

HYNIX SEMICONDUCTOR
8-BIT SINGLE-CHIP MICROCONTROLLERS

HMS81004E

HMS81008E

HMS81016E

HMS81024E

HMS81032E

User's Manual (Ver. 1.00)



Version 1.00

Published by
SP MCU Application Team

©2001 Hynix Semiconductor, Inc. All right reserved.

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.

Table of Contents

| | | | |
|---|----|--|-----|
| 1. OVERVIEW | 1 | Oscillation Circuit | 34 |
| Description | 1 | 11. BASIC INTERVAL TIMER | 36 |
| Features | 1 | 12. WATCH DOG TIMER | 38 |
| Development Tools | 2 | 13. Timer0, Timer1, Timer2 | 39 |
| 2. BLOCK DIAGRAM | 3 | 14. INTERRUPTS | 47 |
| 3. PIN ASSIGNMENT (Top View) | 4 | Interrupt priority and sources | 48 |
| 4. PACKAGE DIMENSION | 5 | Interrupt control register | 48 |
| 5. PIN FUNCTION | 8 | Interrupt accept mode | 49 |
| 6. PORT STRUCTURES | 10 | Interrupt Sequence | 50 |
| 7. ELECTRICAL CHARACTERISTICS ... | 12 | BRK Interrupt | 52 |
| Absolute Maximum Ratings | 12 | Multi Interrupt | 52 |
| Recommended Operating Conditions | 12 | External Interrupt | 52 |
| DC Electrical Characteristics | 12 | Key Scan Input Processing | 53 |
| REMOUT Port Ioh Characteristics Graph | 13 | 15. STANDBY FUNCTION | 55 |
| REMOUT Port Iol Characteristics Graph | 14 | Sleep Mode | 55 |
| AC Characteristics | 14 | STOP MODE | 55 |
| 8. MEMORY ORGANIZATION | 16 | STANDBY MODE RELEASE | 56 |
| Registers | 16 | RELEASE OPERATION OF STANDBYMODE | 58 |
| Program Memory | 19 | 16. RESET FUNCTION | 60 |
| Data Memory | 22 | EXTERNAL RESET | 60 |
| List for Control Registers | 23 | POWER ON RESET | 60 |
| Addressing Mode | 25 | Low Voltage Detection Mode | 62 |
| 9. I/O PORTS | 30 | A. MASK ORDER SHEET | i |
| R0 Ports | 30 | B. INSTRUCTION | ii |
| R1 Ports | 30 | Terminology List | ii |
| R2 Port | 32 | Instruction Map | iii |
| 10. CLOCK GENERATOR | 33 | Instruction Set | iv |

HMS81004E/08E/16E/24E/32E

CMOS SINGLE- CHIP 8-BIT MICROCONTROLLER FOR UNIVERSAL REMOTE CONTROLLER

1. OVERVIEW

1.1 Description

The HMS81004E/08E/16E/24E/32E is an advanced CMOS 8-bit microcontroller with 4/8/16/24/32K bytes of ROM. The device is one of GMS800 family. The HYNIX HMS81004E/08E/16E/24E/32E is a powerful microcontroller which provides a highly flexible and cost effective solution to many UR applications. The HMS81004E/08E/16E/24E/32E provides the following standard features: 4/8/16/24/32K bytes of ROM, 448 bytes of RAM, 8-bit timer/counter, on-chip oscillator and clock circuitry. In addition, the HMS81004E/08E/16E/24E/32E supports power saving modes to reduce power consumption.

| Device Name | ROM Size | EPROM Size | RAM Size | Package |
|-------------|-----------|------------|--|---|
| HMS81004E | 4K Bytes | - | 448 Bytes (included 256 bytes stack memory) | 20 SOP/PDIP 24 SOP/Skinny DIP 28 SOP/Skinny DIP |
| HMS81008E | 8K Bytes | - | | |
| HMS81016E | 16K Bytes | - | | |
| HMS81024E | 24K Bytes | - | | |
| HMS81032E | 32K Bytes | - | | |
| HMS81020TL | - | 20K Bytes | | |
| HMS81032TL | - | 32K Bytes | | |

1.2 Features

- Instruction Cycle Time:

- 1us at 4MHz

- Programmable I/O pins

| | 20 PIN | 24 PIN | 28 PIN |
|--------|--------|--------|--------|
| INPUT | 3 | 3 | 3 |
| OUTPUT | 2 | 2 | 2 |
| I/O | 13 | 17 | 21 |

- Operating Voltage

- 2.0 ~ 3.6 V @ 4MHz (MASK)

- 2.0 ~ 4.0 V @ 4MHZ (OTP)

- Timer

- Timer / Counter 16Bit * 1ch

- 8Bit * 2ch

- Basic Interval Timer 8Bit * 1ch

- Watch Dog Timer 6Bit * 1ch

- 8 Interrupt sources

- Nested Interrupt control is available.

- External input: 2

- Keyscan input

- Basic Interval Timer

- Watchdog timer

- Timer : 3

- Power On Reset

- Power saving Operation Modes

- STOP Operation

- SLEEP Operation

- Low Voltage Detection Circuit

- Watch Dog Timer Auto Start (During 1second after Power on Reset)

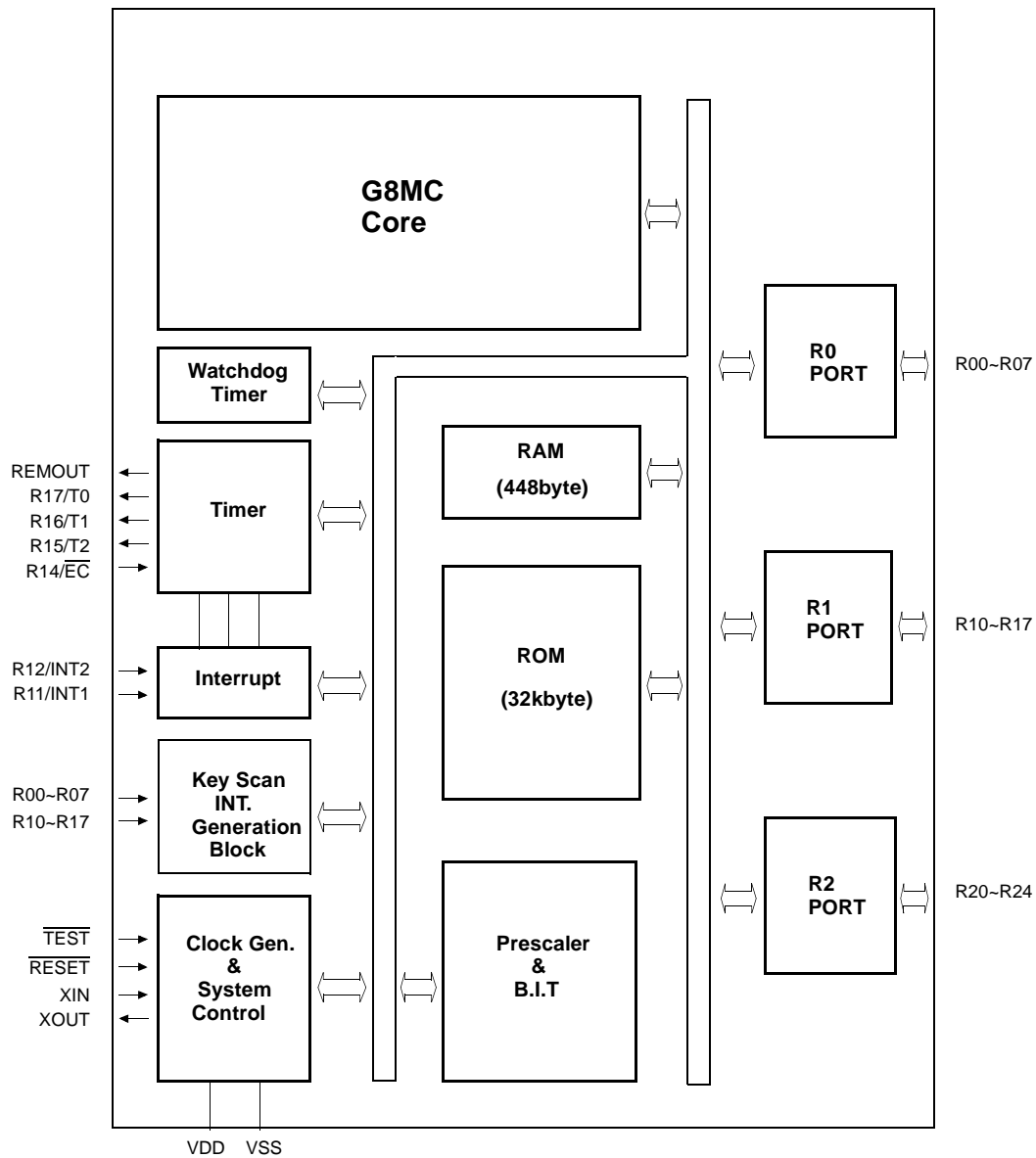
1.3 Development Tools

The HMS81004E/08E/16E/24E/32E are supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr.TM and OTP programmers. Macro assembler operates under the MS-Windows 95/98TM /NT4/W2000.

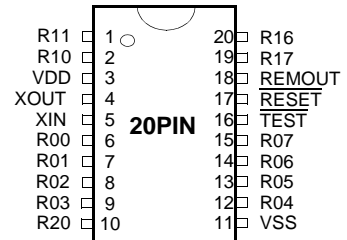
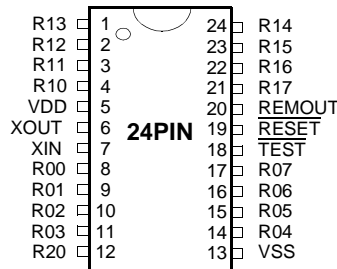
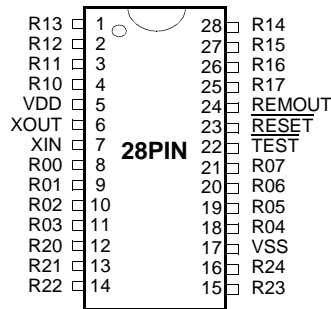
Please contact sales part of HYNIX

| | |
|------------------------|--|
| Software | - MS- Window base assembler - Linker / Editor / Debugger |
| Hardware (Emulator) | - CHOICE-Dr. - CHOICE-Dr. EVA 81C5EVA |
| OTP programmer | - Universal single programmer. - 4 gang programmer - stand alone |

2. BLOCK DIAGRAM

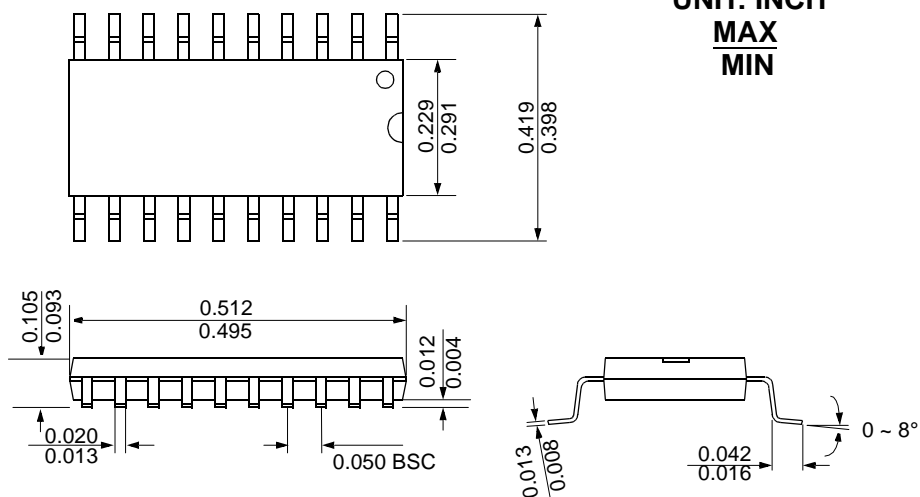


3. PIN ASSIGNMENT (Top View)

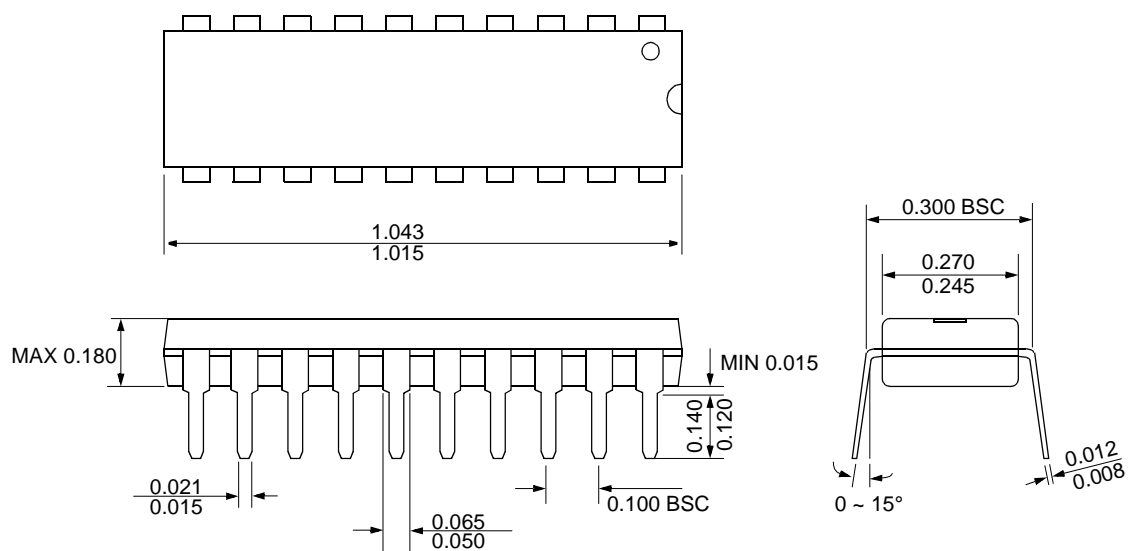


4. PACKAGE DIMENSION

20 SOP

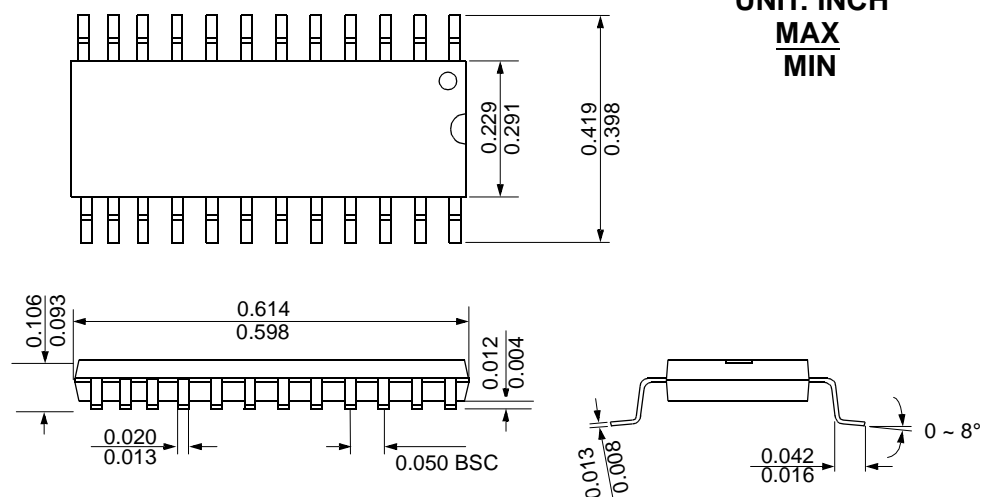
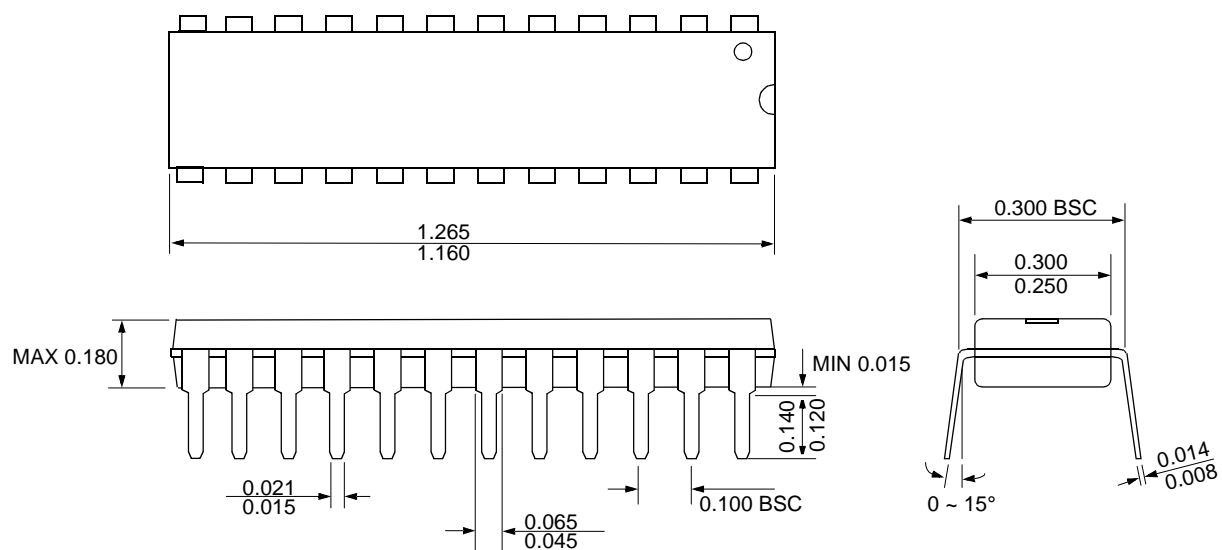


20 PDIP

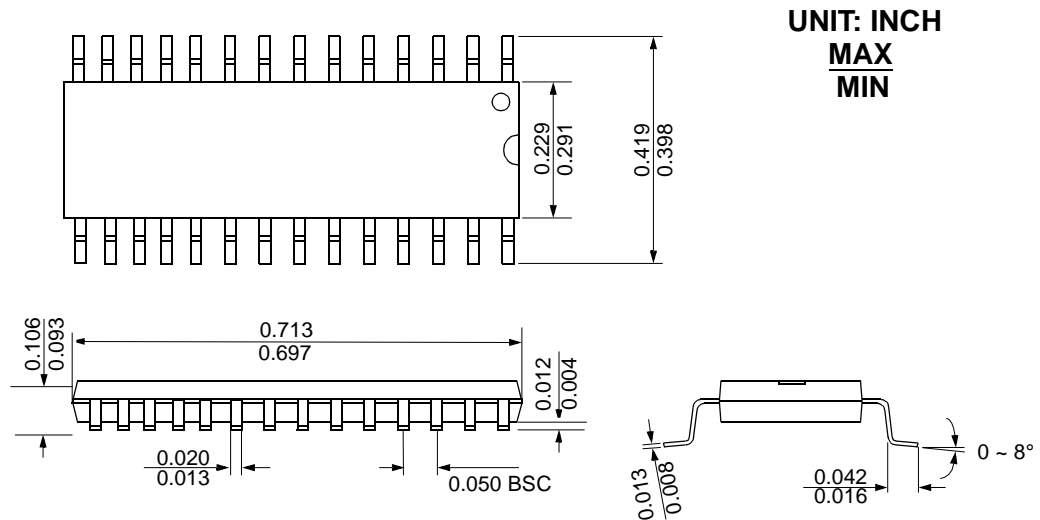


24 SOP

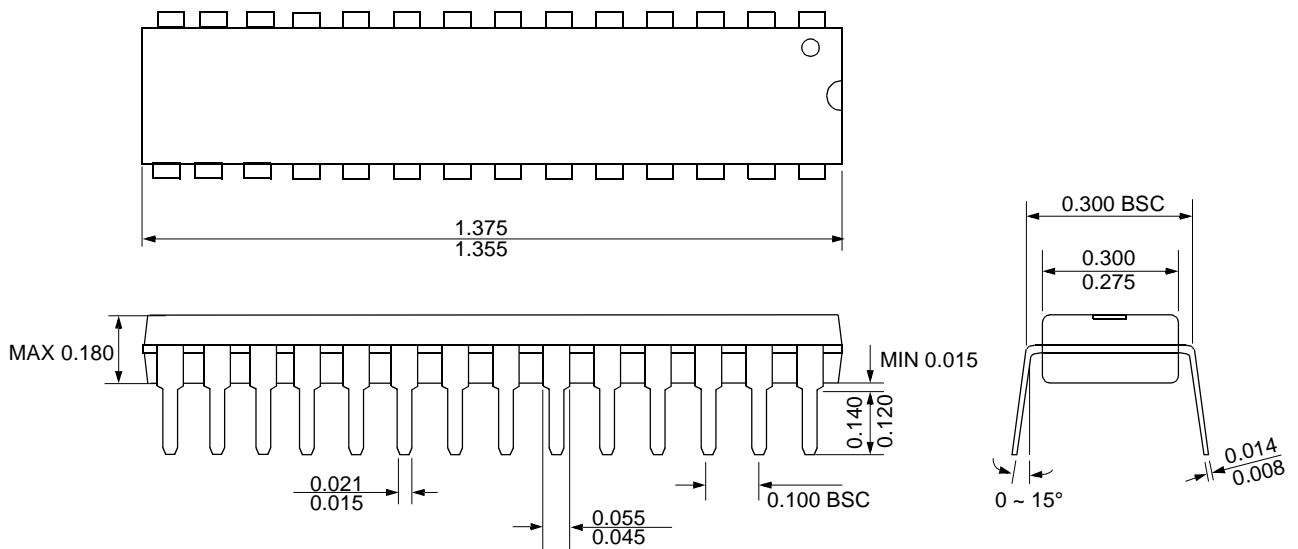
UNIT: INCH
MAX
MIN

**24 SKDIP**

28 SOP



28 SKDIP



5. PIN FUNCTION

V_{DD}: Supply voltage.

V_{SS}: Circuit ground.

TEST: Used for shipping inspection of the IC. For normal operation, it should be connected to V_{DD}.

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 CMOS bidirectional I/O port. R0 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

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

used as outputs or inputs.

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

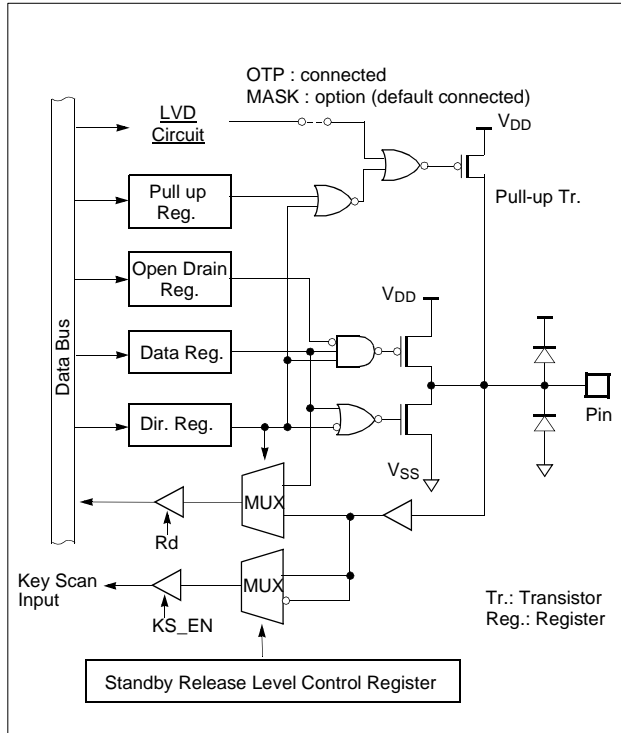
| Port pin | Alternate function |
|----------|--|
| R11 | INT1 (External Interrupt input 1) |
| R12 | INT2 (External Interrupt input 2) |
| R14 | \overline{EC} (Event Counter input) |
| R15 | T2 (Timer / Counter input 2) |
| R16 | T1 (Timer / Counter input 1) |
| R17 | T0 (Timer / Counter input 0) |

R20~R24: R2 is an 8-bit CMOS bidirectional I/O port. R2 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs .

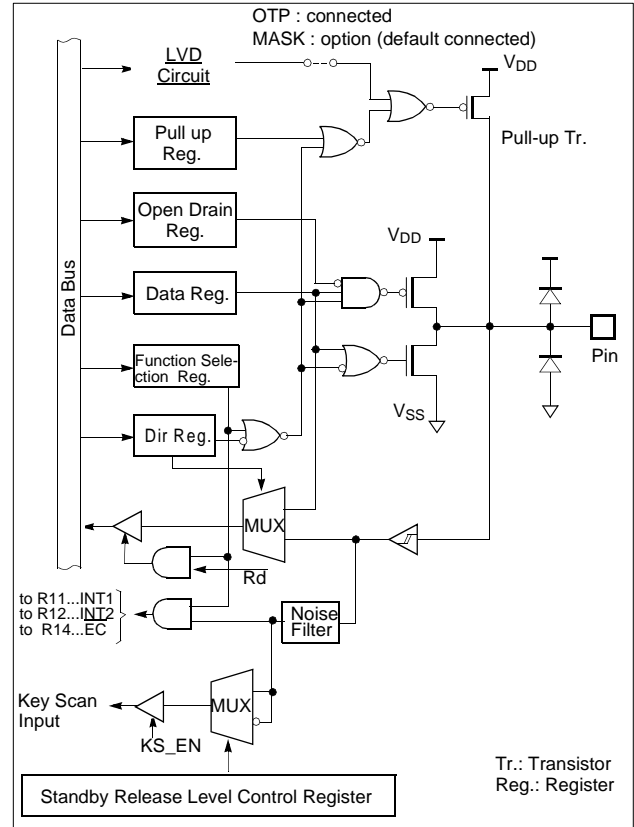
| PIN NAME | INPUT/ OUTPUT | Function | @RESET | @STOP |
|----------------------|------------------|---|------------|----------------------|
| R00 | I/O | <ul style="list-style-type: none"> - Each bit of the port can be individually configured as an input or an output by user software - Push-pull output - CMOS input with pull-up resistor (option) - Can be programmable as key scan input - Pull-up resistors are automatically disabled at output mode | INPUT | State of before Stop |
| R01 | I/O | | | |
| R02 | I/O | | | |
| R03 | I/O | | | |
| R04 | I/O | | | |
| R05 | I/O | | | |
| R06 | I/O | | | |
| R07 | I/O | <ul style="list-style-type: none"> - Each bit of the port can be individually configured as an input or an output by user software - Push-pull output - CMOS input with pull-up resistor (option) - Can be programmable as key scan input or open drain output - Pull-up resistors are automatically disabled at output mode - Direct driving of LED(N-Tr.) | INPUT | State of before Stop |
| R10 | I/O | | | |
| R11/INT1 | I/O | | | |
| R12/INT2 | I/O | | | |
| R13 | I/O | | | |
| R14/ \overline{EC} | I/O | | | |
| R15/T2 | I/O | | | |
| R16/T1 | I/O | <ul style="list-style-type: none"> - Each bit of the port can be individually configured as an input or an output by user software - Push-pull output - CMOS input with pull-up resistor (option) - Pull-up resistors are automatically disabled at output mode - Direct driving of LED(N-Tr.) | INPUT | State of before Stop |
| R17/T0 | I/O | | | |
| R20 | I/O | | | |
| R21 | I/O | | | |
| R22 | I/O | | | |
| R23 | I/O | <ul style="list-style-type: none"> - High current output | 'L' output | 'L' output |
| R24 | I/O | | | |
| XIN | I | Oscillator input | | Low |
| XOUT | O | Oscillator output | | High |
| REMOUT | O | High current output | 'L' output | 'L' output |
| \overline{RESET} | I | Includes pull-up resistor | 'L' level | state of before stop |
| \overline{TEST} | I | Includes pull-up resistor | | |
| VDD | P | Positive power supply | | |
| VSS | P | Groud | | |

6. PORT STRUCTURES

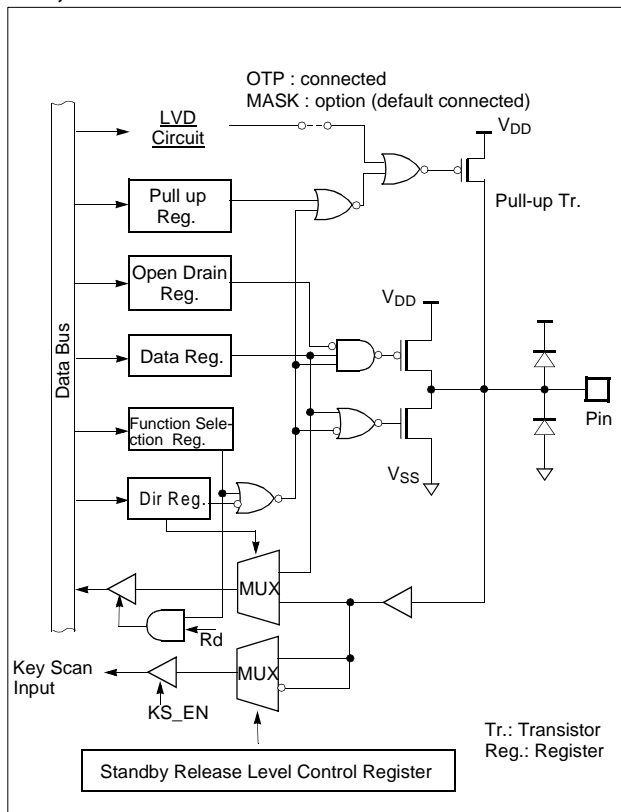
R0[0:7]



