ABOV SEMICONDUCTOR
8-BIT SINGLE-CHIP MICROCONTROLLERS

# MC80F5132/5232

*User's Manual (Ver. 1.02)*

---

## REVISION HISTORY

### VERSION 1.02 (MAY. 2008)

Add C Flag to the **No.53**(INC A) of **ARITHMETIC/LOGIC OPEATION** in **APENDIX.B**.
Change the full-name of device into MC80F5132C.

Remove the Mask Option contents of **17.3 Low Voltage Detection Mode** in Page 68.

### VERSION 1.01 (MAR. 2008)

Fix error in Figure 19-1 : Remove capacitors

### VERSION 1.00 (AGU., 2007)

First Edition (Preliminary)

---

**Version 1.00**

**Published by**
**MCU Field Application Team**



---

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

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

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

# MC80F5132/5232

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

## 1. OVERVIEW

### 1.1 Description

The MC80F5132/5232 is an advanced CMOS 8-bit microcontroller with 32K bytes of ROM. The device is one of GMS800 family. The ABOV MC80F5132/5232 is a powerful microcontroller which provides a highly flexible and cost effective solution to many UR applications.The MC80F5132/5232 provides the following standard features: 32K bytes of code flash memory, 512 bytes of RAM, 8-bit timer/counter, on-chip oscillator and clock circuitry. In addition, the MC80F5132/5232 supports power saving modes to reduce power consumption.

| Device Name | ROM Size (Flash) | RAM Size | Package |
|---|---|---|---|
| MC80F5132 | 32K Bytes | 512 Bytes (included 256 bytes stack memory) | 28 SOP |
| MC80F5232 | | | 24 SOP |

### 1.2 Features

- **Instruction Cycle Time:**
  - **1us at 4MHz**
- **Programmable I/O pins**

| | 24 PIN | 28 PIN | Description |
|---|---|---|---|
| INPUT | 2 | 2 | /RESET, XIN |
| OUTPUT | 2 | 2 | REMOUT, XOUT |
| I/O | 18 | 22 | Programmable I/O |

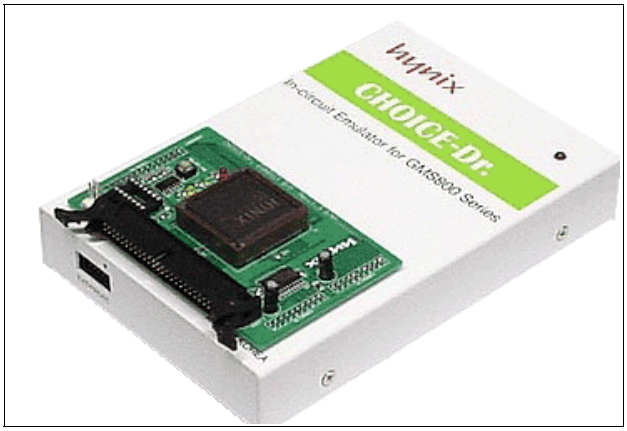
- **Operating Voltage**
  - **2.0 ~ 4.0 V @ 4MHZ**
- **Timer**
  - **One 8-bit Basic Interval Timer**
  - **Four 8-bit Timer/counters (or two 16-bit Timer/counter)**
  - **One 6-bit Watchdog timer- Watch Dog Timer**
- **8 Interrupt sources**
  - **Nested Interrupt control is available.**
  - **External 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)**
- **Package**
  - **24SOP, 28SOP**

## 1.3 Development Tools

The MC80F5132/5232 are supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr.TM and Flash programmers. There are two different type of programmers such as single type and gang type. Macro assembler operates under the MS-Windows 95 and upversioned Windows OS and HMS800C operates under upversioned the MS-Windows NT.

Please contact sales part of ABOV semiconductor.

| | |
|---|---|
| Software | - MS-Windows based assembler<br>- MS-Windows based Debugger<br>- HMS800 C compiler |
| Hardware (Emulator) | - CHOICE-Dr.<br>- CHOICE-Dr. EVA81C5 B/D |
| FLASH Writer | - CHOICE - SIGMA I/II (Single writer)<br>- PGM Plus III (Single writer)<br>- Standalone GANG4 I/II (Gang writer) |

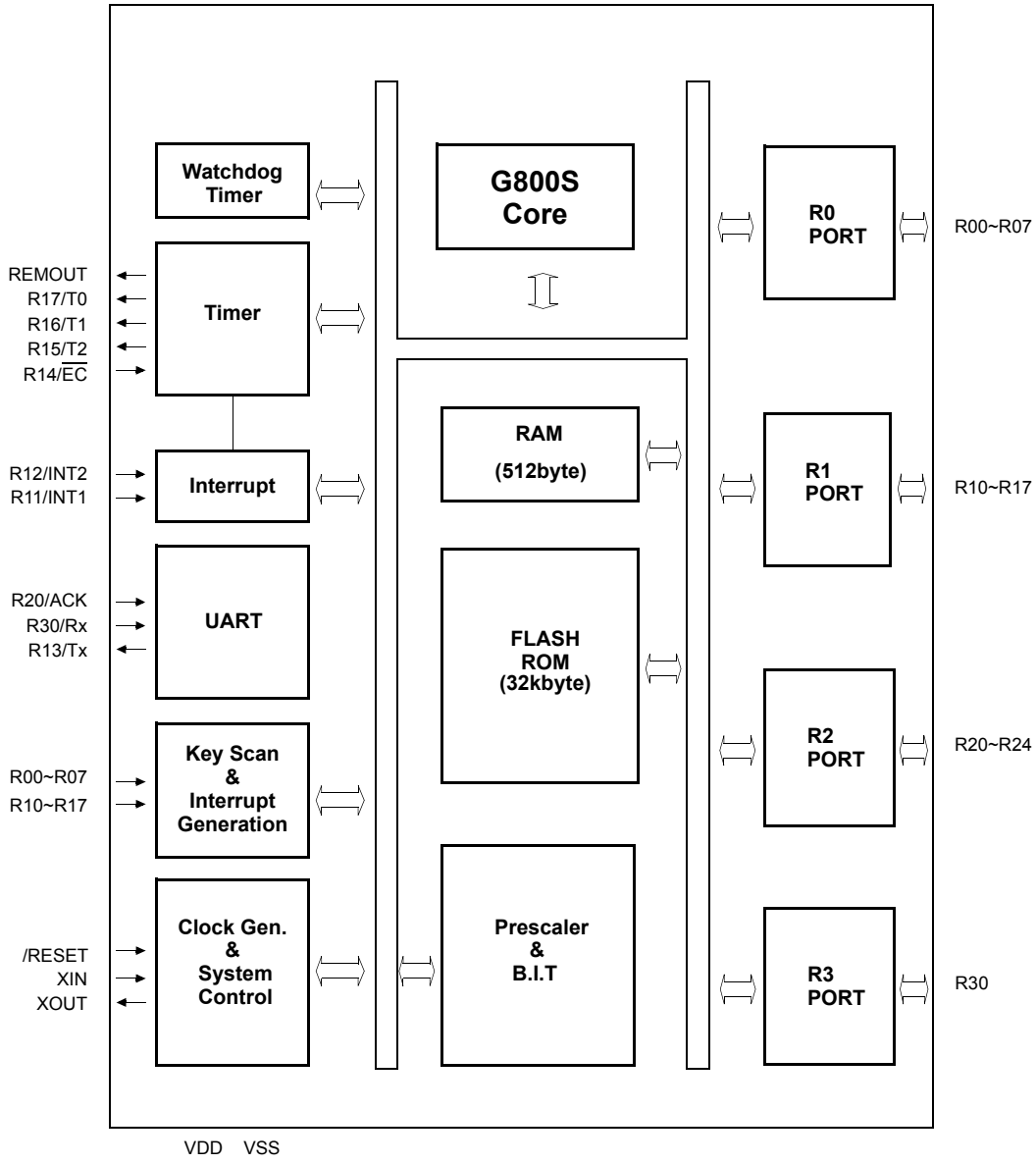



Choice-Dr. (Emulator)


PGMplus III ( Single Writer )


Standalone Gang4 II ( Gang Writer )

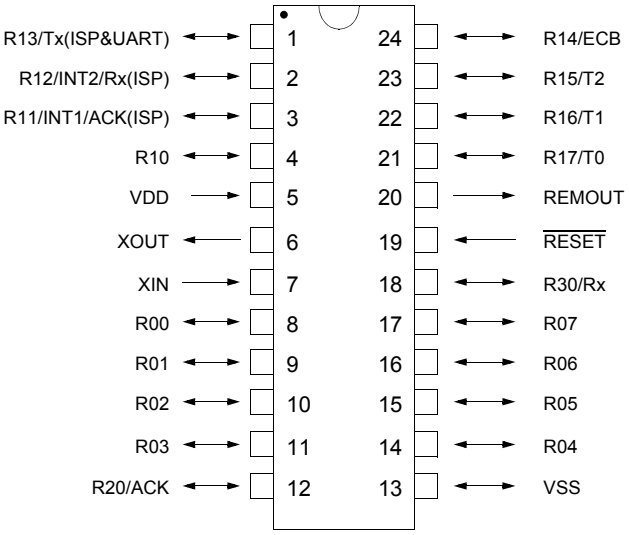
## 2. BLOCK DIAGRAM



## 3. PIN ASSIGNMENT (Top View)

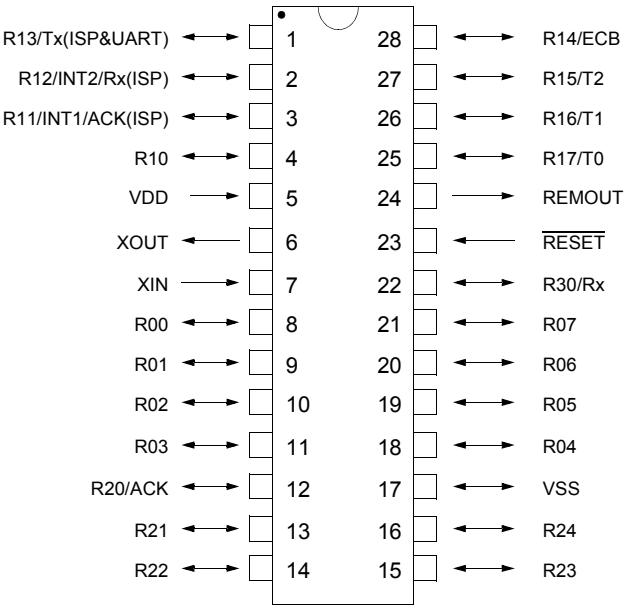
### MC80F5232

**24 SOP**




### MC80F5132

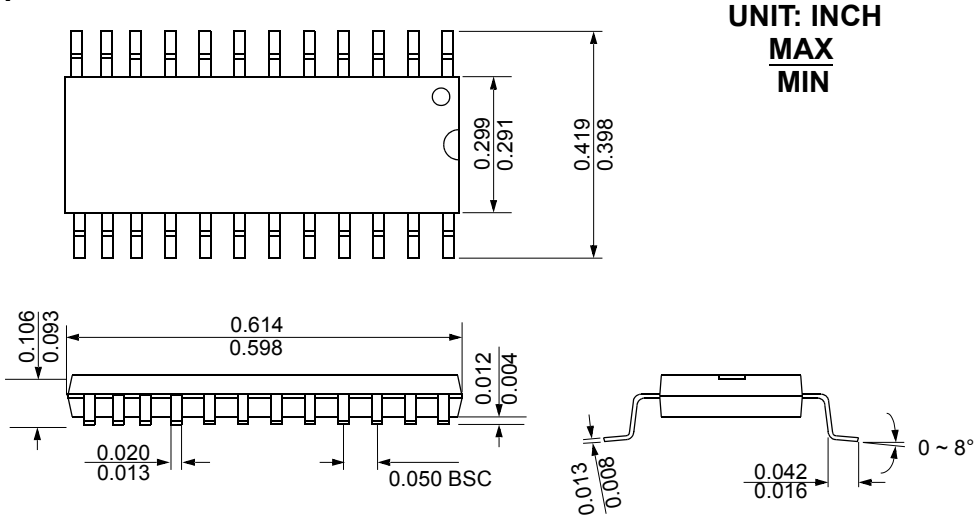
**28 SOP**



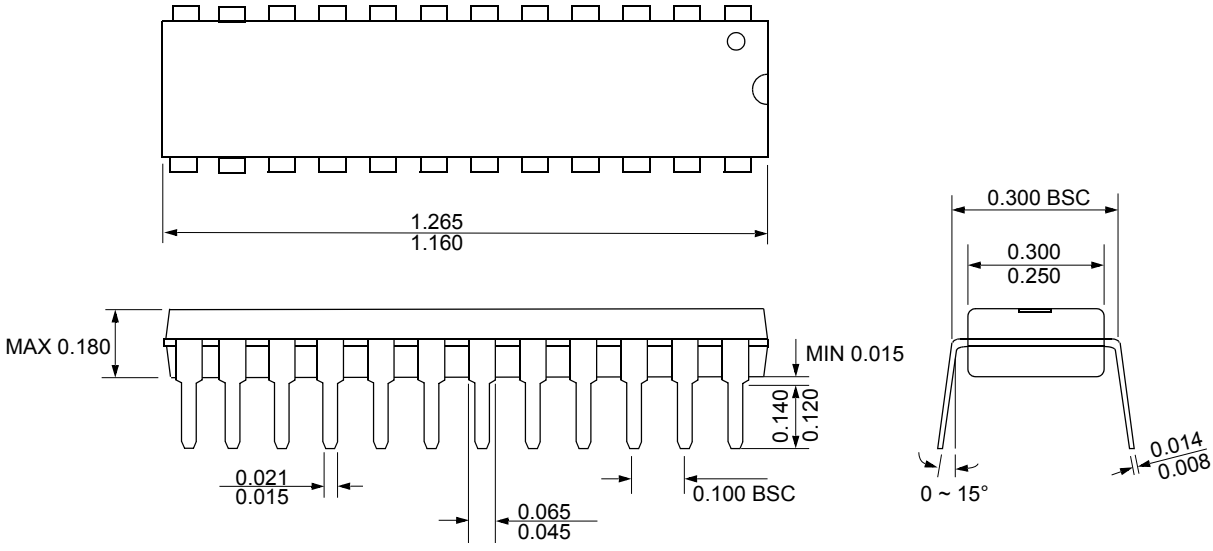
## 4. PACKAGE DIMENSION

### 24 SOP

UNIT: INCH
MAX
MIN

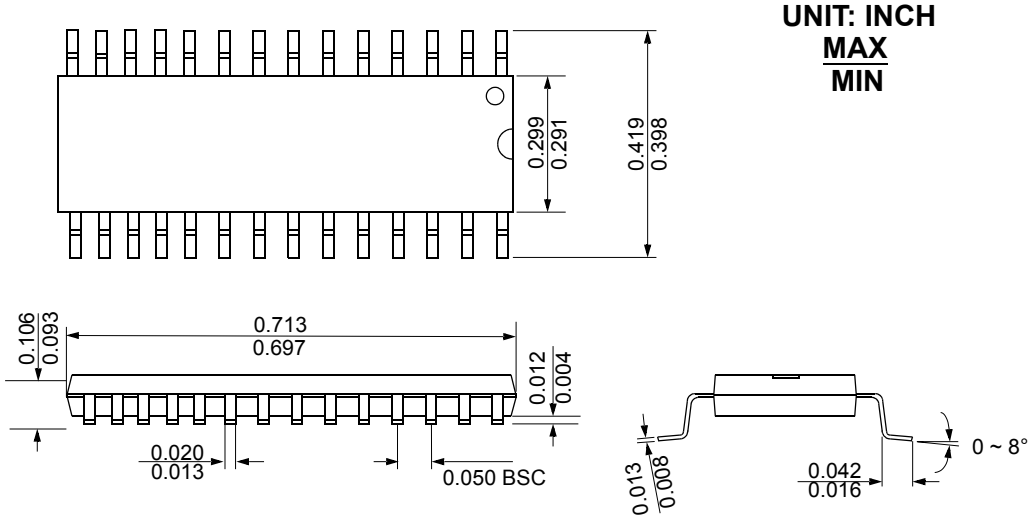



### 24 SKDIP

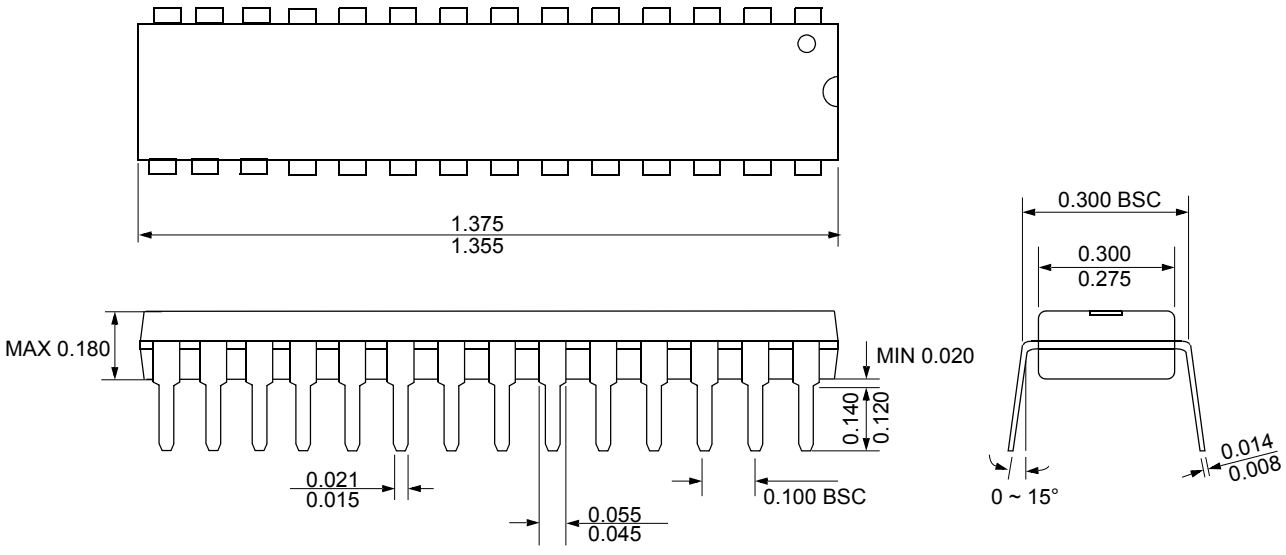


## 28 SOP

UNIT: INCH
MAX
MIN




## 28 SKDIP



## 5. PIN FUNCTION

**$V_{DD}$**: Supply voltage.

**$V_{SS}$**: Circuit ground.

**$\overline{\text{RESET}}$**: Reset the MCU.

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

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

**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) |
| R13 | UART Tx |
| R14 | $\overline{\text{EC}}$ (Event Counter input) |
| R15 | T2 (Timer / Counter output 2) |
| R16 | T1 (Timer / Counter output 1) |
| R17 | T0 (Timer / Counter output 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.

| Port pin | Alternate function |
|---|---|
| R20 | UART ACK |

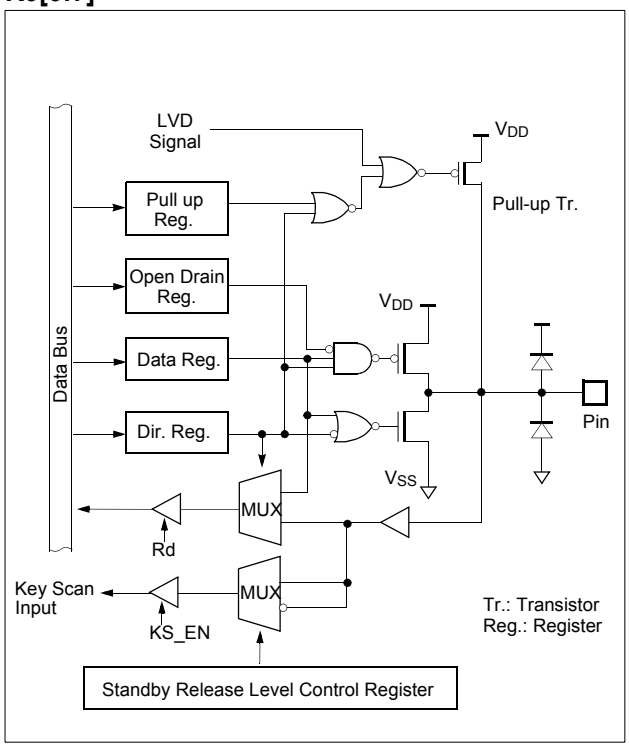
**R30** : R30 is an 8-bit CMOS bidirectional I/O port. R30 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs.

| Port pin | Alternate function |
|---|---|
| R30 | UART Rx |

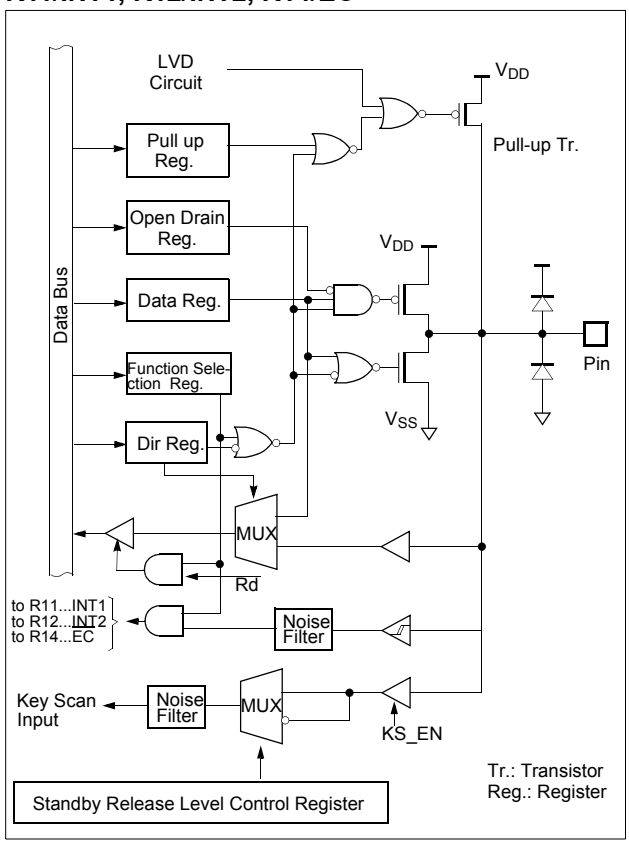
| PIN NAME | INPUT/ OUTPUT | Function | @RESET | @STOP |
|---|---|---|---|---|
| R00 | I/O | - Each bit of the port can be individually configured as an input or an output by user software<br>- Push-pull output<br>- CMOS input with pull-up resister (option)<br>- Can be programmable as key scan input<br>- Pull-up resisters 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 | | | |
| R10 | I/O | - Each bit of the port can be individually configured as an input or an output by user software<br>- Push-pull output<br>- CMOS input with pull-up resister (option)<br>- Can be programmable as key scan input or open drain output<br>- Pull-up resisters are automatically disabled at output mode<br>- Direct driving of LED(N-Tr.) | INPUT | State of before Stop |
| R11/INT1 | I/O | | | |
| R12/INT2 | I/O | | | |
| R13 | I/O | | | |
| R14/$\overline{EC}$ | I/O | | | |
| R15/T2 | I/O | | | |
| R16/T1 | I/O | | | |
| R17/T0 | I/O | | | |
| R20 | I/O | - Each bit of the port can be individually configured as an input or an output by user software<br>- Push-pull output<br>- CMOS input with pull-up resister (option)<br>- Pull-up resisters are automatically disabled at output mode<br>- Direct driving of LED(N-Tr.) | INPUT | State of before Stop |
| R21 | I/O | | | |
| R22 | I/O | | | |
| R23 | I/O | | | |
| 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 |
| R30 | I/O | - I/O, Internal pull-up(auto-disabled at output mode)<br>- 1bit push-pull output | | |
| VDD | P | Positive power supply | | |
| VSS | P | Ground | | |

## 6. PORT STRUCTURES

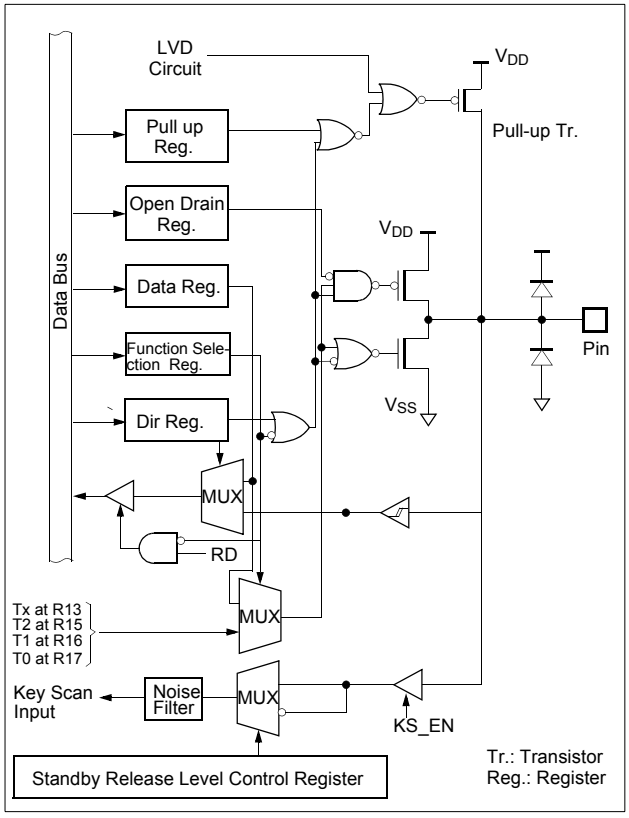
### R0[0:7]



