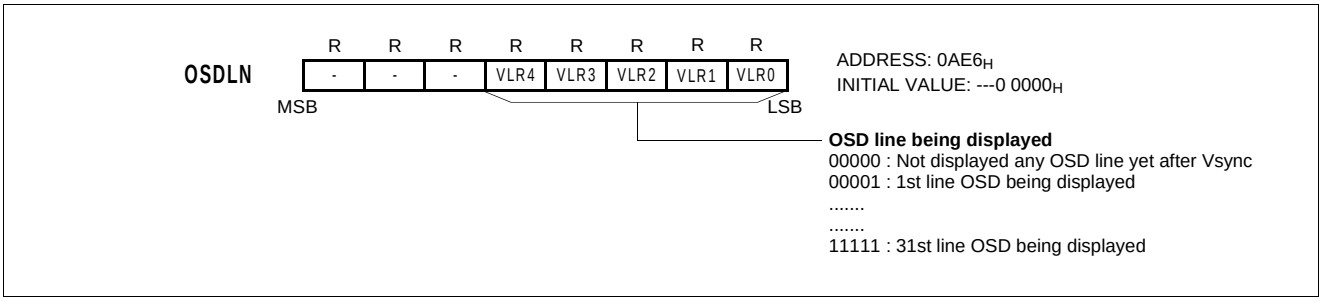


Figure 17-9 OSD Line Register

OSDLN

bit 4 ~ bit 0 : VLR4 ~ VLR0

It shows current display OSD line from 1 to 31.

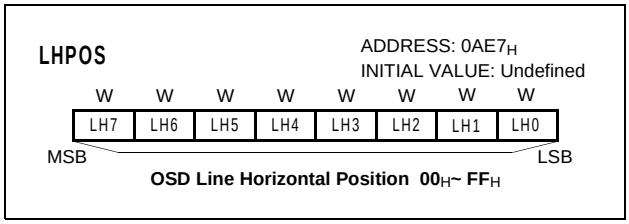


Figure 17-10 OSD Line Horizontal Position Register

LHPOS

bit 7 ~ bit 0 : LH7 ~ LH0

It control OSD line horizontal position. Position value from 00h to FFh.

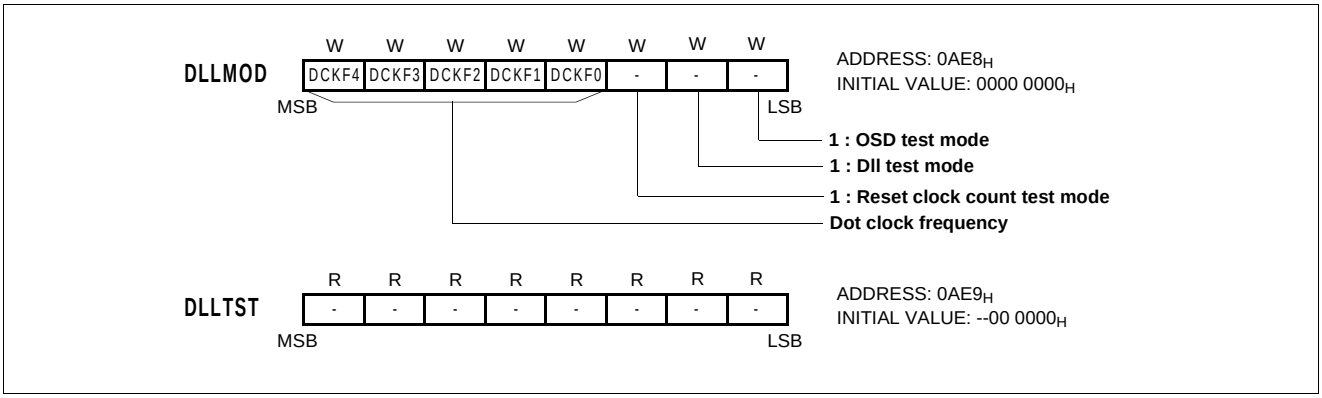


Figure 17-11 DLL mode Register

DLLMOD

bit 2 ~ 0 : If you set this bit to 1, the status is changed test mode.

bit 7 ~ bit 3 : DCKF4 ~ DCKF0

It control dot clock frequency.

Dot clock frequency is as below.

Value					Frequency
DCKF4	DCKF4	DCKF4	DCKF4	DCKF4	
0	0	0	0	0	stop dll clock
0	0	0	0	1	reserved
0	0	0	1	0	reserved
0	0	0	1	1	64.00MHz
0	0	1	0	0	51.20MHz
0	0	1	0	1	42.67MHz
0	0	1	1	0	36.57MHz
0	0	1	1	1	32.00MHz
0	1	0	0	0	28.44MHz
0	1	0	0	1	25.60MHz
0	1	0	1	0	23.27MHz
0	1	0	1	1	21.33MHz
0	1	1	0	0	19.69MHz
0	1	1	0	1	18.29MHz
0	1	1	1	0	17.07MHz
0	1	1	1	1	16.00MHz
1	0	0	0	0	15.05MHz
1	0	0	0	1	14.22MHz
1	0	0	1	0	13.47MHz
1	0	0	1	1	12.80MHz
1	0	1	0	0	12.19MHz
1	0	1	0	1	11.63MHz
1	0	1	1	0	11.13MHz
1	0	1	1	1	10.67MHz
1	1	0	0	0	reserved
1	1	0	0	1	reserved

Table 17-1 Dot Clock Frequency (f_{ex}=4Mhz)

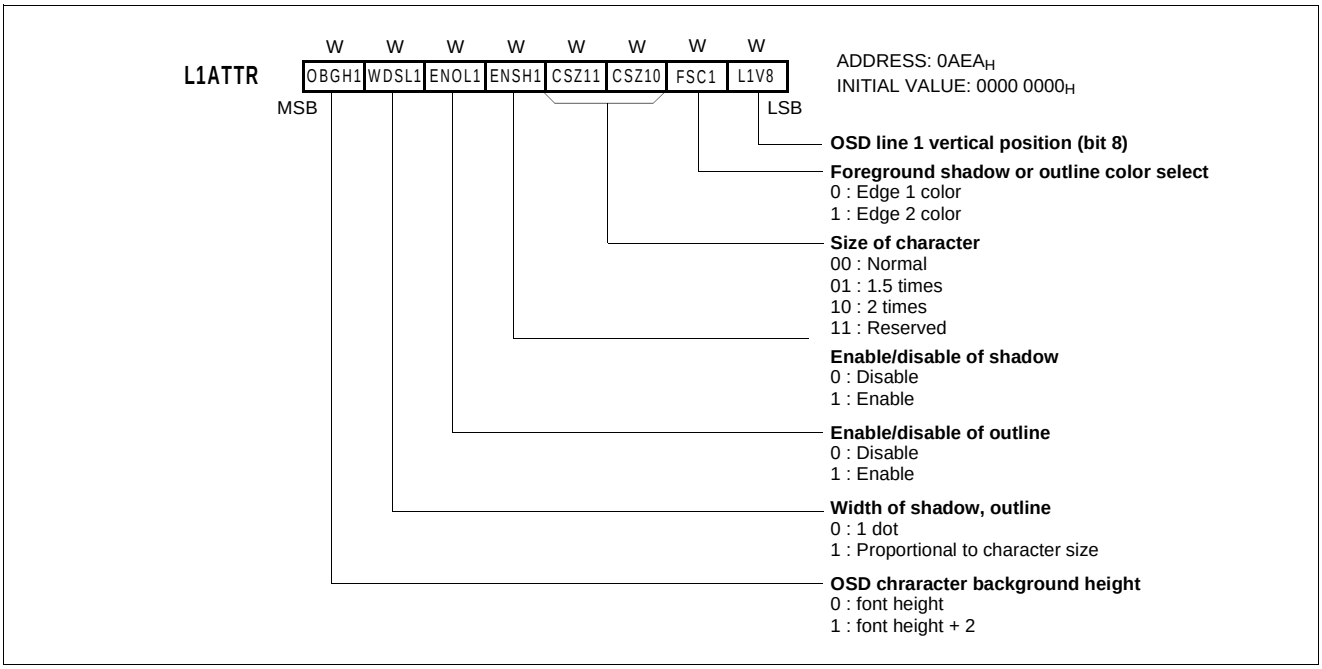


Figure 17-12 OSD line 1 attribute register

L1ATTR

bit 0 : L1V8

It is equivalent to L1VPOS's most significant bit(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 of 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 (normal, 1.5 times, 2 times). You can use this register and DDCLK, DLINE bit, horizontal / vertical size can be controlled (x1, x1.5, x2).

bit 4: ENSH1

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

bit 5: ENOL1

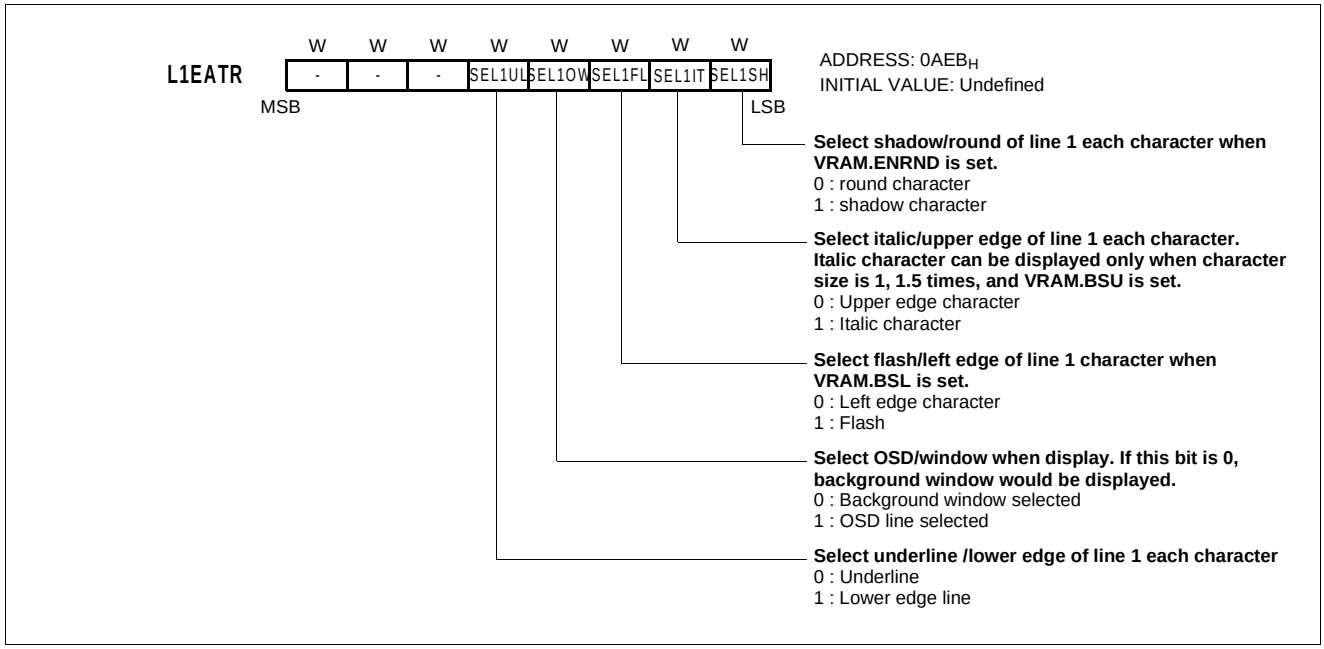
It enables line 1's character(foreground) outline.

bit 6: WDSL1

It shows thickness of line 1's shadow and outline.This bit is set than one dot and bit clear is proportional to character size. If only character size is 2 times, 2 times per vertically and horizontally. In case 1 dot width would be enable.

