ABOV SEMICONDUCTOR Co. Ltd.
8-BIT SINGLE-CHIP MICROCONTROLLERS

# MC80F7708A

*User's Manual (Ver. 1.11)*

---

ABOV SEMICONDUCTOR

**Version 1.11**

**Published by**
**FAE Team**

**REVISION HISTORY**

**VERSION 1.11 (December 3, 2012) This book**

ABOV logo is renewed on this book.

Single and Gang writer are added in "1.3 Development Tools" on page 3.

VDD voltage for sub-active mode is changed to 3.0~5.5V in "7.2 Recommended Operating Conditions" on page 16.

**VERSION 1.1 (March 2, 2010)**

The errata are fixed at "Figure 17-2 Interrupt Enable Registers and Interrupt Request Registers" on page 73

**VERSION 1.0 (December 21, 2009)**

The note of watch timer interrupt request flag(WTIF) is added in "13. WATCH TIMER" on page 61.

The value of conversion time at "7.5 A/D Converter Characteristics" on page 18 is modified to prevent confusion.

ADC interrupt function is eliminated in MC80F7708A, so the desription of "ADC interrupt function" is deleted in this manual. ie. ADCE(bit0 of IENM) is eliminated.

The description of "configuration option bit" is modified to prevent confusion in "25.1 FLASH Configuration Byte" on page 99.

The description of $f_{MAIN}$ is added in this manual. ie. $f_{MAIN}$ is main-clock frequency at main-active mode and it is sub-clock frequency at sub-active mode.

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**VERSION 0.1 (August 21, 2009)**

The figures of flash writer were updated in "1. OVERVIEW" on page 1.

**VERSION 0.0 (July 2, 2009)**

First Edition (Preliminary)

# Table of Contents

# MC80F7708A

## CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER WITH LCD CONTROLLER/DRIVER

## 1. OVERVIEW

### 1.1 Description

The MC80F7708A are an advanced CMOS 8-bit microcontroller with 8K bytes of FLASH ROM(ISP). This device is one of the MC800 family and a powerful microcontroller which provides a high flexibility and cost effective solution to many LCD applications. The MC80F7708A provide the following standard features: 8K bytes of FLASH ROM, 256 bytes of RAM, 20 bytes of segment LCD display RAM, 8/16-bit timer/counter, 8-bit A/D converter, 7-bit watch dog timer, 21-bit watch timer with 7-bit auto reload counter, on-chip oscillator and clock circuitry. In addition, this device supports power saving modes to reduce power consumption. So the MC80F7708A is the best controller solution in system which uses charatered LCD display and ADC.

| Device name | Memory (Bytes) | | ADC | PWM | I/O | LCD | Operating Voltage | Package |
|---|---|---|---|---|---|---|---|---|
| | ROM | RAM | | | | | | |
| MC80F7708AQ | 8K | 256 | 6ch. | 1ch. | 42 | (20SEG x 4COM) | 2.2 ~ 5.5V @4Mhz<br>4.5 ~ 5.5V @12Mhz | 44MQFP |
| MC80F7708AK | 8K | 256 | 4ch | 1ch | 30 | (10SEG x 4COM) | 2.2 ~ 5.5V @4Mhz<br>4.5 ~ 5.5V @12Mhz | 32SDIP |
| MC80F7708AD | 8K | 256 | 4ch. | 1ch. | 30 | (10SEG x 4COM) | 2.2 ~ 5.5V @4Mhz<br>4.5 ~ 5.5V @12Mhz | 32SOP |

### 1.2 Features

- **8K Bytes On-chip FLASH ROM (ISP)**
- **FLASH Memory**
  - Endurance : 100 cycles
  - Data Retention : 10 years
- **256 Bytes On-chip Data RAM**
- **20 bytes Display RAM**
- **Instruction Cycle Time**
  - 333ns at 12MHz (2 cycle NOP instruction)
- **LCD display/controller**
  - 1/4 Duty Mode (20Seg(10Seg) × 4Com, 1/3 Bias)
- **Four 8-bit Timer/Counter**
  (They can be used as two 16-bit Timer/Counter)
- **One 7-bit Watch Dog Timer**
- **One 21-bit Watch Timer**
  - 1 minute interrupt available
- **One 8-bit Basic Interval Timer**
- **One 6-bit Buzzer Driving Port**
- **Dual Clock Operation**
  - Main Clock : 400kHz ~ 12MHz
  - Sub Clock : 32.768kHz
- **Main Clock Oscillation**
  - **Crystal**
  - **Ceramic Resonator**
  - **Internal Oscillation : 8MHz/4MHz/2MHz**
- **Operating Temperature : -40~85 °C**
- **Built-in Noise Immunity Circuit**
  - Noise Filter
- **Power Down Mode**
  - Main Clock : STOP, SLEEP,Sub_active mode
- **400kHz to 12MHz Wide Operating Frequency**

- **On-Chip POR (Power On Reset)**
  - 2 level (1.9V, 2.7V)
- **Internal Resistor for LCD Bias**
- **Supply Voltage Level Detector(SVLD)**
  - 4 level detector(2.4V,2.7V,3.0V, 3.9V)
- **42/40 Programmable I/O Pins**

| | |
|---|---|
| MC80F7708AQ | I/O: 17<br>I : 1<br>I/O with SEG/COM:24 |
| MC80F7708AK | I/O: 15<br>I : 1<br>I/O with SEG/COM:14 |
| MC80F7708AD | I/O: 15<br>I : 1<br>I/O with SEG/COM:14 |

- **6/4-channel 8-bit On-chip A/D Converter**

| | |
|---|---|
| MC80F7708AQ | 6-channel ADC |
| MC80F7708AK | 4-channel ADC |
| MC80F7708AD | 4-channel ADC |

- **One 10-bit High Speed PWM Output**
- **11 Interrupt sources**
  - External Interrupt : 4
  - Timer : 4
  - WDT, WT, BIT
- **2.2V to 5.5V Wide Operating Voltage Range**
- **44MQFP, 32SDIP, 32SOP Package Types**
  - Available Pb free package

| | |
|---|---|
| MC80F7708AQ | 44MQFP |
| MC80F7708AK | 32SDIP |
| MC80F7708AD | 32SOP |

### 1.3 Development Tools

The MC80F7708A are supported by a full-featured macro assembler, an in-circuit emulator CHOICE-Dr.$^{TM}$ and OTP/FLASH programmers. There are two different type of programmers such as single type and gang type. For mode detail, Macro assembler operates under the MS-Windows 95 and upversioned Windows OS. And HMS800C compiler only operates under the MS-Windows 2000 and upversioned Windows OS.

Please contact sales part of ABOV semiconductor.

| | |
|---|---|
| Software | - MS-Windows based assembler<br>- MS-Windows based Debugger<br>- MC800 C compiler |
| Hardware (Emulator) | - CHOICE-Dr.<br>- CHOICE-Dr. EVA80C77x B/D |
| POD Name | - POD80C77D-44MQ-1010 |
| FLASH Writer | - PGM Plus USB (Single writer)<br>- Stand Alone PGM Plus(Single writer)<br>- Standalone GANG4/8 USB (Gang writer)<br>- USB-SIO-ISP Board |


**Figure 1-1 Choice-Dr. (Emulator, USB Interface)**


**Figure 1-2 PGM plus USB (Single Writer)**


**Figure 1-3 Stand Alone PGM_Plus(ISP)**


**Figure 1-4 Stand Alone Gang4 USB (Gang Writer)**

**Figure 1-5 Standalone Gang8 (Gang Writer)**

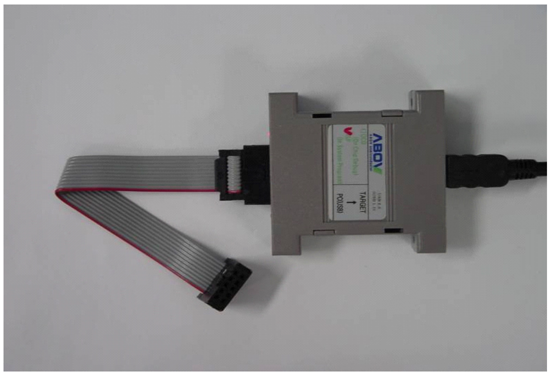
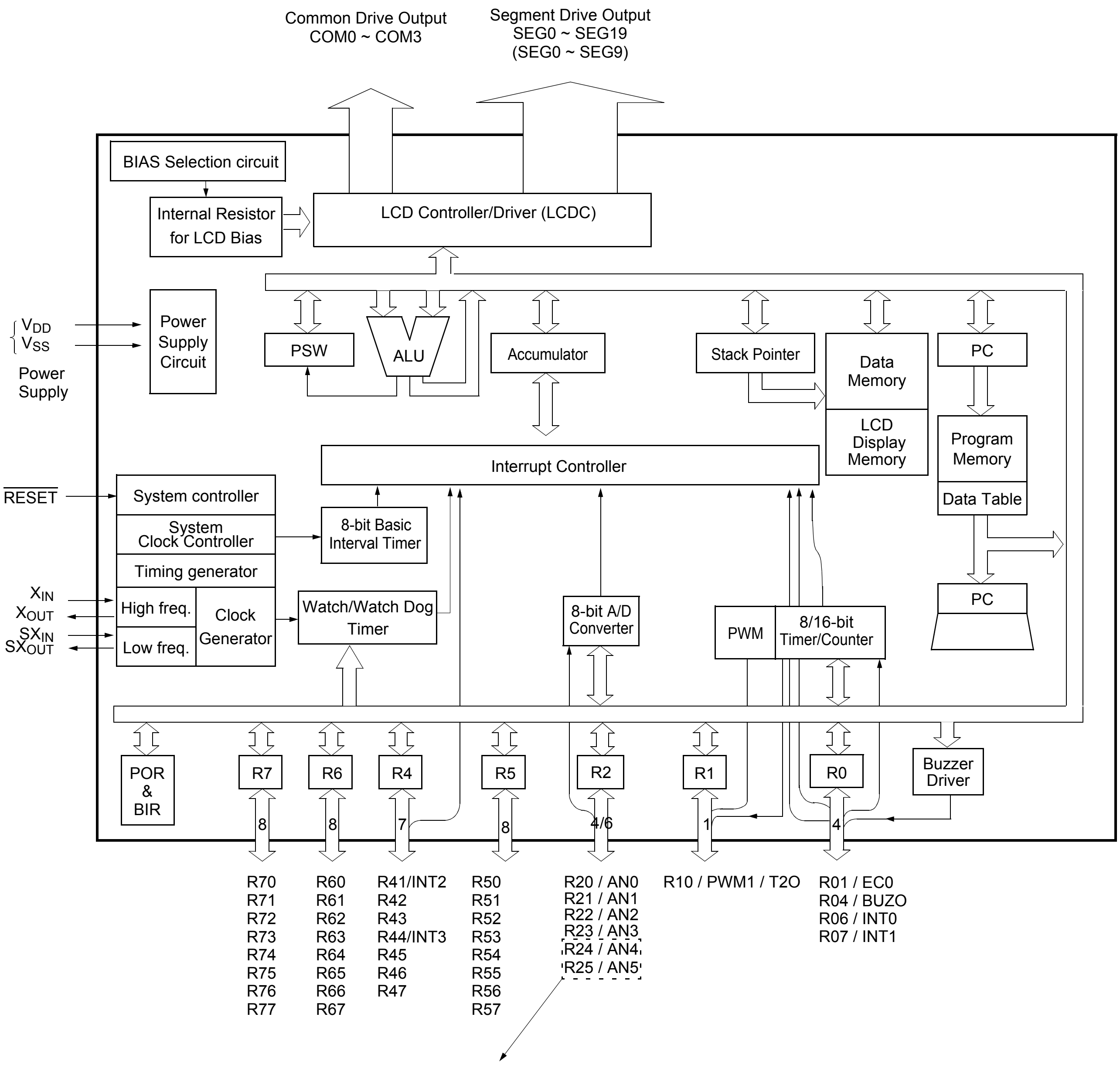


**Figure 1-6 USB-SIO-ISP Board**

## 1.4 Ordering Information

| Device name | ROM Size | RAM size | Package |
|---|---|---|---|
| MC80F7708AQ<br>MC80F7708AK<br>MC80F7708AD | 8K bytes FLASH<br>8K bytes FLASH<br>8K bytes FLASH | 256 bytes<br>256 bytes<br>256 bytes | 44MQFP<br>32SDIP<br>32SOP |

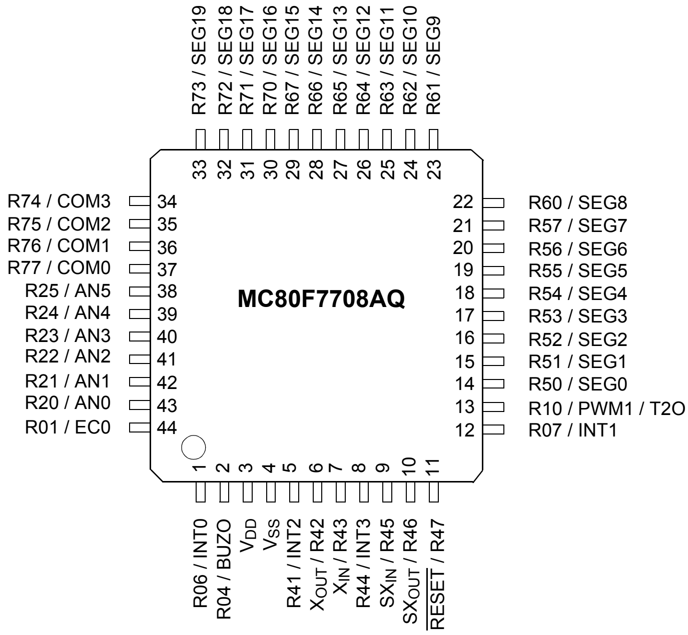
## 2. BLOCK DIAGRAM



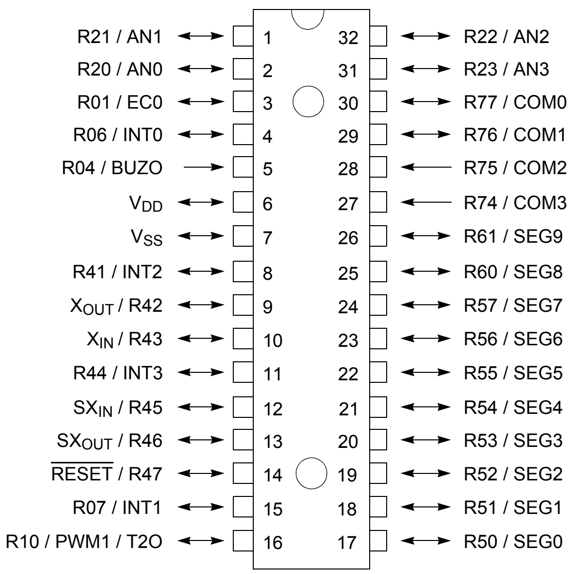

*R24/AN3 and R25/AN5 are not supported in MC80F7708AK/MC80F7708AD(32pin).
MC80F7708AQ : R20/AN0 ~ R25/AN5
MC80F7708AK/MC80F7708AD : R20/AN0 ~ R23/AN3

# 3. PIN ASSIGNMENT

44MQFP
(Top View)

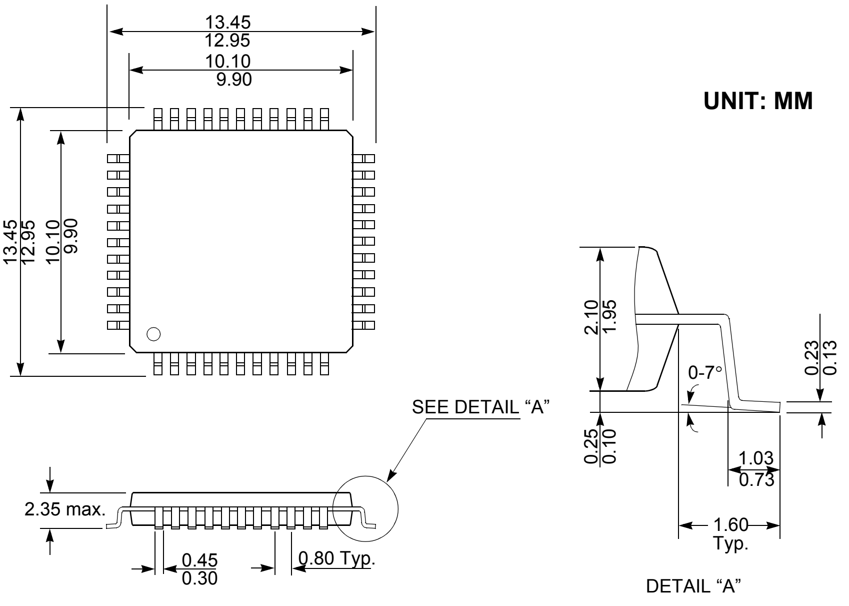



32SDIP/SOP

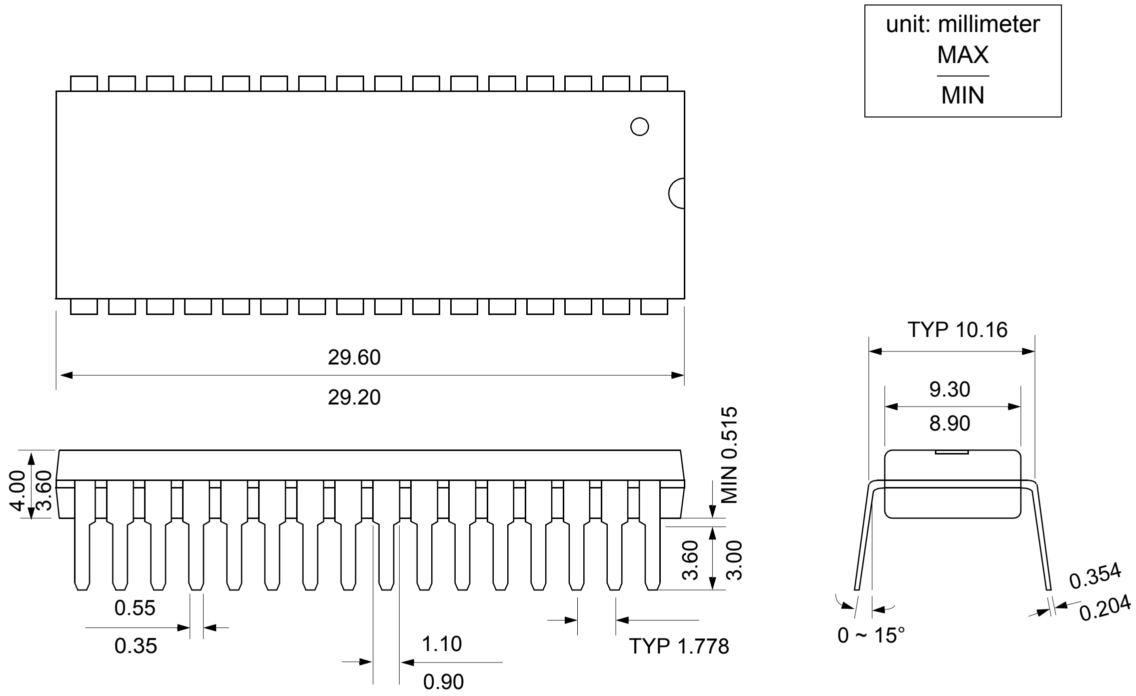


## 4. PACKAGE DIAGRAM

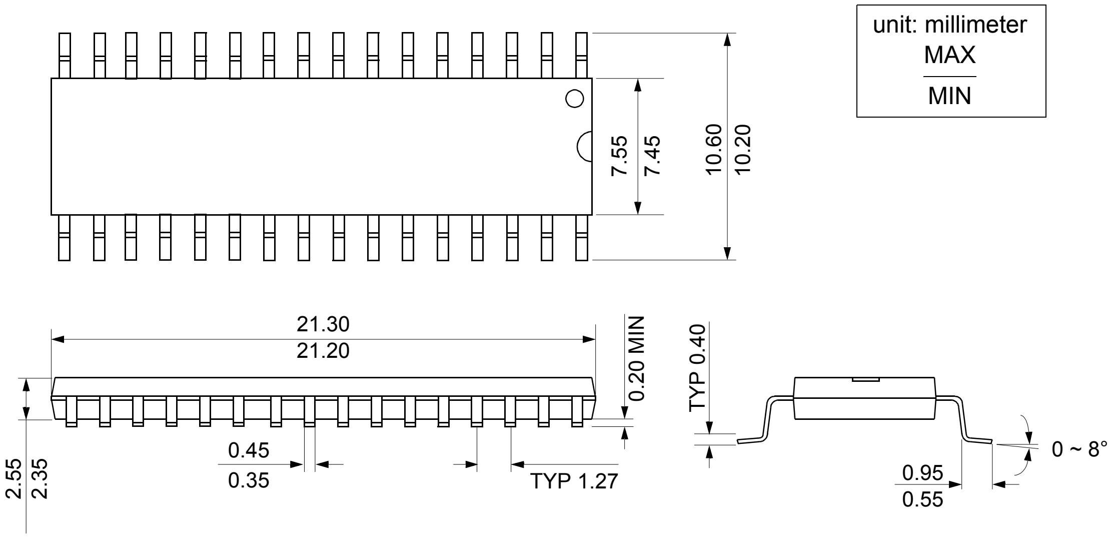
44MQFP




32 SDIP



## 32 SOP


unit: millimeter
MAX
MIN
7.55
7.45
10.60
10.20
21.30
21.20
0.20 MIN
TYP 0.40
2.55
2.35
0.45
0.35
TYP 1.27
0.95
0.55
0 ~ 8°

## 5. PIN FUNCTION

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

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

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

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

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

**$SX_{IN}$:** Input to the internal sub system clock operating circuit.

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

**SEG0~SEG19:** Segment signal output pins for the LCD display. See "18. LCD DRIVER" on page 78 for details. Also SEG0~SEG19 are shared with normal I/O ports.

**COM0~COM3:** Common signal output pins for the LCD display. See "18. LCD DRIVER" on page 78 for details. Also COM0~SEG3 are shared with normal I/O ports.

**R01, R04, R06, R07:** R0 is a 4-bit CMOS bidirectional I/O port. R0 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs. Also, pull-up resistors and open-drain outputs can be assigned by software.

In addition, R0 serves the functions of the various following special features.

| Port pin | Alternate function |
|---|---|
| R01 | EC0 (Timer 0 Event Count Input) |
| R04 | BUZO (Buzzer Output) |
| R06 | INT0 (External Interrupt 0 Request Input) |
| R07 | INT1 (External Interrupt 1 Request input) |

**R10 :** R1 is an 1-bit CMOS bidirectional I/O port. R1 pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs or schmitt trigger inputs. Also, pull-up resistors and open-drain outputs can be assigned by software.

In addition, R1 serves the function of the following special feature.

| Port pin | Alternate function |
|---|---|
| R10 | PWM1/T2O<br>(Timer3 PWM Output / Timer2 Output) |

**R20~R25:** R2 is a 4/6-bit CMOS bidirectional I/O port. Each pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs. Also, pull-up resistors and open-drain outputs can be assigned by software.

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

| Port pin | Alternate function |
|---|---|
| R20 | AN0 (Analog Input Port0) |
| R21 | AN1 (Analog Input Port1) |
| R22 | AN2 (Analog Input Port2) |
| R23 | AN3 (Analog Input Port3) |
| *R24* | *AN4 (Analog Input Port4)* |
| *R25* | *AN5 (Analog Input Port5)* |

***Note:** R24/AN3 and R25/AN5 are not not supported in MC80F7708AK/AD(32pin).*

**R41~R47:** R4 is a 7-bit CMOS bidirectional I/O port. Each pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs. Also, pull-up resistors and open-drain outputs can be assigned by software.

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

| Port pin | Alternate function |
|---|---|
| R41 | INT2 (External Interrupt 2 Request input) |
| R42 | $X_{OUT}$ |
| R43 | $X_{IN}$ |
| R44 | INT3 (External Interrupt 3 Request input) |
| R45 | $SX_{IN}$ |
| R46 | $SX_{OUT}$ |
| R47 | $\overline{RESET}$ |

**R50~R57:** R5 is an 8-bit CMOS bidirectional I/O port or LCD segment output. Each pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs. And each pins can also be set in segment output mode in 1-bit units by R5PSR Register.

| Port pin | Alternate function |
|---|---|
| R50 | SEG0 (Segment Output 0) |
| R51 | SEG1 (Segment Output 1) |
| R52 | SEG2 (Segment Output 2) |
| R53 | SEG3 (Segment Output 3) |
| R54 | SEG4 (Segment Output 4) |
| R55 | SEG5 (Segment Output 5) |
| R56 | SEG6 (Segment Output 6) |
| R57 | SEG7 (Segment Output 7) |

**R60~R67:** R6 is an 8-bit CMOS bidirectional I/O port or LCD segment output. Each pins 1 or 0 written to the Port Direction Register can be used as outputs or inputs. And each pins can also be set in segment output mode in 1-bit units by R6PSR Register.

| Port pin | Alternate function |
|---|---|
| R60 | SEG8 (Segment Output 8) |
| R61 | SEG9 (Segment Output 9) |
| R62 | SEG10 (Segment Output 10) |
| R63 | SEG11 (Segment Output 11) |
| R64 | SEG12 (Segment Output 12) |
| R65 | SEG13 (Segment Output 13) |
| R66 | SEG14 (Segment Output 14) |
| R67 | SEG15 (Segment Output 15) |

**R70~R77:** R7 is a 8-bit CMOS input port or LCD segment output. Each pins can be set in digital input or segment output mode in 1-bit units by R7PSR Register.

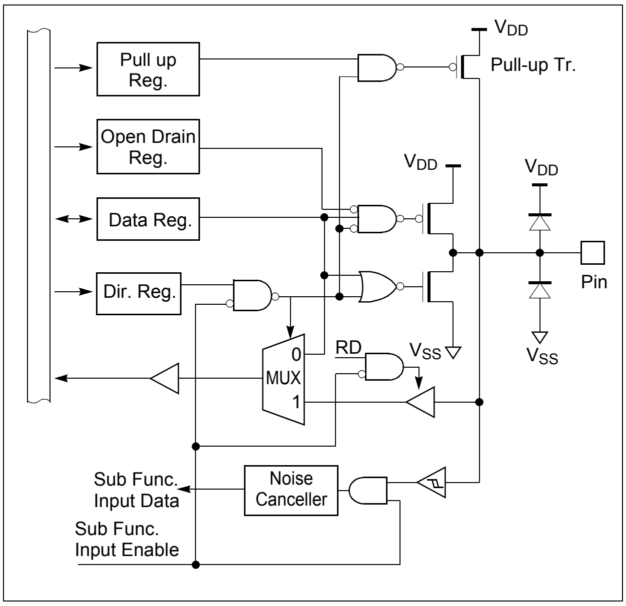
| Port pin | Alternate function |
|---|---|
| R70 | SEG16 (Segment Output 16) |
| R71 | SEG17 (Segment Output 17) |
| R72 | SEG18 (Segment Output 18) |
| R73 | SEG19 (Segment Output 19) |
| R74 | COM3 (Common Output 3) |
| R75 | COM2 (Common Output 2) |
| R76 | COM1 (Common Output 1) |
| R77 | COM0 (Common Output 0) |

| PIN NAME | Pin No. | | Primary Function | | Secondary Function | | State @ Reset | State @ STOP |
|---|---|---|---|---|---|---|---|---|
| | MC80F7708AQ | MC80F7708AK | I/O | Description | I/O | Description | | |
| $V_{DD}$ | 3 | 6 | - | Supply Voltage | - | - | - | - |
| $V_{SS}$ | 4 | 7 | - | Circuit Ground | - | - | - | - |
| $\overline{RESET}$ / R47 | 11 | 14 | I | Reset (low active) | - | - | 'L' input | 'H' input |
| $X_{IN}$/R43, $X_{OUT}$/R42 | 7,6 | 10,9 | I,O | Main clock oscilla-tor | - | - | Oscillation | 'L', 'H' |
| $SX_{IN}$/R45, $SX_{OUT}$/R46 | 9,10 | 12,13 | I,O | Sub clock oscillator | - | - | Oscillation | |
| R53/SEG3 ~ R57/SEG7 | 17~21 | 20~24 | I/O | General I/O port | O | LCD segment output | Input port | State of before STOP |
| R60/SEG8 ~ R61/SEG9 | 22~23 | 25~26 | I/O | General I/O port | O | LCD segment output | | |
| R62/SEG10 ~ R67/SEG15 | 24~29 | - | I/O | General I/O port | O | LCD segment output | | |
| R70/SEG16 ~ R73/SEG19 | 30~33 | - | I/O | General I/O ports | O | LCD segment output | | |
| R74/COM3 ~ R77/COM0 | 34~37 | 27~30 | I/O | General I/O ports | O | LCD common output | | |
| R01/EC0 | 44 | 3 | I/O | General I/O port | I | Event Counter Input | | |
| R04/BUZO | 2 | 5 | I/O | | O | Buzzer Output | | |
| R06/INT0 | 1 | 4 | I/O | | I | Interrupt Input | | |
| R07/INT1 | 12 | 15 | I/O | | I | Interrupt Input | | |
| R10/PWM1/ T2O | 13 | 16 | I/O | | O | Timer3 PWM Output Timer2 Output | | |
| R20/AN0 ~ R23/AN3 | 43~40 | 2,1,32,31 | I/O | | I | A/D Converter Analog Input | | |
| R24/AN4 ~ R25/AN5 | 39,38 | - | I/O | | I | A/D Converter Analog Input | | |
| R50/SEG0 | 14 | 17 | I/O | General I/O ports | O I | LCD Segment Output | | |
| R51/SEG1 | 15 | 18 | I/O | General I/O ports | O | LCD Segment Output | | |
| R52/SEG2 | 16 | 19 | I/O | General I/O ports | O | LCD Segment Output | | |

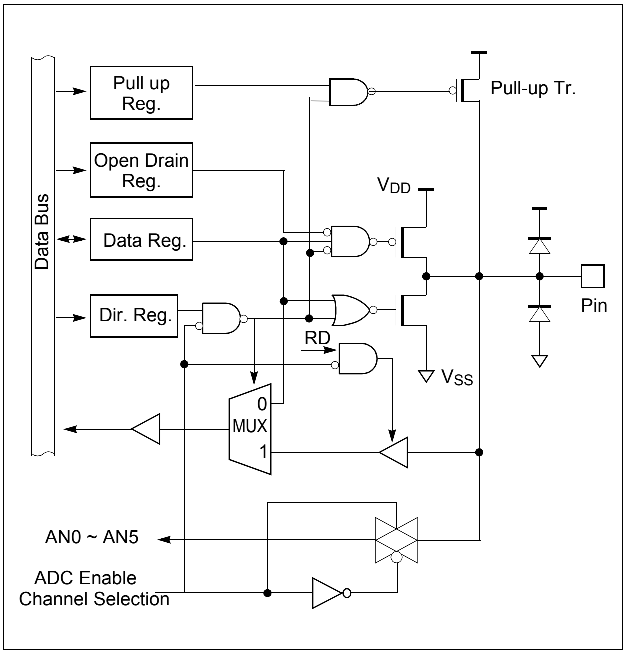
**Table 5-1 Port Function Description**

# 6. PORT STRUCTURES

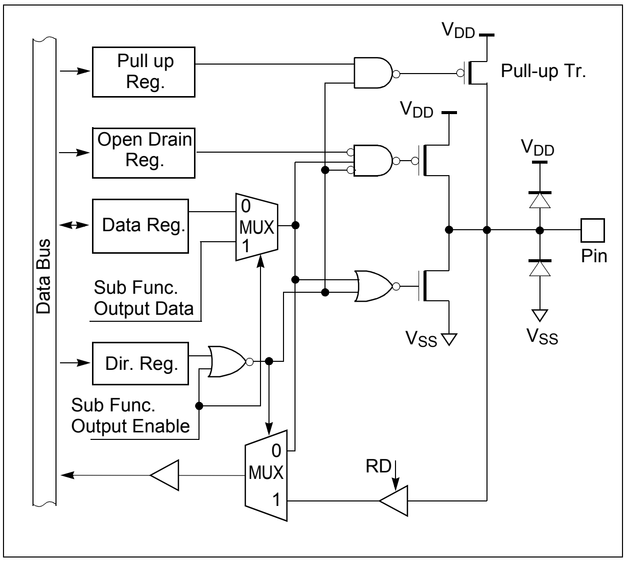
## R01/EC0, R06/INT0, R07/INT1, R41/IN2, R44/INT3




## R20/AN0~R23/AN3, R24/AN4~R25/AN5

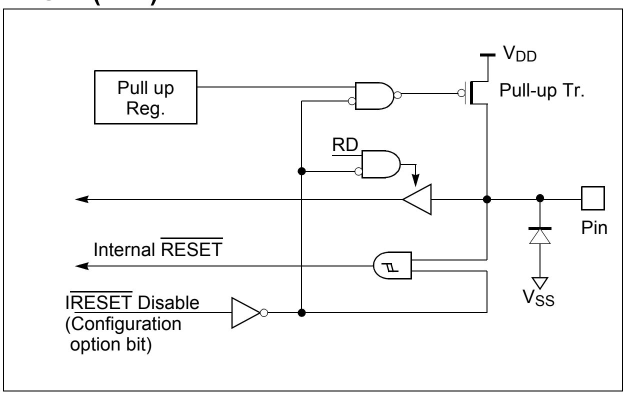



## R04/BUZO, R10/PWM1/T2O

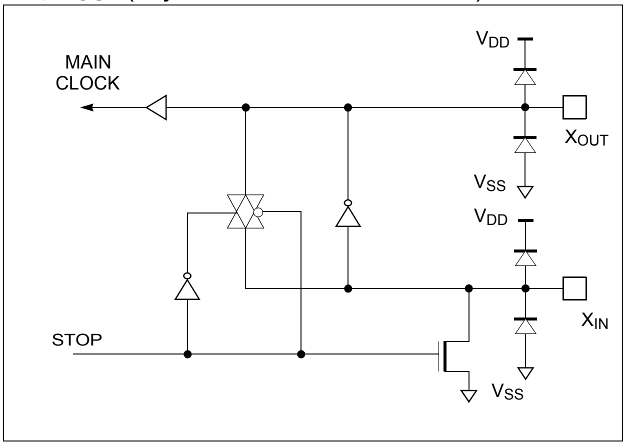



***Note:*** *R24/AN4 and R25/AN5 are available to MC80F7708AK/AD.*

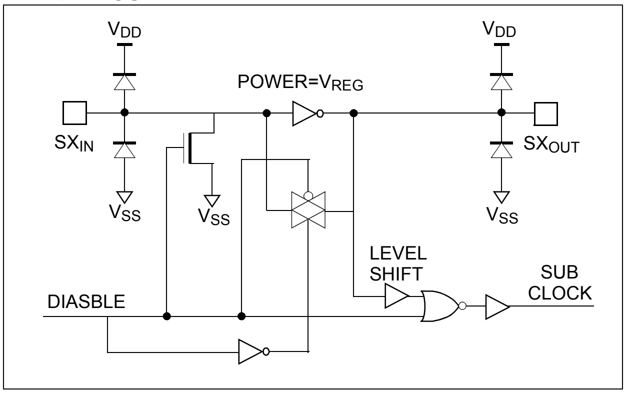
## RESET(R47)



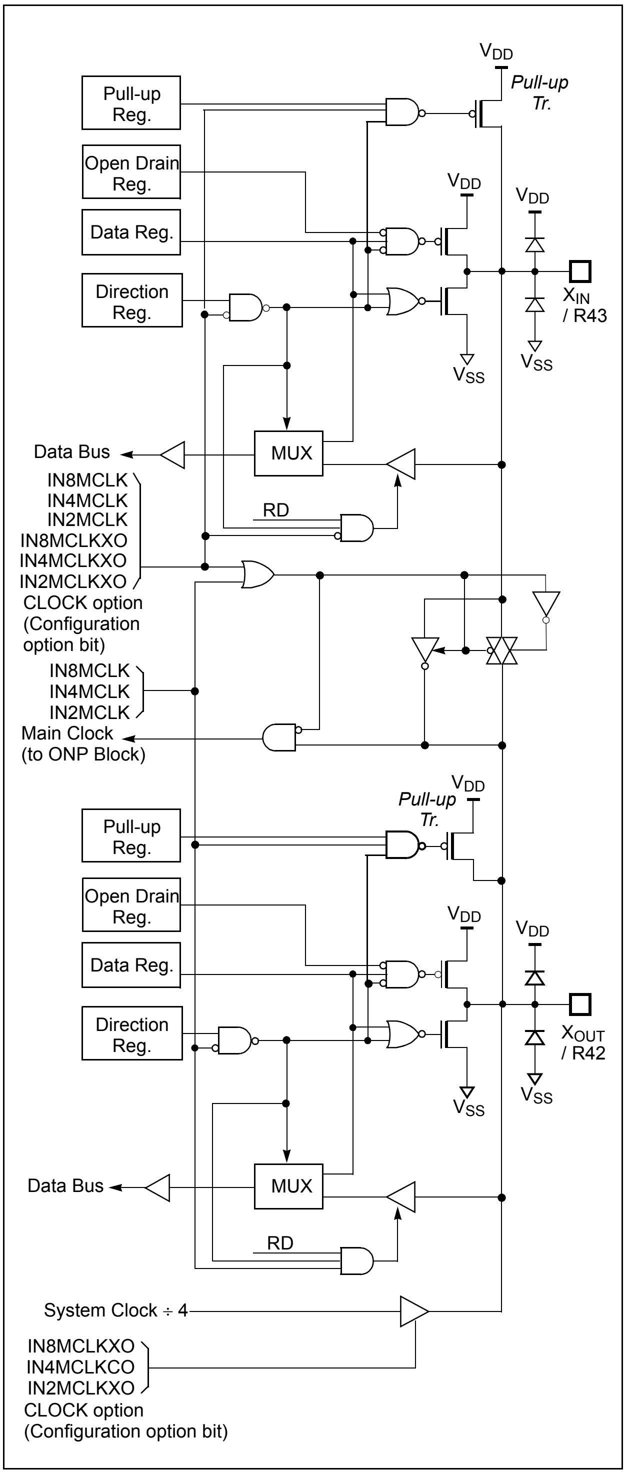
### $X_{IN}$, $X_{OUT}$ (Crystal or Ceramic resonator)




### $SX_{IN}$, $SX_{OUT}$



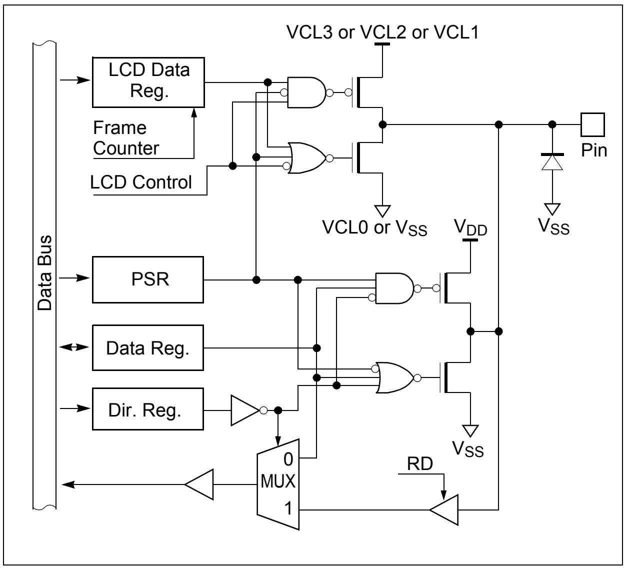

### R43 ($X_{IN}$), R42 ($X_{OUT}$)



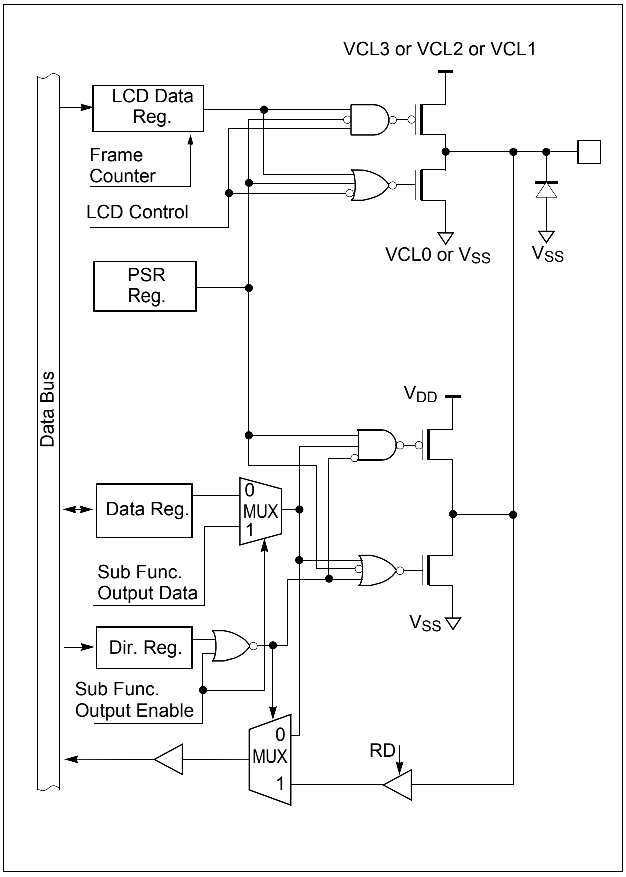
**R53/SEG3~R57/SEG7, R60/SEG8~R67/SEG15**

**R70/SEG16~R70/SEG19**

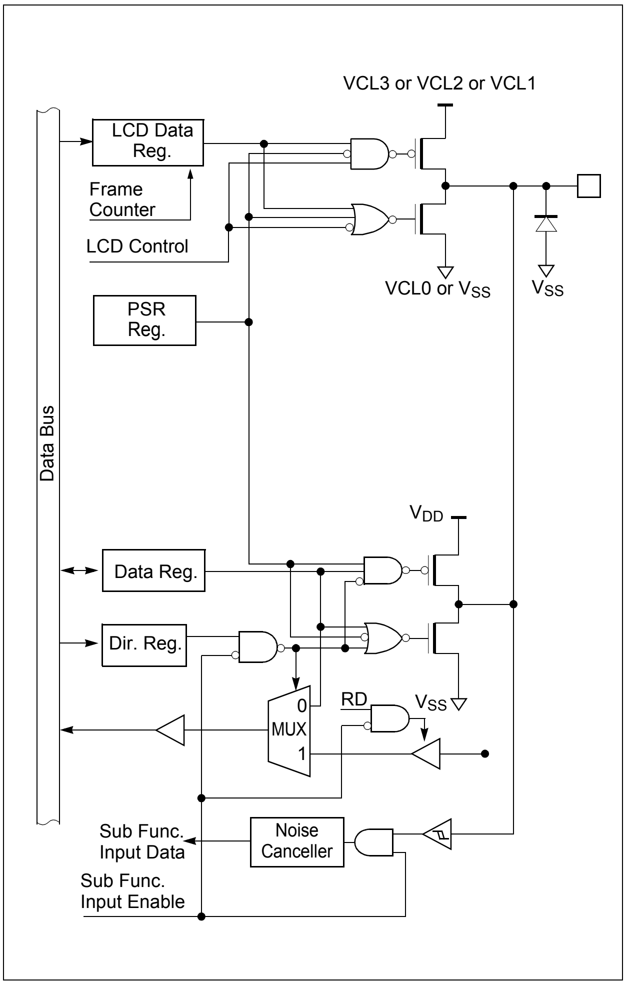
**R74/COM3~R77/COM0**




**R51/SEG1**




**R50/SEG0, R52/SEG2**



# 7. ELECTRICAL CHARACTERISTICS

## 7.1 Absolute Maximum Ratings

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

Storage Temperature ................................-45 to +125 °C

Voltage on any pin with respect to Ground ($V_{SS}$) ............................................................... -0.3 to $V_{DD}$+0.3

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

Maximum output current sourced by ($I_{OH}$ per I/O Pin) ...............................................................................15 mA

Maximum current ($\Sigma I_{OL}$) ...................................100 mA

Maximum current ($\Sigma I_{OH}$).....................................60 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. | Typ. | Max. | |
| Supply Voltage | $V_{DD}$ | $f_{MAIN}$=4MHz | 2.2 | - | 5.5 | V |
| Main Operating Frequency | $f_{MAIN}$ | $V_{DD}$[1]=2.2~5.5V | 0.4 | - | 4.0 | MHz |
| | | $V_{DD}$=4.5~5.5V | 0.4 | - | 12.0 | |
| | $f_{SUB}$ | $V_{DD}$=3.0~5.5V | - | 32.768 | - | kHz |
| Sub Operating Frequency | $f_{SUB}$ | $V_{DD}$=$V_{DD}$ | - | 32.768 | - | kHz |
| Operating Temperature | $T_{OPR}$ | | -40 | - | 85 | °C |

1. FLASH MCU Operating Voltage.

### 7.3 DC Electrical Characteristics

($T_A$= -40~85°C, $V_{DD}$=2.2~5.5V, $V_{SS}$=0V)

| Parameter | Symbol | Pin / Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Input High Voltage | $V_{IH1}$ | R0~R7 | $0.7V_{DD}$ | - | $V_{DD}+0.3$ | V |
| | $V_{IH2}$ | $\overline{RESET}$, $X_{IN}$, $SX_{IN}$, INT0~3, EC0 | $0.8V_{DD}$ | - | $V_{DD}+0.3$ | |
| Input Low Voltage | $V_{IL1}$ | R0~R7 | -0.3 | - | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | $\overline{RESET}$, $X_{IN}$, $SX_{IN}$, INT0~3, EC0 | -0.3 | - | $0.2V_{DD}$ | |
| Output High Voltage | $V_{OH1}$ | R0~R4 ($V_{DD}$=4.5V, $I_{OH1}$=-1.6mA) | $V_{DD}$-0.3 | - | - | V |
| | $V_{OH2}$ | R5~R7 ($V_{DD}$=4.5V, $I_{OH2}$=-1.6mA) | $V_{DD}$-1.0 | - | - | |
| | $V_{OH3}$[1] | SEG0~19, COM0~3 ($V_{DD}$=4.5V, VCL3~0=3V, $I_{OH3}$=-15μA) | VCL3-0.4 | - | - | |
| Output Low Voltage | $V_{OL1}$ | R0~R4 ($V_{DD}$=4.5V, $I_{OL1}$=1.6mA) | - | - | 0.35 | V |
| | $V_{OL2}$ | R5~R7 ($V_{DD}$=4.5V, $I_{OL2}$=1.6mA) | | | 0.4 | |
| | $V_{OL3}$[2] | SEG0~19, COM0~3 ($V_{DD}$=4.5V, VCL3~0=3V, $I_{OL3}$=15μA) | | | 0.12 | |
| Input High Leakage Current | $I_{IH}$ | All input pins including R5~R7 ($V_{IN}$=$V_{DD}$) | - | - | 1 | μA |
| Input Low Leakage Current | $I_{IL}$ | All input pins including R5~R7 ($V_{IN}$=$V_{ss}$) | -1 | - | - | |
| LVR (Low Voltage Reset) | $V_{LVR}$ | SVLD[2:0]=001 | TBD | 2.4 | TBD | V |
| | | SVLD[2:0]=010 | TBD | 2.7 | TBD | |
| | | SVLD[2:0]=011 | TBD | 3.0 | TBD | |
| | | SVLD[2:0]=100 | TBD | 3.9 | TBD | |
| Power Supply Voltage rising time | $T_R$ | $V_{DD}$=5V (4μS/V) | 20 | - | - | μS |
| LVR Circuit Current consumption | $I_{LVR}$ | $V_{DD}$=5V | - | 3 | - | μA |
| POR(Power on Reset) Level | $V_{POR}$ | $V_{DD}$ ($T_A$=25°C) | 2.4 | 2.7 | 2.8 | V |
| | | | 1.7 | 1.9 | 2.1 | |
| POR Start Voltage[3] | $V_{START}$ | $V_{DD}$ ($T_A$=25°C) | | | 1.8 | V |
| POR Rising Time[3] | $T_{POR}$ | $V_{DD}$ ($T_A$=25°C) | | | 40 | ms/V |
| VDD rising Time[3] | $T_{VDD}$ | $V_{DD}$ ($T_A$=25°C) | | | 40 | ms/V |
| Hysteresis | VT+ ~ VT- | $\overline{RESET}$, INT0~3, EC0 ($V_{DD}$=5V) | $0.2V_{DD}$ | - | $0.8V_{DD}$ | V |
| Pull-up Current | $I_{PU}$ | R0~R4 ($V_{DD}$=3.0V, $V_{PIN}$=0V) | 20 | - | 60 | μA |
| Current dissipation in active mode[4] | $I_{DD}$ | $V_{DD}$( $f_{MAIN}$=12MHz, $V_{DD}$=5.5V, $f_{SUB}$=0 ) | - | 5 | 15 | mA |
| Current dissipation in sleep mode[5] | $I_{SLEEP}$ | $V_{DD}$ ( $f_{MAIN}$=12MHz, $V_{DD}$=5.5V, $f_{SUB}$=0 ) | - | 2 | 4 | |
| Current dissipation in sub active mode[6] | $I_{subactive}$ | $f_{MAIN}$=off, $V_{DD}$=3V, $f_{SUB}$=32.768kHz | - | 35 | 50 | μA |
| | $I_{subsleep}$ | $f_{MAIN}$=off, $V_{DD}$=3V, $f_{SUB}$=32.768kHz | | 7 | 14 | uA |

| Parameter | Symbol | Pin / Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Current dissipation in stop mode | $I_{STOP}$ | $f_{MAIN}$=off, $V_{DD}$=5.5V, $f_{SUB}$=0 | - | 3 | 7 | µA |
| | $I_{SUB}$ | $f_{MAIN}$=off, $V_{DD}$=5.5V, $f_{SUB}$=32.768kHz | | 10 | TBD | uA |
| Internal 8MHz Oscillation Frequency | $f_{IN8M}$ | VDD=5V, $T_A$=25°C | TBD | 8 | TBD | MHz |
| Internal 4MHz Oscillation Frequency | $f_{IN4M}$ | VDD=5V, $T_A$=25°C | TBD | 4 | TBD | MHz |
| Internal 2MHz Oscillation Frequency | $f_{IN2M}$ | VDD=5V, $T_A$=25°C | TBD | 2 | TBD | MHz |

1. $V_{OH3}$ is the voltage when VCL3, VCL2, VCL1 and VCL0 are supplied at pads.
2. $V_{OL3}$ is the voltage when $V_{SS}$ is supplied at pad.
3. These parameters are presented for design guidance only and not tested or guaranteed.
4. Current dissipation is proportioned according to operation voltage and frequency.
5. In sleep mode, oscillation continues and peripherals are operated normally but internal CPU clock stops.
6. In sub sleep mode, sub oscillation continues and peripherals are operated normally but internal CPU clock stops.