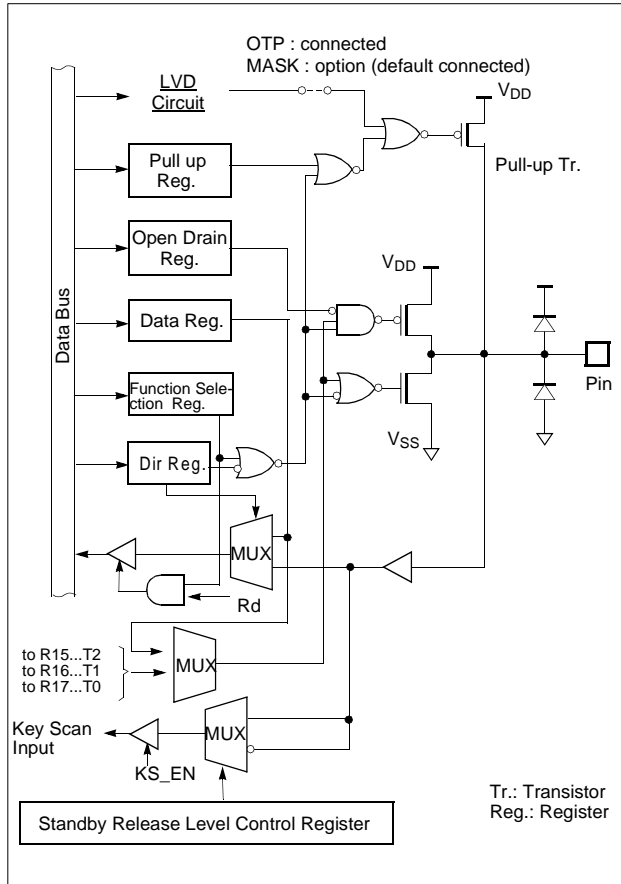
R11/INT1, R12/INT2, R14/ \overline{EC}



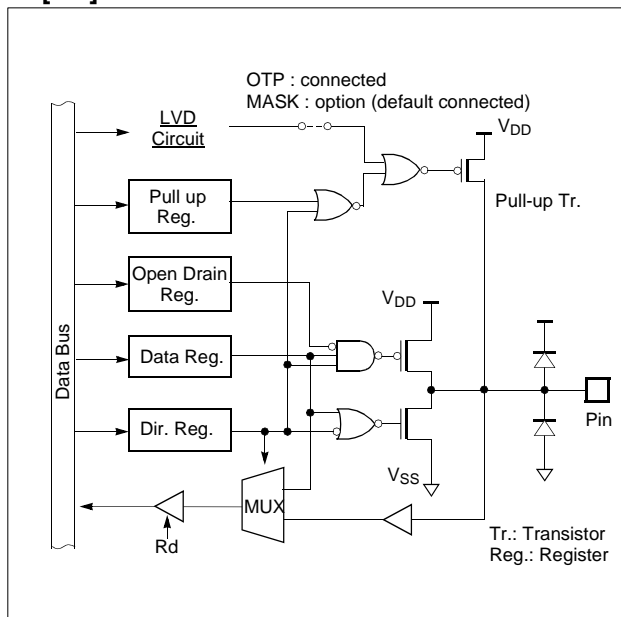
R10, R13



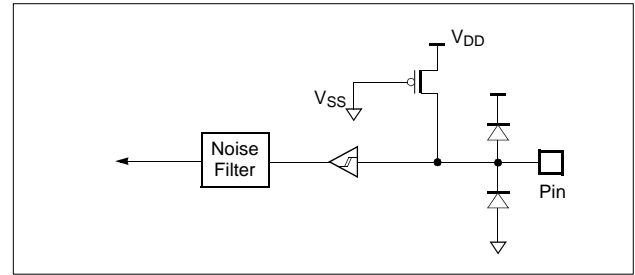
R15/T2, R16/T1, R17/T0



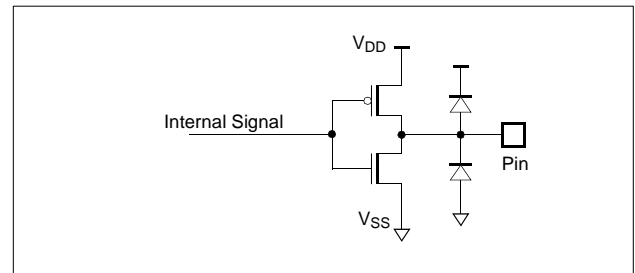
R2[0:4]



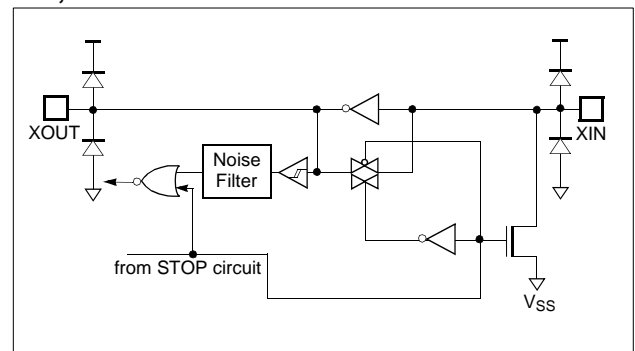
TEST



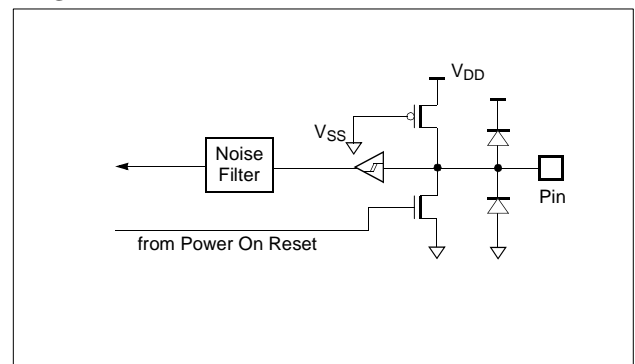
REMOUT



XIN, XOUT



RESET



7. ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

| | |
|----------------------------|------------------------|
| Supply voltage | -0.3 to +5.0 V |
| Input Voltage | -0.3 to $V_{DD}+0.3$ V |
| Output Voltage | -0.3 to $V_{DD}+0.3$ V |
| Operating Temperature..... | 0~70°C |
| Storage Temperature | -65~150°C |
| Power Dissipation..... | 700 mW |

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

7.2 Recommended Operating Conditions

| Parameter | Symbol | Condition | Specifications | | Unit |
|-----------------------|-----------|------------------------------|----------------|------|------|
| | | | Min. | Max. | |
| Supply Voltage | V_{DD} | $f_{XIN}=4\text{MHz}$ | 2.0 | 3.6 | V |
| Operating Frequency | f_{XIN} | $V_{DD}=2.0\sim 3.6\text{V}$ | 1.0 | 4.0 | MHz |
| Operating Temperature | T_{OPR} | - | 0 | +70 | °C |

7.3 DC Electrical Characteristics

($T_A=0\sim 70^\circ\text{C}$, $V_{DD}=2.0\sim 3.6\text{V}$, $GND=0\text{V}$)

| Parameter | Symbol | Condition | Specifications | | | Unit |
|-----------------------------------|-----------|---|----------------|------|--------------|---------------|
| | | | Min. | Typ. | Max. | |
| High level input Voltage | V_{IH1} | R11,R12,R14, $\overline{\text{RESET}}$ | $0.8 V_{DD}$ | - | V_{DD} | V |
| | V_{IH2} | R0,R1(except R11,R12,R14), R2 | $0.7 V_{DD}$ | - | V_{DD} | V |
| Low level input Voltage | V_{IL1} | R11,R12,R14, $\overline{\text{RESET}}$ | 0 | - | $0.2 V_{DD}$ | V |
| | V_{IL2} | R0,R1(except R11,R12,R14), R2 | 0 | - | $0.3 V_{DD}$ | V |
| High level input Leakage Current | I_{IH} | R0,R1,R2, $\overline{\text{RESET}}$, $V_{IH}=V_{DD}$ | - | - | 1 | μA |
| Low level input Leakage Current | I_{IL} | R0,R1,R2, $\overline{\text{RESET}}$ (without pull-up), $V_{IL}=0$ | - | - | -1 | μA |
| High level output Voltage | V_{OH1} | R0, $I_{OH}=-0.5\text{mA}$ | $V_{DD}-0.4$ | - | - | V |
| | V_{OH2} | R1[6:0], R2, $I_{OH}=-1.0\text{mA}$ | $V_{DD}-0.4$ | - | - | V |
| | V_{OH3} | XIN, XOUT, $I_{OH}=-200\mu\text{A}$ | $V_{DD}-0.9$ | - | - | V |
| Low level output Voltage | V_{OL1} | R0, $I_{OL}=1\text{mA}$ | - | - | 0.4 | V |
| | V_{OL2} | R1, R2, $I_{OL}=5\text{mA}$ | - | - | 0.8 | V |
| | V_{OL3} | XIN, XOUT, $I_{OL}=200\mu\text{A}$ | - | - | 0.8 | V |
| High level output Leakage Current | I_{OHL} | R0,R1,R2, $V_{OH}=V_{DD}$ | - | - | 1 | μA |
| Low level output Leakage Current | I_{OLL} | R0,R1,R2, $V_{OL}=0$ | - | - | -1 | μA |

| Parameter | Symbol | Condition | Specifications | | | Unit |
|------------------------------|------------|---|----------------|------|------|---------|
| | | | Min. | Typ. | Max. | |
| High Level output current | I_{OH} | REMOUT, R17, $V_{OH}=2V$ | -30 | -12 | -5 | mA |
| Low Level output current | I_{OL} | REMOUT, $V_{OL}=1V$ | 0.5 | - | 3 | mA |
| Input pull-up current | I_p | R0,R1,R2, \overline{RESET} , VDD=3V | 15 | 30 | 60 | μA |
| Power Supply Current | I_{DD1} | Operating current ,fxin=4Mhz, VDD=2.0V | - | 2.4 | 6 | mA |
| | I_{DD2} | Operating current ,fxin=4Mhz, VDD=3.6V | - | 4 | 10 | mA |
| | I_{SLP1} | Sleep mode current ,fxin=4Mhz, VDD=2.0V | - | 1 | 2 | mA |
| | I_{SLP2} | Sleep mode current ,fxin=4Mhz, VDD=3.6V | - | 2 | 3 | mA |
| | I_{STP1} | Stop mode current ,Oscillator Stop VDD=2.0V | - | 2 | 8 | μA |
| | I_{STP2} | Stop mode current ,Oscillator Stop VDD=3.6V | - | 3 | 10 | μA |
| RAM retention supply voltage | V_{RET} | - | 0.7 | - | - | V |

7.4 REMOUT Port Ioh Characteristics Graph

(typical process & room temperature)

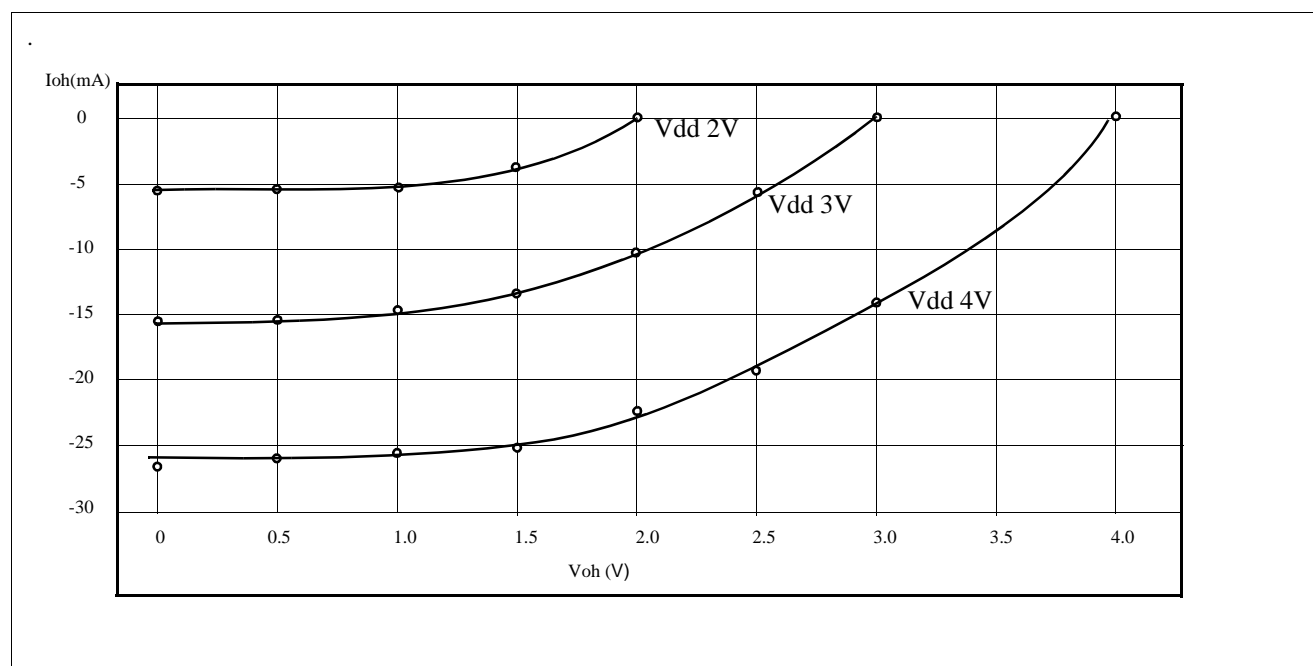


Figure 7-1 Ioh vs Voh

7.5 REMOUT Port Iol Characteristics Graph

(typical process & room temperature)

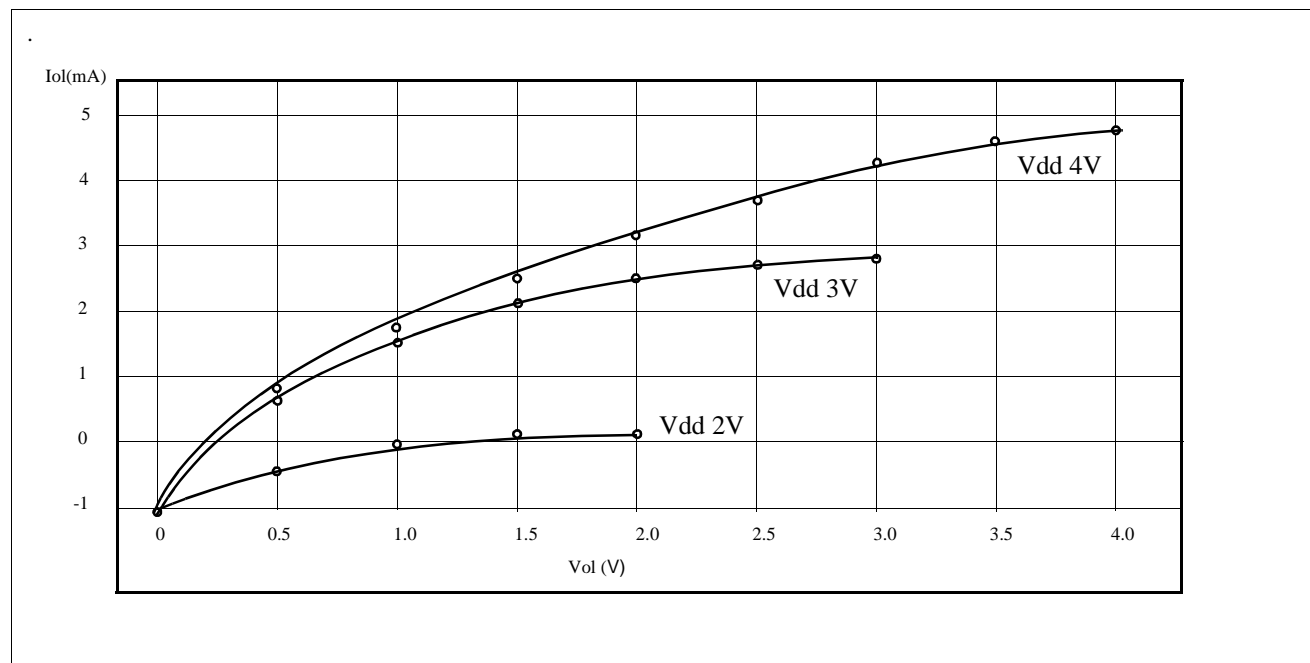


Figure 7-2 Iol vs Vol

7.6 AC Characteristics

($T_A=0\sim+70^{\circ}\text{C}$, $V_{DD}=2.0\sim3.6\text{V}$, $V_{SS}=0\text{V}$)

| Parameter | Symbol | Pins | Specifications | | | Unit |
|---|------------|---------------------------|----------------|------|------|-----------|
| | | | Min. | Typ. | Max. | |
| External clock input cycle time | t_{CP} | X_{IN} | 250 | 500 | 1000 | ns |
| System clock cycle time | t_{SYS} | | 500 | 1000 | 2000 | ns |
| External clock pulse width High | t_{CPH} | X_{IN} | 40 | - | - | ns |
| External clock pulse width Low | t_{CPL} | X_{IN} | 40 | - | - | ns |
| External clock rising time | t_{RCP} | X_{IN} | - | - | 40 | ns |
| External clock falling time | t_{FCP} | X_{IN} | - | - | 40 | ns |
| Interrupt pulse width High | t_{IH} | INT1, INT2 | 2 | - | - | t_{SYS} |
| Interrupt pulse width Low | t_{IL} | INT1, INT2 | 2 | - | - | t_{SYS} |
| $\overline{\text{RESET}}$ Input pulse width low | t_{RSTL} | $\overline{\text{RESET}}$ | 8 | - | - | t_{SYS} |
| Event counter input pulse width high | t_{ECH} | $\overline{\text{EC}}$ | 2 | - | - | t_{SYS} |
| Event counter input pulse width low | t_{ECL} | $\overline{\text{EC}}$ | 2 | - | - | t_{SYS} |
| Event counter input pulse rising time | t_{REC} | $\overline{\text{EC}}$ | - | - | 40 | ns |
| Event counter input pulse falling time | t_{FEC} | $\overline{\text{EC}}$ | - | - | 40 | ns |

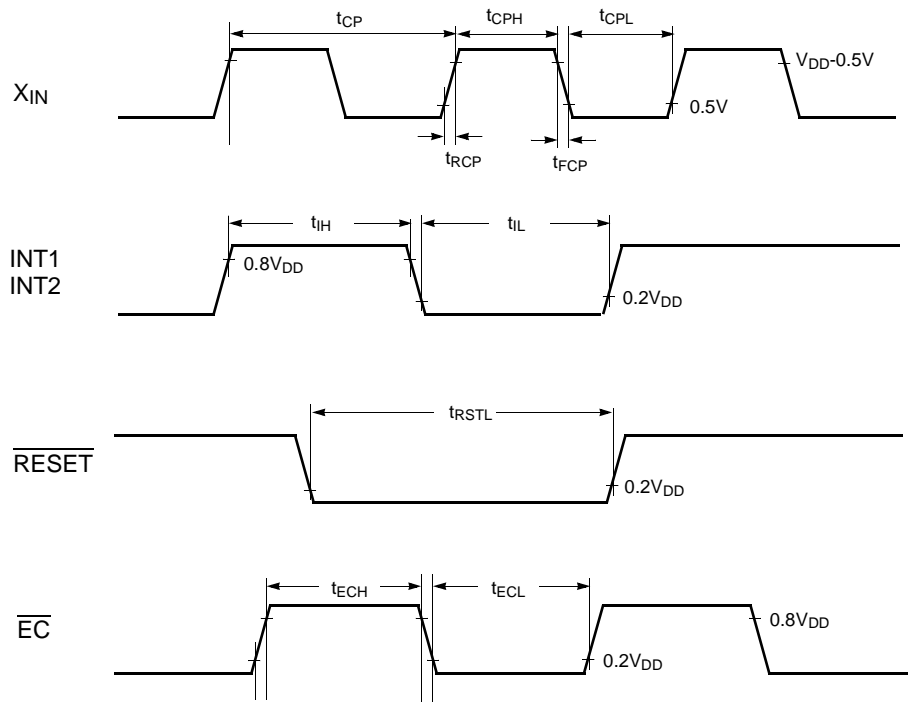


Figure 7-3 Timing Diagram

8. MEMORY ORGANIZATION

The HMS81004E/08E/16E/24E/32E has separate address spaces for Program memory and Data Memory. Program memory can only be read, not written to. It can be up to

32K bytes of Program memory. Data memory can be read and written to up to 448 bytes including the stack area.

8.1 Registers

This device has six registers that are the Program Counter (PC), an 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.

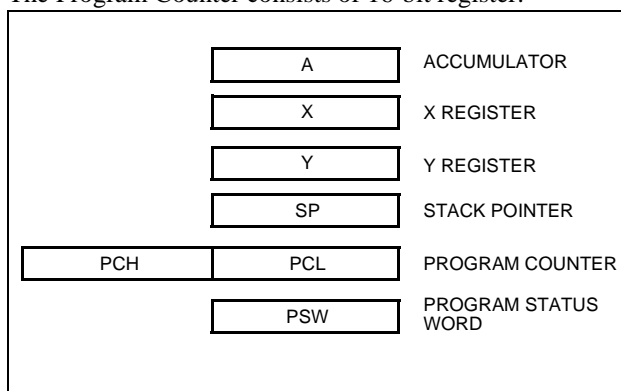


Figure 8-1 Configuration of Registers

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.

In the case of multiplication instruction, execute as a multiplier register. After multiplication operation, the lower 8-bit of the result enters. ($Y * A \Rightarrow YA$). In the case of division instruction, execute as the lower 8-bit of dividend. After division operation, quotient enters.

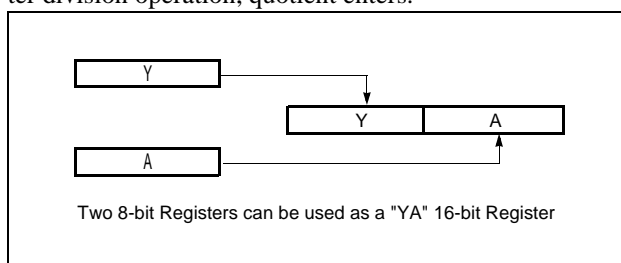


Figure 8-2 Configuration of YA 16-bit Register

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.

• X Register

In the case of division instruction, execute as register.

• Y Register

In the case of 16-bit operation instruction, execute as the upper 8-bit of YA. (16-bit accumulator). In the case of multiplication instruction, execute as a multiplicand register. After multiplication operation, the upper 8-bit of the result enters. In the case of division instruction, execute as the upper 8-bit of dividend. After division operation, remains enters. Y register can be used as loop counter of conditional branch command. (e.g. DBNE Y, rel)

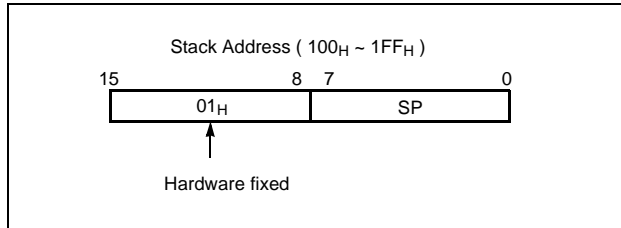
Stack Pointer:

The Stack Pointer is an 8-bit register used for occurrence interrupts, calling out subroutines and PUSH, POP, RETI, RET instruction. 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 SP is post-decremented when a subroutine call or a push instruction is executed, or when an interrupt is accepted. The SP is pre-incremented when a return or a pop instruction is executed.

The stack can be located at any position within 100_H to 1FF_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
```

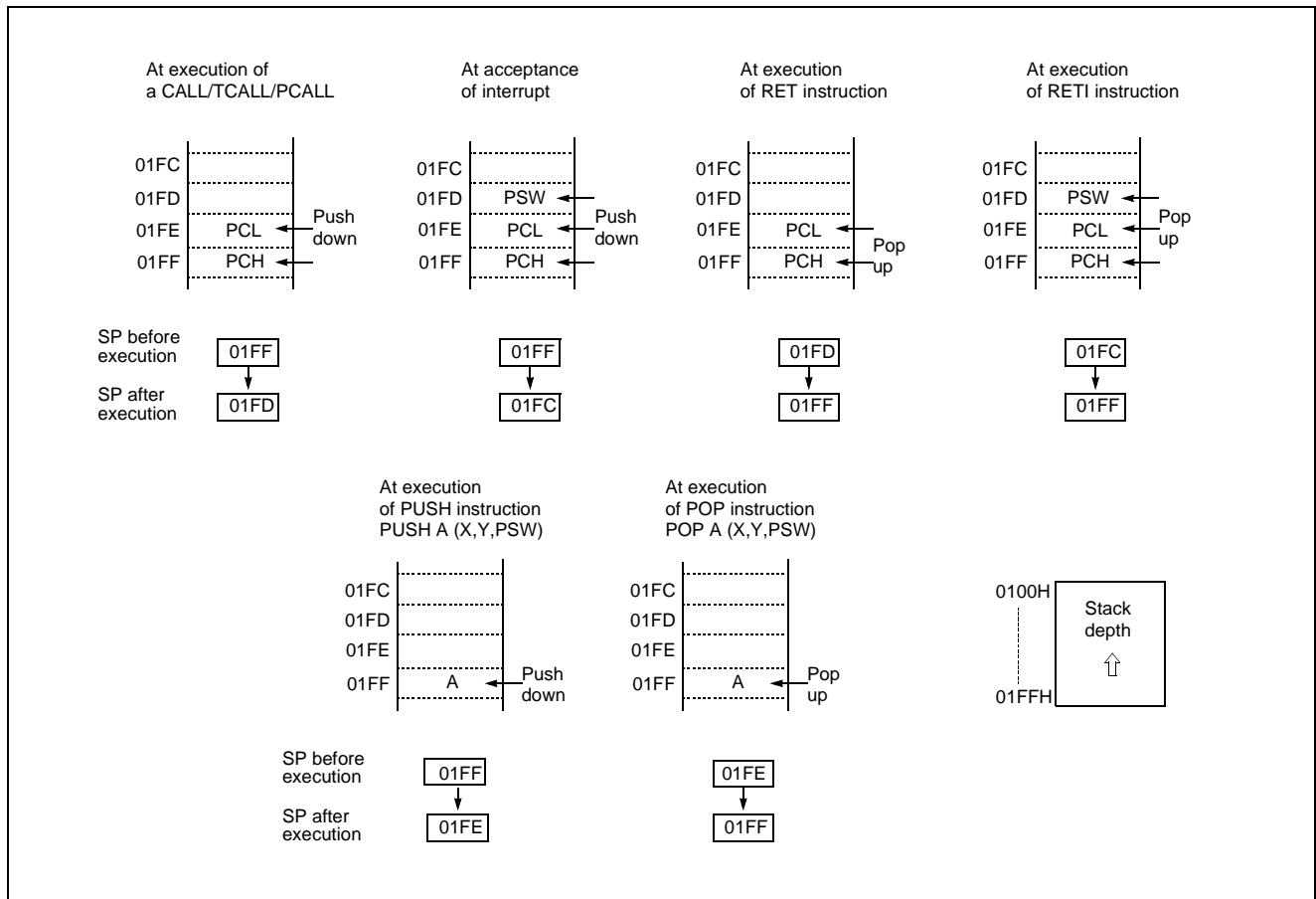


Figure 8-3 Stack Operation

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 8-4. 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.

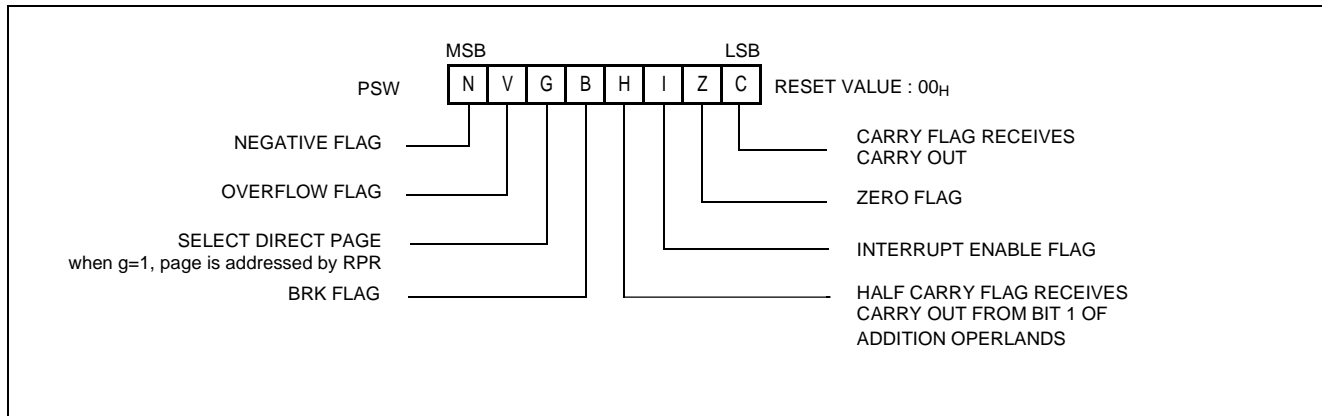


Figure 8-4 PSW (Program Status Word) Register

[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 CLR_V instruction with Overflow flag (V).

[Break flag B]

This flag is set by software BRK instruction to distinguish BRK from T_{CALL} 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 1 Page. It is set by SET_G instruction and cleared by CLR_G.

[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 CLR_V instruction clears the overflow flag. There is no set instruction. When the BIT instruction is executed, bit 6 of memory is copied to this flag.

[Negative flag N]

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

8.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but this device has 4/8/16/24/32K bytes program memory space only physically implemented. Accessing a location above FFFF_H will cause a wrap-around to 0000_H.

Figure 8-5 , shows a map of Program Memory. After reset, the CPU begins execution from reset vector which is stored in address FFFE_H and FFFF_H as shown in Figure 8-6 .

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

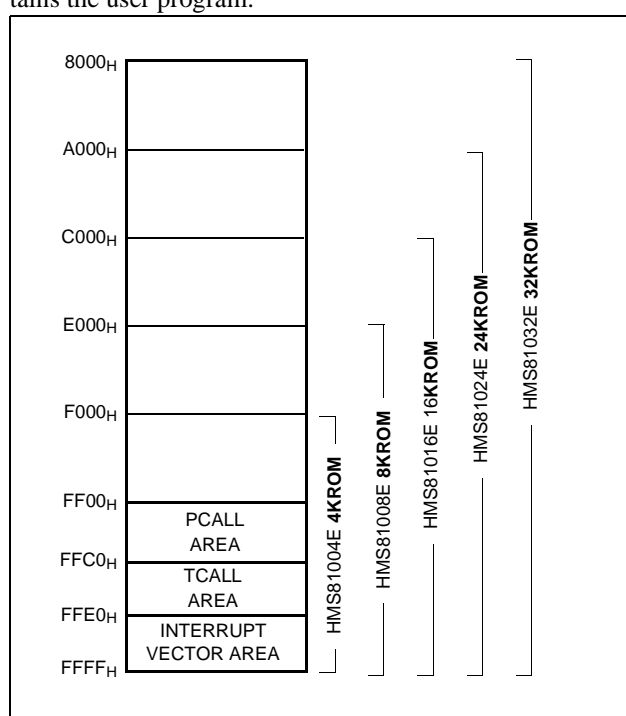


Figure 8-5 Program Memory Map

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

Table Call (TCALL) causes the CPU to jump to each TCALL address, where it commences the execution of the service routine. The Table Call service area spaces 2-byte for every TCALL: 0FFC0_H for TCALL15, 0FFC2_H for TCALL14, etc., as shown in Figure 8-7 .

Example: Usage of TCALL