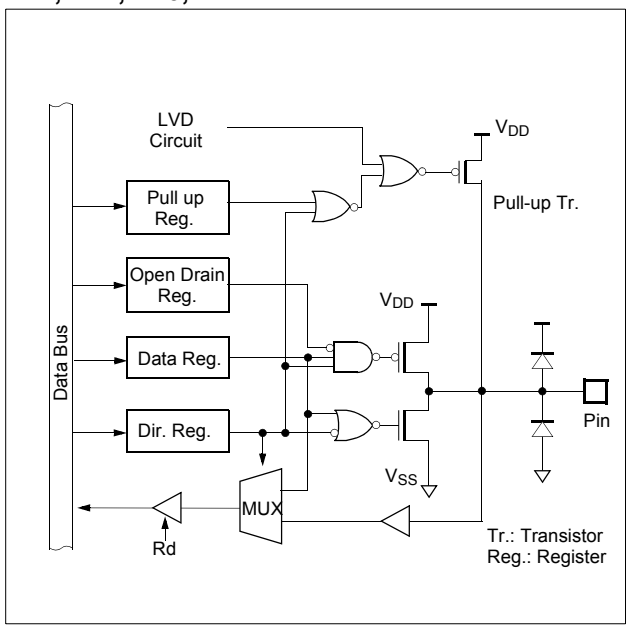

### R11/INT1, R12/INT2, R14/$\overline{EC}$



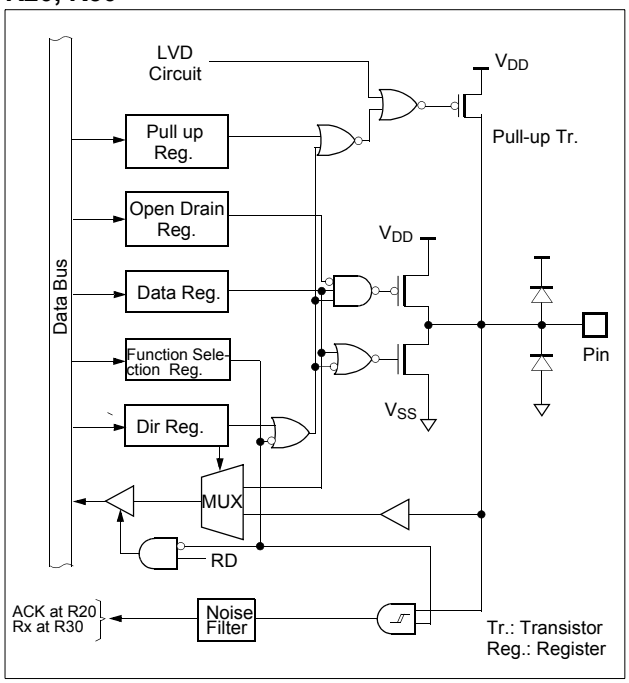

### R13, R15, R16, R17



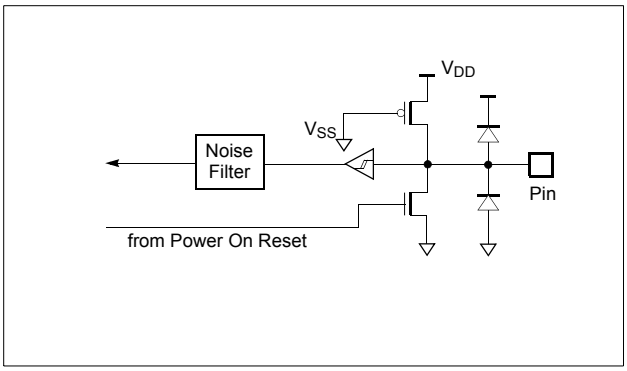

### R21, R22, R23, R24



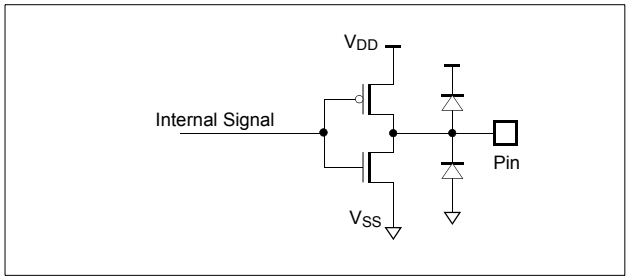
### R20, R30



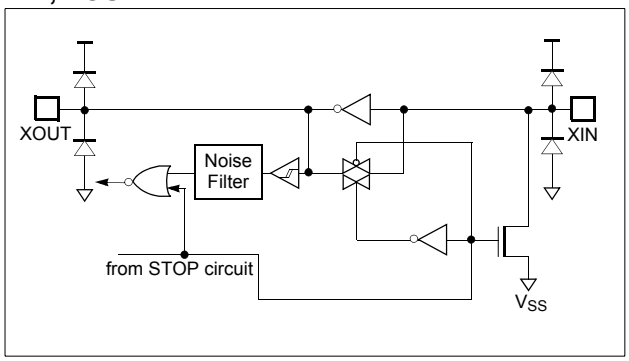

### RESET




### REMOUT




### XIN, XOUT



# 7. ELECTRICAL CHARACTERISTICS

## 7.1 Absolute Maximum Ratings

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

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

## 7.2 Recommended Operating Conditions

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

## 7.3 DC Electrical Characteristics

($T_A$=-0~70°C, $V_{DD}$=2.0~3.6V, GND=0V)

| Parameter | Symbol | Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| High level input Voltage | $V_{IH1}$ | R11,R12,R14,R30,$\overline{\text{RESET}}$ | 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,R30,$\overline{\text{RESET}}$ | 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,R3<br>$\overline{\text{RESET}}$ (without pull-up),$V_{IH}$= VDD | - | - | 1 | μA |
| Low level input Leakage Current | $I_{IL}$ | R0,R1,R2,R3<br>$\overline{\text{RESET}}$ (without pull-up),$V_{IL}$= 0 | - | - | -1 | μA |
| High level output Voltage | $V_{OH1}$ | R0, $I_{OH}$=-0.5mA | VDD-0.4 | - | - | V |
| | $V_{OH2}$ | R1, R2, R3, $I_{OH}$=-1.0mA | VDD-0.4 | - | - | V |
| | $V_{OH3}$ | XOUT,$I_{OH}$=-200μA | VDD-0.9 | - | - | V |
| Low level output Voltage | $V_{OL1}$ | R0, $I_{OL}$=1mA | - | - | 0.4 | V |
| | $V_{OL2}$ | R1, R2, R3, $I_{OL}$=5mA | - | - | 0.8 | V |
| | $V_{OL3}$ | XOUT, $I_{OL}$=200μA | - | - | 0.8 | V |
| High level output Leakage Current | $I_{OHL}$ | R0,R1,R2,R3, $V_{OH}$= VDD | - | - | 1 | μA |
| Low level output Leakage Current | $I_{OLL}$ | R0,R1,R2,R3, $V_{OL}$= 0 | - | - | -1 | μA |

| Parameter | Symbol | Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| High Level output current | $I_{OH}$ | REMOUT, $V_{OH}$ =2V | -24 | -12 | -5 | mA |
| Low Level output current | $I_{OL}$ | REMOUT, $V_{OL}$ =1V | 0.3 | - | 3 | mA |
| Input pull-up current | $I_p$ | R0,R1,R2,R3, $\overline{\text{RESET}}$, VDD=3V | 15 | 30 | 60 | μA |
| Feedback resistance | $R_{FD}$ | XIN, VDD=3V | 0.1 | 0.4 | 1.0 | MΩ |
| Power Supply Current | $I_{DD1}$ | Operating current, fxin=4Mhz, VDD=2.0V | - | 2.4 | 6 | mA |
| | $I_{DD2}$ | Operating current, fxin=4Mhz, VDD=4.0V | - | 4 | 10 | mA |
| | $I_{SLP1}$ | Sleep mode current, fxin=4Mhz, VDD=2.0V | - | 1 | 2 | mA |
| | $I_{SLP2}$ | Sleep mode current, fxin=4Mhz, VDD=4.0V | - | 2 | 3 | mA |
| | $I_{STP1}$ | Stop mode current, Oscillator Stop VDD=2.0V | - | 2 | 8 | μA |
| | $I_{STP2}$ | Stop mode current, Oscillator Stop VDD=4.0V | - | 3 | 10 | μA |
| RAM retention supply voltage | $V_{RET}$ | - | 0.7 | - | - | V |

## 7.4 REMOUT Port Ioh Characteristics Graph

(typical process & room temperature)

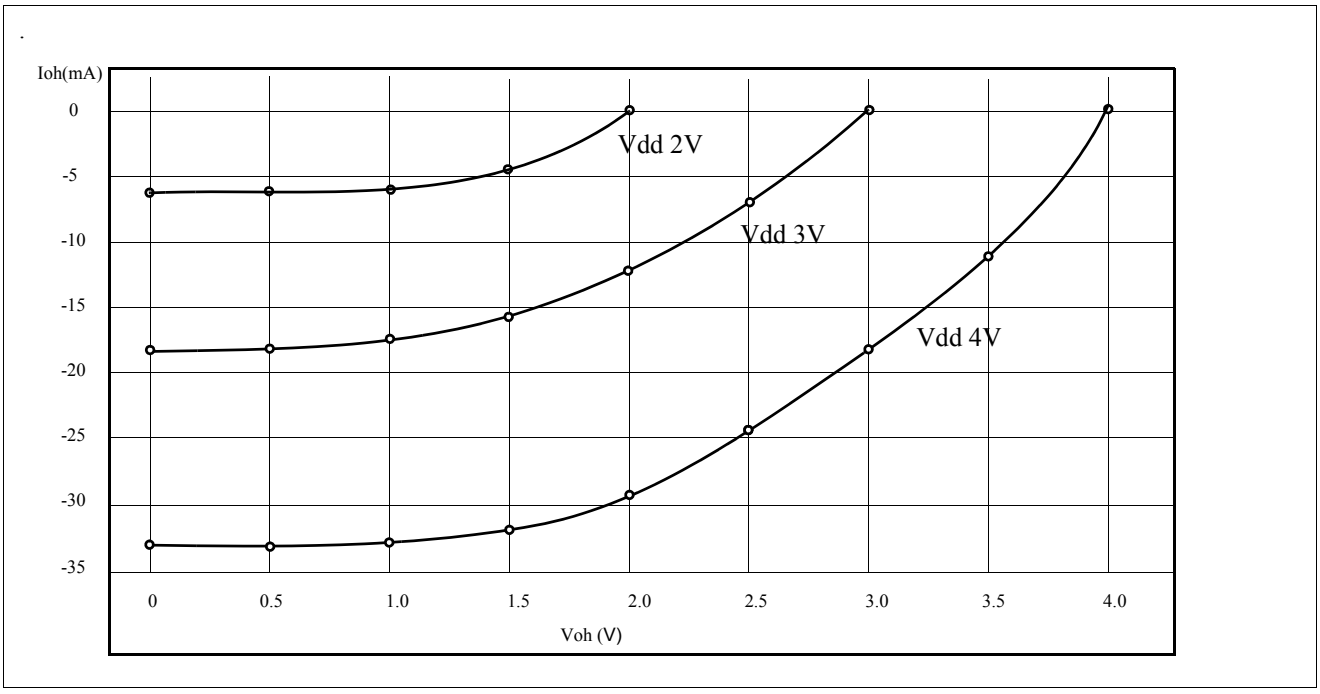



**Figure 7-1 Ioh vs Voh**

## 7.5 REMOUT Port Iol Characteristics Graph

(typical process & room temperature)

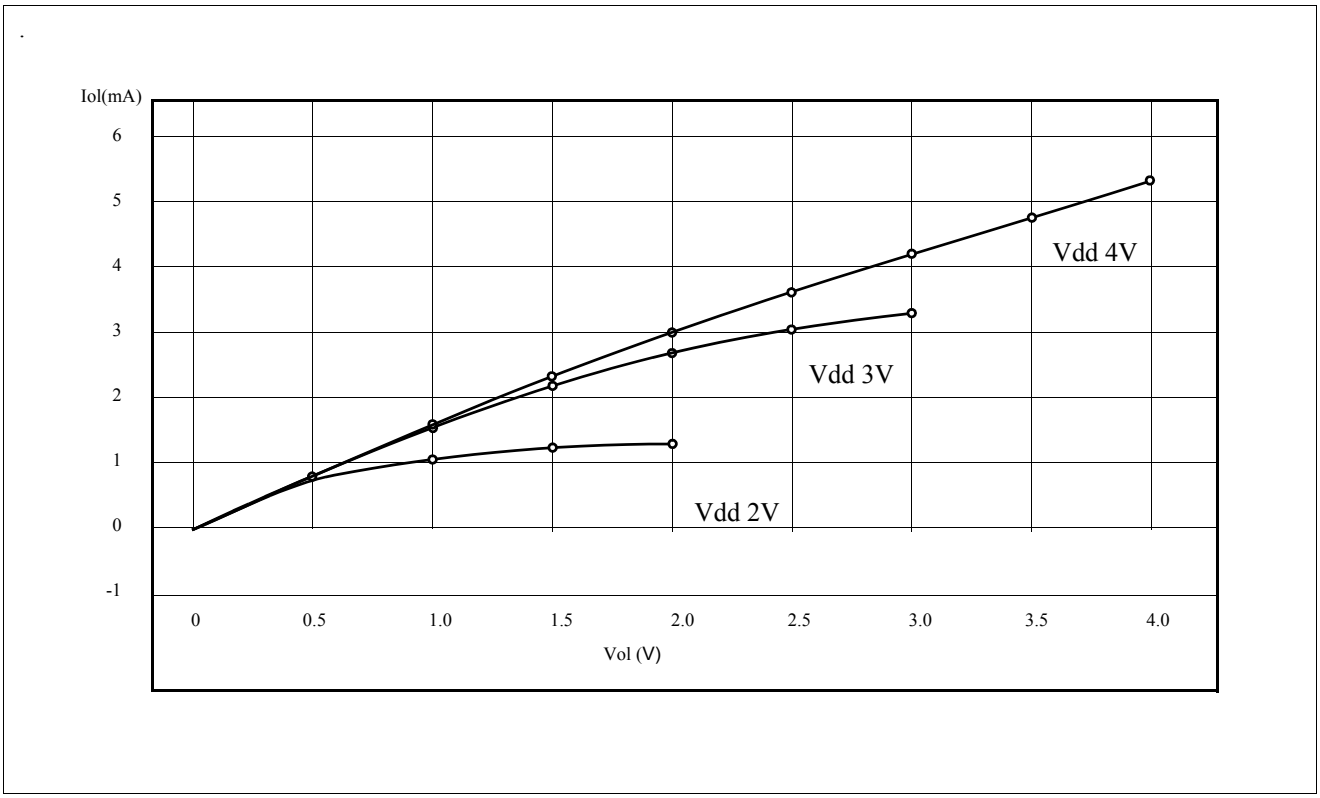



**Figure 7-2 Iol vs Vol**

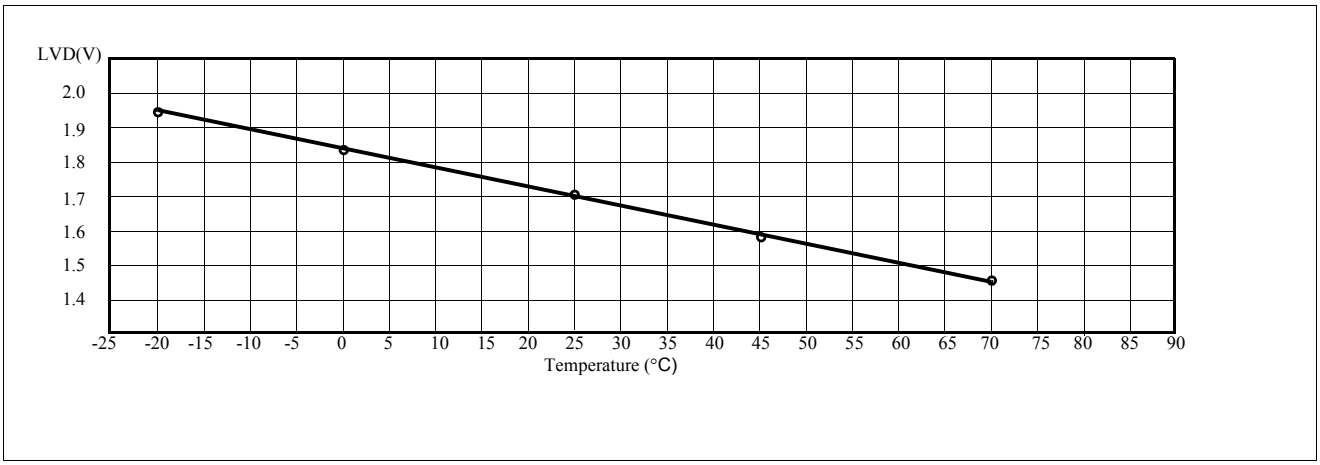



**Figure 7-3 Low Voltage vs Temperature**

## 7.6 AC Characteristics

($T_A$=0~+70°C, $V_{DD}$=2.0~3.6V, $V_{SS}$=0V)

| 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 |

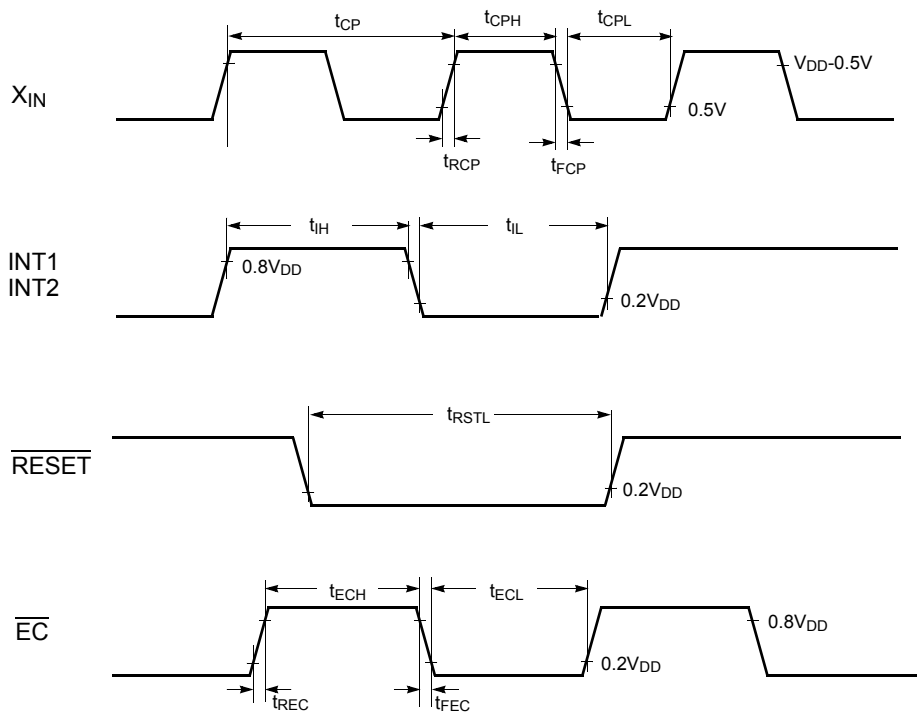



**Figure 7-4 Timing Diagram**

# 8. MEMORY ORGANIZATION

The MC80F5132/5232 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.

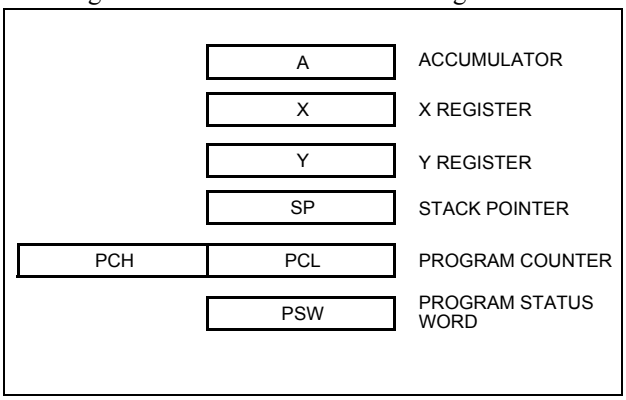



**Figure 8-1 Configuration of Registers**

### 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 => YA). In the case of division instruction, execute as the lower 8-bit of dividend. After division operation, quotient enters.

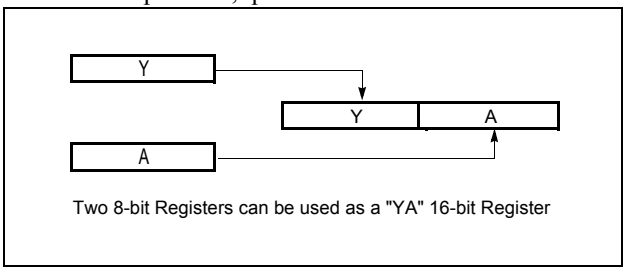



**Figure 8-2 Configuration of YA 16-bit Register**

### 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**

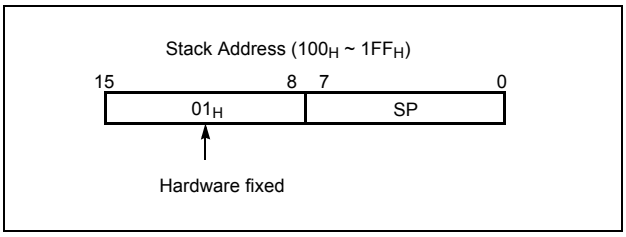
In the case of 16-bit operation instruction, execute as the upper 8-bit of YA. (16-bit accumulator). In the case of multiplication instruction, execute as a multiplicand register. After multiplication operation, the upper 8-bit of the result enters. In the case of division instruction, execute as the 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
```

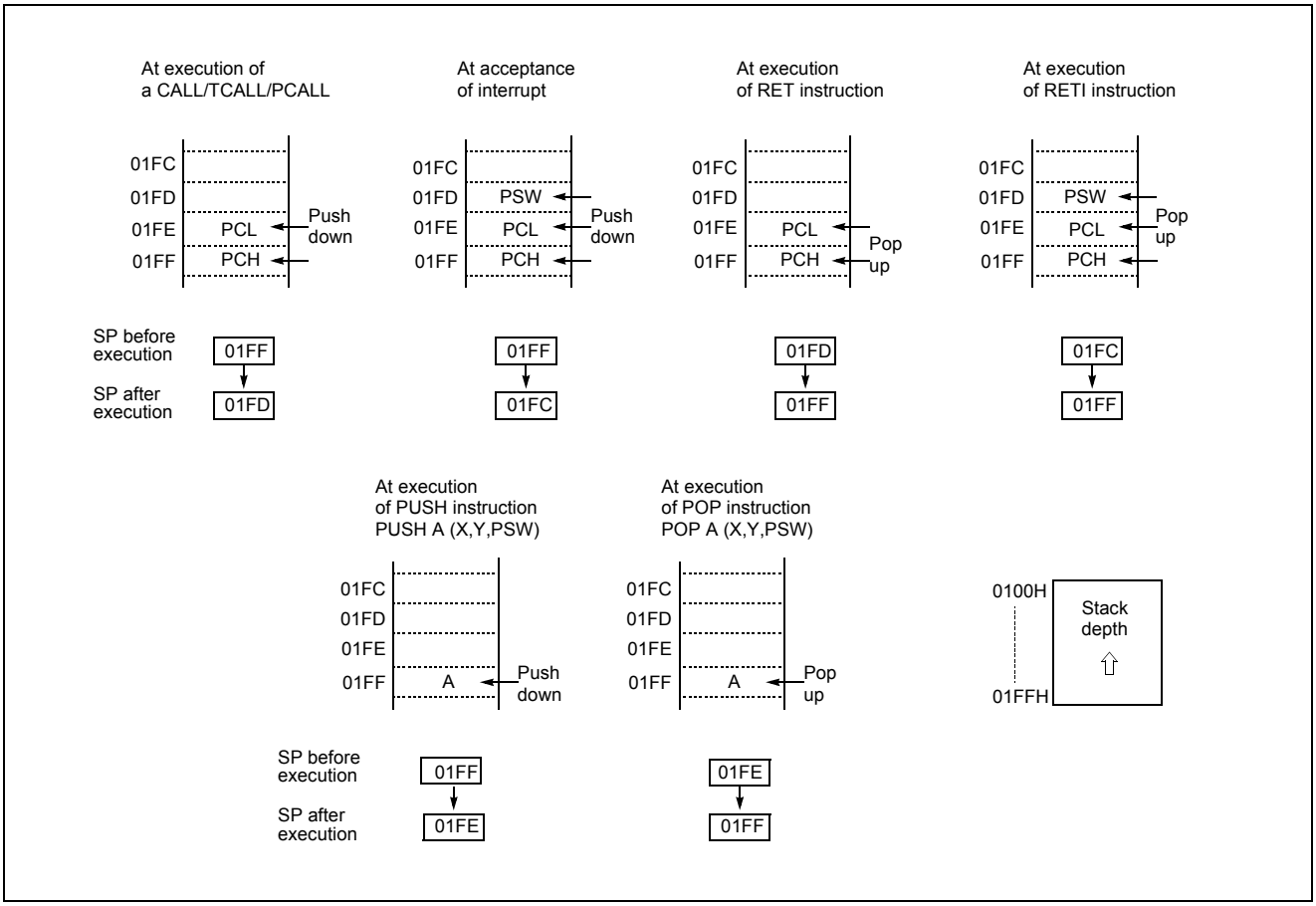



**Figure 8-3 Stack Operation**

### 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.

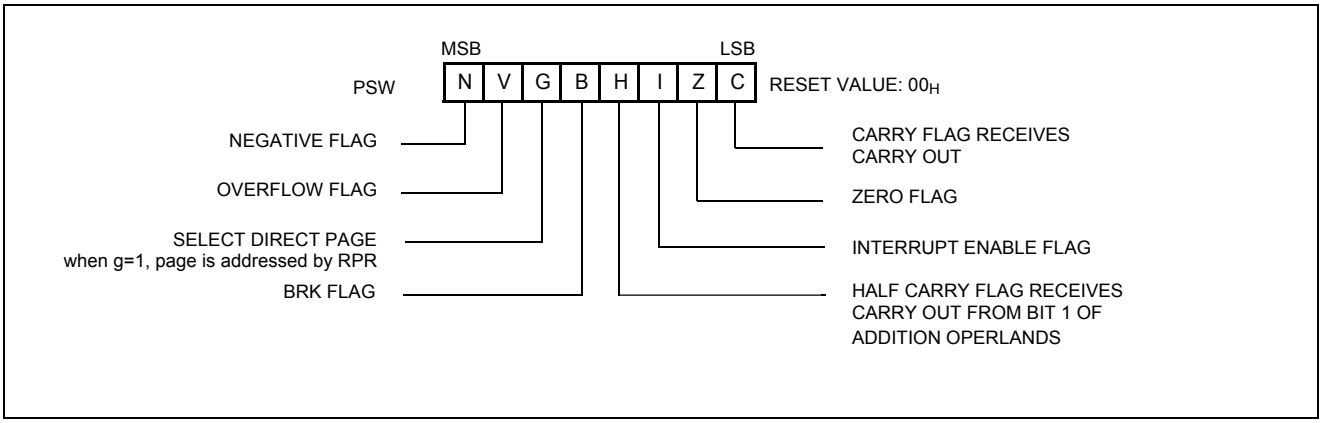



**Figure 8-4 PSW (Program Status Word) Register**

**[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 CLRV instruction with Overflow flag (V).

**[Break flag B]**

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

**[Direct page flag G]**

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

**[Overflow flag V]**

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

**[Negative flag N]**

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

## 8.2 Program Memory

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

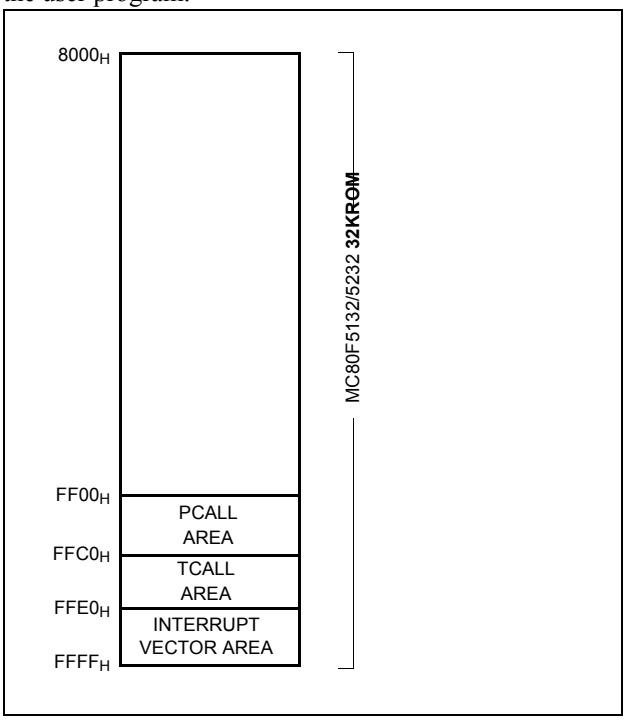



**Figure 8-5 Program Memory Map**

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

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

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 | UART Tx Interrupt Vector Area |
| EC | UART Rx Interrupt Vector Area |
| 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**

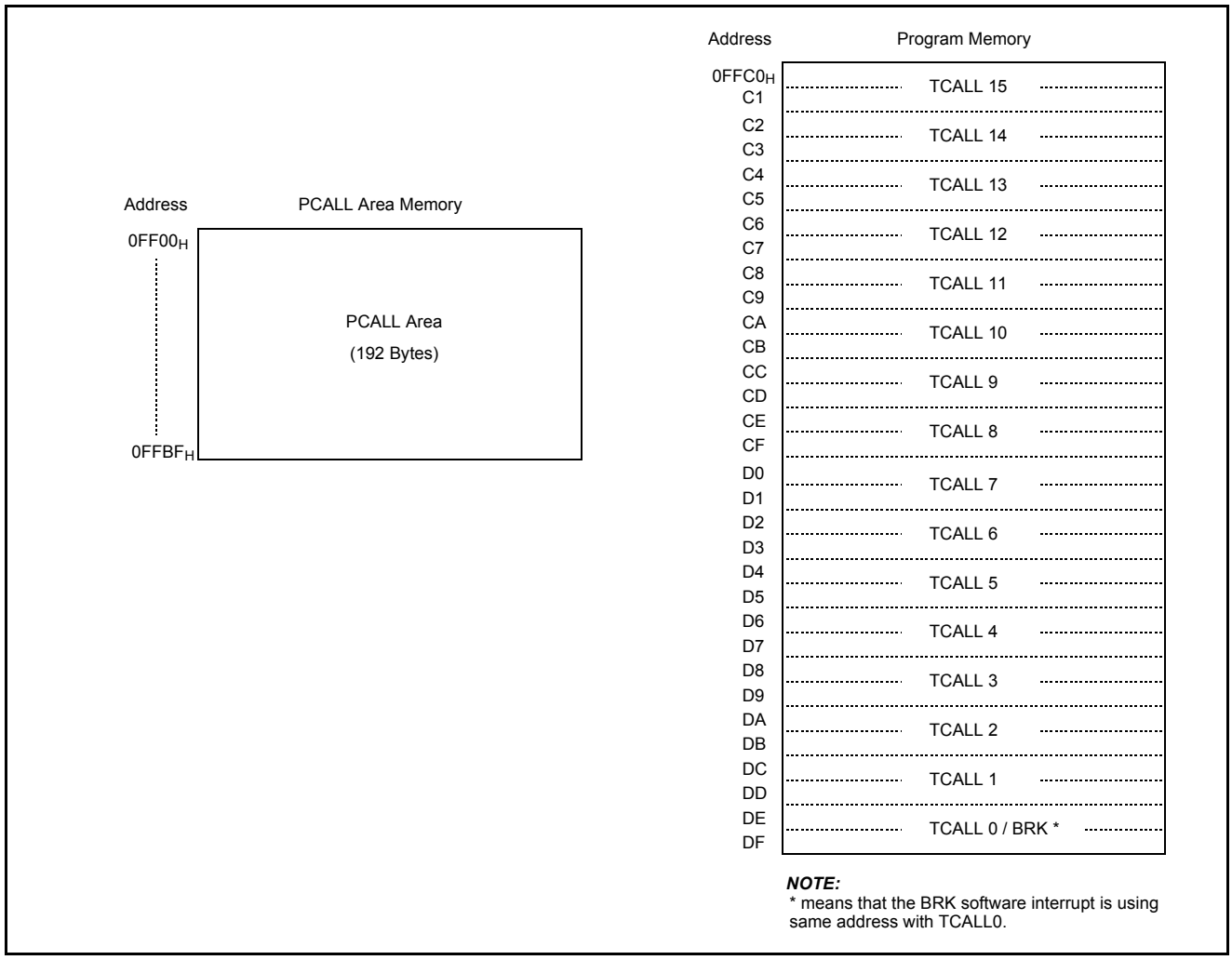
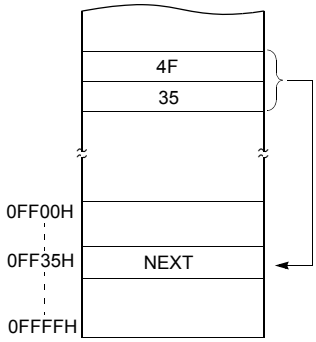



**Figure 8-7 PCALL and TCALL Memory Area**

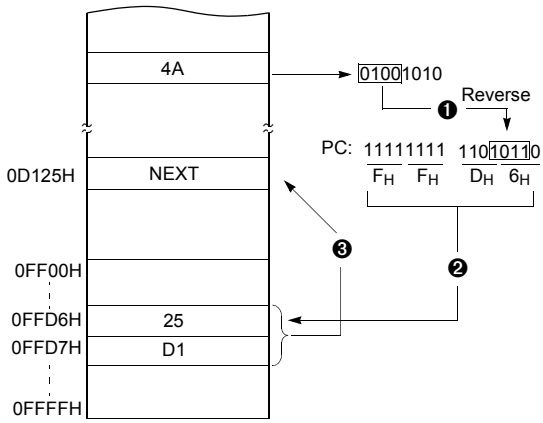
### PCALL→ rel

```
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                  ;MC80F5132/5232 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.