### 7.4 LCD Characteristics

($T_A$= -40~85°C, $V_{DD}$=2.2~5.5V, $V_{SS}$=0V)

| Parameter | Symbol | Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| LCD Common Output Current | $I_{COM}$ | Output Voltage Deviation=0.2V | 30 | - | - | µA |
| LCD Segment Output Current | $I_{SEG}$ | Output Voltage Deviation=0.2V | 5 | - | – | |

### 7.5 A/D Converter Characteristics

(TA=-40~85°C, $AV_{DD}$=$V_{DD}$=5.12V,2.56, $V_{SS}$=0V)

| Parameter | Symbol | Condition | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Analog Power Supply Input Voltage Range | $AV_{DD}$ | - | $V_{DD}$ | - | $V_{DD}$ | V |
| Analog Input Voltage Range | $V_{AN}$ | - | $V_{SS}$-0.3 | - | $AV_{DD}$+0.3 | |
| Current Following Between $AV_{DD}$ and $AV_{SS}$ | $IAV_{DD}$ | - | - | - | 200 | µA |
| Overall Accuracy | CAIN | - | - | ±1.0 | ±2.0 | LSB |
| Non Linearity Error | NNLE | - | - | ±1.0 | ±2.0 | |
| Differential Non Linearity Error | NDNLE | - | - | ±1.0 | ±2.0 | |
| Zero Offset Error | NZOE | - | - | ±0.5 | ±1.5 | |
| Full Scale Error | NFSE | - | - | ±0.25 | ±0.5 | |
| Gain Error | NGE | - | - | ±1.0 | ±1.5 | |
| Conversion Time | TCONV | $f_{MAIN}$=8MHz | - | - | 10 | µS |
| | | $f_{MAIN}$=4MHz | - | - | 20 | |

### 7.6 AC Characteristics

($TA=25°C$, $V_{DD}=4V$, $AV_{DD}=4V$, $V_{SS}=AV_{SS}=0V$)

| Parameter | Symbol | Pins | Specifications | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Main Operating Frequency | $f_{MCP}$ | $X_{IN}$ | 0.4 | - | 12 | MHz |
| Sub Operating Frequency | $f_{SCP}$ | $SX_{IN}$ | 30 | 32.768 | 35 | kHz |
| System Clock Frequency[1] | $t_{SYS}$ | - | 166 | - | 5000 | nS |
| Main Oscillation Stabilization Time (4MHz) | $t_{MST}$ | $X_{IN}$, $X_{OUT}$ | - | - | 20 | mS |
| Sub Oscillation Stabilization Time | $t_{SST}$ | $SX_{IN}$, $SX_{OUT}$ | - | 1 | 2 | S |
| External Clock “H” or “L” Pulse Width | $t_{MCPW}$ | $X_{IN}$ | 35 | - | - | nS |
| | $t_{SCPW}$ | $SX_{IN}$ | 5 | - | - | μS |
| External Clock Transition Time | $t_{RCP}$, $t_{FCP}$ | $X_{IN}$ | - | - | 20 | nS |
| Interrupt Pulse Width | $t_{IW}$ | INT0, INT1, INT2, IN3 | 2 | - | - | $t_{SYS}$ |
| $\overline{RESET}$ Input Pulse “L” Width | $t_{RST}$ | $\overline{RESET}$ | 8 | - | - | $t_{SYS}$ |
| Event Counter Input “H” or “L” Pulse Width | $t_{ECW}$ | EC0 | 2 | - | - | $t_{SYS}$ |
| Event Counter Transition Time | tREC, tFEC | EC0 | - | - | 20 | nS |

1.SCMR=XXXX000$X_B$ that is $f_{MAIN}÷2$

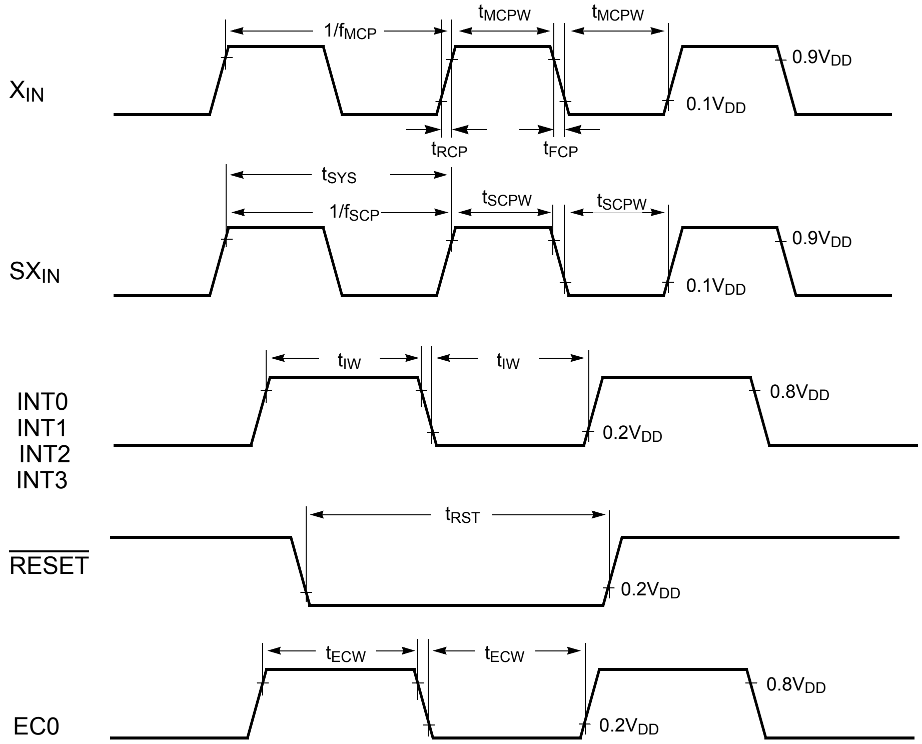



**Figure 7-1 AC Timing Chart**

## 7.7 Typical Characteristics

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

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

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

# 8. MEMORY ORGANIZATION

The have separate address spaces for Program memory, Data Memory and Display memory. Program memory can only be read, not written to. It can be up to 8K bytes of Program memory. Data memory can be read and written to up to 1024 bytes including the stack area. Display memory has prepared 27 nibbles for LCD.

## 8.1 Registers

This device has six registers that are the Program Counter (PC), a Accumulator (A), two index registers (X, Y), the Stack Pointer (SP), and the Program Status Word (PSW). The Program Counter consists of 16-bit register.

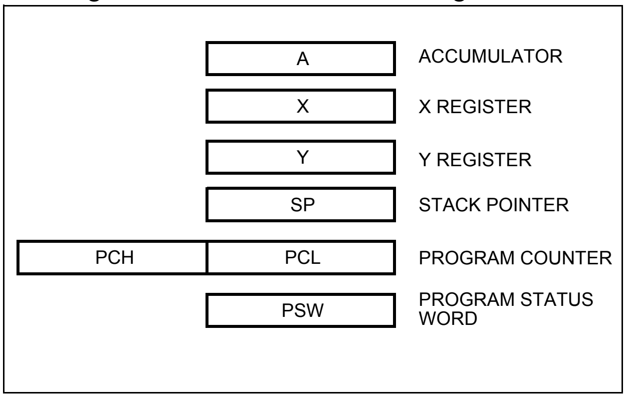



**Figure 8-1 Configuration of Registers**

**Accumulator:** The Accumulator is the 8-bit general purpose register, used for data operation such as transfer, temporary saving, and conditional judgement, etc.

The Accumulator can be used as a 16-bit register with Y Register as shown below.

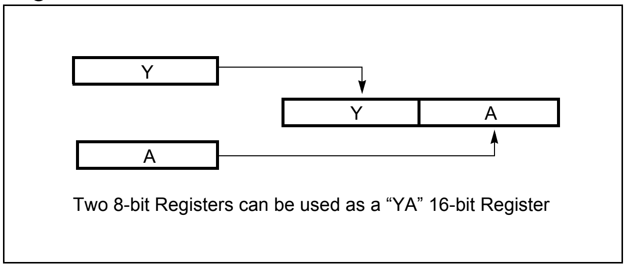


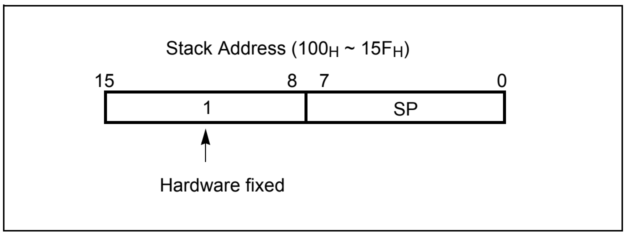

**Figure 8-2 Configuration of YA 16-bit Register**

**X, Y Registers:** In the addressing mode which uses these index registers, the register contents are added to the specified address, which becomes the actual address. These modes are extremely effective for referencing subroutine tables and memory tables. The index registers also have increment, decrement, comparison and data transfer functions, and they can be used as simple accumulators.

**Stack Pointer:** The Stack Pointer is an 8-bit register used for occurrence interrupts and calling out subroutines. Stack Pointer identifies the location in the stack to be accessed (save or restore).

Generally, SP is automatically updated when a subroutine call is executed or an interrupt is accepted. However, if it is used in excess of the stack area permitted by the data memory allocating configuration, the user-processed data may be lost.

The stack can be located at any position within $100_H$ to $15F_H$ 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 “$15F_H$” is used.




***Caution:***

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

Example: To initialize the SP

```
LDX    #05FH      ;
TXSP              ;SP ← 05FH
```

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

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

[Carry flag C]

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

[Zero flag Z]

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

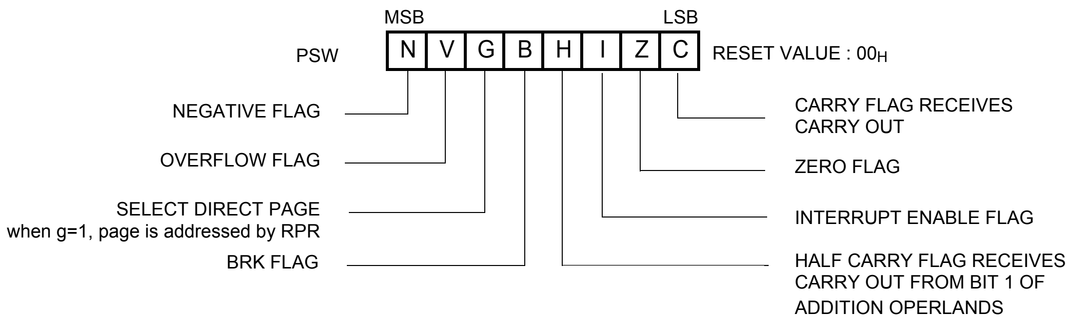



**Figure 8-3 PSW (Program Status Word) Register**

[Interrupt disable flag I]

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

[Half carry flag H]

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

[Break flag B]

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

[Direct page flag G]

This flag assigns RAM page for direct addressing mode. In the direct addressing mode, addressing area is from zero page $00_H$ to $0FF_H$ when this flag is "0". If it is set to "1", addressing area is assigned by RPR register (address $0F3_H$). It is set by SETG instruction and cleared by CLRG.

[Overflow flag V]

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

[Negative flag N]

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

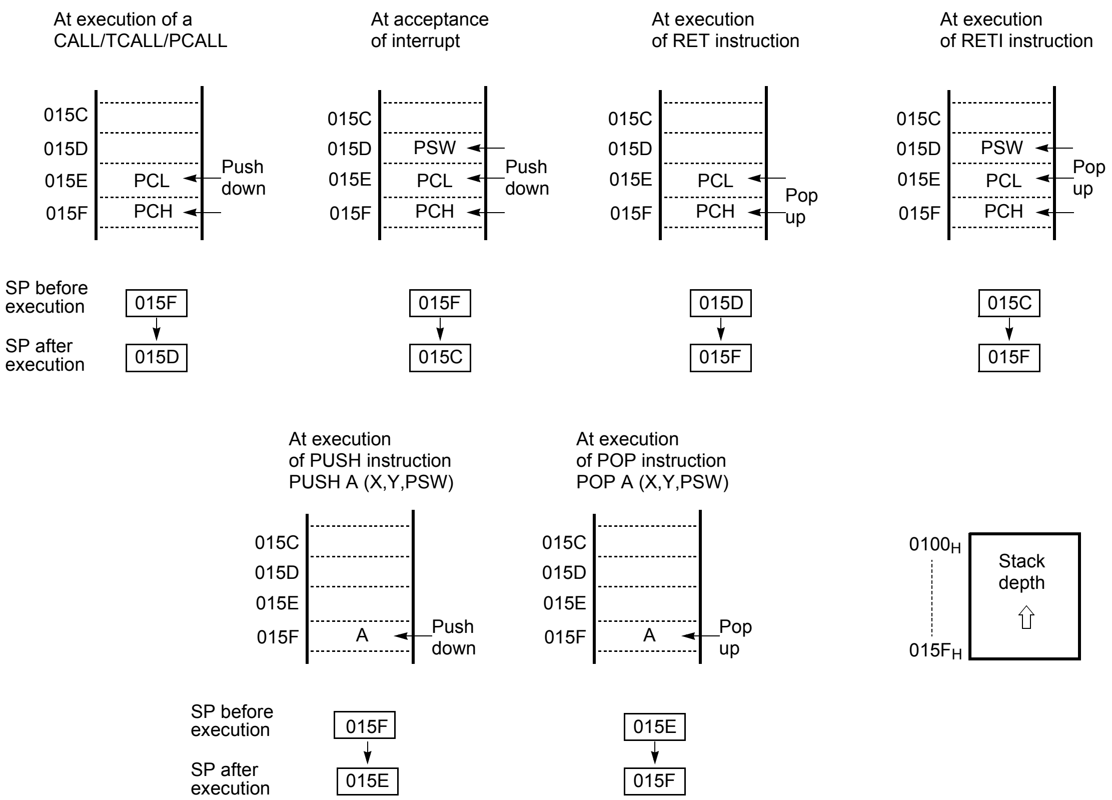



**Figure 8-4 Stack Operation**

## 8.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but this device has 8K 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.

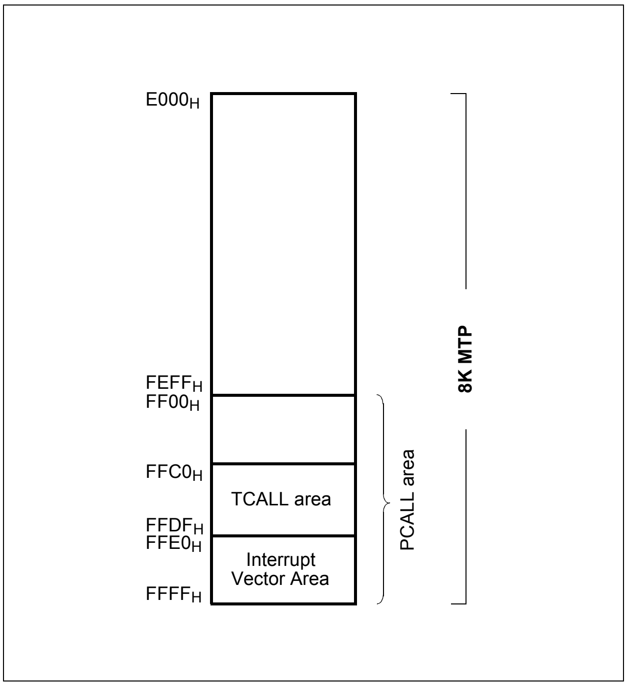



**Figure 8-5 Program Memory Map**

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 |
|---|---|
| $0FFE0_H$ | Watch Timer interrupt Vector Area<br>Watch dog timer interrupt Vector |
| E2 | Basic Interval Timer Interrupt Vector Area |
| E4 | - |
| E6 | - |
| E8 | Timer/Counter 3 Interrupt Vector Area |
| EA | Timer/Counter 2 Interrupt Vector Area |
| EC | Timer/Counter 1 Interrupt Vector Area |
| EE | Timer/Counter 0 Interrupt Vector Area |
| F0 | - |
| F2 | -<br>- |
| F4 | External Interrupt 3 Vector Area |
| F6 | External Interrupt 2 Vector Area |
| F8 | External Interrupt 1 Vector Area |
| FA | External Interrupt 0 Vector Area |
| FC | - |
| FE | Reset Interrupt Vector Area |

**Figure 8-6 Interrupt Vector Area**

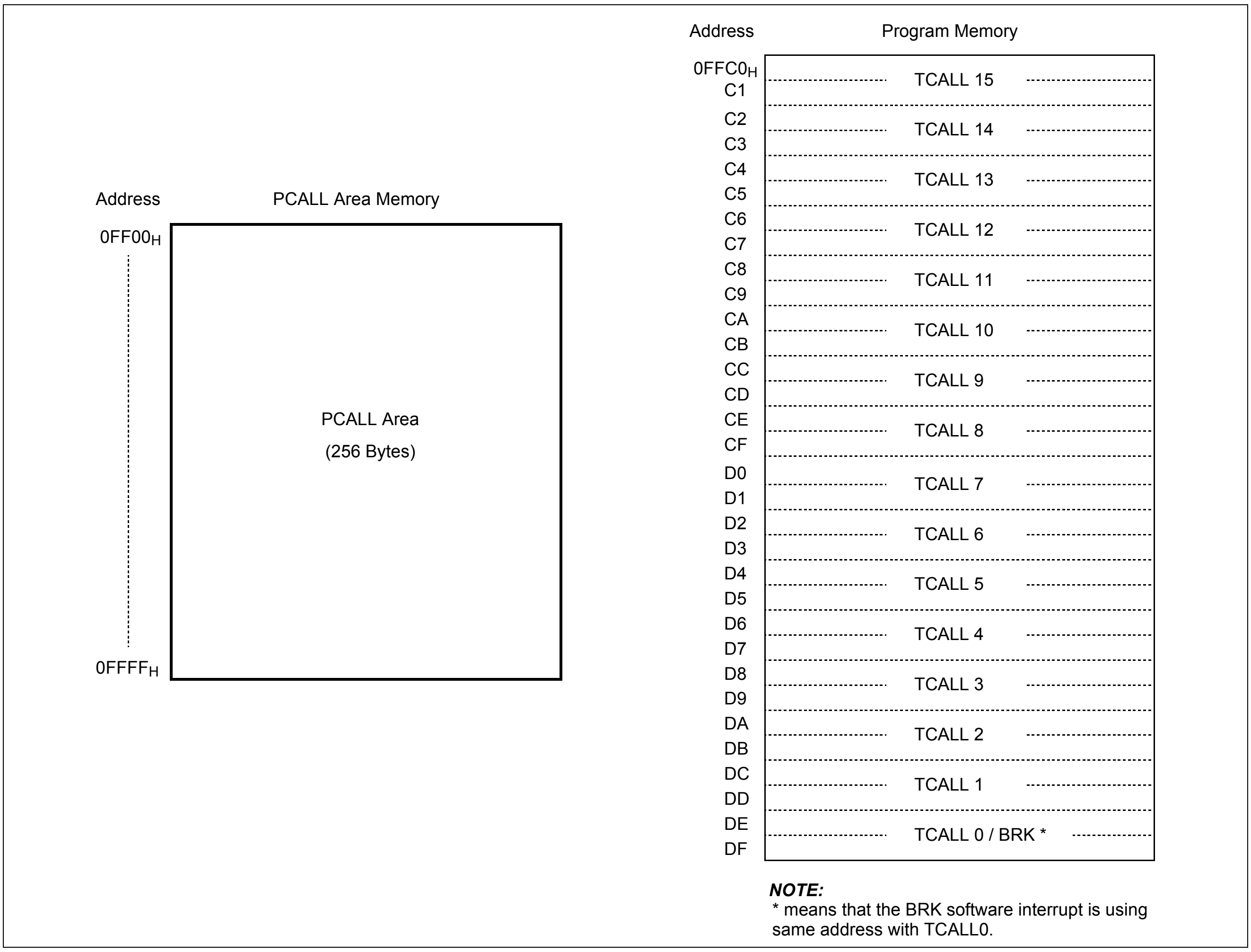
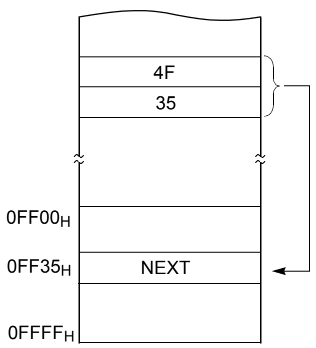



**Figure 8-7 PCALL and TCALL Memory Area**

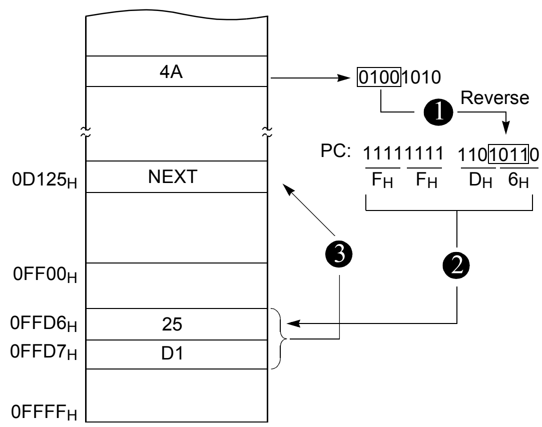
**PCALL→ rel**

```
4F35    PCALL  35H
```




**TCALL→ n**

```
4A      TCALL  4
```



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

```
        ORG  0FFE0H; Device : MC807708A

        DW   WT_INT        ; Watch Timer / Watch Dog Timer
        DW   BIT_INT       ; Basic Interval Timer
        DW   NOT_USED      ; NOT_USED
        DW   NOT_USED      ; Not Used
        DW   TMR3_INT      ; Timer-3
        DW   TMR2_INT      ; Timer-2
        DW   TMR1_INT      ; Timer-1
        DW   TMR0_INT      ; Timer-0
        DW   NOT_USED      ; Not Used
        DW   NOT_USED      ; Not Used
        DW   EX3_INT       ; INT.3
        DW   EX2_INT       ; INT.2
        DW   EX1_INT       ; Int.1
        DW   EX0_INT       ; Int.0
        DW   NOT_USED      ; Not used
        DW   RESET         ; Reset

;********************************************
;           MAIN     PROGRAM               *
;********************************************

        ORG  0E000H

RESET:  DI                 ;Disable All Interrupts
        CLRG
        LDX  #0
        LDA  #0
RAM_CLR1:STA {X}+          ;Page0 RAM Clear(!0000H->!009FH)
        CMPX #0A0H
        BNE  RAM_CLR1

        LDM  RPR,#0000_0001B;Page1 RAM Clear(!0100H->!015FH)
        SETG
        LDX  #0
        LDA  #0
RAM_CLR2:STA {X}+
        CMPX #060H
        BNE  RAM_CLR2
        CLRG

        LDX  #05FH         ;Stack Pointer Initialize
        TXSP

        LDM  RPR,#0000_0000B;Page0 selection

        CALL LCD_CLR       ;Clear LCD display memory

        LDM  R0, #0        ;Normal Port 0
        LDM  R0IO,#1000_0010B;Normal Port Direction
        LDM  R0PU,#1000_0010B;Pull Up Selection Set
        LDM  R0OD,#0000_0001B;R0 port Open Drain control
        :
        :
        LDM  SCMR,#1111_0000B;System clock control
        :
        :
```

## 8.3 Data Memory

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

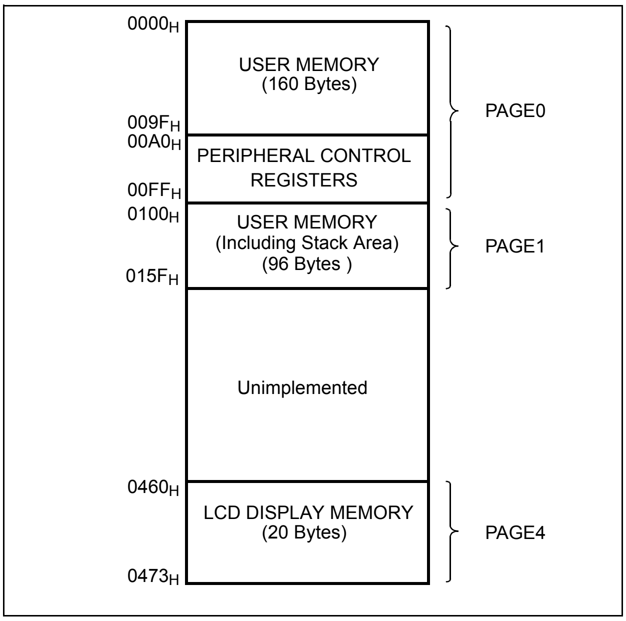



**Figure 8-8 Data Memory Map**

### User Memory

The MC80F7708A have 256 × 8 bits for the user data memory (RAM). There are three pages internal RAM. Page is selected by G-flag and RAM page selection register RPR. When G-flag is cleared to "0", always page 0 is selected regardless of RPR value. If G-flag is set to "1", page will be selected according to RPR value.

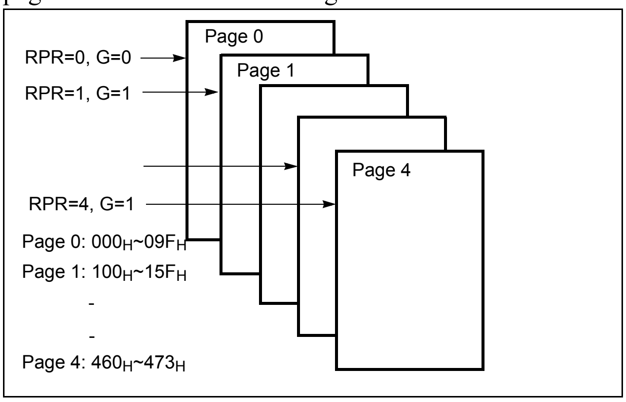
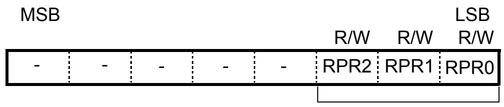



**Figure 8-9 RAM page configuration**

**RPR (RAM Page Selection Register)**

ADDRESS: $0CF_H$
INITIAL VALUE: ----_-$001_B$




**RPR[2:0] (RAM Page Selection)**

| RPR2 | RPR1 | RPR0 | RAM Page Selection |
|---|---|---|---|
| 0 | 0 | 0 | PAGE 0 |
| 0 | 0 | 1 | PAGE 1 |
| 0 | 1 | 0 | - |
| 0 | 1 | 1 | - |
| 1 | 0 | 0 | PAGE 4 |
| 1 | 0 | 1 | - |
| 1 | 1 | 0 | - |
| 1 | 1 | 1 | - |

Caution1 : After setting RPR, be sure to execute SETG instruction.
Caution2 : When executing CLRG instruction, be selected PAGE0 regardless of RPR.

**Figure 8-10 RAM Page Selection Register**

### 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 $0A0_H$ to $0FF_H$.

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

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

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

Example; To write at CKCTLR

```
LDM    CKCTLR,#05H ;Divide ratio ÷8
```

### Stack Area

The stack provides the area where the return address is saved before a jump is performed during the processing routine at the execution of a subroutine call instruction or the acceptance of an interrupt.

When returning from the processing routine, executing the subroutine return instruction [RET] restores the contents of the program counter from the stack; executing the interrupt return instruction [RETI] restores the contents of the program counter and flags.

The save/restore locations in the stack are determined by the stack pointer (SP). The SP is automatically decreased after the saving, and increased before the restoring. This means the value of the SP indicates the stack location number for the next save. Refer to Figure 8-4 on page 24.

### LCD Display Memory

LCD display data area is handled in LCD section.

See "18.3 LCD Display Memory" on page 82.

| Address | Register Name | Symbol | R/W | Initial Value | | | | | | | | Addressing Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 00A0H | R0 Open Drain Control Register | R0OD | W | 0 | 0 | - | 0 | - | - | 0 | - | byte[1] |
| 00A1H | R1 Open Drain Control Register | R1OD | W | - | - | - | - | - | - | - | 0 | byte |
| 00A2H | R2Open Drain Control Register | R2OD | W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00A4H | R4Open Drain Control Register | R4OD | W | - | 0 | 0 | 0 | 0 | 0 | 0 | - | byte |
| 00A5H | R0 Pull-up Register | R0PU | W | 0 | 0 | - | 0 | - | - | 0 | - | byte |
| 00A6H | R1 Pull-up Register | R1PU | W | - | - | - | - | - | - | - | 0 | byte |
| 00A7H | R2 Pull-up Register | R2PU | W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00A9H | R4 Pull-up Register | R2PU | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | byte |
| 00AAH | Port Selection Register 0 | PSR0 | W | 0 | 0 | - | 0 | - | - | 0 | - | byte |
| 00ABH | Port Selection Register 1 | PSR1 | W | - | - | - | - | - | 0 | 0 | 0 | byte |
| 00ACH | R5 Port Selection Register | R5PSR | R / W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte, bit[2] |
| 00ADH | R6 Port Selection Register | R6PSR | R / W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte, bit |
| 00AEH | R7 Port Selection Register | R7PSR | R / W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte, bit |
| 00B0H | R7 Data Register | R7 | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00B2H | LCD Control Register | LCR | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00B3H | LCD BIAS Control Register | LBCR | R / W | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | byte, bit |
| 00B4H | R7 Direction Register | R7IO | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00B8H | - | ASIMR0 | R / W | 0 | 0 | 0 | 0 | - | 0 | 0 | - | byte, bit |
| 00B9H | - | ASISR0 | R | - | - | - | - | - | 0 | 0 | 0 | byte |
| 00BAH | - | BRGCR0 | R / W | - | 0 | 0 | 1 | 0 | 0 | 0 | 0 | byte, bit |
| 00BBH | - | RXBR0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| | - | TXSR0 | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| 00C0H | R0 port data register | R0 | R / W | 0 | 0 | - | 0 | - | - | 0 | - | byte, bit |
| 00C1H | R0 Direction Register | R0IO | W | 0 | 0 | - | 0 | - | - | 0 | - | byte |
| 00C2H | R1 port data register | R1 | R / W | - | - | - | - | - | - | - | 0 | byte, bit |
| 00C3H | R1 Direction Register | R1IO | W | - | - | - | - | - | - | - | 0 | byte |
| 00C4H | R2 port data register | R2 | R / W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00C5H | R2 Direction Register | R2IO | W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00C8H | R4 port data register | R2 | R / W | - | 0 | 0 | 0 | 0 | 0 | 0 | - | byte, bit |

**Table 8-1 Control Registers**

| Address | Register Name | Symbol | R/W | Initial Value | | | | | | | | Addressing Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 00C9H | R4 Direction Register | R4IO | W | - | 0 | 0 | 0 | 0 | 0 | 0 | - | byte |
| 00CAH | R5 port data register | R5 | R/W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00CBH | R5 Direction Register | R5IO | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00CCH | R6 port data register | R6 | R/W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00CDH | R6 Direction Register | R6IO | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00CEH | Buzzer Driver Register | BUZR | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| 00CFH | Ram Page Selection Register | RPR | R / W | - | - | - | - | - | 0 | 0 | 1 | byte, bit |
| 00D0H | Timer 0 Mode Control Register | TM0 | R / W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00D1H | Timer 0 Register | T0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| | Timer 0 Data Register | TDR0 | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| | Timer 0 Capture Data Register | CDR0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00D2H | Timer 1 Mode Control Register | TM1 | R / W | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00D3H | Timer 1 Data Register | TDR1 | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| 00D4H | Timer 1 Register | T1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| | Timer 1 Capture Data Register | CDR1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00D6H | Timer 2 Mode Control Register | TM2 | R / W | - | - | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00D7H | Timer 2 Register | T2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| | Timer 2 Data Register | TDR2 | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| | Timer 2 Capture data Register | CDR2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00D8H | Timer 3 Mode Control Register | TM3 | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00D9H | Timer 3 Data Register | TDR3 | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| | Timer 3 PWM Period Register | T3PPR | W | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| 00DAH | Timer 3 Register | T3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| | Timer 3 PWM Duty Register | T3PDR | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| | Timer 3 Capture Data Register | CDR3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |
| 00DBH | Timer 3 PWM High Register | T3PWHR | W | - | - | - | - | 0 | 0 | 0 | 0 | byte |
| 00E2H | 8bit A/D Converter Mode Control Register | ADCM[3] | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | byte, bit |
| 00E3H | 8bit A/D Converter Result Register Low | ADCRL | R | Undefined | | | | | | | | byte |
| 00E4H | - | ADCRH | W, R | 0 | 1 | 0 | - | - | - | X | X | byte, bit |
| 00E5H | Supply Voltage Level Detector Register | SVLDR | R / W | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00E6H | Basic Interval Timer Register | BITR | R | Undefined | | | | | | | | byte |
| | Clock Control Register | CKCTLR | W | - | - | 0 | 1 | 0 | 1 | 1 | 1 | byte |
| 00E7H | System Clock Mode Register | SCMR | R / W | - | - | - | - | - | 0 | 0 | 0 | byte |
| 00E8H | Watch Dog Timer Register | WDTR | W | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| | Watch Dog Timer Data Register | WDTDR | R | Undefined | | | | | | | | byte |
| | Watch Timer Register | WTR | W | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | byte |
| 00E9H | Stop & Sleep Mode Control Register | SSCR | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte |

**Table 8-1 Control Registers**

| Address | Register Name | Symbol | R/W | Initial Value | | | | | | | | Addressing Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 00EAH | Watch Timer Mode Register | WTMR | R / W | 0 | 0 | - | - | 0 | 0 | 0 | 0 | byte, bit |
| 00F4H | - | INTFH | R / W | - | - | - | - | 0 | 0 | - | - | byte, bit |
| 00F5H | Interrupt Generation Flag Register Low | INTFL | R / W | - | - | - | - | - | - | 0 | 0 | byte, bit |
| 00F6H | - | IENH | R / W | - | 0 | 0 | 0 | 0 | 0 | - | - | byte, bit |
| 00F7H | Interrupt Enable Register Middle | IENM | R / W | 0 | 0 | 0 | 0 | - | - | - | - | byte, bit |
| 00F8H | Interrupt Enable Register Low | IENL | R / W | - | 0 | 0 | 0 | 0 | - | - | - | byte, bit |
| 00F9H | - | IRQH | R / W | - | 0 | 0 | 0 | 0 | 0 | - | - | byte, bit |
| 00FAH | Interrupt Request Register Middle | IRQM | R / W | 0 | 0 | 0 | 0 | - | - | - | - | byte, bit |
| 00FBH | Interrupt Request Register Low | IRQL | R / W | - | 0 | 0 | 0 | 0 | - | - | - | byte, bit |
| 00FCH | Interrupt Edge Selection Register | IEDS | R / W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |
| 00FFH | Watch timer Counter high read Register | WTRH | R | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | byte, bit |

**Table 8-1 Control Registers**

1. “byte”, “bit” means that register can be addressed by not only bit but byte manipulation instruction.
2. “byte” means that register can be addressed by only byte manipulation instruction. On the other hand, do not use any read-modify-write instruction such as bit manipulation.
3. bit 0 of ADCM is read only.

## 8.4 Addressing Mode

The MC80F7708A use six addressing modes;

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

### (1) Register Addressing

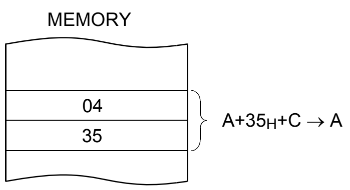
Register addressing accesses the A, X, Y, C and PSW.

### (2) Immediate Addressing → #imm

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

Example:

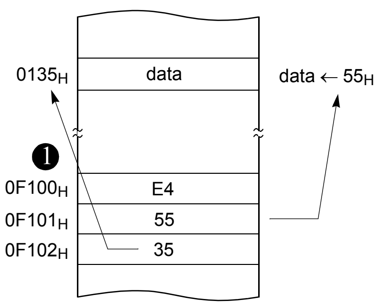
0435 ADC #$35_H$




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

Example: G=1, RPR=$01_H$

E45535 LDM $35_H$,#$55_H$

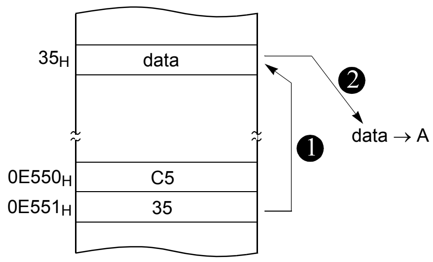



### (3) Direct Page Addressing → dp

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

Example; G=0

C535 LDA $35_H$ ;A ←RAM[$35_H$]



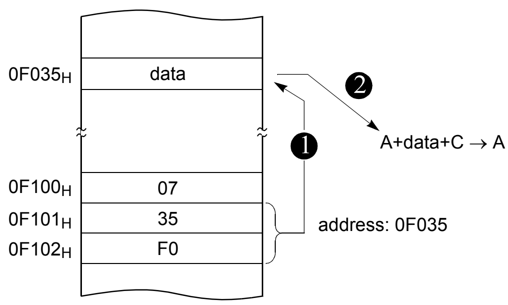

### (4) Absolute Addressing → !abs

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;

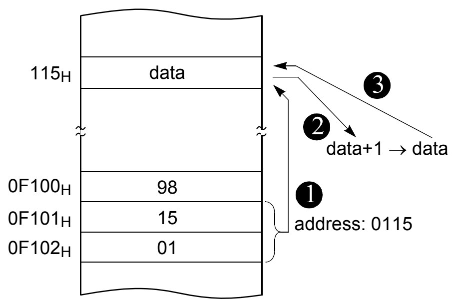
0735F0 ADC !$0F035_H$ ;A ←ROM[$0F035_H$]



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.