```
LDA      #5
TCALL    0FH
:
:
; 1BYTE INSTRUCTION
; INSTEAD OF 2 BYTES
; NORMAL CALL

; TABLE CALL ROUTINE
;
FUNC_A:  LDA    LRG0
        RET
;
FUNC_B:  LDA    LRG1
        RET
;
; TABLE CALL ADD. AREA
;
ORG      0FFC0H
DW       FUNC_A
DW       FUNC_B
```

The interrupt causes the CPU to jump to specific location, where it commences the execution of the service routine. The External interrupt 0, for example, is assigned to location 0FFFA_H. The interrupt service locations spaces 2-byte interval: 0FFF8_H and 0FFF9_H for External Interrupt 1, 0FFFA_H and 0FFFB_H for External Interrupt 0, etc.

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 |
|--------------------|--|
| 0FFDE _H | S/W Interrupt Vector Area |
| E0 | - |
| E2 | - |
| E4 | - |
| E6 | Basic Interval Timer Interrupt Vector Area |
| E8 | Watch Dog Timer Interrupt Vector Area |
| EA | - |
| EC | - |
| EE | Timer2 Interrupt Vector Area |
| F0 | Timer1 Interrupt Vector Area |
| F2 | Timer0 Interrupt Vector Area |
| F4 | - |
| F6 | External Interrupt 2 Vector Area |
| F8 | External Interrupt 1 Vector Area |
| FA | Key Scan Interrupt Vector Area |
| FC | - |
| FE | RESET Vector Area |

NOTE:

"-" means reserved area.

Figure 8-6 Interrupt Vector Area

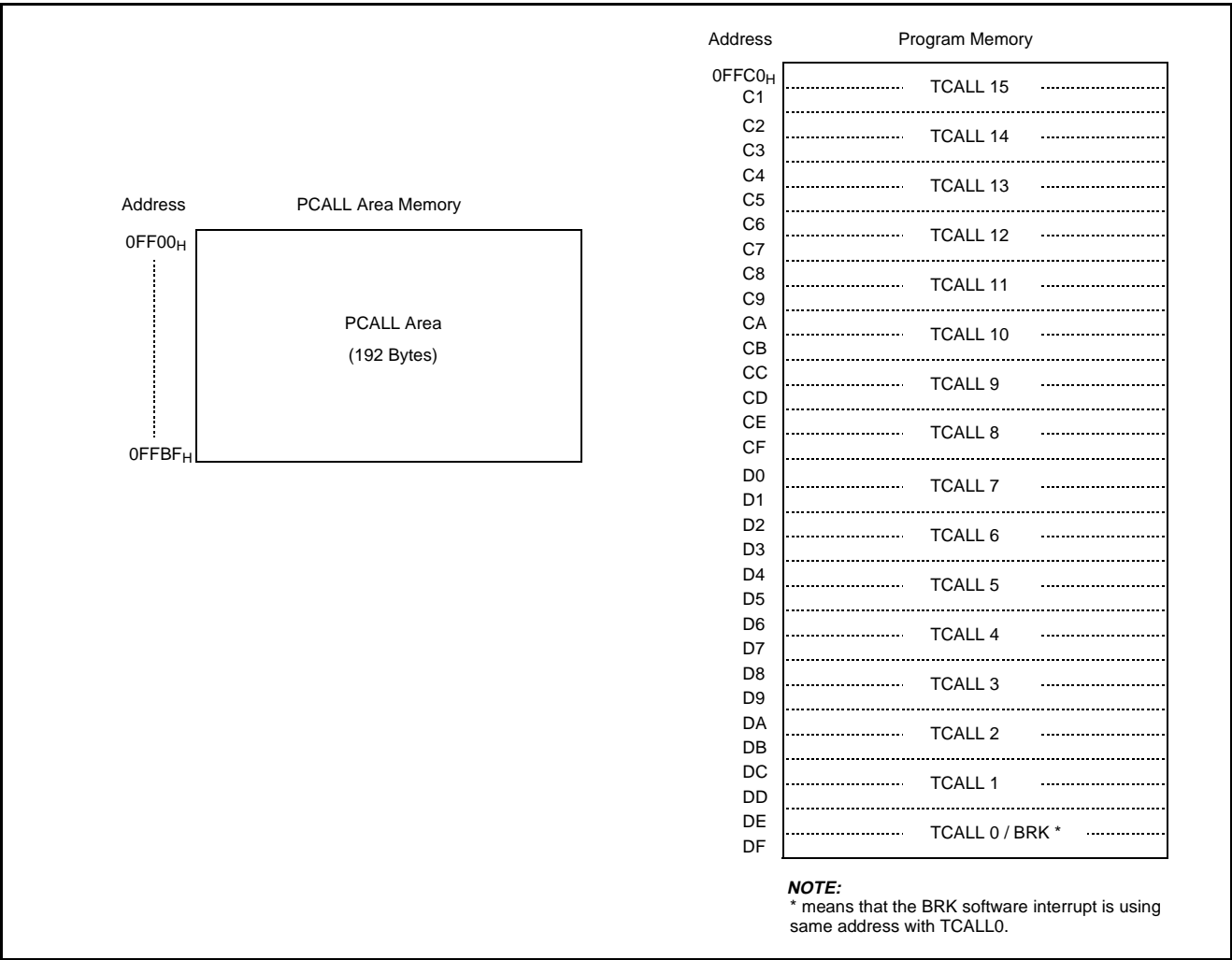
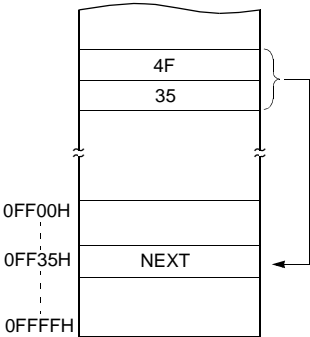


Figure 8-7 PCALL and TCALL Memory Area

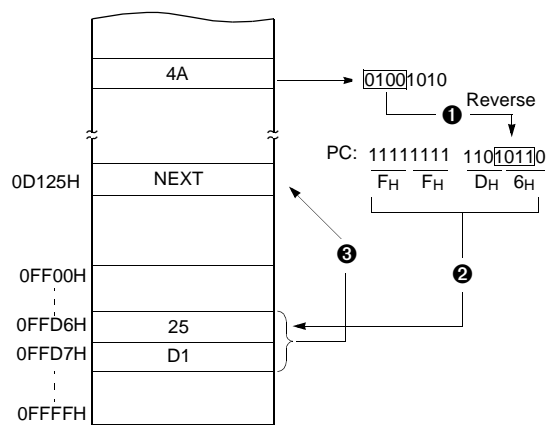
PCALL→rel

4F35 PCALL 35H



TCALL→n

4A TCALL 4



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

```

        ORG      0FFE0H

        DW      NOT_USED
        DW      NOT_USED
        DW      NOT_USED
        DW      BIT_INT           ; BIT
        DW      WDT_INT          ; Watch Dog Timer
        DW      NOT_USED
        DW      NOT_USED
        DW      TMR2_INT         ; Timer-2
        DW      TMR1_INT         ; Timer-1
        DW      TMR0_INT         ; Timer-0
        DW      NOT_USED        ;
        DW      INT2             ; Int.2
        DW      INT1             ; Int.1
        DW      KEY_INT          ; Key Scan
        DW      NOT_USED        ;
        DW      RESET            ; Reset


        ORG      08000H          ;HMS81032E Program start address

;*****
;          MAIN      PROGRAM      *
;*****
;
RESET:    NOP
          CLRG
          DI           ;Disable All Interrupts
          LDX          #0
RAM_CLR:  LDA          #0           ;RAM Clear(!0000H->!00BFH)
          STA          {X}+
          CMPX         #0C0H
          BNE          RAM_CLR      ;

          LDX          #0FFH        ;Stack Pointer Initialize
          TXSP

          LDM          R0, #0        ;Normal Port 0
          LDM          R0DD,#1000_0010B ;Normal Port Direction
          LDM          P0PC,#1000_0010B ;Pull Up Selection Set
          LDM          PMR1,#0000_0010B ;R1 port / int
          :
          :
          LDM          CKCTLR,#0011_1101B ;WDT ON , 16mS Time delay after stop mode release
          :
          :

```


8.3 Data Memory

Figure 8-8 shows the internal Data Memory space available. Data Memory is divided into 3 groups, a user RAM, control registers, Stack.

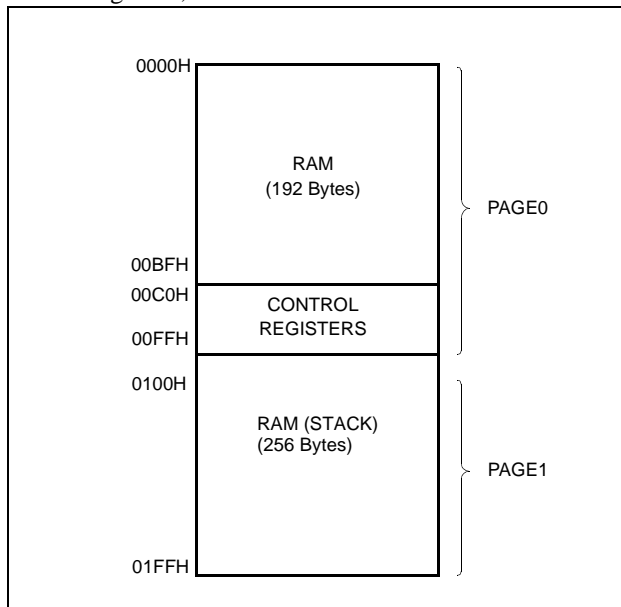


Figure 8-8 Data Memory Map

User Memory

The HMS81004E/08E/16E/24E/32E has 448×8 bits for the user memory (RAM).

Control Registers

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

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

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

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

Example; To write at CKCTRL

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

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 8-3 on page 17.

8.4 List for Control Registers

| Address | Function Register | Symbol | Read Write | RESET Value |
|---------|--------------------------------|--------|------------|-------------|
| 00C0h | PORT R0 DATA REG. | R0 | R/W | undefined |
| 00C1h | PORT R0 DATA DIRECTION REG. | R0DD | W | 00000000b |
| 00C2h | PORT R1 DATA REG. | R1 | R/W | undefined |
| 00C3h | PORT R1 DATA DIRECTION REG. | R1DD | W | 00000000b |
| 00C4h | PORT R2 DATA REG. | R2 | R/W | undefined |
| 00C5h | PORT R2 DATA DIRECTION REG. | R2DD | W | 00000000b |
| 00C6h | reserved | | | |
| 00C7h | CLOCK CONTROL REG. | CKCTLR | W | --110111b |
| | BASIC INTERVAL REG. | BTR | R | undefined |
| 00C8h | WATCH DOG TIMER REG. | WDTR | W | -0001111b |
| 00C9h | PORT R1 MODE REG. | PMR1 | W | 00000000b |
| 00CAh | INT. MODE REG. | IMOD | R/W | -0000000b |
| 00CBh | EXT. INT. EDGE SELECTION | IEDS | W | 00000000b |
| 00CCh | INT. ENABLE REG. LOW | IENL | R/W | -00-----b |
| 00CDh | INT. REQUEST FLAG REG. LOW | IRQL | R/W | -00-----b |
| 00CEh | INT. ENABLE REG. HIGH | IENH | R/W | 000-000-b |
| 00CFh | INT. REQUEST FLAG REG. HIGH | IRQH | R/W | 000-000-b |
| 00D0h | TIMER0 (16bit) MODE REG. | TM0 | R/W | 00000000b |
| 00D1h | TIMER1 (8bit) MODE REG. | TM1 | R/W | 00000000b |
| 00D2h | TIMER2 (8bit) MODE REG. | TM2 | R/W | 00000000b |
| 00D3h | TIMER0 HIGH-MSB DATA REG. | T0HMD | W | undefined |
| 00D4h | TIMER0 HIGH-LSB DATA REG. | T0HLD | W | undefined |
| 00D5h | TIMER0 LOW-MSB DATA REG. | T0LMD | W | undefined |
| | TIMER0 HIGH-MSB COUNT REG. | | R | undefined |
| 00D6h | TIMER0 LOW-LSB DATA REG. | T0LLD | W | undefined |
| | TIMER0 LOW-LSB COUNT REG. | | W | undefined |
| 00D7h | TIMER1 HIGH DATA REG. | T1HD | W | undefined |
| 00D8h | TIMER1 LOW DATA REG. | T1LD | W | undefined |
| | TIMER1 LOW COUNT REG. | | R | undefined |
| 00D9h | TIMER2 DATA REG. | T2DR | W | undefined |
| | TIMER2 COUNT REG. | | R | undefined |
| 00DAh | TIMER0 / TIMER1 MODE REG. | TM01 | R/W | 00000000b |
| 00DBh | Reserved | | | |
| 00DCh | STANDBY MODE RELEASE REG0 | SMPR0 | R/W | 00000000b |
| 00DDh | STANDBY MODE RELEASE REG0 | SMPR1 | R/W | 00000000b |
| 00DEh | PORT R1 OPEN DRAIN ASSIGN REG. | R1ODC | R/W | 00000000b |

| | | | | |
|-------|------------------------------------|-------|-----|-----------|
| 00DFh | PORT R2 OPEN DRAIN ASSIGN REG. | R2ODC | R/W | 00000000b |
| 00E0h | Reserved | | | |
| 00E1h | Reserved | | | |
| 00E2h | Reserved | | | |
| 00E3h | Reserved | | | |
| 00E4h | PORT R0 OPEN DRAIN ASSIGN REG. | R0ODC | R/W | 00000000b |
| 00E5h | Reserved | | | |
| 00E6h | Reserved | | | |
| 00E7h | Reserved | | | |
| 00E8h | Reserved | | | |
| 00E9h | Reserved | | | |
| 00EAh | Reserved | | | |
| 00EBh | Reserved | | | |
| 00ECh | Reserved | | | |
| 00EDh | Reserved | | | |
| 00EEh | Reserved | | | |
| 00EFh | Reserved | | | |
| 00F0h | SLEEP MODE REG. | SLPM | W | ----- 0b |
| 00F1h | Reserved | | | |
| 00F2 | Reserved | | | |
| 00F3h | Reserved | | | |
| 00F4h | Reserved | | | |
| 00F5h | Reserved | | | |
| 00F6h | STANDBY RELEASE LEVEL CONT. REG. 0 | SRLC0 | W | 00000000b |
| 00F7h | STANDBY RELEASE LEVEL CONT. REG. 1 | SRLC1 | W | 00000000b |
| 00F8h | PORT R0 PULL-UP REG. CONT. REG. | R0PC | W | 00000000b |
| 00F9h | PORT R1 PULL-UP REG. CONT. REG. | R1PC | W | 00000000b |
| 00FAh | PORT R2 PULL-UP REG. CONT. REG. | R2PC | W | 00000000b |
| 00FBh | Reserved | | | |
| 00FCh | Reserved | | | |
| 00FDh | Reserved | | | |
| 00FEh | Reserved | | | |
| 00FFh | Reserved | | | |

W

Registers are controlled by byte manipulation instruction such as LDM etc., do not use bit manipulation instruction such as SET1, CLR1 etc. If bit manipulation instruction is used on these registers, content of other seven bits are may varied to unwanted value.

R/W

Registers are controlled by both bit and byte manipulation instruction.

- : this bit location is reserved.

8.5 Addressing Mode

The HMS81004E/08E/16E/24E/32E uses six addressing modes;

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

(1) Register Addressing

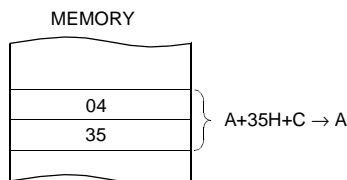
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

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

Example:

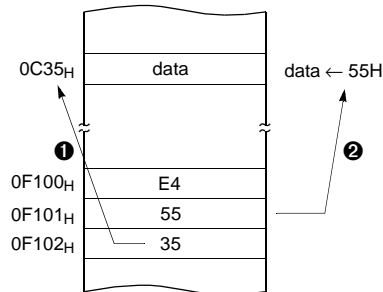
```
0435    ADC    #35H
```



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=0CH

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

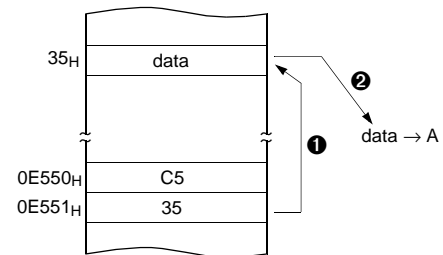


(3) Direct Page Addressing → dp

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

Example; G=0

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



(4) Absolute Addressing → !abs

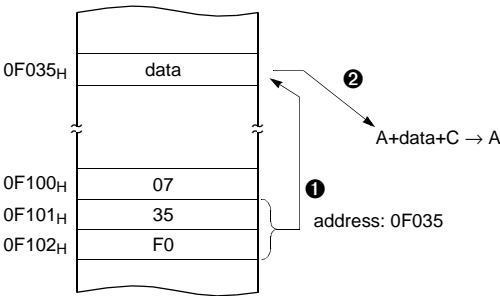
Absolute addressing sets corresponding memory data to Data, i.e. second byte(Operand I) of command becomes lower level address and third byte (Operand II) becomes upper level address.

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

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

Example;

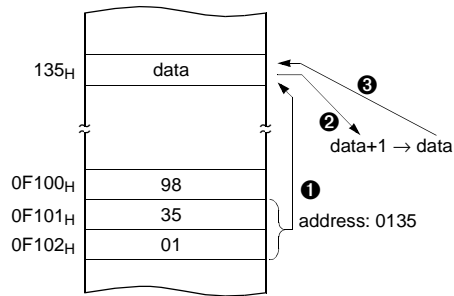
```
0735F0  ADC    !0F035H    ;A ←ROM[0F035H]
```



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

Example; Addressing accesses the address 0135H regardless of G-flag and RPR.

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



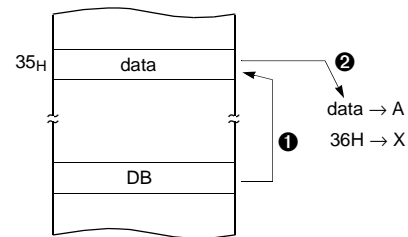
X indexed direct page, auto increment → {X}+

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=35H

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



(5) Indexed Addressing

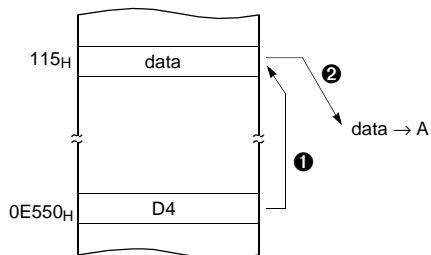
X indexed direct page (no offset) → {X}

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

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

Example; X=15H, G=1, RPR=01H