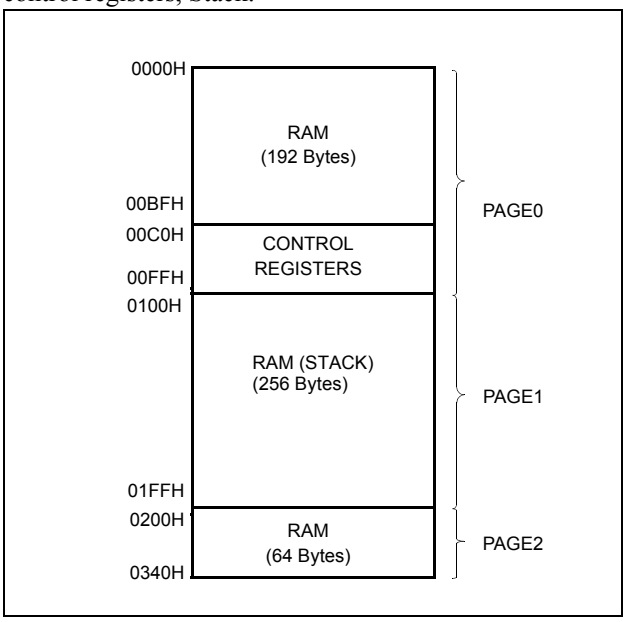



**Figure 8-8 Data Memory Map**

### User Memory

The MC80F5132/5232 has 512 × 8 bits for the user memory (RAM).

### Control Registers

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

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

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

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

Example; To write at CKCTLR

```
LDM    CLCTLR,#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 16.

## 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 |
| 00C7h | CLOCK CONTROL REG. | CKCTLR | W | --110111b |
| | BASIC INTERVAL REG. | BITR | 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 | --000000b |
| 00CBh | EXT. INT. EDGE SELECTION | IEDS | W | --0000--b |
| 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. | | R | 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 | RAM Page Register | RPR | R/W | -----001b |
| 00DCh | STANDBY MODE RELEASE REG0 | SMPR0 | W | 00000000b |
| 00DDh | STANDBY MODE RELEASE REG0 | SMPR1 | W | 00000000b |
| 00DEh | PORT R1 OPEN DRAIN ASSIGN REG. | R1ODC | W | 00000000b |
| 00DFh | PORT R2 OPEN DRAIN ASSIGN REG. | R2ODC | W | 00000000b |

| | | | | |
|---|---|---|---|---|
| 00E0h | PORT R3 OPEN DRAIN ASSIGN REG. | R3ODC | W | -------0b |
| 00E4h | PORT R0 OPEN DRAIN ASSIGN REG. | R0ODC | W | 00000000b |
| 00E5h | PORT R3 DATA REG. | R3 | R/W | undefined |
| 00E6h | PORT R3 DATA DIRECTION REG. | R3DD | W | -------0b |
| 00EBh | Asynschronous Serial Interface Mode REG. | ASIMR | R/W | 0000-00-b |
| 00ECh | Asynschronous Serial Interface Status REG. | ASISR | R | -----000b |
| 00EDh | Baud Rate Generate Control REG. | BRGCR | R/W | -0010000b |
| 00EEh | Recevie Buffer REG. | RXBR | R | 00000000b |
| | Transmit Shift REG. | TXBR | W | 11111111b |
| 00EFh | Low Voltage Indication REG. | LVIR | R | ------00b |
| 00F0h | SLEEP MODE REG. | SLPM | W | - - - - - - - 0b |
| 00F1h | LVD Control REG. | LVDC | W | ----00--b |
| 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 | PORT R3 PULL-UP REG. CONT. REG. | R3PC | W | -------0b |

| | |
|---|---|
| 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 MC80F5132/5232 uses six addressing modes;

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

### (1) Register Addressing

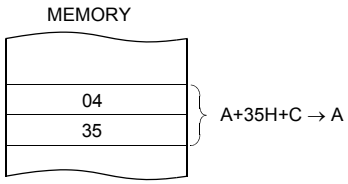
Register addressing accesses the A, X, Y, C and PSW.

### (2) Immediate Addressing → #imm

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

Example:

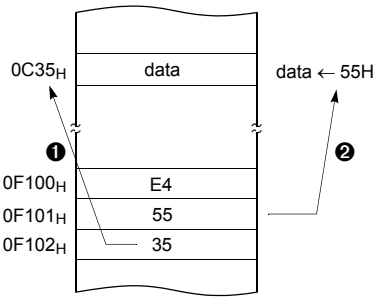
```
0435     ADC    #35H
```




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
```

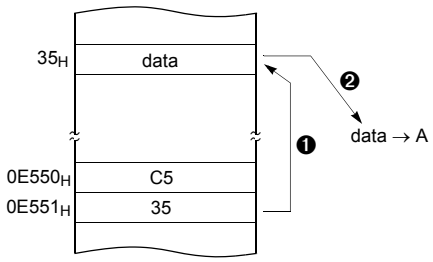



### (3) Direct Page Addressing → dp

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

Example; G=0

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



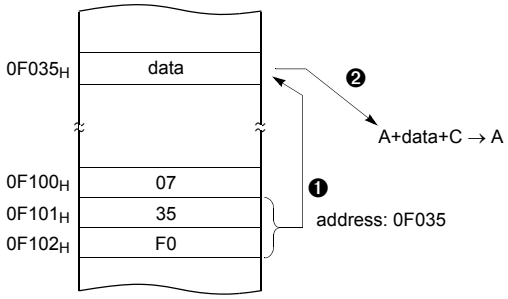

### (4) Absolute Addressing → !abs

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;

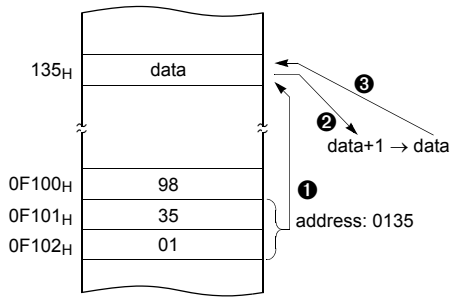
```
0735F0   ADC    !0F035H      ;A←ROM[0F035H]
```




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

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

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




### (5) Indexed Addressing

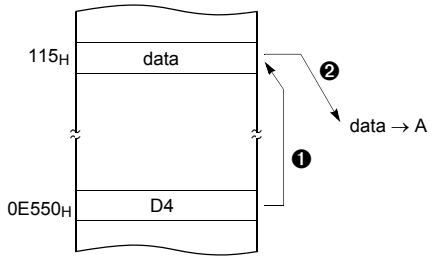
#### X indexed direct page (no offset) → {X}

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

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

Example; X=$15_H$, G=1, RPR=$01_H$

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



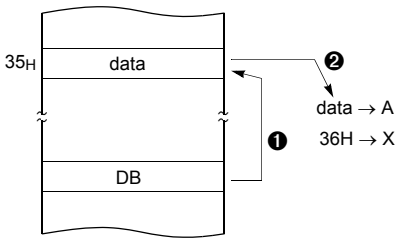

#### X indexed direct page, auto increment→ {X}+

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

LDA, STA

Example; G=0, X=$35_H$

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




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

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

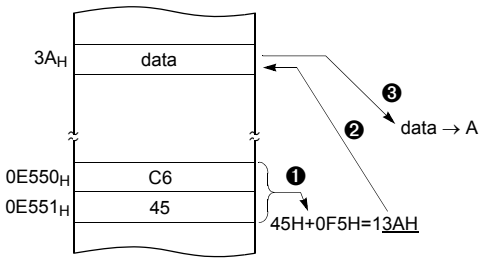
ADC, AND, CMP, EOR, LDA, LDY, OR, SBC, STA STY, XMA, ASL, DEC, INC, LSR, ROL, ROR

Example; G=0, X=$0F5_H$

Example; G=0, X=$0F5_H$

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

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




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

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

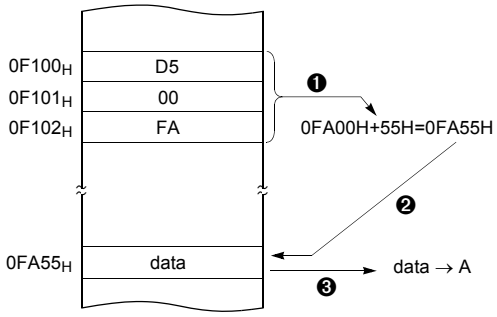
This is same with above (2). Use Y register instead of X.

### Y indexed absolute → !abs+Y

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

Example; Y=$55_H$

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




## (6) Indirect Addressing

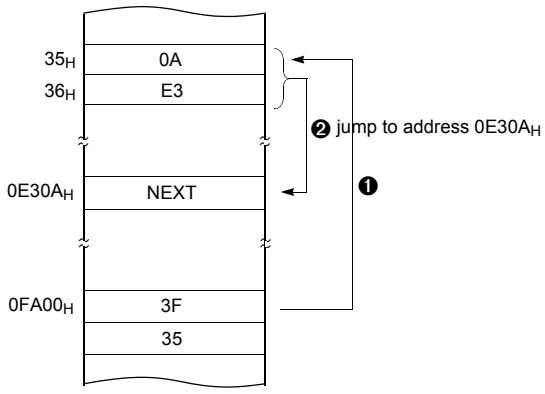
### Direct page indirect → [dp]

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

JMP, CALL

Example; G=0

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



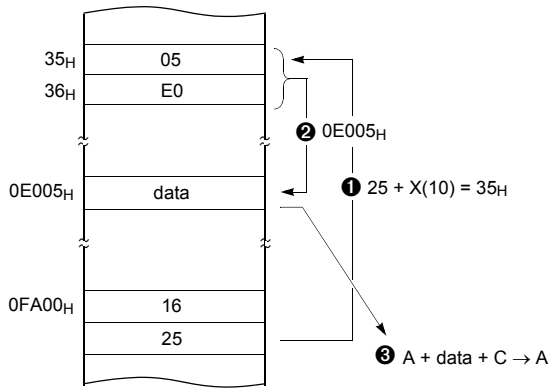

### X indexed indirect → [dp+X]

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

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

Example; G=0, X=$10_H$

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



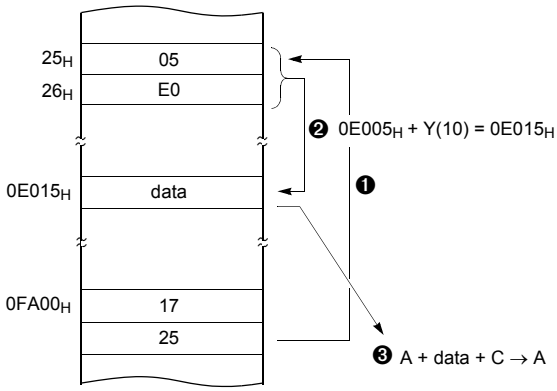
### Y indexed indirect → [dp]+Y

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

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

Example; G=0, Y=$10_H$

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



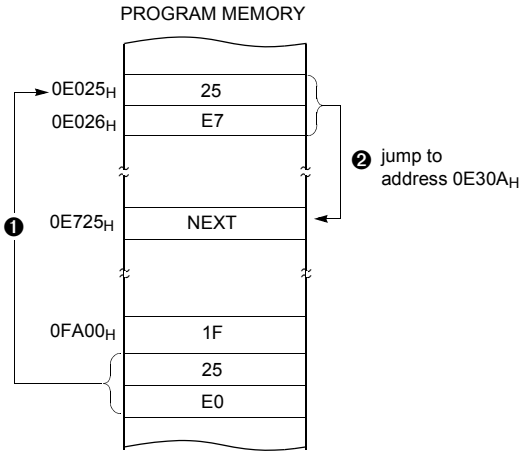

### Absolute indirect → [!abs]

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

JMP

Example; G=0

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



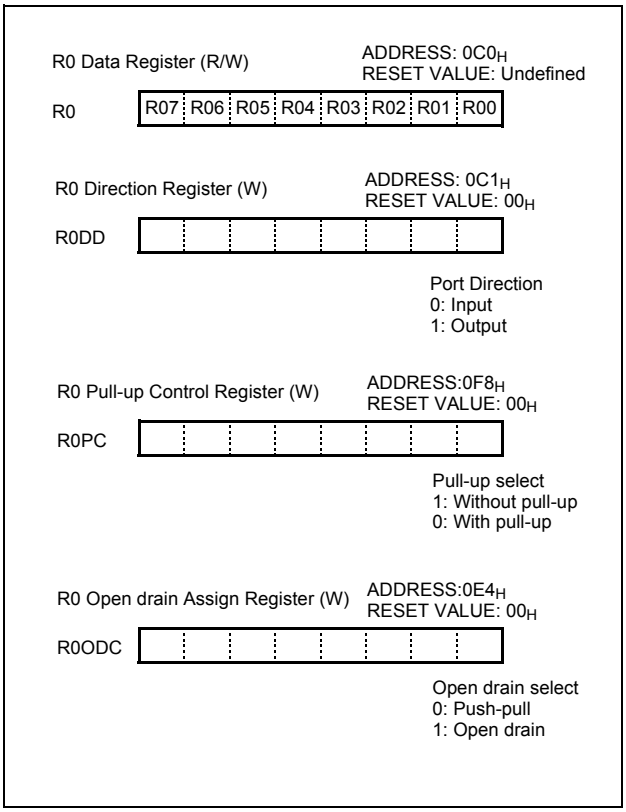
# 9. I/O PORTS

The MC80F5132/5232 has 22 I/O ports which are PORT0(8 I/O), PORT1 (8 I/O), PORT2 (5 I/O) and PORT3(1 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 $0C0_H$). Each I/O pin can independently used as an input or an output through the R0DD register (address $0C1_H$).

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 output 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.

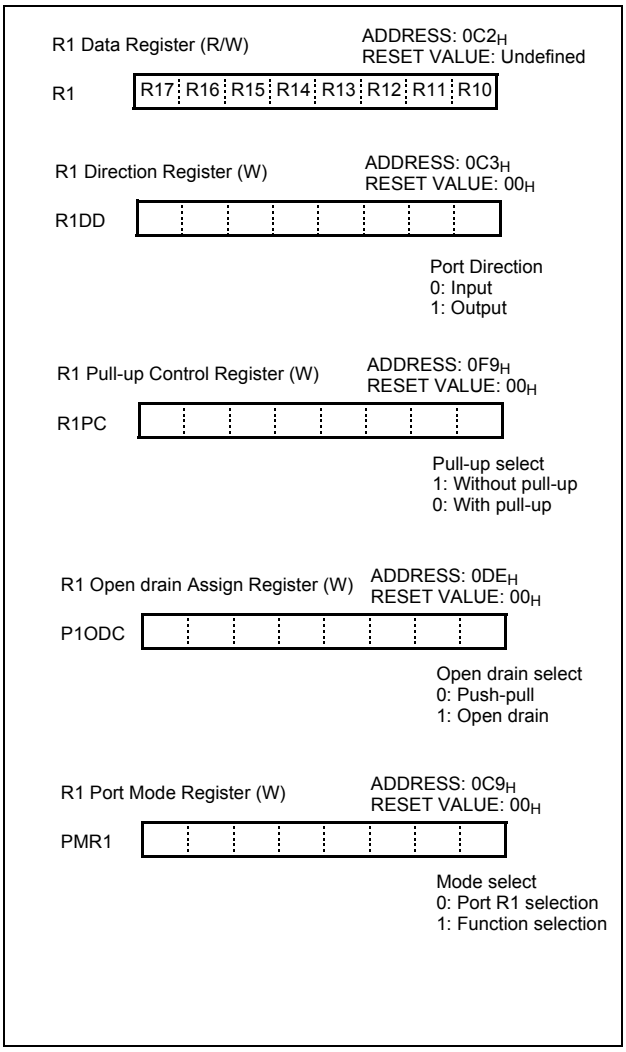
### (4) R0 Pull-up Control Register (R0PC)

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 $0C2_H$). Each I/O pin can independently used as an input or an output through the R1DD register (address $0C3_H$).

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 output 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".

| Bit Name | PMR1 | Selection Mode | Remarks |
|---|---|---|---|
| T0S | 0 | R17 (I/O) | - |
| | 1 | T0 (O) | Timer0 Output |
| T1S | 0 | R16 (I/O) | - |
| | 1 | T1 (O) | Timer1 Output |
| T2S | 0 | R15 (I/O) | - |
| | 1 | T2 (O) | Timer2 Output |
| 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) | |

**Table 9-1 Selection mode of PMR1**

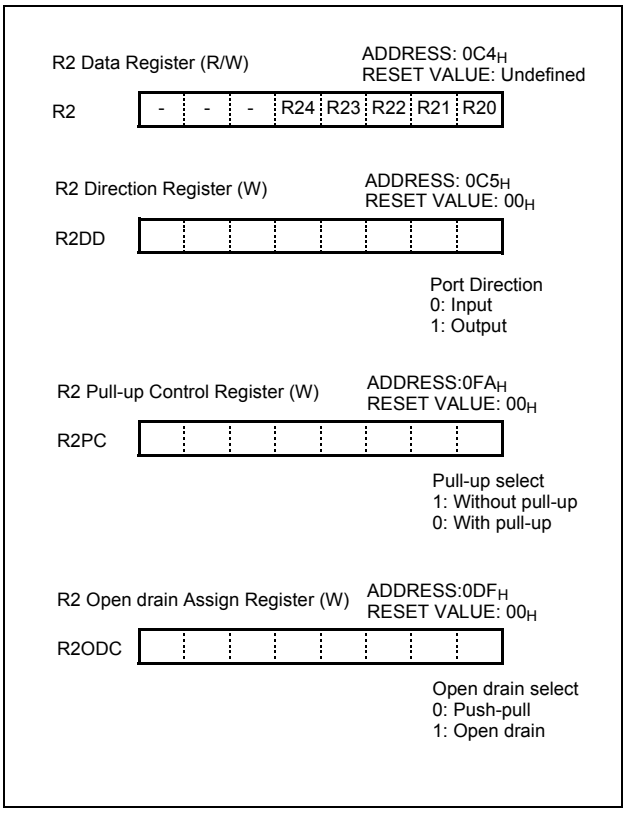
### (5) R1 Pull-up Control Register (R1PC)

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 is 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 5-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 5-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 5-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 5bit 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 5-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 ia 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.

## 9.4 R3 Port

R3 is an 1-bit CMOS bi-directional I/O port(address 0E5H). The I/O pin can independently used as an input or an output through the R3DD register(address 0E6H).R3 has internal pull-up that is independently connected or disconnected by R3PC(address 0FBH). The control registers for R3 are shown as below

### (1) R3 Data Register(R3)

R3 data register(R3) is 1-bit register to store data of port R3. When set as the ouput state by R3DD, and data is written in R3, data is outputted into R3 pin. When set as the input state input state of pin is read. The initial value of R3 is unknown in reset state.

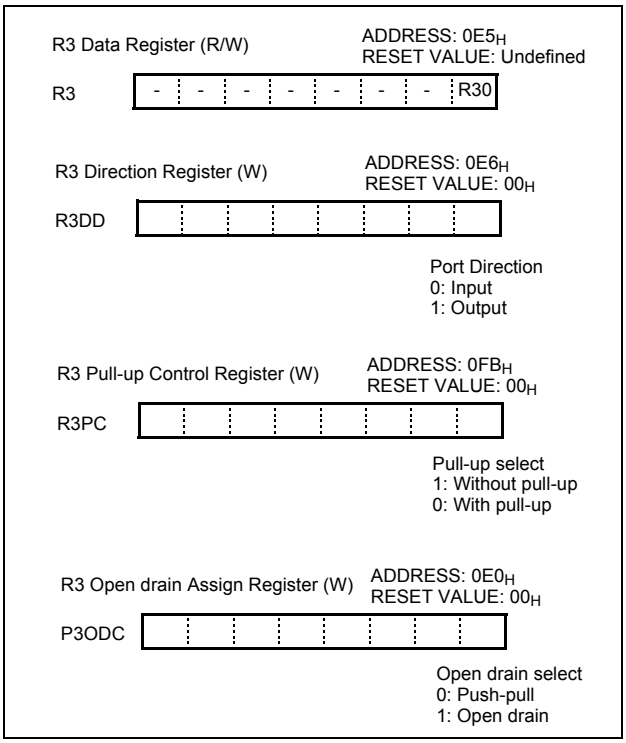
### (2) R3 Data Direction Reigster(R3DD)

R3 I/O Data Direction REgister(R3DD) is 1-bit register, and can assign input state or output state to R3 port bit. If R3DD is "1", port R3 is in the ouput state, and if "0", it is in the input state. R3DD is write-only register. Since R3DD is initialized as "00H" in reset state, the port R3 becomes input state.

### (3) R3 Open Drain Assign Register(R3ODC)

R3 Port Open Drain Assign Register(R3ODC) is 1-bit register, and cna assign R3 port as open drain ouput port, if corresponding port is selected as output. If R3ODC is selected as "1", R3 Port is open drian output, and if selected as "0", it is push-pull output R3ODC is write-only register and initialized as "00H" in reset state.

.




### (4) R3 Pull-up Control Register(R3PC)

R3 Port Pull-up Resistor Control Register(R3PC) is 1-bit register and can control pull-up on or off, if corresponding port is selected as input. If R3PC is selected as “1”, pull-up is disabled and if selected as “0”, it is enabled R3PC is write-only register and initialized as “00H’in reset state. The pull-up is automatically disabled, if corresponding port is selected as output.

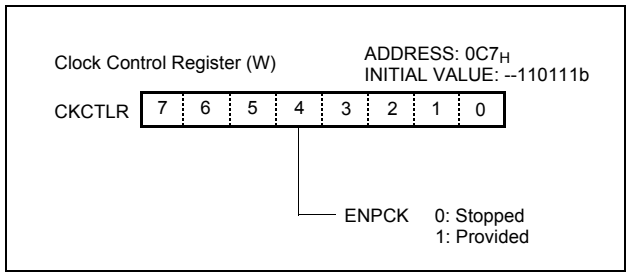
## 10. CLOCK GENERATOR

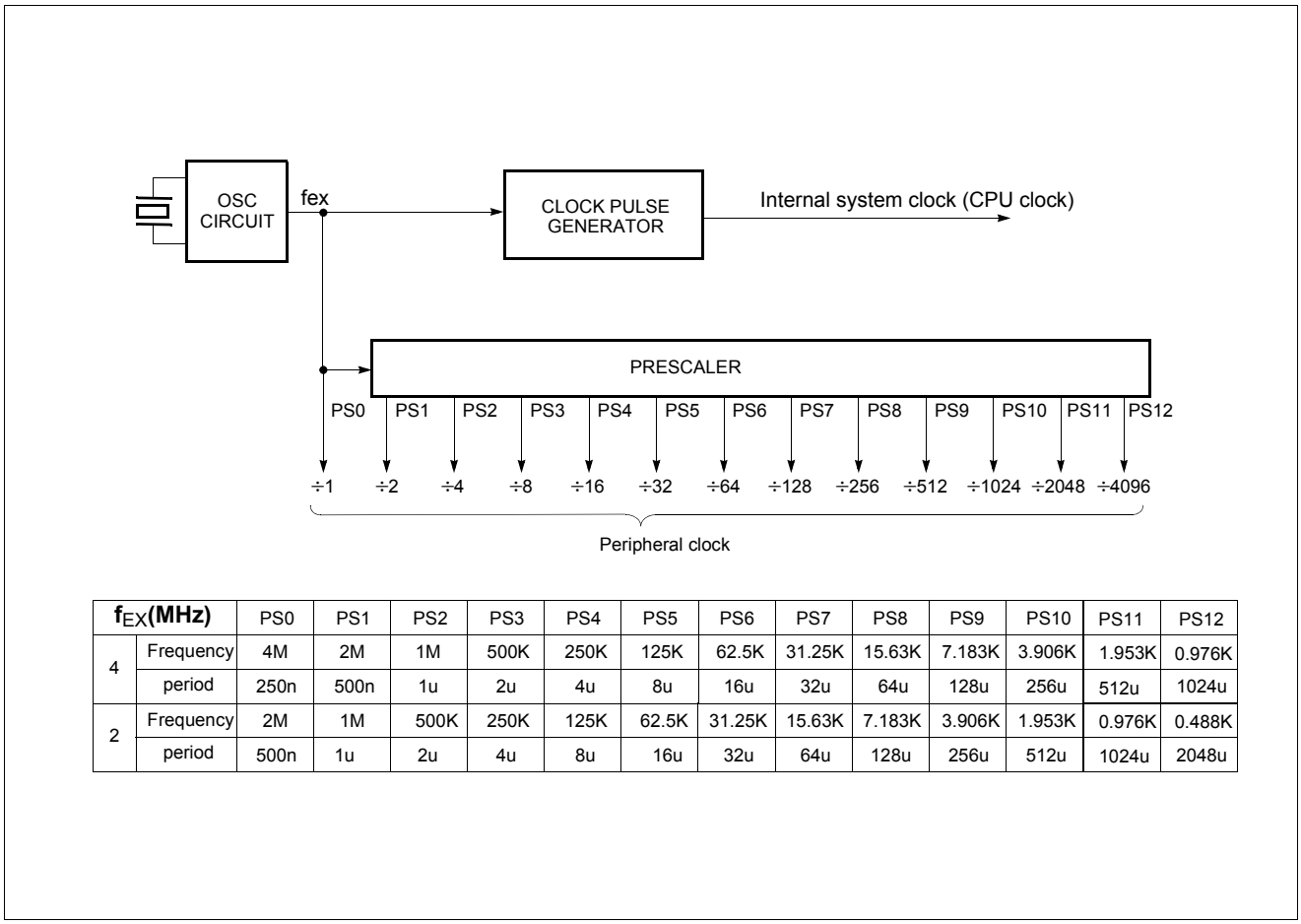
Clock generating circuit consists of Clock Pulse Generator (C.P.G), Prescaler, Basic Interval Timer (B.I.T) and Watch Dog Timer. The clock applied to the Xin pin divided by two is used as the internal system clock.