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




### (5) Indexed Addressing

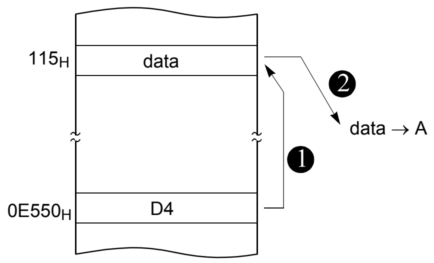
### X indexed direct page (no offset) → {X}

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

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

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

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



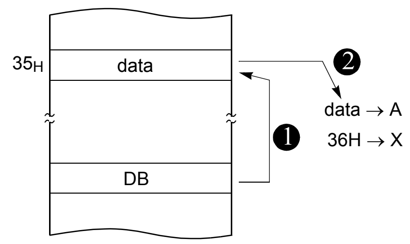

### X indexed direct page, auto increment→ {X}+

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}+
```



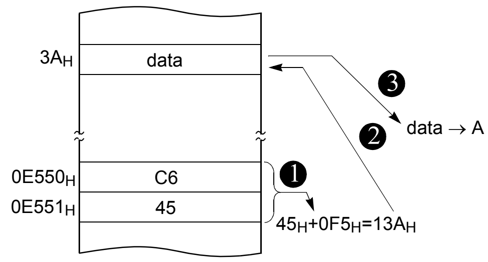

### X indexed direct page (8 bit offset) → dp+X

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, $X=0F5_H$

```
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.

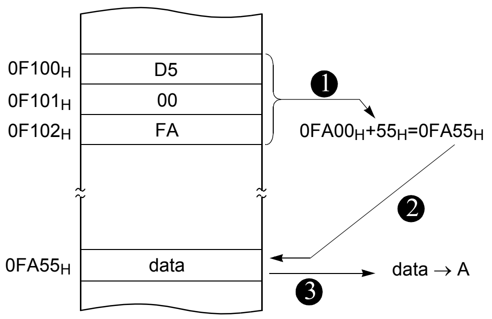
This is same with above. Use Y register instead of X.

### 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

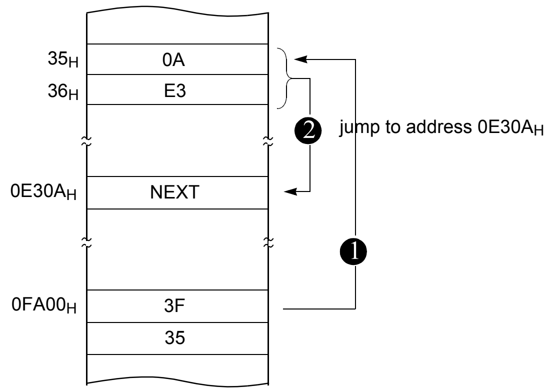
### Direct page indirect → [dp]

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]
```



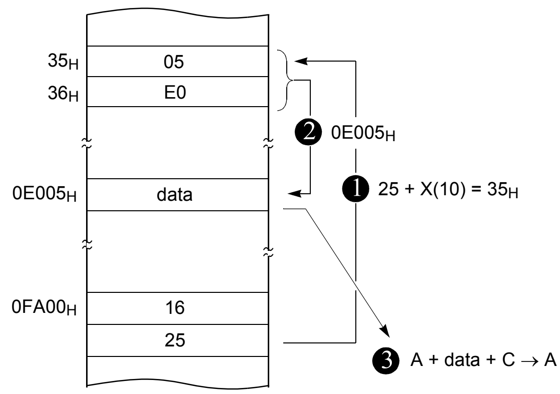

### X indexed indirect → [dp+X]

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]
```



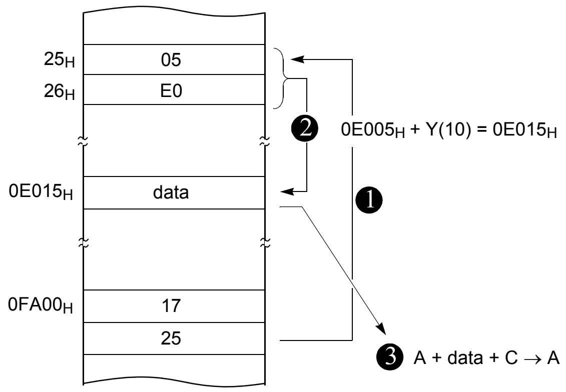
### Y indexed indirect → [dp]+Y

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
```



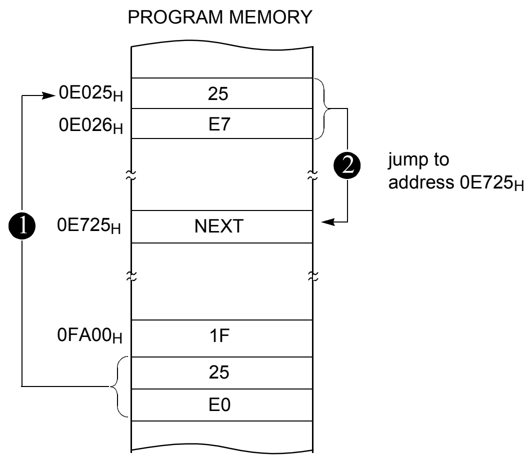

### Absolute indirect → [!abs]

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

JMP

Example; G=0

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



# 9. I/O PORTS

The MC80F7708A have seven I/O ports, LCD segment ports (R0, R1, R2, R4, R50/SEG0 ~ R73/SEG19) and LCD common ports (R77/COM0 ~ R74/COM3).

These ports pins may be multiplexed with an alternate function for the peripheral features on the device.

## 9.1 Registers for Ports

### Port Data Registers

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

### Port Direction Registers

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

All the port direction registers in the MC80F7708A have 0 written to them by reset function. Therefore, its initial status is input.

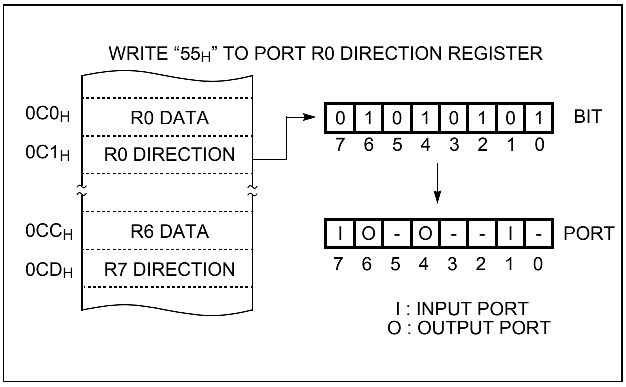



**Figure 9-1 Example of port I/O assignment**

### Pull-up Control Registers

The R0, R1, R2 and R4 ports have internal pull-up resistors. Figure 9-2 shows a functional diagram of a typical pull-up port. It is connected or disconnected by Pull-up Control register (R*n*PU). The value of that resistor is typically 100kΩ. Refer to DC characteristics for more details.

When a port is used as key input, input logic is firmly either low or high, therefore external pull-down or pull-up resisters are required practically. The MC80F7708A have internal pull-up, it can be logic high by pull-up that can be able to configure either connect or disconnect individually by pull-up control registers R*n*PU.

When ports are configured as inputs and pull-up resistor is selected by software, they are pulled to high.

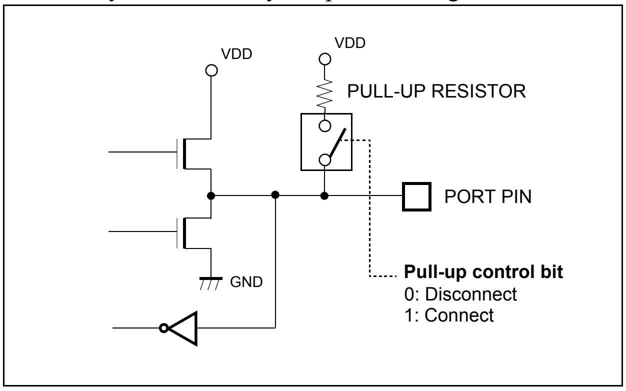



**Figure 9-2 Pull-up Port Structure**

### Open drain port Registers

The R0, R1, R2 and R4 ports have open drain port resistors R0OD~R4OD.
Figure 9-3 shows an open drain port configuration by control register. It is selected as either push-pull port or open-drain port by R0OD, R1OD, R2OD and R4OD.

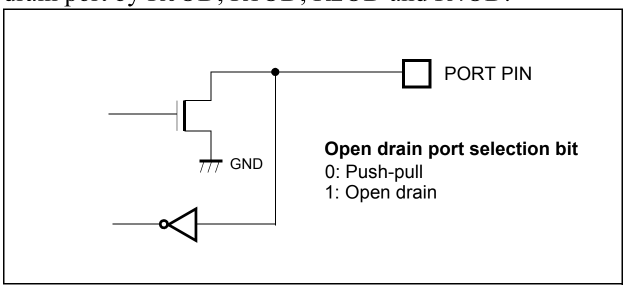



**Figure 9-3 Open-drain Port Structure**

## 9.2 I/O Ports Configuration

### R0 Port

R0 is a 4-bit CMOS bidirectional I/O port (address $0C0_H$). Each I/O pin can independently used as an input or an output through the R0IO register (address $0C1_H$).

R0 has internal pull-ups that is independently connected or disconnected by R0PU. The control registers for R0 are shown below.

In addition, Port R0 is multiplexed with various special features. The control register PSR0 (address $0AA_H$) controls the selection of alternate function. After reset, this value is "0", port may be used as normal I/O port.
To use alternate function such as External Interrupt rather than normal I/O, write "1" in the corresponding bit of PSR0.

**R0 Data Register** — ADDRESS : $0C0_H$, RESET VALUE : $00\text{-}0\text{-}\text{-}0\text{-}_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R0 | R07 | R06 | - | R04 | - | - | R01 | - |

**R0 Direction Register** — ADDRESS : $0C1_H$, RESET VALUE : $00\text{-}0\text{-}\text{-}0\text{-}_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R0IO | | | - | | - | - | | - |

Port Direction
0: Input
1: Output

**R0 Pull-up Selection Register** — ADDRESS : $0A5_H$, RESET VALUE : $00\text{-}0\text{-}\text{-}0\text{-}_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R0PU | | | - | | - | - | | - |

Pull-up select
0: Without pull-up
1: With pull-up

**R0 Open Drain Selection Register** — ADDRESS : $0A0_H$, RESET VALUE : $00\text{-}0\text{-}\text{-}0\text{-}_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R0OD | | | - | | - | - | | - |

Open Drain select
0: No Open Drain
1: Open Drain

**Port Selection Register 0** — ADDRESS : $0AA_H$, RESET VALUE : $00\text{-}0000\text{-}_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PSR0 | INT1I | INT0I | - | BUZO | 0 | 0 | EC0I | - |

INT1I (External Interrupt 1)
0: R07 Port
1: INT1

INT0I (External Interrupt 0)
0: R06 Port
1: INT0

BUZO (Buzzer Output)
0: R04 Port
1: BUZO

EC0I (Timer0 Event Input)
0: R01 Port
1: EC0

*** Unused bit of PSR0 should be set as**
**Bit3 : "0"**
**Bit2 : "0"**

| Port Pin | Alternate Function |
|---|---|
| R01 | EC0 (Timer0 Event Input) |
| R04 | BUZO (Buzzer Output) |
| R06 | INT0 (External Interrupt 0) |
| R07 | INT1 (External Interrupt 1) |

***Note:*** *R0IO, R0PU, P0OD and PSR0 are write-only registers. They can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

### R1 Ports

R1 is an 1-bit CMOS bidirectional I/O port (address $0C2_H$). Each I/O pin can independently used as an input or an output through the R1IO register (address $0C3_H$).

R1 has internal pull-up that is independently connected or disconnected by register R1PU. The control registers for R1 are shown below.

Port R1 is multiplexed with two special features. The control register controls the selection of alternate function. After reset, this value is "0", port may be used as normal I/O port. The way to select alternate function such as PWM1 or Timer Output Wave will be shown in each peripheral section.

***Note:*** *R1IO, R1PU, P1OD and PSR1 are write-only registers. They can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

**R1 Data Register** — ADDRESS : $0C2_H$ RESET VALUE : $-------0_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R1 | - | - | - | - | - | - | - | R10 |

**R1 Direction Register** — ADDRESS : $0C3_H$ RESET VALUE : $-------0_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R1IO | - | - | - | - | - | - | - | |

Port Direction
0: Input
1: Output

**R1 Pull-up Selection Register** — ADDRESS : $0A6_H$ RESET VALUE : $-------0_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R1PU | - | - | - | - | - | - | - | |

Pull-up select
0: Without pull-up
1: With pull-up

**R1 Open Drain Selection Register** — ADDRESS :$0A1_H$ RESET VALUE : $-------0_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R1OD | - | - | - | - | - | - | - | |

Open Drain select
0: No Open Drain
1: Open Drain

**Port Selection Register 1** — ADDRESS :$0AB_H$ RESET VALUE : $-----000_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PSR1 | - | - | - | - | - | INT3I | INT2I | PWM1O |

INT2I (External Interrupt 2)
0: R41 Port
1: INT2

PWM1O (PWM1 Output)
0: R10 Port
1: PWM1/T2O

INT3I (External Interrupt 3)
0: R44 Port
1: INT3

## R2 Ports

R2 is a 6-bit CMOS bidirectional I/O port (address $0C4_H$). Each I/O pin can independently used as an input or an output through the R2IO register (address $0C5_H$).

R2 has internal pull-ups that are independently connected or disconnected by R2PU (address $0A7_H$). The control registers for R2 are shown as below.

***Note:*** *R2IO, R2PU and P2OD are write-only registers. They can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

***Note:*** *The R24 and R25 are not supported in the MC80F7708AK.*

**R2 Data Register** — ADDRESS: $0C4_H$ RESET VALUE: $--000000_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R2 | - | - | R25 | R24 | R23 | R22 | R21 | R20 |

**R2 Direction Register** — ADDRESS : $0C5_H$ RESET VALUE : $--000000_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R2IO | - | - | | | | | | |

Port Direction
0: Input
1: Output

**R2 Pull-up Selection Register** — ADDRESS : $0A7_H$ RESET VALUE : $--000000_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R2PU | - | - | | | | | | |

Pull-up select
0: Without pull-up
1: With pull-up

**R2 Open Drain Selection Register** — ADDRESS : $0A2_H$ RESET VALUE : $--000000_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R2OD | - | - | | | | | | |

Open Drain select
0: No Open Drain
1: Open Drain

**A/D Converter Mode Register** — ADDRESS : $0E2_H$ RESET VALUE : $-0-00001_B$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ADCM | ADEN | 1 | 0 | ADS2 | ADS1 | ADS0 | ADST | ADF |

Analog input of A/D converter is selected by ADS0~ADS2

## R4 Port

R4 is a 7-bit CMOS bidirectional I/O port (address $0C8_H$). Each I/O pin can independently used as an input or an output through the R4IO register (address $0C9_H$).

R4 has internal pull-ups that is independently connected or disconnected by R4PU. The control registers for R4 are shown below.

**R4 Data Register** ADDRESS : $0C8_H$ RESET VALUE : $\text{-000000-}_B$

| R4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R47 | R46 | R45 | R44 | R43 | R42 | R41 | - |

**R4 Direction Register** ADDRESS : $0C9_H$ RESET VALUE : $\text{-000000-}_B$

| R4IO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | - | | | | | | | - |

Port Direction
0: Input
1: Output

**R4 Pull-up Selection Register** ADDRESS :$0A9_H$ RESET VALUE : $\text{0000000-}_B$

| R4PU | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | - |

Pull-up select
0: Without pull-up
1: With pull-up

**R4 Open Drain Selection Register** ADDRESS :$0A4_H$ RESET VALUE : $\text{-000000-}_B$

| R0OD | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | - | | | | | | | - |

Open Drain select
0: No Open Drain
1: Open Drain

**Port Selection Register 1** ADDRESS :$0AB_H$ RESET VALUE : $\text{-----000}_B$

| PSR1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | INT3I | INT2I | PWM1O |

INT2I (External Interrupt 2)
0: R41Port
1: INT2

INT3I (External Interrupt 3)
0: R41Port
1: INT3

PWM1O (PWM1 Output)
0: R10 Port
1: PWM1/T2O

**LCD Control Register** ADDRESS :$0B2_H$ RESET VALUE : $00000000_B$

| LCR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | SCKD | 1 | LCDEN | 0 | 1 | LCDDC | LCK1 | LCK0 |

SCKD (Sub Clock Disable)
0: Sub Clock Oscillation Eaable (SXIN/SXOUT)
1: Sub Clock Oscillation Disable (R45/R46)

In addition, Port R4 is multiplexed with oscillation input/output, reset and interrupt input pins. The control register PSR1 (address $0AB_H$) controls the selection of alternate function. After reset, this value is "0", port may be used as normal I/O port. To use alternate function such as External Interrupt rather than normal I/O, write "1" in the corresponding bit of PSR1.
Main oscillation input/output and reset pin can be used as normal I/O ports (R43/R42) and normal input port(R47) by selecting configuration options in flash writing. Sub oscillation input/output pin can be used as normal I/O ports by writing "1" to the SCKD bit of the LCR register

| Port Pin | Alternate Function |
|---|---|
| R41 | INT2 (External Interrupt 2) |
| R42 | $X_{OUT}$ (Oscillation Output) |
| R43 | $X_{IN}$ (Oscillation Input) |
| R44 | INT3 (External Interrupt 3) |
| R45 | $SX_{IN}$ (Subsystem Oscillation Input) |
| R46 | $SX_{OUT}$ (Subsystem Oscillation Output) |
| R47 | $\overline{RESET}$ (System Reset Input) |

***Note:** R4IO, R4PU, P4OD and PSR1 are write-only registers. They can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

## R5 Ports

R5 is an 8-bit CMOS bidirectional I/O port (address $0CA_H$). Each I/O pin can independently used as an input or an output through the R5IO register (address $0CB_H$).

R5 is multiplexed with LCD segment output(SEG0 ~ SEG7), which can be selected by writing appropriate value into the R5PSR(address $0AC_H$).

R50, R1, R52 is also multiplxed with RX0, TX0 and ACK, which can be selected by writing appropriate value into the ASIMR0(address $0B8_H$).

**R5 Data Register** ADDRESS: $0CA_H$ RESET VALUE: $00000000_B$

| R5 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R57 | R56 | R55 | R54 | R53 | R52 | R51 | R50 |

**R5 Direction Register** ADDRESS : $0CB_H$ RESET VALUE : $00000000_B$

| R5IO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Port Direction
0: Input
1: Output

**R5/LCD Port Selection Register** ADDRESS : $0AC_H$ RESET VALUE : $11111111_B$

| R5PSR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R5PS7 | R5PS6 | R5PS5 | R5PS4 | R5PS3 | R5PS2 | R5PS1 | R5PS0 |

0: Seg Selection(seg7~seg0)
1: Port Selection

***Note:** R5IO is write-only register. It can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

### R6 Ports

R6 is an 8-bit CMOS bidirectional I/O port (address $0CC_H$). Each I/O pin can independently used as an input or an output through the R6IO register (address $0CD_H$).

R6 is multiplexed with LCD segment output(SEG8 ~ SEG15), which can be selected by writing appropriate value into the R6PSR(address $0AD_H$).

**R6 Data Register**

ADDRESS: $0CC_H$
RESET VALUE: $00000000_B$

| R6 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R67 | R66 | R65 | R64 | R63 | R62 | R61 | R60 |

**R6 Direction Register**

ADDRESS : $0CD_H$
RESET VALUE : $00000000_B$

| R6IO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Port Direction
0: Input
1: Output

**R6/LCD Port Selection Register**

ADDRESS : $0AD_H$
RESET VALUE : $11111111_B$

| R6PSR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R6PS7 | R6PS6 | R6PS5 | R6PS4 | R6PS3 | R6PS2 | R6PS1 | R6PS0 |

0: Seg Selection(seg15~seg8)
1: Port Selection

***Note:*** *R6IO is write-only register. It can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

### R7 Ports

R7 is a 8-bit CMOS bidirectional I/O port (address $0B0_H$). Each I/O pin can independently used as an input or an output through the R7IO register (address $0B4_H$).

R7 is multiplexed with LCD common output (COM4 ~ COM0) and segment output(SEG16 ~ SEG19), which can be selected by writing appropriate value into the R7PSR(address $0AE_H$) and LCR(address $0B2_H$).

**R7 Data Register**

ADDRESS: $0B0_H$
RESET VALUE: $00000000_B$

| R7 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R77 | R76 | R75 | R74 | R73 | R72 | R71 | R70 |

**R7 Direction Register**

ADDRESS : $0B4_H$
RESET VALUE : $00000000_B$

| R7IO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Port Direction
0: Input
1: Output

**R7/LCD Port Selection Register**

ADDRESS : $0AE_H$
RESET VALUE : $11111111_B$

| R7PSR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R7PS7 | R7PS6 | R7PS5 | R7PS4 | R7PS3 | R7PS2 | R7PS1 | R7PS0 |

0: Seg or Com Selection(seg16~seg19, COM3 ~ COM0)
1: Port Selection

***Note:*** *R7IO is write-only register. It can not be read and can not be accessed by bit manipulation instruction. Do not use read or read-modify-write instruction. Use byte manipulation instruction.*

### SEG0~SEG19

Segment signal output pins for the LCD display. See "18. LCD DRIVER" on page 78 for details.

### COM0~COM3

Common signal output pins for the LCD display. See "18. LCD DRIVER" on page 78 for details.

SEG16~SEG19 and COM3~COM0 are selected by LCDD of the LCR register.

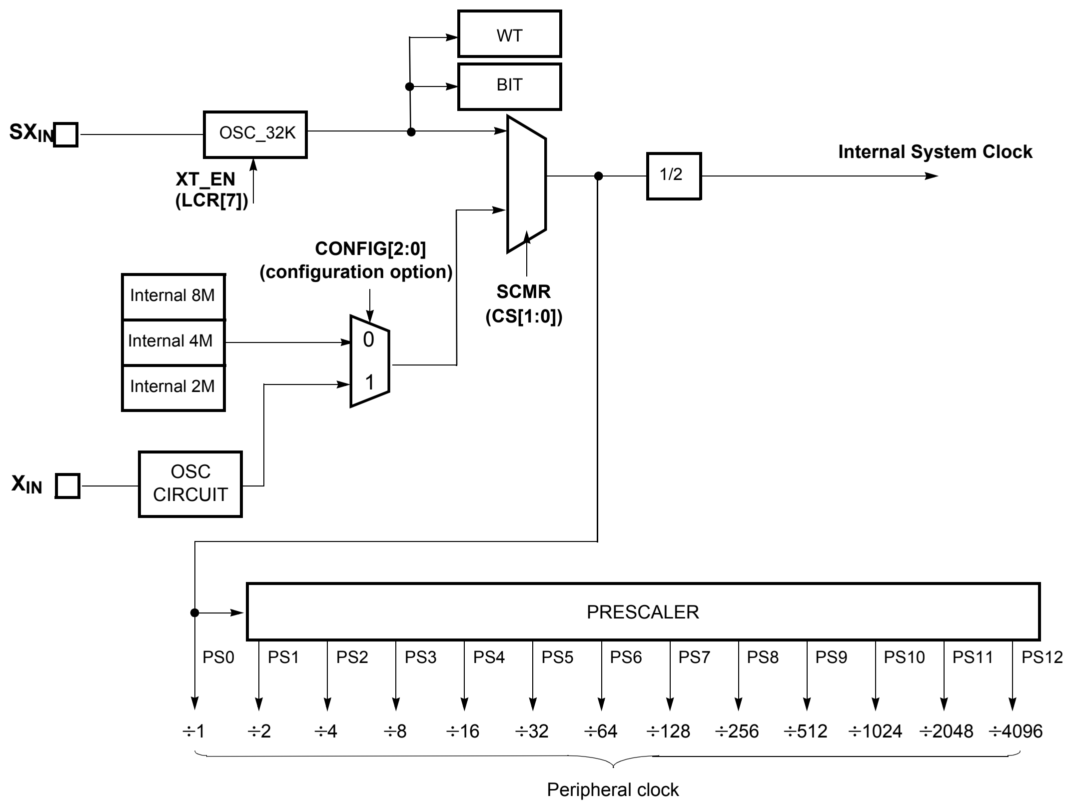
## 10. CLOCK GENERATOR

As shown in Figure 10-1, the clock generator produces the basic clock pulses which provide the system clock to be supplied to the CPU and the peripheral hardware. It contains two oscillators which are main-frequency clock oscillator and a sub-frequency clock oscillator.The system clock can also be obtained from the external oscillator. By setting configuration option, the internal 8MHz, 4MHz, 2MHz can also be selected for system clock source.

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

The internal system clock should be selected to main oscillation by setting bit1 and bit0 of the system clock mode register (SCMR). The registers are shown in Figure 10-2.

To the peripheral block, the clock among the not-divided original clocks, divided by 2, 4,..., up to 4096 can be provided. Peripheral clock is enabled or disabled by STOP instruction. The peripheral clock is controlled by clock control register (CKCTLR). See "11. BASIC INTERVAL TIMER" on page 43 for details.




| $f_{MAIN}$(Hz) | | PS0 | PS1 | PS2 | PS3 | PS4 | PS5 | PS6 | PS7 | PS8 | PS9 | PS10 | PS11 | PS12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4M | Frequency | 4M | 2M | 1M | 500K | 250K | 125K | 62.5K | 31.25K | 15.63K | 7.183K | 3.906K | 1.953K | 976 |
| | period | 250n | 500n | 1u | 2u | 4u | 8u | 16u | 32u | 64u | 128u | 256u | 512u | 1.024m |

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**Figure 10-1 Block Diagram of Clock Generator**

The SCMR should be set to operate by main oscillation. Bit2, bit1 and bit0 of the SCMR should be set to “000” or “001” to select main oscillation.

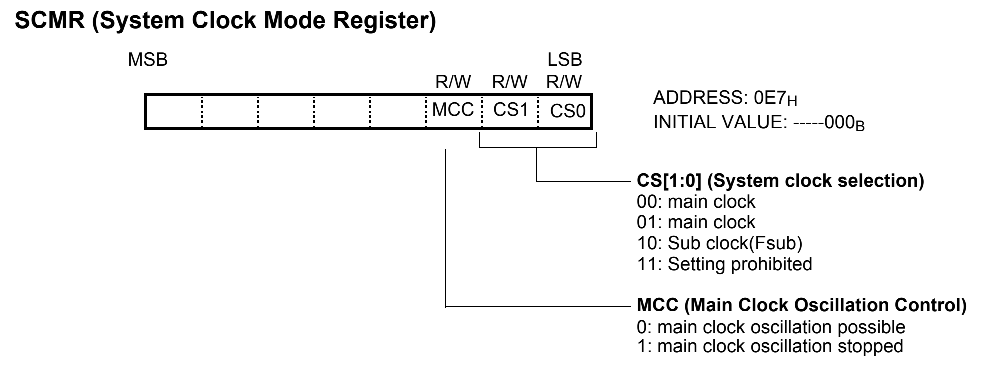



**Figure 10-2 SCMR : System Clock Mode Register**

# 11. BASIC INTERVAL TIMER

The MC80F7708A have 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 Register (BITR) is increased every internal count pulse which is divided by prescaler. Since prescaler has division ratio from 8 to 1024, the count rate is 1/8 to 1/1024 of the oscillator frequency. After reset, the BCK bits are all set, so the longest oscillation stabilization time is obtained.

It also provides a Basic interval timer interrupt (BITF). The count overflow of BITR from $FF_H$ to $00_H$ causes the interrupt to be generated. The Basic Interval Timer is controlled by the clock control register (CKCTLR) shown in Figure 11-2.

Source clock can be selected by lower 3 bits of CKCTLR. When write “1” to bit BCL of CKCTLR, BITR register is cleared to “0” and restart to count up. The bit BCL becomes “0” automatically after one machine cycle by hardware.

BITR and CKCTLR are located at same address, and address $0E6_H$ is read as a BITR, and written to CKCTLR.

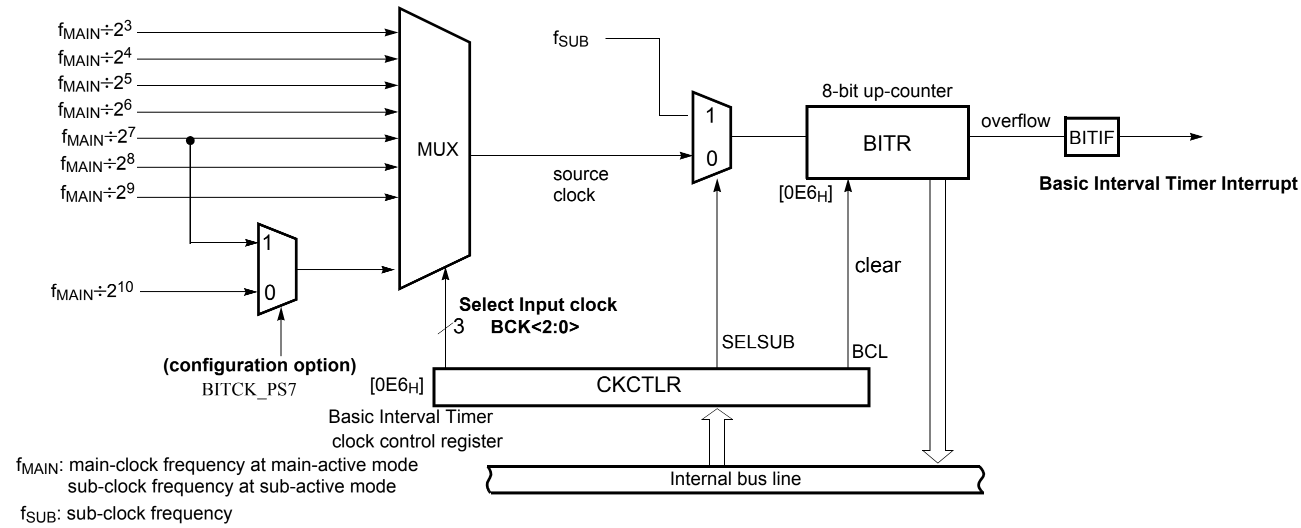



**Figure 11-1 Block Diagram of Basic Interval Timer**

| BCK<2:0> | Source clock | Interrupt (overflow) Period |
|---|---|---|
| | SCMR[1:0] = 00 or 01 | At $f_{MAIN}$ = 4MHz |
| 000 | $f_{MAIN} \div 2^3$ | 0.512 ms |
| 001 | $f_{MAIN} \div 2^4$ | 1.024 |
| 010 | $f_{MAIN} \div 2^5$ | 2.048 |
| 011 | $f_{MAIN} \div 2^6$ | 4.096 |
| 100 | $f_{MAIN} \div 2^7$ | 8.192 |
| 101 | $f_{MAIN} \div 2^8$ | 16.384 |
| 110 | $f_{MAIN} \div 2^9$ | 32.768 |
| 111 | $f_{MAIN} \div 2^{10}$ | 65.536 |

**Table 11-1 Basic Interval Timer Interrupt Time**

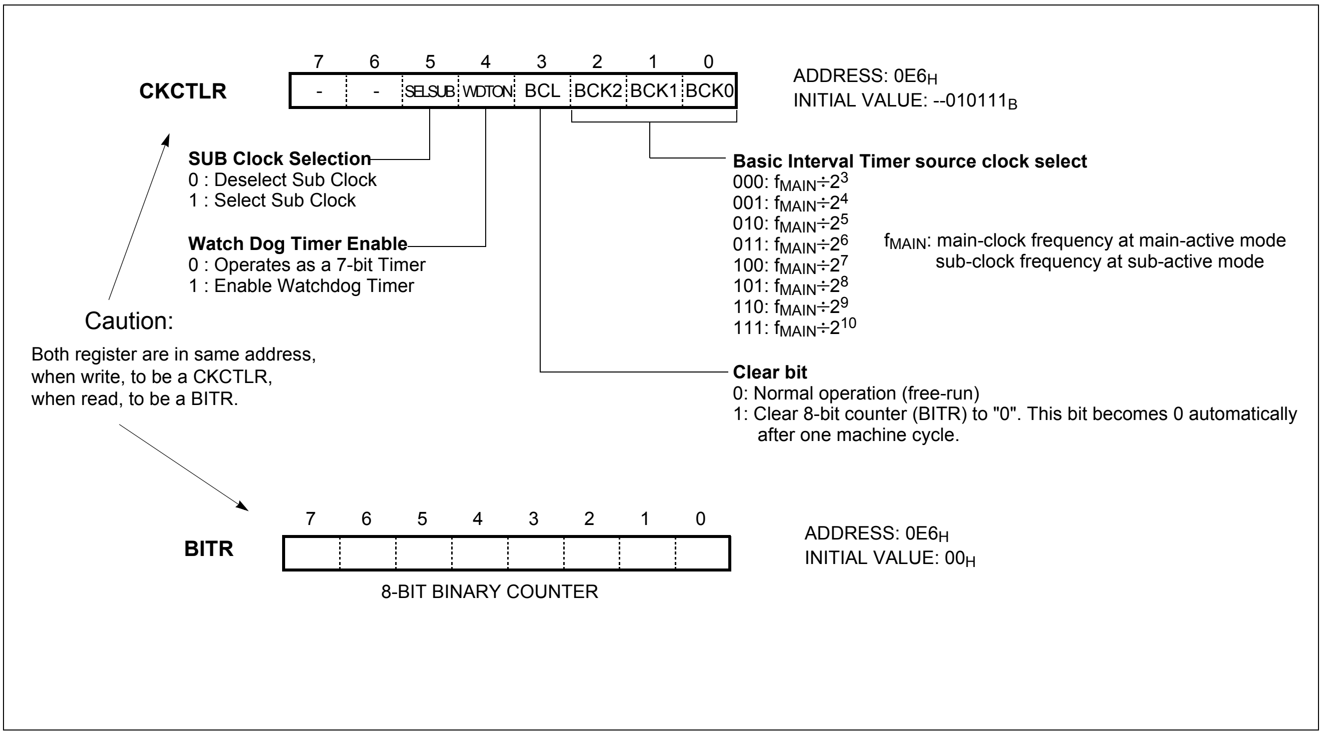



**Figure 11-2 BITR: Basic Interval Timer Mode Register**

**Example 1**:

Interrupt request flag is generated every 8.192ms at 4MHz.

```
:
LDM    CKCTLR,#0CH
SET1   BITE
EI
:
```

## 12. TIMER / COUNTER

Timer/Event Counter consists of prescaler, multiplexer, 8-bit timer data register, 8-bit counter register, mode register, input capture register and Comparator as shown in Figure 12-4. And the PWM high register for PWM is consisted separately.

The timer/counter has seven operating modes.
- 8 Bit Timer/Counter Mode
- 8 Bit Capture Mode
- 8 Bit Compare Output Mode
- 16 Bit Timer/Counter Mode
- 16 Bit Capture Mode
- 16 Bit Compare Output Mode
- PWM Mode

In the "timer" function, the register is increased every internal clock input. Thus, one can think of it as counting internal clock input. Since a least clock consists of 2 and most clock consists of 2048 oscillator periods, the count rate is 1/2 to 1/2048 of the oscillator frequency in Timer0. And Timer1 can use the same clock source too. In addition, Timer1 has more fast clock source (1/1 to 1/8).

In the "counter" function, the register is increased in response to a 0-to-1 (rising edge) transition at its corresponding external input pin EC0 (Timer 0).

In addition the "capture" function, the register is increased in response external or internal clock interrupt same with timer/counter function. When external interrupt edge input, the count register is captured into capture data register TMx.

Timer3 is shared with "PWM" function and Timer2 is shared with "Compare output" function.

**Example 1:**

Timer 0 = 8-bit timer mode, 8ms interval at 4MHz
Timer 1 = 8-bit timer mode, 4ms interval at 4MHz

```
LDM   SCMR,#0        ;Main clock mode
LDM   TDR0,#249
LDM   TM0,#0001_0011B
LDM   TDR1,#124
LDM   TM1,#0000_1111B

SET1  T0E
SET1  T1E
EI
 :
 :
 :
```

**Example 2:**

Timer0 = 16-bit timer mode, 0.5s at 4MHz

```
LDM   SCMR,#0        ;Main clock mode
LDM   TDR0,#23H
LDM   TDR1,#0F4H
LDM   TM0,#0FH       ;FMAIN/32, 8us
LDM   TM1,#4CH

SET1  T0E
EI
 :
 :
 :
```

**Example 3:**

Timer0 = 8-bit event counter

Timer2 = 8-bit capture mode, 2us sampling count.

```
LDM   TDR0,#0FFH       ;don't care
LDM   TM0,#1FH          ;event counter
LDM   R0IO,#1XXX_XX1XB ;R07, R01 input

LDM   IEDS,#XXXX_01XXB  ;FALLING
LDM   PSR0,#1XXX_XX1XB;INT1,EC0
LDM   TDR2,#0FFH
LDM   TM2,#0010_1011B   ;2us

SET1  T0E               ;ENABLE TIMER 0
SET1  T2E               ;ENABLE TIMER 1
SET1  INT1E             ;ENABLE INT1
EI
 :
```

X: don't care.

**Example 4:**

Timer0 = 16-bit capture mode, 8us sampling count. at 4MHz