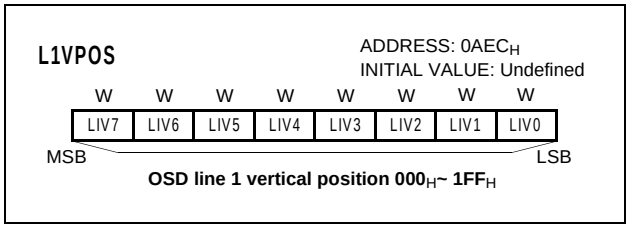
bit 7: OBGH1

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



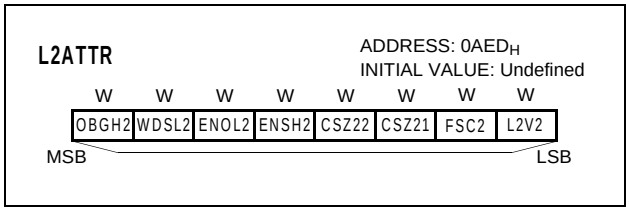
L1EATR

It shows OSD line 1 extend attribute register.



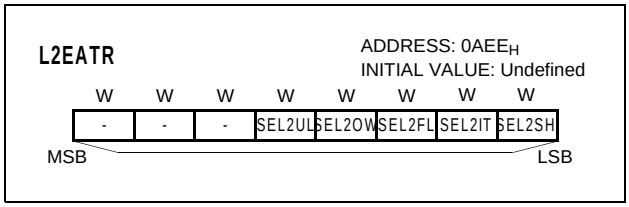
L1VPOS

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



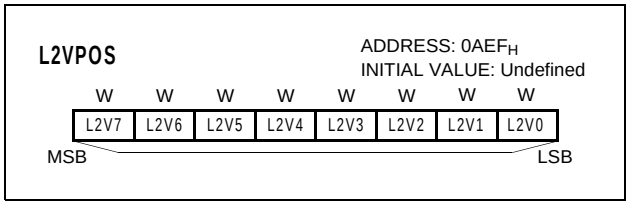
L2ATTR

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



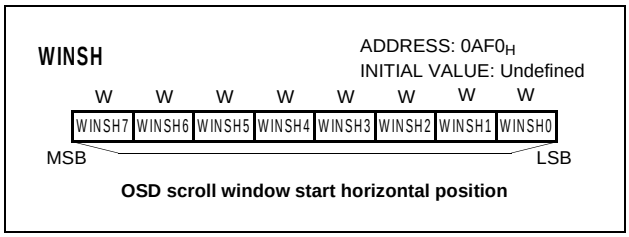
L2EATR

It shows OSD line 2's extened attribute register.



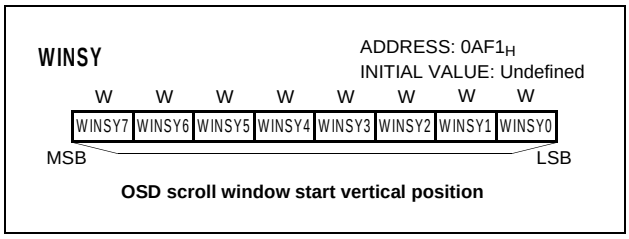
L2VPOS

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



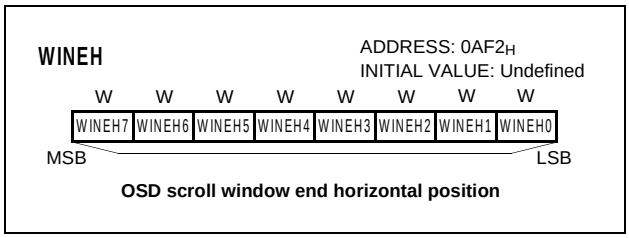
WINSH

It shows OSD scroll window start horizontal position.



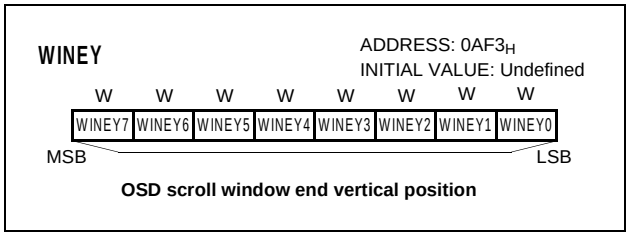
WINSY

It shows OSD scroll window start vertical position.



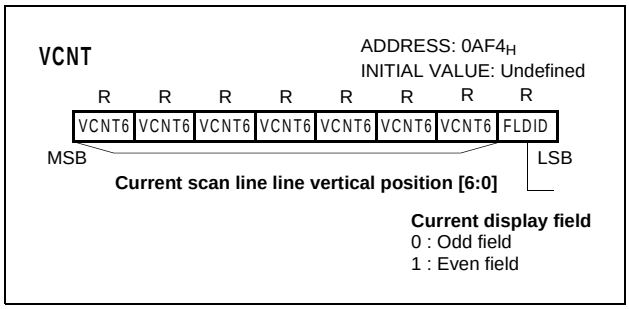
WINEH

It shows OSD scroll window end horizontal position.



WINEY

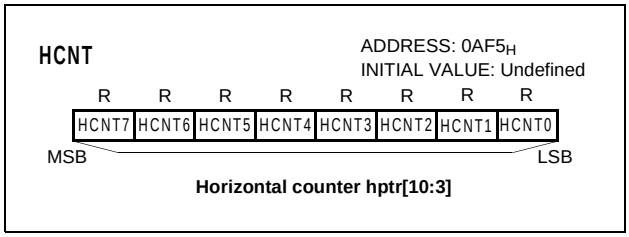
It shows OSD scroll window end vertical position.



VCNT

It shows Vsync count register and counted by pixel clock.

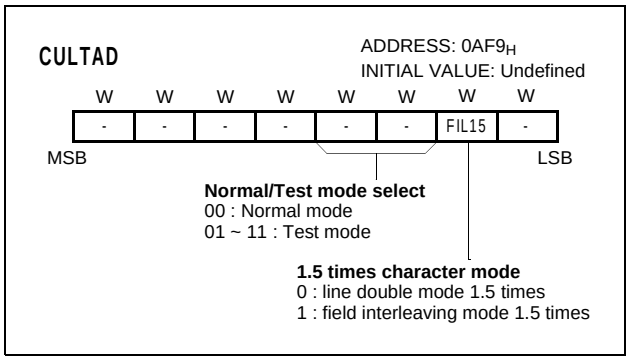
VCNT counter clock start at Vsync start edge.



HCNT

It shows Hsync count register and counted by pixel clock.

HCNT counter clock start at Hsync start edge.



CULTAD

It shows normal and test mode and 1.5 times mode.

17.3 VRAM

VRAM contains a OSD line buffer, 48 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 add. No.	Address (bit 47~0) Hexa decimal		
1	1	A80	A40	A00
	2	A81	A41	A01
	3	A82	A42	A02
	:	:	:	:
	46	AAD	A6D	A2D
	47	AAE	A6E	A2E
	48	AAF	A6F	A2F
2	1	B80	B40	B00
	2	B81	B41	B01
	3	B82	B42	B02
	:	:	:	:
	46	BAD	B6D	B2D
	47	BAE	B6E	B2E
	48	BAF	B6F	B2F

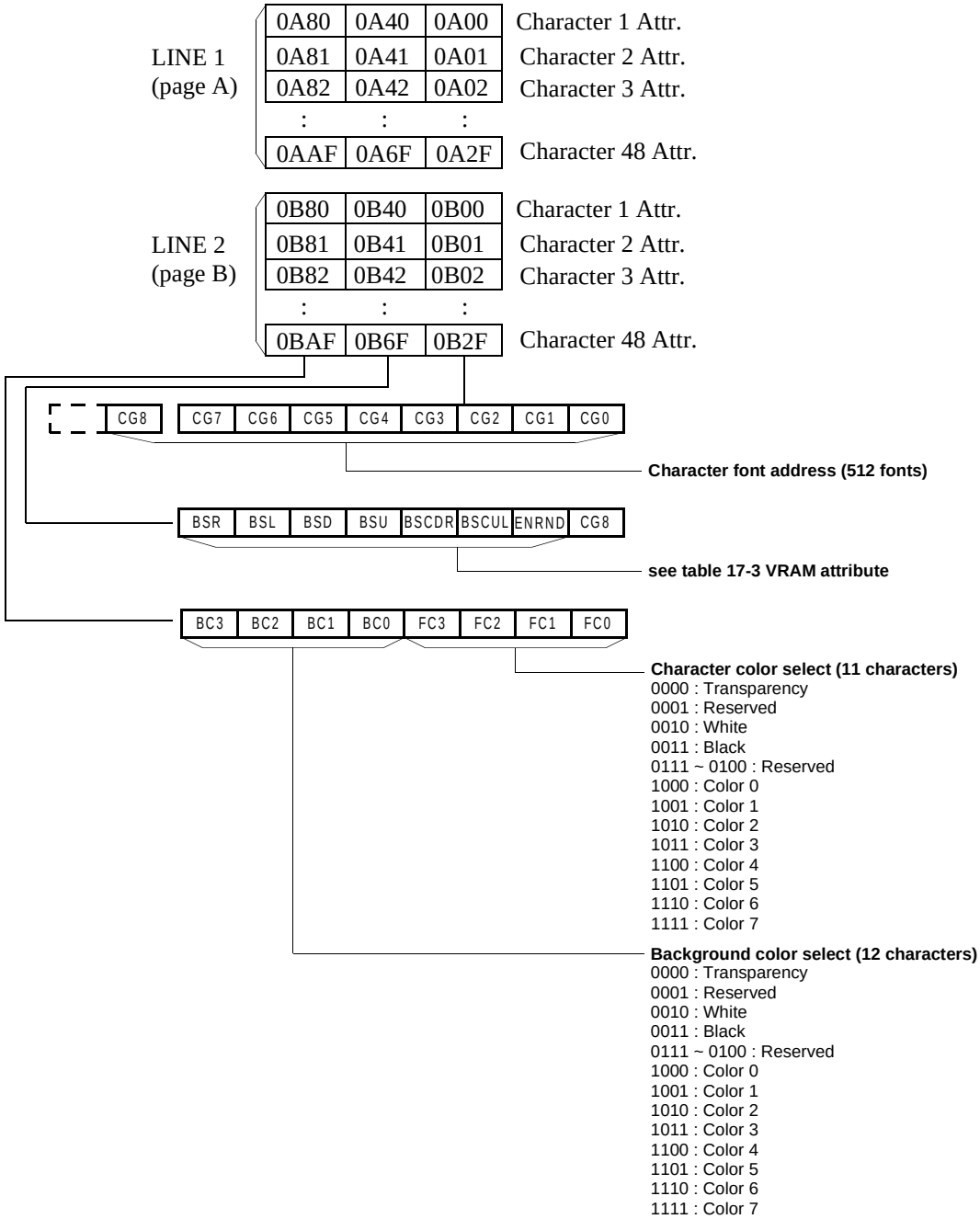
Table 17-2 VRAM memory map

Bit No.	Name	Function
00~08	CG8 ~CG0	Character font code 1FFh ~ 000h
09	ENRND	Round enable/disable 0 : disable 1 : enable
0A	BSCUL	Edge color of upper and left background shadow edge 0 : edge 1 color 1 : edge 2 color
0B	BSCDR	Edge color of lower and right background shadow edge 0 : edge 1 color 1 : edge 2 color
0C	BSU	Background shadow upper eddge control/italic depend on LxEATR.SELxIT 0 : disable 1 : enable if(LxEATR.SELxIT == 0) background shadow upper edge enable else(LxEATR.SELxIT == 1) italic enable
0D	BSD	Background shadow lower edge control/underline depend on LxEATR.SELxUL 0 : disable 1 : enable if(LxEATR.SELxUL == 0) background shadow lower edge enable else(LxEATR.SELxUL == 1)underline enable
0E	BSL	Background shadoww left edge control/flash(blInking) depend on LxEATR.SELxFL 0 : disable 1 : enable if(LxEATR.SELxFL == 0) background shadow left edge enable else(LxEATR.SELxFL == 1) flash(flicking) enable
0F	BSR	Background shadow right edge control 0 : disable 1 : enable
10~13	FC3 ~FC0	Foreground color for character (11 colors)
14~17	BC3 ~BC0	Background color for character (12 colors)

Table 17-3 VRAM attribute

Composition of VRAM

RESET VALUE: Undefined



17.4 Character ROM

The HMS81C4x60 Character ROM are used 512 types of Font Dot Pattern data. As displayed one character, need 12 × 10 ~ 16 × 18bits Dot Pattern data.