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 peripheral hardware

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




| $f_{EX}$(MHz) | | PS0 | PS1 | PS2 | PS3 | PS4 | PS5 | PS6 | PS7 | PS8 | PS9 | PS10 | PS11 | PS12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | Frequency | 4M | 2M | 1M | 500K | 250K | 125K | 62.5K | 31.25K | 15.63K | 7.183K | 3.906K | 1.953K | 0.976K |
| | period | 250n | 500n | 1u | 2u | 4u | 8u | 16u | 32u | 64u | 128u | 256u | 512u | 1024u |
| 2 | Frequency | 2M | 1M | 500K | 250K | 125K | 62.5K | 31.25K | 15.63K | 7.183K | 3.906K | 1.953K | 0.976K | 0.488K |
| | period | 500n | 1u | 2u | 4u | 8u | 16u | 32u | 64u | 128u | 256u | 512u | 1024u | 2048u |

**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 a oscillator between Xout and Xin. Clock 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.

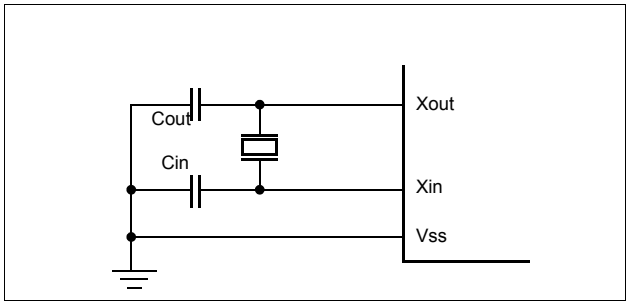



**Figure 10-2 External Crystal(Ceramic) oscillator circuit**

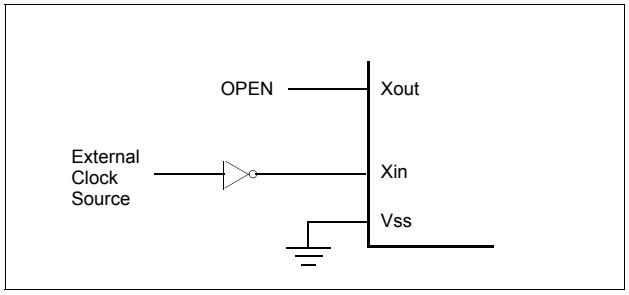



**Figure 10-3 External clock input circuit**

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.*

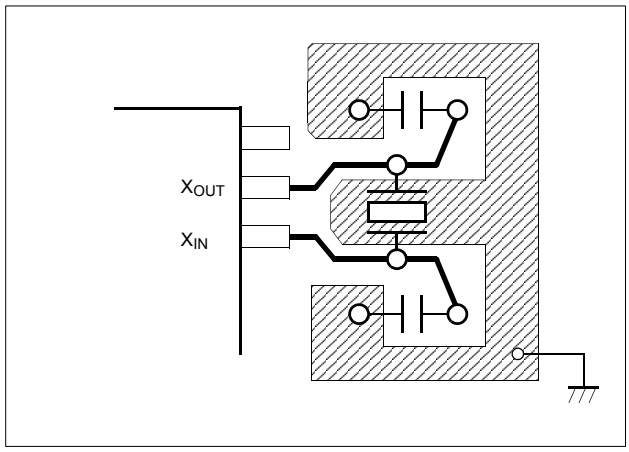



**Figure 10-4 Recommend Layout of Oscillator PCB circuit**

| 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 |
| | CORETECH | CRT4.00MS | Cin=Cout=open | 2.0~3.6 |
| | CORETECH | CRM4.00MS | Cin=Cout=30pF | 2.0~3.6 |

**Table 10-1 recommendatory resonator**

## 11. BASIC INTERVAL TIMER

The MC80F5132/5232 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 2us to 256us by clock input selection bits (BTS2~BTS0). (at fex = 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.

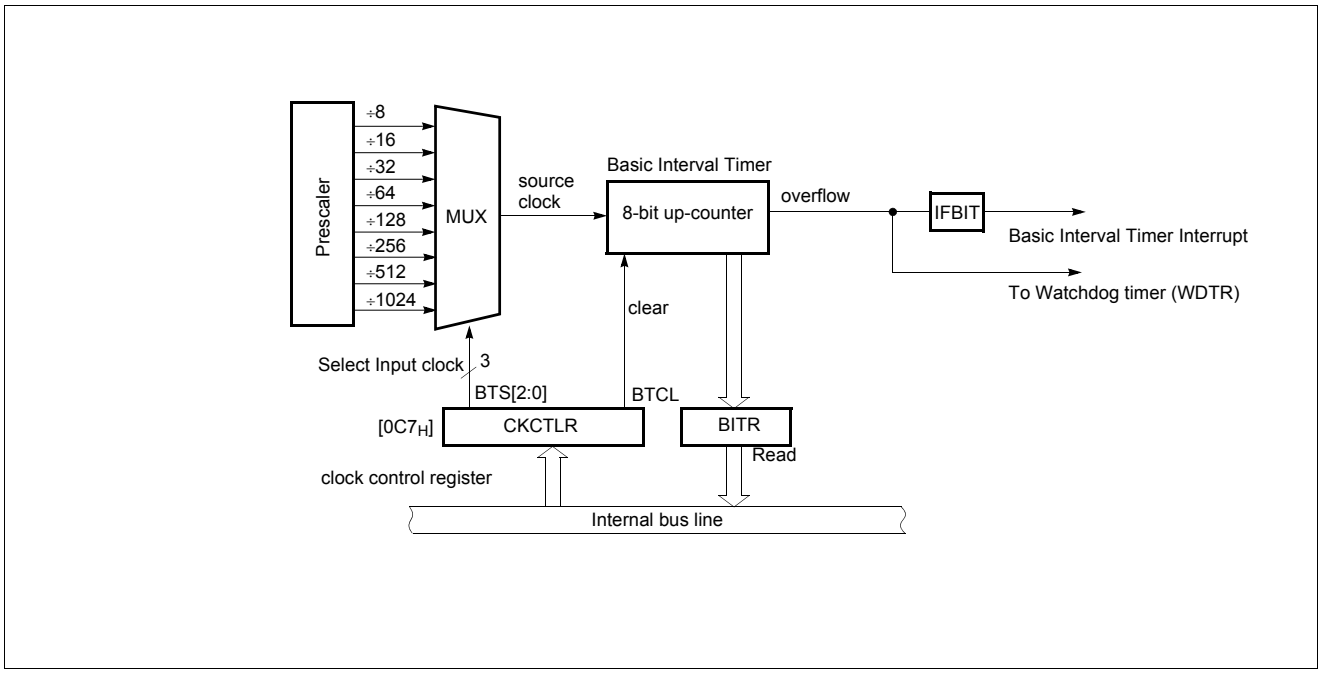



**Figure 11-1 Block diagram of Basic Interval Timer**

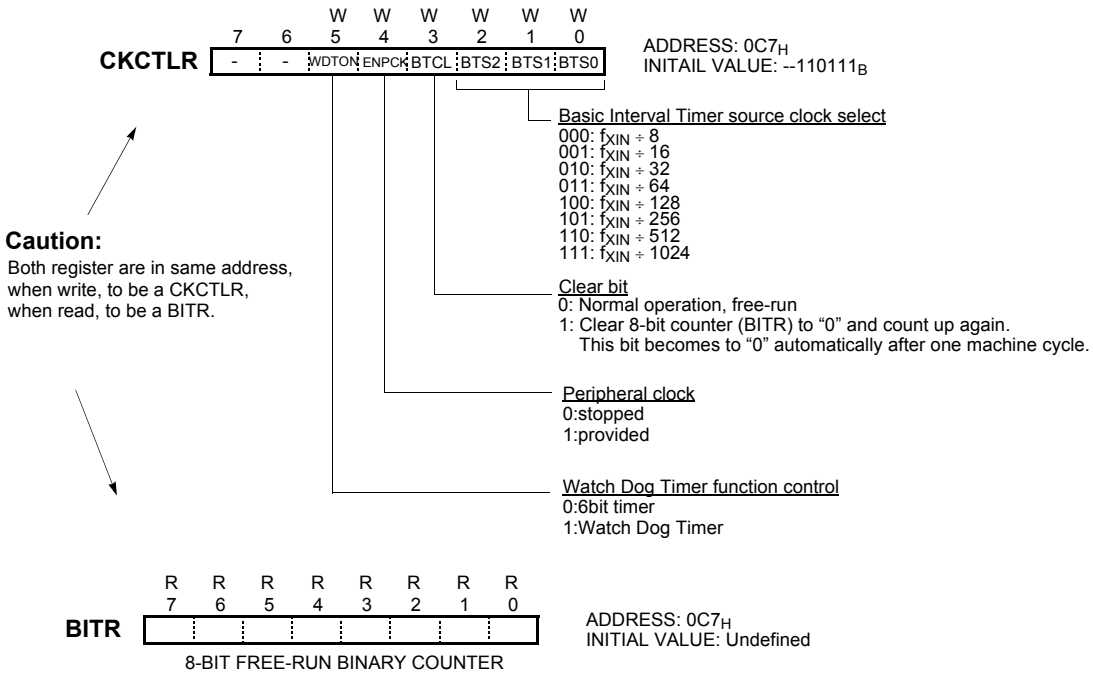



**Figure 11-2 CKCTLR AND BITR**

| 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 (CKCTLR).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 CKCTLR and WDTR before WDT overflow.

-Reset WDTR value = $0F_h$,=15

-Interval of WDT = 65,536 * 15 = 983040 us

(about 1second)

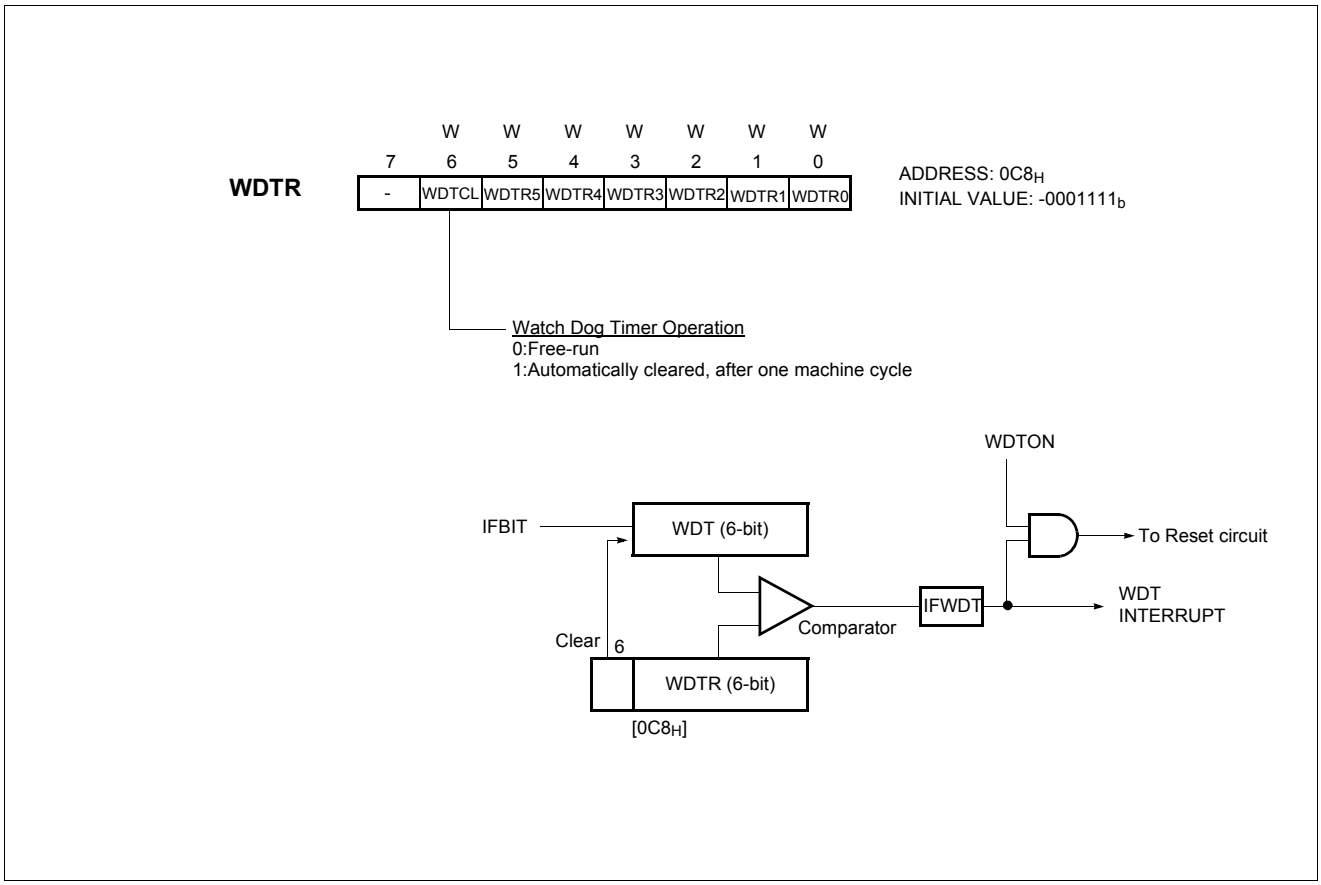



**Figure 12-1 Block diagram of Watch Dog Timer**

***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.

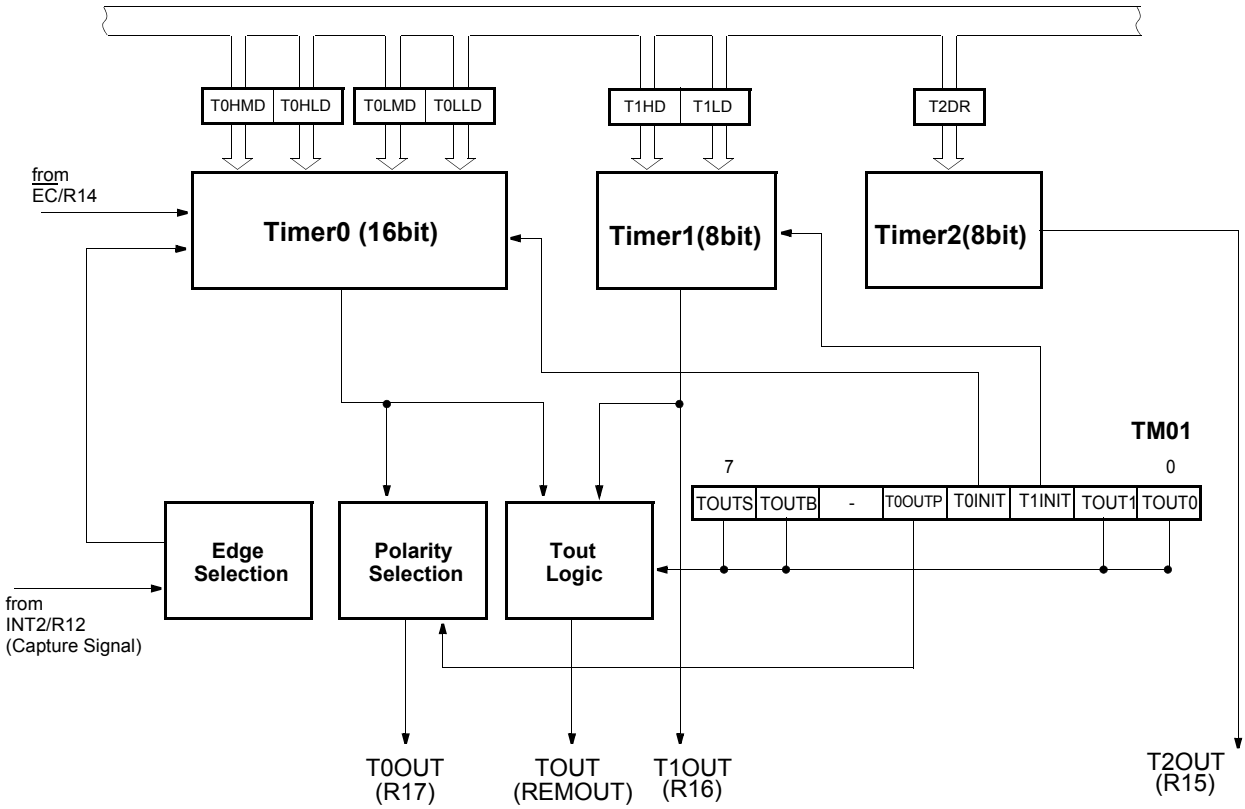
* Relevant Port Mode Register (PMR1: 00C9h) value should be assigned for event counter.

<table>
<tr><td>Timer0</td><td>- 16-bit Interval Timer<br>- 16-bit Event Counter<br>- 16-bit Input Capture<br>- 16-bit rectangular-wave output</td><td rowspan="2">- Single/Modulo-N Mode<br>- Timer Output Initial Value Setting<br>- Timer0~Timer1 combination Logic Output<br>- One Interrupt Generating Every 2nd<br>Counter Overflow</td></tr>
<tr><td>Timer1</td><td>- 8-bit Interval Timer<br>- 8-bit rectangular-wave output</td></tr>
<tr><td>Timer2</td><td colspan="2">- 8-bit Interval Timer<br>- 8-bit rectangular-wave output<br>- Modulo-N Mode</td></tr>
</table>

**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**



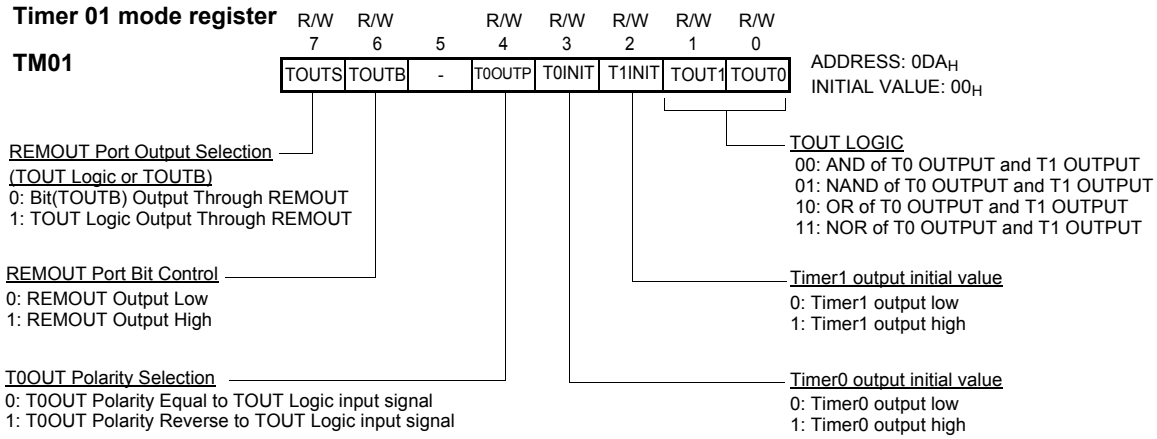
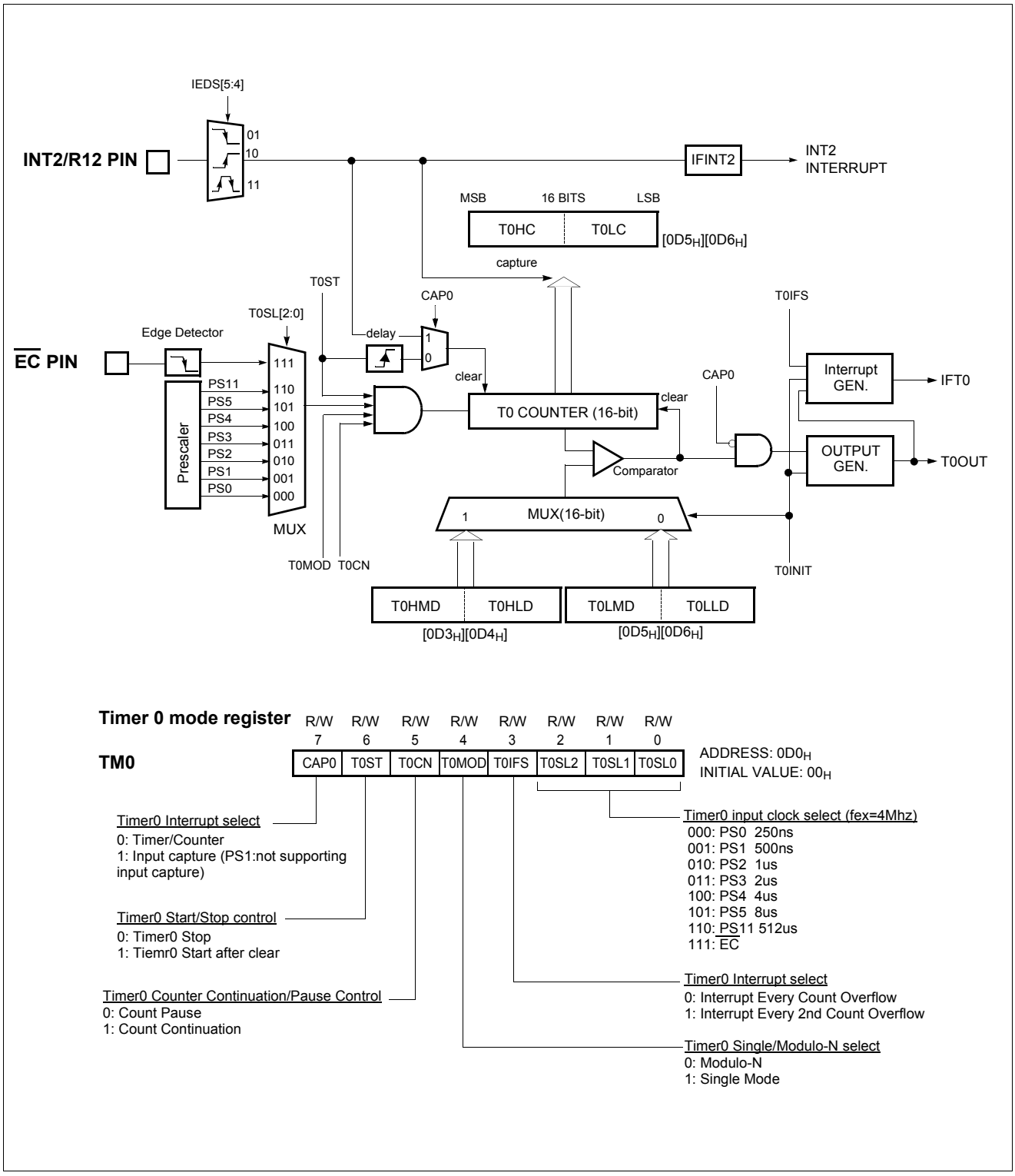




**Figure 13-1 Block Diagram of Timer/Counter**




**Timer 0 mode register**

**TM0**

| R/W 7 | R/W 6 | R/W 5 | R/W 4 | R/W 3 | R/W 2 | R/W 1 | R/W 0 |
|---|---|---|---|---|---|---|---|
| CAP0 | T0ST | T0CN | T0MOD | T0IFS | T0SL2 | T0SL1 | T0SL0 |

ADDRESS: $0D0_H$
INITIAL VALUE: $00_H$

Timer0 Interrupt select
0: Timer/Counter
1: Input capture (PS1:not supporting input capture)

Timer0 Start/Stop control
0: Timer0 Stop
1: Tiemr0 Start after clear

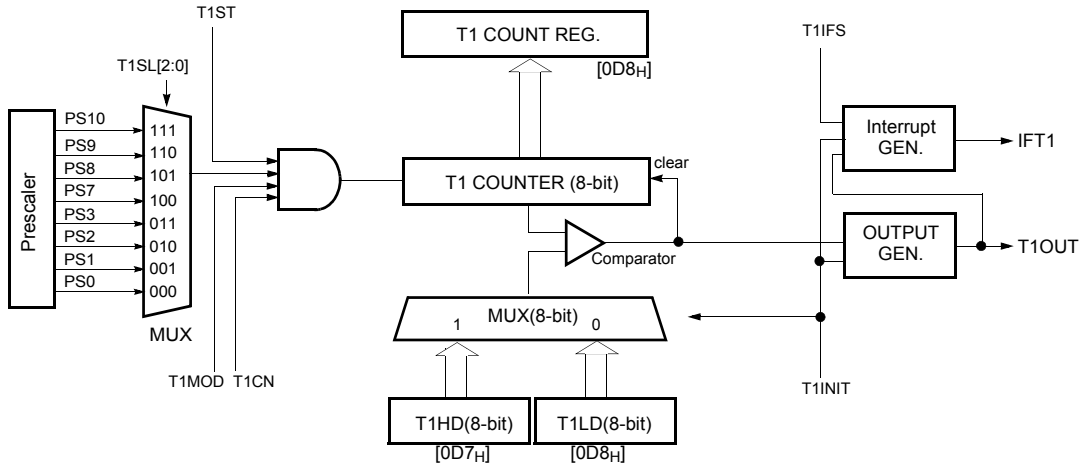
Timer0 Counter Continuation/Pause Control
0: Count Pause
1: Count Continuation

Timer0 input clock select (fex=4Mhz)
000: PS0 250ns
001: PS1 500ns
010: PS2 1us
011: PS3 2us
100: PS4 4us
101: PS5 8us
110: PS11 512us
111: EC

Timer0 Interrupt select
0: Interrupt Every Count Overflow
1: Interrupt Every 2nd Count Overflow

Timer0 Single/Modulo-N select
0: Modulo-N
1: Single Mode

**Figure 13-2 Block Diagram of Timer0**




**Timer 1 mode register**

**TM1**

| R/W 7 | R/W 6 | R/W 5 | R/W 4 | 3 | R/W 2 | R/W 1 | R/W 0 |
|---|---|---|---|---|---|---|---|
| T1ST | T1CN | T1MOD | T1IFS | - | T1SL2 | T1SL1 | T1SL0 |

ADDRESS: $0D1_H$
INITIAL VALUE: $00_H$

Timer1 Start/Stop control
0: Timer1 Stop
1: Tiemr1 Start after clear

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

Timer1 Single/Modulo-N select
0: Modulo-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**