```
LDM   TDR0,#0FFH
LDM   TDR1,#0FFH
LDM   TM0,#2FH
LDM   TM1,#5FH

LDM   IEDS,#XXXX_XX01B
LDM   PSR0,#X1XX_XXXXB ;AS INT0

SET1  T0E    ;ENABLE TIMER 0
SET1  INT0E  ;ENABLE EXT. INT0
EI
 :
```

X: don't care.

## TM0, TM2 (Timer0, 2 Mode Control Register)

**TM0** — ADDRESS: $0D0_H$, INITIAL VALUE: $--000000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | R/W | R/W | R/W | R/W | R/W | R/W |
| TM0 | - | - | CAP0 | T0CK2 | T0CK1 | T0CK0 | T0CN | T0ST |

**TM2** — ADDRESS: $0D6_H$, INITIAL VALUE: $--000000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | R/W | R/W | R/W | R/W | R/W | R/W |
| TM2 | - | - | CAP2 | T2CK2 | T2CK1 | T2CK0 | T2CN | T2ST |

**CAP0,CAP2 (Capture Mode Selection Bit)**
0: Timer/Counter Mode
1: Capture Mode

**T0CN,T2CN (Timer 0,2 Continue Start)**
0: Pause Counting
1: Continue Counting

**T0ST,T2ST (Timer 0,2 Start Control)**
0: Stop Counting
1: Clear the counter and Start counting again

**T0CK[2:0],T2CK[2:0] (Timer 0,2 Input Clock Selection)**

| T0CK[2:0] | T2CK[2:0] |
|---|---|
| 000: $f_{MAIN} \div 2$ | 000: $f_{MAIN} \div 2$ |
| 001: $f_{MAIN} \div 2^2$ | 001: $f_{MAIN} \div 2^2$ |
| 010: $f_{MAIN} \div 2^3$ | 010: $f_{MAIN} \div 2^4$ |
| 011: $f_{MAIN} \div 2^5$ | 011: $f_{MAIN} \div 2^6$ |
| 100: $f_{MAIN} \div 2^7$ | 100: $f_{MAIN} \div 2^8$ |
| 101: $f_{MAIN} \div 2^9$ | 101: $f_{MAIN} \div 2^{10}$ |
| 110: $f_{MAIN} \div 2^{11}$ | 110: $f_{MAIN} \div 2^{12}$ |
| 111: External Event clock EC0 | 111: Reserved |

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

## TM1, TM3 (Timer1, 3 Mode Control Register)

**TM1** — ADDRESS: $0D2_H$, INITIAL VALUE: $-0--0000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| TM1 | - | 16BIT | - | - | T1CK1 | T1CK0 | T1CN | T1ST |

**TM3** — ADDRESS: $0D8_H$, INITIAL VALUE: $000-0000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| TM3 | POL | 16BIT | PWM1E | - | T3CK1 | T3CK0 | T3CN | T3ST |

**POL (PWM Output Polarity Selection)**
0: Duty Active Low
1: Duty Active High

**16BIT (16 Bit Mode Selection)**
0: 8-Bit Mode
1: 16-Bit Mode

**PWM1E (PWM Enable Bit)**
0: PWM1 Disable (T2O Enable)
1: PWM1 Enable (T2O Disable)

**T1CK[1:0],T3CK[1:0] (Timer 1,3 Input Clock Selection)**

| T1CK[1:0] | T3CK[1:0] |
|---|---|
| 00: $f_{MAIN}$ | 00: $f_{MAIN}$ |
| 01: $f_{MAIN} \div 2$ | 01: $f_{MAIN} \div 2$ |
| 10: $f_{MAIN} \div 2^3$ | 10: $f_{MAIN} \div 2^4$ |
| 11: Timer0 Clock | 11: Timer2 Clock |

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**T1CN,T3CN (Timer 1,3 Continue Start)**
0: Stop Counting
1: Start Counting

**T1ST,T3ST (Timer 1,3 Start Control)**
0: Stop counting
1: Clear the counter and start count again

*****The counter will be cleared and restarted only when the TxST bit cleared and set again. If TxST bit set again when TxST bit is set, the counter can't be cleared but only start again.***

| T0CK2 | T0CK1 | T0CK0 | | 4MHz | 8MHz | 10MHz |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | ($f_{MAIN} \div 2$) | 500nS | 250nS | 200nS |
| 0 | 0 | 1 | ($f_{MAIN} \div 2^2$) | 1uS | 500nS | 400nS |
| 0 | 1 | 0 | ($f_{MAIN} \div 2^3$) | 2uS | 1uS | 800nS |
| 0 | 1 | 1 | ($f_{MAIN} \div 2^5$) | 8uS | 4uS | 3.2uS |
| 1 | 0 | 0 | ($f_{MAIN} \div 2^7$) | 32uS | 16uS | 12.8uS |
| 1 | 0 | 1 | ($f_{MAIN} \div 2^9$) | 128uS | 64uS | 51.2uS |
| 1 | 1 | 0 | ($f_{MAIN} \div 2^{11}$) | 512uS | 256uS | 204.8uS |

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**Figure 12-1 Timer0,1,2,3 Registers**

**T0 (Timer0 Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | T07 | T06 | T05 | T04 | T03 | T02 | T01 | T00 |

ADDRESS: $0D1_H$
INITIAL VALUE:00H

**CDR0 (Timer0 Input Capture Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | CDR07 | CDR06 | CDR05 | CDR04 | CDR03 | CDR02 | CDR01 | CDR00 |

ADDRESS: $0D1_H$
INITIAL VALUE:00H

*In Timer mode, this register is the value of Timer 0 counter and in Capture mode, this register is the value of input capture.*

**TDR0 (Timer 0 Data Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | TDR07 | TDR06 | TDR05 | TDR04 | TDR03 | TDR02 | TDR01 | TDR00 |

ADDRESS: $0D1_H$
INITIAL VALUE:$FF_H$

*If the counter of Timer 0 and the data of TDR0 is equal, interrupt is occurred.*

**T1 (Timer1 Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | T17 | T16 | T15 | T14 | T13 | T12 | T11 | T10 |

ADDRESS: $0D4_H$
INITIAL VALUE:00H

**CDR1 (Timer1 Input Capture Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | CDR17 | CDR16 | CDR15 | CDR14 | CDR13 | CDR12 | CDR11 | CDR10 |

ADDRESS: $0D4_H$
INITIAL VALUE:$00_H$

*In Timer mode, this register is the value of Timer 1 counter and in Capture mode, this register is the value of input capture.*

**TDR1 (Timer1 Data Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | TDR17 | TDR16 | TDR15 | TDR14 | TDR13 | TDR12 | TDR11 | TDR10 |

ADDRESS: $0D3_H$
INITIAL VALUE:$FF_H$

*If the counter of Timer 1 and the data of TDR1 is equal, interrupt is occurred.*

**Figure 12-2 Related Registers with Timer/Counter0, 1**

**T2 (Timer2 Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | T27 | T26 | T25 | T24 | T23 | T22 | T21 | T20 |

ADDRESS: $0D7_H$
INITIAL VALUE:00H

**CDR2 (Timer2 Input Capture Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | CDR27 | CDR26 | CDR25 | CDR24 | CDR23 | CDR22 | CDR21 | CDR20 |

ADDRESS: $0D7_H$
INITIAL VALUE:00H

*In Timer mode, this register is the value of Timer 2 counter and in Capture mode, this register is the value of input capture.*

**TDR2 (Timer2 Data Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | TDR27 | TDR26 | TDR25 | TDR24 | TDR23 | TDR22 | TDR21 | TDR20 |

ADDRESS: $0D7_H$
INITIAL VALUE:$FF_H$

*If the counter of Timer 0 and the data of TDR0 is equal, interrupt is occurred.*

**TDR3 (Timer3 Data Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | TDR37 | TDR36 | TDR35 | TDR34 | TDR33 | TDR32 | TDR31 | TDR30 |

ADDRESS: $0D9_H$
INITIAL VALUE:$FF_H$

*If the counter of Timer 1 and the data of TDR1 is equal, interrupt is occurred.*

**T3PPR (Timer3 PWM Period Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | PWM1PR7 | PWM1PR6 | PWM1PR5 | PWM1PR4 | PWM1PR3 | PWM1PR2 | PWM1PR1 | PWM1PR0 |

ADDRESS: $0D9_H$
INITIAL VALUE:$FF_H$

*The period is decided by PWM.*

**T3 (Timer3 Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | T37 | T36 | T35 | T34 | T33 | T32 | T31 | T30 |

ADDRESS: $0DA_H$
INITIAL VALUE:00H

**T3PDR (Timer3 PWM Duty Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W/R | W/R | W/R | W/R | W/R | W/R | W/R | W/R |
| | PWM1DR7 | PWM1DR6 | PWM1DR5 | PWM1DR4 | PWM1DR3 | PWM1DR2 | PWM1DR1 | PWM1DR0 |

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

*In PWM mode, decide the pulse duty.*

**CDR3 (Timer3 Input Capture Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | CDR37 | CDR36 | CDR35 | CDR34 | CDR33 | CDR32 | CDR31 | CDR30 |

ADDRESS: $0DA_H$
INITIAL VALUE:00H

*In Timer mode, this register is the value of Timer 2 counter and in Capture mode, this register is the value of input capture.*

**T3PWHR (Timer3 High Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | W | W | W | W | W |
| | - | - | - | - | PWM1HR3 | PWM1HR2 | PWM1HR1 | PWM1HR0 |

ADDRESS: $0DB_H$
INITIAL VALUE:----$00000_B$

**Figure 12-3 Related Registers with Timer/Counter2, 3**

| 16BIT | CAP0 | - | T0CK[2:0] | T1CK[1:0] | Timer 0 | Timer 1 |
|---|---|---|---|---|---|---|
| 0 | 0 | - | XXX | XX | 8 Bit Timer | 8 Bit Timer |
| 0 | 0 | - | 111 | XX | 8 Bit Event Counter | 8 Bit Timer |
| 0 | 1 | - | XXX | XX | 8 Bit Capture | 8 Bit Compare Output |
| 1 | 0 | - | XXX | 11 | 16 Bit Timer | |
| 1 | 0 | - | 111 | 11 | 16 Bit Event Counter | |
| 1 | 1 | - | XXX | 11 | 16 Bit Capture | |
| 1 | 0 | - | XXX | 11 | 16 Bit Compare Output | |

**Table 12-1 Operating Modes of Timer 0 and Timer 1**

## 12.1 8-Bit Timer/Counter Mode

The MC80F7708A have four 8-bit Timer/Counters, Timer0, Timer1, Timer2 and Timer3 as shown in Figure 12-4.

The "timer" or "counter" function is selected by mode registers TMx (x=0,1,2,3) as shown in Figure 12-1 and Table 12-1. To use as an 8-bit timer/counter mode, bit CAPx of TMx is cleared to "0" and bits 16BIT of TM1(3) should be cleared to "0" (Table 12-1 ).

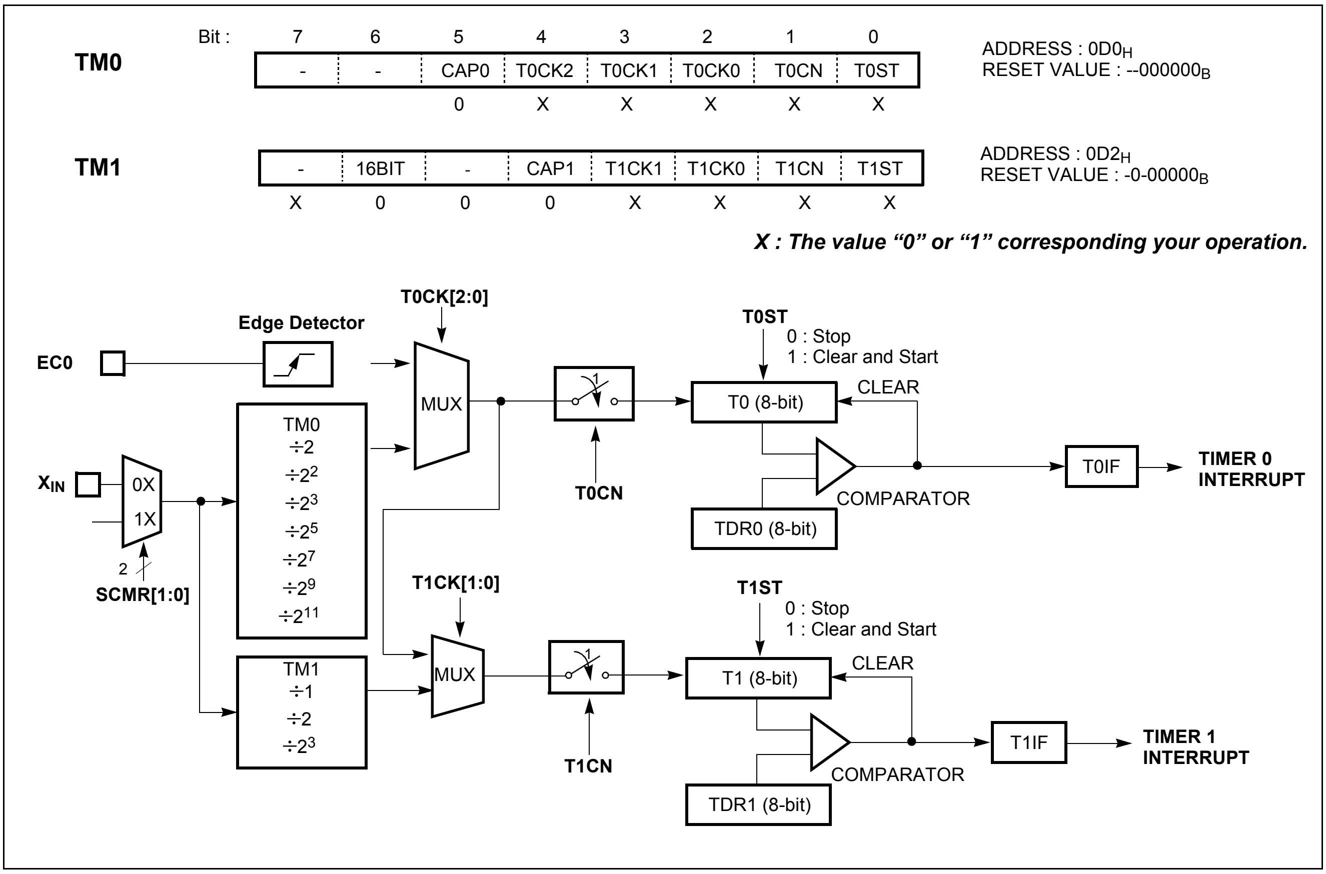



**Figure 12-4 Block Diagram of Timer/Event Counter0,1**

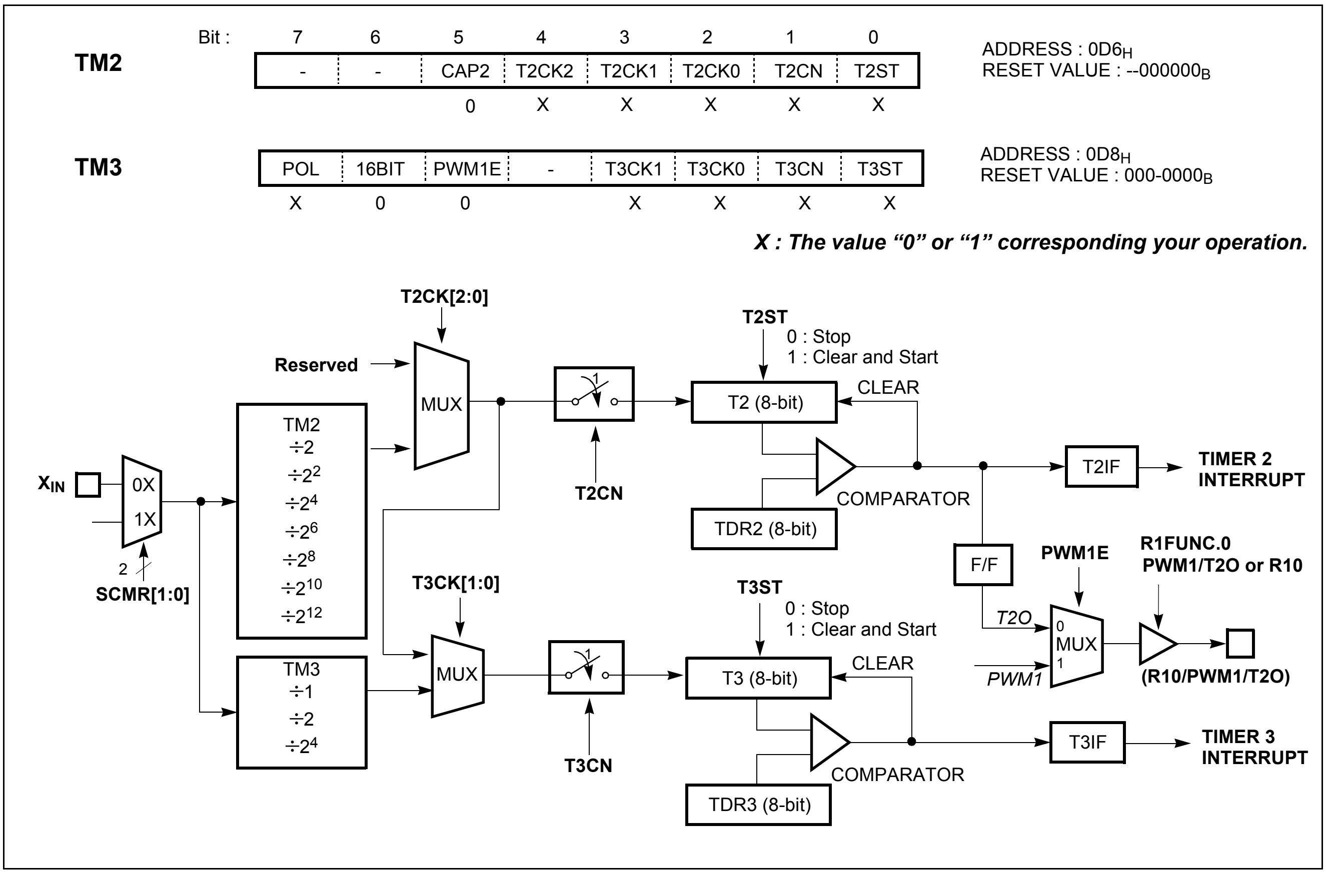



**Figure 12-5 Block Diagram of Timer 2,3**

These timers have each 8-bit count register and data register. The count register is increased by every internal or external clock input. The internal clock has a prescaler divide ratio option of 2, 4, 8, 32,128, 512, 2048 (selected by control bits TxCK2, TxCK1 and TxCK0 of register TM0(2)) and 1, 2, 8 (selected by control bits TxCK1 and TxCK0 of register TM1(3)).

In the Timer, timer register Tx increases from $00_H$ until it matches TxDR and then reset to $00_H$. If the value of Tx is equal with TxDR, Timer x interrupt is occurred (latched in TxIF bit). TxDR and T0 register are in same address, so this register is read from T0 and written to TDR0.

In counter function, the counter is increased every 0-to 1 (rising edge) transition of EC0 pin. In order to use counter function, the bit R01 of the R0 Direction Register (R0IO) should be set to "0" and the bit EC0E of Port Selection Register PSR0 should set to "1". The Timer 0 can be used as a counter by pin EC0 input, but other timers can not used as a event counter.

***Note:*** *The contents of TDR0, TDR1, TDR2 and TDR3 must be initialized (by software) with the value between $1_H$ and $0FF_H$, not $0_H$.*

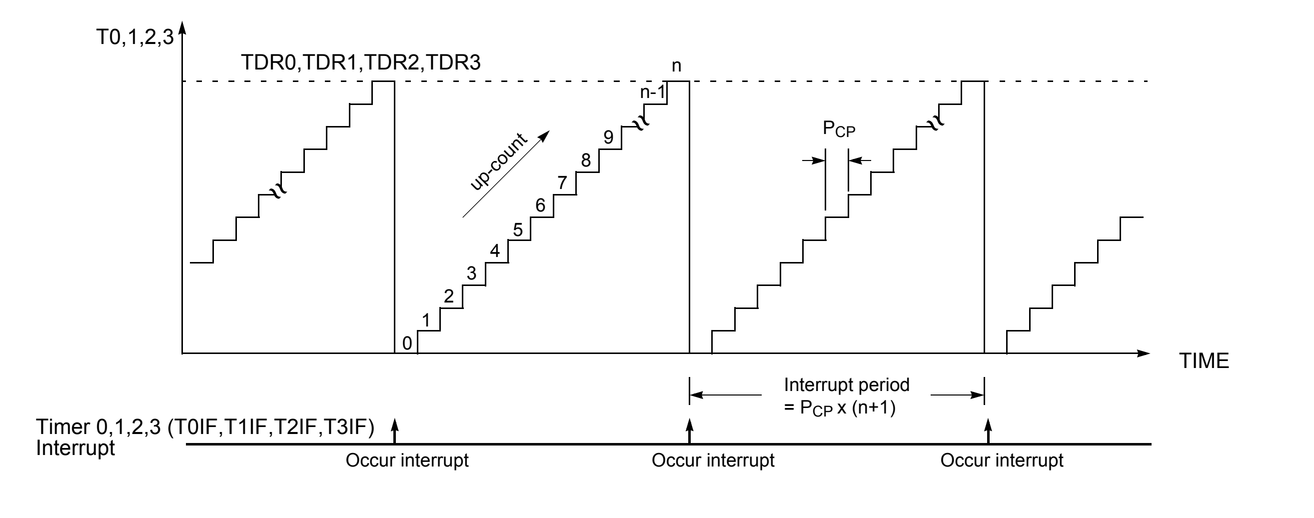



**Figure 12-6 Counting Example of Timer Data Registers**

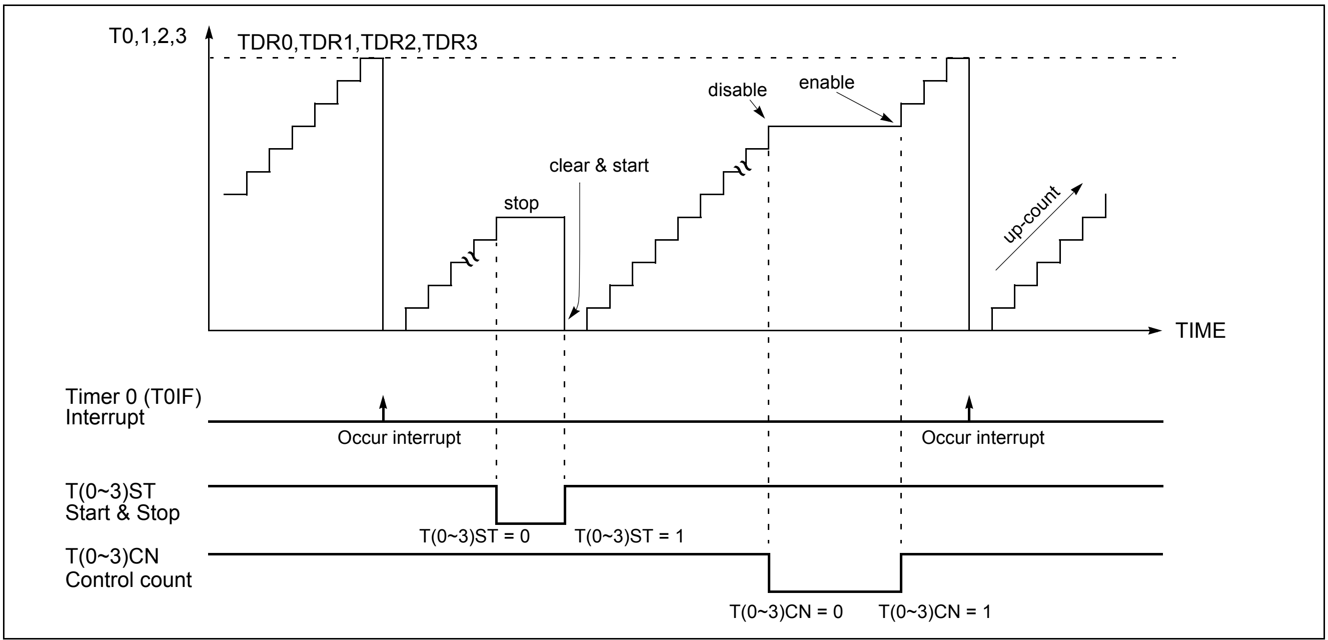



**Figure 12-7 Timer Count Operation**

## 12.2 16 Bit Timer/Counter Mode

The Timer register is running with 16 bits. A 16-bit timer/ counter register T0, T1 are increased from $0000_H$ until it matches TDR0, TDR1 and then resets to $0000_H$. The match output generates Timer 0 interrupt not Timer 1 interrupt.

The clock source of the Timer 0 is selected either internal or external clock by bit T0CK2, T0CK1 and T0CK0.

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

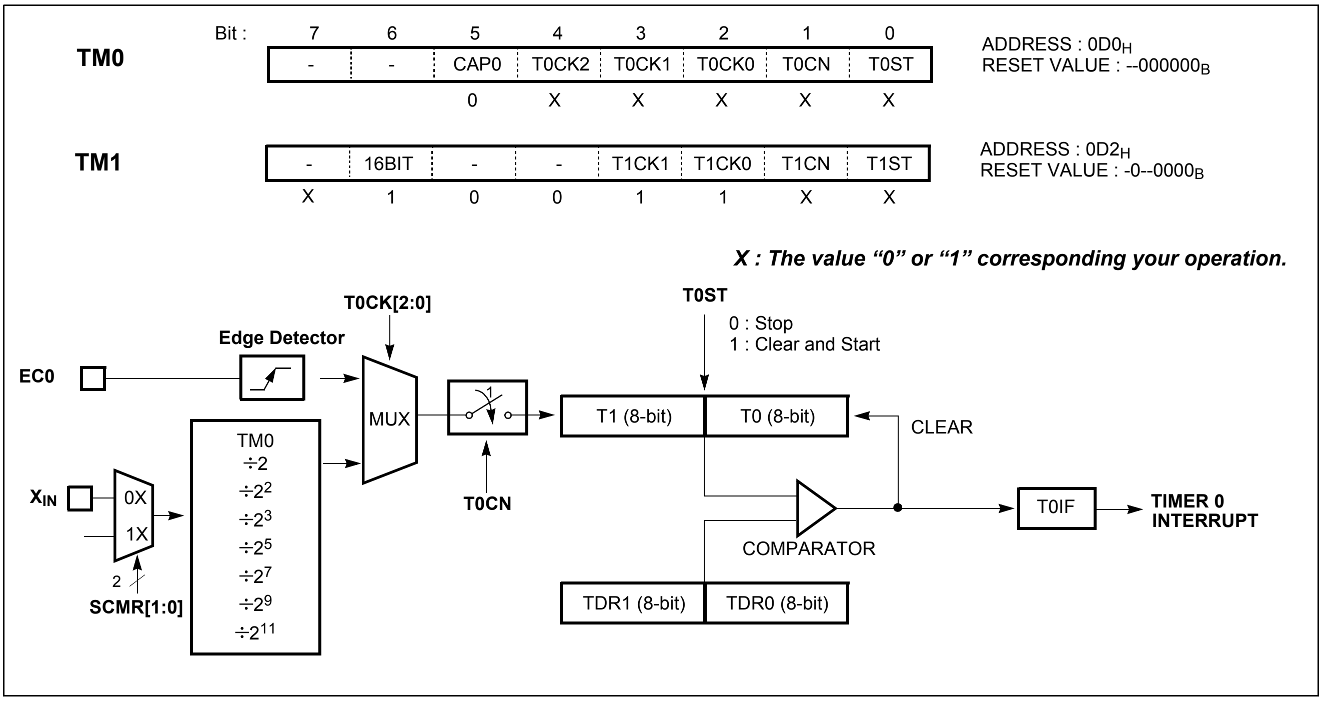



**Figure 12-8 16-bit Timer / Counter Mode 0**

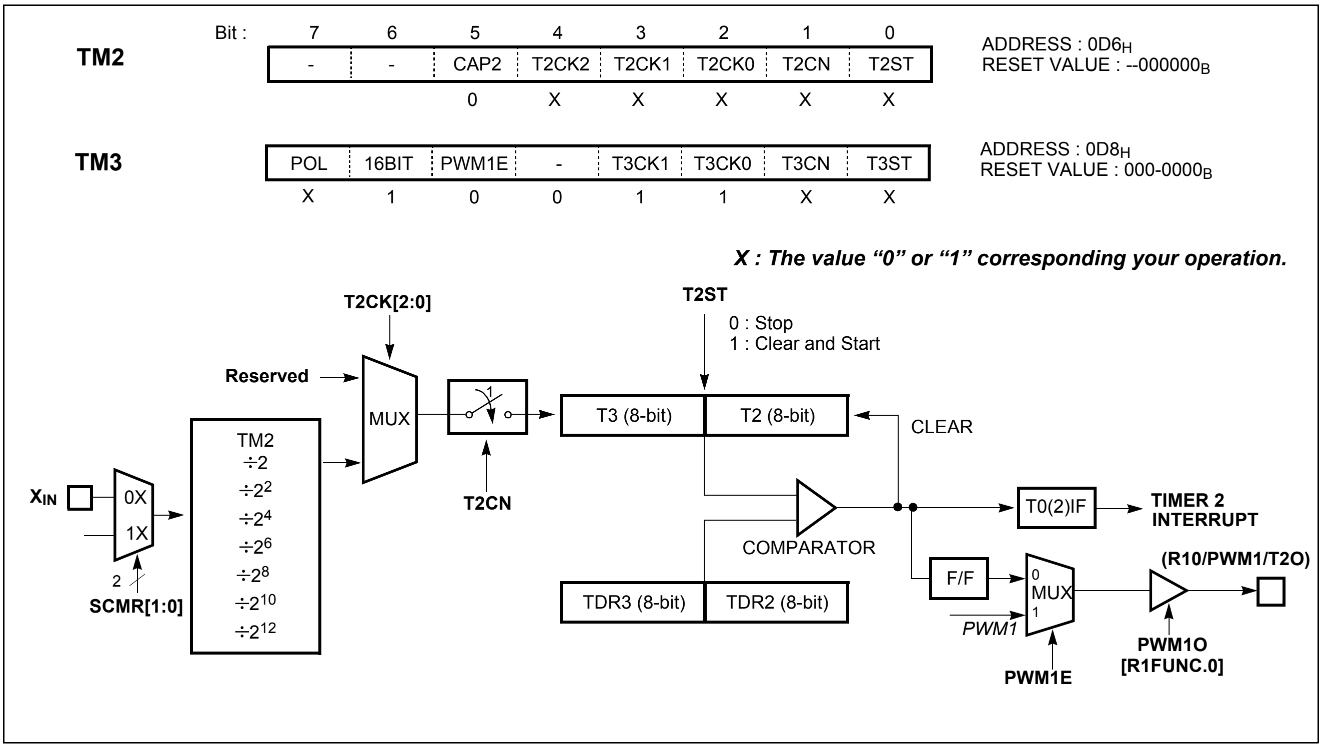



**Figure 12-9 16-bit Timer / Counter Mode 2**

### 12.3 8-Bit Capture Mode

The Timer 0 capture mode is set by bit CAP0 of timer mode register TM0 (bit CAPx of timer mode register TMx for Timer 1,2,3) as shown in Figure 12-10.

As mentioned above, not only Timer 0 but Timer 1,2,3 can also be used as a capture mode.

The Timer/Counter register is increased in response internal or external input. This counting function is same with normal timer mode, and Timer interrupt is generated when timer register T0 (T1,2,3) increases and matches TDR0 (TDR1,TDR2,TDR3).

This timer interrupt in capture mode is very useful when the pulse width of captured signal is more wider than the maximum period of Timer.

For example, in Figure 12-13, the pulse width of captured signal is wider than the timer data value ($FF_H$) over 2 times. When external interrupt is occurred, the captured value ($13_H$) is more little than wanted value. It can be obtained correct value by counting the number of timer overflow occurrence.

Timer/Counter still does the above, but with the added feature that a edge transition at external input INTx pin causes the current value in the Timer x register (T0,T1,T2,T3), to be captured into registers CDRx (x=0,1,2,3), respectively. After captured, Timer x register is cleared and restarts by hardware.

It has three transition modes: “falling edge”, “rising edge”, “both edge” which are selected by interrupt edge selection register IEDS (Refer to External interrupt section). In addition, the transition at INTx pin generate an interrupt.

**Note:** *The CDR0, TDR0 and T0 are in same address. In the capture mode, reading operation is to read the CDR0 and in timer mode, reading operation is read the T0. TDR0 is only for writing operation.*
*The CDR1, T1 are in same address, the TDR1 is located in different address. In the capture mode, reading operation is to read the CDR1*

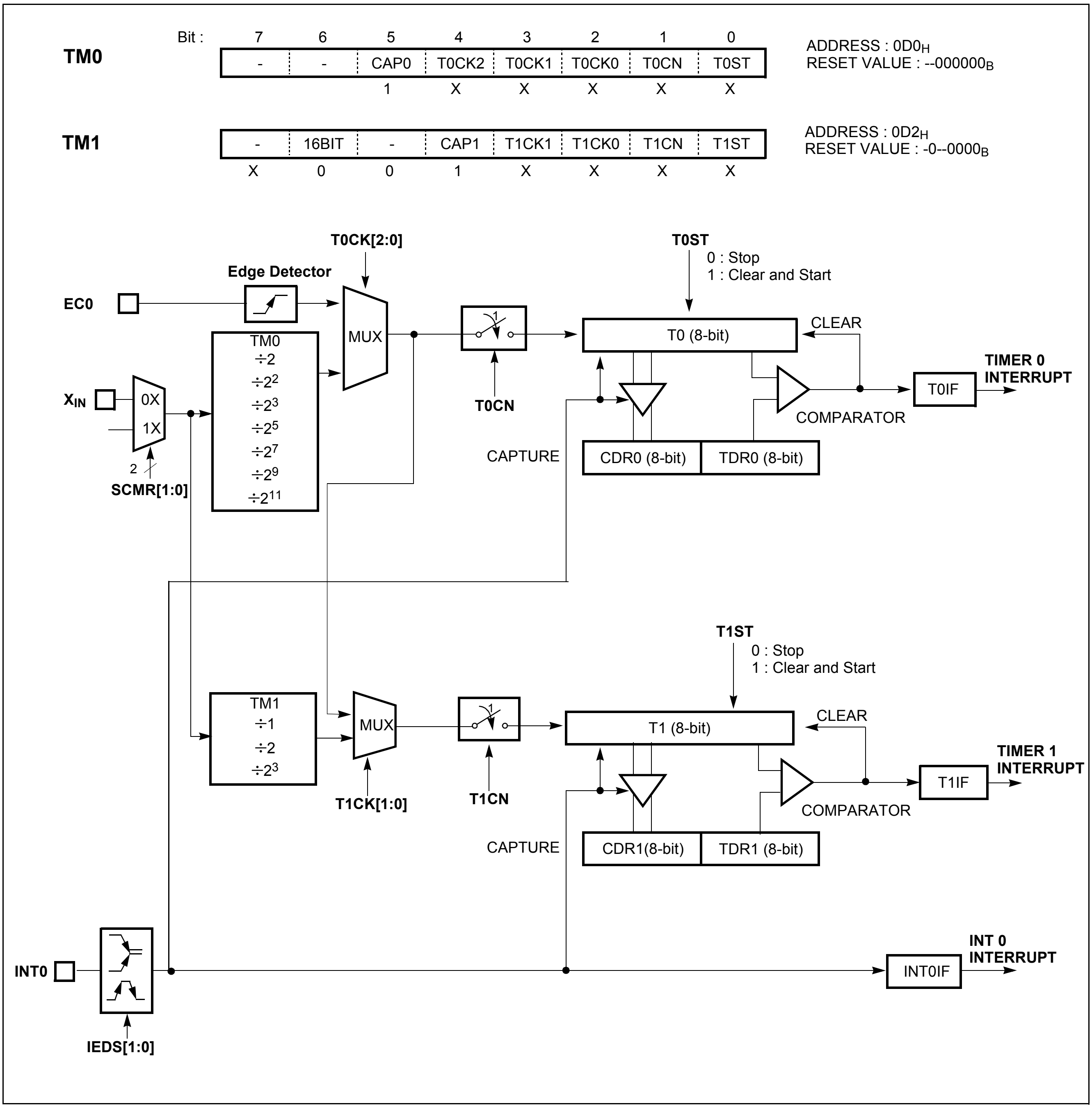



**Figure 12-10 8-bit Capture Mode (Timer0, Timer1)**

**TM2**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | CAP2 | T2CK2 | T2CK1 | T2CK0 | T2CN | T2ST |
| | | | 1 | X | X | X | X | X |

ADDRESS : $0D6_H$
RESET VALUE : $--000000_B$

**TM3**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| POL | 16BIT | PWM1E | CAP3 | T3CK1 | T3CK0 | T3CN | T3ST |
| X | 0 | 0 | 1 | X | X | X | X |

ADDRESS : $0D8_H$
RESET VALUE : $000\text{-}0000_B$

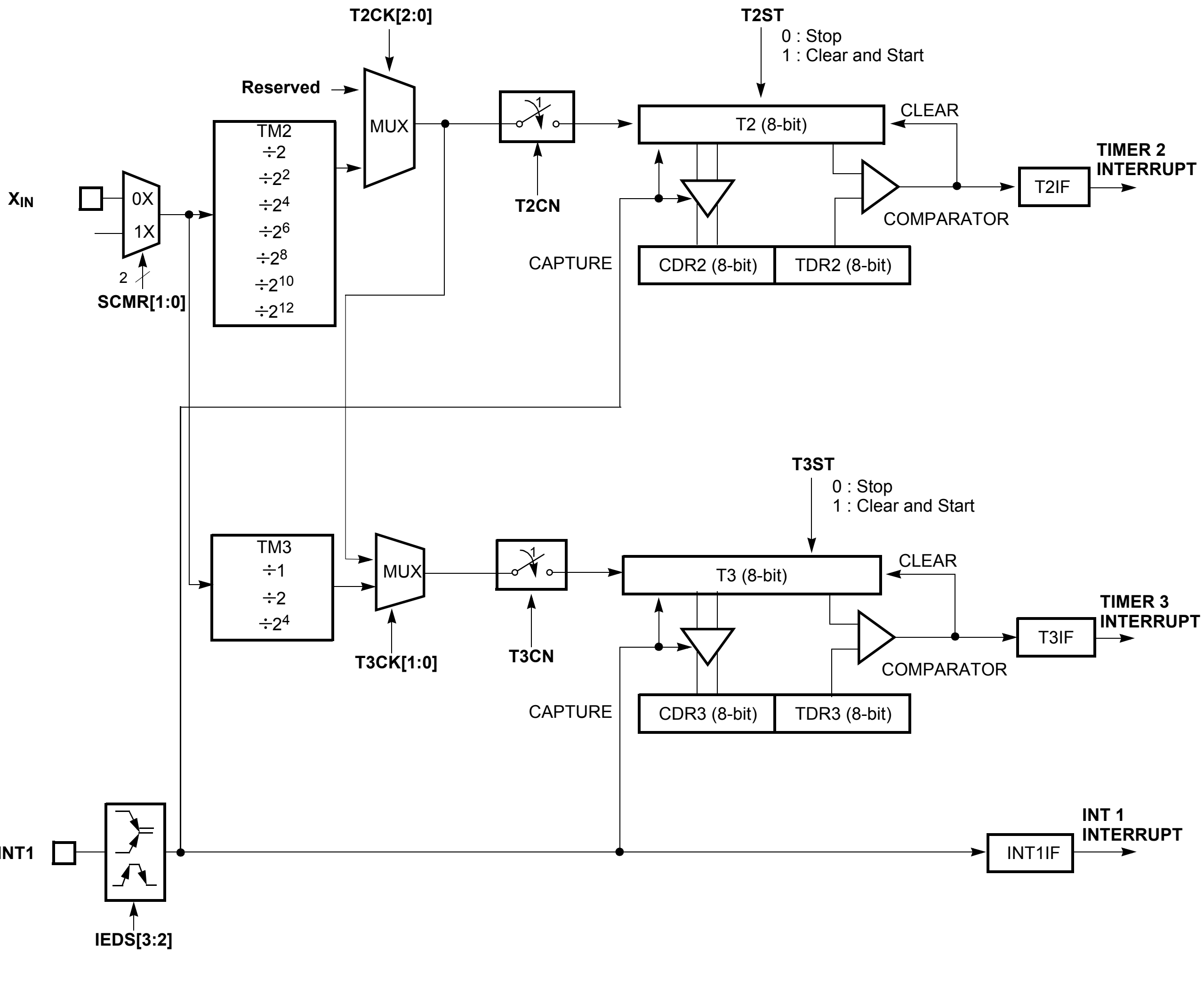



**Figure 12-11 8-bit Capture Mode (Timer2, Timer3)**

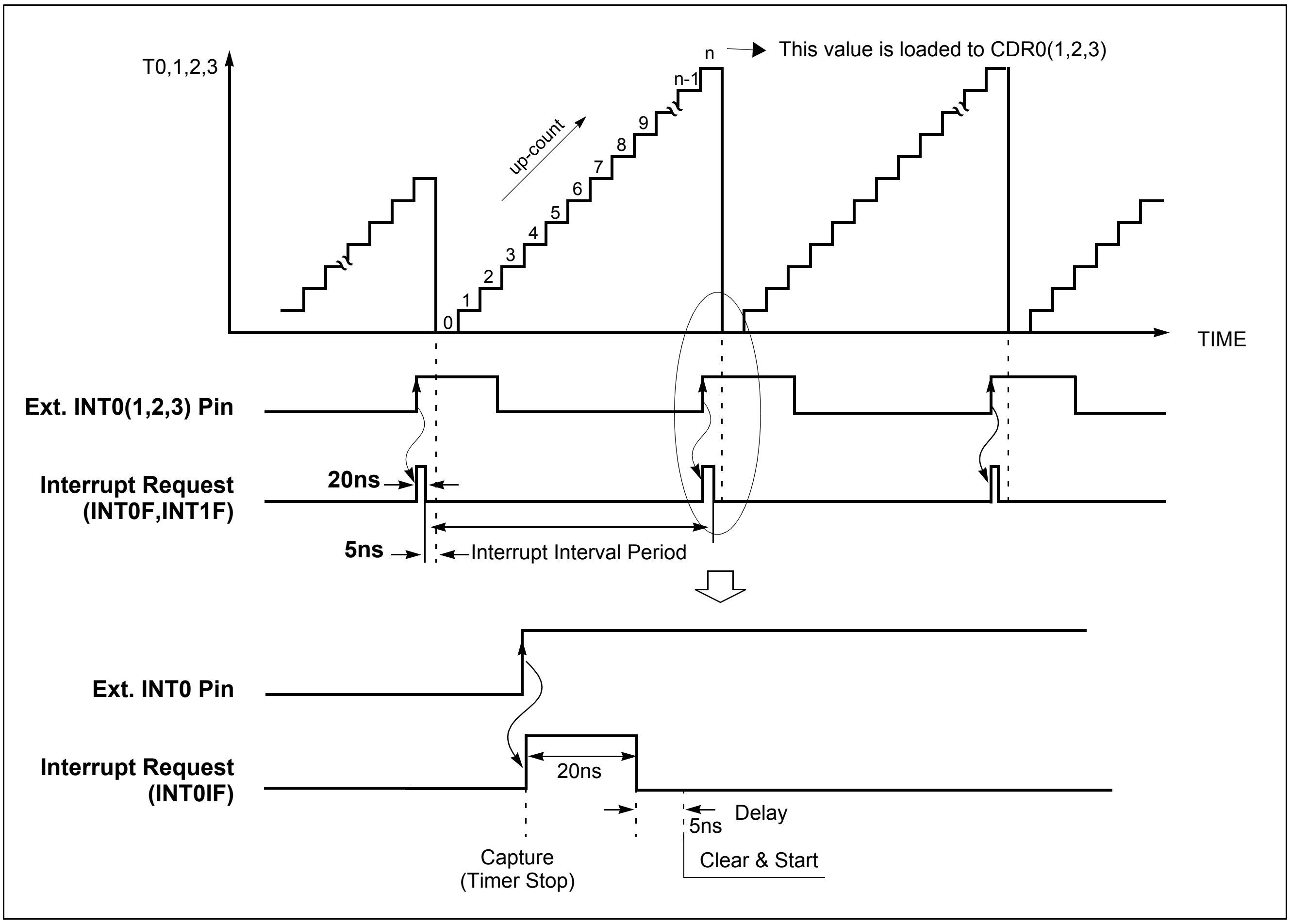



**Figure 12-12 Input Capture Operation**

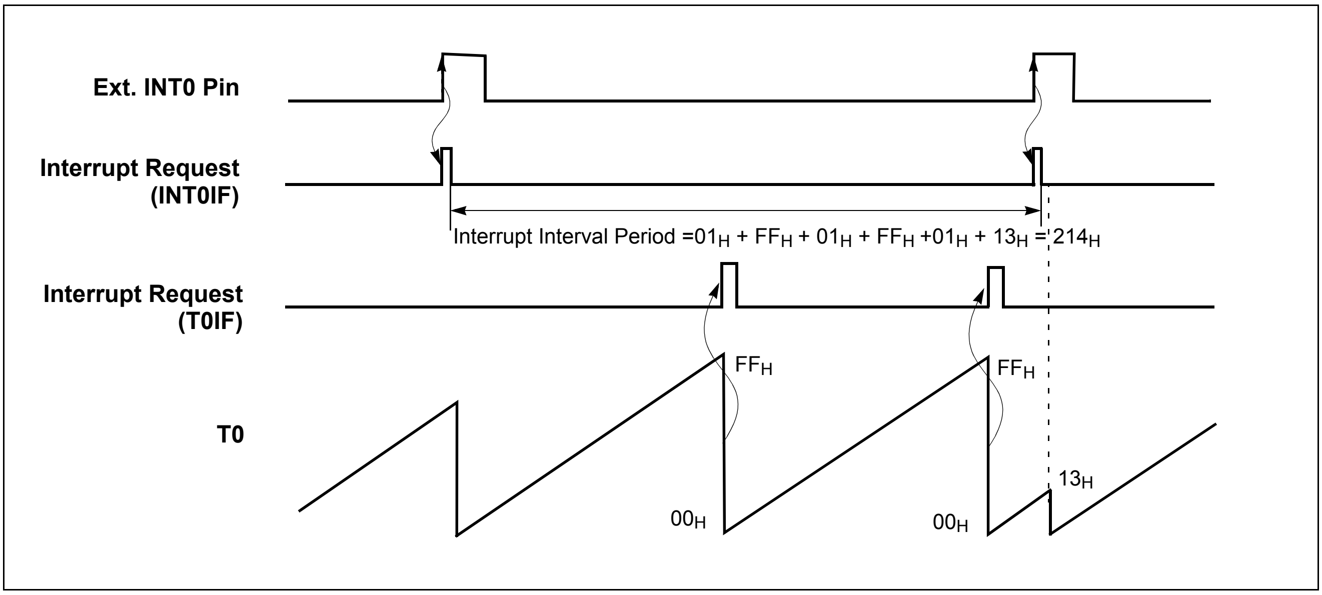



**Figure 12-13 Excess Timer Overflow in Capture Mode**

## 12.4 16-bit Capture Mode

16-bit capture mode is the same as 8-bit capture, except that the Timer register is running with 16 bits.

The clock source of the Timer 0,2 is selected either internal or external clock by bit TxCK2, TxCK1 and TxCK0.

In 16-bit mode, the bits TxCK1,TxCK0 and 16BIT of TM1,TM3 should be set to “1” respectively.

**TM0** — ADDRESS : $0D0_H$, RESET VALUE : $\text{--}000000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TM0 | - | - | CAP0 | T0CK2 | T0CK1 | T0CK0 | T0CN | T0ST |
| | | | 1 | X | X | X | X | X |

**TM1** — ADDRESS : $0D2_H$, RESET VALUE : $\text{-}0\text{-}00000_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TM1 | - | 16BIT | - | CAP1 | T1CK1 | T1CK0 | T1CN | T1ST |
| | X | 1 | 0 | 1 | 1 | 1 | X | X |

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

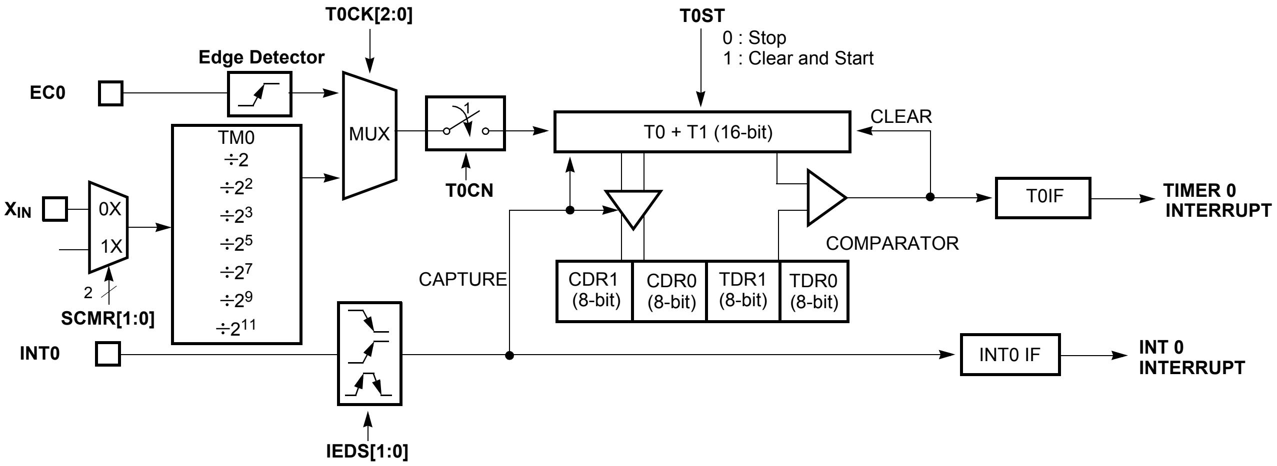



**Figure 12-14 16-bit Capture Mode (Timer0,1)**

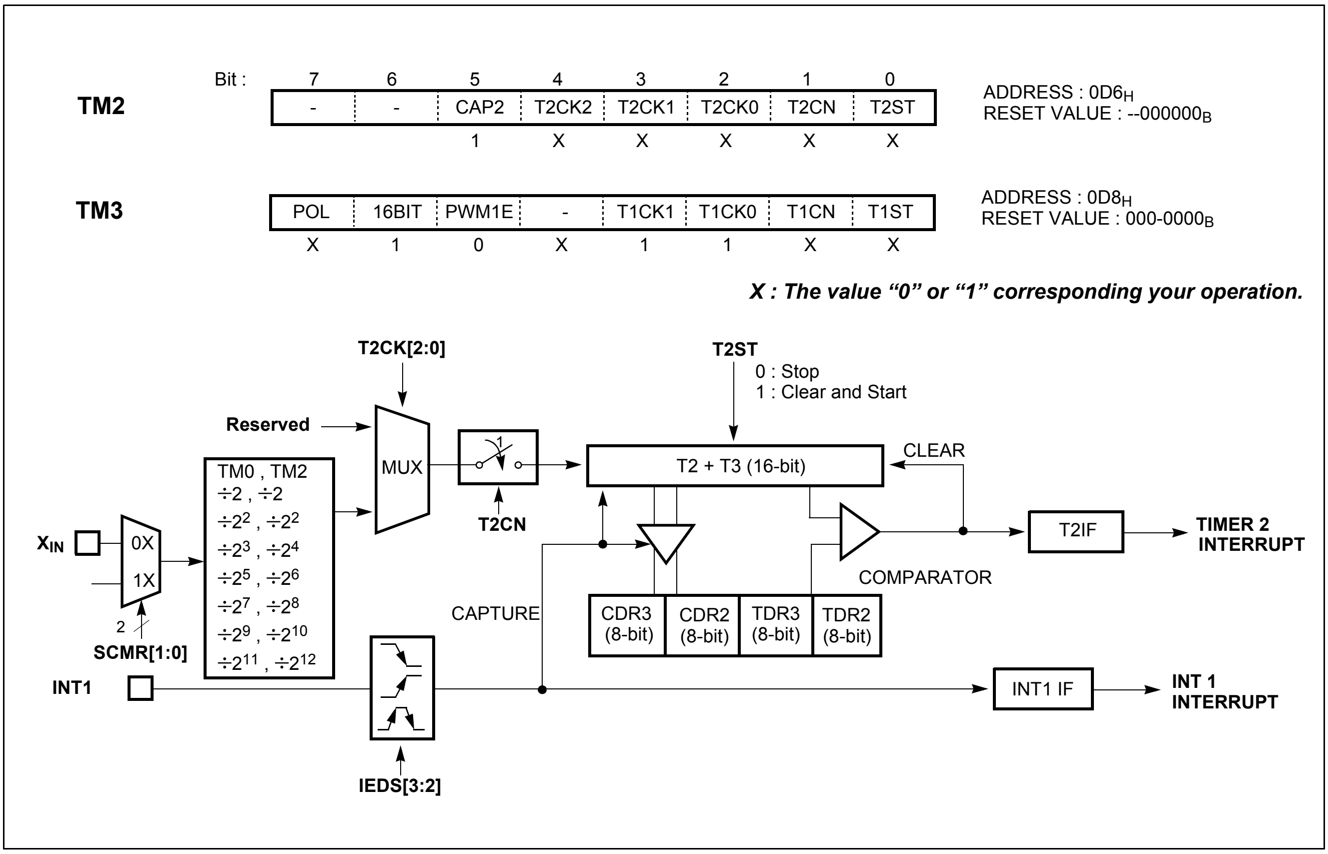



**Figure 12-15 16-bit Capture Mode (Timer2,3)**

## 12.5 8-Bit (16-Bit) Compare Output Mode

The MC80F7708A have a function of Timer Compare Output. To pulse out, the timer match can goes to port pin (R10) as shown in Figure 12-4 and Figure 12-8. Thus, pulse out is generated by the timer match. These operation is implemented to pin, R10/PWM1/T2O.

In this mode, the bit PWM1O of Port Mode Register R1FUNC should be set to “1”, and the bit PWM1E of Timer3 Mode Register (TM3) should be cleared to “0”.

In addition, 16-bit Compare output mode is available, also.

This pin output the signal having a 50: 50 duty square wave, and output frequency is same as below equation

$$f_{COMP} = \frac{f_{XIN}}{2 \times PrescalerValue \times (TDR + 1)}$$

## 12.6 PWM Mode

The MC80F7708A has one high speed PWM (Pulse Width Modulation) function which shared with Timer3. In PWM mode, the R10/PWM1 pins operate as a 10-bit resolution PWM output port. For this mode, the bit PWM1O of Port Mode Register (R1FUNC) and the bit PWM1E of timer3 mode register (TM3) should be set to “1” respectively.

The period of the PWM output is determined by the T3PPR (T3 PWM Period Register) and T3PWHR[3:2] (bit3, 2 of T3 PWM High Register) and the duty of the PWM output is determined by the T3PDR (T3 PWM Duty Register) and T3PWHR[1:0] (bit1, 0 of T3PWM High Register).

The user can use PWM data by writing the lower 8-bit period value to the T3PPR and the higher 2-bit period value to the T3PWHR[3:2]. And the duty value can be used with the T3PDR and the T3PWHR[1:0] in the same way.

The T3PDR is configured as a double buffering for glitchless PWM output. In Figure 12-16, the duty data is transferred from the master to the slave when the period data matched to the counted value. (i.e. at the beginning of next duty cycle).

The bit POL1 of TM3 decides the polarity of duty cycle.

The duty value can be changed when the PWM outputs. However the changed duty value is output after the current period is over. And it can be maintained the duty value at present output when changed only period value shown as Figure 12-18. As it were, the absolute duty time is not changed in varying frequency.

**Note:** *If the user need to change mode from the Timer3 mode to the PWM mode, the Timer3 should be stopped firstly, and then set period and duty register value. If user writes register values and changes mode to PWM mode while Timer3 is in operation, the PWM data would be different from expected data in the beginning.*

The relation of frequency and resolution is in inverse proportion. Table 12-2 shows the relation of PWM frequency vs. resolution.

***PWM Period = [T3PWHR[3:2]T3PPR+1] X Source Clock***

***PWM Duty = [T3PWHR[1:0]T3PDR+1] X Source Clock***

If it needed more higher frequency of PWM, it should be reduced resolution.

**Note:** *If the duty value and the period value are same, the PWM output is determined by the bit POL1 (1: High, 0: Low). And if the duty value is set to "$00_H$", the PWM output is determined by the bit POL1(1: Low, 0: High). The period value must be same or more than the duty value, and $00_H$ cannot be used as the period value.*

| Resolution | Frequency | | |
|---|---|---|---|
| | T3CK[1:0] =00 (250nS) | T3CK[1:0] =01 (500nS) | T3CK[1:0] =10 (2uS) |
| 10-bit | 3.9kHz | 1.95kHz | 0.49kHz |
| 9-bit | 7.8kHz | 3.9kHz | 0.98kHz |
| 8-bit | 15.6kHz | 7.8kHz | 1.95kHz |
| 7-bit | 31.2kHz | 15.6kHz | 3.90kHz |

**Table 12-2 PWM Frequency vs. Resolution at 4MHz**

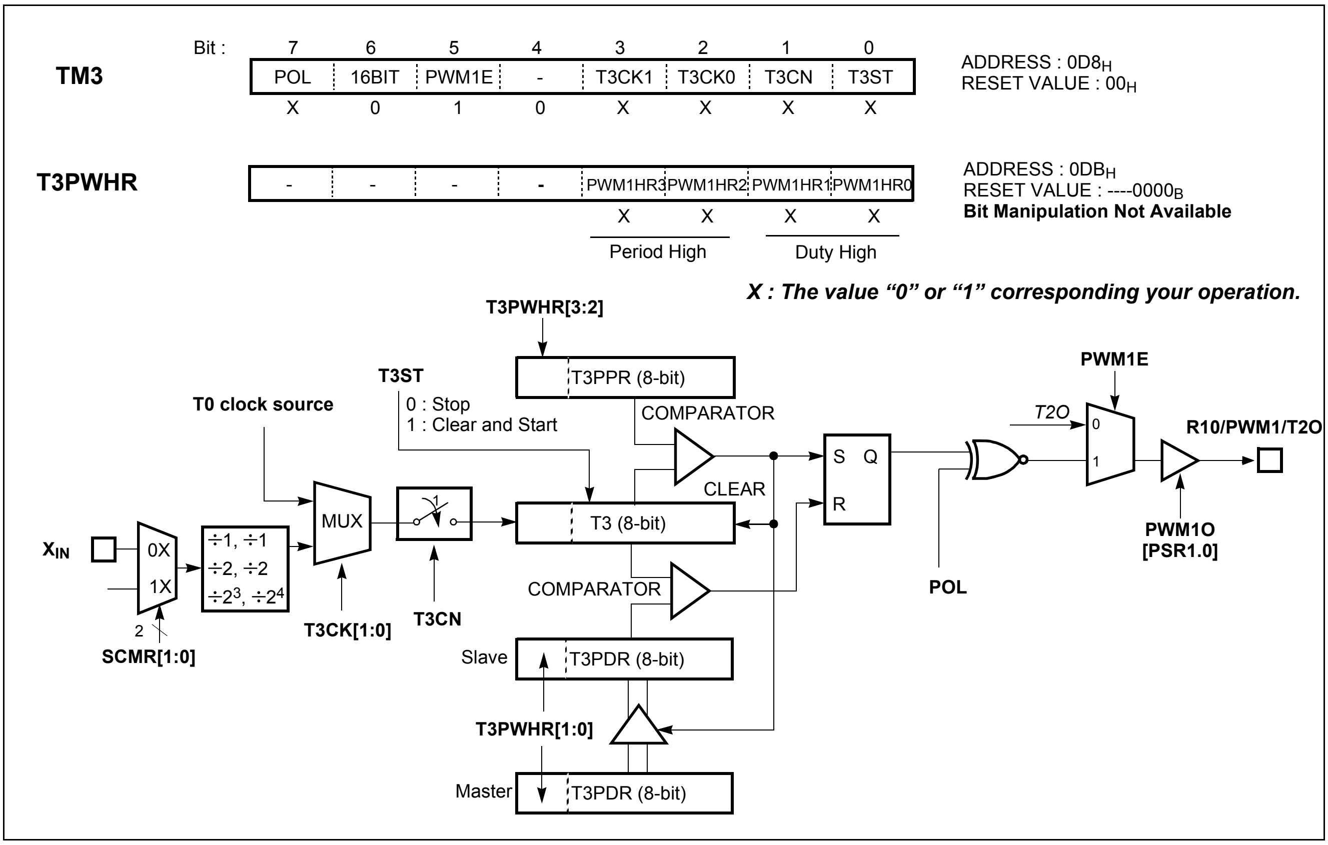



**Figure 12-16 PWM Mode**

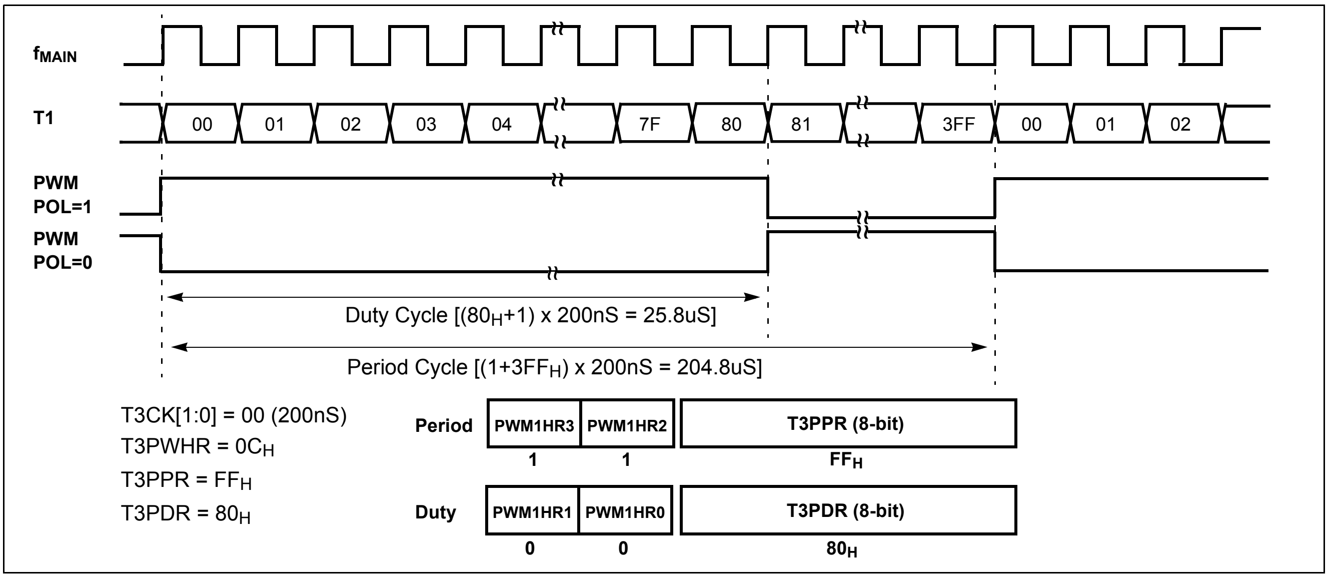



**Figure 12-17 Example of PWM at 5MHz**

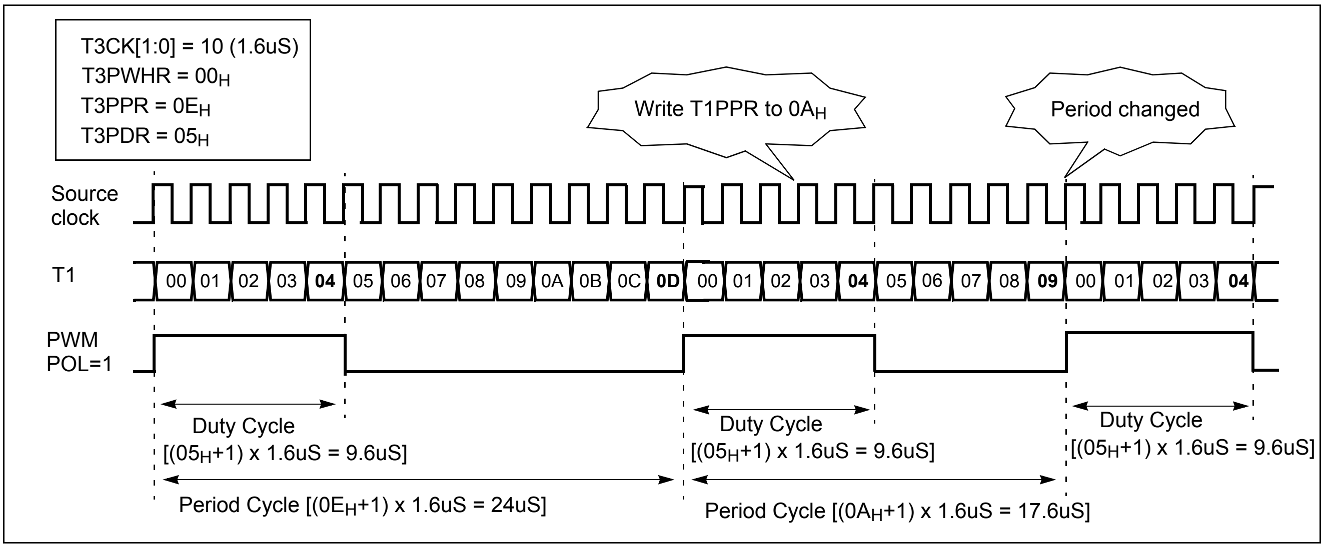



**Figure 12-18 Example of Changing the Period in Absolute Duty Cycle (@5MHz)**

**Example:**