1. Each horizontal data (12dots) needs 2bytes ROM.

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

3. HMS81C4x60 contains 512 characters. Total Font ROM memory size is calculated as 16,384bytes (32bytes / character × 512 characters)

4. Font ROM memory is located from 10000_H ~ $17FFF_H$, this memory can not be accessed by user program.

Character code	Address range	
	Upper 8bit	Lower 8bit
000_H	14000_H ~ 14011_H	10000_H ~ 10011_H
001_H	14020_H ~ 14031_H	10020_H ~ 10031_H
002_H	14040_H ~ 14051_H	10040_H ~ 10051_H
:	:	:
xyz_H	(14000H + xyz0H) ~ (14000H + 2*xyzFH)	(10000H + xyz0H) ~ (10000H + 2*xyzFH)
:	:	:
$1FD_H$	$17FA0_H$ ~ $17FB1_H$	$13FA0_H$ ~ $13FD1_H$
$1FE_H$	$17FC0_H$ ~ $17FD1_H$	$13FC0_H$ ~ $13FD1_H$
$1FF_H$	$17FE0_H$ ~ $17FF1_H$	$13FE0_H$ ~ $13FF1_H$

Table 17-4 Font ROM memory map

5. A character's address and dot position in font ROM is described in Figure 17-13.

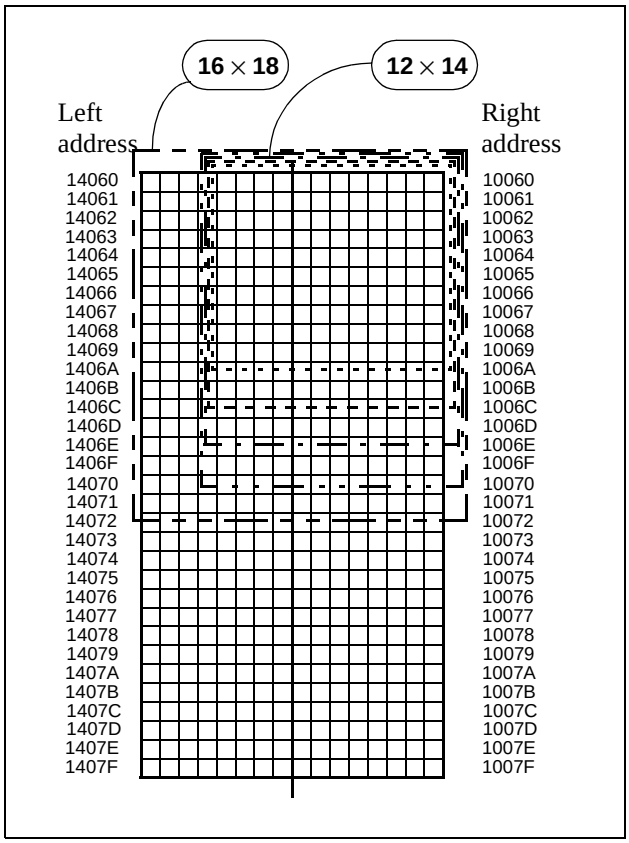


Figure 17-13 Character Dot Pattern

17.5 Color Look Up Table

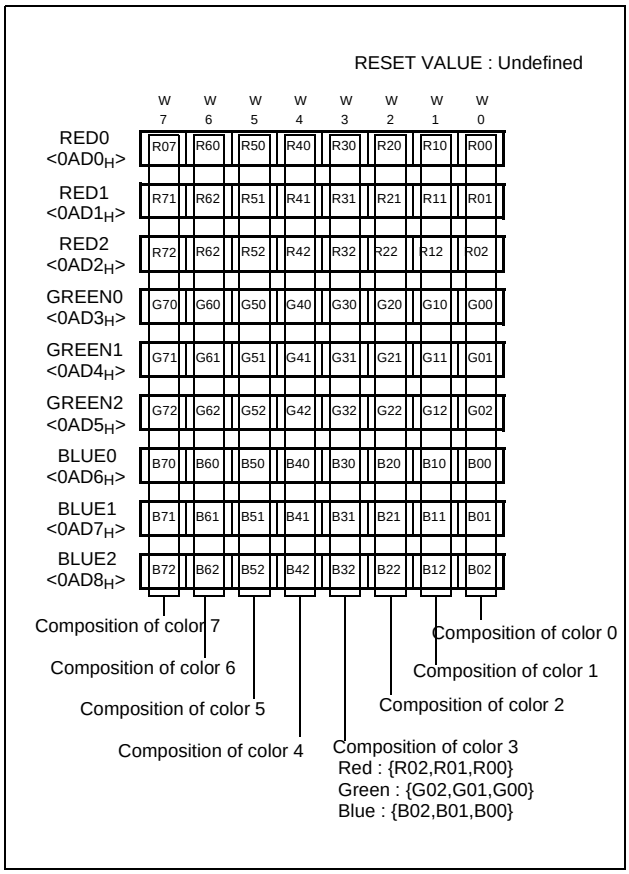


Figure 17-14 Color look up table

[Example] Color data table

```
Color_example_table:
   db  0000_0000b  ;color 0 = Gray
   db  0000_0011b  ;color 1 = Red
   db  0010_1011b  ;color 2 = Green
   ;
   db  0000_0000b  ;color 3 = Yellow
   db  0000_0101b  ;color 4 = Blue
   db  0100_1101b  ;color 5 = Magenta
   ;
   db  0000_0000b  ;color 6 = Cyan
   db  1001_0001b  ;color 7 = half blue
   db  1111_0001b
```

18. DATA SLICER

HMS81C4x60 supports Closed Caption decoding standard with 0.5MHz data rate. Also it can capture 4 horizontal lines information per frame, because it has 4 horozontal lines capture memory. It is able to select even or odd field at one field interval. Data Slicer captures caption information from line 21 in vertical blanking interval of CVBS, and stores these data to buffer memory.

18.1 Data Slicer Circuit

Figure 18-1 shown the data slicer circuit.

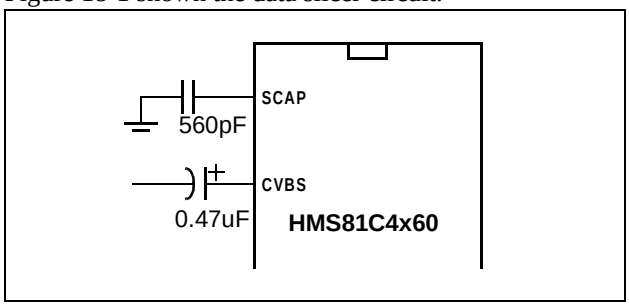


Figure 18-1 Data Slicer Circuit

CVBS signal is entered to CVBS pin via 0.47uF capacitor. The black level of signal is about 2V. SCAP pin is connected to external 560pF capacitor which adjust the referance voltage of comparator. Its slicer level is adapted to input signal.

18.2 Configuration of Data Slicer

Figure 18-2 shows the block diagram of the Data Slicer.

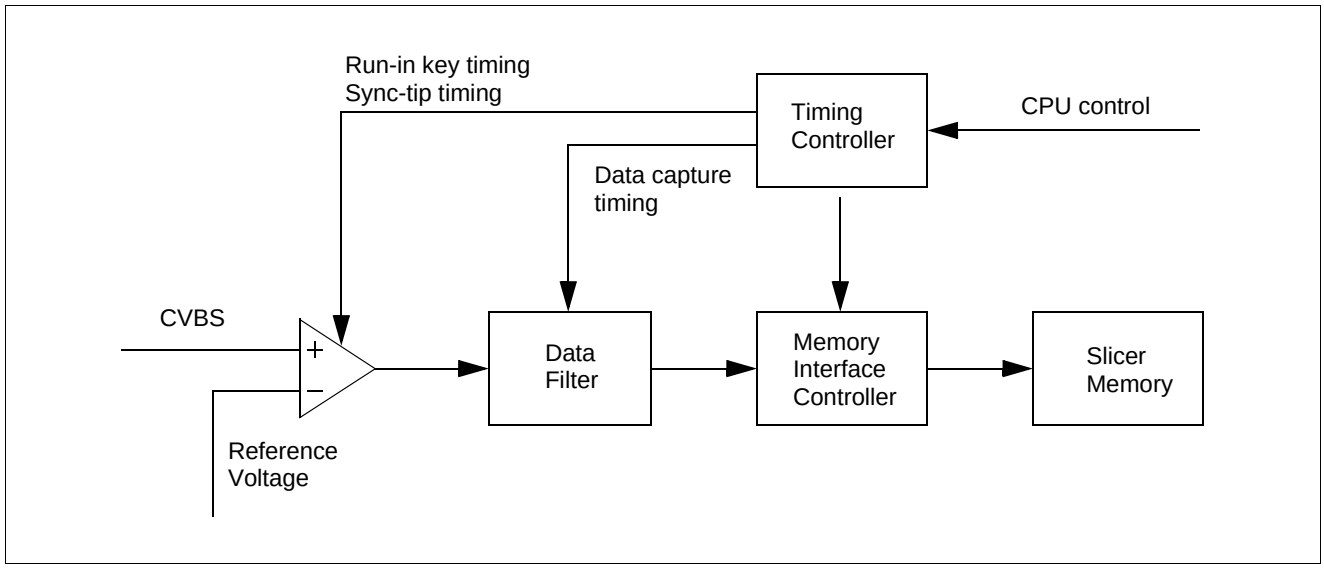


Figure 18-2 Data Slicer Block Diagram

This data slicer block separates caption information from CVBS signal. Data slicer composes high speed comparator and on-chip low pass filter. The output data of comparator is stored in memory through the filter and memory interface controller, which should be decoded to caption data by software. Slicer memory addressed 600h ~ 6FFh.

18.3 Slicer Registers

Slicer Control Register

Slicer Control Register is the specific control register, which select operating frequency of the slicer, slicer decoding method and switch slicer on/off.