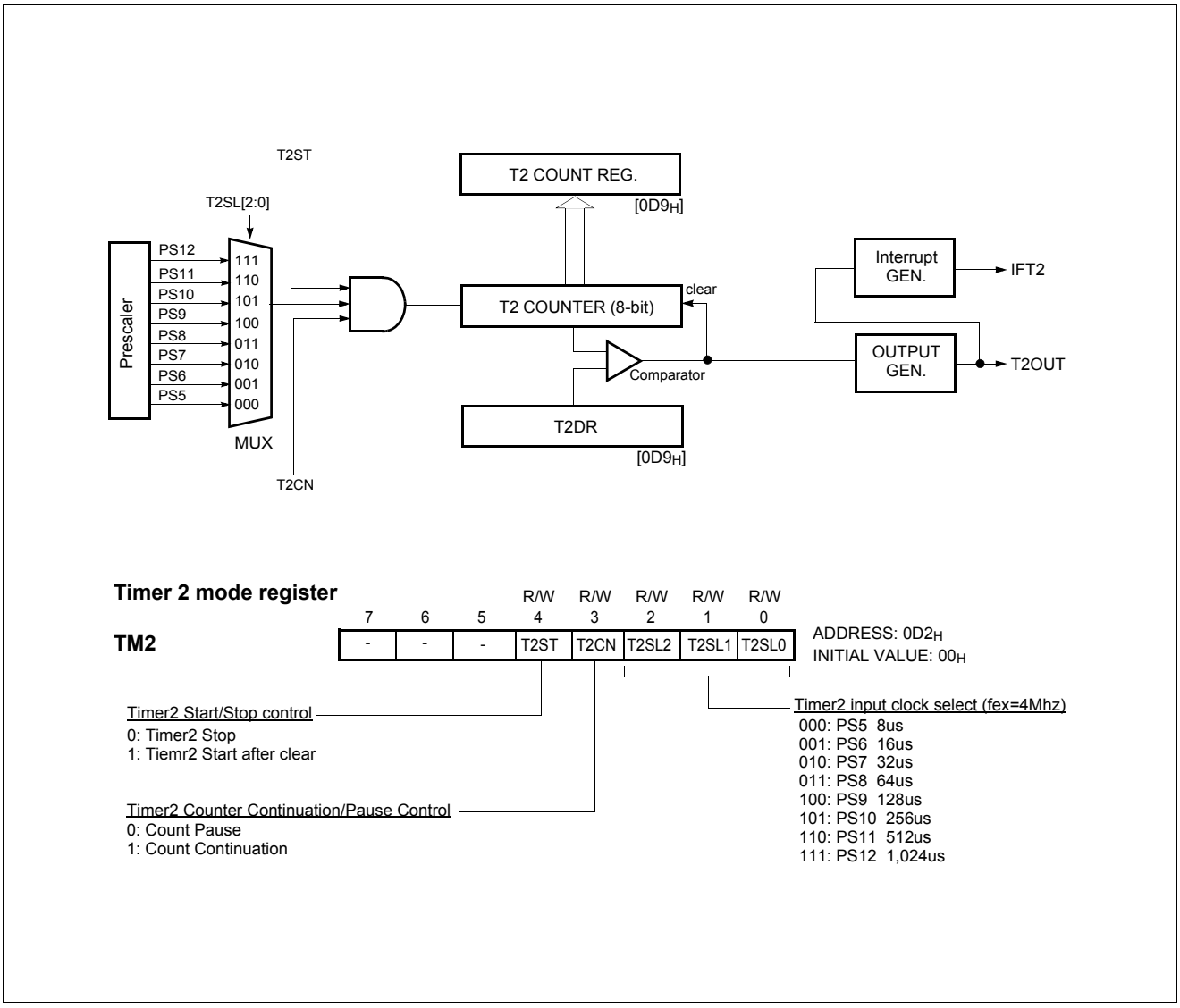



**Figure 13-4 Block Diagram of Timer2**

### 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.

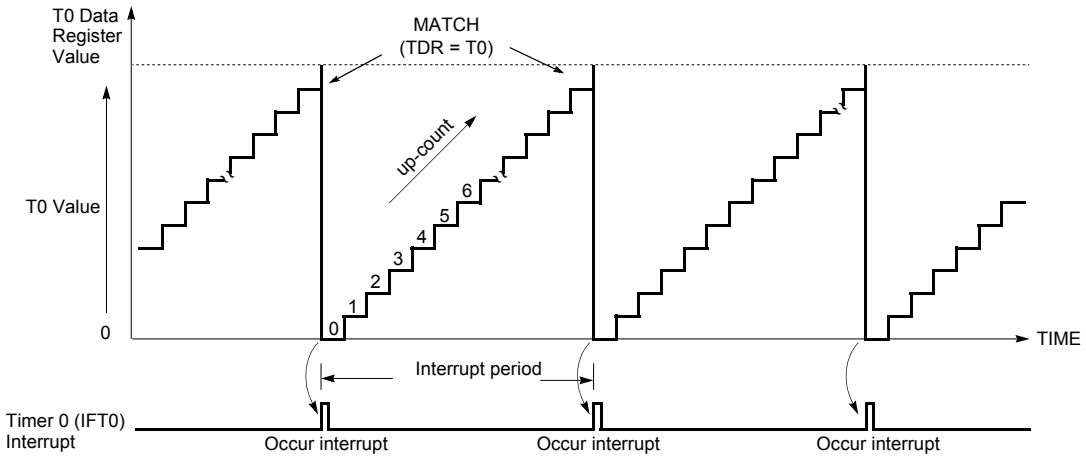



**Figure 13-5 Operation of Timer0**

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 (RESET=”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

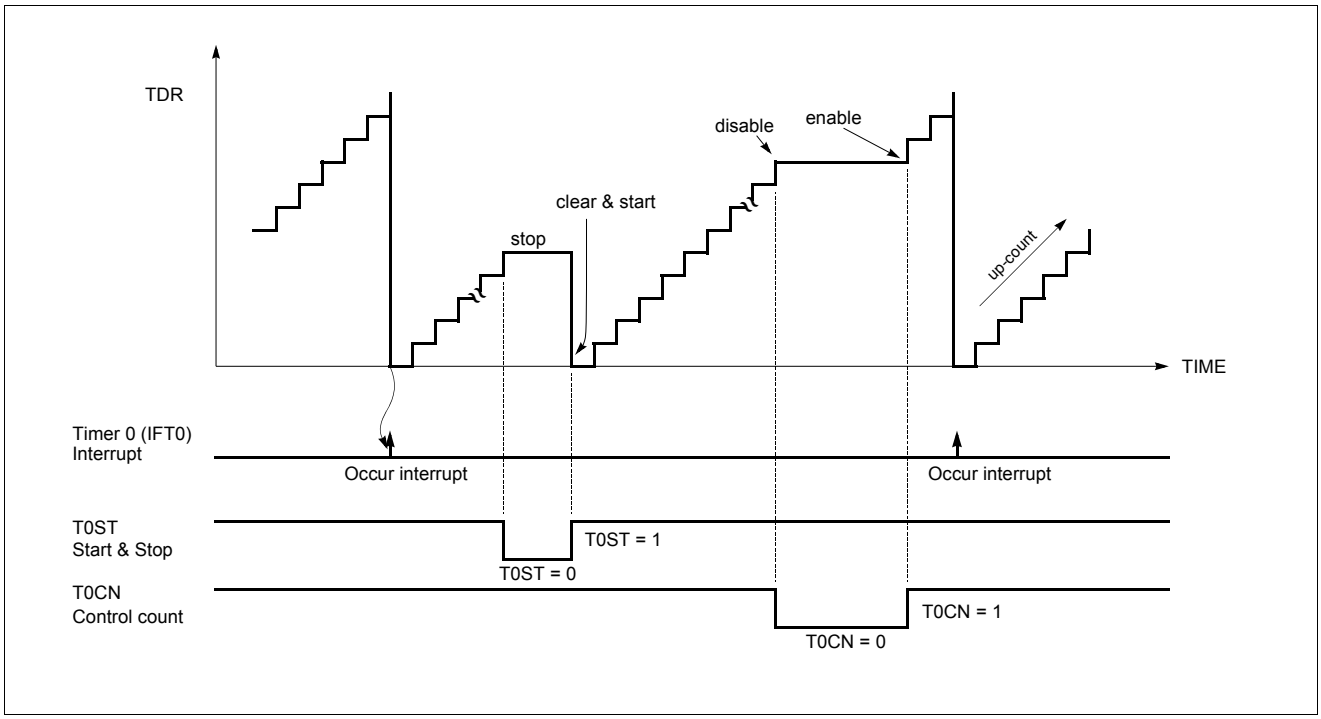



**Figure 13-6 Start/Stop Operation of Timer0**

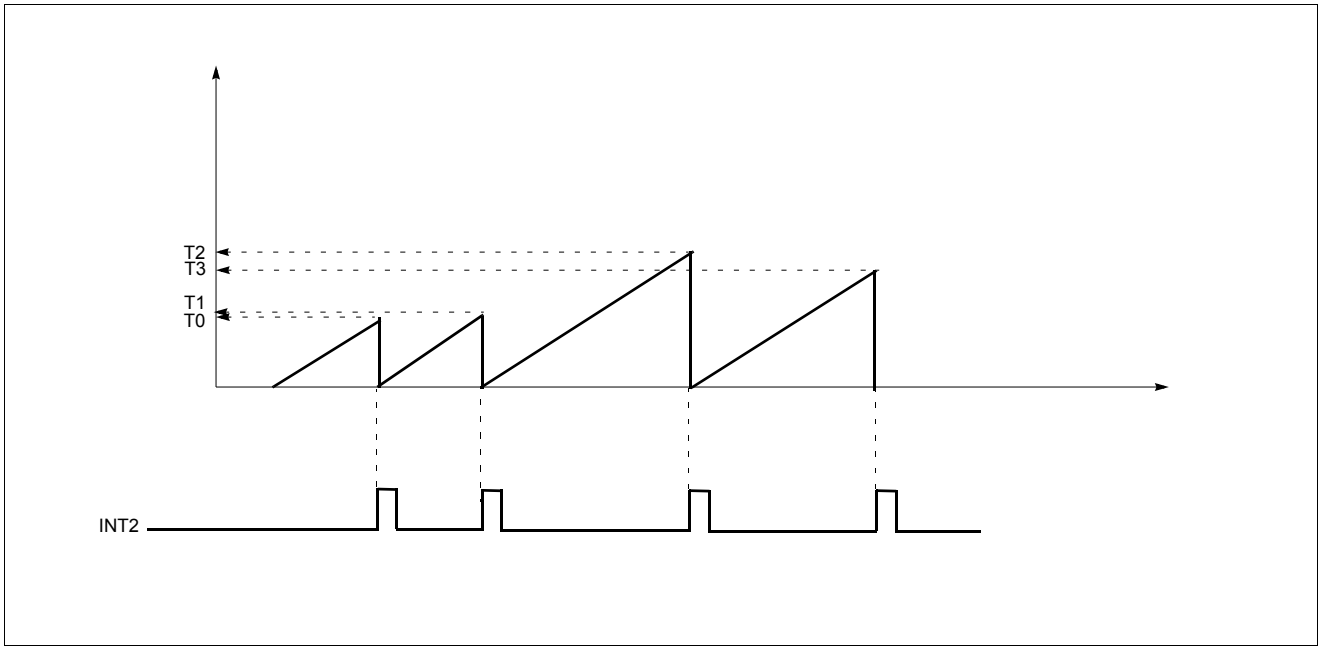



**Figure 13-7 Input capture operation of Timer0**

### 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*

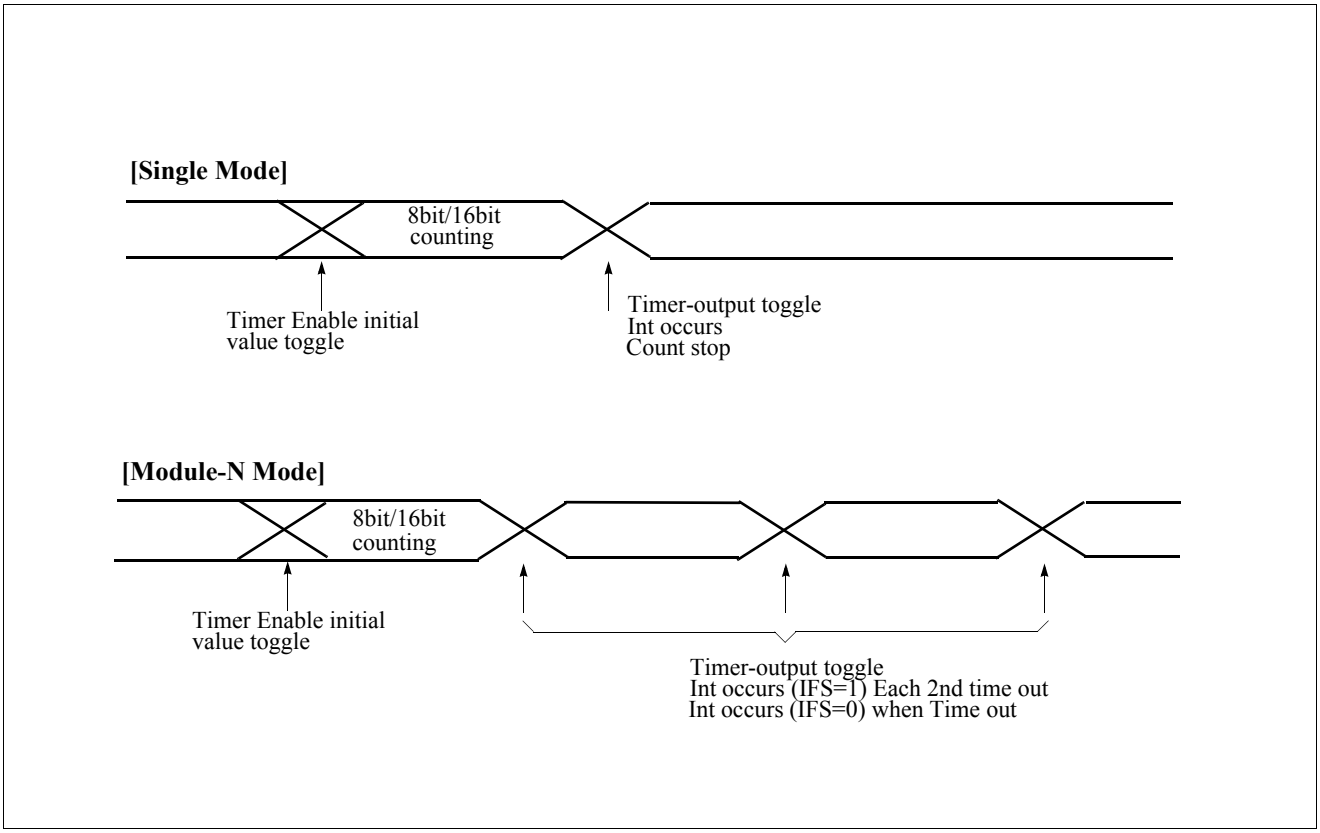



**Figure 13-8 Operation Diagram for Single/Modulo-N Mode**

### (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. UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER (UART)

## 14.1 UART Serial Interface Functions

The Universal Asynchronous Receiver / Transmitter (UART) enables full-duplex operation wherein one byte of data after the start bit is transmitted and received. The on-chip baud rate generator dedicated to UART enables communications using a wide range of selectable baud rates. In addition, a baud rate can also be defined by dividing clocks input to the ACLK pin.

The UART driver consists of RXR, TXR, ASIMR, ASISR and BRGCR register. Universal asynchronous serial I/O mode (UART) can be selected by ASIMR register. Figure 14-1 shows a block diagram of the UART driver.

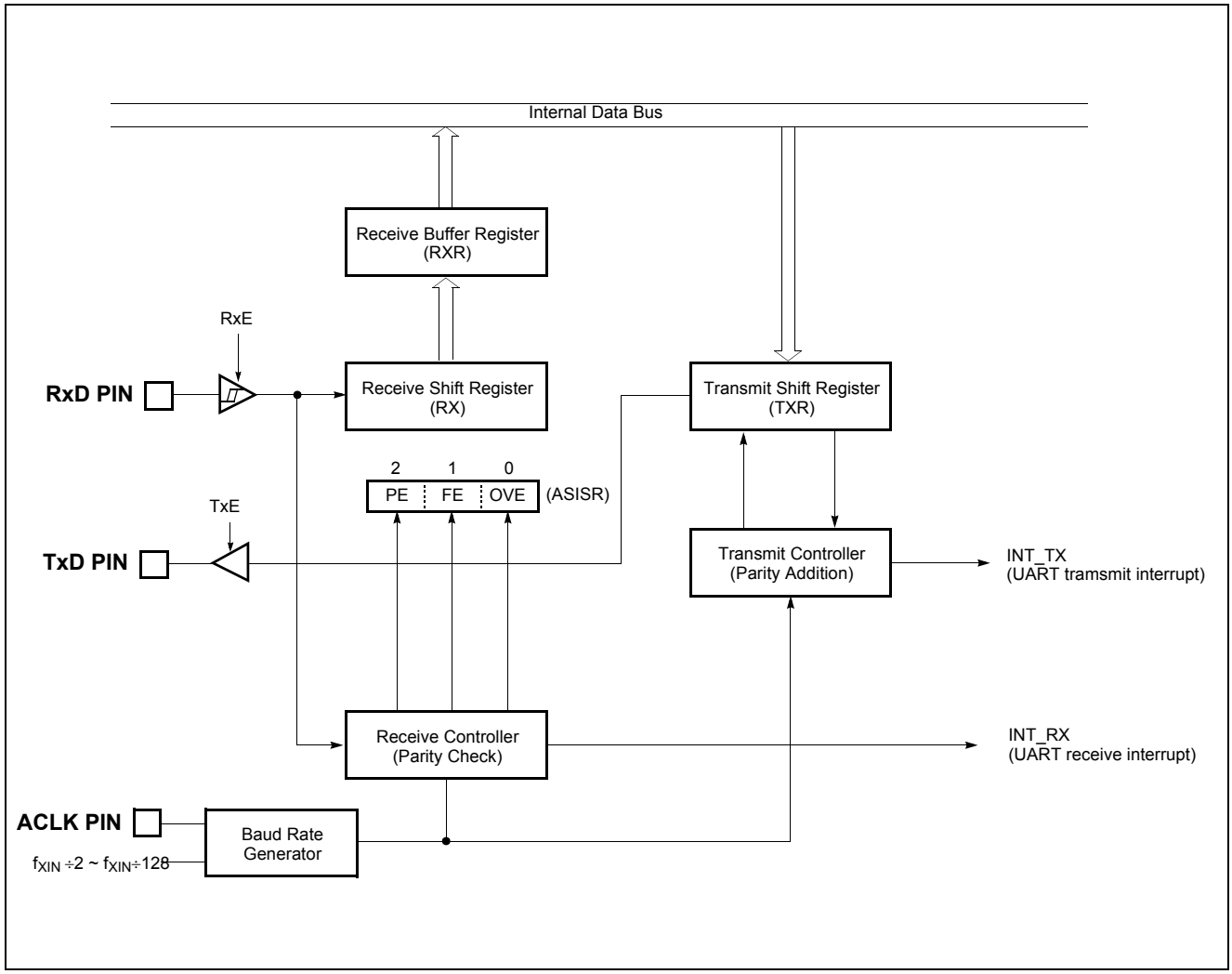



**Figure 14-1 UART Block Diagram**

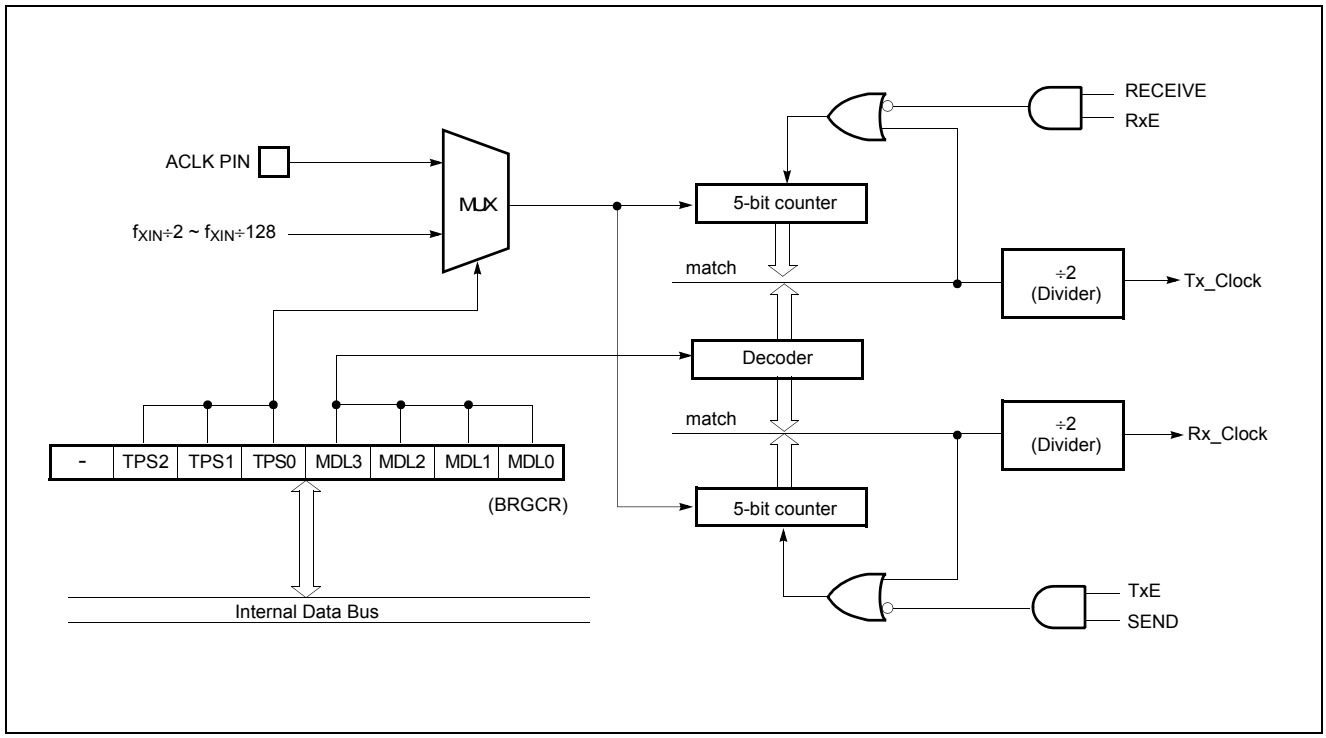



**Figure 14-2 Baud Rate Generator Block Diagram**

## 14.2 Serial Interface Configuration

The UART interface consists of the following hardware.

| Item | Configuration |
|---|---|
| Register | Transmit shift register (TXR)<br>Receive buffer register (RXR)<br>Receive shift register |
| Control register | Serial interface mode register (ASIMR)<br>Serial interface status register (ASISR)<br>Baud rate generator control register (BRGCR) |

**Table 14-1 Serial Interface Configuration**

### Transmit shift register (TXR)

This is the register for setting transmit data. Data written to TXR is transmitted as serial data. When the data length is set as 7 bit, bit 0 to 6 of the data written to TXR are transferred as transmit data. Writing data to TXR starts the transmit operation.

TXR can be written by an 8 bit memory manipulation instruction. It cannot be read. The $\overline{\text{RESET}}$ input sets TXR to $0FF_H$.

### Receive buffer register (RXR)

This register is used to hold receive data. When one byte of data is received, one byte of new receive data is transferred from the receive shift register (RXSR). When the data length is set as 7 bits, receive data is sent to bits 0 to 6 of RXR. In this case, the MSB of RXR always becomes 0.

RXR can be read by an 8 bit memory manipulation instruction. It cannot be written. The $\overline{\text{RESET}}$ input sets RXR to $00_H$.

### Receive shift register

This register converts serial data input via the RXD pin to paralleled data. When one byte of data is received at this register cannot be manipulated directly by a program.

### Asynchronous serial interface mode register (ASIMR)

This is an 8 bit register that controls UART serial transfer operation. ASIMR is set by a 1 bit or 8 bit memory manipulation intruction. The $\overline{\text{RESET}}$ input sets ASIMR to 0000_-00-$_B$. Figure 14-3 shows the format of ASIMR The RXD / R04 and TXD / R05 pin function selection is shown in Table 14-2.

***Note:*** *Do not switch the operation mode until the current serial transmit/receive operation has stopped.*

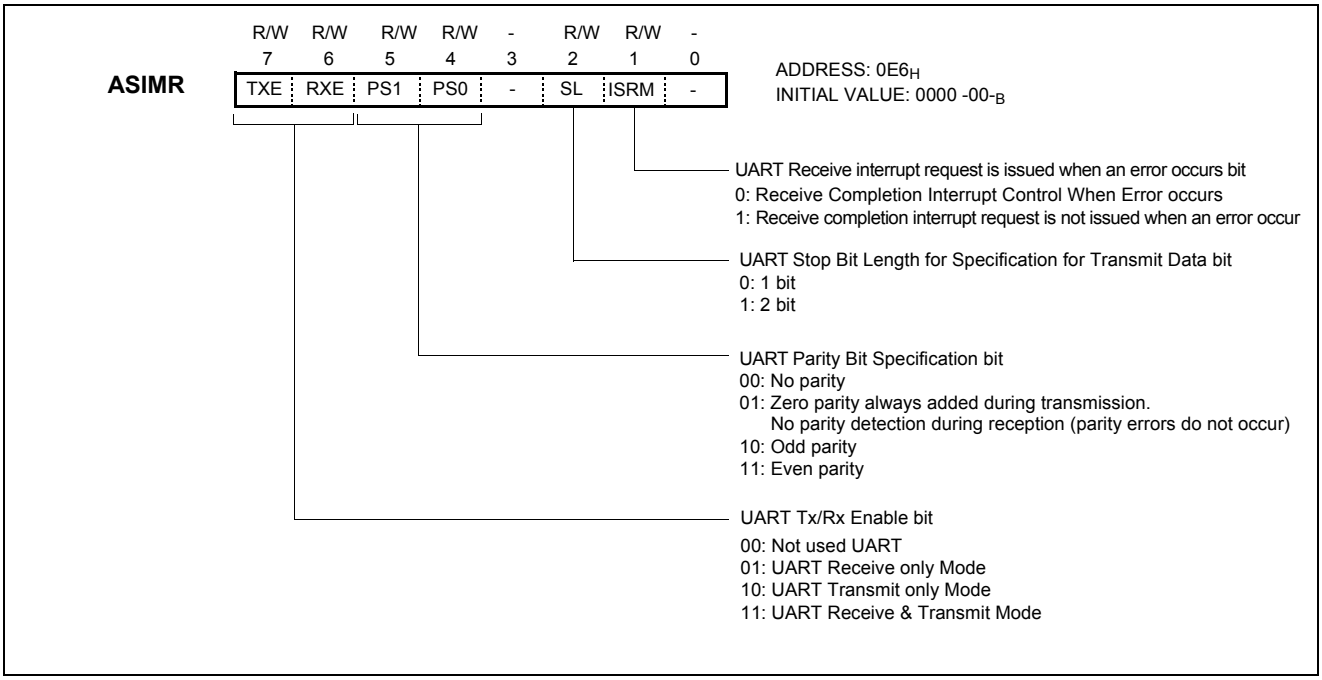



**Figure 14-3 Asynchronous Serial Interface Mode register (ASIMR) Format**

| TXE (ASIMR.7) | RXE(ASIMR.6) | EC0(PSR0.4) | Operation Mode | RXD/R04 | TXD/R05 |
|---|---|---|---|---|---|
| 0 | 0 | X[1] | Operation Stop | R04 | R05 |
| 0 | 1 | 0 | UART mode (Receive only) | RXD | R05 |
| 1 | 0 | X | UART mode (Transmit only) | R04 | TXD |
| 1 | 1 | 0 | UART mode (Transmit and receive) | RXD | TXD |

**Table 14-2 UART mode and RXD/TXD pin function**

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

### Asynchronous serial interface status register (ASISR)

When a receive error occurs during UART mode, this register indicates the type of error. ASISR can be read by an 8 bit memory manipulation instruction. The $\overline{\text{RESET}}$ input sets ASISR to ----_-000B. Figure 14-4 shows the format of ASISR.

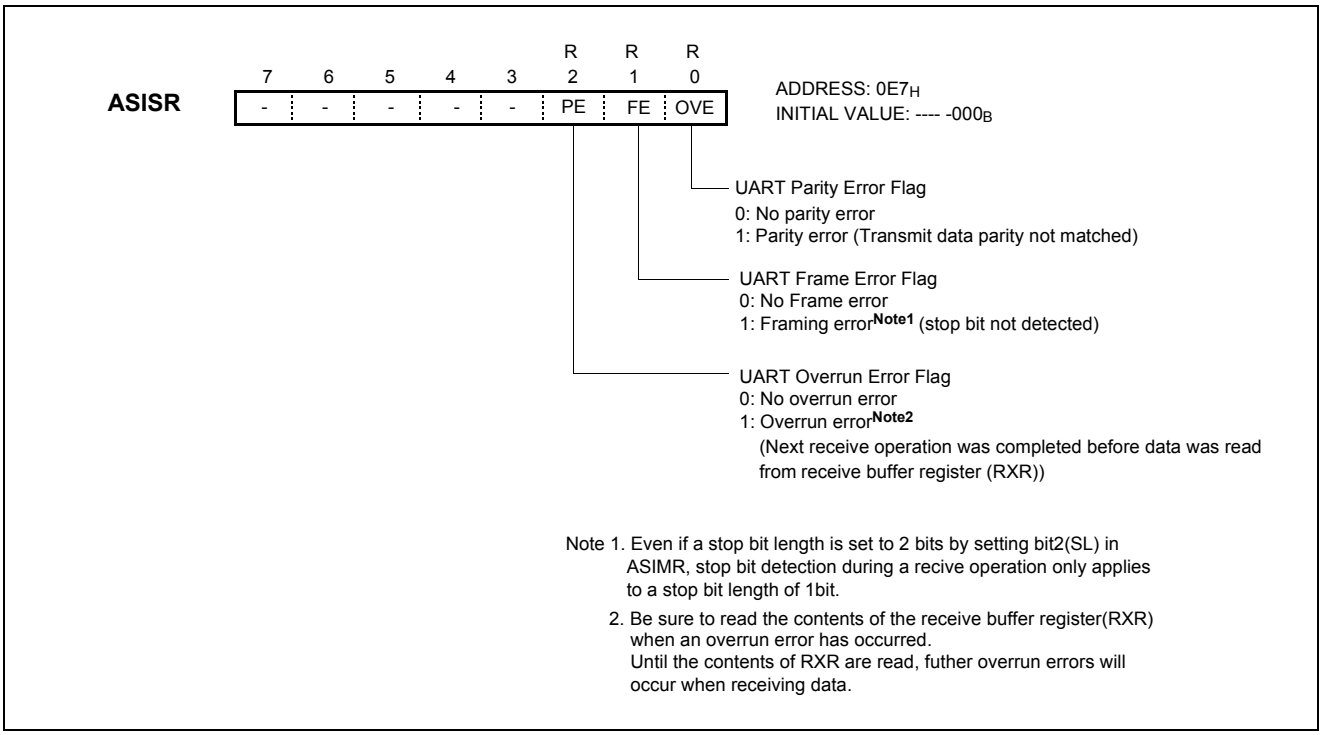



**Figure 14-4 Asynchronous Serial Interface Status Register (ASISR) Format**

### Baud rate generator control register (BRGCR)

This register sets the serial clock for serial interface. BRGCR is set by an 8 bit memory manipulation instruction. The $\overline{\text{RESET}}$ input sets BRGCR to -001_0000B.