```
Timer1 @4Mhz, 4kHz -20% duty PWM mode

   LDM  R1IO,#0000_XXX1B    ;R00 output
   LDM  TM3,#0010_0000B     ;pwm enable
   LDM  T3PWHR,#0000_1100B  ;20% duty
   LDM  T3PPR,#1110_0111B   ;period 250uS
   LDM  T3PDR,#1100_0111B   ;duty 50uS
   LDM  PSR1,#XXXX_XXX1B  ;set pwm port.
   LDM  TM3,#0010_0011B     ;timer1 start
```

X means don't care

## 13. WATCH TIMER

The watch timer generates interrupt for watch operation. The watch timer consists of the clock selector, 21-bit binary counter and watch timer mode register. It is a multi-purpose timer. It is generally used for watch design.

The bit 0, 1, 2 of WTMR select the clock source of watch timer among sub-clock, $f_{MAIN}÷2^8$ ,$f_{MAIN}÷2^7$ ,$f_{MAIN}$ or $f_{MAIN}÷2$ of main-clock and $f_{MAIN}$ of main-clock. The $f_{MAIN}$ of main-clock is used usually for watch timer test, so generally it is not used for the clock source of watch timer. The $f_{MAIN}÷2^7$ or $f_{MAIN}÷2^8$ clock is used when the single clock system is organized. If $f_{MAIN}÷2^8$ or $f_{MAIN}÷2^7$ clock is used as watch timer clock source, when the CPU enters into stop mode, the main clock is stopped and then watch timer is also stopped. If the sub-clock is used as the watch timer source clock, the watch timer count cannot be stopped. Therefore, the sub-clock does not stop and continues to oscillate even when the CPU is in the STOP mode. The timer counter consists of 21-bit binary counter and it can count to max 60 seconds at sub-clock.

The bit 3, 4 of WTMR select the interrupt request interval of watch timer among 2Hz, 4Hz, 16Hz and 1/64Hz.

***Note:*** *The Clock source of watch timer is also applied to LCD driver clock source. When selecting LCD driver clock source, the WTCK[2:0] should be set to appropriate value.*

***Note: Interrupt request flag(WTIF) must be erased after maximum 40us instruction executing time at watch timer interrupt service routine because it is may not cleared itself during interrupt acceptance processing. In case of releasing STOP mode, internal WTIF signal is not cleared for next sub-clock(0~30.5us at 32.768kHz).***

**WTRH (Watch Timer Read High Register)**

| | x | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|---|
| Bit : | - | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | WTRH6 | WTRH5 | WTRH4 | WTRH3 | WTRH2 | WTRH1 | WTRH0 |

ADDRESS: $0FF_H$
INITIAL VALUE:$0000\_0000_B$

**WTRH[6:0]: (WT data capture Value)**

**WTR (Watch Timer Register)**

| | W | W | W | W | W | W | W | W |
|---|---|---|---|---|---|---|---|---|
| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | WTCL | WT6 | WT5 | WT4 | WT3 | WT2 | WT1 | WT0 |

ADDRESS: $0E8_H$
INITIAL VALUE:$0111\_1111_B$

**WTCL (WT Clear)**
0: Free Run
1: WT Clear(Auto clear after 1cycle)

**WT[6:0] (WT Interrupt Interval Value)**
WT Interrupt Interval(IFWT) = $(fwck/2^{14})$ x (7bit WT Value+1)

**WTMR (Watch Timer Mode Register)**

| | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|
| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | WTEN | LOADEN | - | WTIN1 | WTIN0 | WTCK2 | WTCK1 | WTCK0 |

ADDRESS: $0EA_H$
INITIAL VALUE:$00\text{--}\_0000_B$

**WTEN (Watch Timer Enable Bit)**
0: Watch Timer Disable
1: Watch Timer Enable

**LOADEN (7bit reload Counter Write Enable Bit)**
0: Watch Dog Timer Write Enable
1: Watch Timer Write Enable

**WTIN[1:0] (Watch Timer Interrupt Interval Selection)**
00: $f_{wck} / 2^{11}$ [16Hz]*
01: $f_{wck} / 2^{13}$ [4Hz]*
10: $f_{wck} / 2^{14}$ [2Hz]*
11: $f_{wck} / 2^{14}$ x (7bit WT value+1) [2Hz x (7bit WT value+1)]*

**WTCK[2:0] (Watch Timer Clock Source Selection) : $f_{wck}$**
000: Sub. Clock ($f_{SUB}$)
001: Main Clock ($f_{MAIN}\div2^8$)
010: Main Clock ($f_{MAIN}\div2^7$)
011: Main Clock ($f_{MAIN}$)
100: Main Clock ($f_{MAIN}\div2$)

** When $f_{SUB}$ = 32.768 kHz and $f_{MAIN}$ = 4.19 MHz( $f_{MAIN} \div2^7$)*

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**Example:**

```
; 1 minute watch timer interrupt selection

   LDM   WTMR, #1101_1000B    ; T = 1/fSUB x 2^14 x (count+1)
   LDM   WTR, #1111_0111B     ; 080h + 119(count)
```

**Figure 13-1 Watch Timer Mode Register**

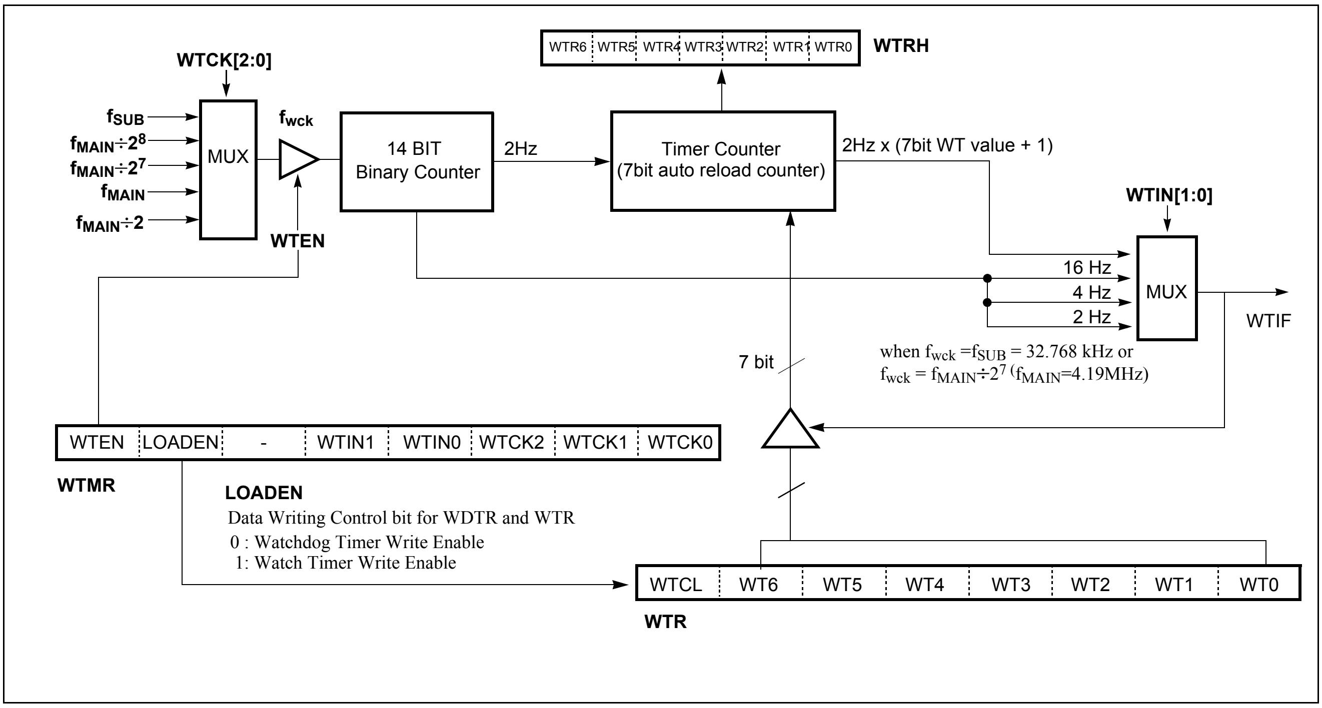



**Figure 13-2 Watch Timer Block Diagram**

### Usage of Watch Timer in STOP Mode

When the system is off and the watch should be kept working, follow the steps below.

1. Set the clock source of watch timer to sub-clock.
2. Enters into STOP mode.
3. After released by watch timer interrupt, counts up timer and refreshes LCD Display. When performing count up and refresh the LCD, the CPU operates in main frequency mode. **And clear WTIF approximately after maximum 40us delay time. If WTIF is not cleared by software, watch timer interrupt may be happened twice because internal WTIF signal is set during next input clock.**
4. Enters into STOP mode again.
5. Repeats 3 and 4.

When using STOP mode, if the watch timer interrupt interval is selected to 2Hz, the power consumption can be reduced considerably.

## 14. WATCH DOG TIMER

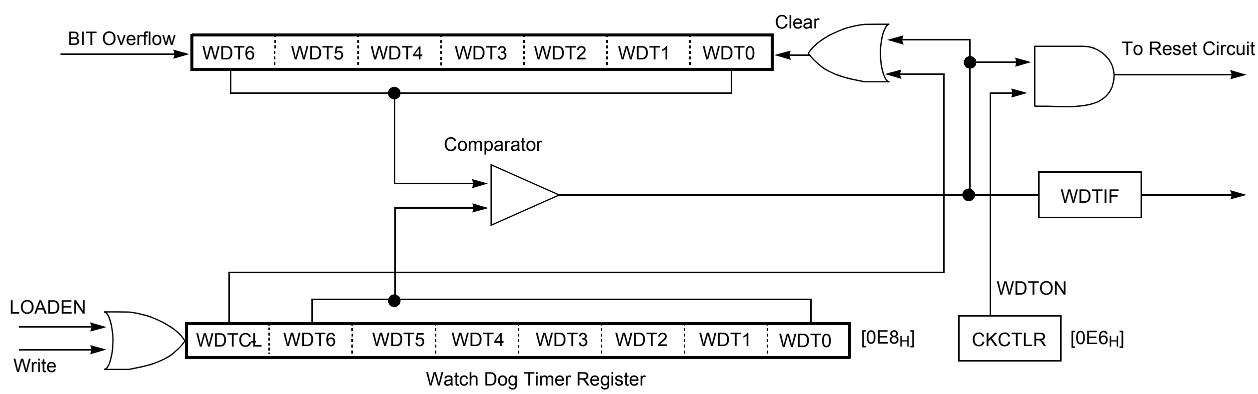
The watch dog timer (WDT) function is used for checking program malfunction due to external noise or other causes and return the operation to the normal contion.

The watchdog timer consists of 7-bit binary counter and the watchdog timer register(WDTR). The source clock of WDT is overflow of Basic Interval Timer. When the value of 7-bit binary counter is equal to the lower 7-bits of WDTR, the interrupt request flag is generated. This can be used as WDT interrupt or CPU reset signal in accordance with the bit WDTON. When WDTCL is set, 7-bit counter of WDT is reset. After one cycle, it is cleared by hardware.

When writing WDTR, the LOADEN bit of WTMR register should be cleared to “0”.

***Note:*** *WDTR and WTR has same address 0E8h. The LOADEN bit is used to select WDTR or WTR. When LOADEN of watch timer mode register(WTMR) is set to “1”, WDTR can not be wrote and WTR is wrote.*
*The LOADEN bit should be cleared to “0” when writing any value to WDTR.*

***Note:*** *When using watch dog timer, don’t write WDT[6:0] to “0000000”.*




**WDTR (Watch Dog Timer Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | W | W | W | W | W | W | W |
| | WDTCL | WDT6 | WDT5 | WDT4 | WDT3 | WDT2 | WDT1 | WDT0 |

ADDRESS: $0E8_H$
INITIAL VALUE:$0111\_1111_B$

**WDTCL (WDT Clear)**
0: Free Run
1: WDT Clear(Auto clear after 1cycle)

**WDT[6:0] (WDT Interrupt Value)**
WDT Interrupt Interval(IFWDT) = (BIT Interrupt Interval) x (WDT value)

**WTMR (Watch Timer Mode Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | WTEN | LOADEN | - | WTIN1 | WTIN0 | WTCK2 | WTCK1 | WTCK0 |

ADDRESS: $0EA_H$
INITIAL VALUE:$00\text{--}\_0000_B$

**LOADEN (7bit reload Counter Write Enable Bit)**
0: Watch Dog Timer Write Enable
1: Watch Timer Write Enable

**Figure 14-1 Block Diagram of Watch Dog Timer**

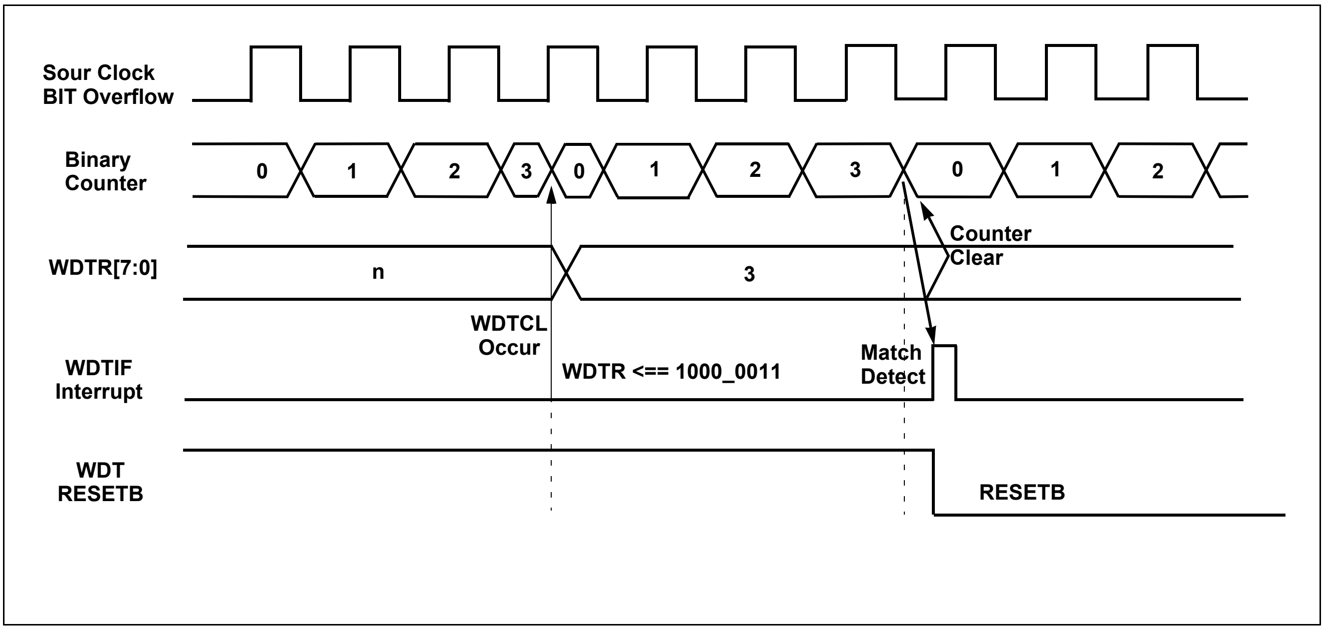



Figure 14-2 Watch Dog Timer Interrupt Time

## 15. ANALOG TO DIGITAL CONVERTER

The analog-to-digital(A/D) converter allows conversion of an analog input signal to an corresponding 8-bit digital value. The A/D module has six analog inputs, which are multiplexed into one sample and hold. The output of the sample and hold is the input of the converter, which generates the result via successive approximation. The analog supply voltage is connected to $V_{DD}$ of ladder resistance of A/D module.

The A/D module has three registers which are the control register ADCM and A/D result register ADCRL. The ADCM register, shown in Figure 15-2, controls the operation of the A/D converter module. The port pins can be configured as analog inputs or digital I/O. To use analog inputs, each port should be assigned analog input port by setting R2IO direction register as input mode and setting ADS[2:0] to select the corresponding channel.

The processing of conversion is start when the start bit ADST is set to “1”. After one cycle, it is cleared by hardware. The register ADCRL contain the result of the A/D conversion. ADCRL contains the result of the A/D conversion. When the conversion is completed, the result is loaded into the ADCRL, the A/D conversion status bit ADF is set to “1”. The block diagram of the A/D module is shown in Figure 15-1. The A/D status bit ADF is automatically set when A/D conversion is completed, cleared when A/D conversion is in process. The conversion time takes maximum 10uS(at fXIN=8MHz).

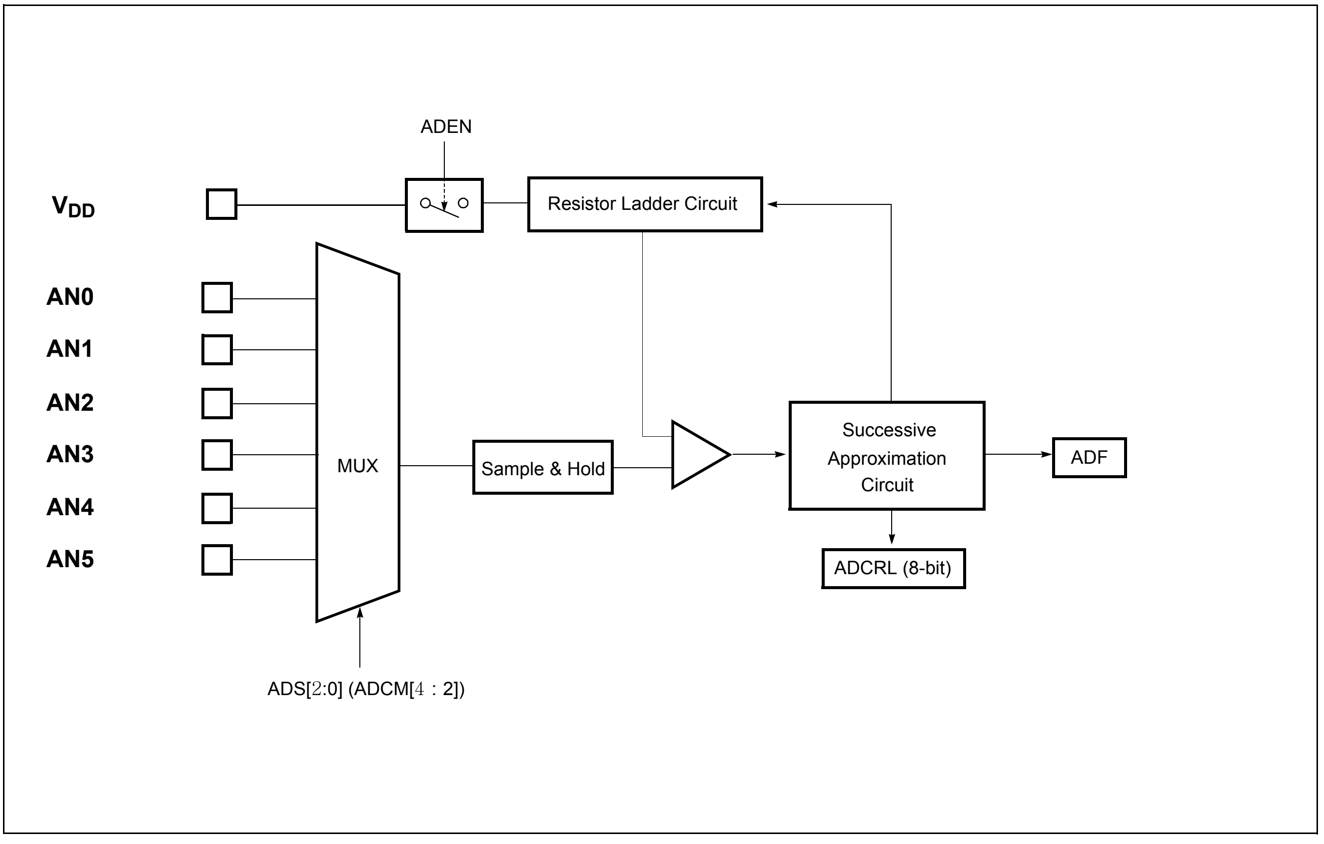



**Figure 15-1 A/D Converter Block Diagram & Registers**

**ADCM (A/D Converter Mode Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | | | | R/W | R/W | R/W | R |
| | ADEN | 1 | 0 | ADS2 | ADS1 | ADS0 | ADST | ADF |

ADDRESS : $0E2_H$
RESET VALUE : $\text{-0-00001}_B$

**ADEN (A/D Converter Enable bit)**
1 : Enable
0 : Disable

**ADST (A/D Start bit)**
1 : A/D Conversion is started
After 1 cycle, cleared to "0"
0 : Bit force to zero

**ADF (A/D Status bit)**
0 : A/D Conversion is in process
1 : A/D Conversion is completed

**ADS[2:0] (A/D Converter Input Selection)**
000 : Channel 0 (R20/AN0)
001 : Channel 1 (R21/AN1)
010 : Channel 2 (R22/AN2)
011 : Channel 3 (R23/AN3)
100 : Channel 4 (R24/AN4)
101 : Channel 5 (R25/AN5)
110 : Reserved
111 : Reserved

**Note : R24/AN4 and R25/AN5 are not supported in MC80F7708AK/AD.**

**ADCRH (A/D Converter Result High Register: EVA only)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | W | W | W | R | R | R | R | R |
| | 0 | 0 | 1 | - | - | - | - | - |

ADDRESS : $0E4_H$
RESET VALUE : Undefined

*** Unused bit of ADCRH should be set as**
**Bit7 : "0"**
**Bit6 : "0"**
**Bit5 : "1"**

**ADCRL (A/D Converter Result Low Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R | R | R | R | R | R | R | R |
| | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |

ADDRESS : $0E3_H$
RESET VALUE : Undefined

***The conversion time takes maximum 10us at 8MHz and 20us at 4MHz.**

**Figure 15-2 A/D Converter Mode & Result Registers**

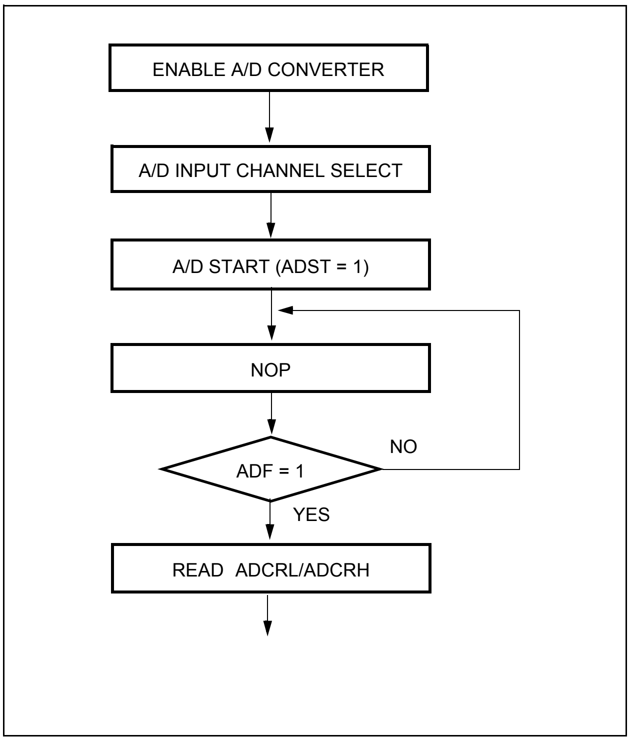



**Figure 15-3 A/D Converter Operation Flow**

### A/D Converter Cautions

(1) Input range of AN0 to AN5

The input voltages of AN0 to AN7 should be within the specification range. In particular, if a voltage above $AV_{DD}$ or below $V_{SS}$ is input (even if within the absolute maximum rating range), the conversion value for that channel can not be determinated. The conversion values of the other channels may also be affected.

(2) Noise counter measures

In order to maintain 8-bit resolution, any attention must be paid to noise on pins $AV_{DD}$ and AN0 to AN7. Since the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor is connected externally as shown below in order to reduce noise.

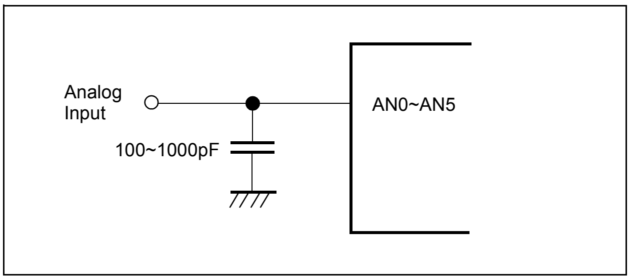



**Figure 15-4 Analog Input Pin Connecting Capacitor**

(3) Pins AN0/R20 to AN5/R25

The analog input pins AN0 to AN5 also function as input/output port (PORT R2) pins. When A/D conversion is performed with any of pins AN0 to AN57 selected, be sure not to execute a PORT input instruction while conversion is in progress, as this may reduce the conversion resolution.

Also, if digital pulses are applied to a pin adjacent to the pin in the process of A/D conversion, the expected A/D conversion value may not be obtainable due to coupling noise. Therefore, avoid applying pulses to pins adjacent to the pin undergoing A/D conversion.

(4) $AV_{DD}$ pin input impedance

A series resistor string of approximately 10KΩ is connected between the $AV_{DD}$ pin and the $V_{SS}$ pin.

Therefore, if the output impedance of the reference voltage source is high, this will result in parallel connection to the series resistor string between the $AV_{DD}$ pin and the $V_{SS}$ pin, and there will be a large reference voltage error.

## 16. BUZZER OUTPUT FUNCTION

The buzzer driver consists of 6-bit binary counter, the buzzer driver register BUZR and the clock selector. It generates square-wave which is very wide range frequency (500 Hz~125 kHz at $f_{MAIN}$ = 4MHz) by user programmable counter.

Pin R04/BUZO is assigned for output port of Buzzer driver by setting the bit BUZO of R0 Function Register(R0FUNC) to "1".

The 6-bit buzzer counter is cleared and start the counting by writing signal to the register BUZR. It is increased from $00_H$ until it matches with BUR[5:0].

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

The bit 0 to 5 of BUZR determines output frequency for buzzer driving. BUZR[5:0] is initialized to $3F_H$ after reset. Note that BUZR is a write-only register. Frequency calculation is following as shown below.

$$f_{BUZ} = \frac{f_{XIN}}{2 \times DivideRatio \times (BUZR[5:0] + 1)}$$

The bits BUCK1, BUCK0 of BUZR select the source clock from prescaler output.

$f_{BUZ}$: Buzzer frequency
$f_{XIN}$: Oscillator frequency
Divide Ratio: Prescaler divide ratio by BUCK[1:0]
BUZR[5:0]: Lower 6-bit value of BUZR. Buzzer control data
.

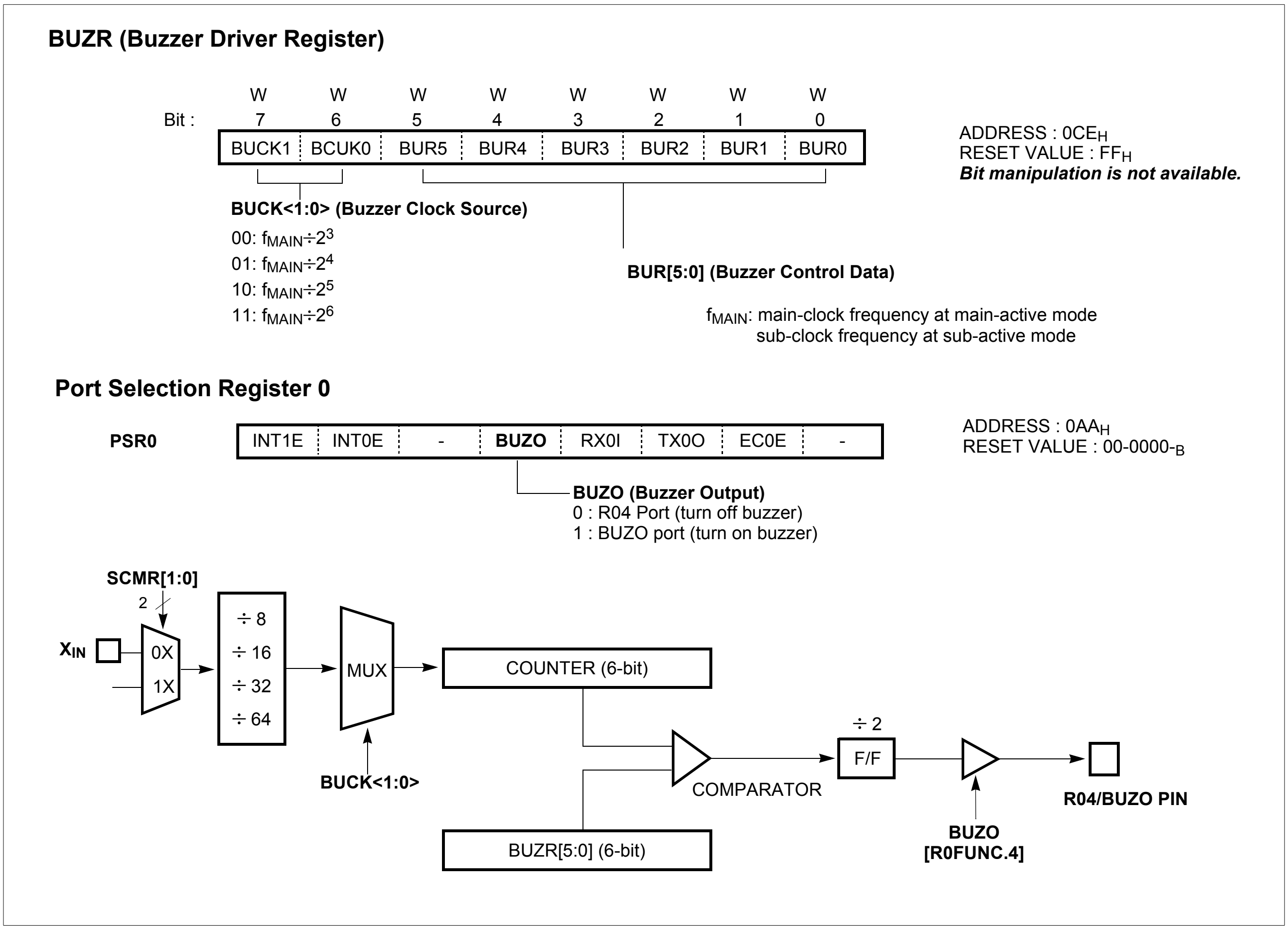



**Figure 16-1 Buzzer Driver**

Example: 2.5kHz output at 4MHz.