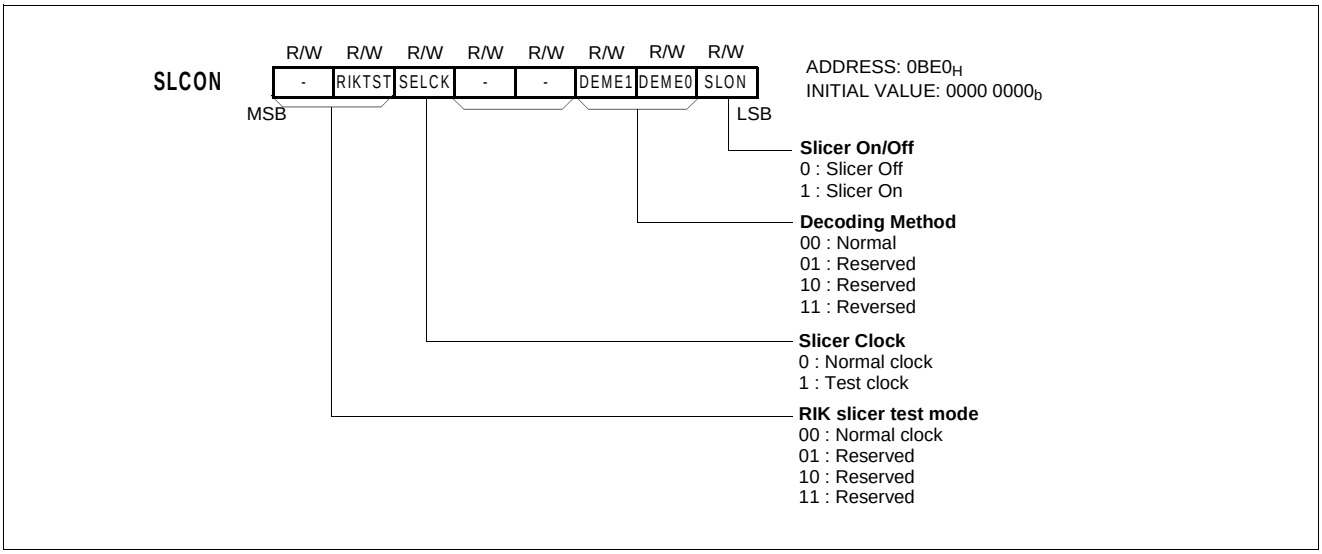


Figure 18-3 Slicer Control Register

Slicer Information Register 0

Slicer Information Register 0 selects even or odd field buffer of line 0 and slicer line 0 position. Also it is used to select line number in Vertical blanking interval.

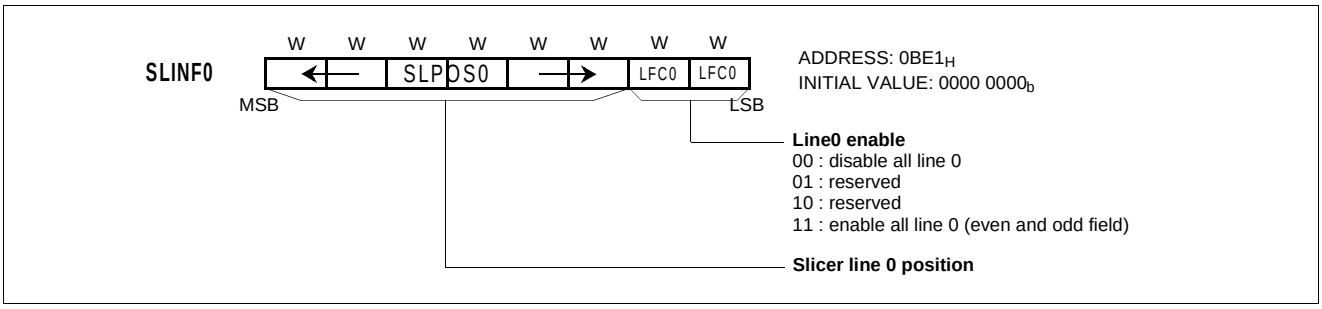


Figure 18-4 Slicer Information Register 0

Slicer Information Register 1

Slicer Information Register 1 selects even or odd field buffer of line 1 and slicer line 1 position. Also it is used to select line number in Vertical blanking interval.

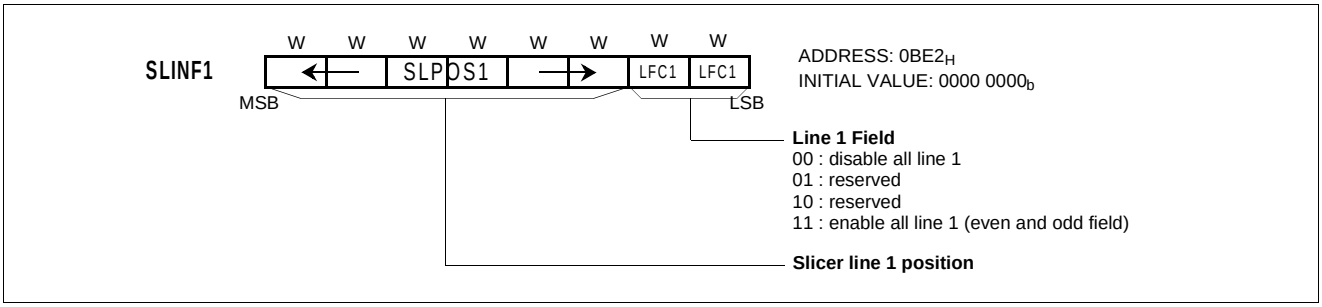


Figure 18-5 Slicer Information Register 1

Run-in key Start/End position Register

RIKST points the start postion of run-in key, it is delayed from start edge of Hsync. RIKED points the end position of run-in key, it is also delayed from start edge of Hsync. Both timmings are counted up by 8MHz clock. The referance voltage of comparator is charged by external signal during this time interval. Figure 18-6 and Figure 18-7 shows the RIK register's configure.

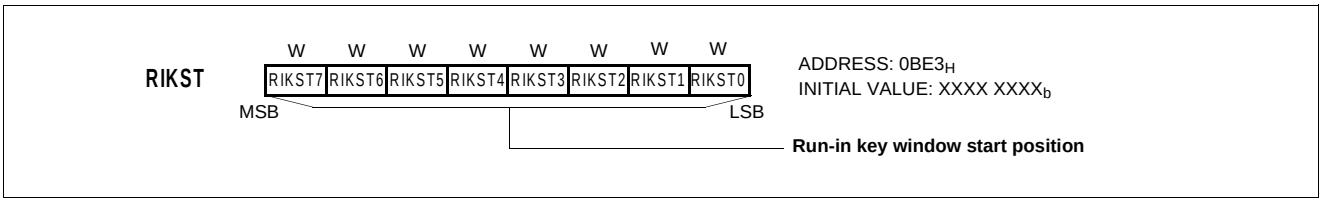


Figure 18-6 Run-in key Start Position Register

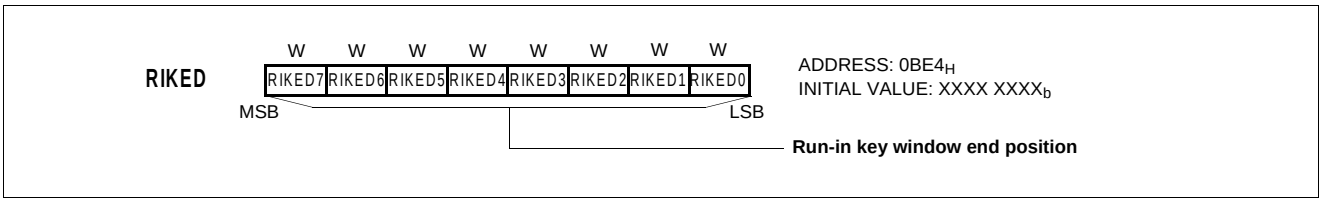


Figure 18-7 Run-in key End Position Register

Sync Start/End Position Register

Sync Start and End position Register are used to make Sync tip window. Both timmings are counted up by 16MHz clock. Figure 18-8 and Figure 18-9 shows the Sync-tip register's configure.

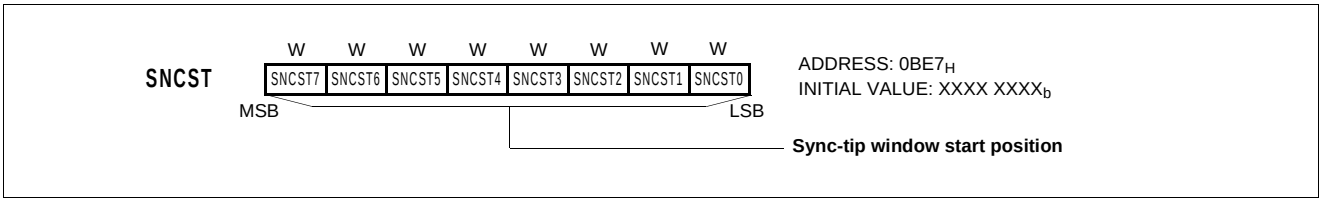


Figure 18-8 Sync-tip start position register

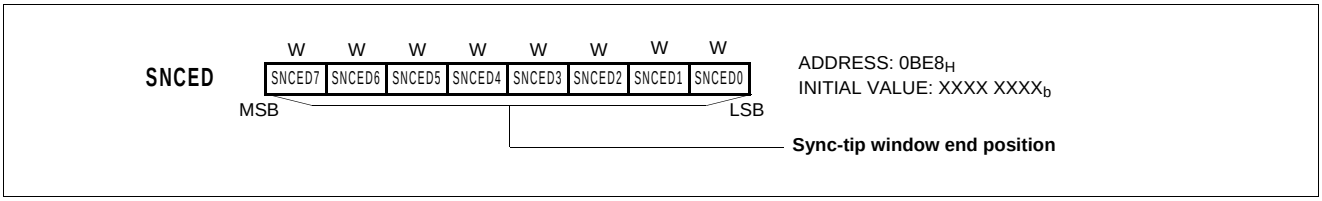


Figure 18-9 Sync-tip end position register

18.4 Data Sampling

Line 21 Closed Caption signal

Figure 18-10 shows the closed caption signal. The signal composes color burst, clock run-in, start bit(001), 16bit ASCII data with 2 parity bit. Sliced raw datas are sampled by 4MHz frequency.

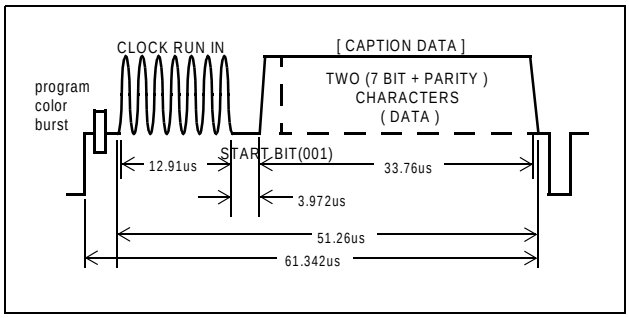


Figure 18-10 Closed caption signal

Address assign

Table 18-1 shows the map of assigned buffer memory.

Setting		Address
First Line	Even Field	0600h ~ 063Fh
	Odd Field	0640h ~ 067Fh
Secont Line	Even Field	0680h ~ 06BFh
	Odd Field	06C0h ~ 06FFh

Table 18-1 Address assign

Interrupt occurrence

The slicer interrupt is occured after writing the sliced two lines data to memory buffer.