```
D4 LDA {X} ; ACC ← RAM[X].
```



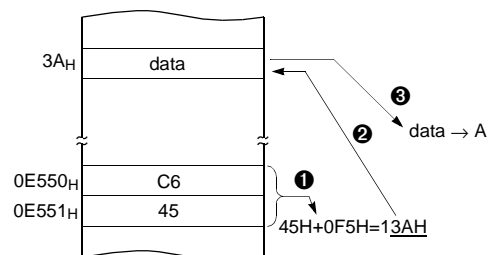
X indexed direct page (8 bit offset) → dp+X

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

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

Example; G=0, X=0F5H

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



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

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

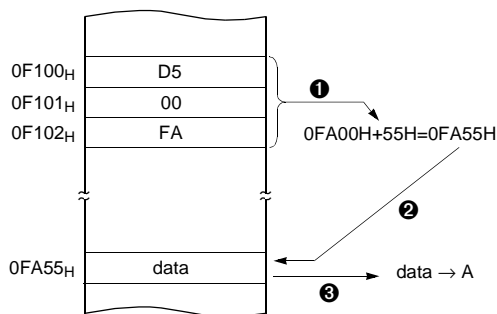
This is same with above (2). Use Y register instead of X.

Y indexed absolute → !abs+Y

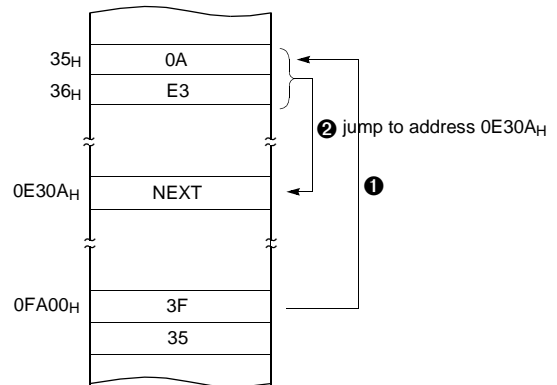
Sets the value of 16-bit absolute address plus Y-register data as Memory. This addressing mode can specify memory in whole area.

Example; Y=55H

D500FA LDA !0FA00H+Y



3F35 JMP [35H]

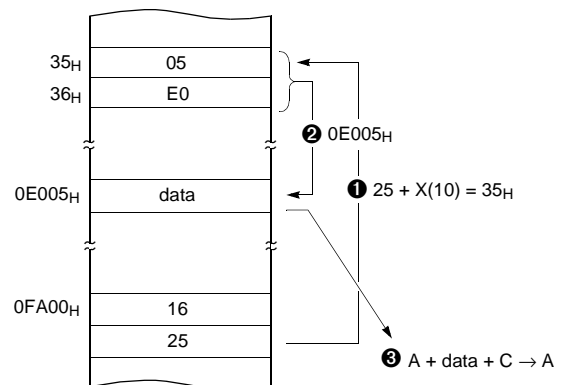
**X indexed indirect → [dp+X]**

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

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

Example; G=0, X=10H

1625 ADC [25H+X]

**(6) Indirect Addressing****Direct page indirect → [dp]**

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

JMP, CALL

Example; G=0

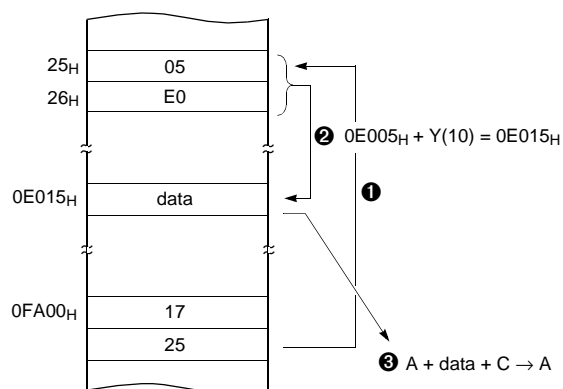
Y indexed indirect → [dp]+Y

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

1725 ADC [25H]+Y



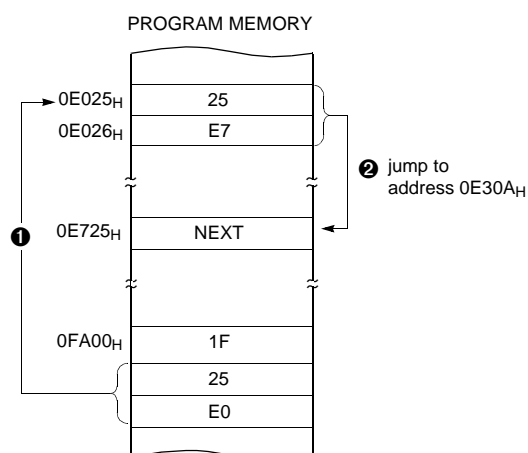
Absolute indirect → [!abs]

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

JMP

Example; G=0

1F25E0 JMP [!0C025H]



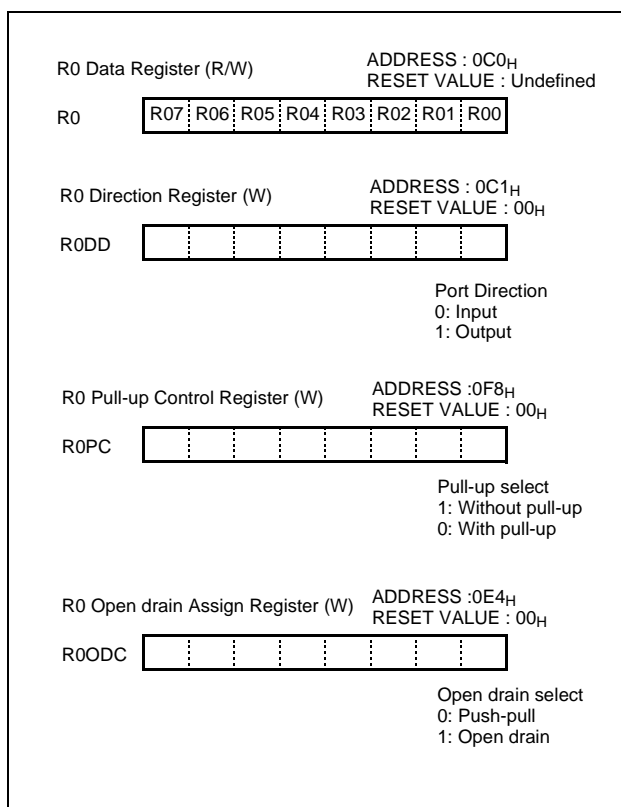
9. I/O PORTS

The HMS81004E/08E/16E/24E/32E has 24 I/O ports which are PORT0(8 I/O), PORT1 (8 I/O), PORT2 (8 I/O). Pull-up resistor of each port can be selectable by program. Each port contains data direction register which controls I/O and data register which stores port data.

9.1 R0 Ports

R0 is an 8-bit CMOS bidirectional I/O port (address 0C0H). Each I/O pin can independently used as an input or an output through the R0DD register (address 0C1H).

R0 has internal pull-ups that is independently connected or disconnected by R0PC. The control registers for R0 are shown below.



(1) R0 I/O Data Direction Register (R0DD)

R0 I/O Data Direction Register (R0DD) is 8-bit register, and can assign input state or output state to each bit. If R0DD is “1”, port R0 is in the output state, and if “0”, it is in the input state. R0DD is write-only register. Since R0DD is initialized as “00h” in reset state, the whole port R0 becomes input state.

(2) R0 Data Register (R0)

R0 data register (R0) is 8-bit register to store data of port R0. When set as the output state by R0DD, and data is written in R0, data is outputted into R0 pin. When set as the input state, input state of pin is read. The initial value of R0 is unknown in reset state.

(3) R0 Open drain Assign Register (R0ODC)

R0 Open Drain Assign Register (R0ODC) is 8bit register, and can assign R0 port as open drain output port each bit, if corresponding port is selected as output. If R0ODC is selected as “1”, port R0 is open drain output, and if selected as, “0” it is push-pull output. R0ODC is write-only register and initialized as “00h” in reset state.

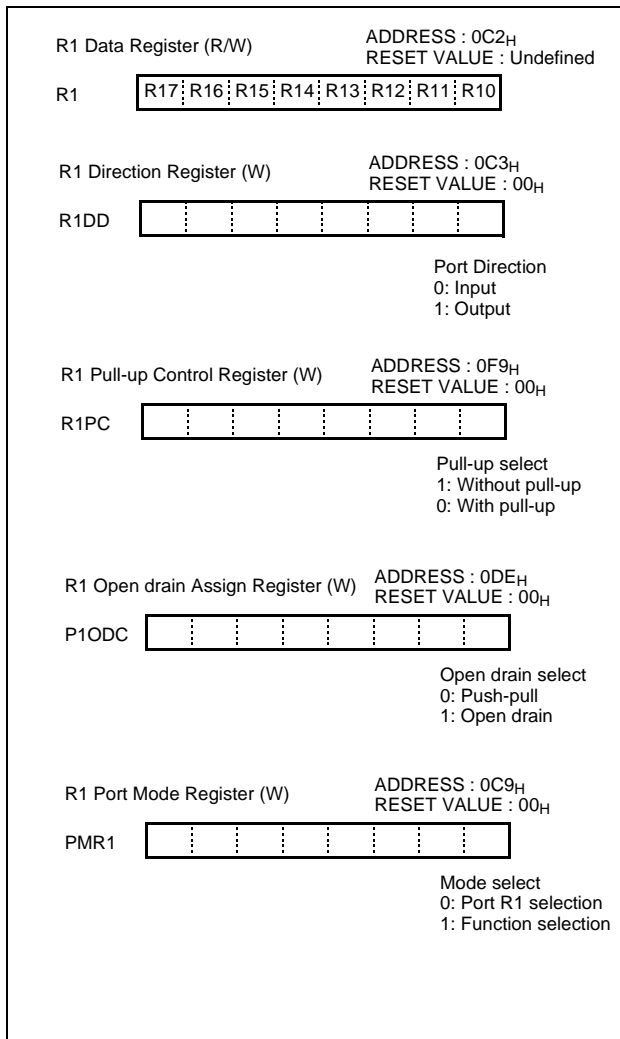
(4) R0 Pull-up Control Register (R0PC)

R0 Pull-up Control Register (R0PC) is 8-bit register and can control pull-up on or off each bit, if corresponding port is selected as input. If R0PC is selected as “1”, pull-up is disabled and if selected as “0”, it is enabled. R0PC is write-only register and initialized as “00h” in reset state. The pull-up is automatically disabled, if corresponding port is selected as output.

9.2 R1 Ports

R1 is an 8-bit CMOS bidirectional I/O port (address 0C2H). Each I/O pin can independently used as an input or an output through the R1DD register (address 0C3H).

R1 has internal pull-ups that is independently connected or disconnected by register R1PC. The control registers for R1 are shown below.



(1) R1 I/O Data Direction Register (R1DD)

R1 I/O Data Direction Register (R1DD) is 8-bit register, and can assign input state or output state to each bit. If R1DD is “1”, port R1 is in the output state, and if “0”, it is in the input state. R1DD is write-only register. Since R1DD is initialized as “00h” in reset state, the whole port R1 becomes input state.

(2) R1 Data Register (R1)

R1 data register (R1) is 8-bit register to store data of port R1. When set as the output state by R1DD, and data is written in R1, data is outputted into R1 pin. When set as the input state, input state of pin is read. The initial value of R1 is unknown in reset state.

(3) R1 Open drain Assign Register (R1ODC)

R1 Open Drain Assign Register (R1ODC) is 8bit register,

and can assign R1 port as open drain output port each bit, if corresponding port is selected as output. If R1ODC is selected as “1”, port R1 is open drain output, and if selected as “0”, it is push-pull output. R1ODC is write-only register and initialized as “00h” in reset state.

(4) R1 Port Mode Register (PMR1)

R1 Port Mode Register (PMR1) is 8-bit register, and can assign the selection mode for each bit. When set as “0”, corresponding bit of PMR1 acts as port R1 selection mode, and when set as “1”, it becomes function selection mode.

PMR1 is write-only register and initialized as “00h” in reset state. Therefore, becomes Port selection mode. Port R1 can be I/O port by manipulating each R1DD bit, if corresponding PMR1 bit is selected as “0”.

| Pin Name | PMR1 | Selection Mode | Remarks |
|----------|------|---------------------|----------------------|
| T0S | 0 | R17 (I/O) | - |
| | 1 | T0 (O) | Timer0 |
| T1S | 0 | R16 (I/O) | - |
| | 1 | T1 (O) | Timer1 |
| T2S | 0 | R15 (I/O) | - |
| | 1 | T2 (O) | Timer2 |
| ECS | 0 | R14 (I/O) | - |
| | 1 | \overline{EC} (I) | Timer0 Event |
| INT2S | 0 | R12 (I/O) | - |
| | 1 | INT2 (I) | Timer0 Input Capture |
| INT1S | 0 | R11 (I/O) | - |
| | 1 | INT1 (I) | - |
| | 0 | | |
| | 1 | | |

Table 9-1 Selection mode of PMR1

(5) R1 Pull-up Control Register (R1PC)

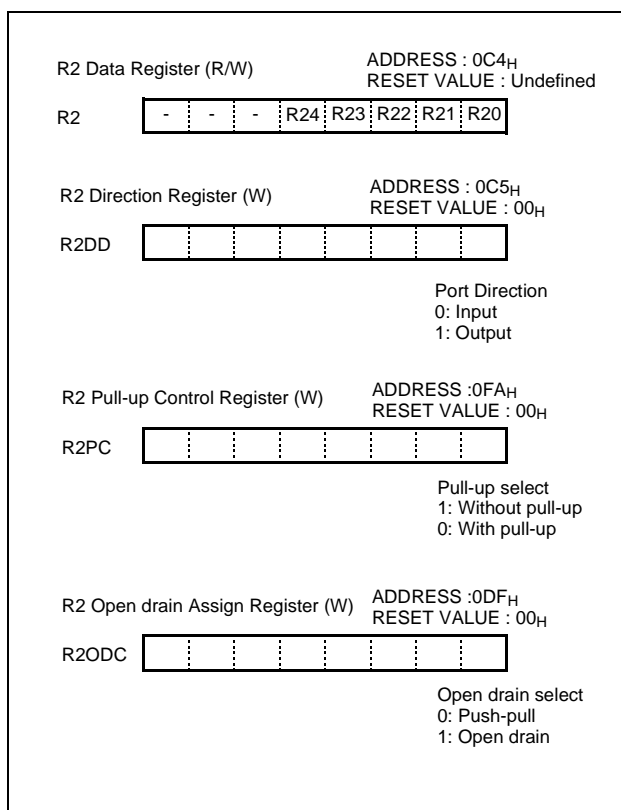
R1 Pull-up Control Register (R1PC) is 8-bit register and can control pull-up on or off each bit, if corresponding port is selected as input. If R1PC is selected as “1”, pull-up ia disabled and if selected as “0”, it is enabled. R1PC is write-only register and initialized as “00h” in reset state. The

pull-up is automatically disabled, if corresponding port is selected as output.

9.3 R2 Port

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

R2 has internal pull-ups that is independently connected or disconnected by R2PC (address 0FA_H). The control registers for R2 are shown as below.



(1) R2 I/O Data Direction Register (R2DD)

R2 I/O Data Direction Register (R2DD) is 8-bit register, and can assign input state or output state to each bit. If R2DD is "1", port R2 is in the output state, and if "0", it is in the input state. R2DD is write-only register. Since R2DD is initialized as "00h" in reset state, the whole port R2 becomes input state.

(2) R2 Data Register (R2)

R2 data register (R2) is 8-bit register to store data of port R2. When set as the output state by R2DD, and data is written in R2, data is outputted into R2 pin. When set as the input state, input state of pin is read. The initial value of R2 is unknown in reset state.

(3) R2 Open drain Assign Register (R2ODC)

R2 Open Drain Assign Register (R2ODC) is 8-bit register, and can assign R2 port as open drain output port each bit, if corresponding port is selected as output. If R2ODC is selected as "1", port R2 is open drain output, and if selected as "0", it is push-pull output. R2ODC is write-only register and initialized as "00h" in reset state.

(4) R2 Pull-up Control Register (R2PC)

R2 Pull-up Control Register (R2PC) is 8-bit register and can control pull-up on or off each bit, if corresponding port is selected as input. If R2PC is selected as "1", pull-up is disabled and if selected as "0", it is enabled. R2PC is write-only register and initialized as "00h" in reset state. The pull-up is automatically disabled, if corresponding port is selected as output.

10. CLOCK GENERATOR

Clock generating circuit consists of Clock Pulse Generator (C.P.G), Prescaler, Basic Interval Timer (B.I.T) and Watch Dog Timer. The clock applied to the Xin pin divided by two is used as the internal system clock.

Prescaler consist of 12-bit binary counter. The clock supplied from oscillation circuit is input to prescaler(fex)

The divided output from each bit of prescaler is provided to periphra hardware

Clock to peripheral hardware can be stopped by bit4 (ENPCK) of CKCTLR Register. ENPCK is set to "1" in reset

state.

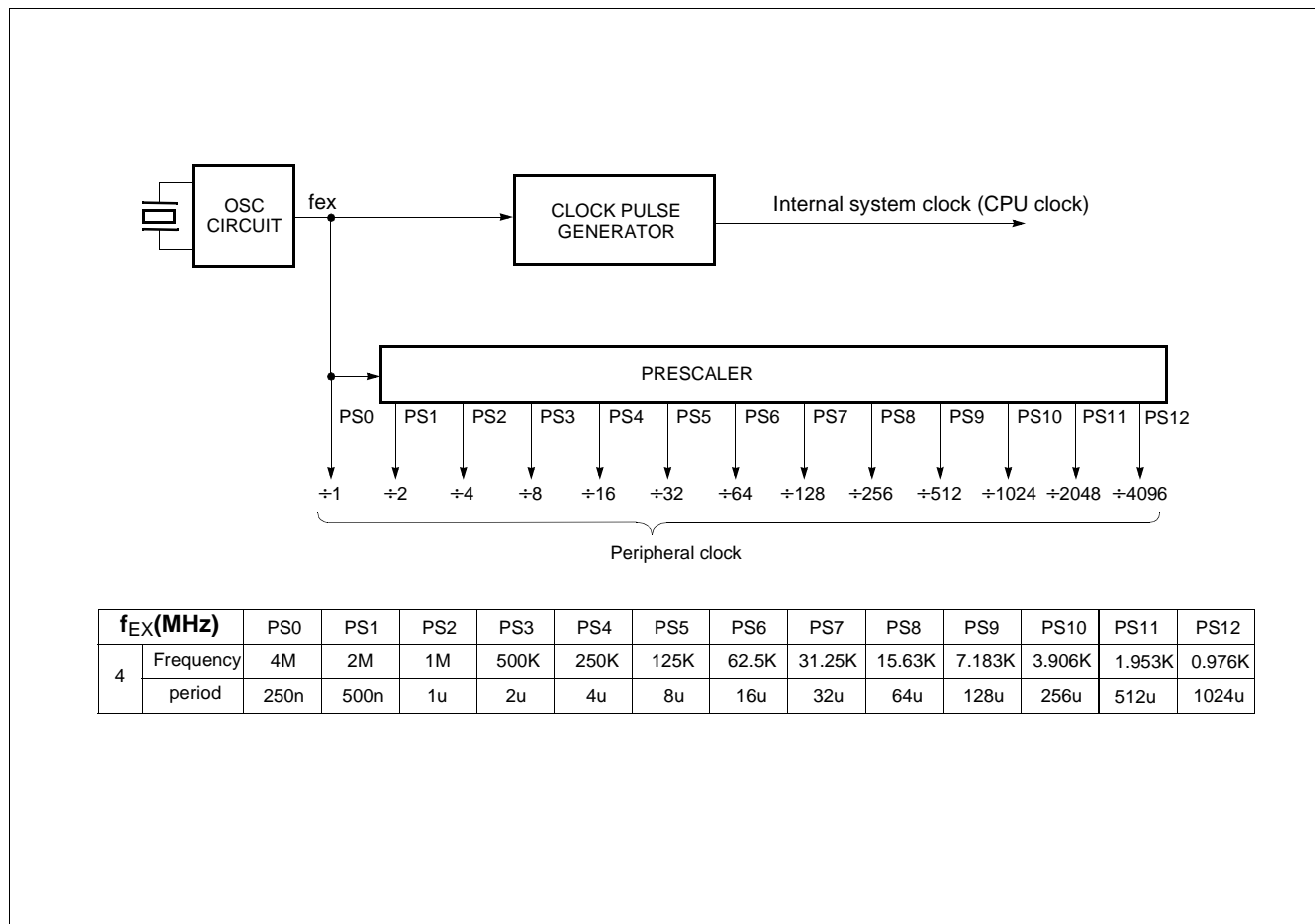
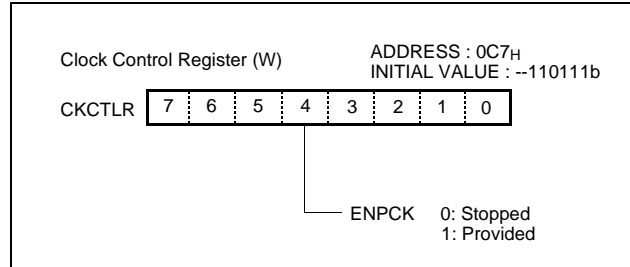


Figure 10-1 Block diagram of Clock Generator

10.1 Oscillation Circuit

Oscillation circuit is designed to be used either with a ceramic resonator or crystal oscillator. Figure 10-2 shows circuit diagrams using a crystal (or ceramic) oscillator. As shown in the diagram, oscillation circuits can be constructed by connecting an oscillator between Xout and Xin. Colck from oscillation circuit makes CPU clock via clock pulse generator, and then enters prescaler to make peripheral hardware clock. Alternately, the oscillator may be driven from an external source as Figure 10-3. In the STOP mode, oscillation stop, Xout state goes to "HIGH", Xin state goes to "LOW", and built-in feed back resistor is disabled.

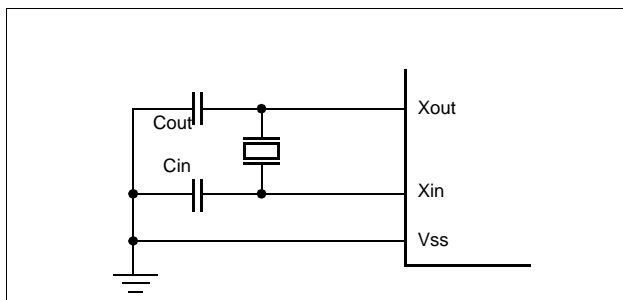


Figure 10-2 External Crystal(Ceramic) oscillator circuit

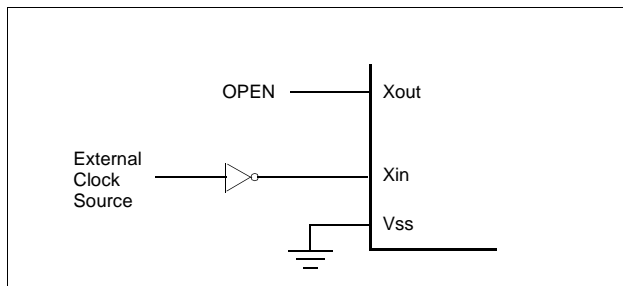


Figure 10-3 External clock input circuit

Oscillation circuit is designed to be used either with a ceramic resonator or crystal oscillator. Since each crystal and ceramic resonator have their own characteristics, the user should consult the crystal manufacturer for appropriate values of external components. In addition, see Figure 10-4 for the layout of the crystal.

Note: Minimize the wiring length. Do not allow the wiring to intersect with other signal conductors. Do not allow the 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 it to any ground pattern where high current is present. Do not fetch signals from the oscillator.

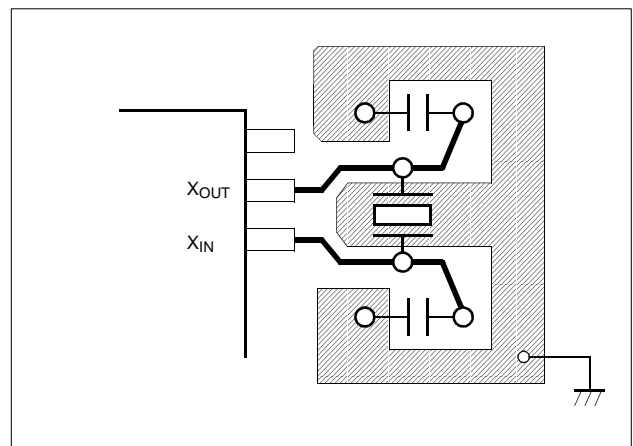


Figure 10-4 Recommend Layout of Oscillator PCB circuit

| Frequency | Resonator Maker | Part Name | Load Capacitor | Operating Voltage |
|-----------|-----------------|-----------------|----------------|-------------------|
| 2.00MHz | CQ | ZTT2.00 | Cin=Cout=open | 2.0~3.6 |
| | CQ | ZTA2.00 | Cin=Cout=30pF | 2.0~3.6 |
| | MURATA | CSTLS2M00G56-B0 | Cin=Cout=open | 2.0~3.6 |
| | MURATA | CSTCC2.00MG0H6 | Cin=Cout=open | 2.0~3.6 |
| | MURATA | CSTCC2M00G56-R0 | Cin=Cout=open | 2.0~3.6 |
| 4.00MHz | CQ | ZTT4.00 | Cin=Cout=open | 2.0~3.6 |
| | CQ | ZTA4.00 | Cin=Cout=30pF | 2.0~3.6 |
| | MURATA | CSTS0400MG06 | Cin=Cout=open | 2.0~3.6 |
| | MURATA | CSTLS4M00G56-B0 | Cin=Cout=open | 2.0~3.6 |
| | MURATA | CSTCR4M00G55-R0 | Cin=Cout=open | 2.0~3.6 |
| | TDK | FCR4.0MC5 | Cin=Cout=open | 2.0~3.6 |
| | TDK | FCR4.0MSC5 | Cin=Cout=open | 2.0~3.6 |
| | CORETECK | CRT4.00MS | Cin=Cout=open | 2.0~3.6 |
| | CORETECK | CRM4.00MS | Cin=Cout=30pF | 2.0~3.6 |

Table 10-1 Recommendable resonator

11. BASIC INTERVAL TIMER

The HMS81004E/08E/16E/24E/32E has one 8-bit Basic Interval Timer that is free-run and can not stop. Block diagram is shown in Figure 11-1 .

The Basic Interval Timer generates the time base for Standby release time, watchdog timer counting, and etc. It also provides a Basic interval timer interrupt (IFBIT). As the count overflow from FF_H to 00_H, this overflow causes the interrupt to be generated.

- 8bit binary up-counter
- Use the bit output of prescaler as input to secure the oscillation stabilization time after power-on
- Secures the oscillation stabilization time in standby mode (stop mode) release
- Contents of B.I.T can be read
- Provides the clock for watch dog timer

The Basic Interval Timer is controlled by the clock control register (CKCTLR) shown in Figure 11-2 . If bit3(BTCL) of CKCTLR is set to "1", B.I.T is cleared, and then, after one machine cycle, BTCL becomes "0", and B.I.T starts counting. BTCL is set to "0" in reset state.

The input clock of B.I.T can be selected from the prescaler within a range of 2 μ s to 256 μ s by clock input selection bits (BTS2~BTS0). (at f_{ex} = 4MHz). In reset state, or power on reset, BTS2="1", BTS1= "1", BTS0= "1" to secure the longest oscillation stabilization time. B.I.T can generate the wide range of basic interval time interrupt request (IFBIT) by selecting prescaler output.

By reading of the Basic Interval Timer Register (BITR), we can read counter value of B.I.T. Because B.I.T can be cleared or read, the spending time up to maximum 65.5ms can be available. B.I.T is read-only register. If B.I.T register is written, then CKCTLR register with same address is written.

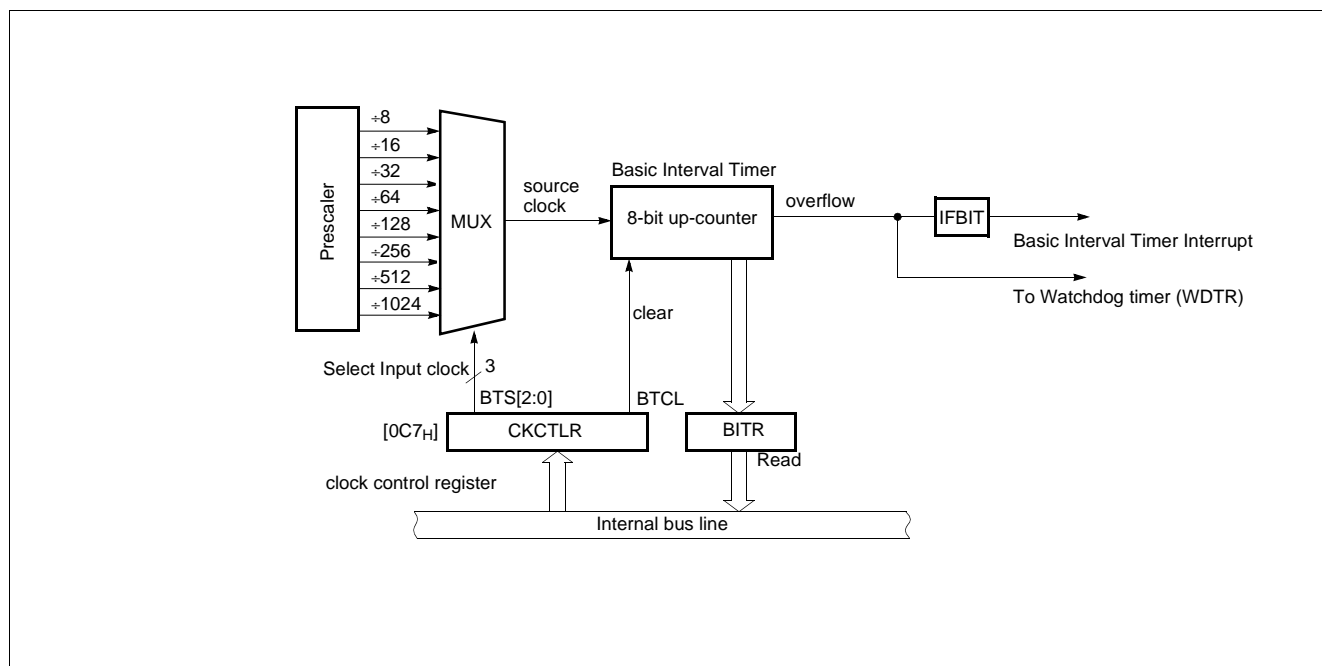


Figure 11-1 Block diagram of Basic Interval Timer

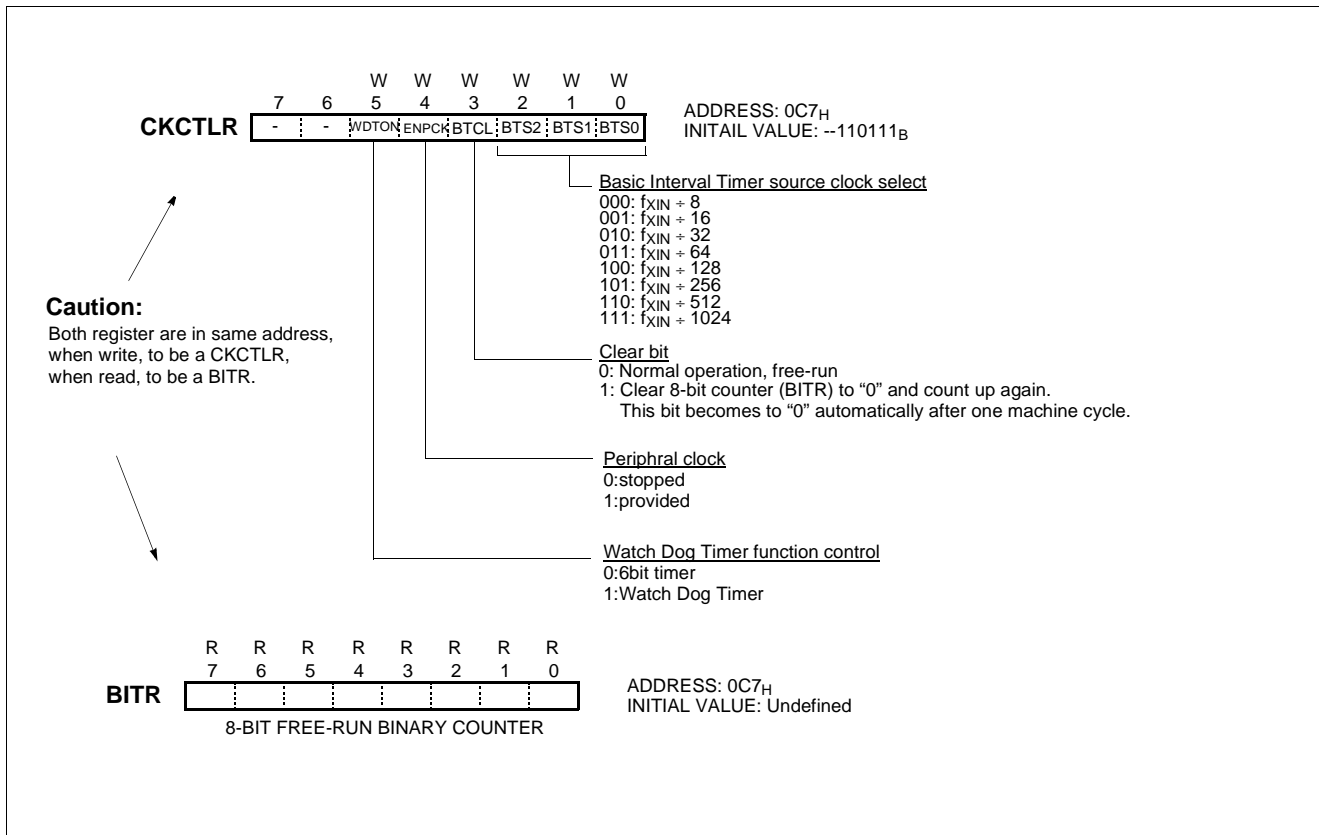


Figure 11-2 CKCTLR AND BITR

| BTS[2:0] | CPU Source clock | B.I.T. Input clock@4Mhz(us) | Standby release time(ms) |