```
LDM    R0FUNC,#XXX1_XXXXB
LDM    BUZR,#1001_1000B
```

X means don't care

### Buzzer Output Frequency

When main-frequency is 4MHz, buzzer frequency is shown as below.

| BUZR [5:0] | Frequency Output (kHz) BUZR[7:6] | | | | BUZR [5:0] | Frequency Output (kHz) BUZR[7:6] | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 00 | 01 | 10 | 11 | | 00 | 01 | 10 | 11 |
| 00 | 250.000 | 125.000 | 62.500 | 31.250 | 20 | 7.576 | 3.788 | 1.894 | 0.947 |
| 01 | 125.000 | 62.500 | 31.250 | 15.625 | 21 | 7.353 | 3.676 | 1.838 | 0.919 |
| 02 | 83.333 | 41.667 | 20.833 | 10.417 | 22 | 7.143 | 3.571 | 1.786 | 0.893 |
| 03 | 62.500 | 31.250 | 15.625 | 7.813 | 23 | 6.944 | 3.472 | 1.736 | 0.868 |
| 04 | 50.000 | 25.000 | 12.500 | 6.250 | 24 | 6.757 | 3.378 | 1.689 | 0.845 |
| 05 | 41.667 | 20.833 | 10.417 | 5.208 | 25 | 6.579 | 3.289 | 1.645 | 0.822 |
| 06 | 35.714 | 17.857 | 8.929 | 4.464 | 26 | 6.410 | 3.205 | 1.603 | 0.801 |
| 07 | 31.250 | 15.625 | 7.813 | 3.906 | 27 | 6.250 | 3.125 | 1.563 | 0.781 |
| 08 | 27.778 | 13.889 | 6.944 | 3.472 | 28 | 6.098 | 3.049 | 1.524 | 0.762 |
| 09 | 25.000 | 12.500 | 6.250 | 3.125 | 29 | 5.952 | 2.976 | 1.488 | 0.744 |
| 0A | 22.727 | 11.364 | 5.682 | 2.841 | 2A | 5.814 | 2.907 | 1.453 | 0.727 |
| 0B | 20.833 | 10.417 | 5.208 | 2.604 | 2B | 5.682 | 2.841 | 1.420 | 0.710 |
| 0C | 19.231 | 9.615 | 4.808 | 2.404 | 2C | 5.556 | 2.778 | 1.389 | 0.694 |
| 0D | 17.857 | 8.929 | 4.464 | 2.232 | 2D | 5.435 | 2.717 | 1.359 | 0.679 |
| 0E | 16.667 | 8.333 | 4.167 | 2.083 | 2E | 5.319 | 2.660 | 1.330 | 0.665 |
| 0F | 15.625 | 7.813 | 3.906 | 1.953 | 2F | 5.208 | 2.604 | 1.302 | 0.651 |
| 10 | 14.706 | 7.353 | 3.676 | 1.838 | 30 | 5.102 | 2.551 | 1.276 | 0.638 |
| 11 | 13.889 | 6.944 | 3.472 | 1.736 | 31 | 5.000 | 2.500 | 1.250 | 0.625 |
| 12 | 13.158 | 6.579 | 3.289 | 1.645 | 32 | 4.902 | 2.451 | 1.225 | 0.613 |
| 13 | 12.500 | 6.250 | 3.125 | 1.563 | 33 | 4.808 | 2.404 | 1.202 | 0.601 |
| 14 | 11.905 | 5.952 | 2.976 | 1.488 | 34 | 4.717 | 2.358 | 1.179 | 0.590 |
| 15 | 11.364 | 5.682 | 2.841 | 1.420 | 35 | 4.630 | 2.315 | 1.157 | 0.579 |
| 16 | 10.870 | 5.435 | 2.717 | 1.359 | 36 | 4.545 | 2.273 | 1.136 | 0.568 |
| 17 | 10.417 | 5.208 | 2.604 | 1.302 | 37 | 4.464 | 2.232 | 1.116 | 0.558 |
| 18 | 10.000 | 5.000 | 2.500 | 1.250 | 38 | 4.386 | 2.193 | 1.096 | 0.548 |
| 19 | 9.615 | 4.808 | 2.404 | 1.202 | 39 | 4.310 | 2.155 | 1.078 | 0.539 |
| 1A | 9.259 | 4.630 | 2.315 | 1.157 | 3A | 4.237 | 2.119 | 1.059 | 0.530 |
| 1B | 8.929 | 4.464 | 2.232 | 1.116 | 3B | 4.167 | 2.083 | 1.042 | 0.521 |
| 1C | 8.621 | 4.310 | 2.155 | 1.078 | 3C | 4.098 | 2.049 | 1.025 | 0.512 |
| 1D | 8.333 | 4.167 | 2.083 | 1.042 | 3D | 4.032 | 2.016 | 1.008 | 0.504 |
| 1E | 8.065 | 4.032 | 2.016 | 1.008 | 3E | 3.968 | 1.984 | 0.992 | 0.496 |
| 1F | 7.813 | 3.906 | 1.953 | 0.977 | 3F | 3.906 | 1.953 | 0.977 | 0.488 |

**Table 16-1 Buzzer Output Frequency**

# 17. INTERRUPTS

The MC80F7708A interrupt circuits consist of Interrupt enable register (IENH, IENM, IENL), Interrupt request flag register(IRQH, IRQM, IRQL), Interrupt flag register(INTFH, INTFL), Interrupt Edge Selection Register (IEDS), priority circuit and Master enable flag ("I" flag of PSW). The interrupts are controlled by the interrupt master enable flag I-flag (bit 2 of PSW), the interrupt enable register and the interrupt request flag register except Power-on reset and software BRK interrupt. The configuration of interrupt circuit is shown in Figure 17-1 and interrupt priority is shown in Table 17-1 .

**Table 17-1 Vector Table**

| Reset/Interrupt | Symbol | Priority | Vector Addr. |
|---|---|---|---|
| Hardware Reset | RESET | 0 | FFFEH |
| External Int. 0 | INTR0 | 1 | FFFAH |
| External Int. 1 | INTR1 | 2 | FFF8H |
| External Int. 2 | INTR2 | 3 | FFF6H |
| External Int. 3 | INTR3 | 4 | FFF4H |
| – | – | – | – |
| – | – | – | – |
| Timer 0 Int. | T0 | 5 | FFEEH |
| Timer 1 Int. | T1 | 6 | FFECH |
| Timer 2 Int. | T2 | 7 | FFEAH |
| Timer 3 Int. | T3 | 8 | FFE8H |
| - | - | - | - |
| BIT Int. | BIT | 10 | FFE2H |
| Watch Dog timer int. | WDT | 11 | FFE0H |
| Watch timer int. | WT | 12 | FFE0H |

Each bit of interrupt request flag registers(IRQH, IRQM, IRQL) in Figure 17-1 is set when corresponding interrupt condition is met. The interrupt request flags that actually generate external interrupts are bit INT0F, INT1F and INT2F in Register IRQH and INT3F in Register IRQL. The External Interrupts INT0, INT1, INT2 and INT3 can each be transition-activated (1-to-0, 0-to-1 and both transition). The RX0 and TX0 of UART0 Interrupts are generated by RX0IF and TX0IF which are set by finishing the reception and transmission of data.

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

The Basic Interval Timer Interrupt is generated by BITIF which is set by overflow of the Basic Interval Timer Register (BITR). The Watch dog Interrupt is generated by WDTIF which set by a match in Watch dog timer register (when the bit WDTON is set to "0"). The Watch Timer Interrupt is generated by WTIF which is set periodically according to the established time interval.

When an interrupt is generated, the bit of interrupt request flag register(IRQH, IRQM, IRQL) that generated it may not cleared itself during interrupt acceptance processing. After interrupt acceptance, it should be cleard in interrupt service routine.

Each bit of Interrupt flag register(INTFH, INTFL) is set when corresponding interrupt flag bit as well as interrupt enable bit are set. The bits of interrupt flag register are never cleared by the hardware although the service routine is vectored to. Therefore, the interrupt flag register can be used to distinguish a right interrupt source from two available ones in a vector address. For example, RX0 and TX0 which have the same vector address($FFF2_H$) may be distinguished by INTFH register.

Interrupt enable registers are shown in Figure 17-2. These registers are composed of interrupt enable bits of each interrupt source, these bits determine whether an interrupt will be accepted or not. When enable bit is "0", a corresponding interrupt source is prohibited. Note that PSW contains also a master enable bit, I-flag, which disables all interrupts at once. When an interrupt is occurred, the I-flag is cleared and disable any further interrupt, the return address and PSW are pushed into the stack and the PC is vectored to.

In an interrupt service routine, any other interrupt may be serviced. The source(s) of these interrupts can be determined by polling the interrupt request flag bits. Then, the interrupt request flag bit(s) must be cleared by software before re-enabling interrupts to avoid recursive interrupts. The Interrupt Request flags are able to be read and written.

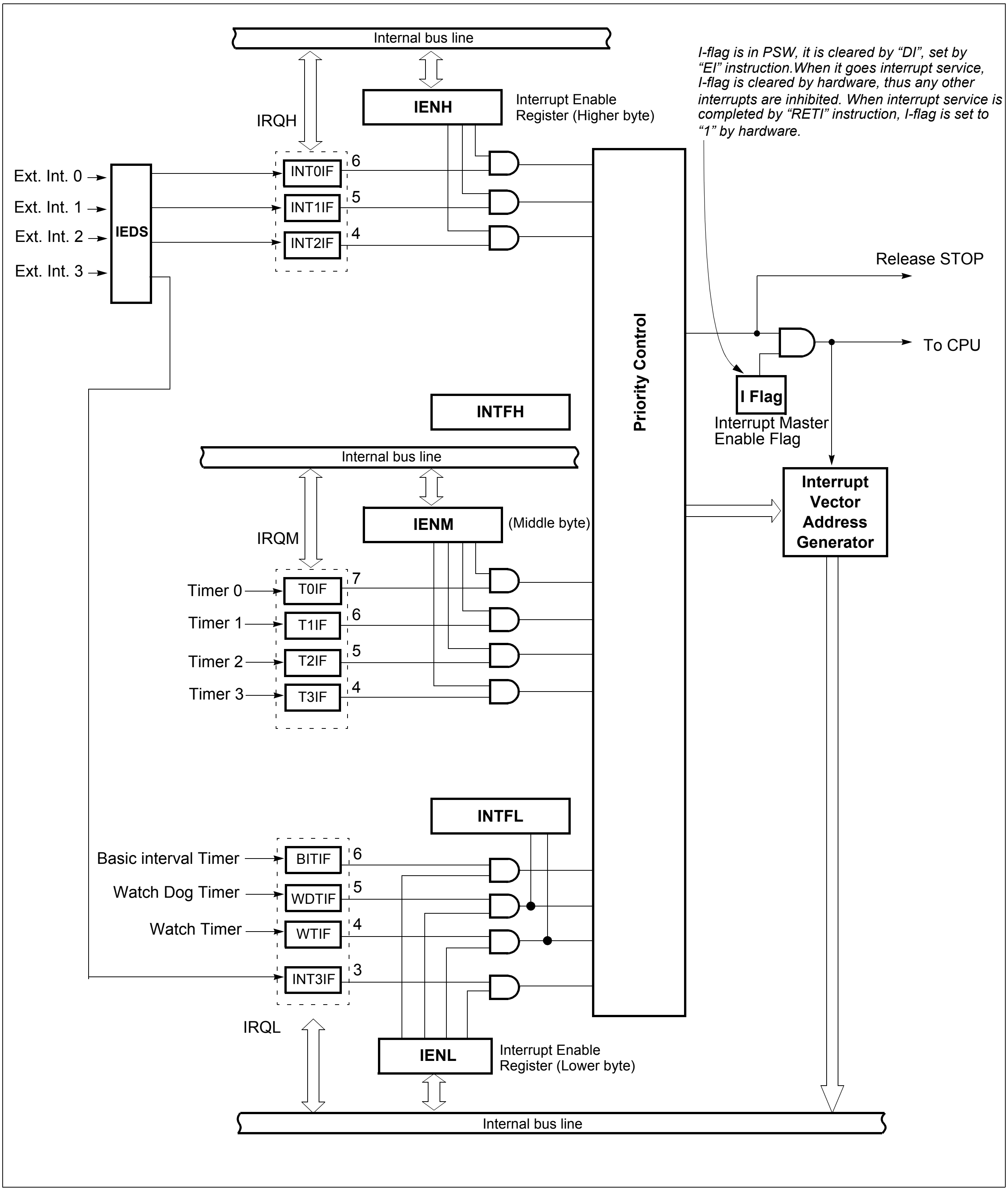



**Figure 17-1 Block Diagram of Interrupt**

**IENH (Interrupt Enable High Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | R/W | | |
| | - | INT0E | INT1E | INT2E | 0 | 0 | - | - |

ADDRESS : $0F6_H$
RESET VALUE : $-00-00--_B$

*** Unused bit of IENH should be set as**
**Bit3 : "0"**
**Bit2 : "0"**

**IENM (Interrupt Enable Middle Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | | | | |
| | T0E | T1E | T2E | T3E | - | - | - | - |

ADDRESS : $0F7_H$
RESET VALUE : $0000---0_B$

**IENL (Interrupt Enable Low Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | | | |
| | - | BITE | WDTE | WTE | INT3E | - | - | - |

ADDRESS : $0F8_H$
RESET VALUE : $-000----_B$

Enables or disables the interrupt individually

0 : Disable
1 : Enable

**IRQH (Interrupt Request Flag High Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | R/W | | |
| | - | INT0IF | INT1IF | INT2IF | 0 | 0 | - | - |

ADDRESS : $0F9_H$
RESET VALUE : $-00000--_B$

*** Unused bit of IRQH should be set as**
**Bit3 : "0"**
**Bit2 : "0"**

**IRQM (Interrupt Request Flag Middle Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | | | R/W | |
| | T0IF | T1IF | T2IF | T3IF | - | - | - | - |

ADDRESS : $0FA_H$
RESET VALUE : $0000---0_B$

**IRQL (Interrupt Request Flag Low Register)**

RESET VALUE : $-000----_B$

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | | | |
| | - | BITIF | WDTIF | WTIF | INT3IF | - | - | - |

ADDRESS : $0FB_H$

0 : Interrupt not occurred
1 : Interrupt Request

**INTFH (Interrupt Flag Register Low:EVA only)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | R/W | R/W | | |
| | - | - | - | - | 0 | 0 | - | - |

ADDRESS : $0F4_H$
RESET VALUE : $00----00_B$

*** Unused bit of INTFH should be set as**
**Bit3 : "0"**
**Bit2 : "0"**

**INTFL (Interrupt Flag Register Low)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | R/W |
| | - | - | - | - | - | - | WTF | WDTF |

ADDRESS : $0F5_H$
RESET VALUE : $00----00_B$

**WTF**(WT Interrupt Flag)
0 : No Generation
1 : Generation

**WDTF**(WDT Interrupt Flag)
0 : No Generation
1 : Generation

**Figure 17-2 Interrupt Enable Registers and Interrupt Request Registers**

## 17.1 Interrupt Sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to "0" by a reset or an instruction. Interrupt acceptance sequence requires 8 $f_{OSC}$ (2 μs at $f_{MAIN}$=4MHz) after the completion of the current instruction execution. The interrupt service task is terminated upon execution of an interrupt return instruction [RETI].

### Interrupt acceptance

1. The interrupt master enable flag (I-flag) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.
2. Interrupt request flag for the interrupt source accepted is cleared to "0".
3. The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack area. The stack pointer decreases 3 times.
4. The entry address of the interrupt service program is read from the vector table address and the entry address is loaded to the program counter.
5. The instruction stored at the entry address of the interrupt service program is executed.

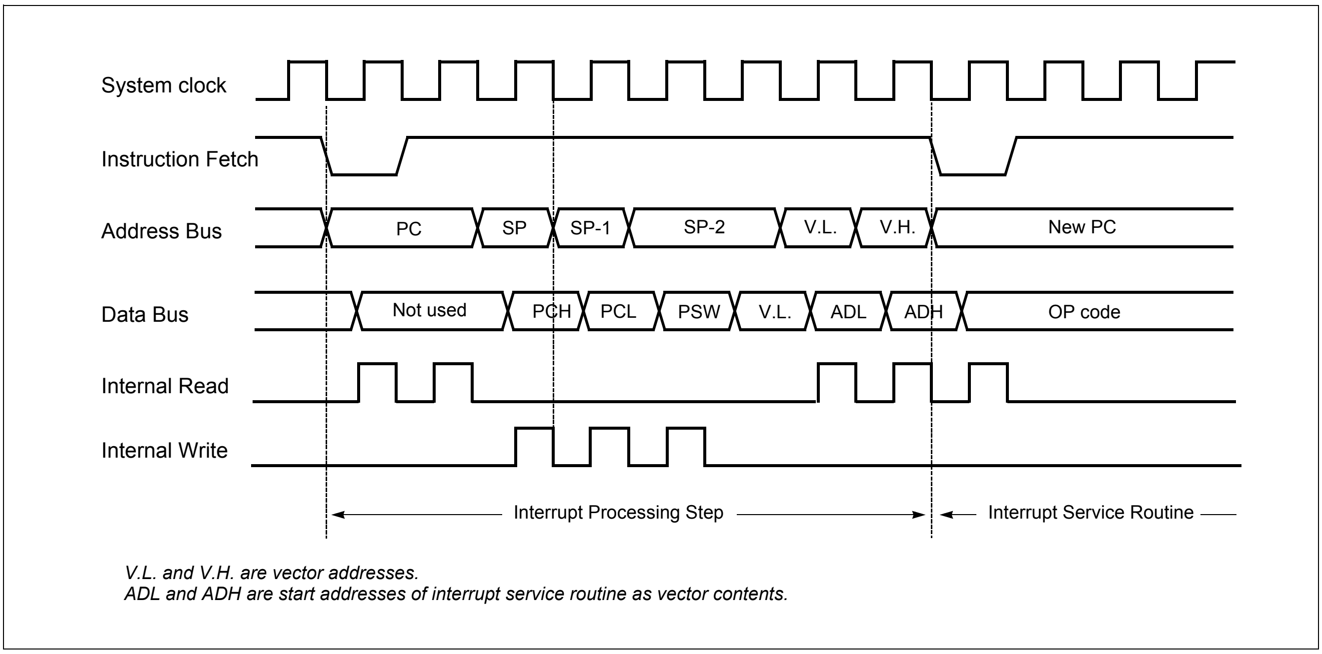
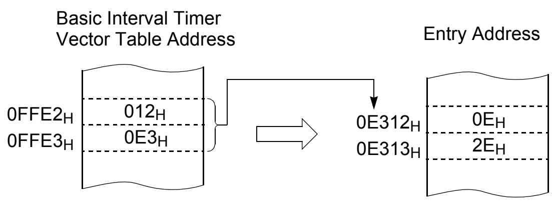



**Figure 17-3 Timing chart of Interrupt Acceptance and Interrupt Return Instruction**




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

An 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. If necessary, these registers should be saved by the software. 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.

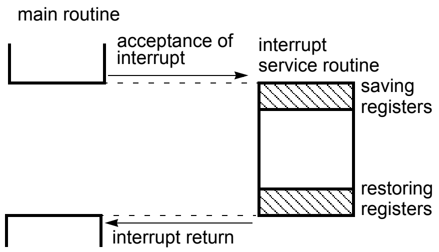
Example: Register saving

```
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 registers are saved or restored by using push and pop instructions.




## 17.2 BRK Interrupt

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

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

Each processing step is determined by B-flag as shown in Figure 17-4.

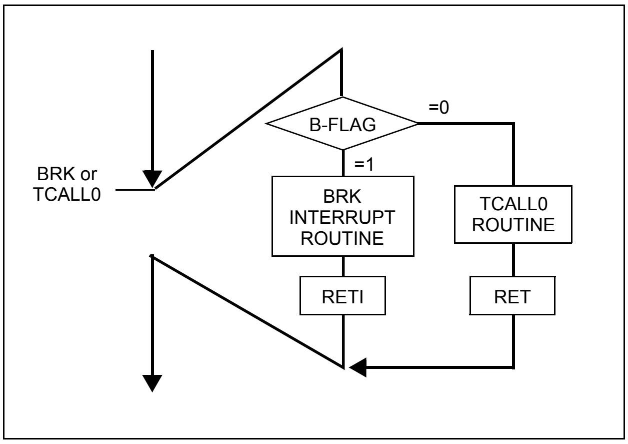



**Figure 17-4 Execution of BRK/TCALL0**

## 17.3 Multi Interrupt

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

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

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

```
TIMER1: PUSH   A
        PUSH   X
        PUSH   Y
```

```
LDM    IENH,#40H    ;Enable INT0 only
LDM    IENM,#0      ;Disable other
LDM    IENL,#0      ;Disable other
EI                  ;Enable Interrupt
:
:
:


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

.

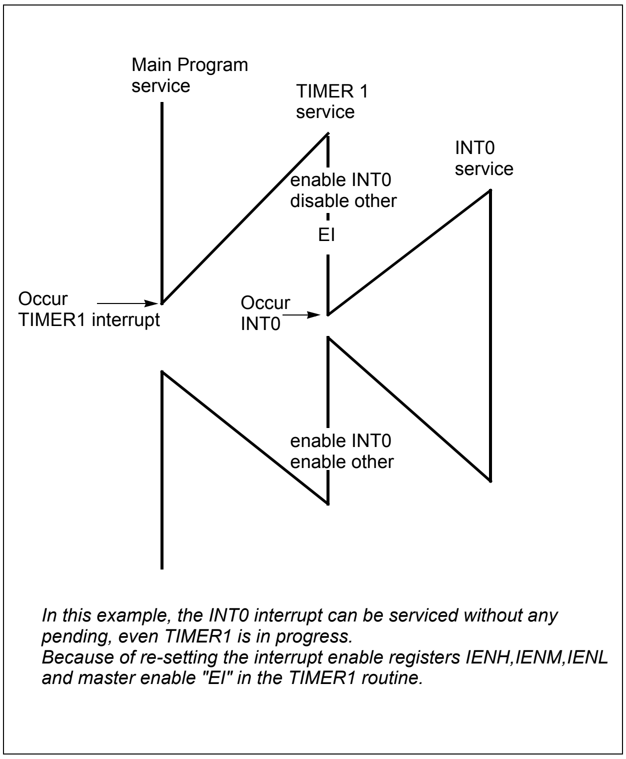



**Figure 17-5 Execution of Multi Interrupt**

## 17.4 External Interrupt

The external interrupt on INT0, INT1, INT2 and INT3 pins are edge triggered depending on the edge selection register IEDS (address $0FC_H$) as shown in Figure 17-6.

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

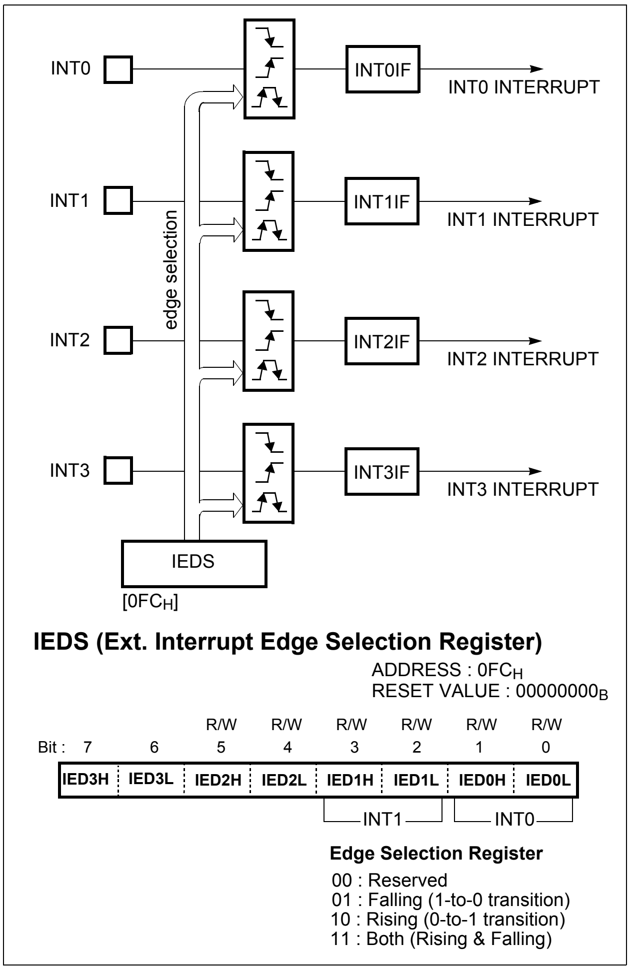



**Figure 17-6 External Interrupt Block Diagram**

Example: To use as an INT0

```
        :
        :
;**** Set port as an input port R0
        LDM    R0IO,#1011_1111B
        ;
;**** Set port as an interrupt port
        LDM    PSR0,#0100_0000B
        ;
;**** Set Falling-edge Detection
        LDM    IEDS,#0000_0001B
        :
        :
        :
```

### Response Time

The INT0, INT1,INT2 and INT3 edge are latched into INT0F, INT1F, INT2F and INT3F 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 maximum 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.

Interrupt response timings are shown in Figure 17-7.

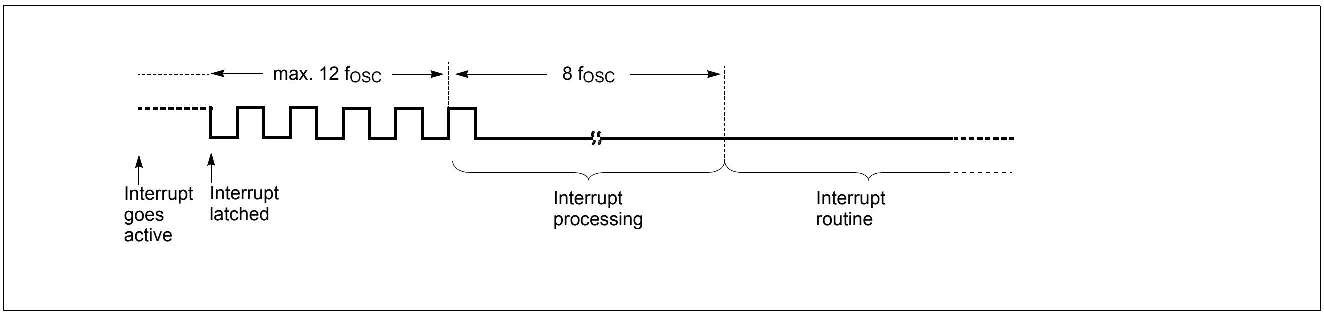



**Figure 17-7 Interrupt Response Timing Diagram**

# 18. LCD DRIVER

The MC80F7708A has the circuit that directly drives the liquid crystal display (LCD) and its control circuit. The segment/common driver directly drives the LCD panel, and the LCD controller generates the segment/common signals according to the RAM which stores display data. VCL3 ~ VCL0 voltage are made by the internal bias resistor circuit.

The MC80F7708A has the segement output port 20 pins (SEG0 ~ SEG19) and Common output port 4 pins (COM0 ~ COM3).

The Figure 18-1 shows the configuration of the LCD driver.

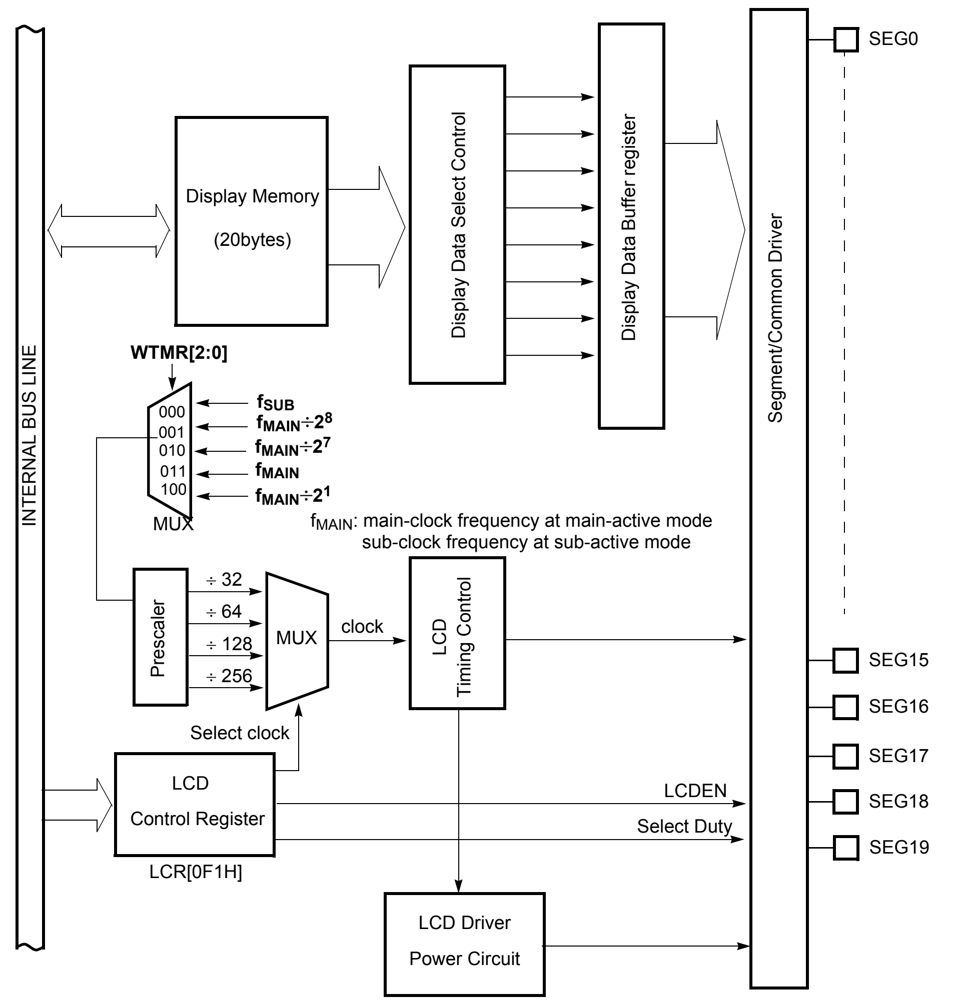



**Figure 18-1 LCD Driver Block Diagram**

## 18.1 Control of LCD Driver Circuit

The LCD driver is controlled by the LCD Control Register (LCR). The LCR[1:0] determines the frequency of COM signal scanning of each segment output. RESET clears the LCD control register LCR values to logic zero. The LCD SEG or COM ports are selected by setting corresponding bits of R5PSR, R6PSR or R7PSR to "0".

The LCD display can continue to operate during SLEEP and STOP modes if sub-frequency clock is used as LCD clock source.

**LCR(LCD Control Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | SCKD | 1 | LCDEN | 0 | 1 | 1 | LCK1 | LCK0 |

ADDRESS : $0B2_H$
RESET VALUE : 000-0100B

**SCKD (Sub Clock Disable)**
0: Sub Clock Oscillation (SXIN, SXOUT)
1: Sub Clock Disable (R45, R46)

**LCDEN (LCD Display Enable Bit)**
0: LCD Display Disable
1: LCD Display Enable

*** Unused bit of LCR should be set as**
**Bit6 : "1"**
**Bit4 : "0"**
**Bit3 : "1"**
**Bit2 : "1"**

**LCK<1:0> (LCD Clock source selection)**
00: $f_S \div 32$ (Frame Frequency 1024Hz When fs is 32.768kHz)
01: $f_S \div 64$ (Frame Frequency 512Hz When fs is 32.768kHz)
10: $f_S \div 128$ (Frame Frequency 256Hz When fs is 32.768kHz)
11: $f_S \div 256$ (Frame Frequency 128Hz When fs is 32.768kHz)

* $f_S$ : $f_{SUB}$ (Sub clock) or $f_{MAIN} \div 2^7$ or $f_{MAIN} \div 2^8$ or $f_{MAIN} \div 2$ or $f_{MAIN}$
(It can be selected by setting WTCK[2:0] of WTMR register.)

$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**R5 / LCD Port Selection Register**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| **R5PSR** | R5PSR7 | R5PSR6 | R5PSR5 | R5PSR4 | R5PSR3 | R5PSR2 | R5PSR1 | R5PSR0 |

ADDRESS : $0AC_H$
RESET VALUE : 1111_1111B
(Seg7 ~ Seg0)

**R6 / LCD Port Selection Register**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| **R6PSR** | R6PSR7 | R6PSR6 | R6PSR5 | R6PSR4 | R6PSR3 | R6PSR2 | R6PSR1 | R6PSR0 |

ADDRESS : $0AD_H$
RESET VALUE : 1111_1111B
(Seg15 ~ Seg8)

**R7 / LCD Port Selection Register**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| **R7PSR** | R7PSR7 | R7PSR6 | R7PSR5 | R7PSR4 | R7PSR3 | R7PSR2 | R7PSR1 | R7PSR0 |

ADDRESS : $0AE_H$
RESET VALUE : 1111_1111B
(COM0~COM3, Seg19 ~ Seg16)

**R5PSR~R7PSR**
0 : Seg or COM Selection
1 : Port Selection

**WTMR (Watch Timer Mode Register)**

| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | WTEN | LOADEN | - | WTIN1 | WTIN0 | WTCK2 | WTCK1 | WTCK0 |

ADDRESS: $0EA_H$
INITIAL VALUE:00--_$0000_B$

**WTEN (Watch Timer Enable Bit)**
0: Watch Timer Disable
1: Watch Timer Enable

**LOADEN (7bit reload Counter Write Enable Bit)**
0: Watch Dog Timer Write Enable
1: Watch Timer Write Enable

**WTIN[1:0] (Watch Timer Interrupt Interval Selection)**
00: $f_{wck} / 2^{11}$ [16Hz]*
01: $f_{wck} / 2^{13}$ [4Hz]*
10: $f_{wck} / 2^{14}$ [2Hz]*
11: $f_{wck} / 2^{14}$ x (7bit WT value+1) [2Hz x (7bit WT value+1)]*

**WTCK[2:0] (Watch Timer and LCD Clock Source Selection) : $f_{wck}$**
000: Sub. Clock ($f_{SUB}$)
001: Main Clock ($f_{MAIN} \div 2^8$)
010: Main Clock ($f_{MAIN} \div 2^7$)
011: Main Clock ($f_{MAIN}$)
111: Main Clock ($f_{MAIN} \div 2$)
$f_{MAIN}$: main-clock frequency at main-active mode
sub-clock frequency at sub-active mode

**Figure 18-2 LCD Control Register**

***Note:*** *If the SCKD is set to "1", the SXIN and SXOUT pin is used as normal I/O pin R45, R46.*

***Note:*** *When the Sub clock is used as internal bias source clock, stabilization time is needed. Normally, the stabilization time is need more than 500ms.*

***Note:*** *When selecting Sub clock as the LCD clock source, the WTCK[2:0] bit of WTMR(Watch Timer Mode Register) should be set to "000" as well as SCKD bit of LCR be set to "0".*

***Note:*** *Bit 6, Bit 4, Bit 3 Bit 2 of LCR should be set to "1", "0", "1" , "1" respectively.*

### Selecting Frame Frequency

Frame frequency is set to the base frequency as shown in the following Table 18-1. The $f_S$ is selected to $f_{SUB}$ (sub clock) which is 32.768kHz.

| LCR[1:0] | LCD clock | Frame Frequency (Hz) | |
|---|---|---|---|
| | | Duty = 1/4 | Duty = 1/8(EVA only) |
| 00 | $f_{SUB} \div 32$ | 128 | 64 |
| 01 | $f_{SUB} \div 64$ | 64 | 32 |
| 10 | $f_{SUB} \div 128$ | 32 | 16 |
| 11 | $f_{SUB} \div 256$ | 16 | 8 |

**Table 18-1 Setting of LCD Frame Frequency**

### The matters to be attended to use LCD driver

In reset state, LCD source clock is sub clock. So, when the power is supplied, the LCD display would be flickered before the oscillation of sub clock is stabilized. It is recommended to use LCD display on after the stabilization time of sub clock is considered enough.

## 18.2 LCD BIAS Control

The MC80F7708A has internal Bias Circuit for driving LCD panel. It alse has the contrast controller of 16 step.

The LCD Bias control register and internal Bias circuit is as shown in the Figure 18-3.

The SYS_SVLD[1:0] and SVLDF of LBCR register is used for controlling SVLD.

***Note:*** *The self bias check reference can be applied to contrast adjustment with VDD voltage variation. Because the VDD voltage can be calculated by reading the ADC value of self bias check reference. Writing appropriate value to CTR[3:0] with VDD level, LCD contrast variation with VDD can be reduced.*

**LBCR(LCD Bias Control Register)**

| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|
| Bit : | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CTR_S | CTR_DS3 | CTR_DS2 | CTR_DS1 | CTR_DS0 | SYS_SVLD1 | SYS_SVLD0 | **SVLDF** |

ADDRESS : $0B3_H$
RESET VALUE : 01111000B

**CTR_S (Voltage Source selection)**
0: Direct Voltage
1: Contrast Controller voltage

**SYS_SVLD<1:0> (Mode selection of SVLD Result)**
00: Reset mode
01: No operation mode(only flag setting)
10: Freeze mode
11: No operation mode(only flag setting)

**SVLDF (SVLD Flag)**
0: **SVLD** No Detect
1: **SVLD** Detect

**CTR_DS<3:0> (Contrast Controller Level Selection)**
0000: VCL2 = VDD / 2
0001: VCL2 = VDD * ( 16 / 30 )
0010: VCL2 = VDD * ( 17 / 30 )
0011: VCL2 = VDD * ( 18 / 30 )
0100: VCL2 = VDD * ( 19 / 30 )
0101: VCL2 = VDD * ( 20 / 30 )
0110: VCL2 = VDD * ( 21 / 30 )
0111: VCL2 = VDD * ( 22 / 30 )
1000: VCL2 = VDD * ( 23 / 30 )
1001: VCL2 = VDD * ( 24 / 30 )
1010: VCL2 = VDD * ( 25 / 30 )
1011: VCL2 = VDD * ( 26 / 30 )
1100: VCL2 = VDD * ( 27 / 30 )
1101: VCL2 = VDD * ( 28 / 30 )
1110: VCL2 = VDD * ( 29 / 30 )
1111: VCL2 = VDD

**Block Diagram of LCD BIAS**

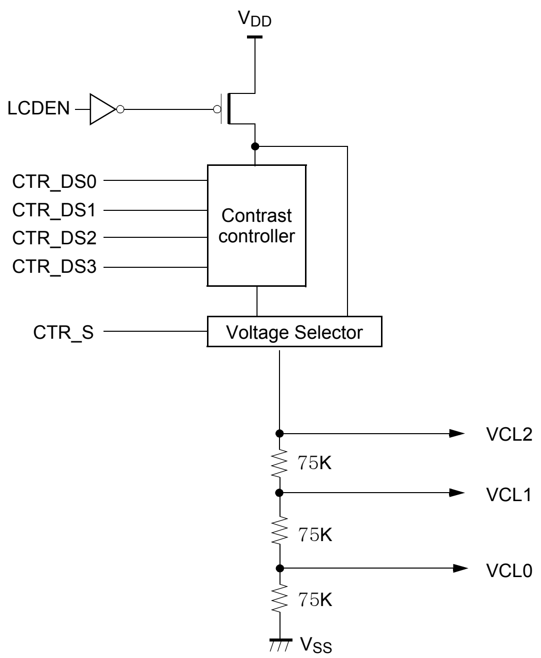



**Figure 18-3 LCD Bias Control**

## 18.3 LCD Display Memory

Display data are stored to the display data area (page 4) in the data memory.
The display datas which stored to the display data area (address $0460_H$-$0473_H$) are read automatically and sent to the LCD driver by the hardware. The LCD driver generates the segment signals and common signals in accordance with the display data and drive method. Therefore, display patterns can be changed by only overwriting the contents of the display data area with a program. The table look up instruction is mainly used for this overwriting.
Figure 18-4 shows the correspondence between the display data area and the SEG/COM pins. The LCD lights when the display data is “1” and turn off when “0”.

The SEG data for display is controlled by RPR (RAM Paging Register).

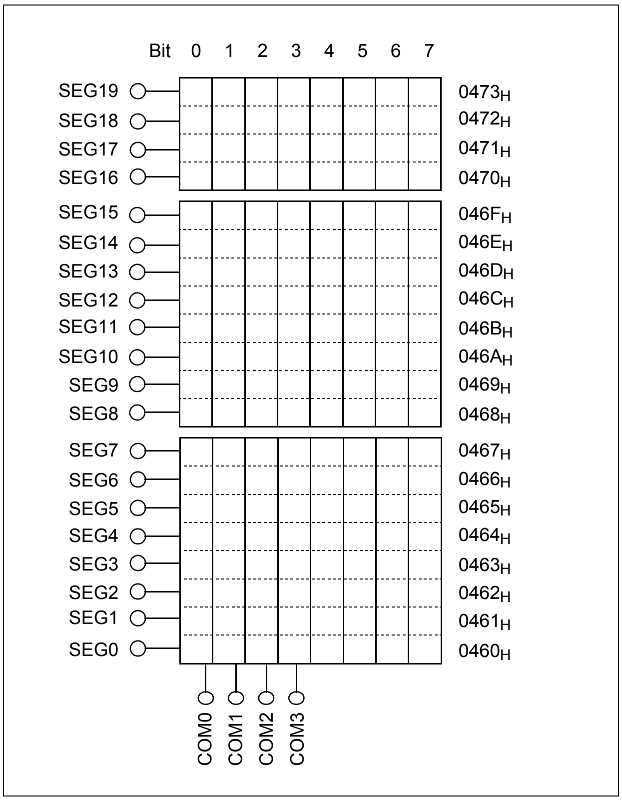



**Figure 18-4 LCD Display Memory**

## 18.4 Control Method of LCD Driver

### Initial Setting

Flow chart of initial setting is shown in Figure 18-5.

Example: Driving of LCD

```
Select Frame Frequency      LDM    LCR,#4DH    ;fF=64Hz, 1/4 duty(fSUB= 32.768kHz)
                            :
                            LDM    RPR,#4;Select LCD Memory(4 page)
                            SETG
Clear                       LDX    #60H
LCD Display      C_LCD1:    LDA    #0 ;RAM Clear
Memory                                ;(0460H->0473H)
                            STA    {X}+
                            CMPX   #074H
                            BNE    C_LCD1
                            CLRG
                            :
Turn on LCD                 SET1   LCR.5;Enable display
                            :
```

.

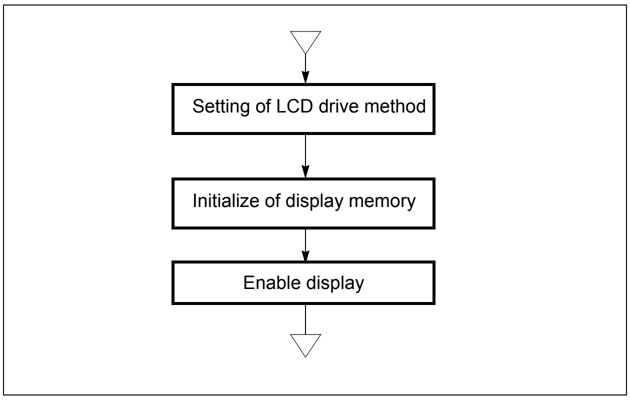



**Figure 18-5 Initial Setting of LCD Driver**

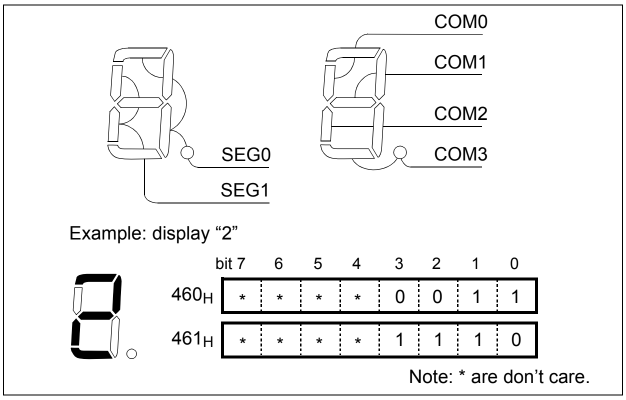



**Figure 18-6 Example of Connection COM & SEG**

### Display Data

Normally, display data are kept permanently in the program memory and then stored at the display data area by the table look-up instruction. This can be explained using character display with 1/4 duty LCD as an example as well as any LCD panel. The COM and SEG connections to the LCD and display data are the same as those shown is Figure 18-6. Following is showing the programming example for displaying character.

***Note:*** *When power on RESET, sub oscillation start up time is required. Enable LCD display after sub oscillation is stabilized, or LCD may occur flicker at power on time shortly.*

Write into the LCD Memory