Signal timing

Figure 18-11 shows an example of variable signals, which includes Vsync(vertical Sync.), Hsync(horizontal Sync.), CVBS(composit video in), SCAP(slicer capacitor), Run-in key and Sync tip. Line 21 closed caption signal run after Vsync interrupt. The signal's black(base) level voltage is charged on Sync-tip switch-on period, and the referance voltage of comparator is charged on RIK switch-on perid. Because RIK time is related to SCAP voltage(comparator referance voltage or slicer level) which is charged by clock run-in signal, user can adjust the slicer level by RIK time. The sliced data is stored to RAM buffer. (0600h~ 06FFh)

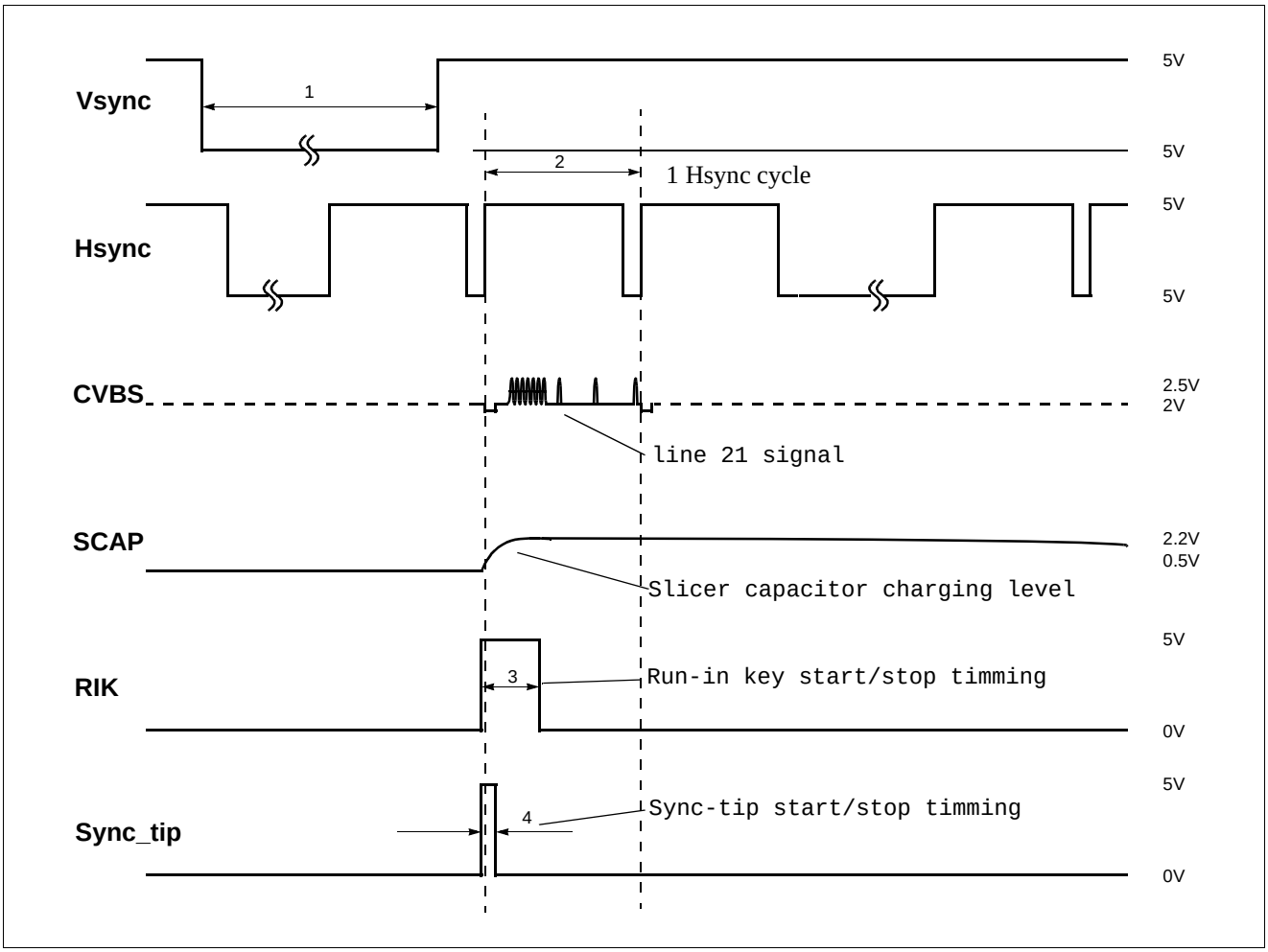


Figure 18-11 Signal timing

[Example]

Initializing slicer register.

```
CCD_INIT:  LDM     SLINF0,#0011_0011b      ; slicer line 21
           LDM     SLINF1,#0000_0000b      ; no field
           LDM     RIKST,#01               ; run-in key start : 1 -> 0.125uS(8MHz)
           LDM     RIKED,#8Ch              ; run-in key end : 8ch -> 17.5uS(8MHz)
           LDM     SNCST,#01               ; sync tip start : 1 -> 0.0625uS(16MHz)
           LDM     SNCED,#58h              ; sync tip end : 58h -> 5.5uS(16MHz)
           LDM     SLCON,#01h              ; normal clock, 16MHz, slicer start
```

19. I²C Bus Interface

The I²C Bus interface circuit is shown in Figure 19-1.

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 19-2~ Figure 19-5.

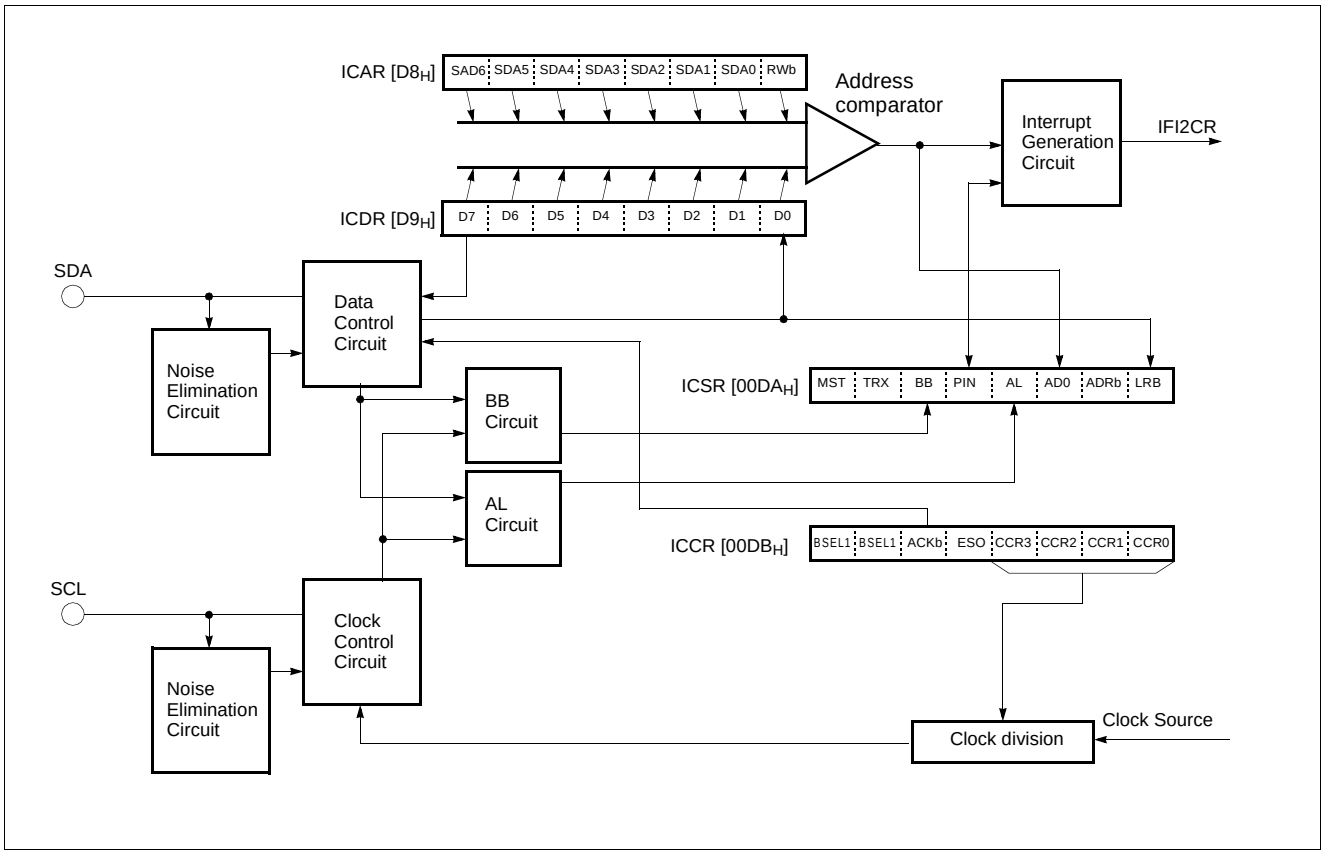


Figure 19-1 Block Diagram of multi-master I²C circuit

Control

The HMS81C4x60 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

I²C address register

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

Bit 7 ~ 1 : Slave address 6~0

Note: *Bit 7~1 (SAD6~0) store slave address. The address data transmitted from the master is compared with the contents of these bits.*

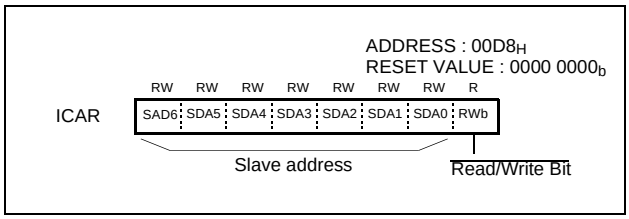


Figure 19-2 I²C address Register

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.

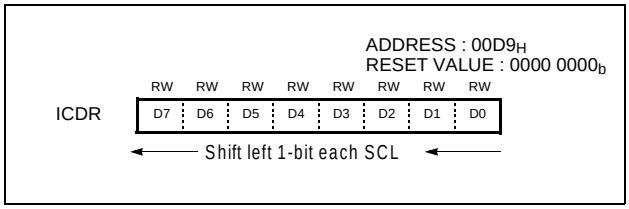


Figure 19-3 Data shift register

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 19-1.

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	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.
1	ADRb	Address represent flag 0 : current contents is address 1 : current contents is data

Bit No.	Name	Function
0	LRB	Last received bit. it is used for receive confirmation. If ACK is returned, LRB=0, or not 1.

Table 19-1 Bit function

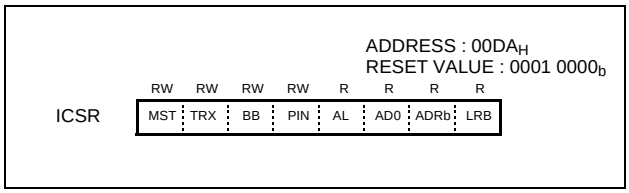


Figure 19-4 I²C status Register

I²C control register

It controls communication data format.

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.