|----------|------------------|-----------------------------|--------------------------|
| 000 | ÷ 8 | 2 | 0.512 |
| 001 | ÷16 | 4 | 1.024 |
| 010 | ÷32 | 8 | 2.048 |
| 011 | ÷64 | 16 | 4.096 |
| 100 | ÷128 | 32 | 8.192 |
| 101 | ÷256 | 64 | 16.384 |
| 110 | ÷512 | 128 | 32.768 |
| 111 | ÷1024 | 256 | 65.536 |

12. WATCH DOG TIMER

Watch Dog Timer (WDT) consists of 6-bit binary counter, 6-bit comparator, and Watch Dog Timer Register (WDTR). Watch Dog Timer can be used 6-bit general Timer or specific Watch dog timer by setting bit5 (WDTON) of Clock Control Register (CKCTL). By assigning bit6 (WDTCL) of WDTR, 6-bit counter can be cleared.

WDT Interrupt (IFWDT) interval is determined by the interrupt IFBIT interval of Basic Interval Timer and the value of WDT Register.

-Interval of IFWDT = (IFBIT interval) * (WDTR value)

As IFBIT (Basic Interval Timer Interrupt Request) is used for input clock of WDT, Input clock cycle is possible from 512 us to 65,536 us by BTS. (at fex = 4MHz)

*At Hardware reset time, WDT starts automatically. Therefore the user must select the CKCTL and WDTR before WDT overflow.

-Reset WDTR value = $0F_h = 15$

-Interval of WDT = $65,536 * 15 = 983040 \text{ us}$

(about 1second)

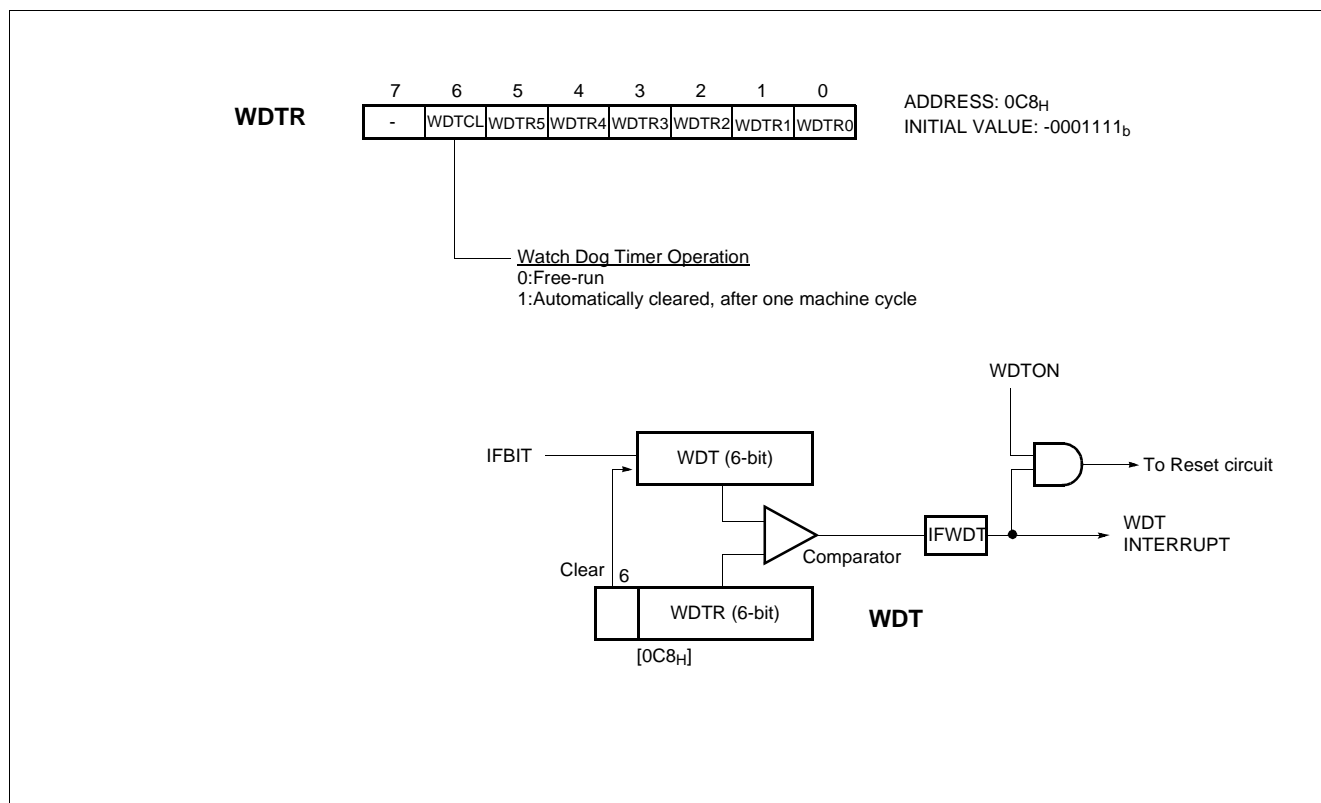


Figure 12-1 Block diagram of Watch Dog Timer

Note: When WDTR Register value is 63 (3Fh)
(Caution) : Do not use "0" for WDTR Register value.

Device come into the reset state by WDT

13. Timer0, Timer1, Timer2

(1) Timer Operation Mode

Timer consists of 16bit binary counter Timer0 (T0), 8bit binary Timer1 (T1), Timer2 (T2), Timer Data Register, Timer Mode Register (TM01, TM0, TM1, TM2) and control circuit. Timer Data Register Consists of Timer0 High-MSB Data Register (T0HMD), Timer0 High-LSB Data Register (T0HLD), Timer0 Low-MSB Data Register (T0LMD), Timer0 Low-LSB Data Register (T0LLD), Timer1 High Data Register (T1HD), Timer1 Low Data

Register (T1LD), Timer2 Data Register (T2DR). Any of the PS0 ~ PS5, PS11 and external event input \overline{EC} can be selected as clock source for T0. Any of the PS0 ~ PS3, PS7 ~ PS10 can be selected as clock T1. Any of the PS5 ~ PS12 can be selected as clock source for T2.

* Relevant Port Mode Register (PMR1 : 00C9h) value should be assigned for event counter.

| | | |
|--------|--|--|
| Timer0 | <ul style="list-style-type: none">- 16-bit Interval Timer- 16-bit Event Counter- 16-bit Input Capture- 16-bit rectangular-wave output | <ul style="list-style-type: none">- Single/Modulo-N Mode- Timer Output Initial Value Setting- Timer0~Timer1 combination Logic Output- One Interrupt Generating Every 2nd Counter Overflow |
| Timer1 | <ul style="list-style-type: none">- 8-bit Interval Timer- 8-bit rectangular-wave output | |
| Timer2 | <ul style="list-style-type: none">- 8-bit Interval Timer- 8-bit rectangular-wave output- Modulo-N Mode | |

Table 13-1 Timer Operation

| 16bit Timer (T0) | | 8bit Timer (T1) | | 8bit Timer (T2) | |
|------------------|--------------|-----------------|------------|-----------------|------------|
| Resolution | MAX. Count | Resolution | MAX. Count | Resolution | MAX. Count |
| PS0 (0.25us) | 16,384us | PS0 (0.25us) | 64us | PS5 (8us) | 2,048us |
| PS1 (0.5us) | 32,768us | PS1 (0.5us) | 128us | PS6 (16us) | 4,096us |
| PS2 (1us) | 65,536us | PS2 (1us) | 256us | PS7 (32us) | 8,192us |
| PS3 (2us) | 131,072us | PS3 (2us) | 512us | PS8 (64us) | 16,384us |
| PS4 (4us) | 262,144us | PS7 (32us) | 8,192us | PS9 (128us) | 32,768us |
| PS5 (8us) | 524,288us | PS8 (64us) | 16,384us | PS10 (256us) | 65,536us |
| PS11 (512us) | 33,554,432us | PS9 (128us) | 32,768us | PS11 (512us) | 131,072us |
| \overline{EC} | - | PS10 (256us) | 65,536us | PS12 (1024us) | 262,144us |

Table 13-2 Function of Timer & Counter

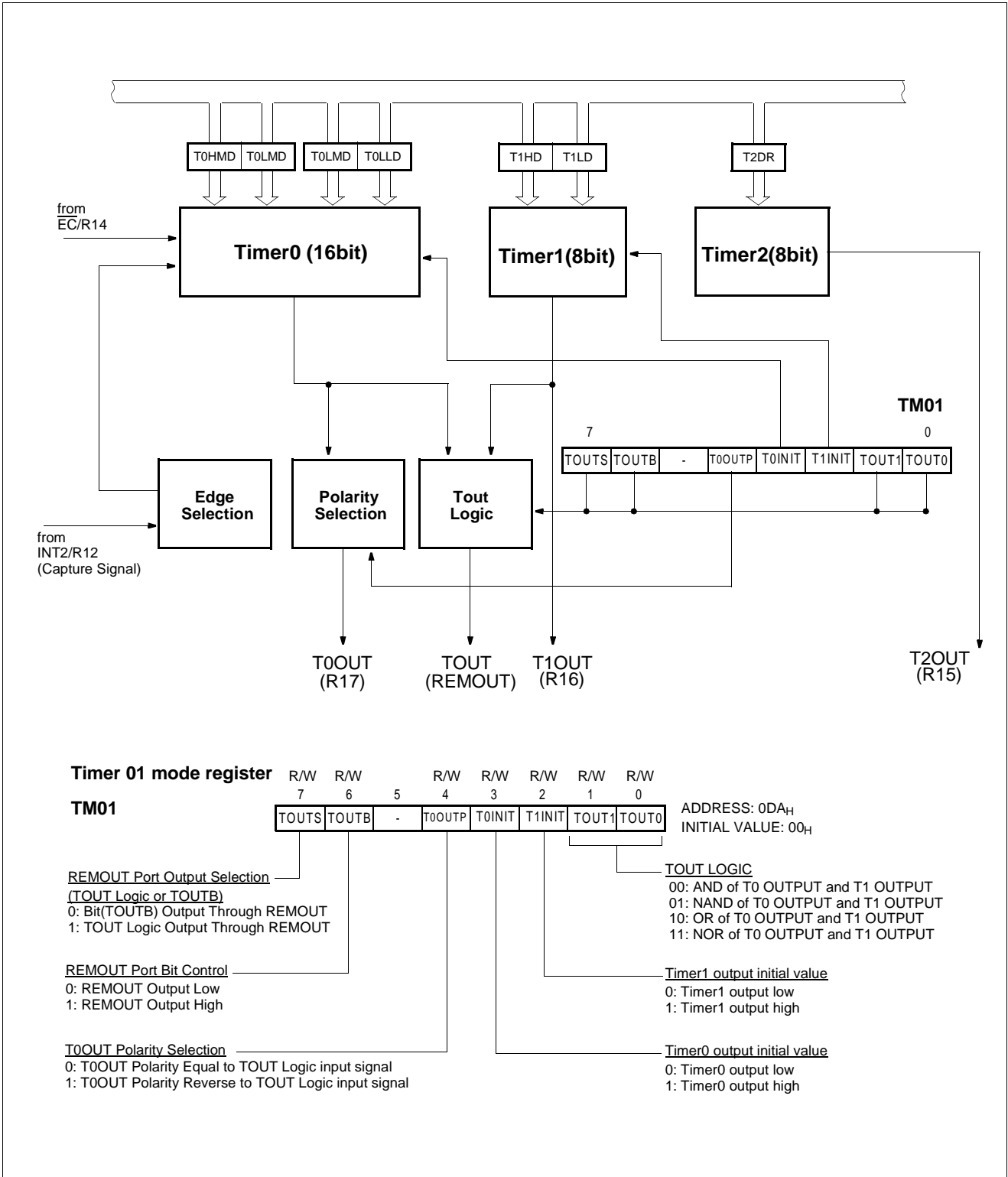


Figure 13-1 Block Diagram of Timer/Counter

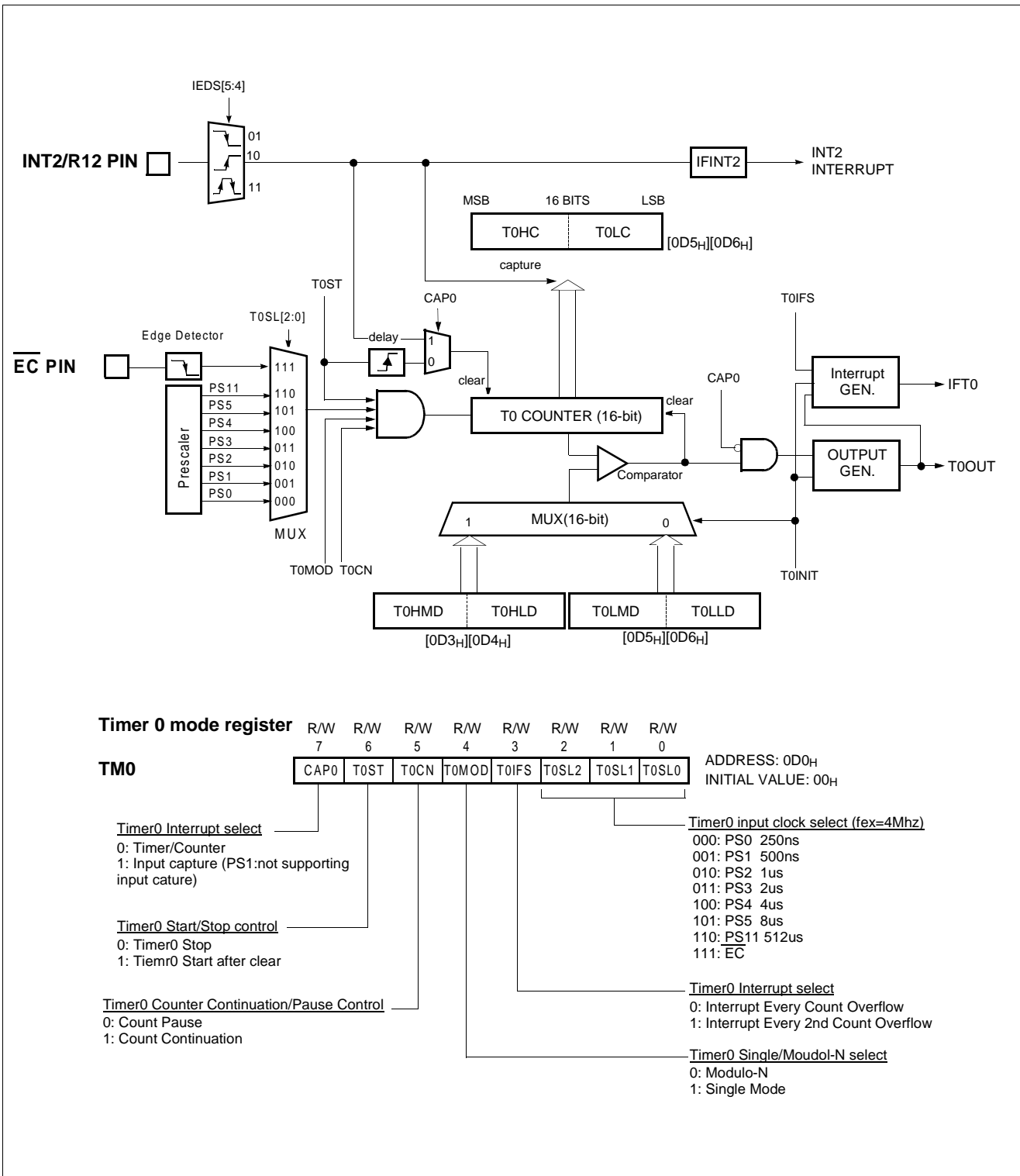
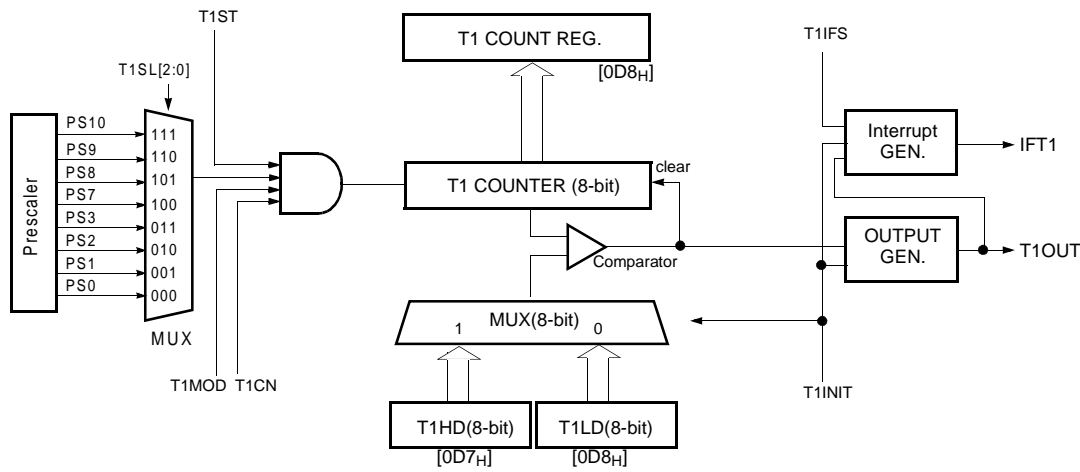


Figure 13-2 Block Diagram of Timer0



Timer 1 mode register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|------------|------|------|-------|-------|---|-------|-------|-------|-------------------------------------|
| TM1 | T1ST | T1CN | T1MOD | T1IFS | - | T1SL2 | T1SL1 | T1SL0 | ADDRESS: 0D1H INITIAL VALUE: 00H |

Timer1 Start/Stop control
0: Timer1 Stop
1: Timer1 Start after clear

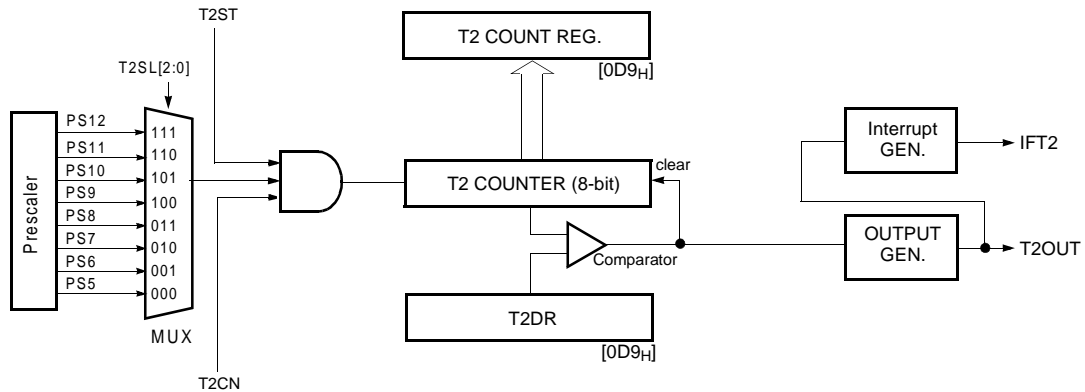
Timer1 Counter Continuation/Pause Control
0: Count Pause
1: Count Continuation

Timer1 Single/Module-N select
0: Module-N
1: Single Mode

Timer1 input clock select (fex=4Mhz)
000: PS0 250ns
001: PS1 500ns
010: PS2 1us
011: PS3 2us
100: PS7 32us
101: PS8 64us
110: PS9 128us
111: PS10 256us

Timer1 Interrupt select
0: Interrupt Every Count Overflow
1: Interrupt Every 2nd Count Overflow

Figure 13-3 Block Diagram of Timer1



Timer 2 mode register

TM2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|------|------|-------|-------|-------|
| - | - | - | T2ST | T2CN | T2SL2 | T2SL1 | T2SL0 |

ADDRESS: 0D2_H
INITIAL VALUE: 00_H

Timer2 Start/Stop control

0: Timer2 Stop
1: Timer2 Start after clear

Timer2 Counter Continuation/Pause Control

0: Count Pause
1: Count Continuation

Timer2 input clock select (fex=4Mhz)

000: PS5 8us
001: PS6 16us
010: PS7 32us
011: PS8 64us
100: PS9 128us
101: PS10 256us
110: PS11 512us
111: PS12 1,024us

Figure 13-4 Block Diagram of Timer2

2) Timer0, Timer1

TIMER0 and TIMER1 have an up-counter. When value of the up-counter reaches the content of Timer Data Register

(TDR), the up-counter is cleared to "00h", and interrupt (IFT0, IFT1) is occurred at the next clock.

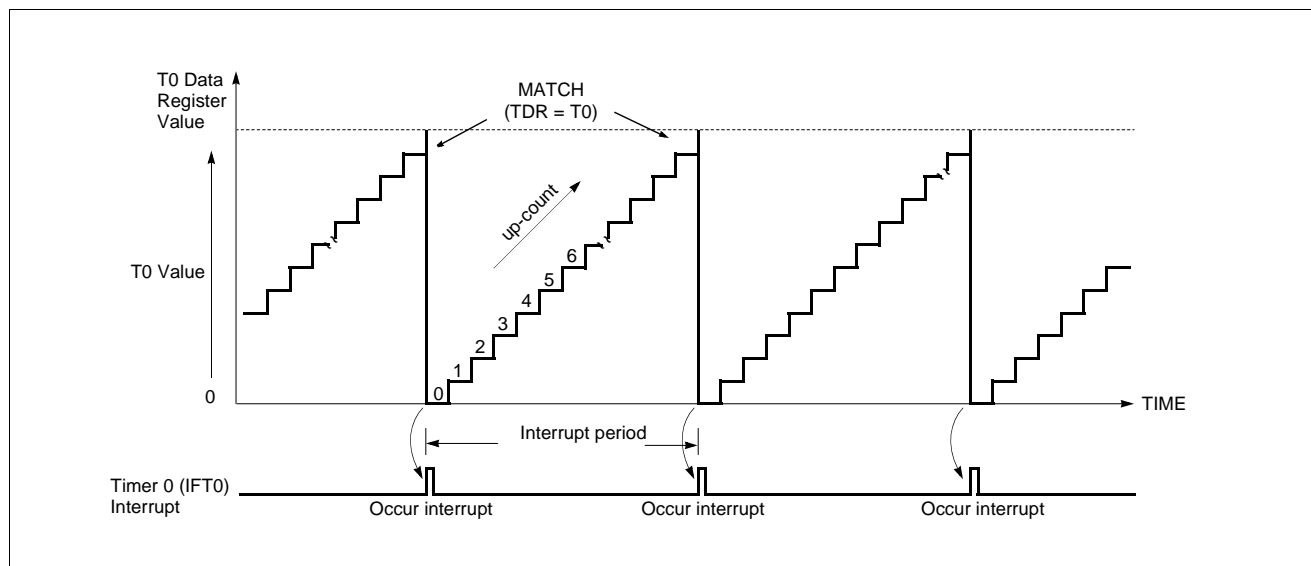


Figure 13-5 Operation of Timer0

For Timer0, the internal clock (PS) and the external clock (\overline{EC}) can be selected as counter clock. But Timer1 and Timer2 use only internal clock. As internal clock. Timer0 can be used as internal-timer which period is determined by Timer Data Register (TDR). Chosen as external clock, Timer0 executes as event-counter. The counter execution of Timer0 and Timer1 is controlled by T0CN, T0ST, CAP0, T1CN, T1ST, of Timer Mode Register TM0 and TM1. T0CN, T1CN are used to stop and start Timer0 and Timer1 without clearing the counter. T0ST, T1ST is used to clear the counter. For clearing and starting the counter, T0ST or T1ST should be temporarily set to "0" and then set to "1". T0CN, T1CN, T0ST and T1ST should be set "1", when Timer counting-up. Controlling of CAP0 enables Timer0 as input capture. By programming of CAP0 to "1", the period of signal from INT2 can be measured and then, event counter value for INT2 can be read. During counting-up, value of counter can be read-Timer execution is stopped by the reset signal(RE-

SET="L")

Note: In the process of reading 16-bit Timer Data, first read the upper 8-bit data. Then read the lower 8-bit data, and read the upper 8-bit data again. If the earlier read upper 8-bit data are matched with the later read upper 8-bit data, read 16-bit data are correct. If not, caution should be taken in the selection of upper 8-bit data.

(Example)

1) Upper 8-bit Read 0A 0A

2) Lower 8-bit Read FF 01

3) Upper 8-bit Read 0B 0B

=====

0AFF 0B01

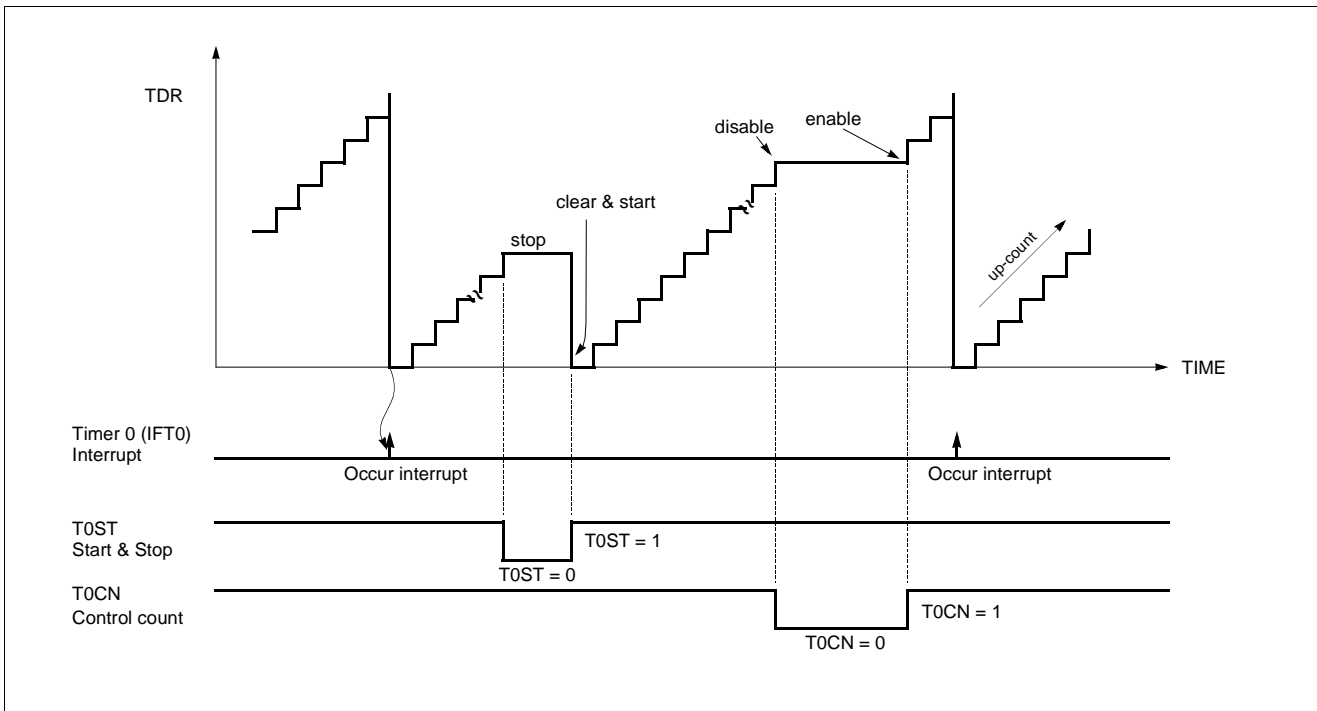


Figure 13-6 Start/Stop Operation of Timer0

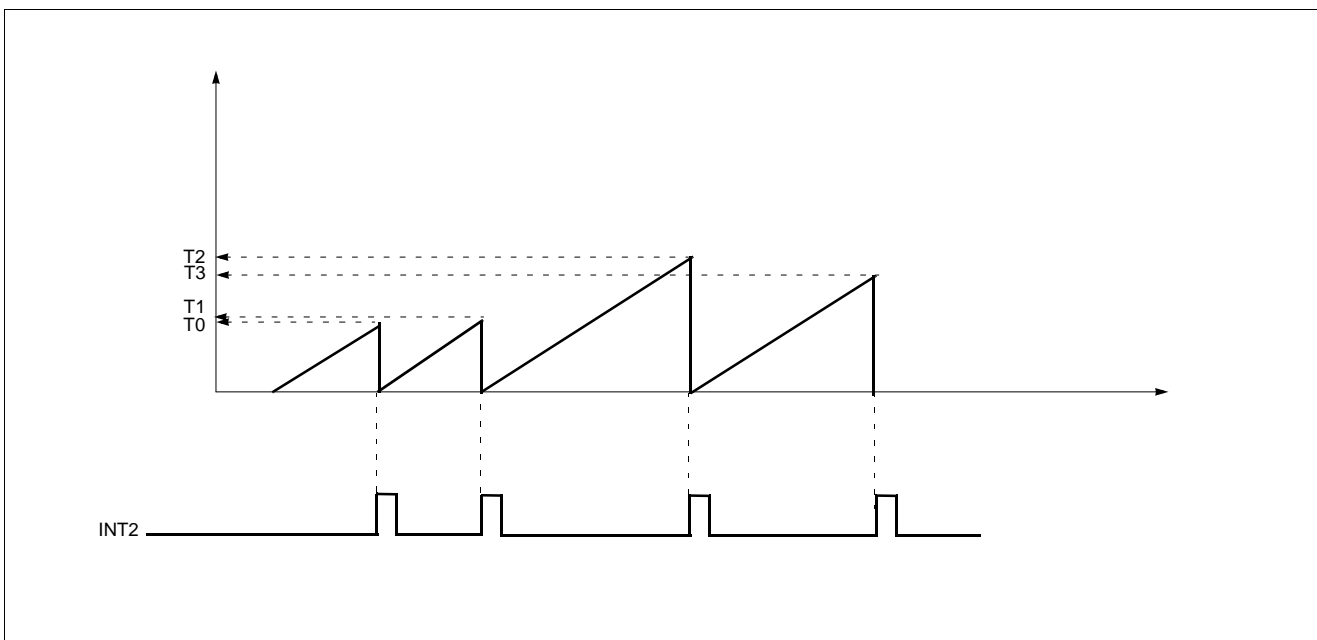


Figure 13-7 Input capture operation of Timer0

3) Single/Modulo-N Mode

Timer0 (Timer1) can select initial (T0INIT, T1INIT of TM01) output level of Timer Output port. If initial level is “L”, Low-Data Register value of Timer Data Register is transferred to comparator and T0OUT (T1OUT) is to be “Low”, if initial level is High? High -Data Register is transferred and to be “High”. Single Mode can be set by Mode Select bit (T0MOD, T1MOD) of Timer Mode Register (TM0, TM1) to “1” When used as Single Mode, Timer counts up and compares with value of Data Register. If the result is same, Time Out interrupt occurs and level of Timer Output port toggle, then counter stops as reset state. When used as Modulo-N Mode, T0MOD (T1MOD)

should be set “0”. Counter counts up until the value of Data Register and occurs Time-out interrupt. The level of Timer Output port toggle and repeats process of

counting the value which is selected in Data Register. During Modulo-N Mode, If interrupt select bit (T0IFS, T1IFS) of Mode Register is “0”, Interrupt occurs on every Time-out. If it is “1”, Interrupt occurs every second time-out.

Note: Timer Output is toggled whenever time out happen

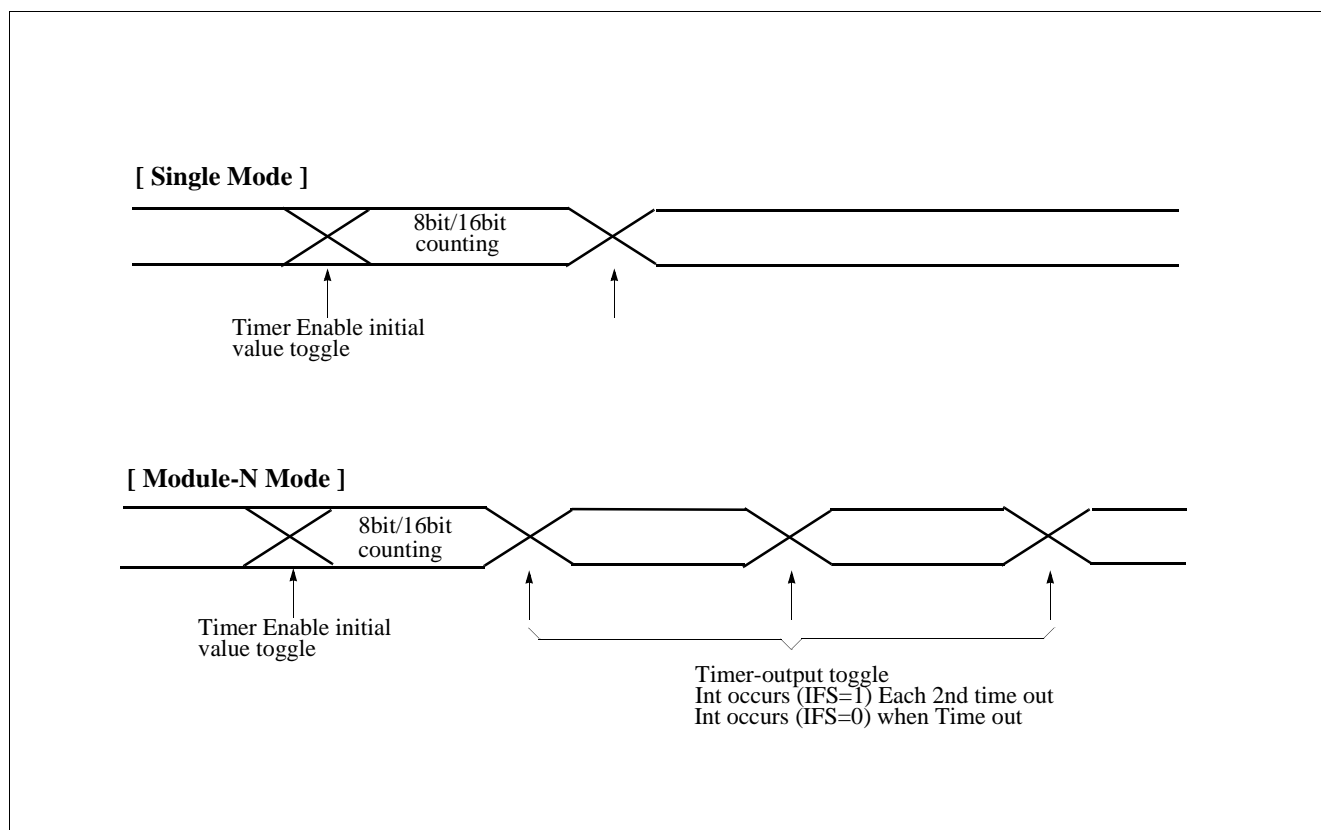


Figure 13-8 Operation Diagram for Single/Modulo-N Mode

(4) Timer 2

Timer2 operates as a up-counter. The content of T2DR are compared with the contents of up-counter. If a match is found. Timer2 interrupt (IFT2) is generated and the up-counter is cleared to “00h”. Therefore, Timer2 executes as a interval timer. Interrupt period is determined by the count source clock for the Timer2 and content of T2DR.

When T2ST is set to “1”, count value of Timer 2 is cleared and starts counting-up. For clearing and starting the Timer2. T2ST have to set to “1” after set to “0”. In order to write a value directly into the T2DR, T2ST should be set to “0”. Count value of Timer2 can be read at any time.

14. INTERRUPTS

The HMS81004E/08E/16E/24E/32E interrupt circuits consist of Interrupt Mode Register (MOD), Interrupt enable register (IENH, IENL), Interrupt request flags of IRQH, IRQL, Priority circuit and Master enable flag ("I" flag of PSW). 8 interrupt sources are provided. The configuration of interrupt circuit is shown in Figure 14-1.

The HMS81004E/08E/16E/24E/32E contains 8 interrupt sources; 3 externals and 5 internals. Nested interrupt services with priority control is also possible. Software interrupt is non-maskable interrupt, the others are all maskable interrupts.

- 8 interrupt source (2Ext, 3Timer, BIT, WDT and Key Scan)
- 8 interrupt vector
- Nested interrupt control is possible
- Programmable interrupt mode (Hardware and software interrupt accept mode)
- Read and write of interrupt request flag are possible.
- In interrupt accept, request flag is automatically cleared.

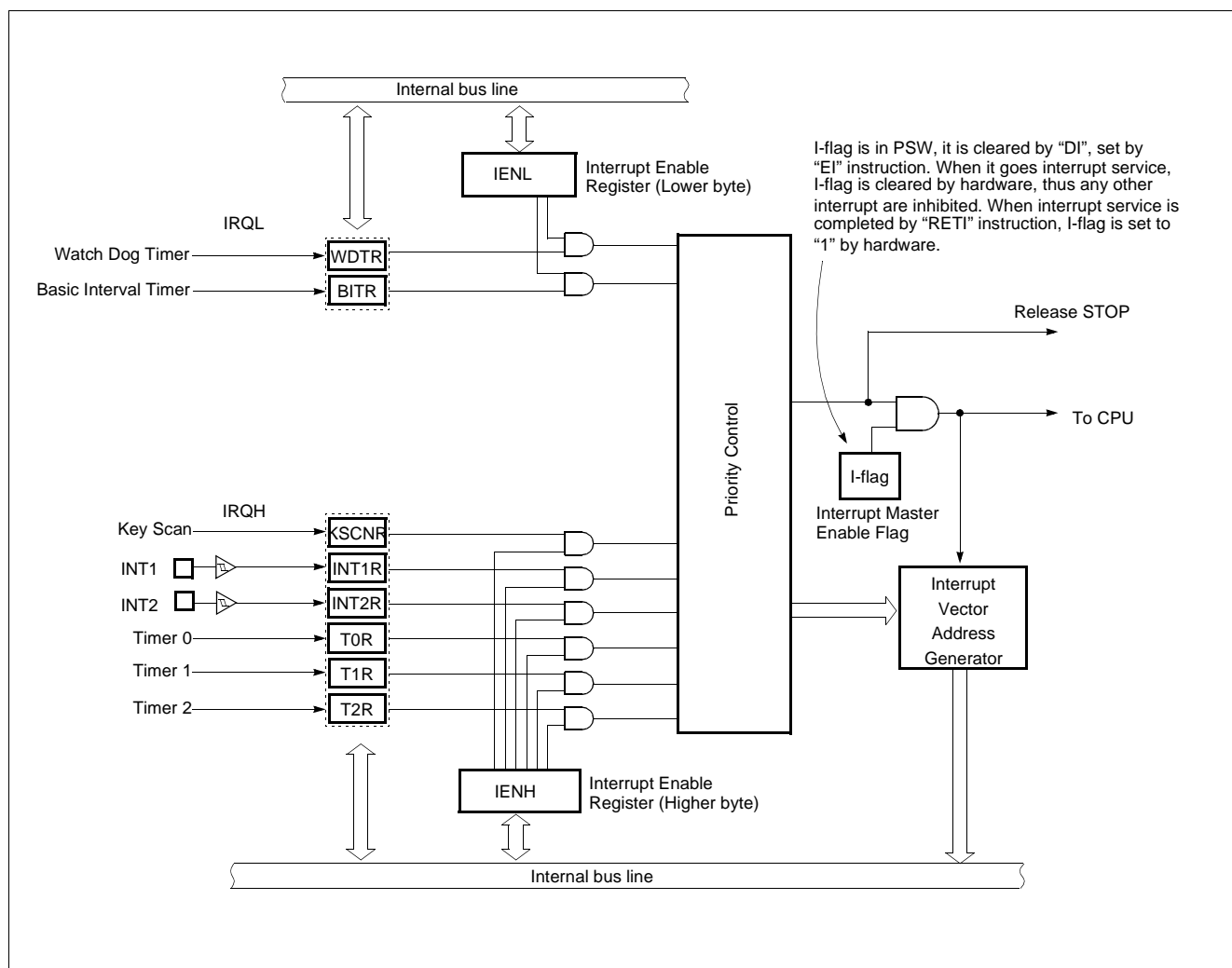


Figure 14-1 Block Diagram of Interrupt

14.1 Interrupt priority and sources

Each interrupt vector is independent and has its own priority. Software interrupt (BRK) is also available. Interrupt

source classification is shown in Table 14-1.

14.2 Interrupt control register

I flag of PSW is a interrupt mask enable flag. When I flag = “0”, all interrupts become disable. When I flag = “1”, interrupts can be selectively enabled and disabled by contents of corresponding Interrupt Enable Register. When interrupt is occurred, interrupt request flag is set, and Interrupt request is detected at the edge of interrupt signal. The accepted interrupt request flag is automatically cleared during interrupt cycle process. The interrupt request flag maintains “1” until the interrupt is accepted or is cleared in program. In reset state, interrupt request flag register (IRQH, IRQL) is cleared to “0”. It is possible to read the state of interrupt register and to manipulate the contents of register and to generate interrupt. (Refer to software interrupt)

| Reset/Interrupt | Symbol | Priority |
|----------------------|---------------------------|----------|
| Hardware Reset | $\overline{\text{RESET}}$ | - |
| Key Scan | KSCNR | 1 |
| External Interrupt1 | INT1R | 2 |
| External Interrupt2 | INT2R | 3 |
| Timer0 | T0R | 4 |
| Timer1 | T1R | 5 |
| Timer2 | T2R | 6 |
| Watch Dog Timer | WDTR | 7 |
| Basic Interval Timer | BITR | 8 |
| BRK Instruction | BRK | - |

Table 14-1 Interrupt Source

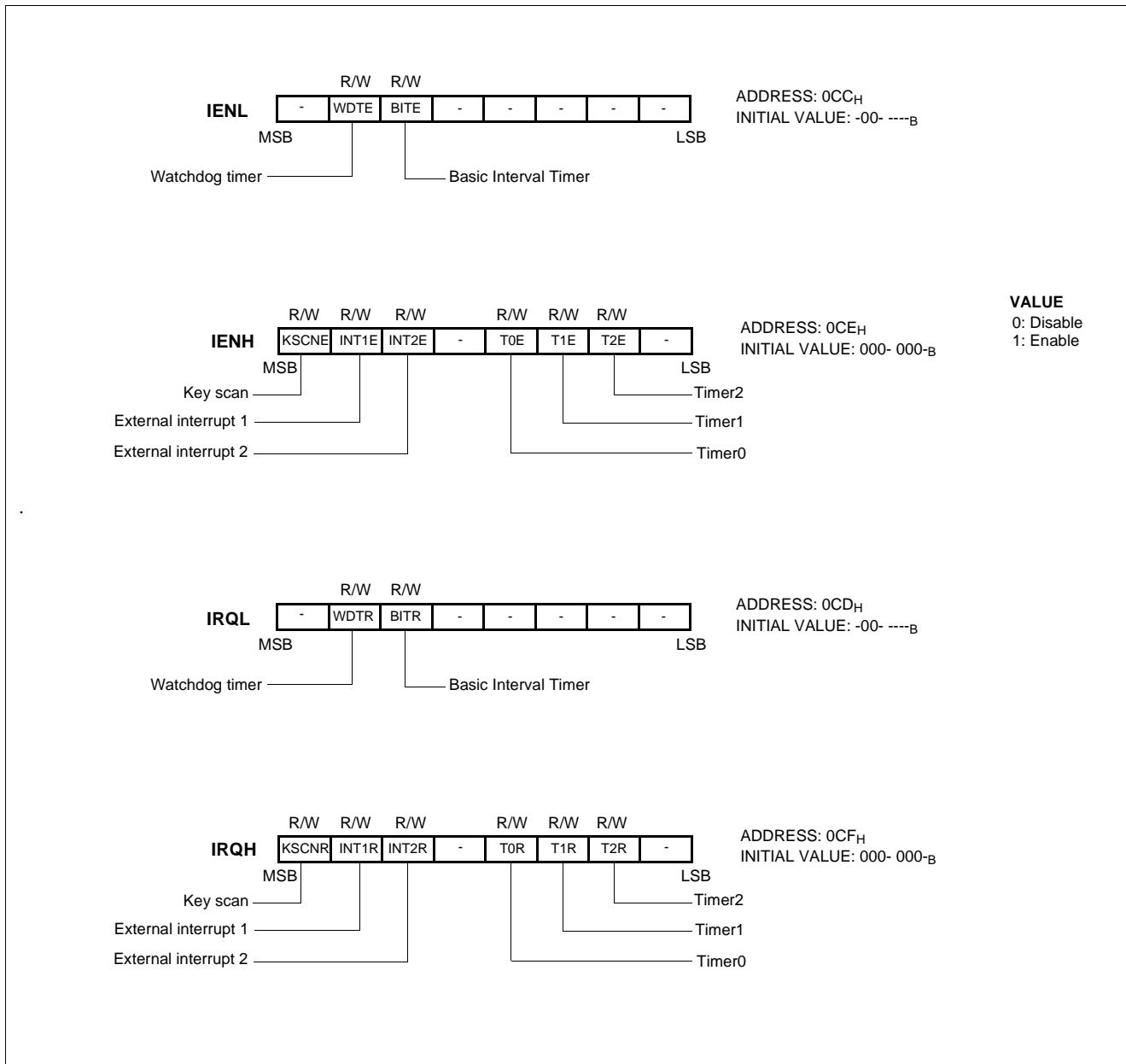


Figure 14-2 Interrupt Enable & Request Flag

14.3 Interrupt accept mode

The interrupt priority order is determined by bit (IM1, IM0) of IMOD register. The condition allow for accepting interrupt is set state of the interrupt mask enable flag and

the interrupt enable bit must be “1”. In Reset state, these IP3 - IP0 registers become all “0”.

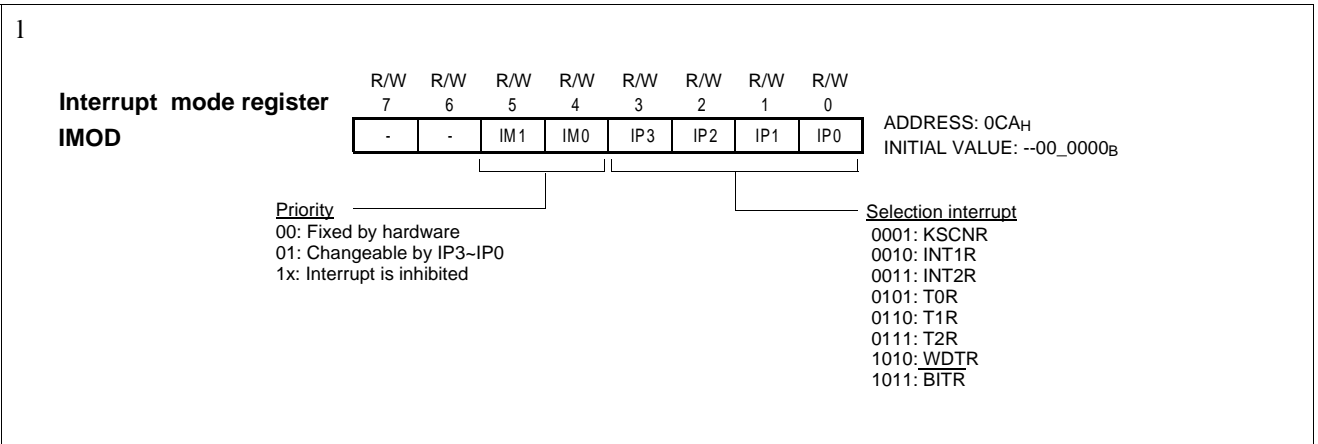


Figure 14-3 Interrupt Accept Mode & Selection by IP3~IP0

14.4 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_{XIN} after the completion of the current instruction execution. The interrupt service task is terminated upon execution of an interrupt return instruction [RETI].

Interrupt acceptance

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

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

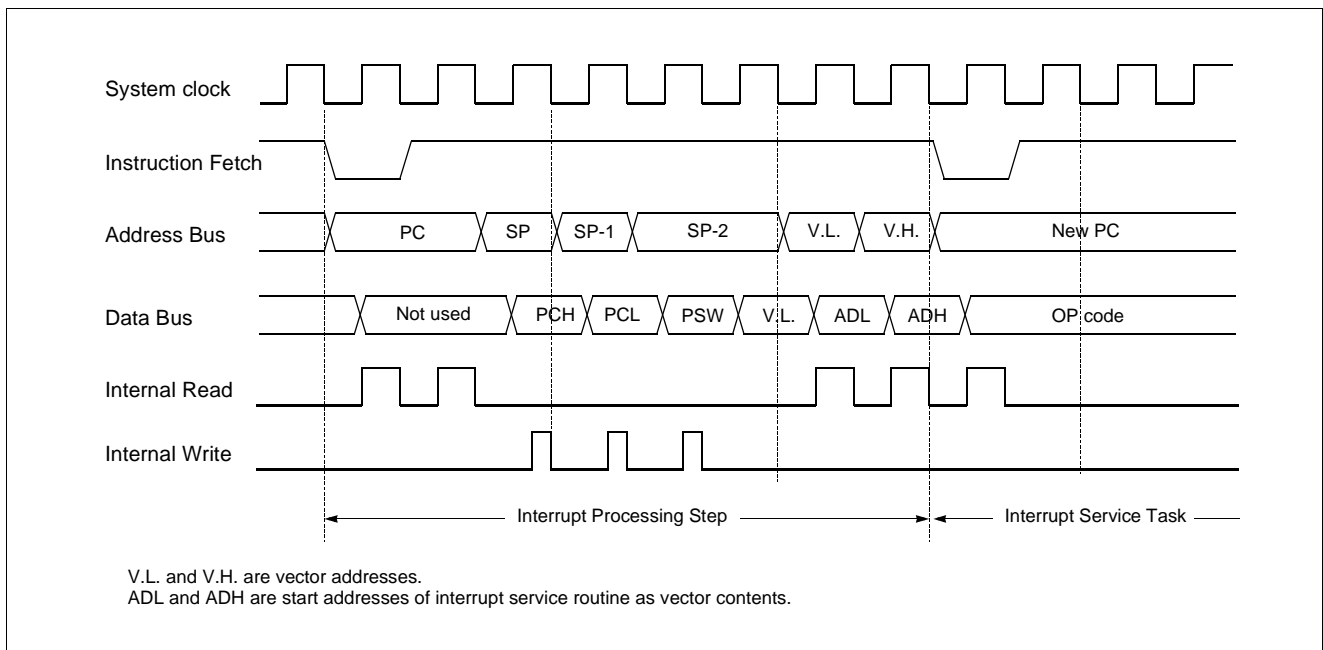
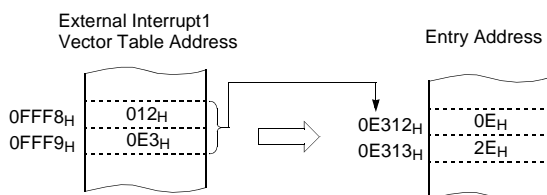


Figure 14-4 Timing chart of Interrupt Acceptance and Interrupt Return Instruction



Correspondence between vector table address for External Interrupt1 and the entry address of the interrupt service program.

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

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

Saving/Restoring General-purpose Register

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

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

registers.

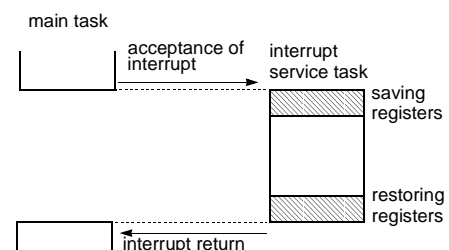
Example: Register save using push and pop instructions