```
        :
        CLRG
        LDX#<DISPRAM;Address included the data
            ;to be displayed.
GOLCD:  LDA{X}
        TAY
        LDA!FONT+Y ;LOAD FONT DATA
        LDMRPR,#4;Set RPR = 4 to access LCD
        SETG   ;Set Page 4
        LDX#60H
        STA{X}+;LOWER 4 BITS OF ACC. seg0
        XCN
        STA{X} ;UPPER 4 BITS OF ACC. seg1
        CLRG   ;Set Page = 0
        :
```

Font data

```
FONT    DB 1101_0111B;    "0"
        DB 0000_0110B;    "1"
        DB 1110_0011B;    "2"
        DB 1010_0111B;    "3"
        DB 0011_0110B;    "4"
        DB 1011_0101B;    "5"
        DB 1111_0101B;    "6"
        DB 0000_0111B;    "7"
        DB 1111_0111B;    "8"
        DB 0011_0111B;    "9"
```

### LCD Waveform

The LCD duty(1/4) can be selected by LCR register. ure 18-7.
The example of 1/4 duty, 1/3 bias are shown in shown Fig-

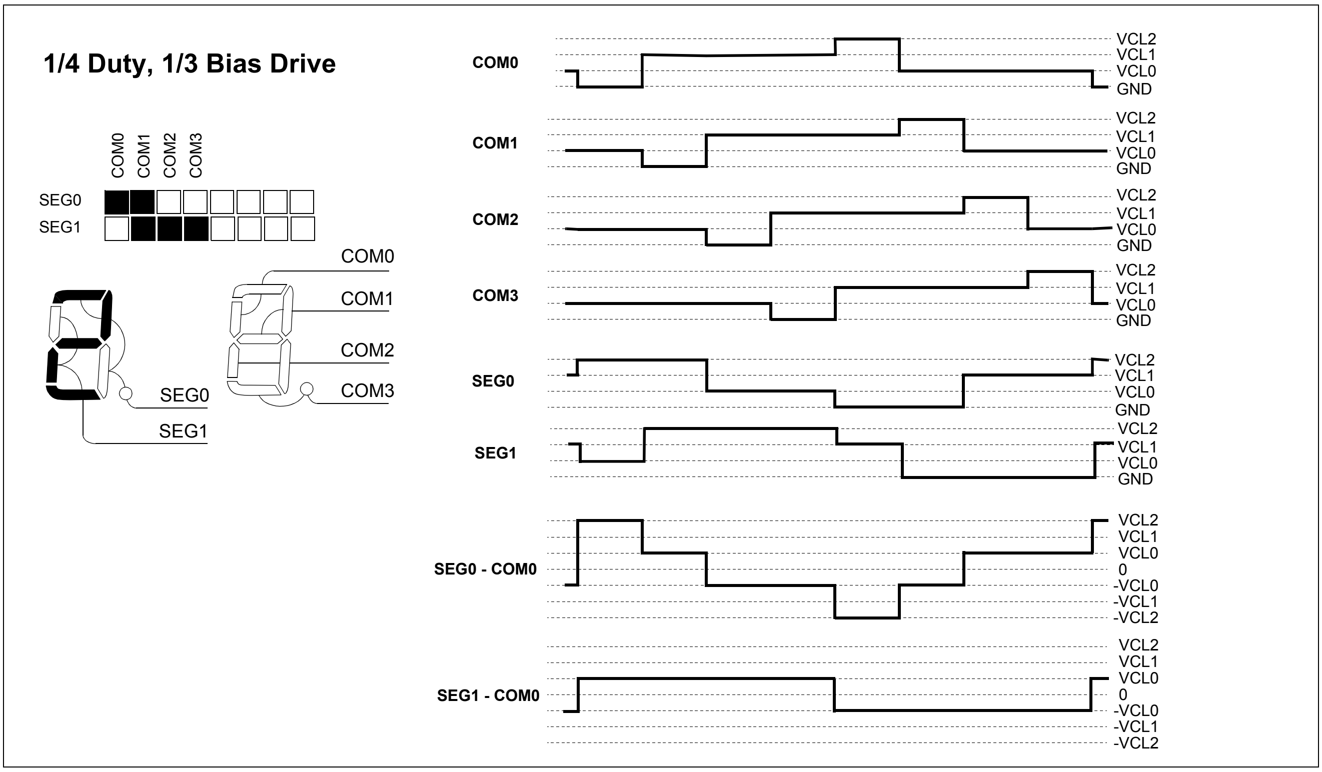
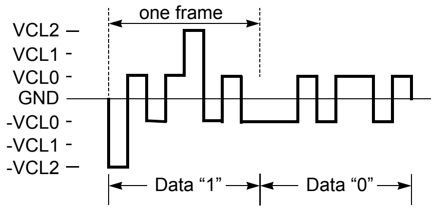



**Figure 18-7 Example of LCD drive output**

## 18.5 Duty and Bias Selection of LCD Driver

4 kinds of driving methods can be selected by LCDD[1:0] (bits 3 and 2 of LCD control register) and connection of BIAS pin externally. Figure 18-8 shows typical driving waveforms for LCD.).




**(a) 1/4 Duty, 1/3 Bias**

**Figure 18-8 LCD Drive Waveform (Voltage COM-SEG Pins)**

# 19. OPERATION MODE

The system clock controller starts or stops the main frequency clock oscillator, which is controlled by system clock mode register (SCMR). Figure 19-1 shows the operating mode transition diagram.

System clock control is performed by the system clock mode register (SCMR). During reset, this register is initialized to “0” so that the main-clock operating mode is selected.

## Main Active mode

This mode is fast-frequency operating mode. The CPU and the peripheral hardwares are operated on the high-frequency clock. At reset release, this mode is invoked.

## SLEEP mode

In this mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

## STOP mode

In this mode, the system operations are all stopped, holding the internal states valid immediately before the stop at the low power consumption level.

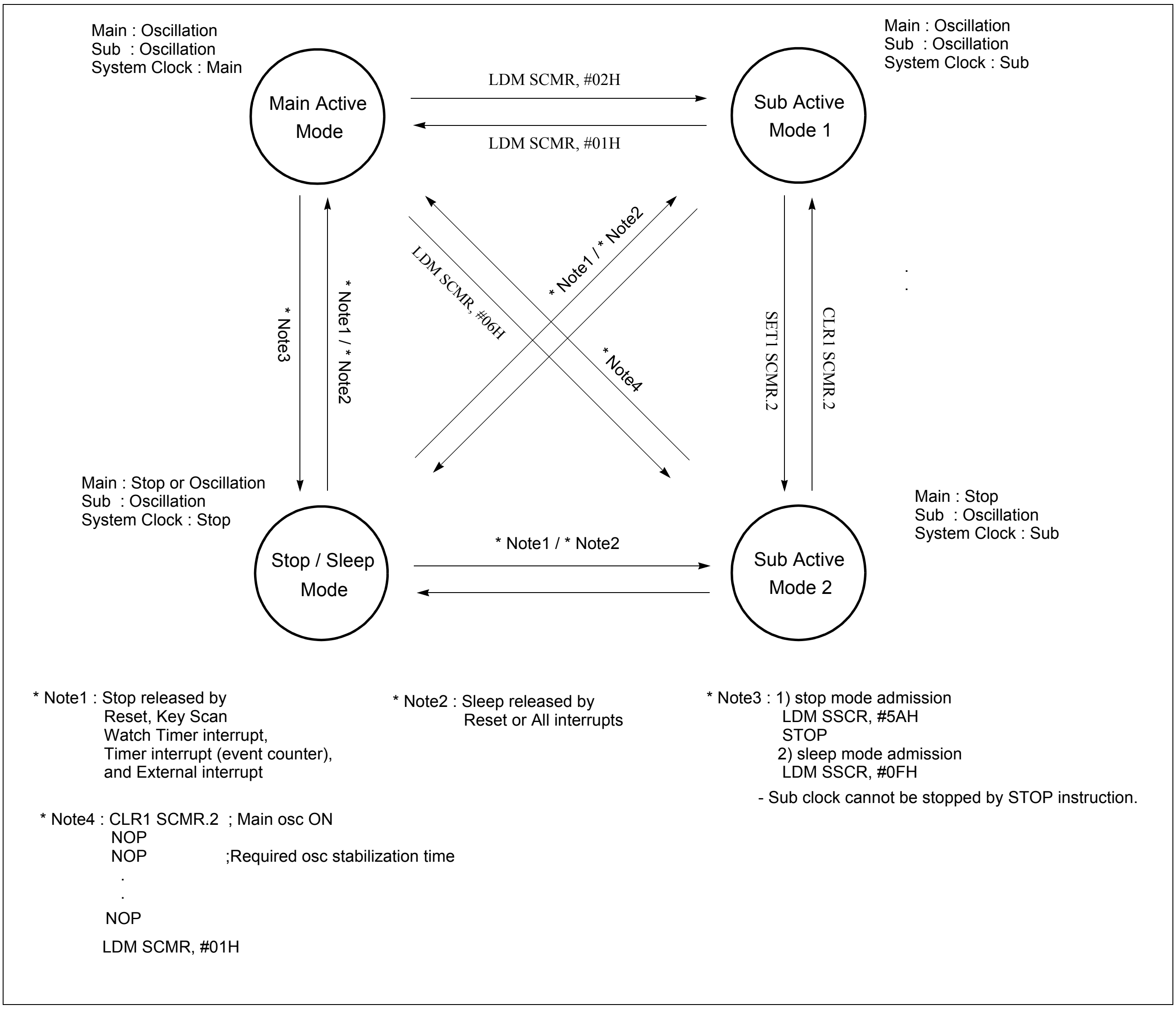



**Figure 19-1 Operating Mode**

## 19.1 Operation Mode Switching

### Shifting from the Normal operation to the SLEEP mode

By writing "$0F_H$" into SSCR which will be explained in "20.1 SLEEP Mode" on page 88, the CPU clock stops and the SLEEP mode is invoked. The CPU stops while other peripherals are operate normally.

The way of release from this mode is RESET and all available interrupts.

For more detail, See "20.1 SLEEP Mode" on page 88

### Shifting from the Normal operation to the STOP mode

By writing "$5A_H$" into SSCR and then executing STOP instruction, the main-frequency clock oscillation stops and the STOP mode is invoked. But sub-frequency clock oscillation is operated continuously.

After the STOP operation is released by reset, the operation mode is changed to Main active mode.

The methods of release are RESET, Key scan interrupt, Watch Timer interrupt, Timer/Event counter1 (EC0 pin) and External Interrupt.

For more details, see "20.2 STOP Mode" on page 89.

***Note:*** *In the STOP and SLEEP operating modes, the power consumption by the oscillator and the internal hardware is reduced. However, the power for the pin interface (depending on external circuitry and program) is not directly associated with the low-power consumption operation. This must be considered in system design as well as interface circuit design.*

# 20. POWER DOWN OPERATION

MC80F7708A have 2 power down mode. In power down mode, power consumption is reduced considerably in Battery operation that Battery life can be extended a lot.

Sleep mode is entered by writing “$0F_H$” into Stop and Sleep Control Register(SSCR), and STOP mode is entered by writing “$5A_H$” into SSCR and then executing STOP instruction.

## 20.1 SLEEP Mode

In this mode, the internal oscillation circuits remain active.

Oscillation continues and peripherals are operate normally but CPU stops. The status of all Peripherals in this mode is shown in Table 20-1. Sleep mode is entered by writing “$0F_H$” into SSCR (address $0E9_H$).

It is released by RESET or all interrupt. To be released by interrupt, interrupt should be enabled before Sleep mode.

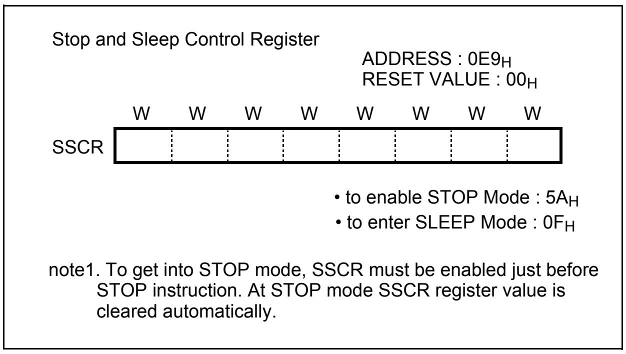



**Figure 20-1 SLEEP Mode Register**

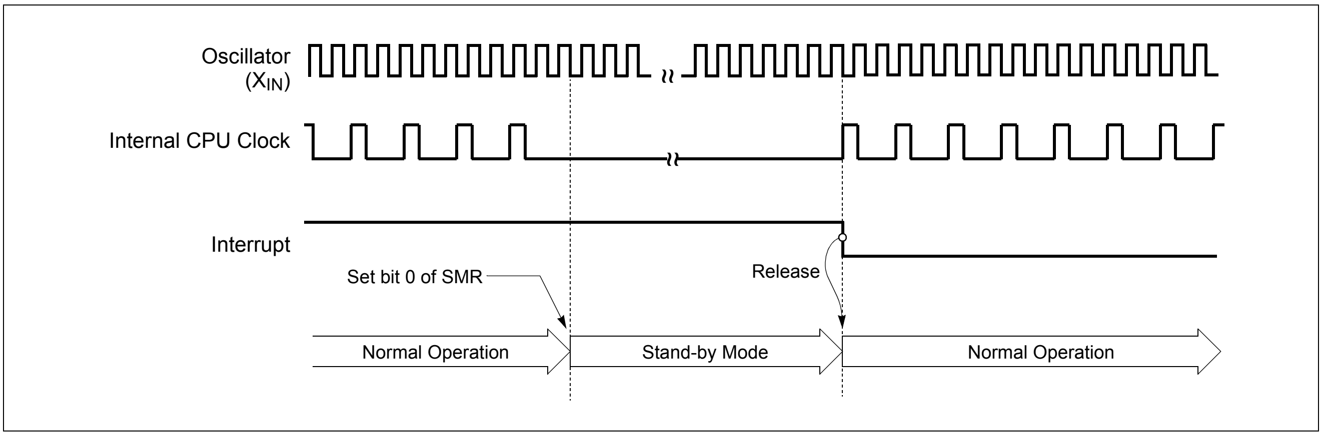



**Figure 20-2 Sleep Mode Release Timing by External Interrupt**

.

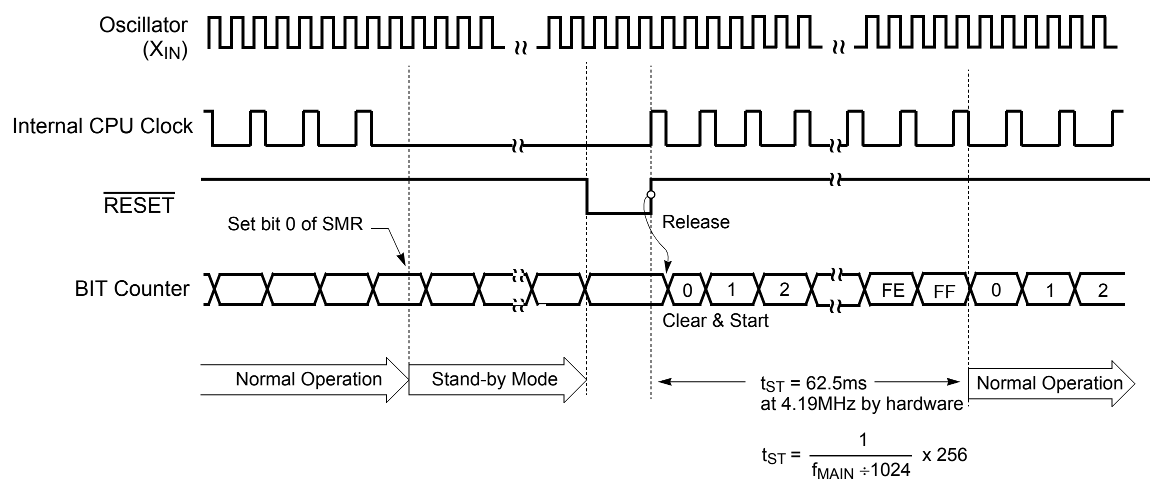



**Figure 20-3 SLEEP Mode Release Timing by RESET pin**

## 20.2 STOP Mode

For applications where power consumption is a critical factor, this device provides STOP mode for reducing power consumption.

### Start The Stop Operation

The STOP mode can be entered by STOP instruction during program execution. In Stop mode, the on-chip main-frequency oscillator, system clock, and peripheral clock are stopped (Watch timer clock is oscillating continuously:. With the clock frozen, all functions are stopped, but the on-chip RAM and Control registers are held. The port pins output the values held by their respective port data register and the port direction registers. The status of peripherals during Stop mode is shown below.

| Peripheral | STOP Mode | Sleep Mode |
|---|---|---|
| CPU | All CPU operations are disabled | All CPU operations are disabled |
| RAM | Retain | Retain |
| LCD driver | Operates continuously | Operates continuously |
| Basic Interval Timer | Halted | Operates continuously |
| Timer/Event counter 0 | Halted (Only when the Event counter mode is enabled, Timer 0 operates normally) | Timer/Event counter 0 operates continuously |
| Watch Timer | Operates continuously | Operates continuously |
| Main-oscillation | Stop ($X_{IN}$=L, $X_{OUT}$=L) | Oscillation[1] |
| Sub-oscillation | Oscillation | Oscillation |
| I/O ports | Retain | Retain |
| Control Registers | Retain | Retain |
| Release method | by RESET, Watch Timer interrupt, Timer interrupt (EC0), and External interrupt | by RESET, All interrupts |

**Table 20-1 Peripheral Operation during Power Down Mode**

1. Refer to the Table 10-2

| Operating Clock source | Main Operating Mode | Main Sleep Mode | Sub Active Operating Mode | Sub Sleep Operating Mode | Stop Mode 1 | Stop Mode 2 |
|---|---|---|---|---|---|---|
| Main Clock | Oscillation | Oscillation | SCMR[2]<br>0 ==>Oscillation<br>1 ==>Stop | SCMR[2]<br>0 ==>Oscillation<br>1 ==>Stop | Stop | Stop |
| Sub Clock | Oscillation | Oscillation | Oscillation | Oscillation | Oscillation | Stop |
| System Clock | Active | Stop | Active | Stop | Stop | Stop |
| Peri. Clock | Active | Active | Active | Active | Stop[1] | Stop |

1. Except watch timer(sub clock) and LCD driver(sub clock)

**Table 20-2 Clock Operation of STOP and SLEEP mode**

***Note:*** *Since the $X_{IN}$ pin is connected internally to GND to avoid current leakage due to the crystal oscillator in STOP mode, do not use STOP instruction when an external clock is used as the main system clock.*

In the Stop mode of operation, $V_{DD}$ can be reduced to minimize power consumption. Be careful, however, that $V_{DD}$ is not reduced before the Stop mode is invoked, and that $V_{DD}$ is restored to its normal operating level before the Stop mode is terminated.

The reset should not be activated before $V_{DD}$ is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize.
And after STOP instruction, at least two or more NOP instruction should be written as shown in example below.

Example)