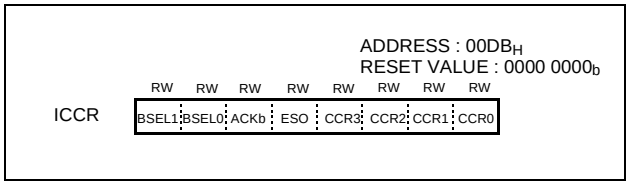


Figure 19-5 I²C control Register

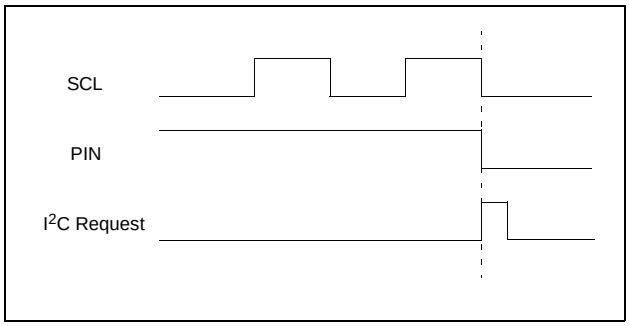


Figure 19-6 Interrupt request signal generation timing

Bit No.	Name	Function	
7 6	BSEL1 BSEL0	I²C connection control. 00: No connection 01: SCL0, SDA0 10: SCL1, SDA1 11: SCL0, SDA0, SCL1, SDA1	
5	ACK	If acknowlege clock is returned, this bit is 0, or not 1.	
4	ESO	I²C Bus interface use enable flag 0: Disabled 1: Enabled	
3 2 1 0	CCR3 CCR2 CCR1 CCR0	SCL Frequency selection SCL frequency = f_{ex} / (12 * CCR)	
		Value	f_{ex} = 4MHz
		0000	Not allowed
		0001	Not allowed
		0010	333.3KHz
		0011	222.2KHz
		0100	166.6KHz
		0101	133.3KHz
		0110	111.1KHz
		0111	95.2KHz
		1000	83.3KHz
		1001	74.1KHz
		1010	66.6KHz
		1011	60.6KHz
		1100	55.5KHz
		1101	51.3KHz
		1110	47.6KHz
		1111	44.4KHz

Table 19-2 Bit function

START condition generation

When the ESO bit of the I^2C control register ($00DB_H$) is "1", writing to the I^2C status register will generate START condition. Refer to Figure 19-7 for the START condition generation timing diagram.

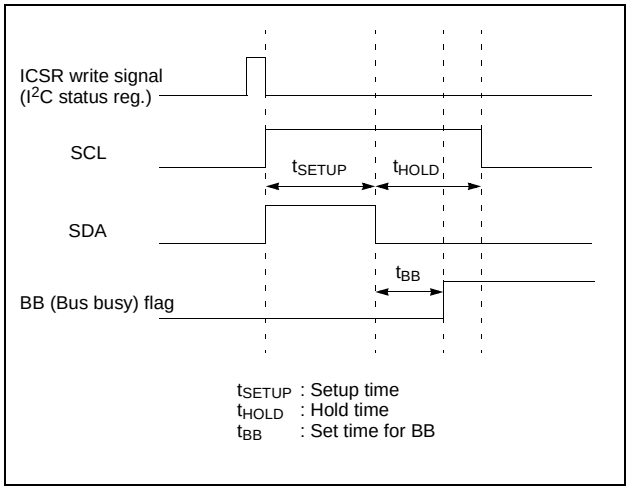


Figure 19-7 START condition generation timing

RESTART condition generation

RESTART condition's setting sequence is as followings.

1. Write 020_H to I^2C status register (ICSR, $00DA_H$)

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

3. Write $0F0_H$ to I^2C status register (ICSR, $00DA_H$)

STOP condition generation

Writing 'C0h' to ICSR will generate a stop condition, when ESO(ICCR bit3) is '1'

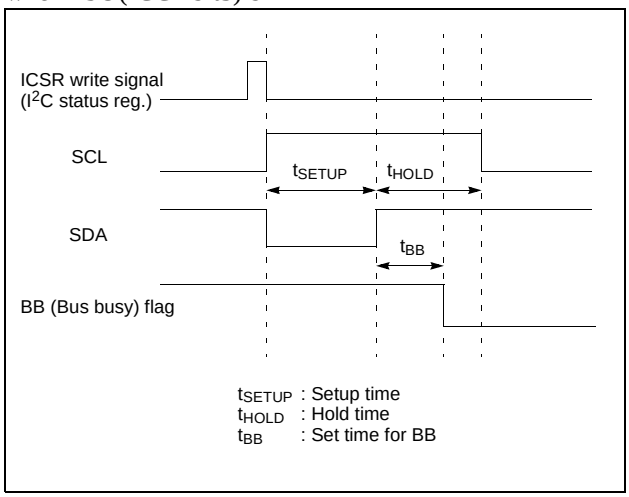


Figure 19-8 STOP condition generating timing diagram

START / STOP condition generation time is shown Table 19-3.

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 19-3 Example time (f_{ex}=4MHz)

START / STOP condition detect

START / STOP condition is detected when Table 19-3 is satisfied.

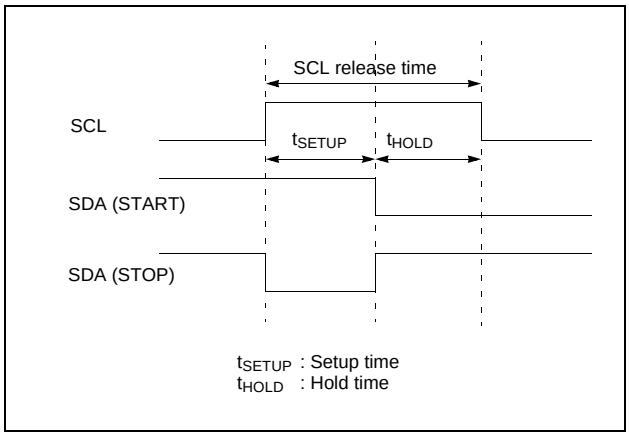


Figure 19-9 START / STOP condition detection timing

START / STOP detection time is showed Table 19-4.

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

Table 19-4 Example time (f_{ex}=4MHz)

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

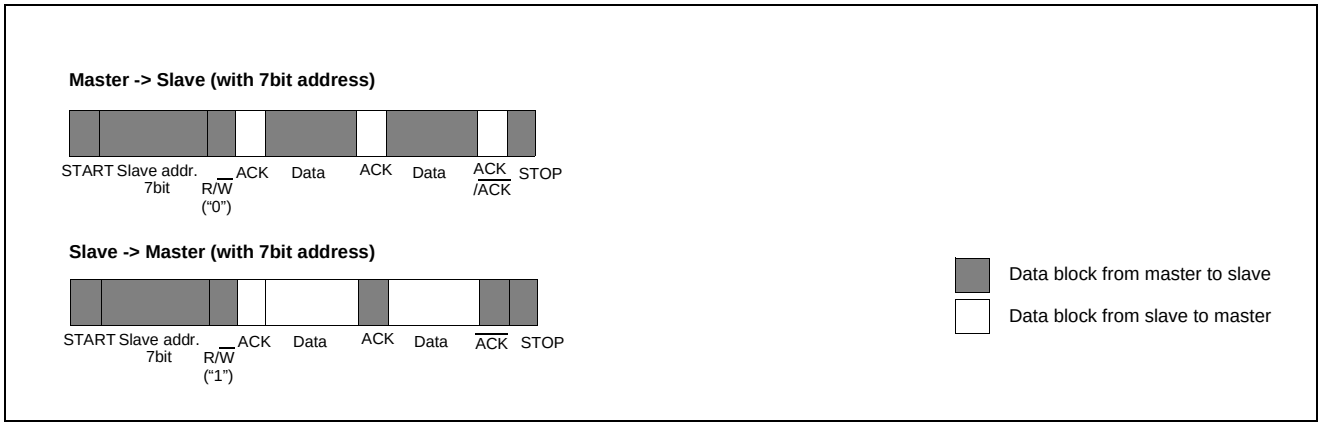


Figure 19-10 Address data communication format

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.

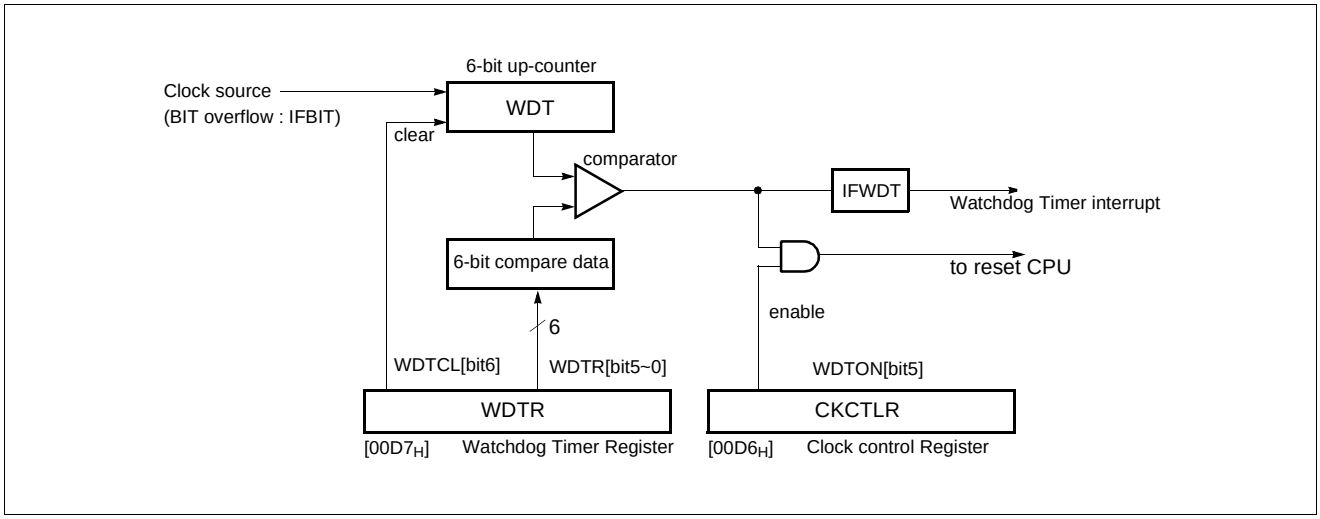


Figure 20-1 Block Diagram of Watchdog Timer

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.