Figure 14-5 shows the format of BRGCR.

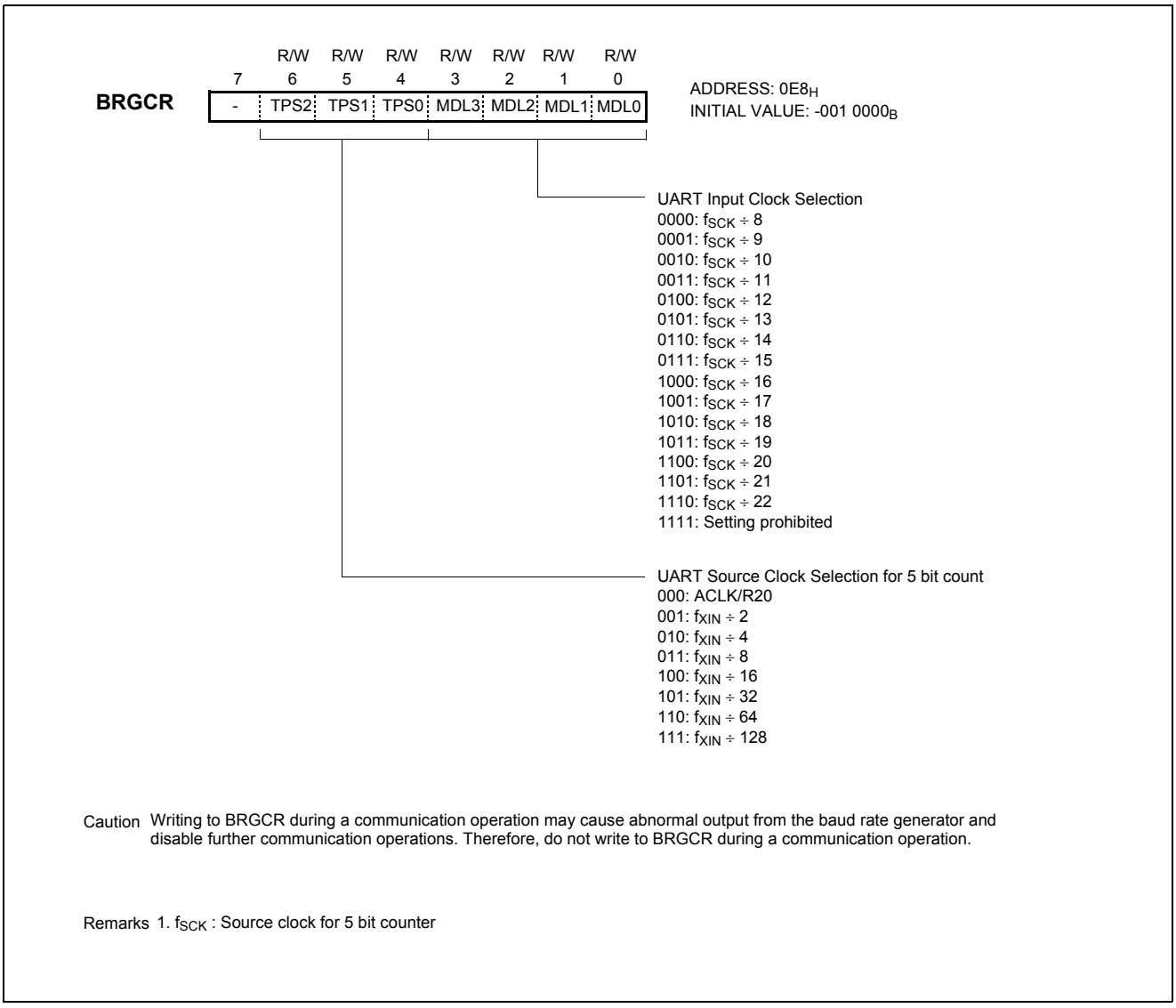


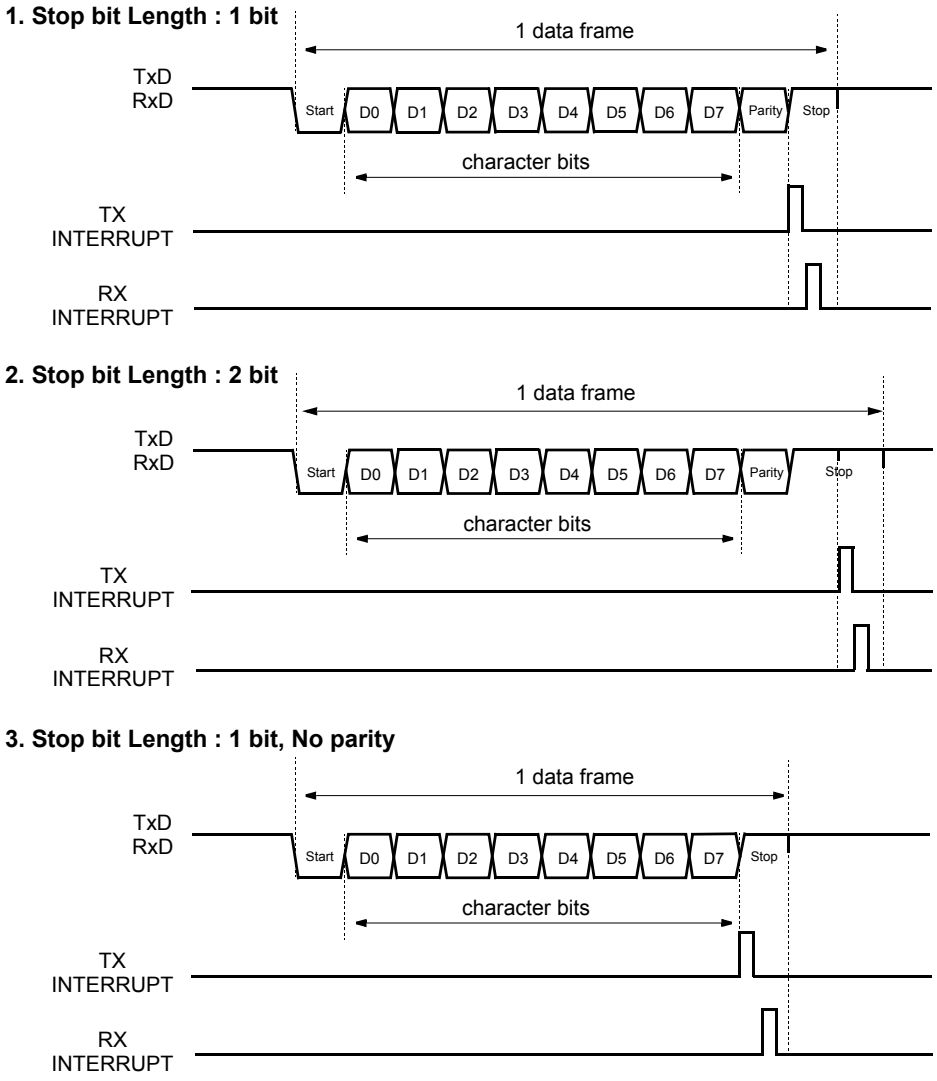

**Figure 14-5 Baud Rate Generator Control Register (BRGCR) Format**

## 14.3 Communication operation

The transmit operation is enabled when bit 7 (TXE) of the asynchronous serial interface mode register (ASIMR) is set to 1. The transmit operation is started when transmit data is written to the transmit shift register (TXR). The timing of the transmit completion interrupt request is shown in Figure 14-6 .

The receive operation is enabled when bit 6 (RXE) of the asynchronous serial interface mode register (ASIMR) is set to 1, and input via the RxD pin is sampled. The serial clock specified by ASIMR is used to sample the RxD pin.

Once reception of one data frame is completed, a receive completion interrupt request (INT_RX) occurs. Even if an error has occurred, the receive data in which the error occurred is still transferred to RXR. When ASIMR bit 1 (ISRM) is cleared to 0 upon occurrence of an error, and INT_RX occurs. When ISRM bit is set to 1, INT_RX does not occur in case of error occurrence. Figure 14-6 shows the timing of the asynchronous serial interface receive completion interrupt request.




1 data frame consists of following bits.
- Start bit : 1 bit
- Character bits : 8 bits
- Parity bit : Even parity, Odd parity, Zero parity, No parity
- Stop bit(s) : 1 bit or 2 bits

**Figure 14-6 UART data format and interrupt timing diagram**

## 14.4 Relationship between main clock and baud rate

The transmit/receive clock that is used to generate the baud rate is obtained by dividing the main system clock. Transmit/Receive clock generation for baud rate is made by using main system clock which is divided. The baud rate generated from the main system clock is determined according to the following formula.

| Baud Rate (bps) | $f_{XIN}$=11.0592M | | $f_{XIN}$=10.5984M | | $f_{XIN}$=8.0M | | $f_{XIN}$=5.2992M | | $f_{XIN}$=4.0M | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BRGCR | ERR(%) | BRGCR | ERR(%) | BRGCR | ERR(%) | BRGCR | ERR (%) | BRGCR | ERR(%) |
| 1200 | - | - | - | - | - | - | 79H | 1.47 | 7DH | 0.16 |
| 2400 | 7AH | 0.00 | 79H | 1.47 | 7DH | 0.16 | 69H | 1.47 | 6DH | 0.16 |
| 4800 | 6AH | 0.00 | 69H | 1.47 | 6DH | 0.16 | 59H | 1.47 | 5DH | 0.16 |
| 9600 | 5AH | 0.00 | 59H | 1.47 | 5DH | 0.16 | 49H | 1.47 | 4DH | 0.16 |
| 19200 | 4AH | 0.00 | 49H | 1.47 | 4DH | 0.16 | 39H | 1.47 | 3DH | 0.16 |
| 31250 | 3EH | 0.54 | 3DH | 0,93 | 40H | 0.00 | 2DH | 0.93 | 30H | 0.00 |
| 38400 | 3AH | 0.00 | 39H | 1.47 | 3DH | 0.16 | 29H | 1.47 | 2DH | 0.16 |
| 57600 | 34H | 0.00 | 34H | 4.16 | 29H | 2.11 | 24H | 4.16 | 19H | 2.12 |
| 76800 | 2AH | 0.00 | 29H | 1.47 | 25H | 0.16 | 19H | 1.47 | 15H | 0.16 |
| 115200 | 24H | 0.00 | 24H | 4.16 | 19H | 2.12 | 14H | 4.16 | 09H | 2.12 |

$$\text{Baud Rate} = f_{XIN} \div ( 2^{n+1}(k+16) )$$

Remarks 1. $f_{XIN}$ : Main system clock oscillation frequency
When ACLK is selected as the source clock of the 5-bit counter, substitute the input clock frequency to ACLK pin for in the above expression.

2. $f_{SCK}$ : Source clock for 5 bit counter
3. n : Value set via TPS0 to TPS2 ( $0 \le n \le 7$ )
4. k : Source clock for 5 bit counter ( $0 \le k \le 14$ )

**Table 14-1 Relationship between main clock and baud rate**

## 15. INTERRUPT

The MC80F5132/5232 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). 10 interrupt sources are provided. The configuration of interrupt circuit is shown in Figure 14-1.

The MC80F5132/5232 contains 10 interrupt sources; 3 externals and 7 internals. Nested interrupt services with priority control is also possible. Software interrupt is non-maskable interrupt, the others are all maskable interrupts.

- 10 interrupt source (2Ext, 3Timer, BIT, WDT, 2UART and Key Scan)
- 10 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.

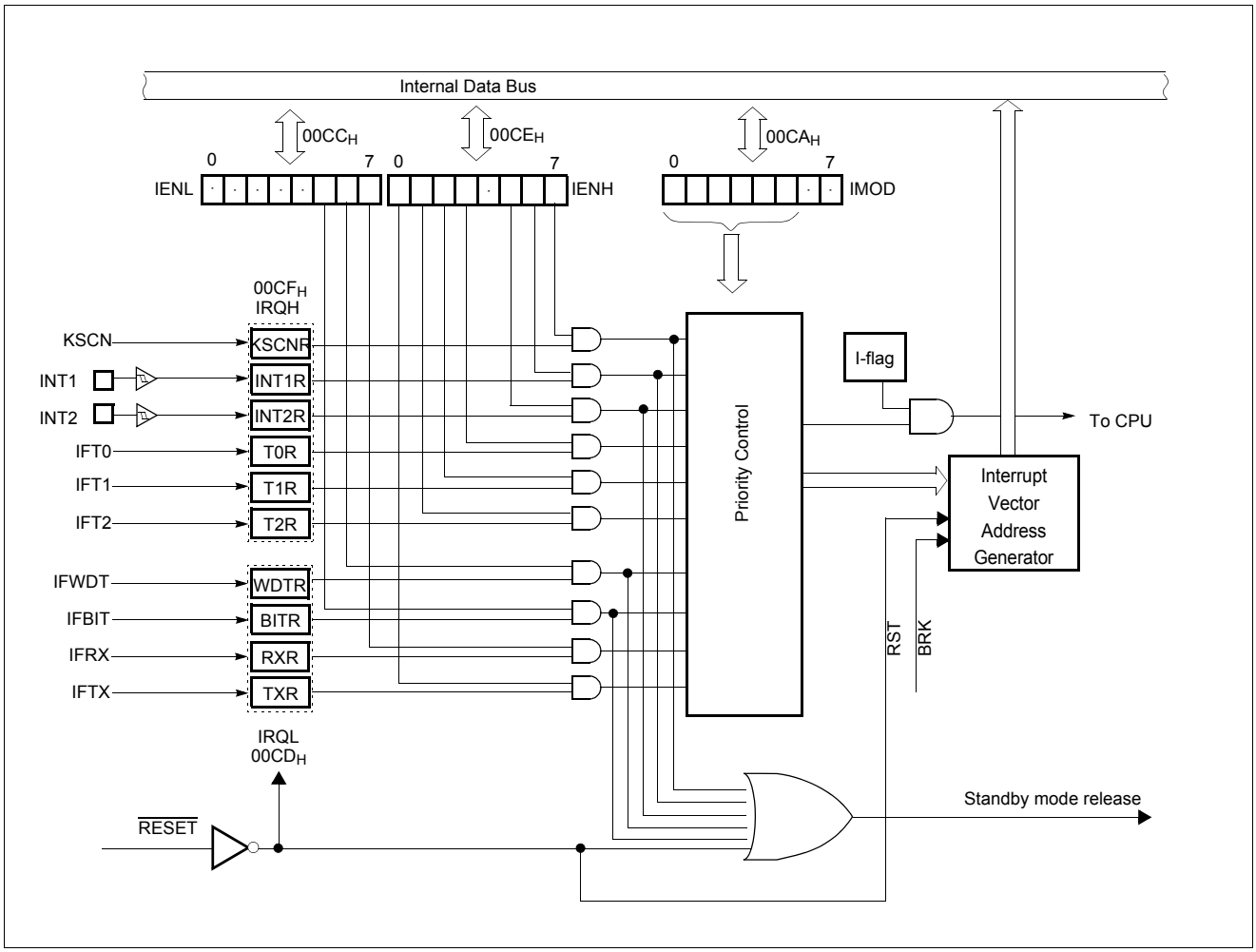



**Figure 15-1 Block Diagram of Interrupt**

## 15.1 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.

## 15.2 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)

| | Mask | Priority | Reset/Interrupt | Symbol | INT Vector H | INT Vector L |
|---|---|---|---|---|---|---|
| Hardware Interrupt | Non-maskable | 1 | Hardware Reset | $\overline{\text{RESET}}$ | FFFF | FFFE |
| | maskable | 2 | Key Scan | KSCNR | FFFB | FFFA |
| | | 3 | External Interrupt1 | INT1R | FFF9 | FFF8 |
| | | 4 | External Interrupt2 | INT2R | FFF7 | FFF6 |
| | | 5 | Timer0 | T0R | FFF3 | FFF2 |
| | | 6 | Timer1 | T1R | FFF1 | FFF0 |
| | | 7 | Timer2 | T2R | FFEF | FFEE |
| | | 8 | UART Rx | RXR | FFED | FFEC |
| | | 9 | UART Tx | TXR | FFEB | FFEA |
| | | 10 | Watch Dog Timer | WDTR | FFE9 | FFE8 |
| | | 11 | Basic Interval Timer | BITR | FFE7 | FFE6 |
| Software Interrupt | - | - | BRK Instruction | BRK | FFDF | FFDE |

**Table 15-1 Interrupt Source**

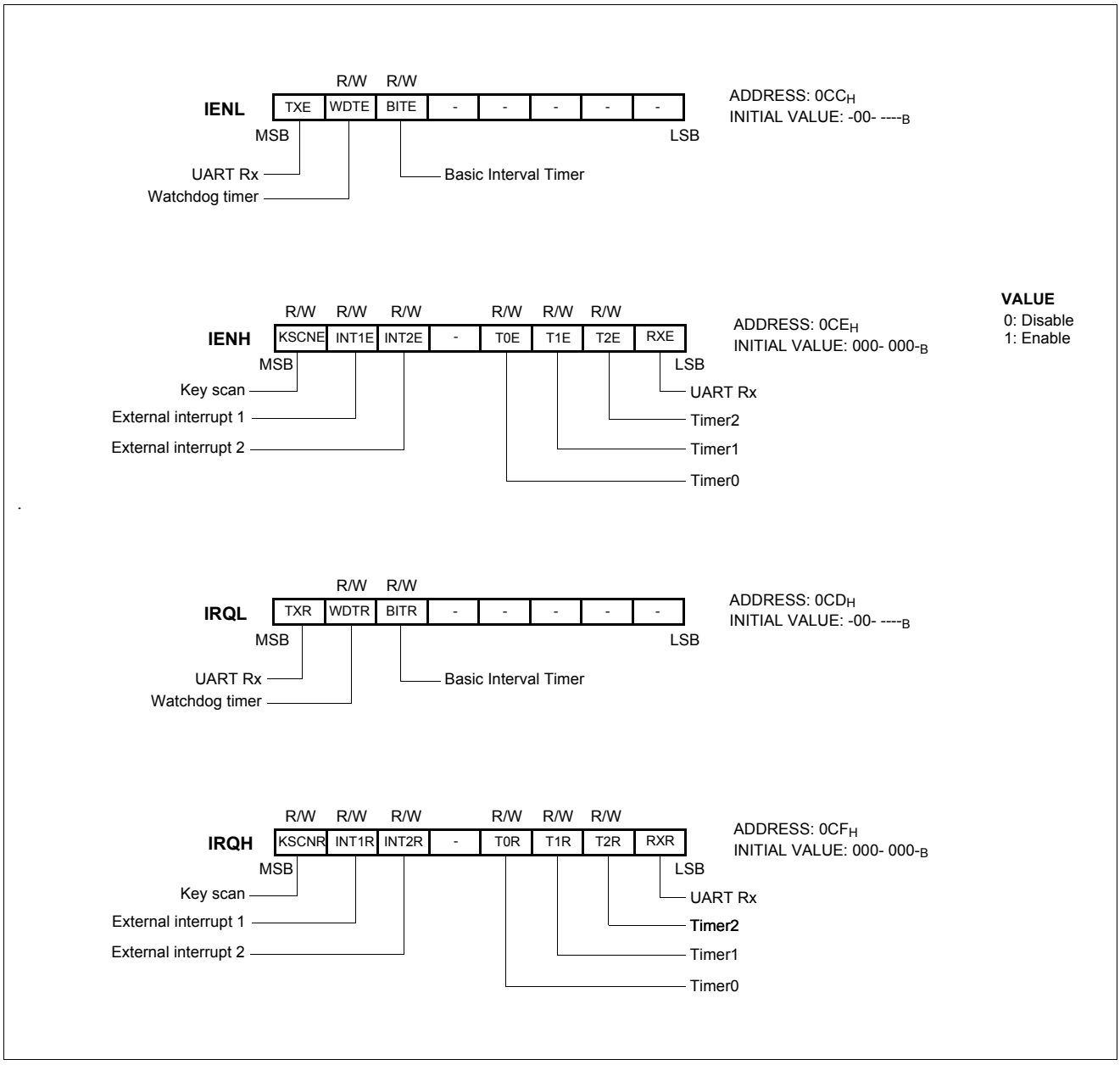



**Figure 15-2 Interrupt Enable & Request Flag**

## 15.3 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”.

l

**Interrupt mode register**
**IMOD**

| R/W 7 | R/W 6 | R/W 5 | R/W 4 | R/W 3 | R/W 2 | R/W 1 | R/W 0 |
|---|---|---|---|---|---|---|---|
| - | - | IM1 | IM0 | IP3 | IP2 | IP1 | IP0 |

ADDRESS: $0CA_H$
INITIAL VALUE: $--000000_B$

Priority
00: Fixed by hardware
01: Changeable by IP3~IP0
1x: Interrupt is inhibited

Selection interrupt
0001: KSCNR
0010: INT1R
0011: INT2R
0101: T0R
0110: T1R
0111: T2R
1000: RXR
1001: TXR
1010: WDTR
1011: BITR

**Figure 15-3 Interrupt Accept Mode & Selection by IP3~IP0**

## 15.4 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.

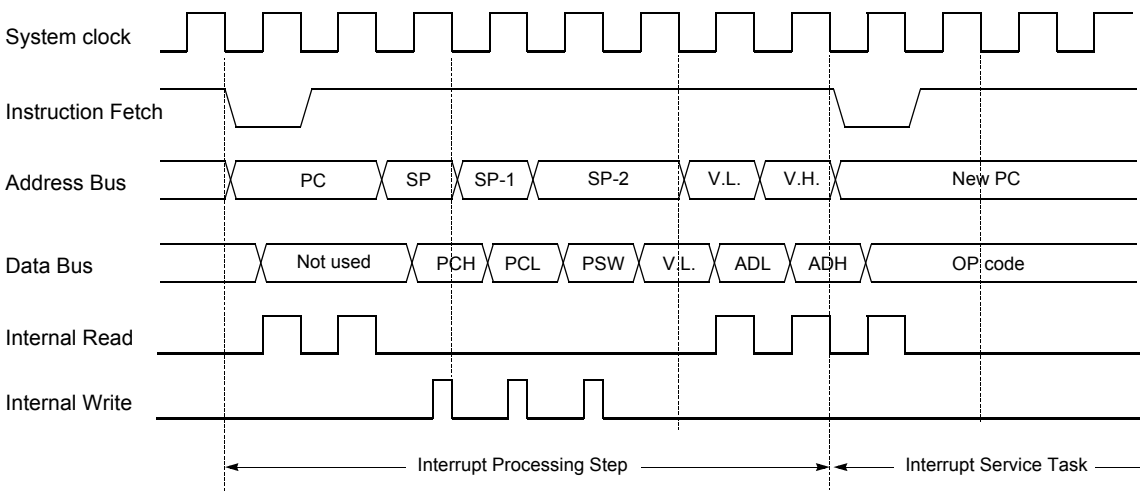
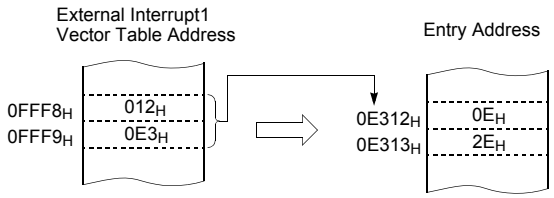



**Figure 15-4 Timing chart of Interrupt Acceptance and Interrupt Return Instruction**




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

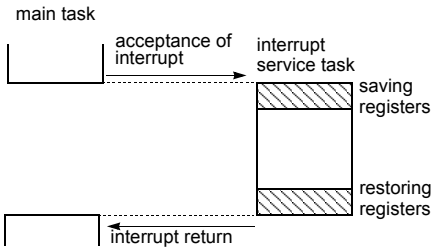
During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but accumulator and other registers are not saved itself. These registers are saved by the software if necessary. Also, when multiple interrupt services are nested, it is necessary 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            ;RETURN
```

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




## 15.5 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 15-5

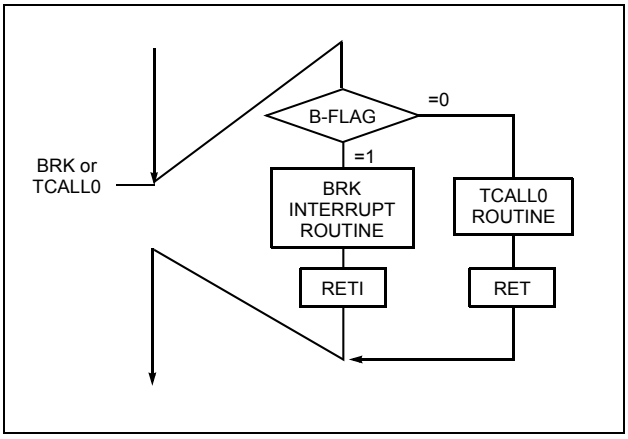



**Figure 15-5 Execution of BRK/TCALL0**

## 15.6 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.

## 15.7 External Interrupt

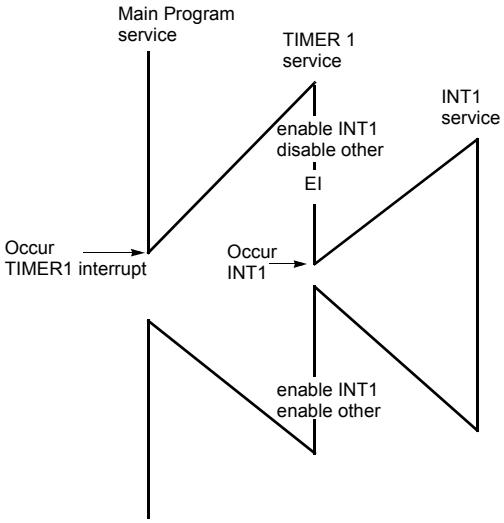
The external interrupt on INT1 and INT2 pins are edge triggered depending on the edge selection register IEDS (address $0D8_H$) as shown in Figure14-7.

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
```




In this example, the INT1 interrupt can be serviced without any pending, even TIMER1 is in progress.
Because of re-setting the interrupt enable registers IENH,IENL and master enable “EI” in the TIMER1 routine.