```
INTxx:  PUSH    A      ;SAVE ACC.
        PUSH    X      ;SAVE X REG.
        PUSH    Y      ;SAVE Y REG.
```

interrupt processing

```
POP     Y      ;RESTORE Y REG.
POP     X      ;RESTORE X REG.
POP     A      ;RESTORE ACC.
RETI
```

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



14.5 BRK Interrupt

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

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

Each processing step is determined by B-flag as shown in Figure 14-5

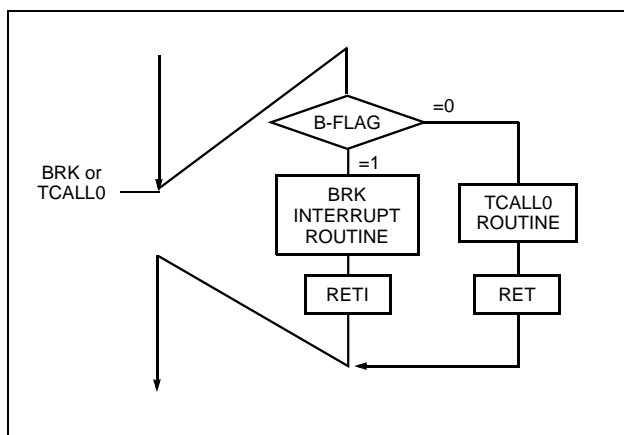


Figure 14-5 Execution of BRK/TCALL0

14.6 Multi Interrupt

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

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

Example: During Timer1 interrupt is in progress, INT1 interrupt serviced without any suspend.