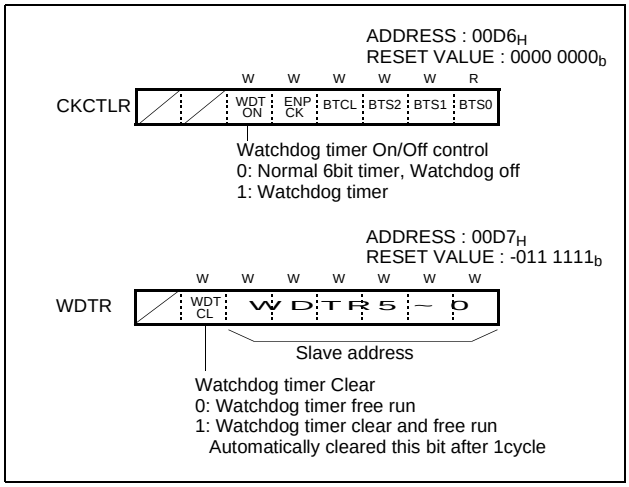


Figure 20-2 Watchdog timer register

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 (4uS)	1,024uS	32,256uS
001_b	PS5 (8uS)	2,028uS	64,512uS
010_b	PS6 (16uS)	4,096uS	129,024uS
011_b	PS7 (32uS)	8,192uS	258,048uS
100_b	PS8 (64uS)	16,384uS	516,096uS
101_b	PS9 (128uS)	32,768uS	1,032,192uS
110_b	PS10 (256uS)	65,536uS	2,064,384uS
111_b	PS11 (512uS)	131,072uS	4,128,768uS

Table 20-1 Watchdog timer MAX. cycle (Ex:f_{ex}=4MHz)

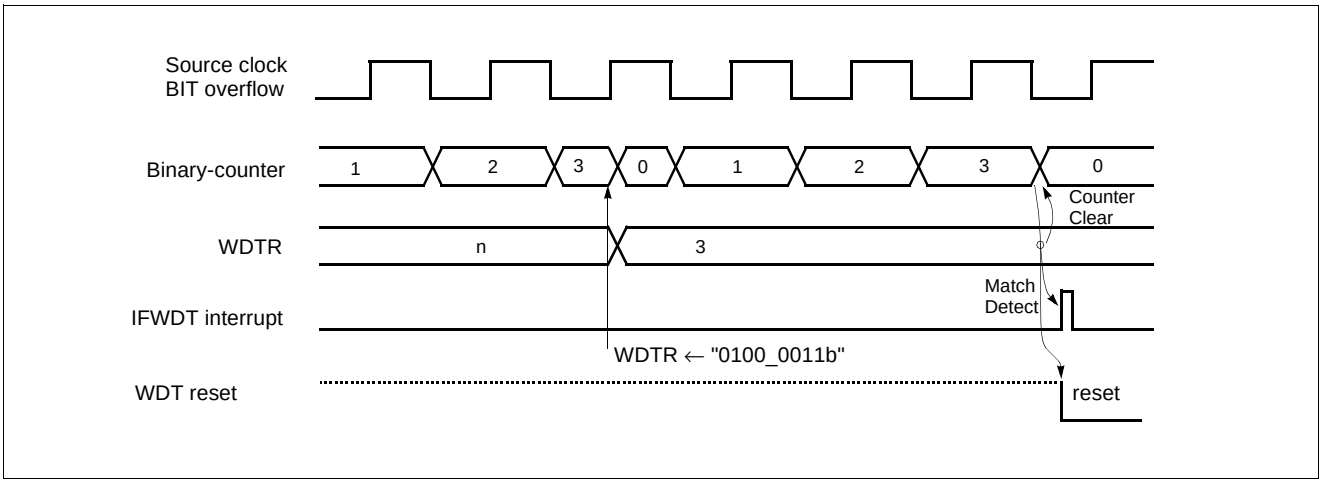


Figure 20-3 Watchdog timer Timing

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.

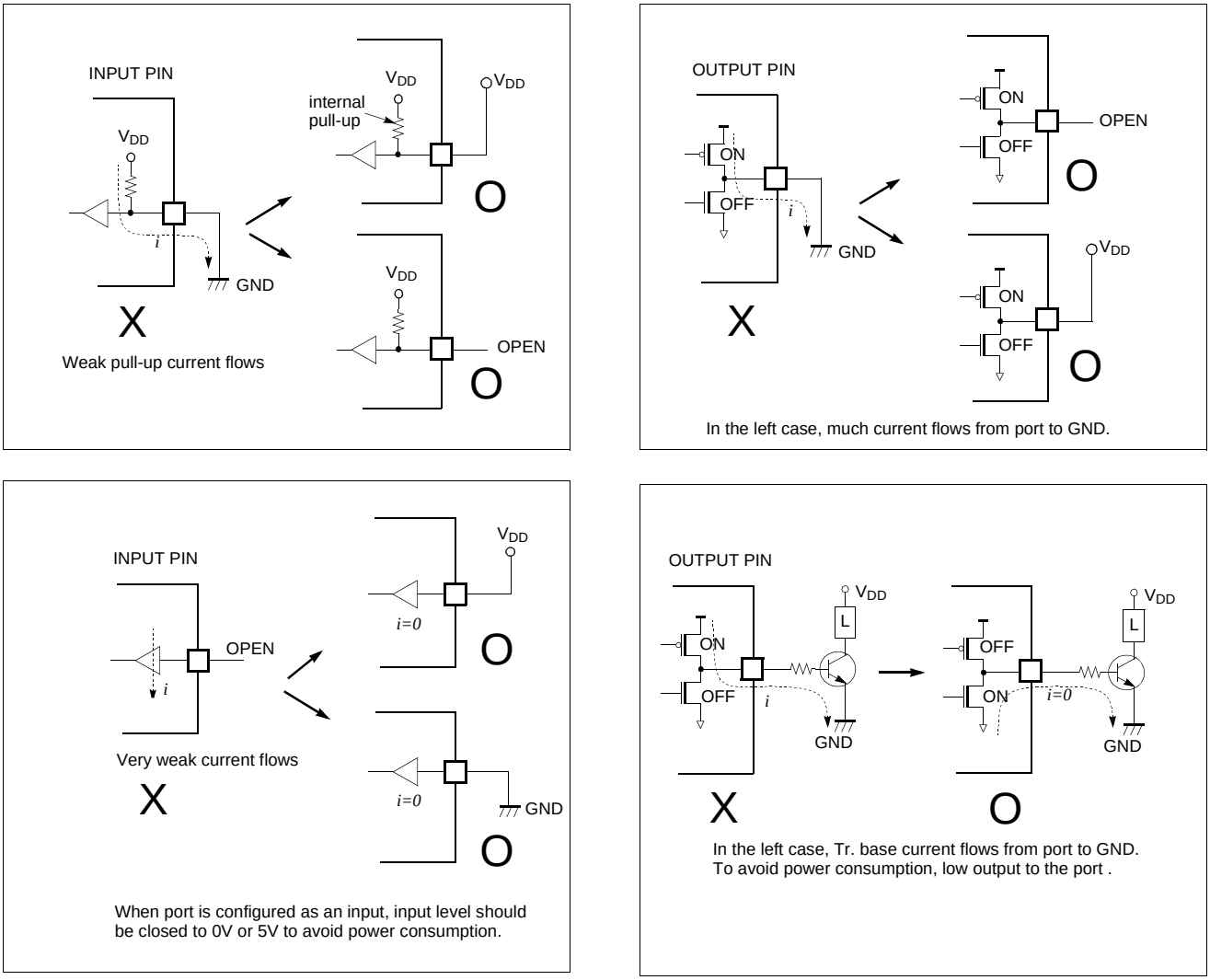


Figure 20-4 Application example of Port under Power Consumption

21. OSCILLATOR CIRCUIT

The HMS81C4x60 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 .

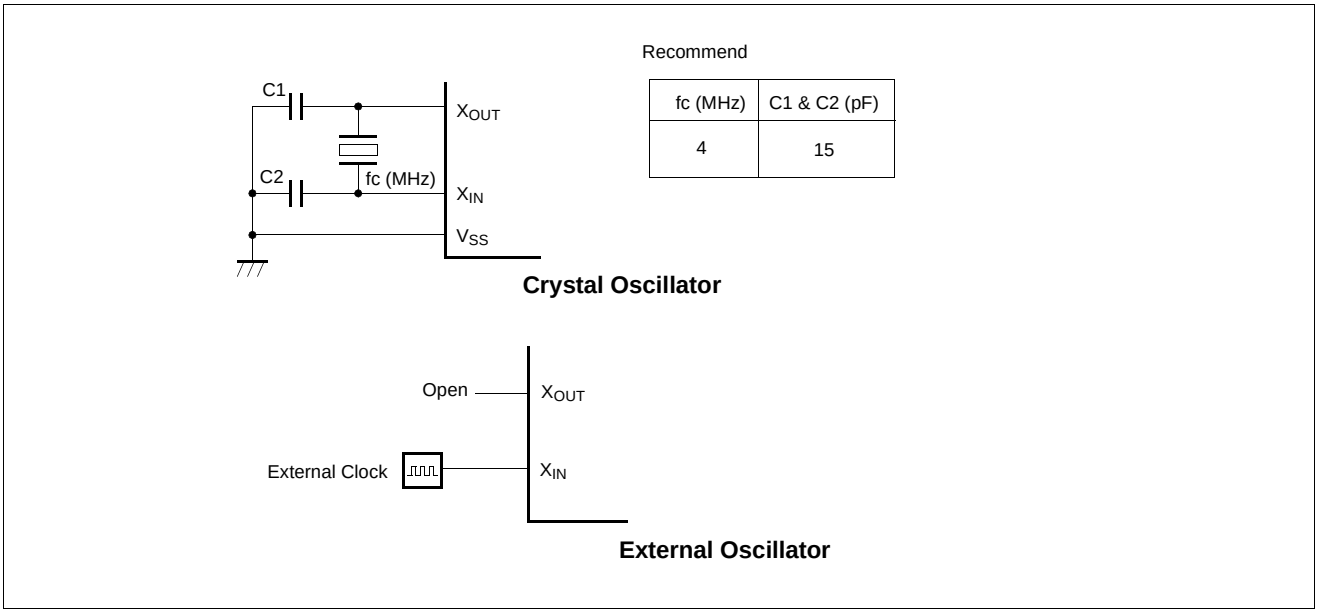


Figure 21-1 Oscillation Circuit

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

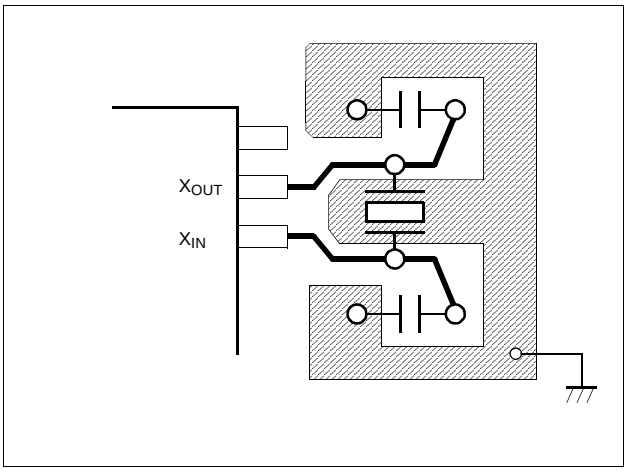


Figure 21-2 Layout example of Oscillator PCB circuit

22. RESET

The HMS81C4x60 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 .