**Figure 15-6 Execution of Multi Interrupt**

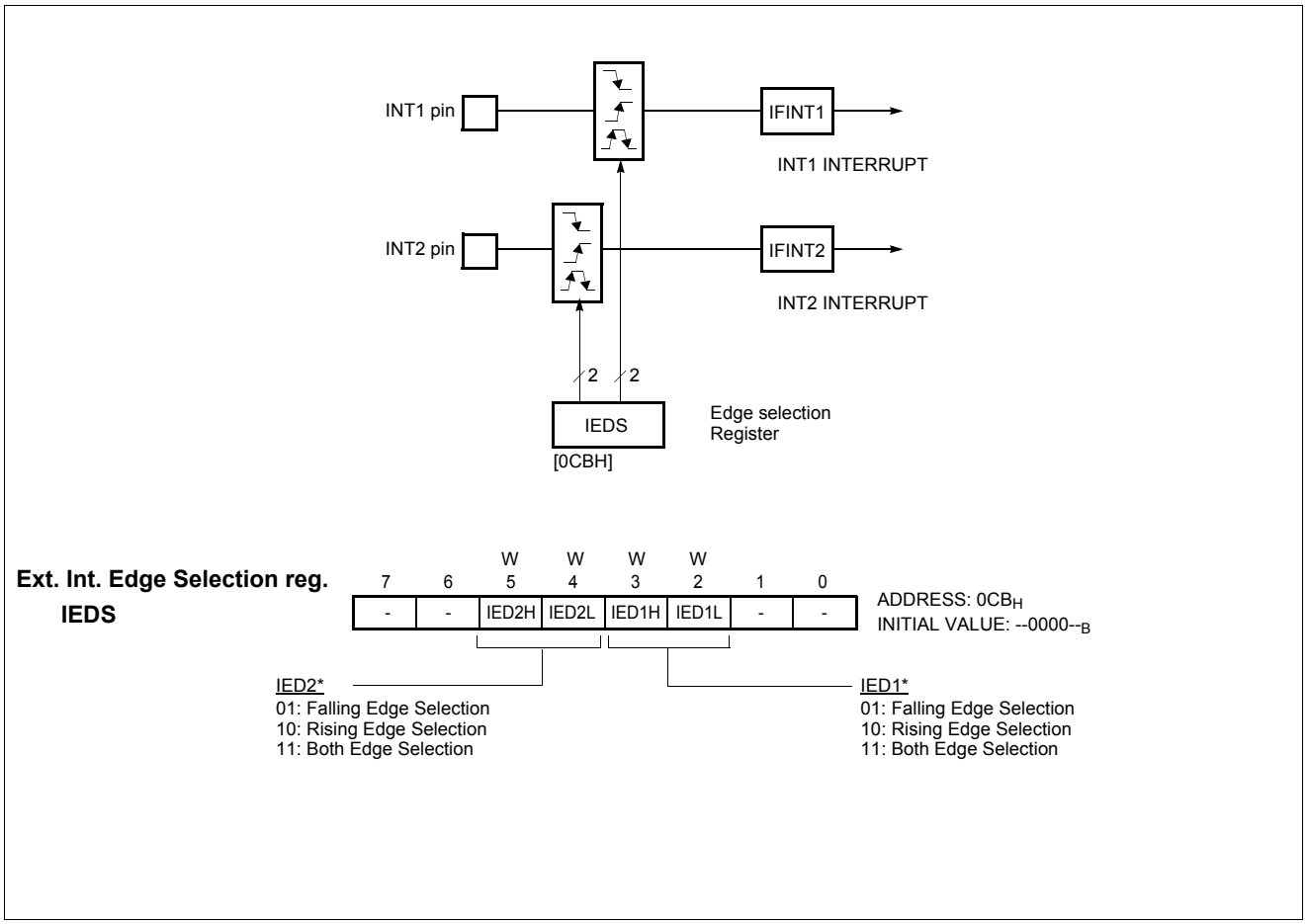



**Figure 15-7 External Interrupt Block Diagram**

### 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.

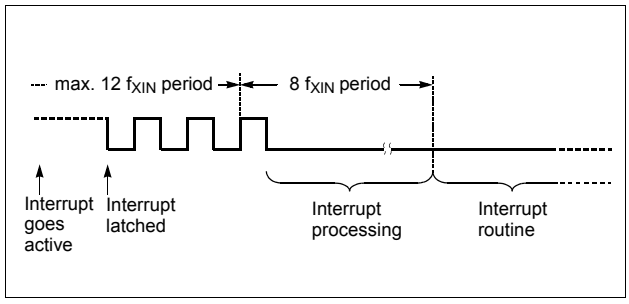



**Figure 15-8 Interrupt Response Timing Diagram**

## 15.8 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.

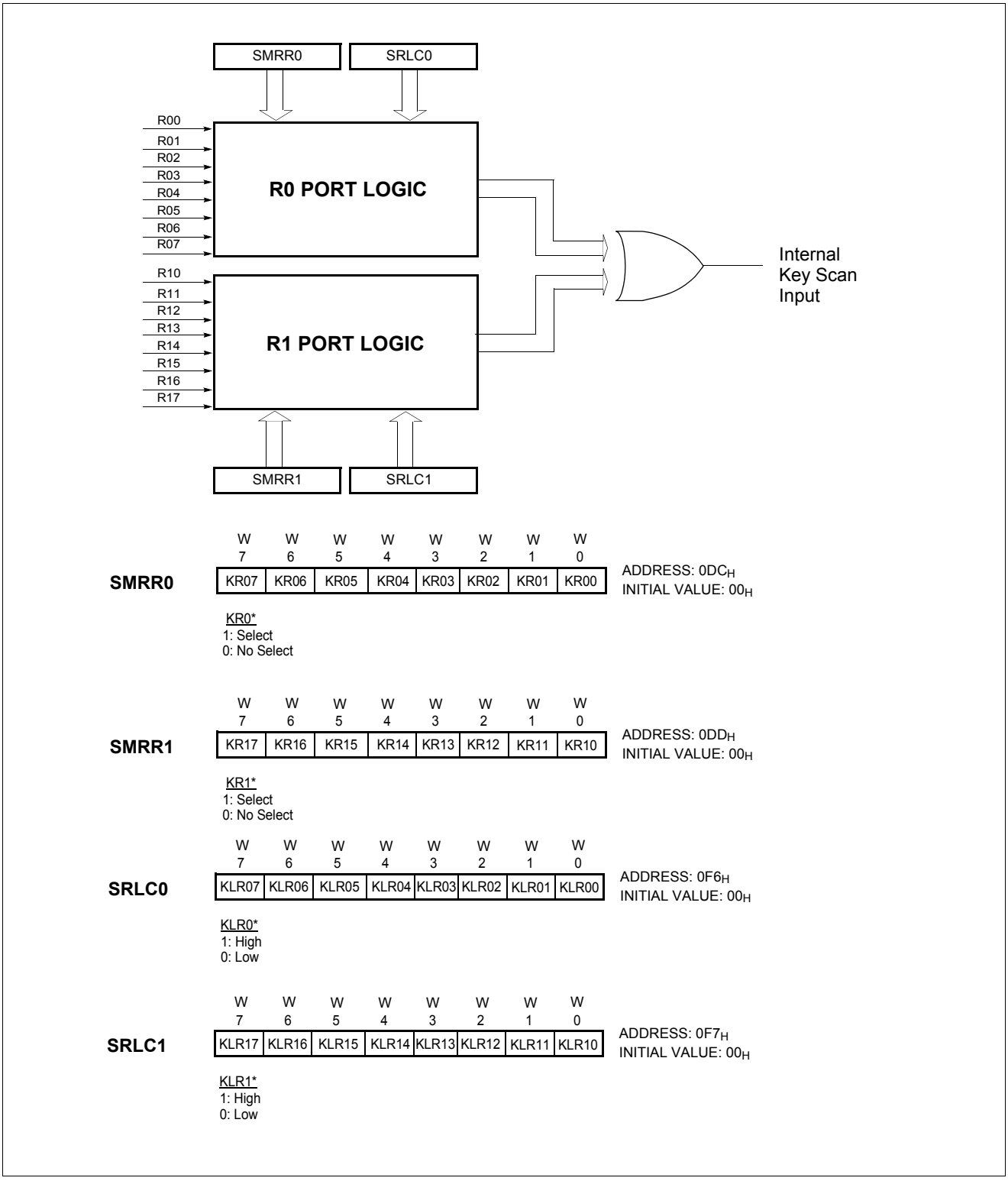


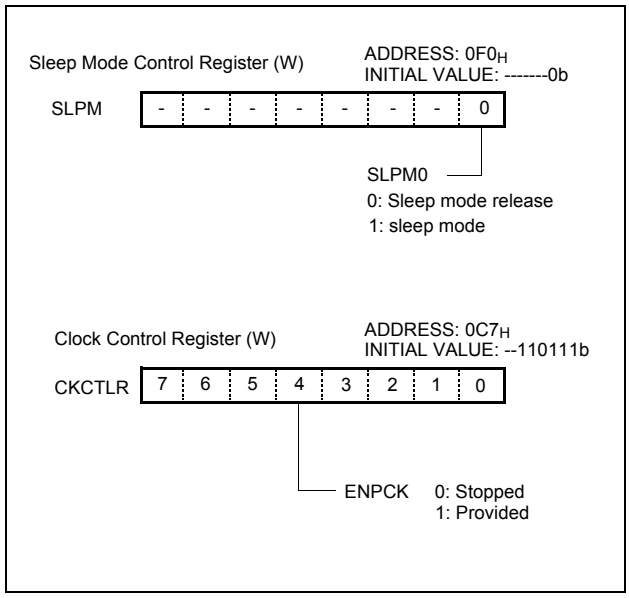

**Figure 15-9 Block Diagram of Key Scan Block**

# 16. STANDBY FUNCTION

## 16.1 Sleep Mode

SLEEP mode can be entered by setting the bit of SLEEP mode register (SLPM). In the mode, CPU clock stops but oscillator keeps running. B.I.T and a part of peripheral hardware execute, but prescaler output which provide clock to peripherals can be stopped by program. (Except, PS10 can't stopped.) In SLEEP mode, more consuming power can be saved by not using other peripheral hardware except for B.I.T. By setting ENPCK (peripheral clock control bit) of CKCTLR (clock control register) to "0", peripheral hardware halted, and SLEEP mode is entered. To release SLEEP mode by BITR (basic interval timer interrupt), bit10 of prescaler should be selected as B.I.T input clock before entering SLEEP mode. "NOP" instruction should be follows setting of SLEEP mode for rising precharge time of data bus line.

(ex) setting of SLEEP mode : set the bit of SLEEP
; mode register (SLPM)
NOP : NOP instruction




## 16.2 Stop Mode

STOP mode can be entered by STOP instruction during program. In STOP mode, oscillator is stopped to make all clocks stop, which leads to less power consumption. All registers and RAM data are preserved. "NOP" instruction should be follows STOP instruction for rising precharge time of Data Bus line.

(ex) STOP : STOP instruction execution
NOP : NOP instruction

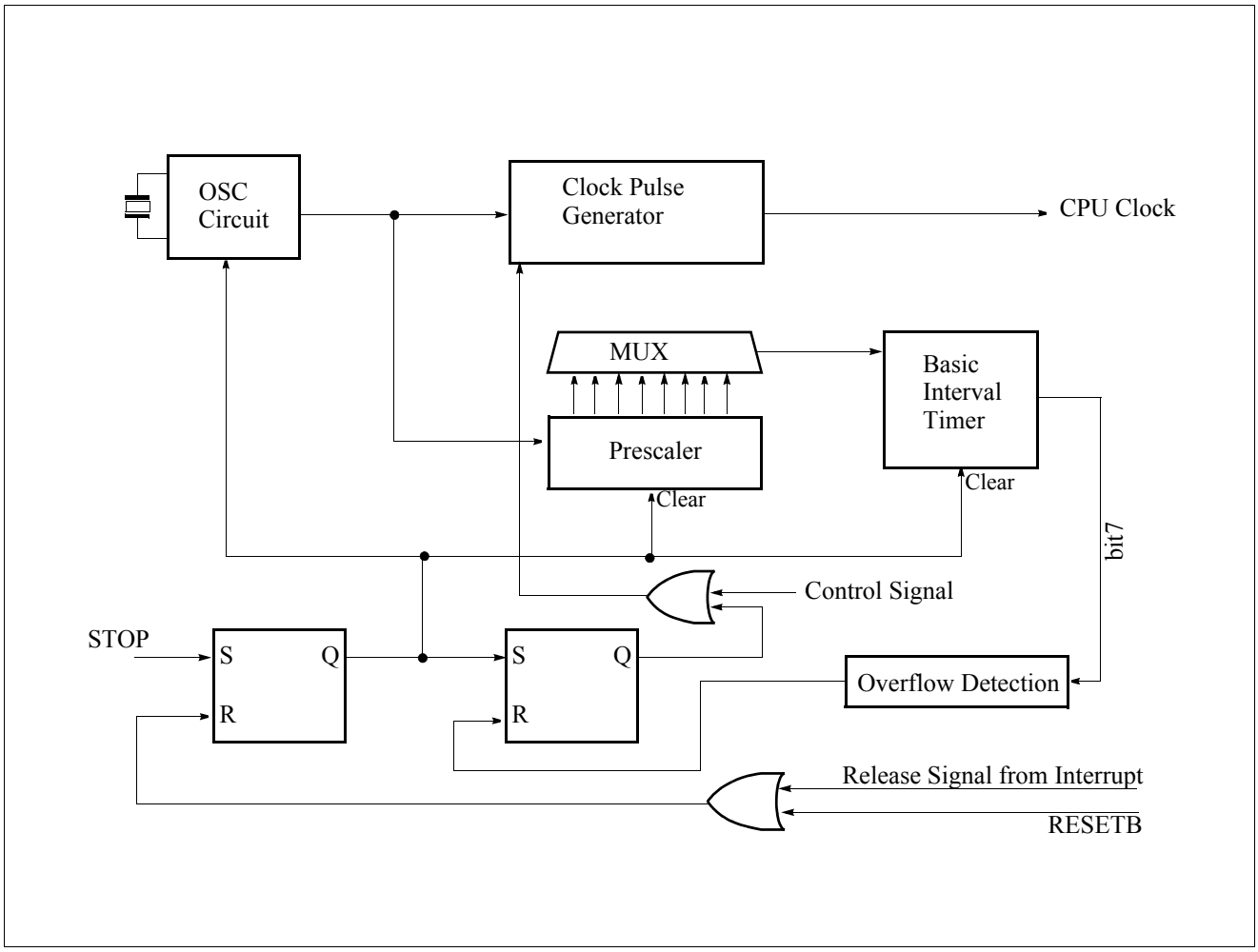



**Figure 16-1 Block Diagram of Standby Circuit**

## 16.3 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 16-1 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 16-2**

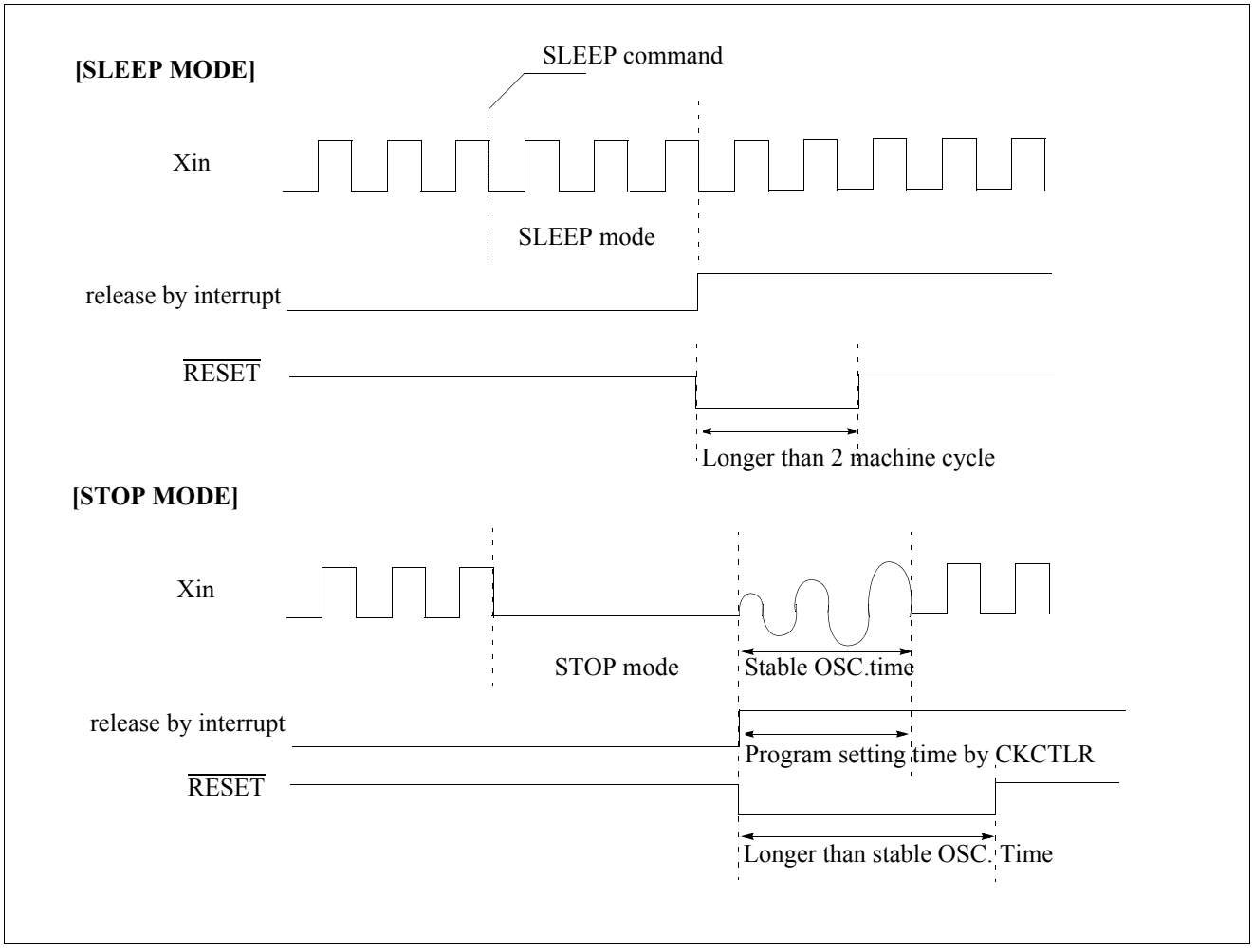



**Figure 16-2 Block Diagram of Standby Circuit**

## 16.4 Operation of standby mode release

After standby mode is released, the operation begins according to content of related interrupt register just before standby mode start (Figure 15-3).

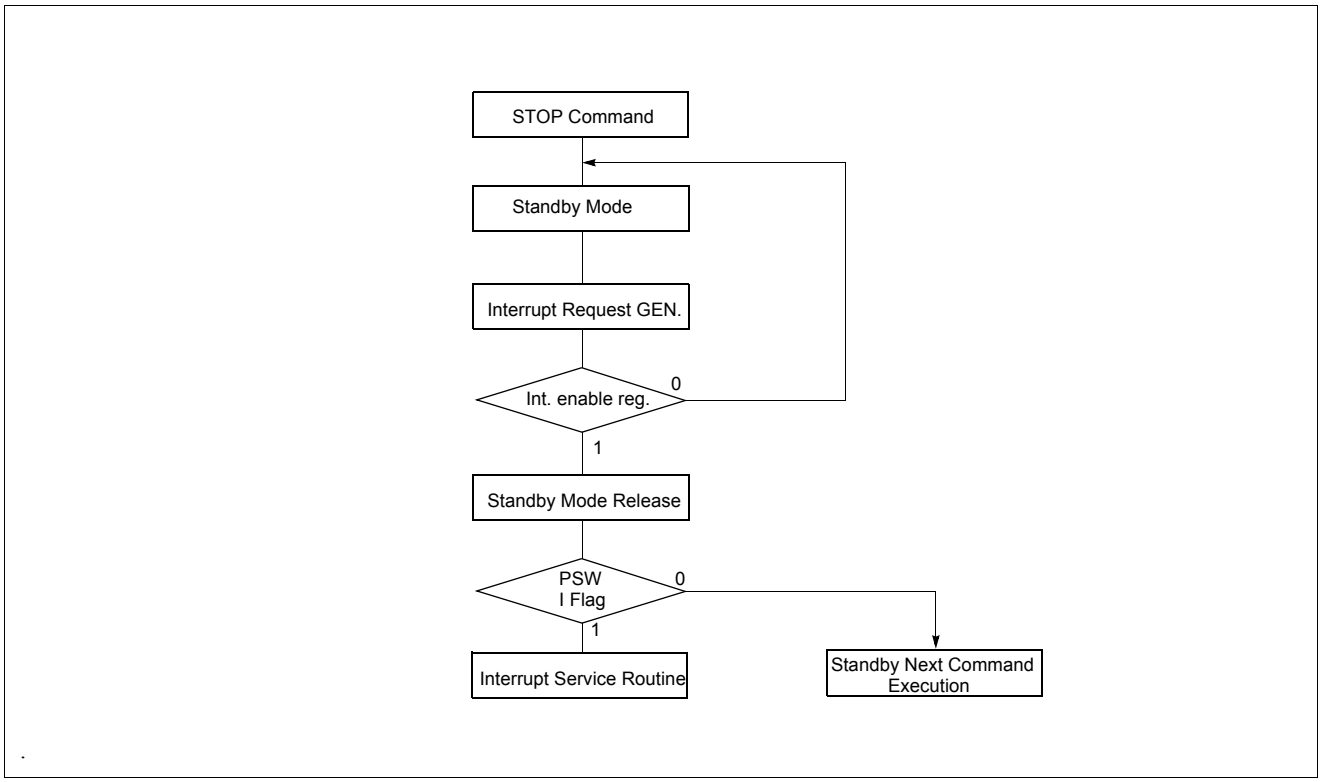



**Figure 16-3 Standby Mode Release Flow**

### (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 entering 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 | Stop |
| Basic Interval Timer | PS10 selected:Active<br>Others: Stop | Stop |
| Watch-dog Timer | Stop | Stop |
| Timer | Stop | Stop |
| Address Bus,Data Bus | Retained | Retained |

**Table 16-3 Operation State in Standby Mode**

# 17. RESET FUNCTION

## 17.1 External RESET

The $\overline{\text{RESET}}$ pin should be held at low for at least 2machine cycles with the power supply voltage within the operating voltage range and must be connected 10kΩ resistor and 0.1uF capacitor for stable system initialization. The RESET pin contains a Schmitt trigger .