```

TIMER1:  PUSH  A
          PUSH  X
          PUSH  Y
          LDM   IENH,#40H ; Enable INT1 only
          LDM   IENL,#00H ; Disable other
          EI      ; Enable Interrupt
          :
          :
          :
          LDM   IENH,#0FFH ; Enable all interrupts
          LDM   IENL,#0FFH
          POP   Y
          POP   X
          POP   A
          RETI
  
```

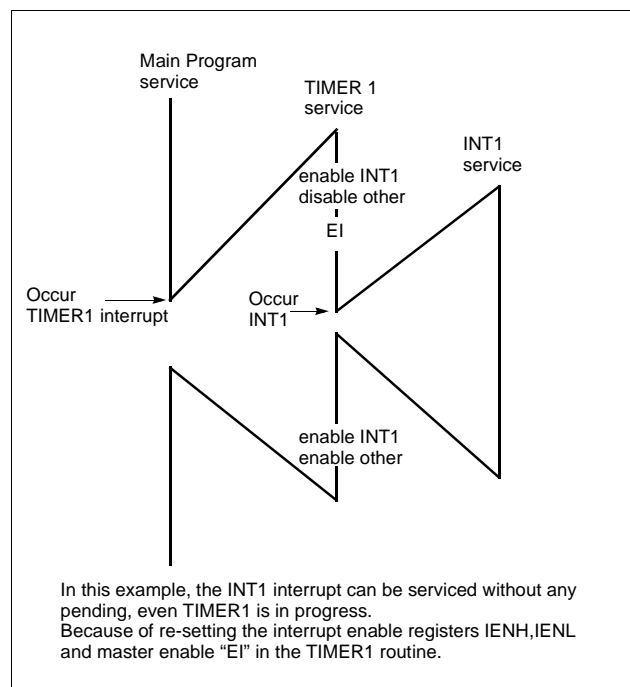


Figure 14-6 Execution of Multi Interrupt

14.7 External Interrupt

The external interrupt on INT1 and INT2 pins are edge triggered depending on the edge selection register IEDS (address 0D8H) as

shown in Figure14-7.

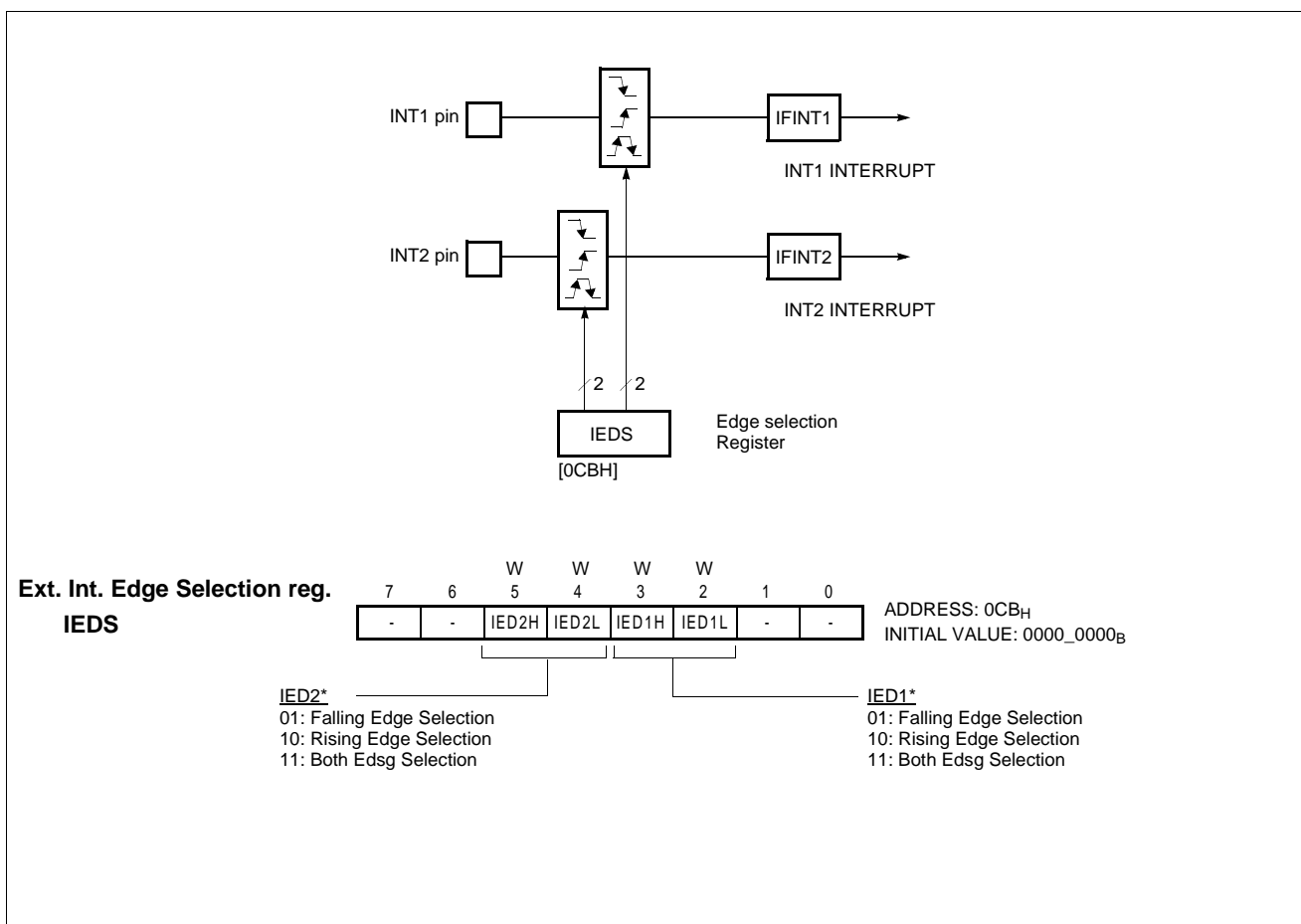


Figure 14-7 External Interrupt Block Diagram

Response Time

The INT1 ~ INT2 edge are latched into IFINT1 ~ IFINT2 at every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The DIV itself takes twelve cycles. Thus, a minimum of twelve complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine.

Figure 14-8 shows interrupt response timings.

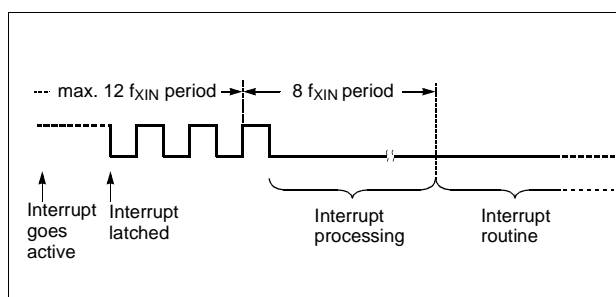


Figure 14-8 Interrupt Response Timing Diagram

14.8 Key Scan Input Processing

Key Scan Interrupt is generated by detecting low or high Input from each Input pin (R0, R1) is one of the sources which release standby (SLEEP, STOP) mode. Key Scan ports are all 16bit which are controlled by Standby Mode Release Register (SMRR0, SMRR1). Key Input is considered as Interrupt, therefore, KSCNE bit of IEHN should be

set for correct interrupt executing, SLEEP mode and STOP mode, the rest of executing is the same as that of external Interrupt. Each SMRR Register bit is allowed for each port (for Bit= "0", no Key Input, for Bit= "1", Key Input available). At reset, SMRR becomes "00h". So, there is no Key Input source.

Standby release level control register (SRLC) can select the key scan input level “L” or “H” for standby release by each bit pin (R0, R1). Standby release level control register

(SRLC) is write-only register and initialized as “00h” in reset state.

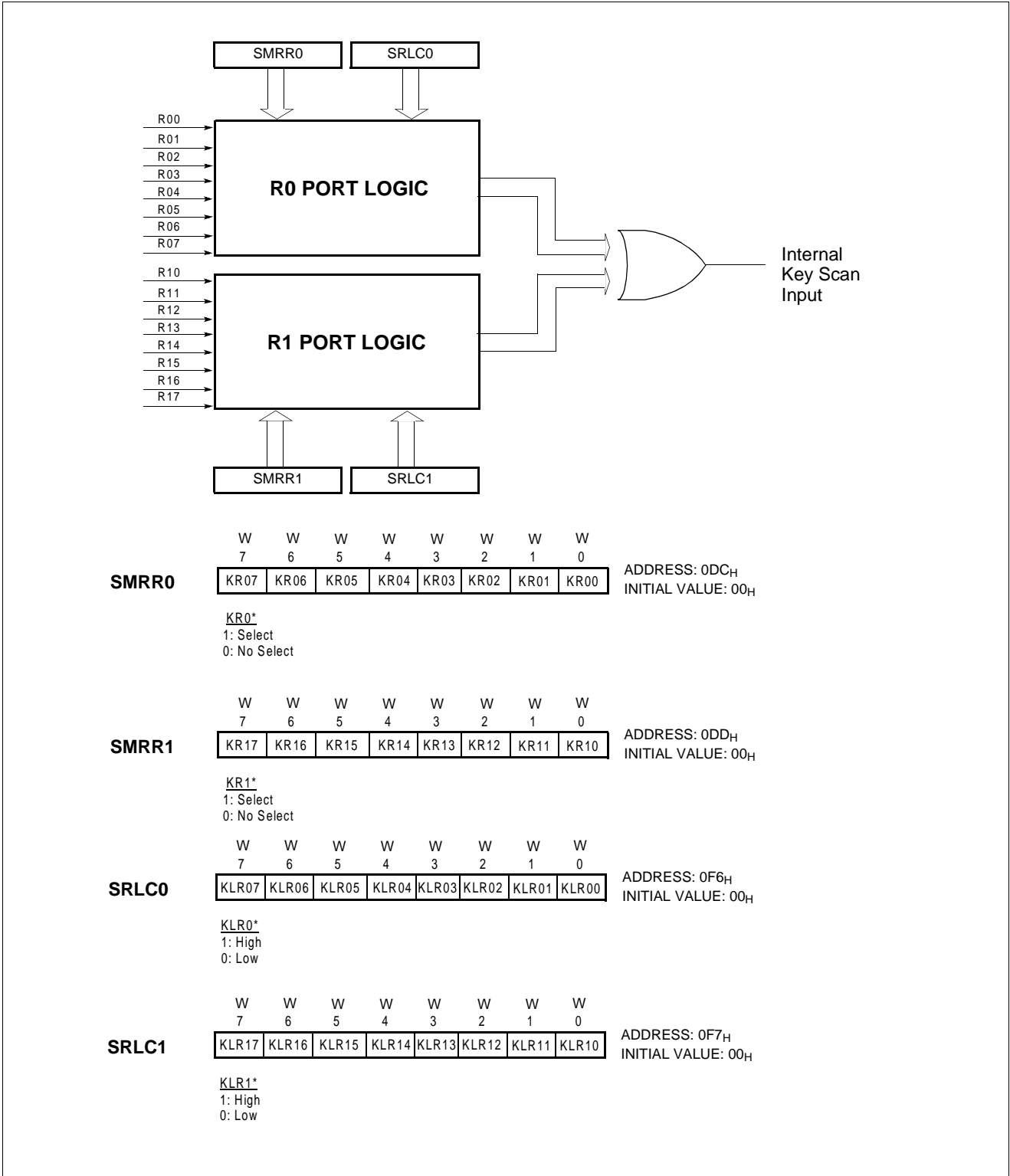


Figure 14-9 Block Diagram of Key Scan Block

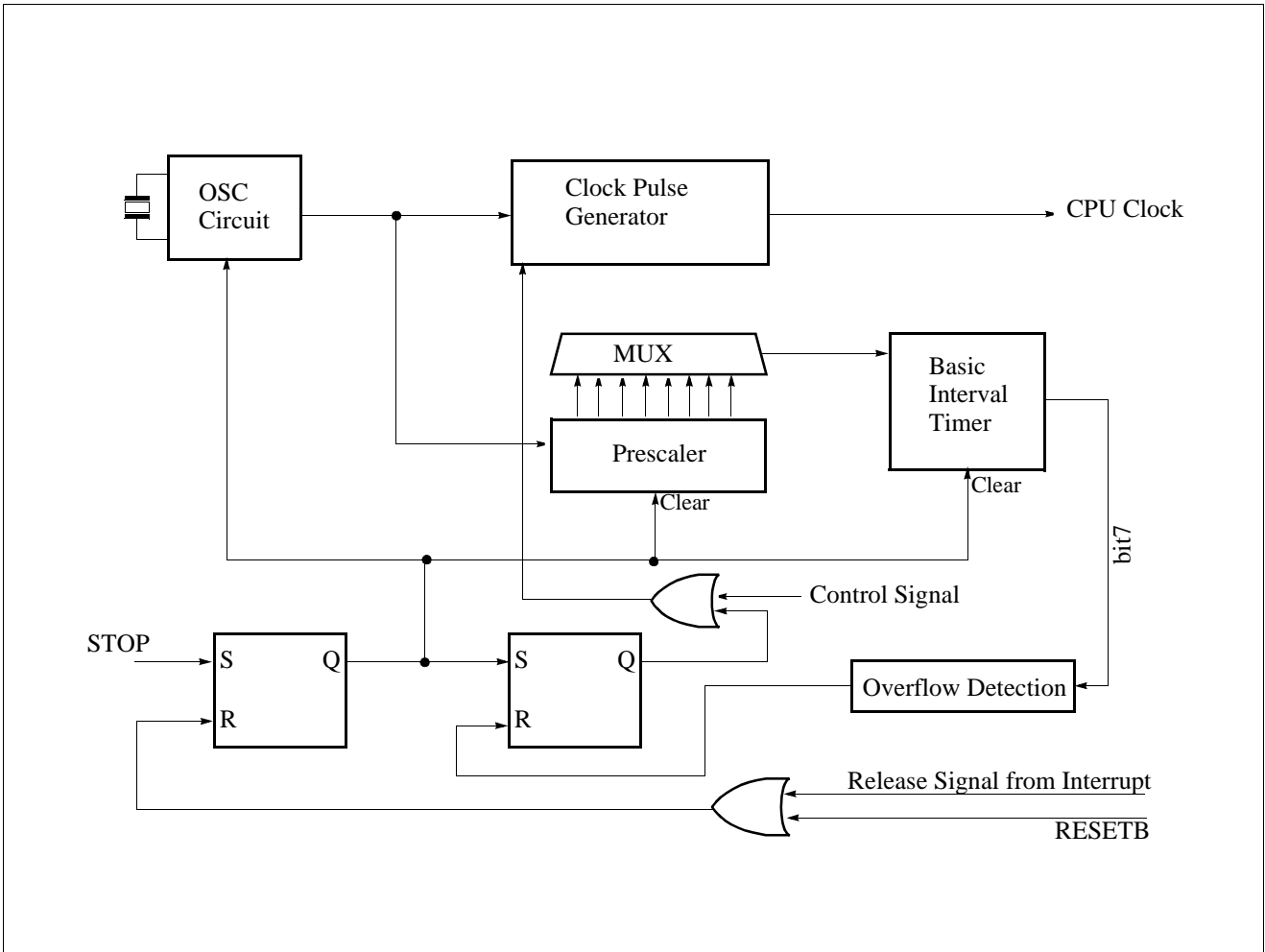


Figure 15-1 Block Diagram of Standby Circuit

15.3 STANDBY MODE RELEASE

Release of STANDBY mode is executed by RESET input and Interrupt signal. Register value is defined when Reset. When there is a release signal of STOP mode (Interrupt, RESET input), the instruction execution starts after stabilization oscillation time is set by value of BTS2 ~ BTS0 and set ENPCK to "1".

| Release Signal | SLEEP | STOP |
|-----------------|-------|------|
| RESETB | O | O |
| KSCN(Key Input) | O | O |
| INT1,INT2 | O | O |
| B.I.T. | O | |

Table 15-1 Release Signal of Standby Mode

| Release Factor | Release Method |
|-----------------------------|---|
| RESETB | By RESETB Pin=Low level, Standby mode is released and system is initialized |
| KSCN(Key Input) | Standby mode is released by low input of selected pin by key scan Input(SMRR0,SMRR1). In case of interrupt mask enable flag= "0", program executes just after standby instruction, if flag= "1" enters each interrupt service routine. |
| INT1,INT2 | When external interrupt (INT1,INT2) enable flag is "1", standby mode is released at the rising edge of each terminal. When standby mode is released at interrupt. Mask Enable flag= "0", program executes from the next instruction of standby instruction. When "1", enters each interrupt service routine. |
| Basic Interval Timer(IFBIT) | When B.I.T. is executed only by bit10 of prescaler(PS10), SLEEP mode can be released. Interrupt release SLEEP mode, when BIT interrupt enable flag is "1". When standby mode is released at interrupt. Mask enable flag= "0", program executes from the next instruction of SLEEP instruction. When "1", enters each interrupt service routine. |

Table 15-2

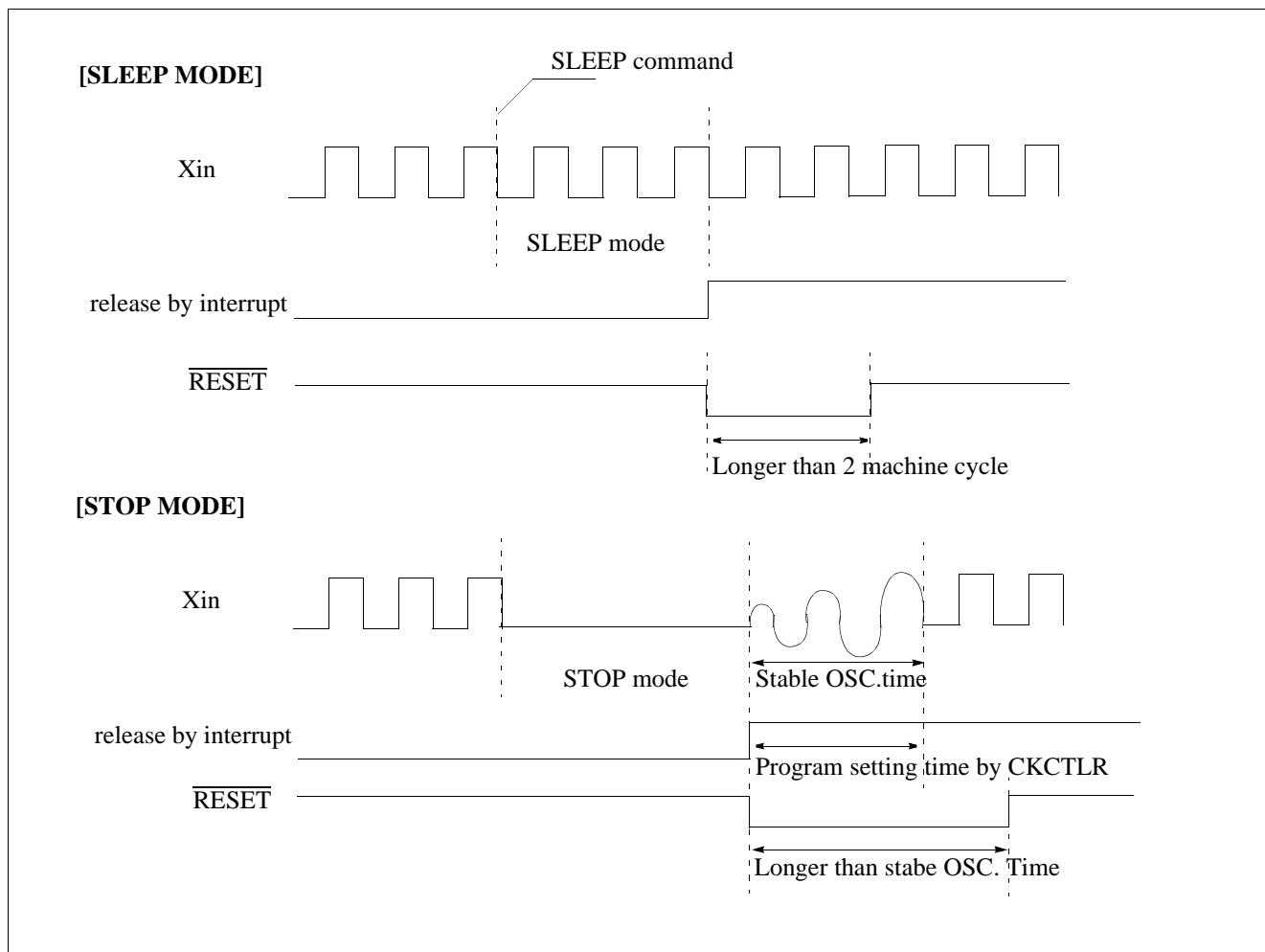


Figure 15-2 Block Diagram of Standby Circuit

15.4 RELEASE OPERATION OF STANDBY MODE

After standby mode is released, the operation begins according to content of related interrupt register just before

standby mode start (Figure 15-3).

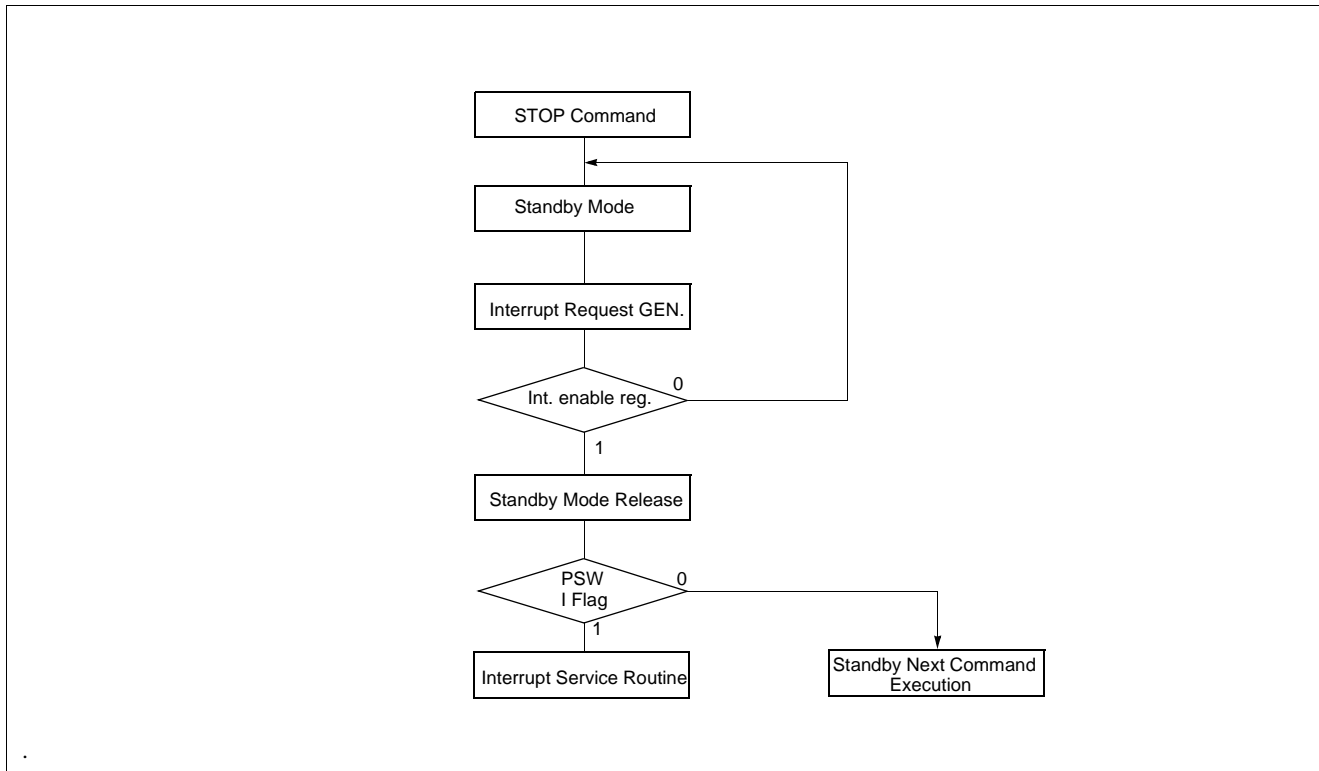


Figure 15-3 Standby Mode Release Flow

(1) Interrupt Enable Flag(I) of PSW = "0"

Release by only interrupt which interrupt enable flag = "1", and starts to execute from next to standby instruction (SLEEP or STOP).

(2) Interrupt Enable Flag(I) of PSW = "1"

Released by only interrupt which each interrupt enable flag = "1", and jump to the relevant interrupt service routine.

Note: When STOP instruction is used, B.I.T should guarantee the stabilization oscillation time. Thus, just before en-

tering STOP mode, clock of bit10 (PS10) of prescaler is selected or peripheral hardware clock control bit (ENPCK) to "1", Therefore the clock necessary for stabilization oscillation time should be input into B.I.T. otherwise, standby mode is released by reset signal. In case of interrupt request flag and interrupt enable flag are both "1", standby mode is not entered.

| Internal circuit | SLEEP mode | STOP mode |
|----------------------|--------------------------------------|-----------|
| Oscillator | Active | Stop |
| Internal CPU | Stop | Stop |
| Register | Retained | Retained |
| RAM | Retained | Retained |
| I/O port | Retained | Retained |
| Prescaler | Active | Retained |
| Basic Interval Timer | PS10 selected:Active Others: Stop | Stop |
| Watch-dog Timer | Stop | Stop |
| Timer | Stop | Stop |
| Address Bus,Data Bus | Retained | Retained |

Table 15-3 Operation State in Standby Mode

16. RESET FUNCTION

16.1 EXTERNAL RESET

The $\overline{\text{RESET}}$ pin should be held at low for at least 2 machine cycles with the power supply voltage within the operating voltage range and must be connected 0.1 μF capacitor for

stable system initialization. The RESET pin contains a Schmitt trigger with an internal pull-up resistor.

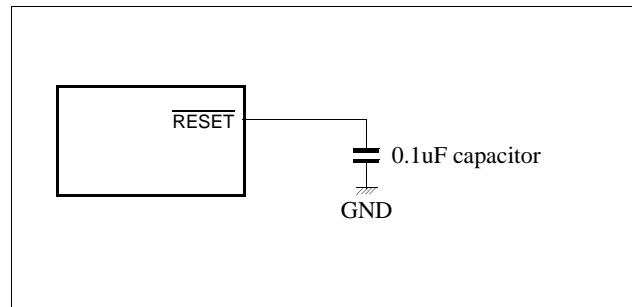


Figure 16-1 $\overline{\text{RESET}}$ Pin connection

16.2 POWER ON RESET

Power On Reset circuit automatically detects the rise of power voltage (the rising time should be within 50ms) the power voltage reaches a certain level, RESET terminal is maintained at "L" Level until a crystal ceramic oscillator oscillates stably. After power applies and starting of oscillation, this reset state is maintained for about oscillation cycle of 219 (about 65.5ms : at 4MHz). The execution of built-in Power On Reset circuit is as follows :

(1) Latch the pulse from Power On Detection Pulse Generator circuit, and reset Prescaler, B.I.T and B.I.T Overflow

detection circuit.

(2) Once B.I.T Overflow detection circuit is reset. Then, Prescaler starts to count.

(3) Prescaler output is inputted into B.I.T and PS10 of Prescaler output is automatically selected. If overflow of B.I.T is detected, Overflow detection circuit is set.

(4) Reset circuit generates maximum period of reset pulse from Prescaler and B.I.T

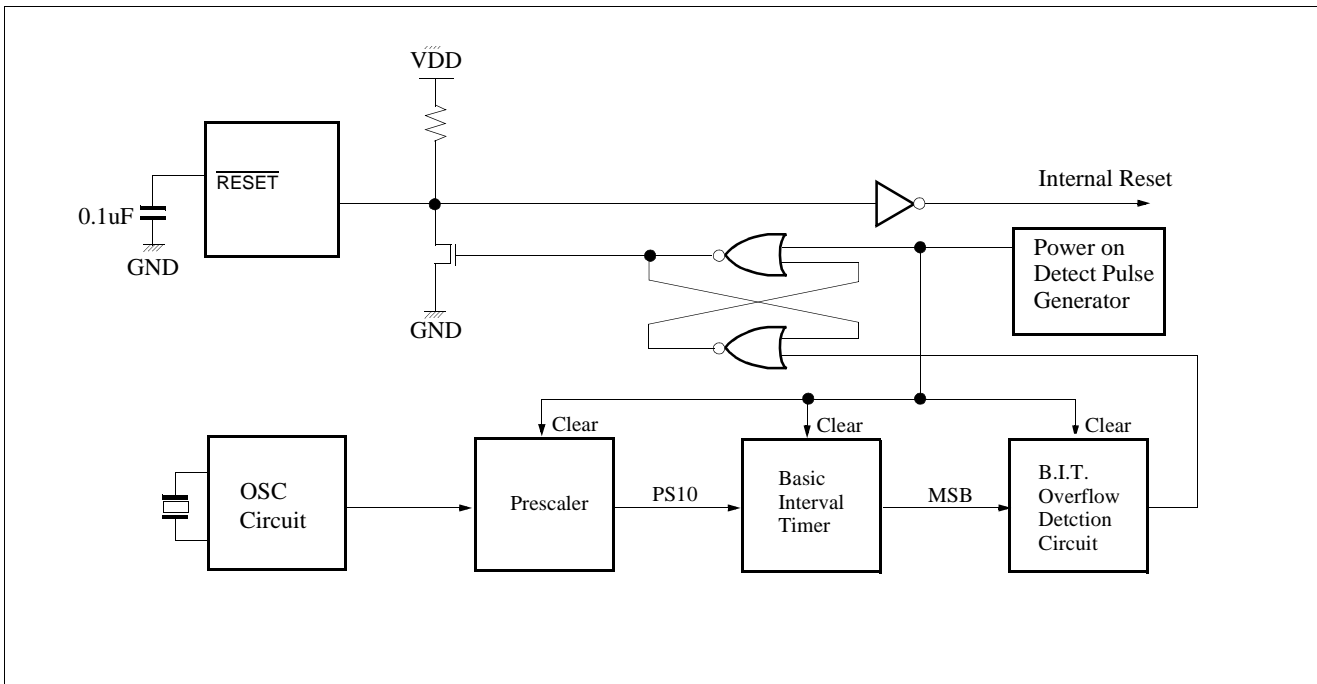


Figure 16-2 Block Diagram of Power On Reset Circuit

Note: When Power On Reset, oscillator stabilization time doesn't include OSC. Start time.

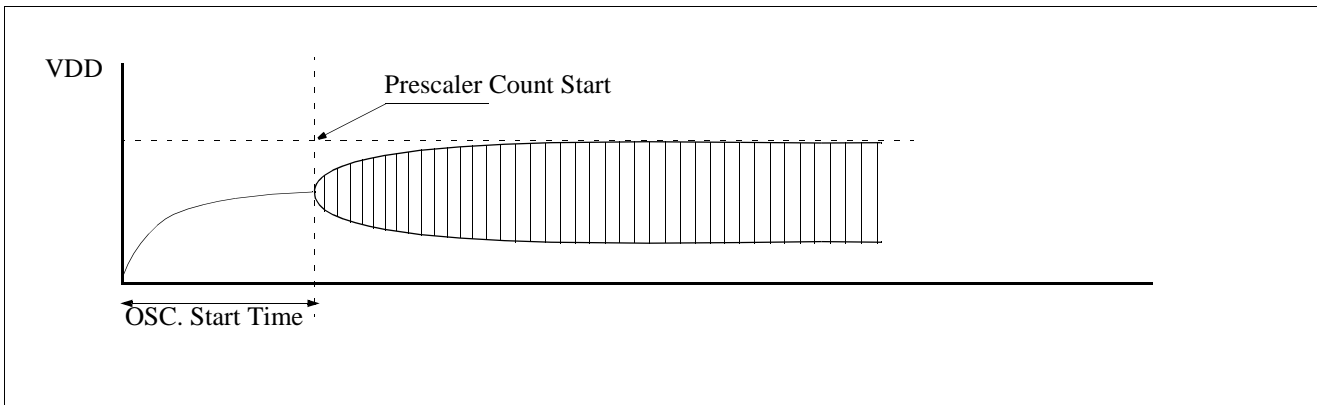


Figure 16-3 Oscillator stabiliaztion diagram

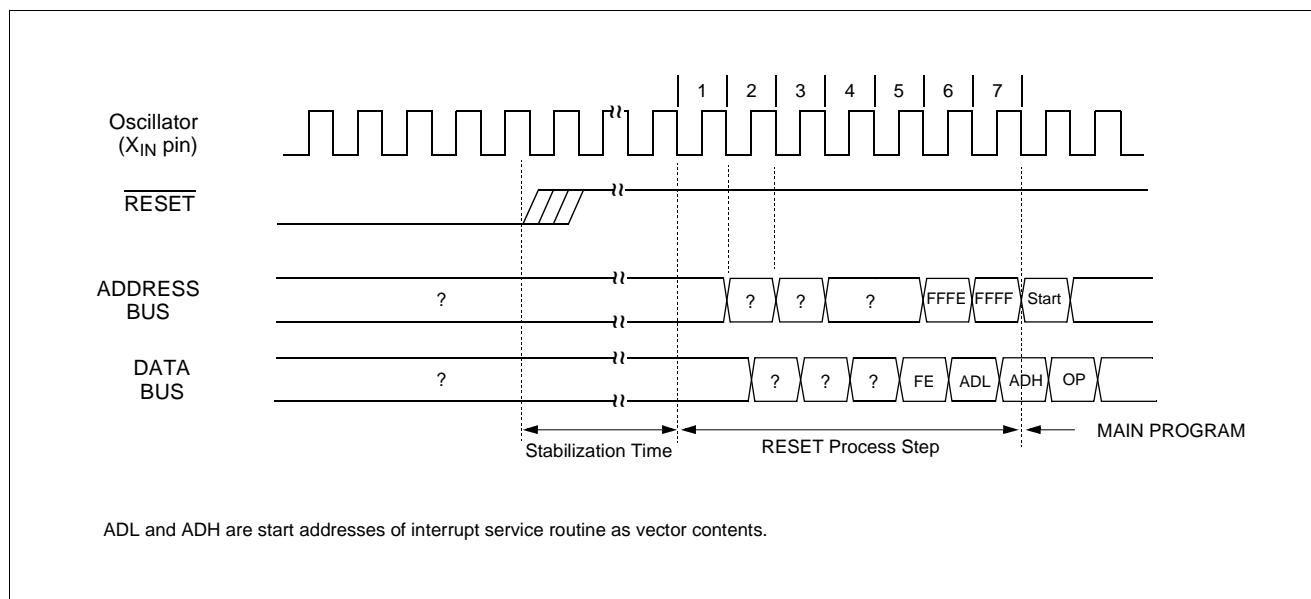


Figure 16-4 Timing Diagram of Reset

16.3 Low Voltage Detection Mode

(1) Low voltage detection condition

An on board voltage comparator checks that VDD is at the required level to ensure correct operation of the device. If VDD is below a certain level, Low voltage detector forces the device into low voltage detection mode.

(2) Low Voltage Detection Mode

There is no power consumption except stop current, stop mode release function is disabled. All I/O port is configured as input mode and Data memory is retained until voltage through external capacitor is worn out. In this mode,

all port can be selected with Pull-up resistor by Mask option. If there is no information on the Mask option sheet, the default pull up option (all port connect to pull-up resistor) is selected.

(3) Release of Low Voltage Detection Mode

Reset signal result from new battery (normally 3V) wakes the low voltage detection mode and come into normal reset state. It depends on user whether to execute RAM clear routine or not

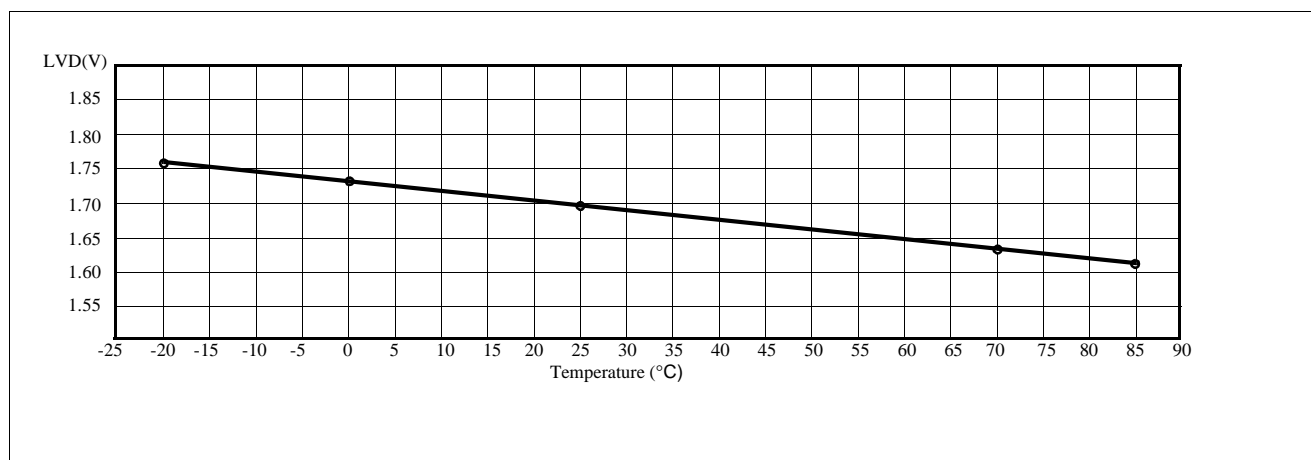


Figure 16-5 Low Voltage vs Temperature

(4) SRAM BACK-UP after Low Voltage Detection.

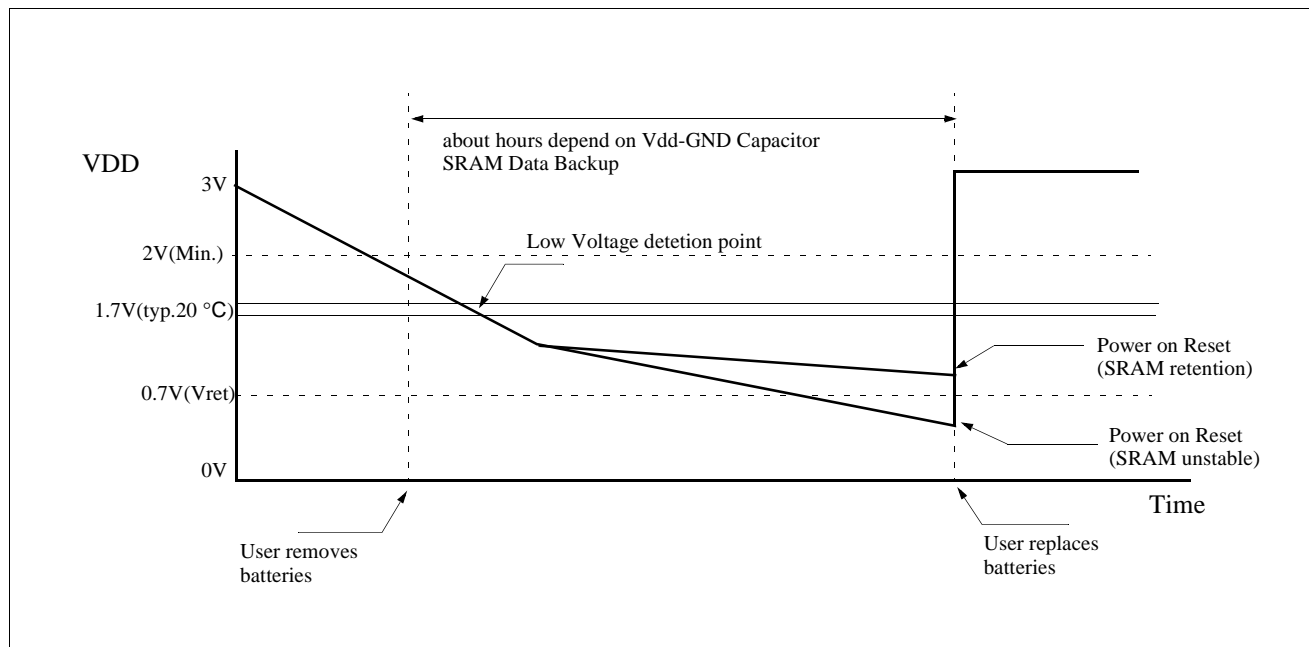


Figure 16-6 Oscillator stabiliazion diagram

| | |
|-------------------------|----------------------|
| Interrupt | disable |
| Stop release | disable |
| All I/O port | input Mode |
| Remout port | Low Level |
| OSC | STOP |
| All I/O port pull-up on | Mask Option |
| SRAM Data | retention until Vret |

Table 16-1 The operation after Low Voltage detection

(5) S/W flow chart example after Reset using SRAM Back-up

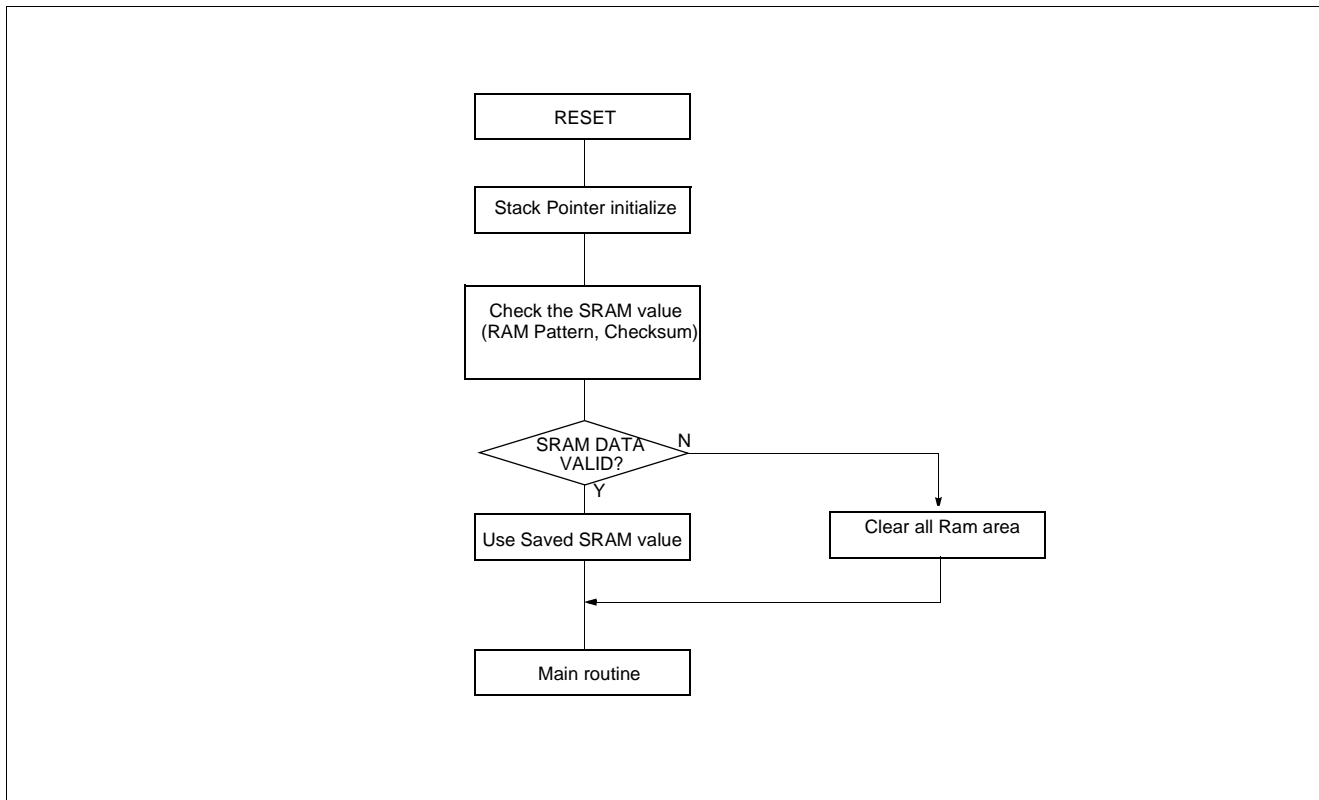


Figure 16-7 S/W flow chart example after Reset using SRAM Back-up

APPENDIX

A. MASK ORDER SHEET

MASK ORDER & VERIFICATION SHEET

HMS810 **E -UE**

Customer should write inside thick line box.

1. Customer Information

| | |
|--|--|
| Company Name | |
| Application | |
| Order Date | <div> <div>YYYY</div> <div>MM</div> <div>DD</div> </div> |
| <div> <div>Tel:</div> <div>Fax:</div> </div> | |
| Name&Signature: | |

2. Device Information

| | |
|------------------------|--|
| Package | <input type="checkbox"/> 20SOP <input type="checkbox"/> 20PDIP <input type="checkbox"/> 24SOP <input type="checkbox"/> 24SKDIP <input type="checkbox"/> 28SOP <input type="checkbox"/> 28SKDIP <input type="checkbox"/> 28PIN DIE |
| Mask Data Check Sum | File Name: (.OTP) (@27c256) |

3. Inclusion of pull-up resistor in Low Volatage Detection mode

[illegible]

*1 : is not available for 20PIN. So default option is pull-up on.

*2 : is not available for 20PIN & 24PIN. So default option is pull-up on.

4. Marking Specification

(Please check mark into ☐)

04/08/16/24/32

hynix

HMS810 **E -UE**

YYWW **KOREA**

Hynix ROM Code Number

Lot Number

Customer's logo

Customer logo is not required

If the customer logo must be used in the special mark, please submit a clean original of the logo.

Customer's part number

[illegible]

5. Delivery Schedule

| | | | |
|-----------------|-------------------|----------|--------------------|
| | Date | Quantity | Hynix Confirmation |
| Customer Sample | YYYY MM DD • • | pcs | |
| Risk Order | YYYY MM DD • • | pcs | |

6. ROM Code Verification

| | | | |
|---|------|------|----|
| Verification Date: | YYYY | MM | DD |
| <div style="text-align: center;">• •</div> | | | |
| <i>Please confirm our verification data.</i> | | | |
| Check Sum: | | | |
| Tel: | | Fax: | |
| Name & | | | |
| Signature: | | | |

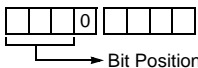
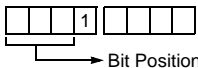
This box is written after “6.Verification”.

| | | | |
|--|------|----|----|
| Approval Date: | YYYY | MM | DD |
| <i>I agree with your verification data and confirm you to make mask set.</i> | | | |
| Tel: | Fax: | | |
| Name & Signature: | | | |

hynix

B. INSTRUCTION

B.1 Terminology List

| Terminology | Description |
|-------------|---|
| A | Accumulator |
| X | X - register |
| Y | Y - register |
| PSW | Program Status Word |
| #imm | 8-bit Immediate data |
| dp | Direct Page Offset Address |