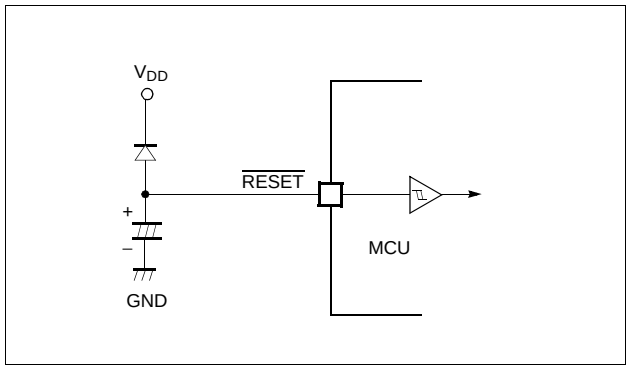


Figure 22-1 Simple Power-on-Reset Circuit

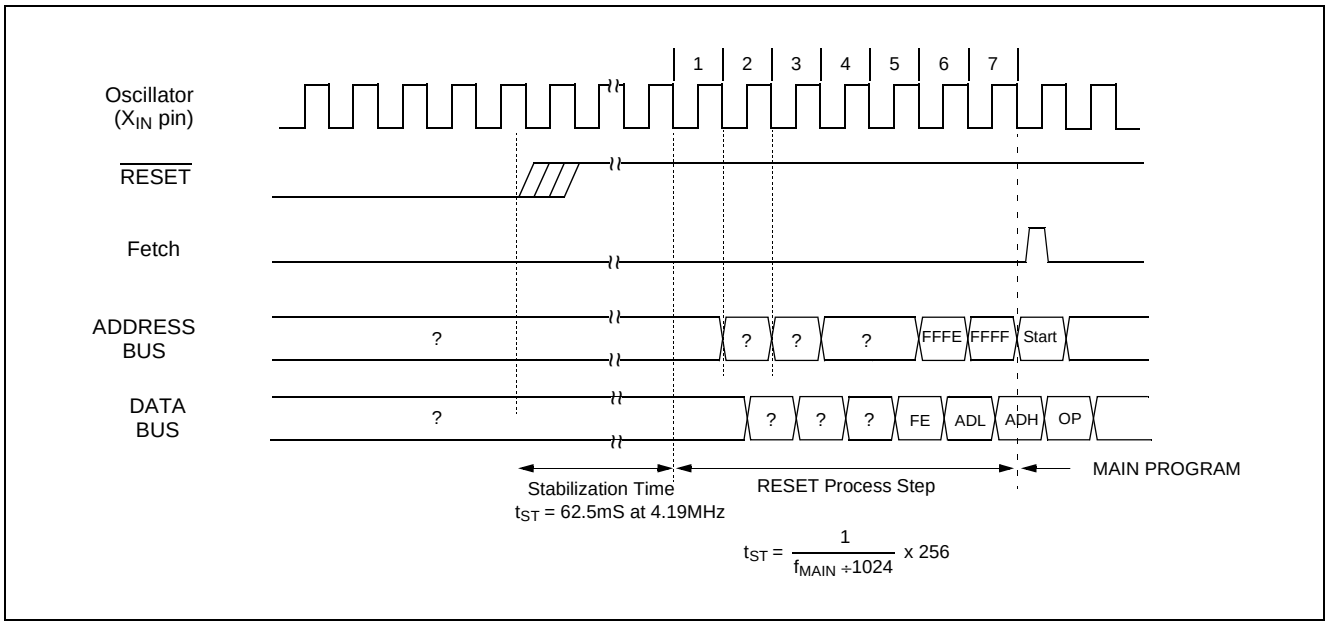


Figure 22-2 Timing Diagram after RESET

22.2 Watchdog Timer Reset

Refer to “20. WATCHDOG TIMER” on page 90.

23. OTP Programming

23.1 HMS87C4x60 OTP Programming

User can burn out HMS87C4x60 OTP through the general Gang programmer using special ROM writer. In Devleopment tool package auxiliary, HMS87C4x60 has ROM writer socket. HMS87C4x60 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 17FFFh.

Blank Check

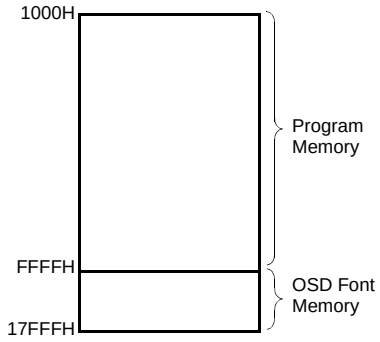


Figure 23-1 HMS87C4x60 OTP Memory Map

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. Burn program OTP file(Set chip target address 1000h ~ FFFFh)

3. Burn font OTP file(Set chip target address 10000h ~17FFFh)

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

1. Add program OTP file and font OTP file

2. Burn OTP file(Set chip target address 1000h ~ 17FFFh)

About other details, refer ROM wirter manual.

23.2 .Device Configuration Data

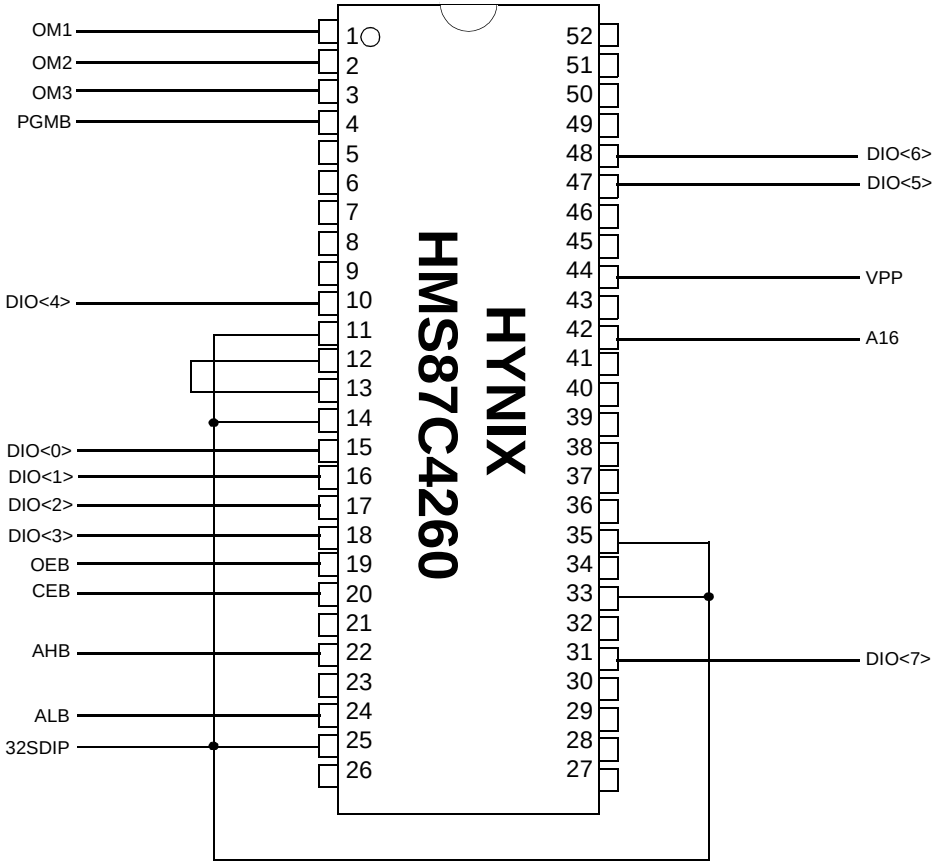


Figure 23-2 Figure Pin Configuration in OTP Programming Mode

	HMS87C4x60			
Mode	VPP	CEB	OEB	PGMB
Program	11.25	Low	High	Low
Verify	11.25	Low	Low	High
Optional Verify	5	Low	Low	X
Gang Write	11.25	Low	High	Low
Gang Verify	11.25, 5	Low	Low	X

Figure 23-3 Figure Mode Table

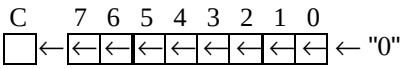
24. Assemble mnemonics

24.1 Instruction Map

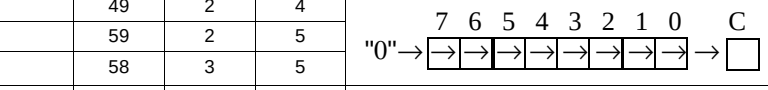
	00000 **00**	**00001** **01**	**00010** **02**	**00011** **03**	**00100** **04**	**00101** **05**	**00110** **06**	**00111** **07**	**01000** **08**	**01001** **09**	**01010** **0A**	**01011** **0B**	**01100** **0C**	**01101** **0D**	**01110** **0E**	**01111** **0F**
000	NOP	SET1 dp.bit	BBS A.bit,rel	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	

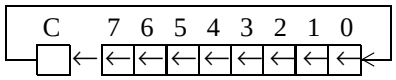
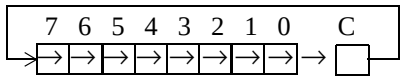
	10000 **10**	**10001** **11**	**10010** **12**	**10011** **13**	**10100** **14**	**10101** **15**	**10110** **16**	**10111** **17**	**11000** **18**	**11001** **19**	**11010** **1A**	**11011** **1B**	**11100** **1C**	**11101** **1D**	**11110** **1E**	**11111** **1F**
000	BPL rel	CLR1 dp.bit	BBC A.bit,rel	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 ← 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 ← A ^ (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	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 ← A ^ M, N ← (M_7), V ← (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) ← (PC_H), SP ← SP - 1, M(s) ← (PC_L), SP ← S - 1, M(SP) ← PSW, SP ← SP - 1, PC_L ← ($0FFDE_H$), PC_H ← ($0FFDF_H$)	- - - 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) ← (PC_H), SP ← SP-1, M(SP) ← (PC_L), SP←SP-1 if !abs, PC ← abs ; if [dp], PC_L ← (dp), PC_H ← (dp+1)	
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 ≠ (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		N - - - - - Z -
83	EOR !abs + Y	B5	3	5		
84	EOR [dp + X]	96	2	6		
85	EOR [dp] + Y	97	2	6		
86	EOR {X}	94	1	3		
87	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : C ← 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		

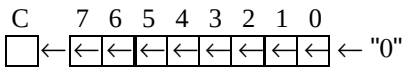
NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
99	LDA #imm	C4	2	2	Load accumulator	N - - - - - Z -
100	LDA dp	C5	2	3	A ← (M)	
101	LDA dp + X	C6	2	4		
102	LDA !abs	C7	3	4		
103	LDA !abs + Y	D5	3	5		
104	LDA [dp + X]	D6	2	6		
105	LDA [dp]+Y	D7	2	6		
106	LDA {X}	D4	1	3		
107	LDA {X}+	DB	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)

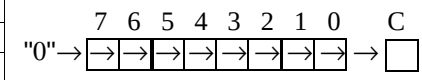
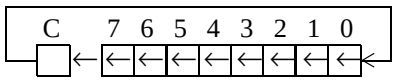
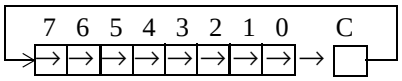
NO.	MNENONIC	OP CODE	BYTE NO.	CYCLE NO	OPERATION	FLAG NVGBHIZC
148	ROL A	28	1	2	Rotate left through carry 	N - - - - - ZC
149	ROL dp	29	2	4		
150	ROL dp + X	39	2	5		
151	ROL !abs	38	3	5		
152	ROR A	68	1	2	Rotate right through carry 	N - - - - - ZC
153	ROR dp	69	2	4		
154	ROR dp + X	79	2	5		
155	ROR !abs	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) ← (PC_H), SP ← SP-1, M(SP) ← (PC_L), SP←SP-1 if !abs, PC ← abs ; if [dp], PC_L ← (dp), PC_H ← (dp+1)	
16	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal ;	- - - - - - - -
17	CBNE dp + X, rel	8D	3	6/8	If A ≠ (M), then PC ← PC + rel.	
18	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal :	- - - - - - - -
19	DBNE Y,rel	7B	2	4/6	if (M) ≠ 0, then PC ← PC + rel.	
20	JMP !abs	1B	3	3	Unconditional jump	- - - - - - - -
21	JMP [!abs]	1F	3	5	PC ← jump address	
22	JMP [dp]	3F	2	4		
23	PCALL	4F	2	6	U-page call : M(SP) ← (PC_H), SP ← SP -1, M(SP) ← (PC_L), SP ← SP -1, PC_L ← (upage), PC_H ←"OFF_H"	- - - - - - - -
24	TCALL n	nA	1	8	Table call : M(SP) ← (PC_H), SP ← SP -1, M(SP) ← (PC_L), SP ← SP -1 PC_L ← (Table vector L), PC_H ← (Table vector H)	- - - - - - - -

6. Control Operation & etc.

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