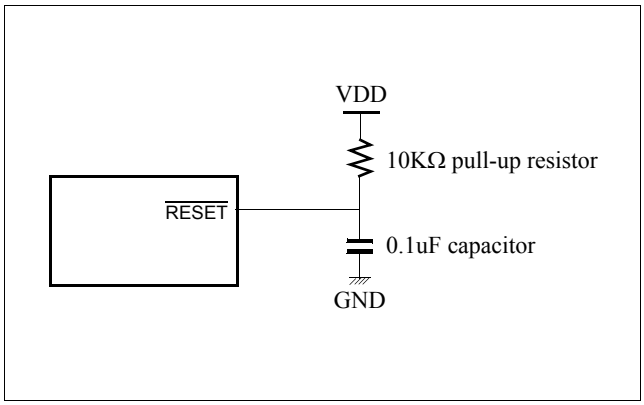



**Figure 17-1 $\overline{\text{RESET}}$ Pin connection**

## 17.2 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

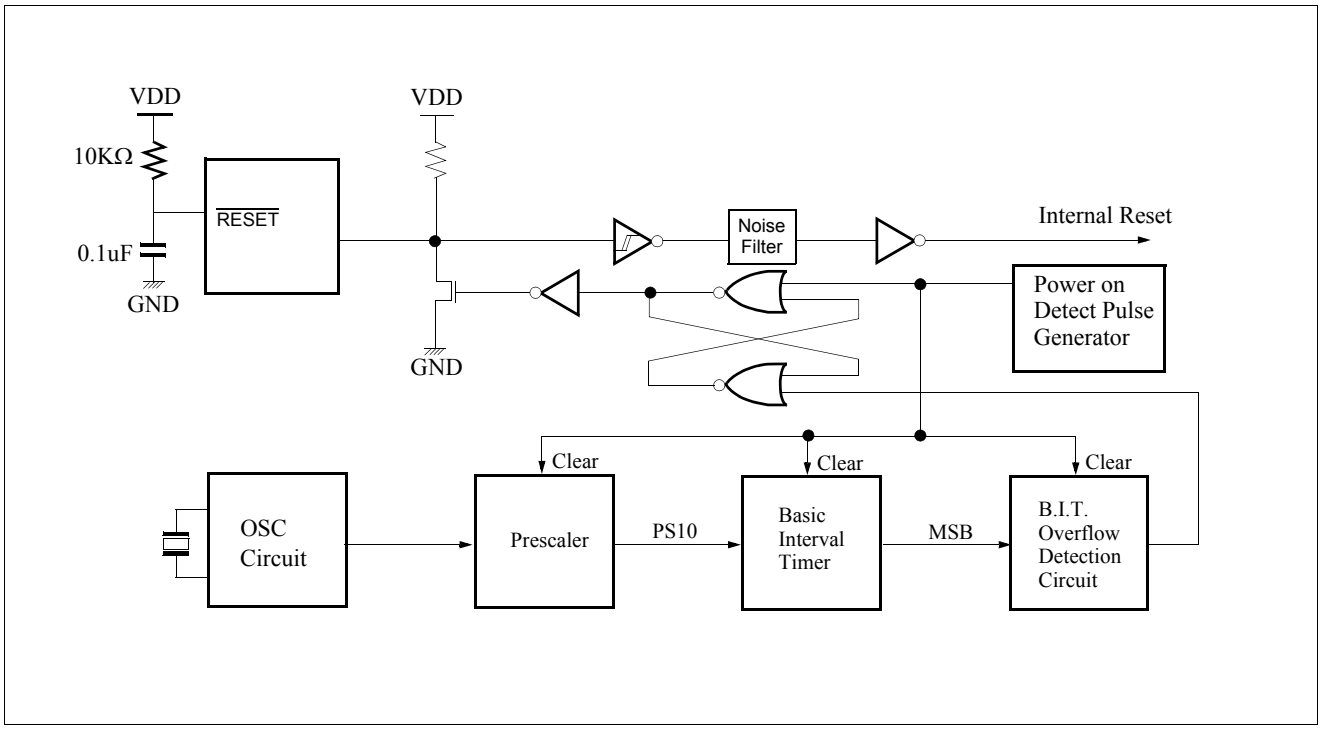



**Figure 17-2 Block Diagram of Power On Reset Circuit**

***Note:*** *When Power On Reset, oscillator stabilization time doesn`t include OSC. Start time.*

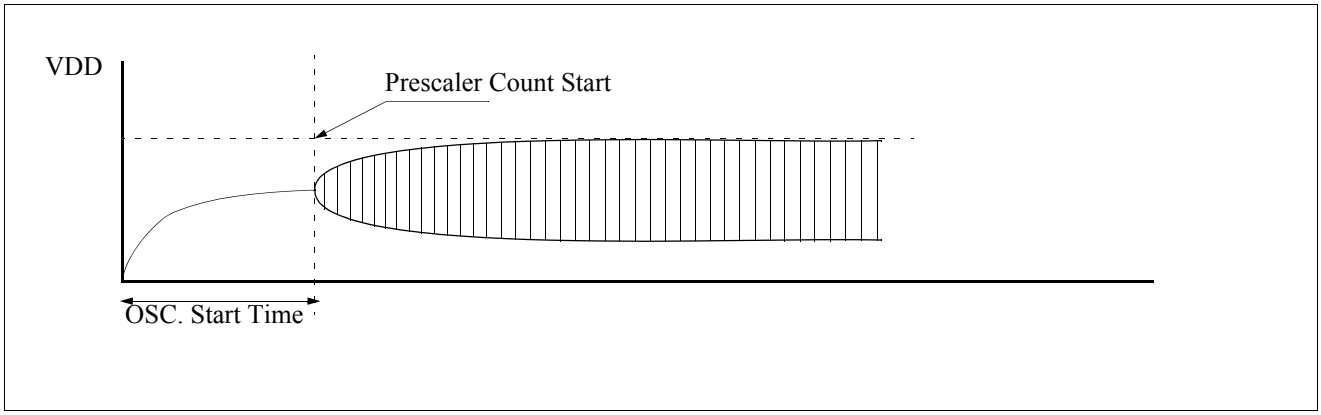



**Figure 17-3 Oscillator stabiliaztion diagram**

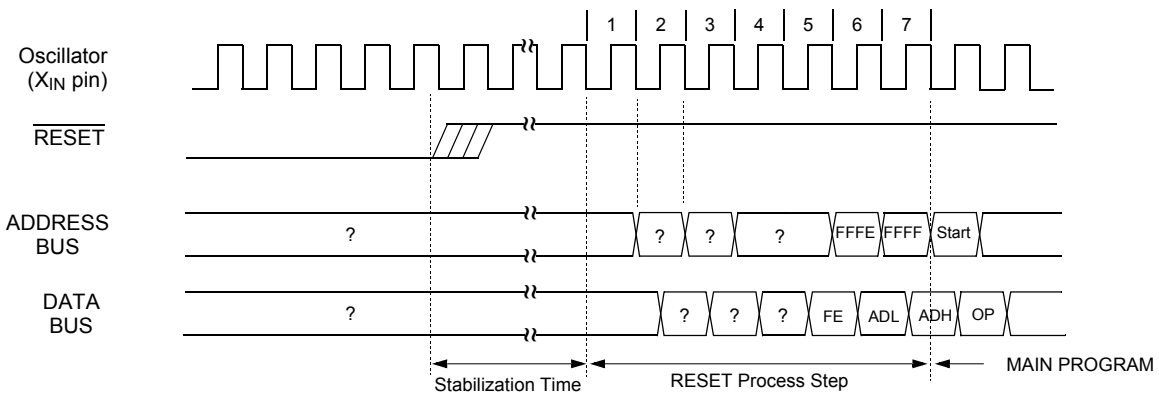



**Figure 17-4 Timing Diagram of Reset**

## 17.3 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.

### (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.

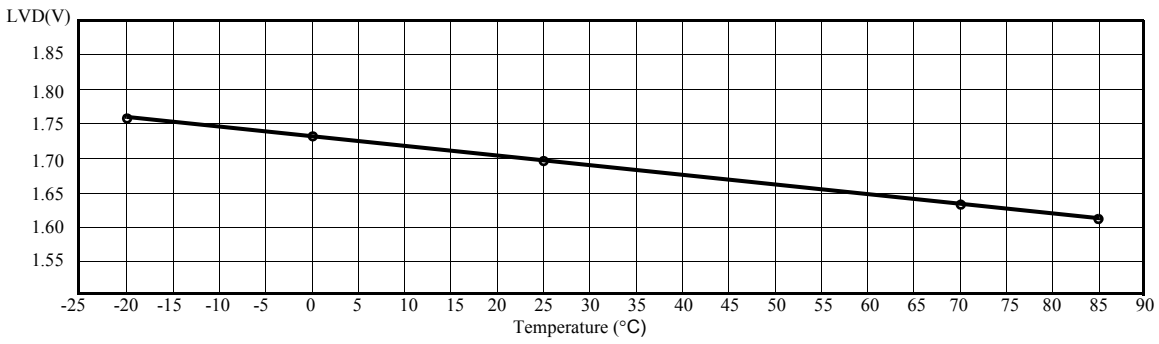



**Figure 17-5 Low Voltage vs Temperature**

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

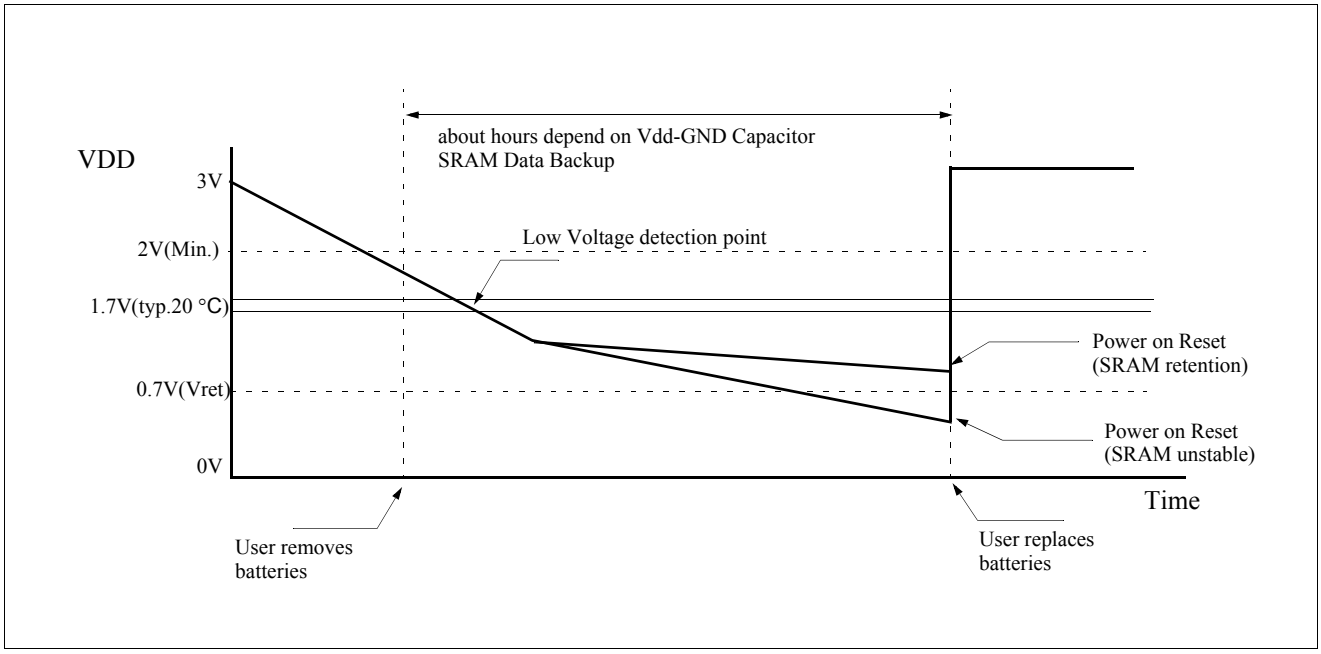



**Figure 17-6 Low voltage detection and protection**

| | |
|---|---|
| Interrupt | disable |
| Stop release | disable |
| All I/O port | input Mode |
| Remout port | Low Level |
| OSC | STOP |
| SRAM Data | retention until Fret |

**Table 17-1 The operation after Low Voltage detection**

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

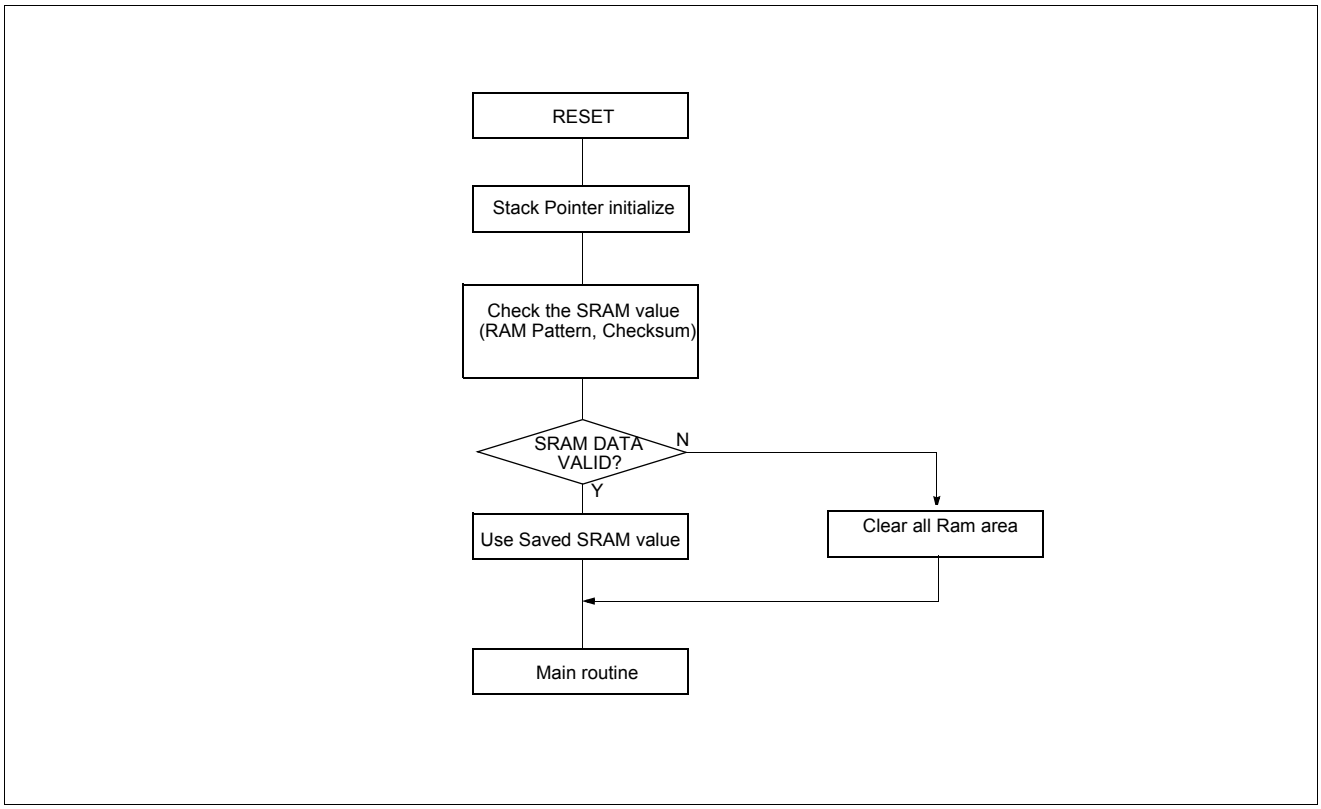
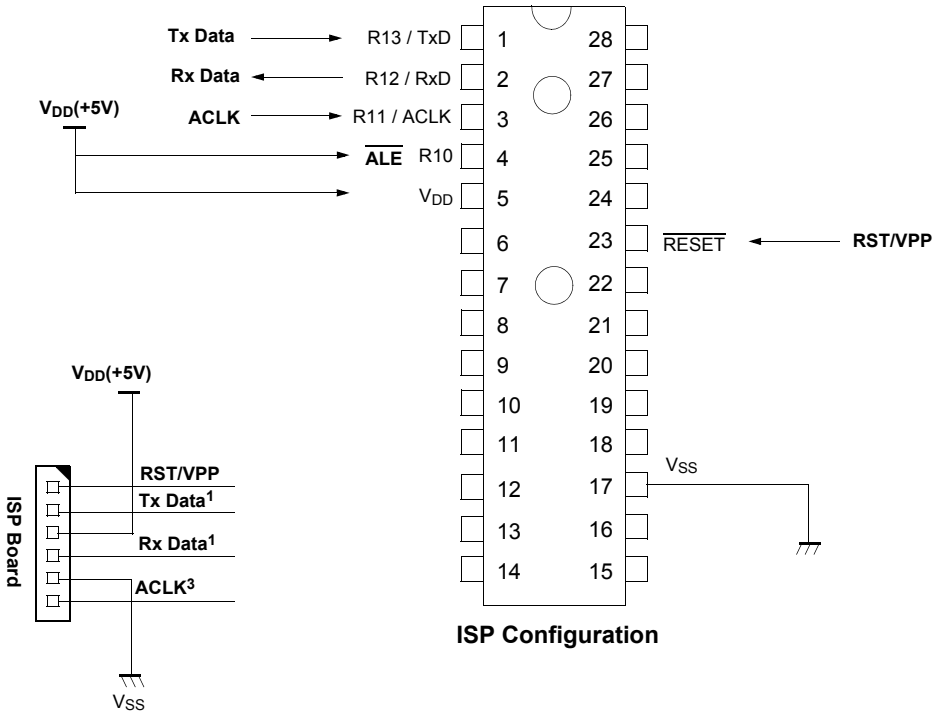



**Figure 17-7 S/W flow chart example after Reset using SRAM Back-up**

## 18. Hardware Conditions to Enter the ISP Mode

The boot loader can be executed by holding ALEB high, RST/$V_{PP}$ as +9V, and ACLK(optional) with OSC. 1.8432MHz. The ISP function uses following pins: TxD, RxD, ALEB, ACLK and RST/$V_{PP}$.



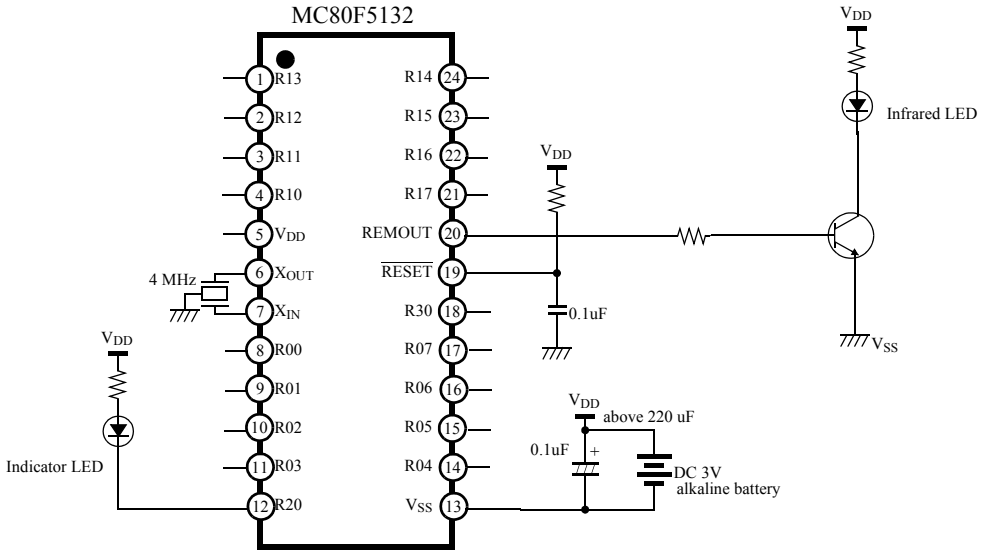

**ISP Configuration**

**1. If other signals affect UART communiction in ISP mode, disconnect these pins by using a jumper or a switch.**
**2. The ALEB can be shared with other function. Toggle the port between ISP and user mode.**
**3. Refer to the section 27.5 explaining the auto bua drate and ACK mode.**

# 19. GENERAL CIRCUIT DIAGRAM

## 19.1 Recommendable circuit diagram of MC80F5132

In case of using high gain Tr. for longer transmission distance application we recommend to attach proper value of capacitor between REMOUT pin of chip and $V_{SS}$ of the system to prevent excessive overshoot voltage of the system power (over the maximum supply $V_{DD}$ of chip) during signal transmission.




1. Tr. and IR LED should be located as far as possible from the IC.
2. Power capacitor and noise filtering capacitor should be located as near as possible from the IC.
3. Ceramic resonator should be located as near as possible from the IC.

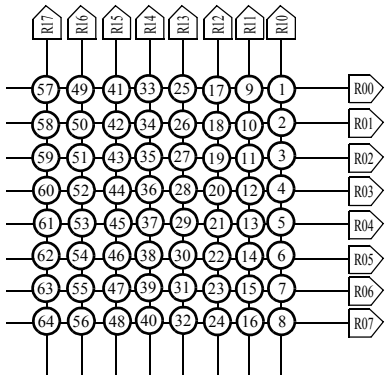



**Figure 19-1 GENERAL CIRCUIT DIAGRAM OF MC80F5132**

# APPENDIX

# A. INSTRUCTION MAP

| LOW / HIGH | 00000 00 | 00001 01 | 00010 02 | 00011 03 | 00100 04 | 00101 05 | 00110 06 | 00111 07 | 01000 08 | 01001 09 | 01010 0A | 01011 0B | 01100 0C | 01101 0D | 01110 0E | 01111 0F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | - | SET1 dp.bit | BBS A.bit,rel | BBS dp.bit,rel | ADC #imm | ADC dp | ADC dp+X | ADC !abs | ASL A | ASL dp | TCALL 0 | SETA1 .bit | BIT dp | POP A | PUSH A | BRK |
| 001 | CLRC | | | | SBC #imm | SBC dp | SBC dp+X | SBC !abs | ROL A | ROL dp | TCALL 2 | CLRA1 .bit | COM dp | POP X | PUSH X | BRA rel |
| 010 | CLRG | | | | CMP #imm | CMP dp | CMP dp+X | CMP !abs | LSR A | LSR dp | TCALL 4 | NOT1 M.bit | TST dp | POP Y | PUSH Y | PCALL Upage |
| 011 | DI | | | | OR #imm | OR dp | OR dp+X | OR !abs | ROR A | ROR dp | TCALL 6 | OR1 OR1B | CMPX dp | POP PSW | PUSH PSW | RET |
| 100 | CLRV | | | | AND #imm | AND dp | AND dp+X | AND !abs | INC A | INC dp | TCALL 8 | AND1 AND1B | CMPY dp | CBNE dp+X | TXSP | INC X |
| 101 | SETC | | | | EOR #imm | EOR dp | EOR dp+X | EOR !abs | DEC A | DEC dp | TCALL 10 | EOR1 EOR1B | DBNE dp | XMA dp+X | TSPX | DEC X |
| 110 | SETG | | | | LDA #imm | LDA dp | LDA dp+X | LDA !abs | TXA | LDY dp | TCALL 12 | LDC LDCB | LDX dp | LDX dp+Y | XCN | DAS (N/A) |
| 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 (N/A) |
| 111 | BEQ rel | | | | STA {X} | STA !abs+Y | STA [dp+X] | STA [dp]+Y | STY !abs | STY dp+X | TCALL 15 | STA {X}+ | STX !abs | CBNE dp | XYX | NOP |

# B. INSTRUCTION SET

## 1. ARITHMETIC/ LOGIC OPERATION

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 1 | ADC #imm | 04 | 2 | 2 | Add with carry. | |
| 2 | ADC dp | 05 | 2 | 3 | A ← ( A ) + ( M ) + C | |
| 3 | ADC dp + X | 06 | 2 | 4 | | |
| 4 | ADC !abs | 07 | 3 | 4 | | NV--H-ZC |
| 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 | |
| 10 | AND dp | 85 | 2 | 3 | A ← ( A ) ∧ ( M ) | |
| 11 | AND dp + X | 86 | 2 | 4 | | |
| 12 | AND !abs | 87 | 3 | 4 | | N-----Z- |
| 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 | |
| 18 | ASL dp | 09 | 2 | 4 | | N-----ZC |
| 19 | ASL dp + X | 19 | 2 | 5 | | |
| 20 | ASL !abs | 18 | 3 | 5 | | |
| 21 | CMP #imm | 44 | 2 | 2 | Compare accumulator contents with memory contents | |
| 22 | CMP dp | 45 | 2 | 3 | ( A ) - ( M ) | |
| 23 | CMP dp + X | 46 | 2 | 4 | | |
| 24 | CMP !abs | 47 | 3 | 4 | | N-----ZC |
| 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 | |
| 30 | CMPX dp | 6C | 2 | 3 | ( X ) - ( M ) | N-----ZC |
| 31 | CMPX !abs | 7C | 3 | 4 | | |
| 32 | CMPY #imm | 7E | 2 | 2 | Compare Y contents with memory contents | |
| 33 | CMPY dp | 8C | 2 | 3 | ( Y ) - ( M ) | N-----ZC |
| 34 | CMPY !abs | 9C | 3 | 4 | | |
| 35 | COM dp | 2C | 2 | 4 | 1'S Complement : ( dp ) ← ~( dp ) | N-----Z- |
| 36 | DAA | - | - | - | Not supported | |
| 37 | DAS | - | - | - | Not supported | |
| 38 | DEC A | A8 | 1 | 2 | Decrement | N-----Z- |
| 39 | DEC dp | A9 | 2 | 4 | M ← ( M ) - 1 | |
| 40 | DEC dp + X | B9 | 2 | 5 | | N-----Z- |
| 41 | DEC !abs | B8 | 3 | 5 | | |
| 42 | DEC X | AF | 1 | 2 | | |
| 43 | DEC Y | BE | 1 | 2 | | |
| 44 | DIV | 9B | 1 | 12 | Divide : YA / X Q: A, R: Y | NV--H-Z- |

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 45 | EOR #imm | A4 | 2 | 2 | Exclusive OR<br>A ← ( A ) ⊕ ( M ) | |
| 46 | EOR dp | A5 | 2 | 3 | | |
| 47 | EOR dp + X | A6 | 2 | 4 | | |
| 48 | EOR !abs | A7 | 3 | 4 | | N-----Z- |
| 49 | EOR !abs + Y | B5 | 3 | 5 | | |
| 50 | EOR [ dp + X ] | B6 | 2 | 6 | | |
| 51 | EOR [ dp ] + Y | B7 | 2 | 6 | | |
| 52 | EOR { X } | B4 | 1 | 3 | | |
| 53 | INC A | 88 | 1 | 2 | Increment<br>M ← ( M ) + 1 | N-----ZC |
| 54 | INC dp | 89 | 2 | 4 | | |
| 55 | INC dp + X | 99 | 2 | 5 | | N-----Z- |
| 56 | INC !abs | 98 | 3 | 5 | | |
| 57 | INC X | 8F | 1 | 2 | | |
| 58 | INC Y | 9E | 1 | 2 | | |
| 59 | LSR A | 48 | 1 | 2 | Logical shift right | |
| 60 | LSR dp | 49 | 2 | 4 | | N-----ZC |
| 61 | LSR dp + X | 59 | 2 | 5 | | |
| 62 | LSR !abs | 58 | 3 | 5 | | |
| 63 | MUL | 5B | 1 | 9 | Multiply : YA ← Y × A | N-----Z- |
| 64 | OR #imm | 64 | 2 | 2 | Logical OR<br>A ← ( A ) ∨ ( M ) | |
| 65 | OR dp | 65 | 2 | 3 | | |
| 66 | OR dp + X | 66 | 2 | 4 | | |
| 67 | OR !abs | 67 | 3 | 4 | | N-----Z- |
| 68 | OR !abs + Y | 75 | 3 | 5 | | |
| 69 | OR [ dp + X ] | 76 | 2 | 6 | | |
| 70 | OR [ dp ] + Y | 77 | 2 | 6 | | |
| 71 | OR { X } | 74 | 1 | 3 | | |
| 72 | ROL A | 28 | 1 | 2 | Rotate left through carry | |
| 73 | ROL dp | 29 | 2 | 4 | | N-----ZC |
| 74 | ROL dp + X | 39 | 2 | 5 | | |
| 75 | ROL !abs | 38 | 3 | 5 | | |
| 76 | ROR A | 68 | 1 | 2 | Rotate right through carry | |
| 77 | ROR dp | 69 | 2 | 4 | | N-----ZC |
| 78 | ROR dp + X | 79 | 2 | 5 | | |
| 79 | ROR !abs | 78 | 3 | 5 | | |
| 80 | SBC #imm | 24 | 2 | 2 | Subtract with carry<br>A ← ( A ) - ( M ) - ~( C ) | |
| 81 | SBC dp | 25 | 2 | 3 | | |
| 82 | SBC dp + X | 26 | 2 | 4 | | |
| 83 | SBC !abs | 27 | 3 | 4 | | NV--HZC |
| 84 | SBC !abs + Y | 35 | 3 | 5 | | |
| 85 | SBC [ dp + X ] | 36 | 2 | 6 | | |
| 86 | SBC [ dp ] + Y | 37 | 2 | 6 | | |
| 87 | SBC { X } | 34 | 1 | 3 | | |
| 88 | TST dp | 4C | 2 | 3 | Test memory contents for negative or zero<br>( dp ) - $00_H$ | N-----Z- |
| 89 | XCN | CE | 1 | 5 | Exchange nibbles within the accumulator<br>$A_7$~$A_4$ ↔ $A_3$~$A_0$ | N-----Z- |

## 2. REGISTER / MEMORY OPERATION

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

## 3. 16-BIT OPERATION

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 1 | ADDW dp | 1D | 2 | 5 | 16-Bits add without carry<br>YA ← ( YA ) + ( dp +1 ) ( dp ) | NV--H-ZC |
| 2 | CMPW dp | 5D | 2 | 4 | Compare YA contents with memory pair contents :<br>(YA) – (dp+1)(dp) | N-----ZC |
| 3 | DECW dp | BD | 2 | 6 | Decrement memory pair<br>( dp+1)( dp) ← ( dp+1) ( dp) - 1 | N-----Z- |
| 4 | INCW dp | 9D | 2 | 6 | Increment memory pair<br>( dp+1) ( dp) ← ( dp+1) ( dp ) + 1 | N-----Z- |
| 5 | LDYA dp | 7D | 2 | 5 | Load YA<br>YA ← ( dp +1 ) ( dp ) | N-----Z- |
| 6 | STYA dp | DD | 2 | 5 | Store YA<br>( dp +1 ) ( dp ) ← YA | -------- |
| 7 | SUBW dp | 3D | 2 | 5 | 16-Bits substact without carry<br>YA ← ( YA ) - ( dp +1) ( dp) | NV--H-ZC |

## 4. BIT MANIPULATION

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 1 | AND1 M.bit | 8B | 3 | 4 | Bit AND C-flag : C ← ( C ) ∧ ( M .bit ) | -------C |
| 2 | AND1B M.bit | 8B | 3 | 4 | Bit AND C-flag and NOT : C ← ( C ) ∧ ~( M .bit ) | -------C |
| 3 | BIT dp | 0C | 2 | 4 | Bit test A with memory :<br>Z ← ( A ) ∧ ( M ) , N ← ( $M_7$ ) , V ← ( $M_6$ ) | MM----Z- |
| 4 | BIT !abs | 1C | 3 | 5 | | |
| 5 | CLR1 dp.bit | y1 | 2 | 4 | Clear bit : ( M.bit ) ← “0” | -------- |
| 6 | CLRA1 A.bit | 2B | 2 | 2 | Clear A bit : ( A.bit )← “0” | -------- |
| 7 | CLRC | 20 | 1 | 2 | Clear C-flag : C ← “0” | -------0 |
| 8 | CLRG | 40 | 1 | 2 | Clear G-flag : G ← “0” | --0----- |
| 9 | CLRV | 80 | 1 | 2 | Clear V-flag : V ← “0” | -0--0--- |
| 10 | EOR1 M.bit | AB | 3 | 5 | Bit exclusive-OR C-flag : C ← ( C ) ⊕ ( M .bit ) | -------C |
| 11 | EOR1B M.bit | AB | 3 | 5 | Bit exclusive-OR C-flag and NOT : C ← ( C ) ⊕ ~(M .bit) | -------C |
| 12 | LDC M.bit | CB | 3 | 4 | Load C-flag : C ← ( M .bit ) | -------C |
| 13 | LDCB M.bit | CB | 3 | 4 | Load C-flag with NOT : C ← ~( M .bit ) | -------C |
| 14 | NOT1 M.bit | 4B | 3 | 5 | Bit complement : ( M .bit ) ← ~( M .bit ) | -------- |
| 15 | OR1 M.bit | 6B | 3 | 5 | Bit OR C-flag : C ← ( C ) ∨ ( M .bit ) | -------C |
| 16 | OR1B M.bit | 6B | 3 | 5 | Bit OR C-flag and NOT : C ← ( C ) ∨ ~( M .bit ) | -------C |
| 17 | SET1 dp.bit | x1 | 2 | 4 | Set bit : ( M.bit ) ← “1” | -------- |
| 18 | SETA1 A.bit | 0B | 2 | 2 | Set A bit : ( A.bit ) ← “1” | -------- |
| 19 | SETC | A0 | 1 | 2 | Set C-flag : C ← “1” | -------1 |
| 20 | SETG | C0 | 1 | 2 | Set G-flag : G ← “1” | --1----- |
| 21 | STC M.bit | EB | 3 | 6 | Store C-flag : ( M .bit ) ← C | -------- |
| 22 | TCLR1 !abs | 5C | 3 | 6 | Test and clear bits with A :<br>A - ( M ) , ( M ) ← ( M ) ∧ ~( A ) | N-----Z- |
| 23 | TSET1 !abs | 3C | 3 | 6 | Test and set bits with A :<br>A - ( M ) , ( M ) ← ( M ) ∨ ( A ) | N-----Z- |

## 5. BRANCH / JUMP OPERATION

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

## 6. CONTROL OPERATION & etc.

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 1 | BRK | 0F | 1 | 8 | Software interrupt : B ← "1", M(sp) ← ($pc_H$), sp ←sp-1, M(s) ← ($pc_L$), sp ← sp - 1, M(sp) ← (PSW), sp ← sp -1, $pc_L$ ← ( $0FFDE_H$ ) , $pc_H$ ← ( $0FFDF_H$) . | ---1-0-- |
| 2 | DI | 60 | 1 | 3 | Disable interrupts : I ← "0" | -----0-- |
| 3 | EI | E0 | 1 | 3 | Enable interrupts : I ← "1" | -----1-- |
| 4 | NOP | FF | 1 | 2 | No operation | -------- |
| 5 | POP A | 0D | 1 | 4 | sp ← sp + 1, A ← M( sp ) | -------- |
| 6 | POP X | 2D | 1 | 4 | sp ← sp + 1, X ← M( sp ) | |
| 7 | POP Y | 4D | 1 | 4 | sp ← sp + 1, Y ← M( sp ) | |
| 8 | POP PSW | 6D | 1 | 4 | sp ← sp + 1, PSW ← M( sp ) | restored |
| 9 | PUSH A | 0E | 1 | 4 | M( sp ) ← A , sp ← sp - 1 | -------- |
| 10 | PUSH X | 2E | 1 | 4 | M( sp ) ← X , sp ← sp - 1 | |
| 11 | PUSH Y | 4E | 1 | 4 | M( sp ) ← Y , sp ← sp - 1 | |
| 12 | PUSH PSW | 6E | 1 | 4 | M( sp ) ← PSW , sp ← sp - 1 | |
| 13 | RET | 6F | 1 | 5 | Return from subroutine<br>sp ← sp +1, $pc_L$ ← M( sp ), sp ← sp +1, $pc_H$ ← M( sp ) | -------- |
| 14 | RETI | 7F | 1 | 6 | Return from interrupt<br>sp ← sp +1, PSW ← M( sp ), sp ← sp + 1,<br>$pc_L$ ← M( sp ), sp ← sp + 1, $pc_H$ ← M( sp ) | restored |
| 15 | STOP | EF | 1 | 3 | Stop mode ( halt CPU, stop oscillator ) | -------- |