| !abs | Absolute Address |
| [] | Indirect expression |
| { } | Register Indirect expression |
| { }+ | Register Indirect expression, after that, Register auto-increment |
| .bit | Bit Position |
| A.bit | Bit Position of Accumulator |
| dp.bit | Bit Position of Direct Page Memory |
| M.bit | Bit Position of Memory Data (000 _H ~0FFF _H) |
| rel | Relative Addressing Data |
| upage | U-page (0FF00 _H ~0FFFF _H) Offset Address |
| n | Table CALL Number (0~15) |
| + | Addition |
| x |  Upper Nibble Expression in Opcode |
| y |  Upper Nibble Expression in Opcode |
| – | Subtraction |
| × | Multiplication |
| / | Division |
| () | Contents Expression |
| ^ | AND |
| ∨ | OR |
| ⊕ | Exclusive OR |
| ~ | NOT |
| ← | Assignment / Transfer / Shift Left |
| → | Shift Right |
| ↔ | Exchange |
| = | Equal |
| ≠ | Not Equal |

B.2 Instruction Map

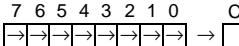
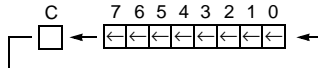
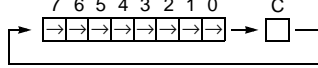
| LOW HIGH | 0000 00 | 00001 01 | 00010 02 | 00011 03 | 00100 04 | 00101 05 | 00110 06 | 00111 07 | 01000 08 | 01001 09 | 01010 0A | 01011 0B | 01100 0C | 01101 0D | 01110 0E | 01111 0F |
|-------------|------------|----------------|------------------|-------------------|----------------|-------------|-------------|-------------|-------------|-------------|-----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

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

B.3 Instruction Set

Arithmetic / Logic Operation

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

| No. | Mnemonic | 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 $A \leftarrow (A) \oplus (M)$ | N-----Z- |
| 46 | EOR dp | A5 | 2 | 3 | | |
| 47 | EOR dp + X | A6 | 2 | 4 | | |
| 48 | EOR !abs | A7 | 3 | 4 | | |
| 49 | EOR !abs + Y | B5 | 3 | 5 | | |
| 50 | EOR [dp + X] | B6 | 2 | 6 | | |
| 51 | EOR [dp] + Y | B7 | 2 | 6 | | |
| 52 | EOR { X } | B4 | 1 | 3 | | |
| 53 | INC A | 88 | 1 | 2 | Increment $M \leftarrow (M) + 1$ | N-----ZC |
| 54 | INC dp | 89 | 2 | 4 | | N-----Z- |
| 55 | INC dp + X | 99 | 2 | 5 | | N-----Z- |
| 56 | INC !abs | 98 | 3 | 5 | | N-----Z- |
| 57 | INC X | 8F | 1 | 2 | | N-----Z- |
| 58 | INC Y | 9E | 1 | 2 | | N-----Z- |
| 59 | LSR A | 48 | 1 | 2 | Logical shift right "0" \rightarrow  \rightarrow C | N-----ZC |
| 60 | LSR dp | 49 | 2 | 4 | | |
| 61 | LSR dp + X | 59 | 2 | 5 | | |
| 62 | LSR !abs | 58 | 3 | 5 | | |
| 63 | MUL | 5B | 1 | 9 | Multiply : $YA \leftarrow Y \times A$ | N-----Z- |
| 64 | OR #imm | 64 | 2 | 2 | Logical OR $A \leftarrow (A) \vee (M)$ | N-----Z- |
| 65 | OR dp | 65 | 2 | 3 | | |
| 66 | OR dp + X | 66 | 2 | 4 | | |
| 67 | OR !abs | 67 | 3 | 4 | | |
| 68 | OR !abs + Y | 75 | 3 | 5 | | |
| 69 | OR [dp + X] | 76 | 2 | 6 | | |
| 70 | OR [dp] + Y | 77 | 2 | 6 | | |
| 71 | OR { X } | 74 | 1 | 3 | | |
| 72 | ROL A | 28 | 1 | 2 | Rotate left through Carry  | N-----ZC |
| 73 | ROL dp | 29 | 2 | 4 | | |
| 74 | ROL dp + X | 39 | 2 | 5 | | |
| 75 | ROL !abs | 38 | 3 | 5 | Rotate right through Carry  | N-----ZC |
| 76 | ROR A | 68 | 1 | 2 | | |
| 77 | ROR dp | 69 | 2 | 4 | | |
| 78 | ROR dp + X | 79 | 2 | 5 | | |
| 79 | ROR !abs | 78 | 3 | 5 | Subtract with Carry $A \leftarrow (A) - (M) - \sim(C)$ | NV--HZC |
| 80 | SBC #imm | 24 | 2 | 2 | | |
| 81 | SBC dp | 25 | 2 | 3 | | |
| 82 | SBC dp + X | 26 | 2 | 4 | | |
| 83 | SBC !abs | 27 | 3 | 4 | | |
| 84 | SBC !abs + Y | 35 | 3 | 5 | | |
| 85 | SBC [dp + X] | 36 | 2 | 6 | | |
| 86 | SBC [dp] + Y | 37 | 2 | 6 | | |
| 87 | SBC { X } | 34 | 1 | 3 | | |
| 88 | TST dp | 4C | 2 | 3 | Test memory contents for negative or zero, (dp) - 00H | N-----Z- |
| 89 | XCN | CE | 1 | 5 | Exchange nibbles within the accumulator $A_7 \sim A_4 \leftrightarrow A_3 \sim A_0$ | N-----Z- |

Register / Memory Operation

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

16-BIT operation

| No. | Mnemonic | Op Code | Byte No | Cycle No | Operation | Flag NVGBHIZC |
|-----|----------|---------|---------|----------|--|------------------|
| 1 | ADDW dp | 1D | 2 | 5 | 16-Bits add without Carry $YA \leftarrow (YA) (dp+1) (dp)$ | NV--H-ZC |
| 2 | CMPW dp | 5D | 2 | 4 | Compare YA contents with memory pair contents : $(YA) - (dp+1)(dp)$ | N-----ZC |
| 3 | DECW dp | BD | 2 | 6 | Decrement memory pair $(dp+1)(dp) \leftarrow (dp+1) (dp) - 1$ | N-----Z- |
| 4 | INCW dp | 9D | 2 | 6 | Increment memory pair $(dp+1) (dp) \leftarrow (dp+1) (dp) + 1$ | N-----Z- |
| 5 | LDYA dp | 7D | 2 | 5 | Load YA $YA \leftarrow (dp+1) (dp)$ | N-----Z- |
| 6 | STYA dp | DD | 2 | 5 | Store YA $(dp+1) (dp) \leftarrow YA$ | ----- |
| 7 | SUBW dp | 3D | 2 | 5 | 16-Bits subtract without carry $YA \leftarrow (YA) - (dp+1) (dp)$ | NV--H-ZC |

Bit Manipulation

| No. | Mnemonic | Op Code | Byte No | Cycle No | Operation | Flag NVGBHIZC |
|-----|-------------|---------|---------|----------|--|------------------|
| 1 | AND1 M.bit | 8B | 3 | 4 | Bit AND C-flag : $C \leftarrow (C) \wedge (M.bit)$ | -----C |
| 2 | AND1B M.bit | 8B | 3 | 4 | Bit AND C-flag and NOT : $C \leftarrow (C) \wedge \sim(M.bit)$ | -----C |
| 3 | BIT dp | 0C | 2 | 4 | Bit test A with memory : | MM-----Z- |
| 4 | BIT !abs | 1C | 3 | 5 | $Z \leftarrow (A) \wedge (M), N \leftarrow (M_7), V \leftarrow (M_6)$ | |
| 5 | CLR1 dp.bit | y1 | 2 | 4 | Clear bit : $(M.bit) \leftarrow "0"$ | ----- |
| 6 | CLRA1 A.bit | 2B | 2 | 2 | Clear A bit : $(A.bit) \leftarrow "0"$ | ----- |
| 7 | CLRC | 20 | 1 | 2 | Clear C-flag : $C \leftarrow "0"$ | -----0 |
| 8 | CLRG | 40 | 1 | 2 | Clear G-flag : $G \leftarrow "0"$ | --0----- |
| 9 | CLRV | 80 | 1 | 2 | Clear V-flag : $V \leftarrow "0"$ | -0--0--- |
| 10 | EOR1 M.bit | AB | 3 | 5 | Bit exclusive-OR C-flag : $C \leftarrow (C) \oplus (M.bit)$ | -----C |
| 11 | EOR1B M.bit | AB | 3 | 5 | Bit exclusive-OR C-flag and NOT : $C \leftarrow (C) \oplus \sim(M.bit)$ | -----C |
| 12 | LDC M.bit | CB | 3 | 4 | Load C-flag : $C \leftarrow (M.bit)$ | -----C |
| 13 | LDCB M.bit | CB | 3 | 4 | Load C-flag with NOT : $C \leftarrow \sim(M.bit)$ | -----C |
| 14 | NOT1 M.bit | 4B | 3 | 5 | Bit complement : $(M.bit) \leftarrow \sim(M.bit)$ | ----- |
| 15 | OR1 M.bit | 6B | 3 | 5 | Bit OR C-flag : $C \leftarrow (C) \vee (M.bit)$ | -----C |
| 16 | OR1B M.bit | 6B | 3 | 5 | Bit OR C-flag and NOT : $C \leftarrow (C) \vee \sim(M.bit)$ | -----C |
| 17 | SET1 dp.bit | x1 | 2 | 4 | Set bit : $(M.bit) \leftarrow "1"$ | ----- |
| 18 | SETA1 A.bit | 0B | 2 | 2 | Set A bit : $(A.bit) \leftarrow "1"$ | ----- |
| 19 | SETC | A0 | 1 | 2 | Set C-flag : $C \leftarrow "1"$ | -----1 |
| 20 | SETG | C0 | 1 | 2 | Set G-flag : $G \leftarrow "1"$ | --1----- |
| 21 | STC M.bit | EB | 3 | 6 | Store C-flag : $(M.bit) \leftarrow C$ | ----- |
| 22 | TCLR1 !abs | 5C | 3 | 6 | Test and clear bits with A : $A - (M), (M) \leftarrow (M) \wedge \sim(A)$ | N-----Z- |
| 23 | TSET1 !abs | 3C | 3 | 6 | Test and set bits with A : $A - (M), (M) \leftarrow (M) \vee (A)$ | N-----Z- |

Branch / Jump Operation

| No. | Mnemonic | Op Code | Byte No | Cycle No | Operation | Flag NVGBHIZC |
|-----|----------------|---------|---------|----------|---|------------------|
| 1 | BBC A.bit,rel | y2 | 2 | 4/6 | Branch if bit clear : | ----- |
| 2 | BBC dp.bit,rel | y3 | 3 | 5/7 | if (bit) = 0 , then $pc \leftarrow (pc) + rel$ | |
| 3 | BBS A.bit,rel | x2 | 2 | 4/6 | Branch if bit set : | ----- |
| 4 | BBS dp.bit,rel | x3 | 3 | 5/7 | if (bit) = 1 , then $pc \leftarrow (pc) + rel$ | |
| 5 | BCC rel | 50 | 2 | 2/4 | Branch if carry bit clear if (C) = 0 , then $pc \leftarrow (pc) + rel$ | ----- |
| 6 | BCS rel | D0 | 2 | 2/4 | Branch if carry bit set if (C) = 1 , then $pc \leftarrow (pc) + rel$ | ----- |
| 7 | BEQ rel | F0 | 2 | 2/4 | Branch if equal if (Z) = 1 , then $pc \leftarrow (pc) + rel$ | ----- |
| 8 | BMI rel | 90 | 2 | 2/4 | Branch if minus if (N) = 1 , then $pc \leftarrow (pc) + rel$ | ----- |
| 9 | BNE rel | 70 | 2 | 2/4 | Branch if not equal if (Z) = 0 , then $pc \leftarrow (pc) + rel$ | ----- |
| 10 | BPL rel | 10 | 2 | 2/4 | Branch if plus if (N) = 0 , then $pc \leftarrow (pc) + rel$ | ----- |
| 11 | BRA rel | 2F | 2 | 4 | Branch always $pc \leftarrow (pc) + rel$ | ----- |
| 12 | BVC rel | 30 | 2 | 2/4 | Branch if overflow bit clear if (V) = 0 , then $pc \leftarrow (pc) + rel$ | ----- |
| 13 | BVS rel | B0 | 2 | 2/4 | Branch if overflow bit set if (V) = 1 , then $pc \leftarrow (pc) + rel$ | ----- |
| 14 | CALL !abs | 3B | 3 | 8 | Subroutine call | |
| 15 | CALL [dp] | 5F | 2 | 8 | $M(sp) \leftarrow (pc_H)$, $sp \leftarrow sp - 1$, $M(sp) \leftarrow (pc_L)$, $sp \leftarrow sp - 1$, if !abs, $pc \leftarrow abs$; if [dp], $pc_L \leftarrow (dp)$, $pc_H \leftarrow (dp+1)$. | ----- |
| 16 | CBNE dp,rel | FD | 3 | 5/7 | Compare and branch if not equal : | ----- |
| 17 | CBNE dp+X,rel | 8D | 3 | 6/8 | if (A) \neq (M) , then $pc \leftarrow (pc) + rel$. | |
| 18 | DBNE dp,rel | AC | 3 | 5/7 | Decrement and branch if not equal : | ----- |
| 19 | DBNE Y,rel | 7B | 2 | 4/6 | if (M) \neq 0 , then $pc \leftarrow (pc) + rel$. | |
| 20 | JMP !abs | 1B | 3 | 3 | Unconditional jump | |
| 21 | JMP [!abs] | 1F | 3 | 5 | $pc \leftarrow$ jump address | ----- |
| 22 | JMP [dp] | 3F | 2 | 4 | | |
| 23 | PCALL upage | 4F | 2 | 6 | U-page call $M(sp) \leftarrow (pc_H)$, $sp \leftarrow sp - 1$, $M(sp) \leftarrow (pc_L)$, $sp \leftarrow sp - 1$, $pc_L \leftarrow (upage)$, $pc_H \leftarrow "0FF_H"$. | ----- |
| 24 | TCALL n | nA | 1 | 8 | Table call : $(sp) \leftarrow (pc_H)$, $sp \leftarrow sp - 1$, $M(sp) \leftarrow (pc_L)$, $sp \leftarrow sp - 1$, $pc_L \leftarrow (Table \text{ vector } L)$, $pc_H \leftarrow (Table \text{ vector } H)$ | ----- |

Control Operation & Etc.

| No. | Mnemonic | Op Code | Byte No | Cycle No | Operation | Flag NVGBHIZC |
|-----|----------|---------|---------|----------|--|------------------|
| 1 | BRK | 0F | 1 | 8 | Software interrupt : $B \leftarrow "1"$, $M(sp) \leftarrow (pc_H)$, $sp \leftarrow sp-1$, $M(s) \leftarrow (pc_L)$, $sp \leftarrow sp - 1$, $M(sp) \leftarrow (PSW)$, $sp \leftarrow sp - 1$, $pc_L \leftarrow (0FFDE_H)$, $pc_H \leftarrow (0FFDF_H)$. | ---1-0-- |
| 2 | DI | 60 | 1 | 3 | Disable all interrupts : $I \leftarrow "0"$ | -----0-- |
| 3 | EI | E0 | 1 | 3 | Enable all interrupt : $I \leftarrow "1"$ | -----1-- |
| 4 | NOP | FF | 1 | 2 | No operation | ----- |
| 5 | POP A | 0D | 1 | 4 | $sp \leftarrow sp + 1$, $A \leftarrow M(sp)$ | restored |
| 6 | POP X | 2D | 1 | 4 | $sp \leftarrow sp + 1$, $X \leftarrow M(sp)$ | |
| 7 | POP Y | 4D | 1 | 4 | $sp \leftarrow sp + 1$, $Y \leftarrow M(sp)$ | |
| 8 | POP PSW | 6D | 1 | 4 | $sp \leftarrow sp + 1$, $PSW \leftarrow M(sp)$ | |
| 9 | PUSH A | 0E | 1 | 4 | $M(sp) \leftarrow A$, $sp \leftarrow sp - 1$ | ----- |
| 10 | PUSH X | 2E | 1 | 4 | $M(sp) \leftarrow X$, $sp \leftarrow sp - 1$ | |
| 11 | PUSH Y | 4E | 1 | 4 | $M(sp) \leftarrow Y$, $sp \leftarrow sp - 1$ | |
| 12 | PUSH PSW | 6E | 1 | 4 | $M(sp) \leftarrow PSW$, $sp \leftarrow sp - 1$ | |
| 13 | RET | 6F | 1 | 5 | Return from subroutine $sp \leftarrow sp + 1$, $pc_L \leftarrow M(sp)$, $sp \leftarrow sp + 1$, $pc_H \leftarrow M(sp)$ | ----- |
| 14 | RETI | 7F | 1 | 6 | Return from interrupt $sp \leftarrow sp + 1$, $PSW \leftarrow M(sp)$, $sp \leftarrow sp + 1$, $pc_L \leftarrow M(sp)$, $sp \leftarrow sp + 1$, $pc_H \leftarrow M(sp)$ | restored |
| 15 | STOP | EF | 1 | 3 | Stop mode (halt CPU, stop oscillator) | ----- |

This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.