```
:
LDM    CKCTLR,#0000_1111B
STOP
NOP
NOP
:
```

The Interval Timer Register CKCTLR should be initialized by software in order that oscillation stabilization time should be longer than 20ms before STOP mode.

### Release the STOP mode

The exit from STOP mode is using hardware reset or external interrupt, watch timer ortimer interrupt (EC0).

To release STOP mode, corresponding interrupt should be enabled before STOP mode.
Specially as a clock source of Timer/Event counter, EC0 pin can release it by Timer/Event counter Interrupt request.

Reset redefines all the control registers but does not change the on-chip RAM. External interrupts allow both on-chip RAM and Control registers to retain their values.

Start-up is performed to acquire the time for stabilizing oscillation. During the start-up, the internal operations are all stopped.

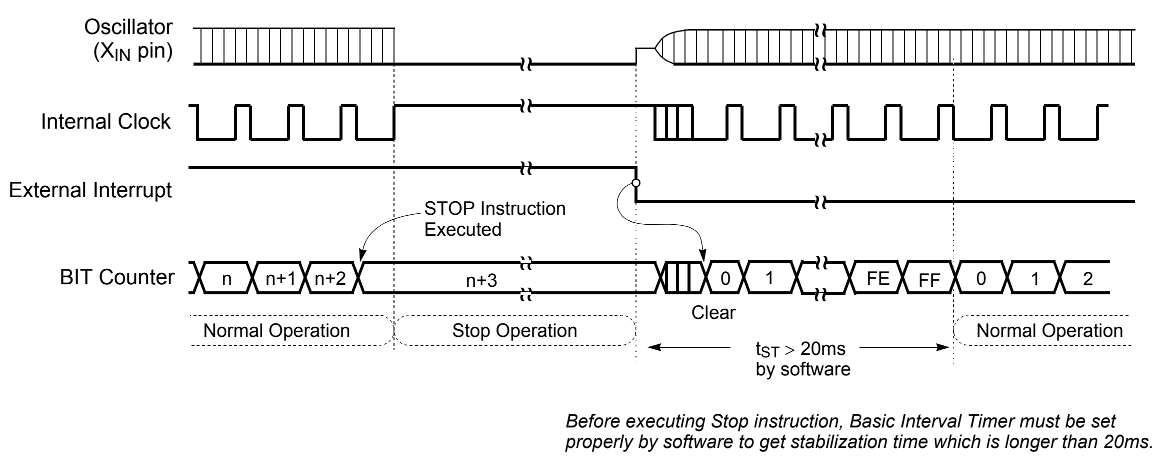



**Figure 20-4 STOP Mode Release Timing by External Interrupt**

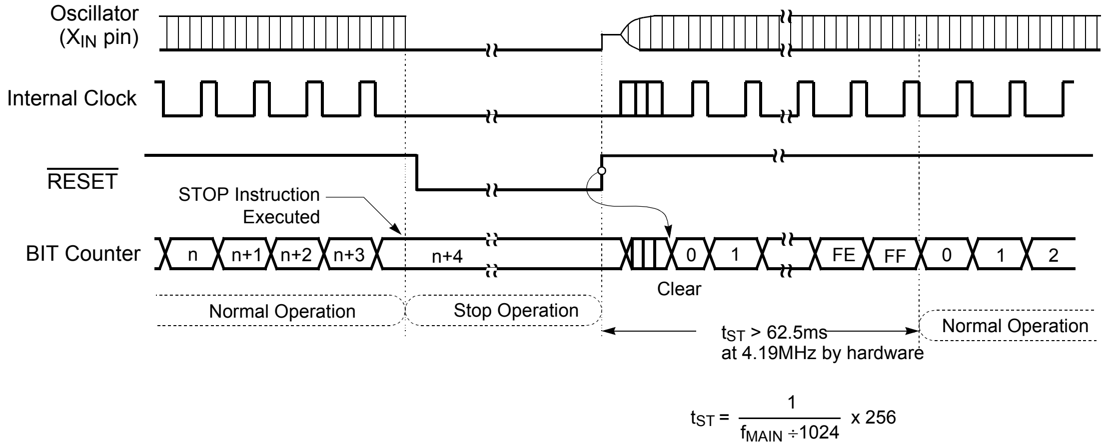



**Figure 20-5 STOP Mode Release Timing by RESET**

## Minimizing Current Consumption

The Stop mode is designed to reduce power consumption. To minimize current drawn during Stop mode, the user should turn-off output drivers that are sourcing or sinking current, if it is practical.

***Note:*** *In the STOP operation, the power dissipation associated with the oscillator and the internal hardware is lowered; however, the power dissipation associated with the pin interface (depending on the external circuitry and program) is not directly determined by the hardware operation of the STOP feature. This point should be little current flows when the input level is stable at the power voltage level ($V_{DD}$/$V_{SS}$); however, when the input level becomes higher than the power voltage level (by approximately 0.3V), a current begins to flow. Therefore, if cutting off the output transistor at an I/O port puts the pin signal into the high-impedance state, a current flow across the ports input transistor, requiring it to fix the level by pull-up or other means.*

It should be set properly that current flow through port doesn't exist.

First consider the setting to input mode. Be sure that there is no current flow after considering its relationship with external circuit. In input mode, the pin impedance viewing from external MCU is very high that the current doesn't flow.

But input voltage level should be $V_{SS}$ or $V_{DD}$. Be careful that if unspecified voltage, i.e. if uniformed voltage level (not $V_{SS}$or $V_{DD}$) is applied to input pin, there can be little current (max. 1mA at around 2V) flow.

If it is not appropriate to set as an input mode, then set to output mode considering there is no current flow. Setting to High or Low is decided considering its relationship with external circuit. For example, if there is external pull-up resistor then it is set to output mode, i.e. to High, and if there is external pull-down resistor, it is set to low.

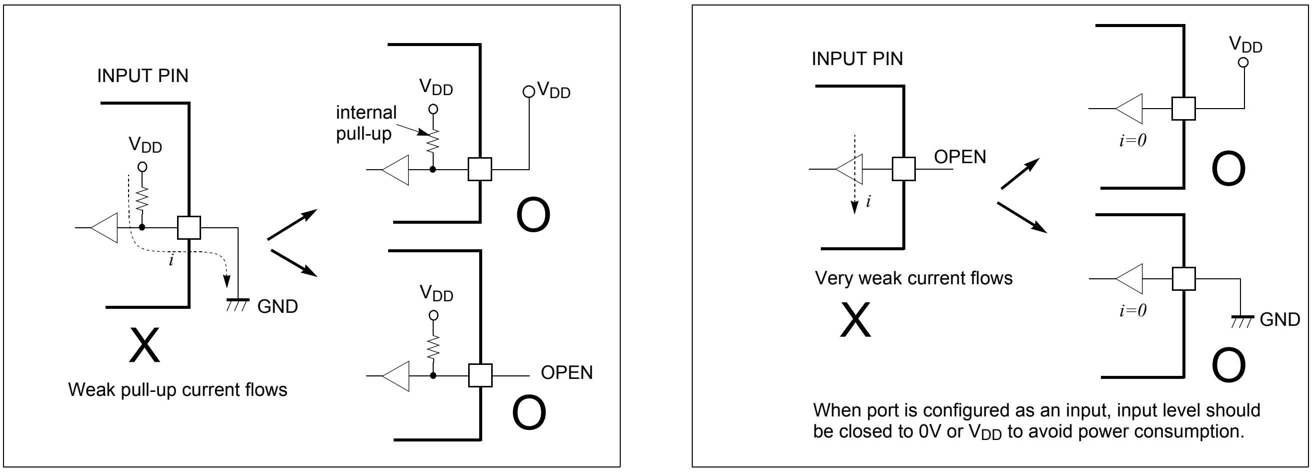



**Figure 20-6 Application Example of Unused Input Port**

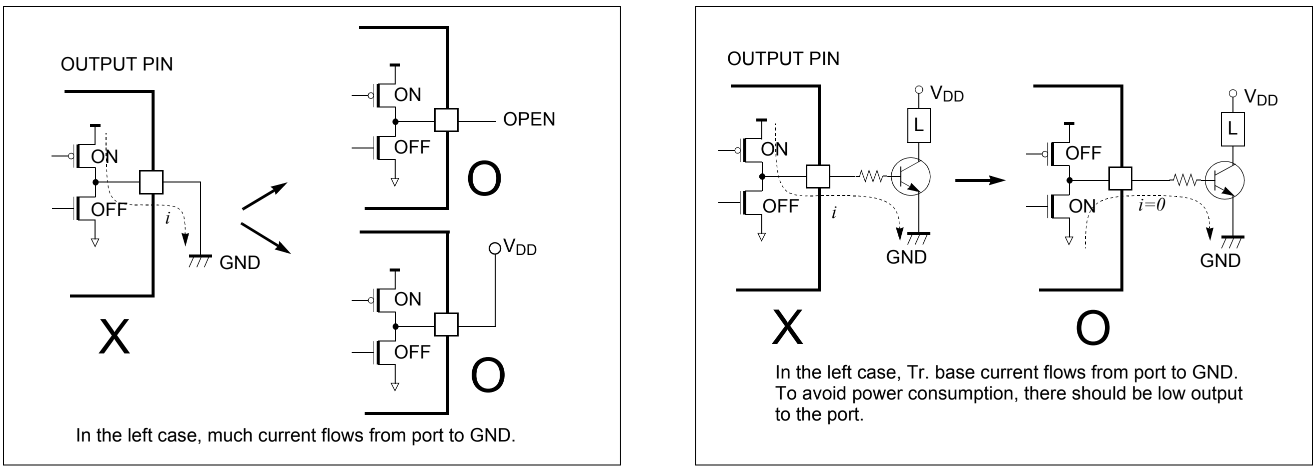



**Figure 20-7 Application Example of Unused Output Port**

## 21. OSCILLATOR CIRCUIT

The MC80F7708A have three oscillation circuits internally. $X_{IN}$ and $X_{OUT}$ are input and output for main frequency and $SX_{IN}$ and $SX_{OUT}$ are input and output for sub frequency, respectively, inverting amplifier which can be configured for being used as an on-chip oscillator, as shown in Figure 21-1.

Example

| | |
|---|---|
| Crystal Oscillator | C1,C2 = 10~30pF |
| Ceramic Resonator | C1,C2 = 10~30pF |

Cautions 1. The example load capacitor value(C1, C2, C3, C4) is common value but may not be appropriate for some crystal or ceramic resonator.

Cautions 2. The subsystem clock oscillator is designed as a low-amplitude circuit for reducing current consumption, and is more prone to malfunction due to noise than the main system clock oscillator. Particular care is therefore required with the wiring method when the subsystem clock is used.

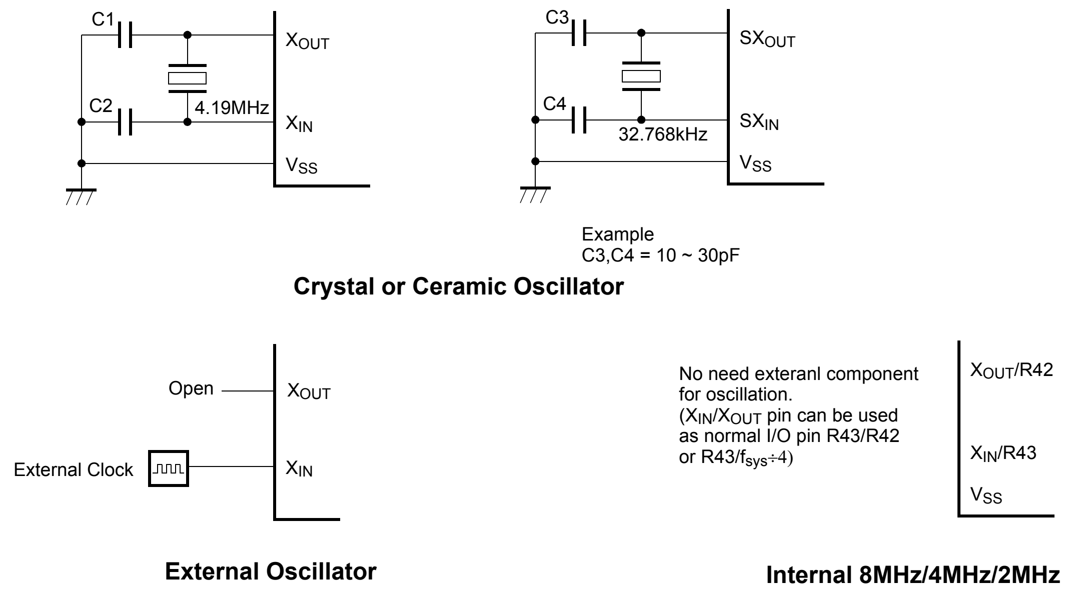



**Figure 21-1 Oscillation Circuit**

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 21-2 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* $V_{SS}$*. Do not ground it to any ground pattern where high current is present. Do not fetch signals from the oscillator.*

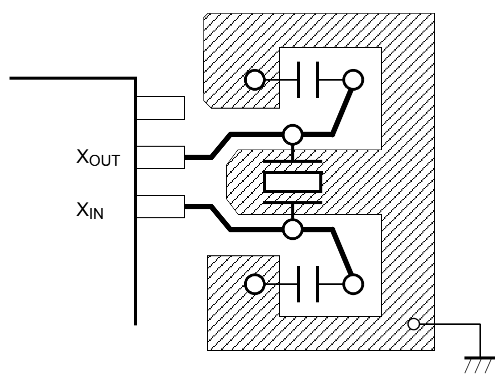



**Figure 21-2 Layout of Oscillator PCB circuit**

## 22. RESET

The MC80F7708A have has four reset generation sources; external reset input, power on reset (POR), Supply Votage level Detector Rest (SVLDR) and watch-dog timer reset.

Table 22-1 shows on-chip hardware initialization by reset action.

| On-chip Hardware | Initial Value | On-chip Hardware | Initial Value |
|---|---|---|---|
| Program counter (PC) | ($FFFF_H$) - ($FFFE_H$) | Operation mode | Main-frequency clock |
| RAM page register (RPR) | 0 | Peripheral clock | On |
| G-flag (G) | 0 | Control registers | Refer to Table 8-1 on page 29 |

**Table 22-1 Initializing Internal Status by Reset Action**

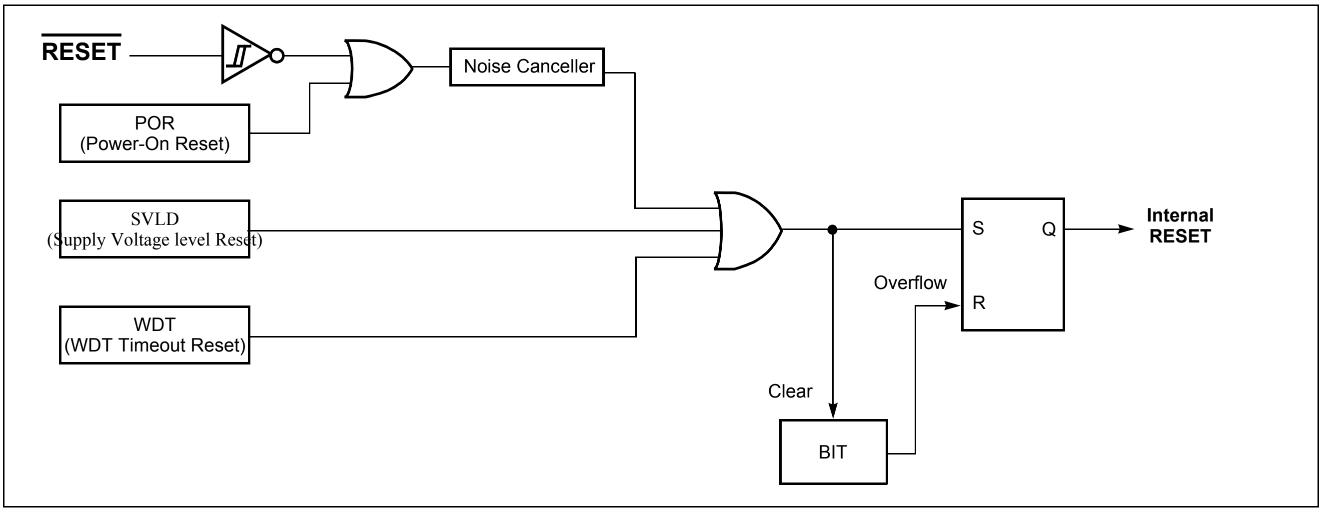



**Figure 22-1 RESET Block Diagram**

### 22.1 External Reset Input

The reset input is the $\overline{RESET}$ pin, which is the input to a Schmitt Trigger. A reset accomplished by holding the $\overline{RESET}$ pin to low for at least 8 oscillator periods, within the operating voltage range and oscillation stable, it is applied, and the internal state is initialized. After reset, 65.5ms (at 4MHz) and 7 oscillator periods are required to start execution as shown in Figure 22-3.

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

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

A connection for normal power-on-reset is shown in Figure 22-2.

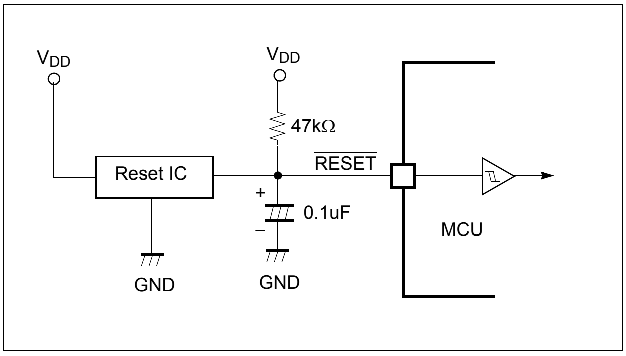



**Figure 22-2 Normal Power-on-Reset Circuit**

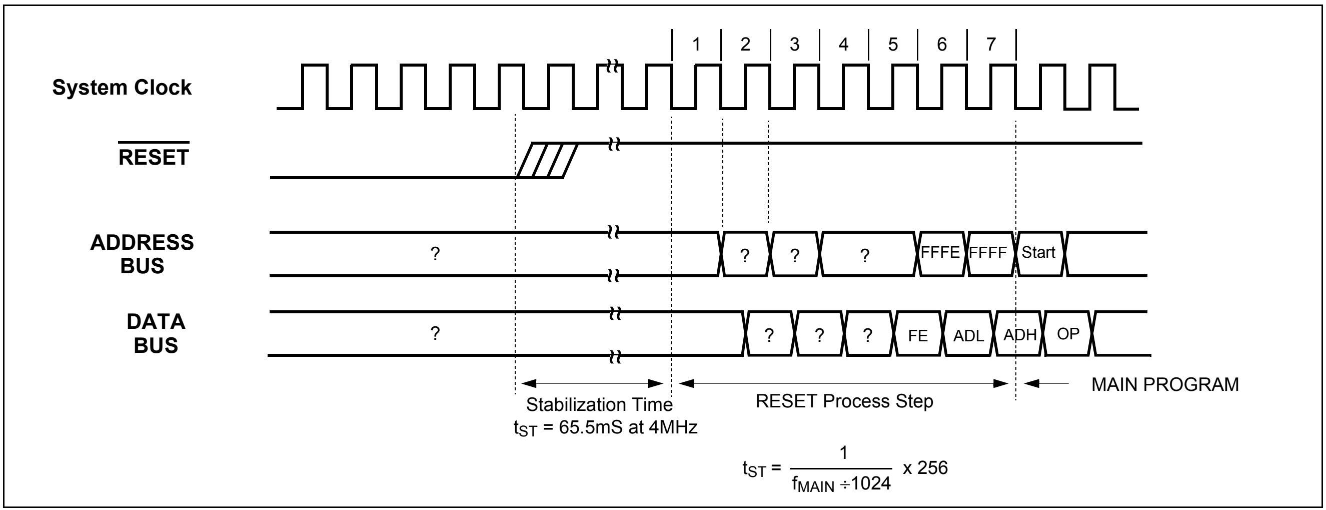



**Figure 22-3 Timing Diagram of RESET**

## 22.2 Power On Reset

The on-chip POR circuit holds down the device in RESET until $V_{DD}$ has reached a high enough level for proper operation. It will eliminate external components such as reset IC or external resistor and capacitor for external reset circuit. In addition that the $\overline{\text{RESET}}$ pin can be used to normal input port R47 by setting "POR" and "R47EN" bit of the Configuration option area(20C7H) in Flash programming. When the device starts normal operation, its operating parmeters (voltage, frequency, temperature...etc) must be met.

***Note:*** *When "POR" option is checked and "R47EN" option is not checked, $\overline{\text{RESET}}$/R47 pin acts as external Reset input pin. In this case, the external reset circuit should be connected to $\overline{\text{RESET}}$ pin. If external reset is not needed, not only "POR", but also "R47EN" option should be checked.*

## 22.3 SVLD Reset

Refer to "23. Supply Voltage Level Detector (SVLD)"

## 22.4 Watchdog Timer Reset

Refer to "14. WATCH DOG TIMER"

# 23. Supply Voltage Level Detector (SVLD)

The MC80F7708A has an on-chip SVLD(Supply Voltage Level Detector) circuitry to immunize against power noise. The SVLD control register SVLDR can enable or disable the built in reset circuitry. The Block diagram of SVLD is shown in the Figure 23-1.

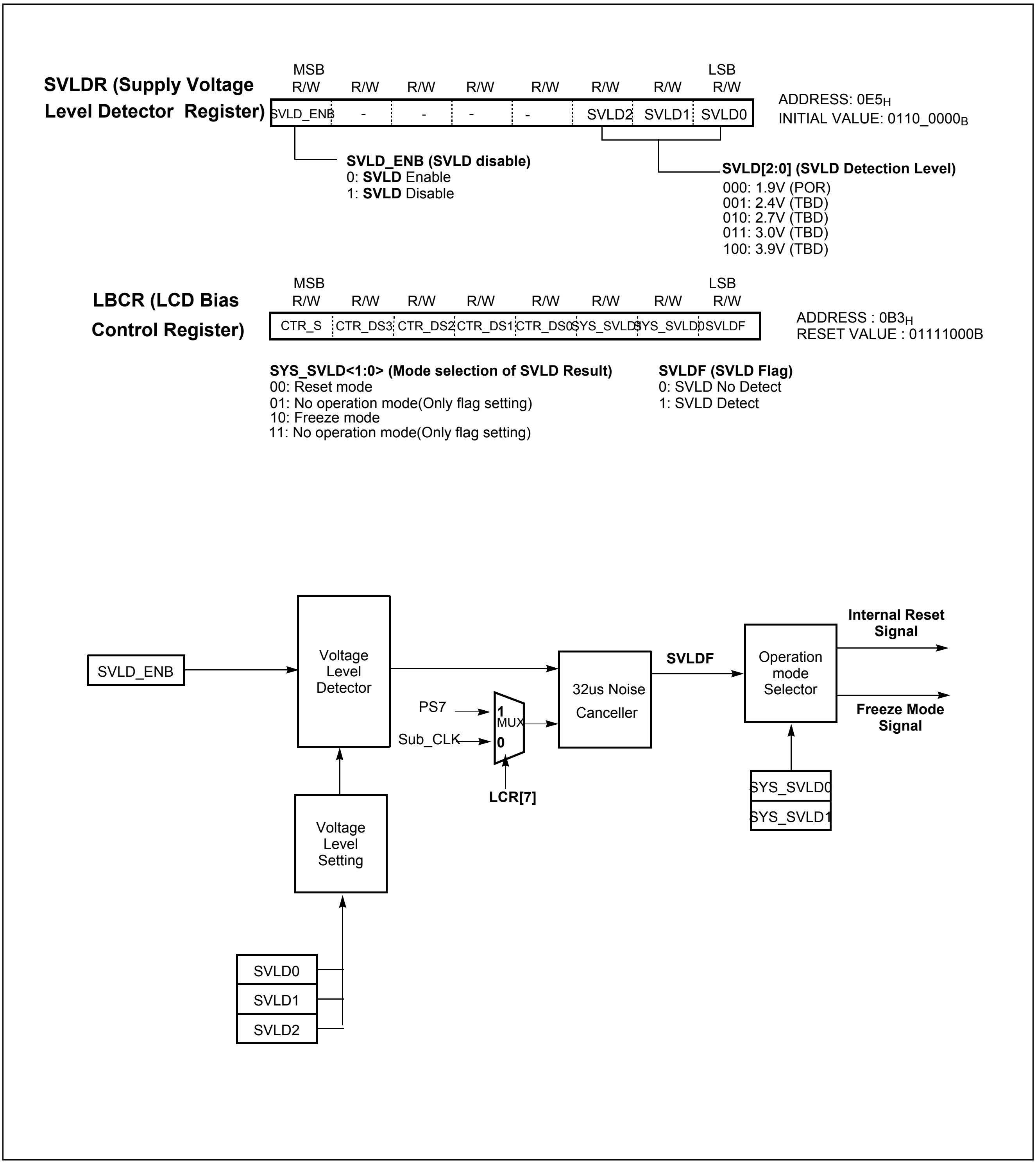



Figure 23-1 Block Diagram of SVLD (Built-In Reset)

The SVLD of MC80F7708A has 4 detection level which can be selected by SVLD[2:0].

SVLD does not generates internal reset signal or freeze mode signal because the 32us noise canceller eleminates low level detection signal less than 32us.

SVLD result can be selected by SYS_SVLD[1:0] of LBCR register. When SYS_SVLD[1:0] is set to “00”, BIR generates reset singnal. If SYS_SVLD[1:0] is set to “10”, it generates freeze mode signal and CPU freeze until the VDD voltage returns to regular level.

SVLDF is set to “1” when SVLD occurs. It can be used to distinguish reset caused by SVLD and other.

When the POR is used, the SVL detection level should be set to the level less than POR level. If the POR level is 1.9V, SVL level 2.4V, 2.7V,3.0V and 3.9V can not operate.

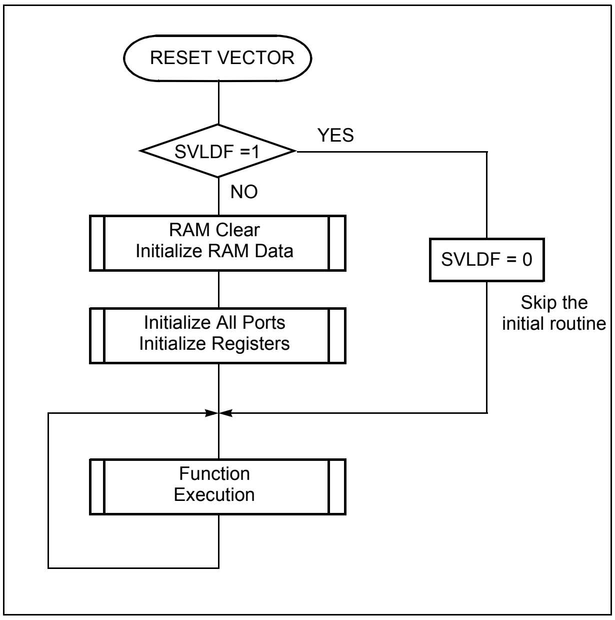



**Figure 23-2 Example Flow of Reset flow by SVLD**

## 24. Osillation Noise Protector

The Oscillation Noise Protector (ONP) is used to supply stable internal system clock by excluding the noise which could be entered into oscillator and recovery the oscillation fail. This function could be enabled or disabled by the "ONP" bit of the Device configuration area ($20C7_H$) for the MC80F7708A.

The ONP function is like below.

- Recovery the oscillation wave crushed or loss caused by high frequency noise.

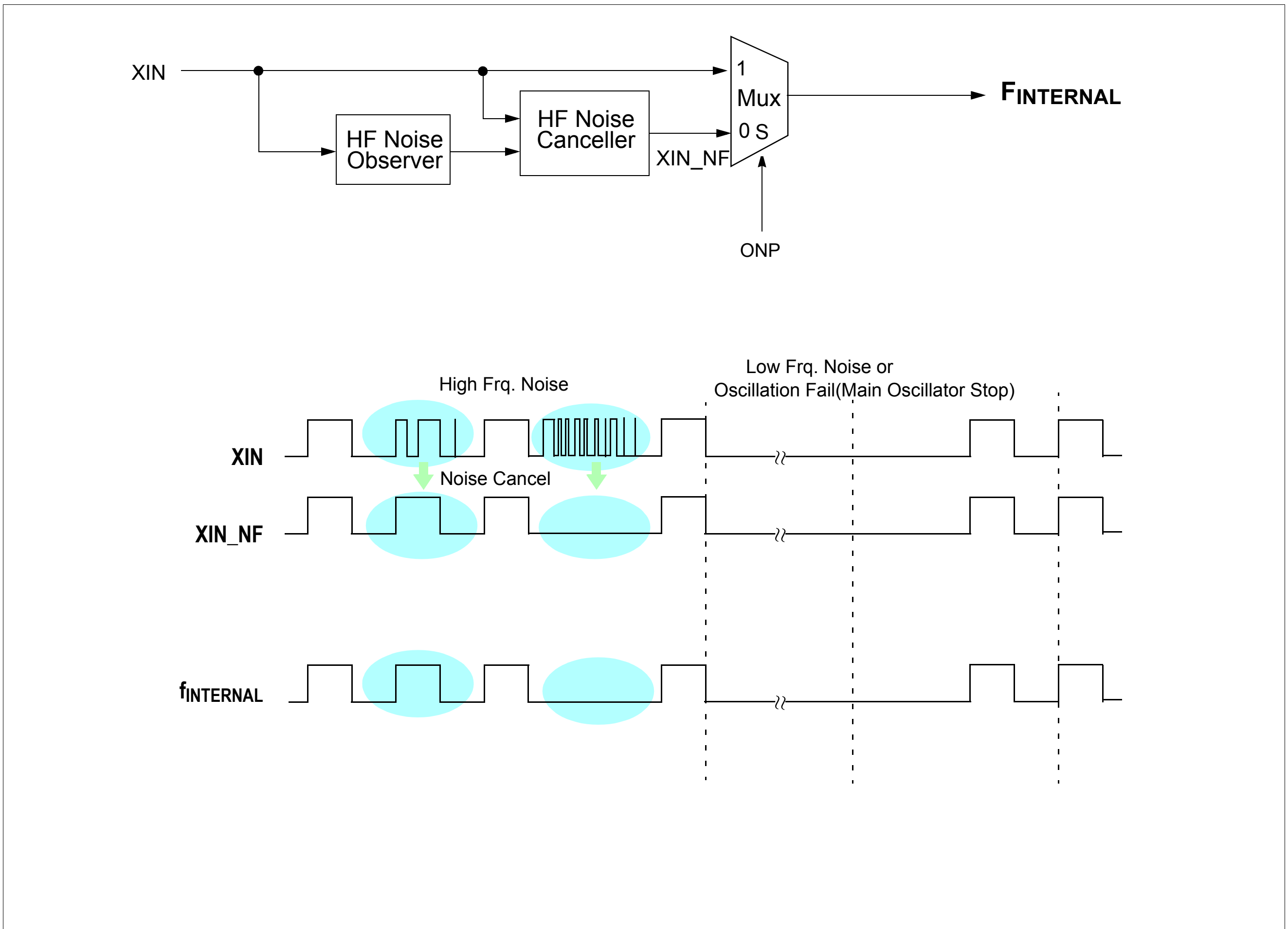



**Figure 24-1 Block Diagram of ONP Respective Wave Forms**

And this function can recover the external clock source when the external clock is recovered to normal state.

The "IN8MCLK", "IN4MCLK", "IN2MCLK", "IN8MCLKXO", "IN4MCLKXO", "IN2MCLKXO" option of the Device Configuration Area enables the function to operate the device by using the internal oscillator clock in ONP block as system clock. There is no need to connect the x-tal, resonator, RC and R externally. After selecting the this option, the period of internal oscillator clock could be checked by $X_{OUT}$ outputting clock divided the internal oscillator clock by 4.

# 25. FLASH PROGRAMMING SPEC.

## 25.1 FLASH Configuration Byte

Except the user program memory, there is configuration byte(address $20C6_H$ and $20C7_H$) for the selection of program lock, BIT clock selection, ONP,oscillation configuration and reset configuration. The configuration byte of FLASH is shown as Figure 25-1. It could be served when user use the FLASH programmer.

***Note:*** *The Configuration Option may not be read exactly when VDD rising time is very slow. It is recommended to adjust the VDD rising time faster than 40ms/V (200ms from 0V to 5V).*

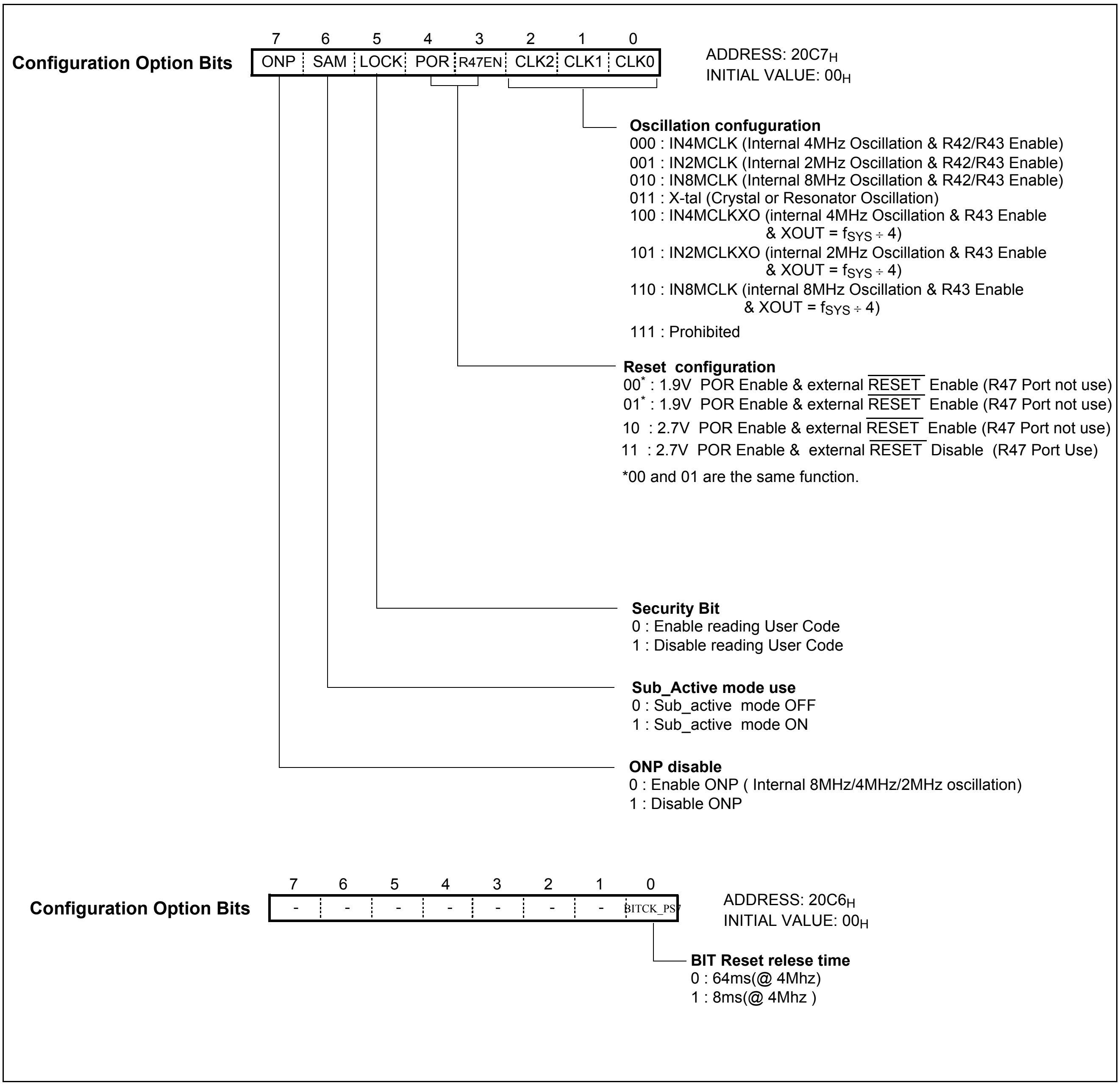



**Figure 25-1 The FLASH Configuration Byte**

## 25.2 FLASH Programming

The MC80F7708A is a MTP microcontroller. Its internal user memory is constructed with FLASH ROM..

Blank FLASH's internal memory is filled by $00_H$, not $FF_H$.

***Note:*** *In any case, you have to use the *.OTP file for programming, not the *.HEX file. After assemble, both OTP and HEX file are generated by automatically. The HEX file is used during program emulation on the emulator.*

### How to Program

To program the FLASH or MTP devices, user can use ABOV own programmer.

#### ABOV own programmer list

Manufacturer: ABOV Semiconductor Programmer:

**Choice-Sigma**
**StandAlone-Gang4**
**PGM-plus**

The Choice-Sigma is a ABOV Universal Single Programmer for all of ABOV FLASH/OTP devices, also the StandAlone-Gang4 can program four FLASH/OTPs at once for ABOV device.

Ask to ABOV sales part for purchasing or more detail.

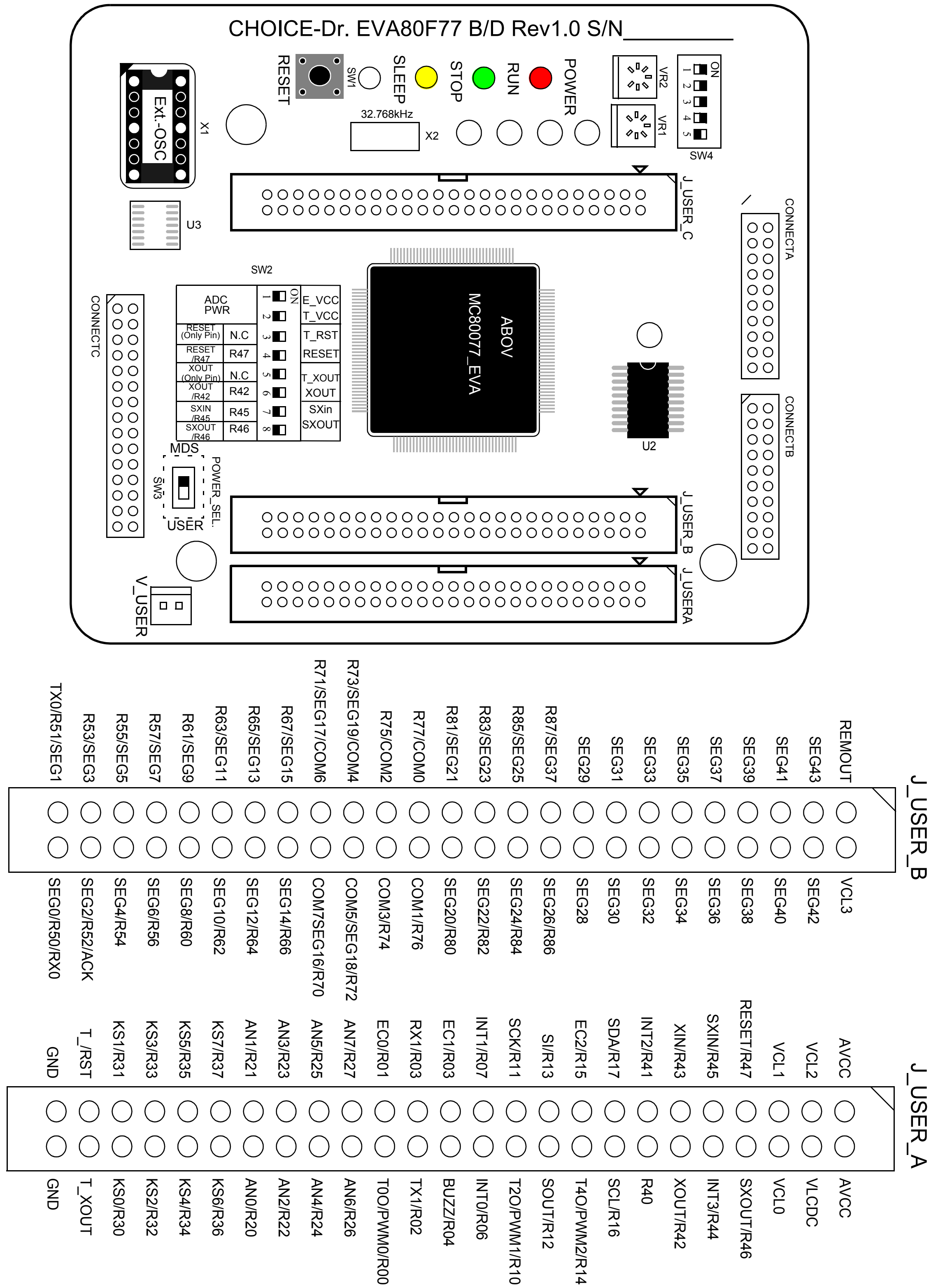
### Programming Procedure

1. Select device MC80F7708A.
2. Load the *.OTP file from the PC. The file is composed of Motorola-S1 format.
3. Set the programming address range as below table.

| Address | Set Value |
|---|---|
| Buffer start address | $E000_H$ |
| Buffer end address | $FFFF_H$ |
| Device start address | $E000_H$ |

4. Mount the socket adapter on the programmer.
5. Start program/verify.

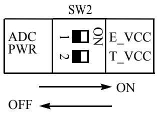
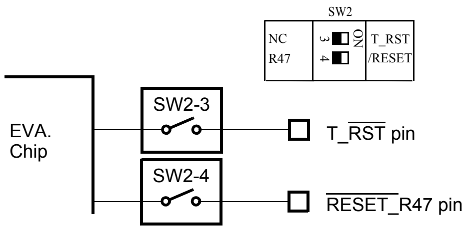
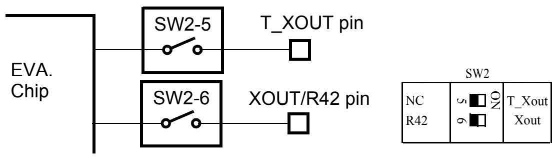
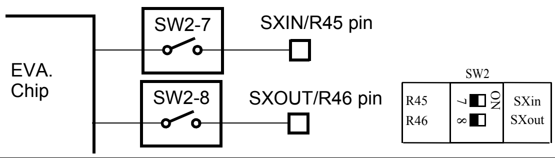
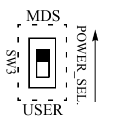
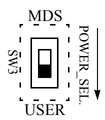
## 26. EMULATOR EVA. BOARD SETTING

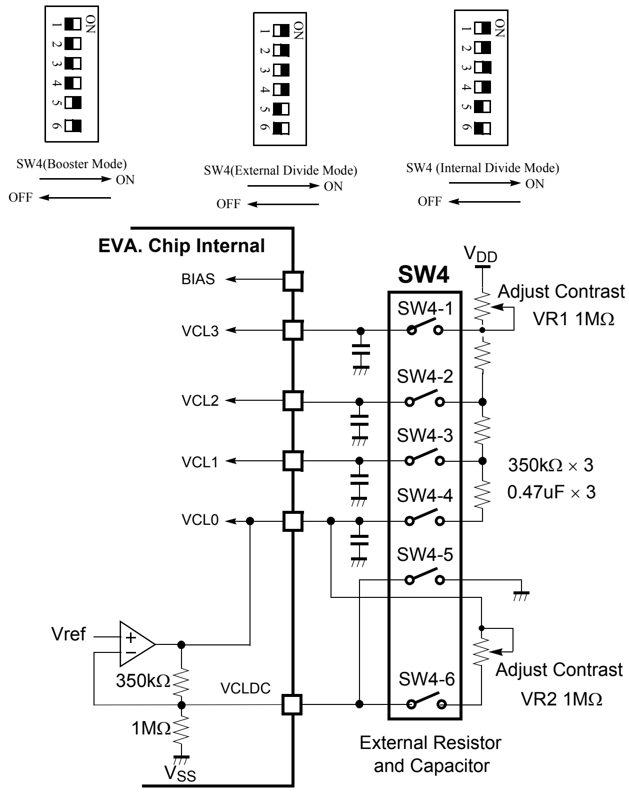
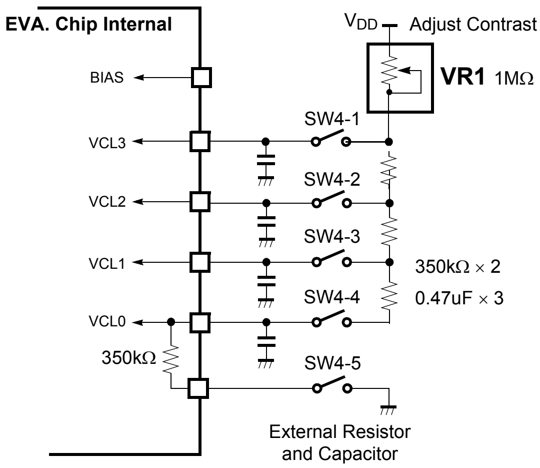


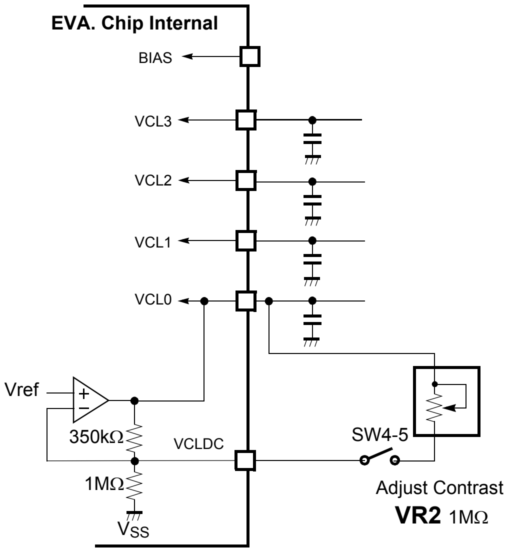

* J_USEC_C is reserved for further use, Unused in MC80F7708A.

## IP Switch and VR Setting

Before execute the user program, keep in your mind the below configuration

| DIP S/W, VR | | Description | ON/OFF Setting |
|---|---|---|---|
| SW1 | - | Emulator Reset Switch. Reset the Emulator. | Reset the Emulator. |
| SW2 | 1<br>2 | ADC voltage is supplied from EVA B/D voltage<br>ADC voltage is supplied from Target(User POD pin) voltage<br> | SW2-1 must be **ON** position.<br>SW2-2 must be **OFF** position. |
| | 3<br>4 | SW2-3 is NC pin.<br> | Normally **OFF**.<br>EVA. chip can be reset by external user target board.<br><br>SW2-3<br>**ON** : T_RST pin connected<br>**OFF** : Disconnect circuit.<br><br>SW2-4<br>**ON** : Reset is available by either user target system board or Emulator RESET switch.<br>**OFF** : Reset the MCU by Emulator RESET switch. Does not work from user target board. |
| | 5<br>6 | SW2-5 is NC pin.<br>POD XOUT pin configuration<br> | Normally **OFF**.<br>MCU XOUT pin is disconnected internally in the Emulator. Some circumstance user may connect this circuit.<br><br>SW2-5<br>**ON** : Output T_XOUT pin connected<br>**OFF** : Disconnect circuit<br>SW2-6<br>**ON** : Target board Port signal<br>**OFF** : EVA board OSC XOUT |
| | 7<br>8 | SW2-7, SW2-8 select sub-clock function.<br> | SW2-7<br>**ON** : Target board Port signal<br>**OFF** : EVA board Sub OSC SXIN<br>SW2-8<br>**ON** : Target board Port signal<br>**OFF** : EVA board Sub OSC SXOUT |
| SW3 | 1 | This switch selects Eva. B/D Power supply source.<br><br>Use MDS Power<br><br>Use User's Power(V_USER) | Normally **MDS**.<br>This switch selects Eva. B/D Power supply source. |

<table>
<tr><th colspan="2">DIP S/W, VR</th><th>Description</th><th>ON/OFF Setting</th></tr>
<tr><td>SW4</td><td>1<br>2<br>3<br>4<br>5<br>6</td><td>External Bias Resistors Connection

</td><td>Must be ON position.<br>It serves the external bias resistors. If this switches are turned off, LCD bias voltage does not supplied, floated because there are no internal bias resistors and bias Tr. inside the Emulator.</td></tr>
<tr><td>VR1</td><td>-</td><td>External Driver Mode : Adjust the LCD contrast. It supply bias voltage and adjust the VCL1 voltage.

</td><td>Adjust the proper position as well as LCD display good.</td></tr>
</table>

<table>
<tr><th colspan="2">DIP S/W, VR</th><th>Description</th><th>ON/OFF Setting</th></tr>
<tr><td>VR2</td><td>-</td><td>Internal boost mode : Adjust the LCD contrast. It supply bias voltage and adjust the VCL1 voltage.<br>
</td><td>Adjust the proper position as well as LCD display good.</td></tr>
</table>

# 27. IN-SYSTEM PROGRAMMING

## 27.1 Getting Started / Installation

The In-System Programming (ISP) is performed without removing the microcontroller from the system. The In-System Programming(ISP) facility consists of a series of internal hardware resources coupled with internal firmware through the serial port. The In-System Programming (ISP) facility has made in-circuit programming in an embedded application possible with a minimum of additional expense in components and circuit board area. The following section details the procedure for accomplishing the installation procedure.

**1. Power off a target system.**

**2. Configure a target system as ISP mode.**
Refer to "27.3 Hardware Conditions to Enter the ISP Mode"

**3. Attach a USB-SIO-ISP B/D into a target system.**

**4. Run the ABOV USB-SIO-ISP software.**

**- Down load the ISP S/W from http://*www.abov.co.kr*.**

**- Unzip the download file and run USB-SIO-ISP.exe**

**5. Select a device in the USB-SIO-ISP S/W.**

**6. Power on a target system.**

**7. Execute ISP command such as read, program, auto... by pressing buttons on the USB-SIO-ISP S/W.**

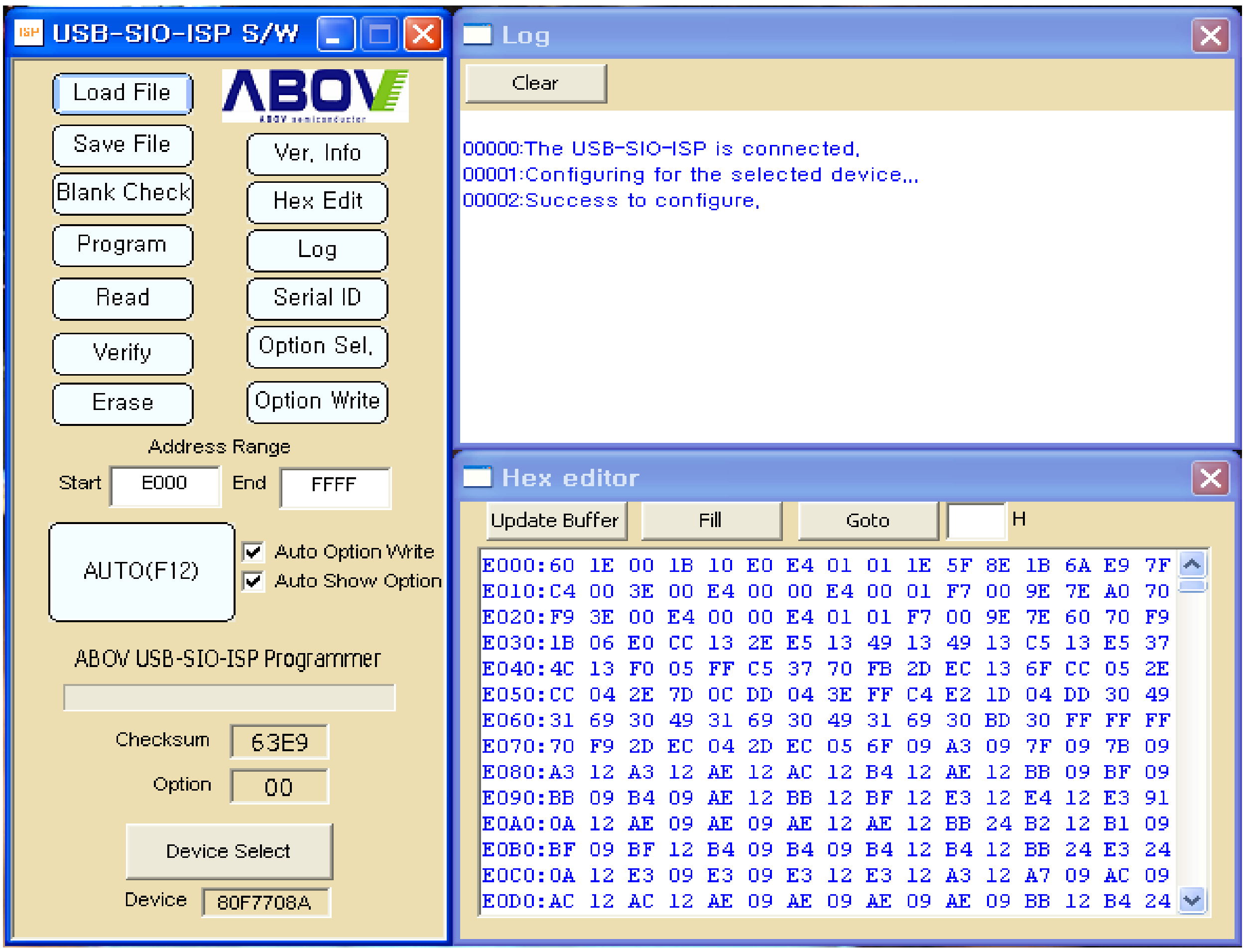



**Figure 27-1 ISP software**

## 27.2 Basic ISP S/W Information

The Figure 27-1 is the ISP software based on Windows™. This software is only supporting devices with ISO. In case of not detecting its baudrates an user manually have to select specific baudrates.

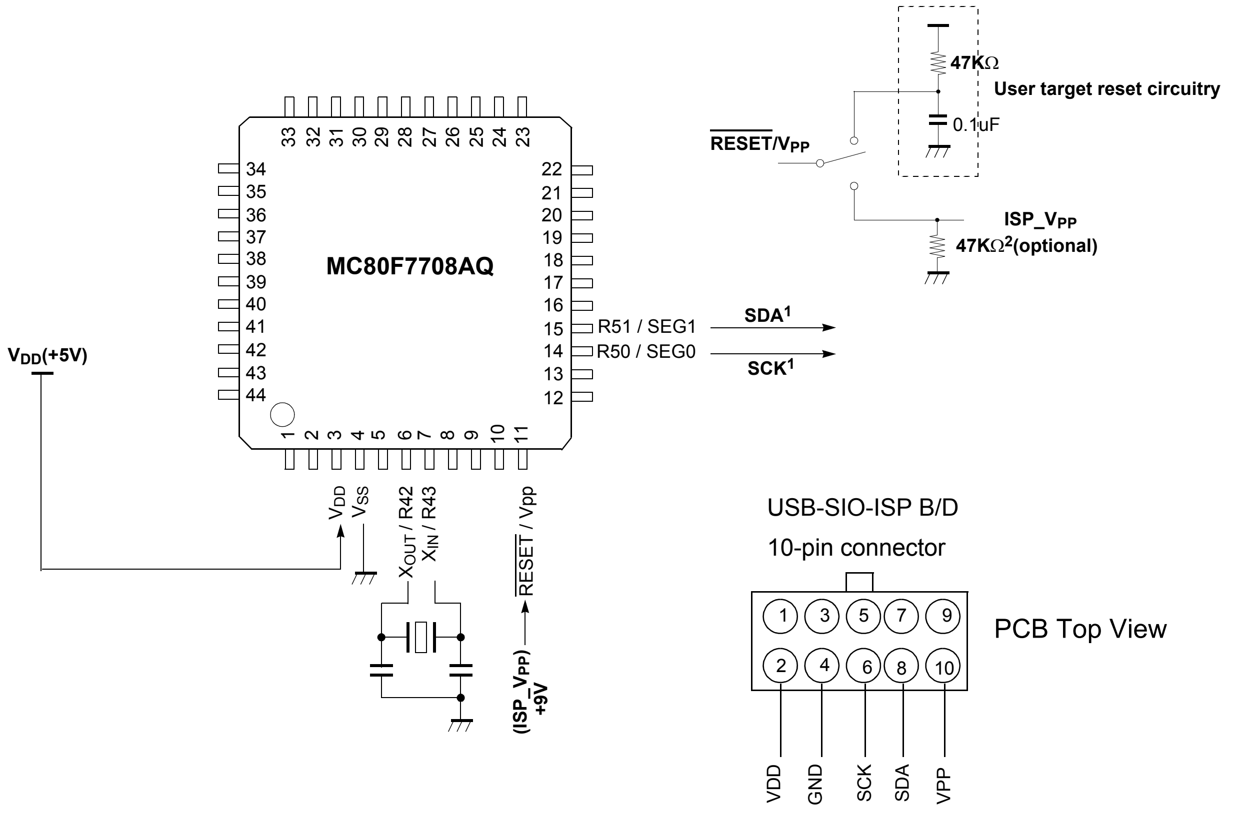
| Function | Description |
|---|---|
| Load File | Load the data from the selected file storage into the memory buffer. |
| Save File | Save the current data in your memory buffer to a disk storage by using the Intel Motorola HEX format. |
| Blank Check | Verify whether or not a device is in an erased or unprogrammed state. |
| Program | This button enables you to place new data from the memory buffer into the target device. |
| Read | Read the data in the target MCU into the buffer for examination. The checksum will be displayed on the checksum box. |
| Verify | Assures that data in the device matches data in the memory buffer. If your device is secured, a verification error is detected. |
| Erase | Erase the data in your target MCU before programming it. |
| Option Selection | Set the configuration data of target MCU. The security locking is set with this button. |
| Option Write | Progam the configuration data of target MCU. The security locking is performed with this button. |
| Start ______ | Starting address |
| End ______ | End address |
| AUTO | Following sequence is performed ; 1.Erase 2.Program 3.Verify 4.Option Write |
| Auto Option Write | If you want to program the option(config) value after pressing the Auto Button, chek this button |
| Auto Show Option | If you check this button, the option(config) dialog is displayed whenever pressing the Auto button. |
| Checksum | Display the checksum(Hexdecimal) after reading the target device. |
| Select Device | Select target device. You need to select a device before turning on the target VDD |
| Update Buffer | Update buffer by pressing this button. |
| Fill | Fill the selected area with a data. |
| Goto | Display the selected page. |
| Serial ID | To program the serial ID. |

**Table 27-1 ISP Function Description**

***Note:*** *MCU configuration value is erased after operation. It must be configured to match with user target board. Otherwise, it is failed to enter ISP mode, or its operation is not desirable.*

## 27.3 Hardware Conditions to Enter the ISP Mode

The boot loader can be executed by holding , RESET/VPP and RESET/VPP. as +9V. The ISP function uses following pins: SCL, SDA,




1. If other signals affect ISP communiction in ISP mode, disconnect these pins by using a jumper or a switch.
2. The pull down resister is optional. If user set to ISP Mode without connection ISP B/D, the Vpp/Reset Port is in floating The pull down resistor is for blocking this undefined state. Do not power on a application B/D in ISP mode without connecting a ISP board.

**Figure 27-2 ISP Configuration**

## 27.4 Sequence to enter ISP mode/user mode

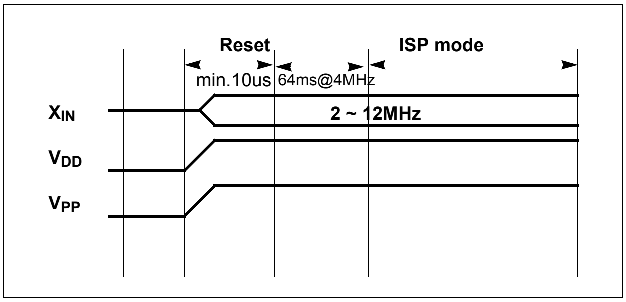



**Figure 27-3 Timing diagram to enter the ISP mode**

Sequence to enter ISP mode from user mode.

***1. Power off a target system.***
***2. Configure a target system as ISP mode.***
***3. Attach a ISP B/D into a target system.***
***4. Run the ISP S/W and Select Device.***
***5. Power on a target system.***

Sequence to enter user mode from ISP mode.

***1. Close the ISP S/W.***
***2. Power off a target system.***
***3. Configure a target system as user mode***
***4. Detach a ISP B/D from a target system.***
***5. Power on.***

## 27.5 Reference ISP Circuit Diagram and ABOV Supplied ISP Board

The ISP software and hardware circuit diagram are provided at www.abov.co.kr.

To get a ISP B/D, contact to sales department. The following circuit diagram is for reference use.

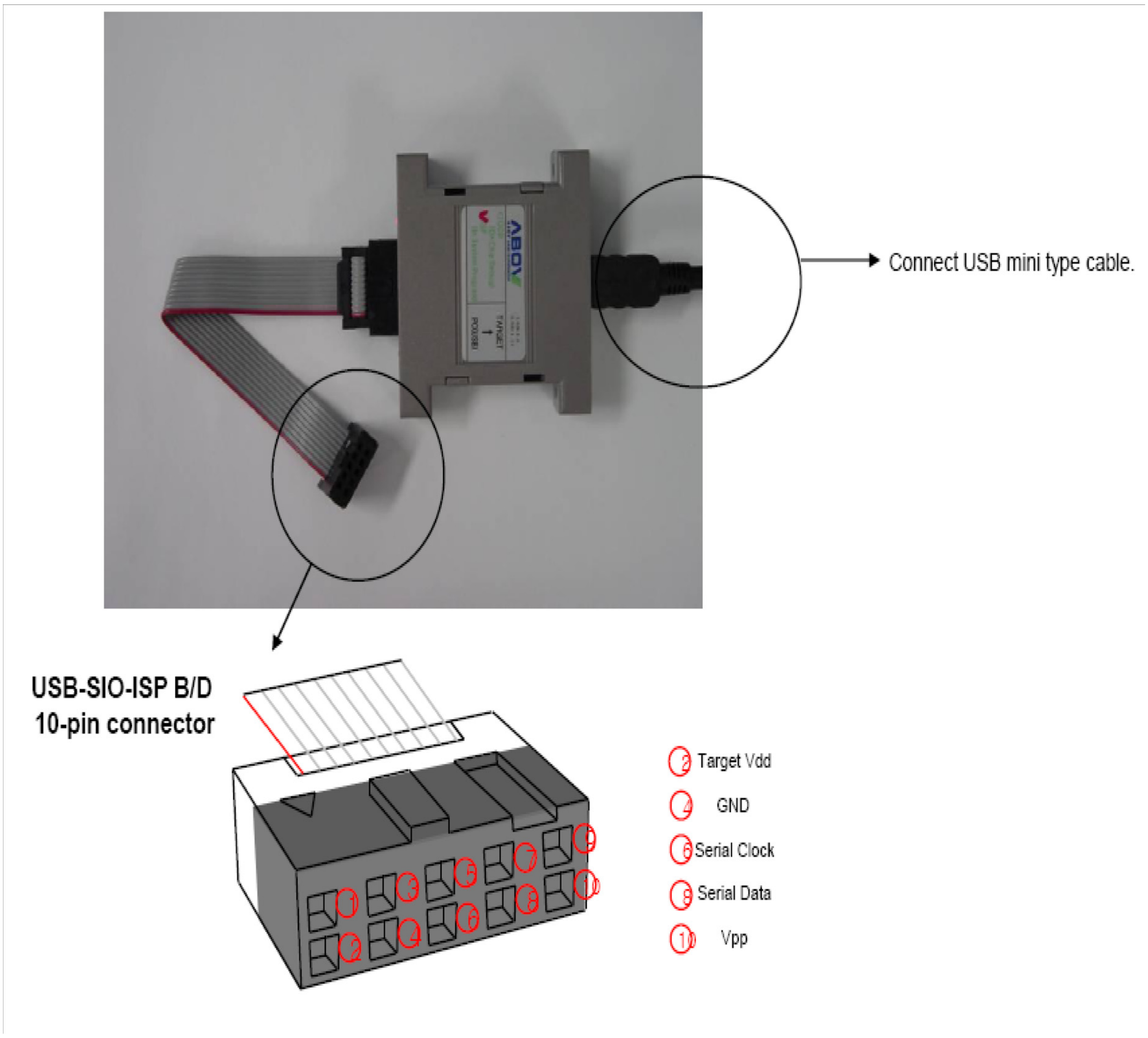
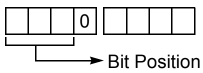
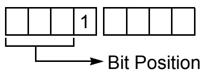



**Figure 27-4 ISP board supplied by ABOV**

# APPENDIX

# A. INSTRUCTION

## A.1 Terminology List

| Terminology | Description |
|---|---|
| A | Accumulator |
| X | X - register |
| Y | Y - register |
| PSW | Program Status Word |
| #imm | 8-bit Immediate data |
| dp | Direct Page Offset Address |
| !abs | Absolute Address |
| [ ] | Indirect expression |
| { } | Register Indirect expression |
| { }+ | Register Indirect expression, after that, Register auto-increment |
| .bit | Bit Position |
| A.bit | Bit Position of Accumulator |
| dp.bit | Bit Position of Direct Page Memory |
| M.bit | Bit Position of Memory Data ($000_H$~$0FFF_H$) |
| rel | Relative Addressing Data |
| upage | U-page ($0FF00_H$~$0FFFF_H$) Offset Address |
| n | Table CALL Number (0~15) |
| + | Addition |
| x |   Upper Nibble Expression in Opcode |
| y |   Upper Nibble Expression in Opcode |
| – | Subtraction |
| × | Multiplication |
| / | Division |
| ( ) | Contents Expression |
| ∧ | AND |
| ∨ | OR |
| ⊕ | Exclusive OR |
| ~ | NOT |
| ← | Assignment / Transfer / Shift Left |
| → | Shift Right |
| ↔ | Exchange |
| = | Equal |
| ≠ | Not Equal |

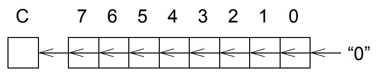
## A.2 Instruction Map

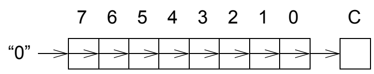
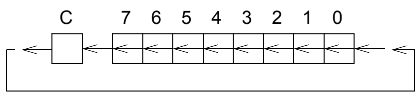
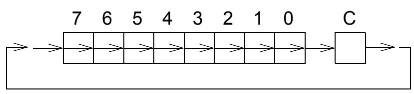
| LOW HIGH | 00000 00 | 00001 01 | 00010 02 | 00011 03 | 00100 04 | 00101 05 | 00110 06 | 00111 07 | 01000 08 | 01001 09 | 01010 0A | 01011 0B | 01100 0C | 01101 0D | 01110 0E | 01111 0F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | - | SET1 dp.bit | BBS A.bit,rel |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

## A.3 Instruction Set

### Arithmetic / Logic Operation

| NO. | MNEMONIC | OP CODE | BYTE NO | CYCLE NO | OPERATION | FLAG NVGBHIZC |
|---|---|---|---|---|---|---|
| 1 | ADC #imm | 04 | 2 | 2 | Add with carry.<br>A ← ( A ) + ( M ) + C | NV--H-ZC |
| 2 | ADC dp | 05 | 2 | 3 | | |
| 3 | ADC dp + X | 06 | 2 | 4 | | |
| 4 | ADC !abs | 07 | 3 | 4 | | |
| 5 | ADC !abs + Y | 15 | 3 | 5 | | |
| 6 | ADC [ dp + X ] | 16 | 2 | 6 | | |
| 7 | ADC [ dp ] + Y | 17 | 2 | 6 | | |
| 8 | ADC { X } | 14 | 1 | 3 | | |
| 9 | AND #imm | 84 | 2 | 2 | Logical AND<br>A ← ( A ) ∧ ( M ) | N-----Z- |
| 10 | AND dp | 85 | 2 | 3 | | |
| 11 | AND dp + X | 86 | 2 | 4 | | |
| 12 | AND !abs | 87 | 3 | 4 | | |
| 13 | AND !abs + Y | 95 | 3 | 5 | | |
| 14 | AND [ dp + X ] | 96 | 2 | 6 | | |
| 15 | AND [ dp ] + Y | 97 | 2 | 6 | | |
| 16 | AND { X } | 94 | 1 | 3 | | |
| 17 | ASL A | 08 | 1 | 2 | Arithmetic shift left<br> | N-----ZC |
| 18 | ASL dp | 09 | 2 | 4 | | |
| 19 | ASL dp + X | 19 | 2 | 5 | | |
| 20 | ASL !abs | 18 | 3 | 5 | | |
| 21 | CMP #imm | 44 | 2 | 2 | Compare accumulator contents with memory contents<br>( A ) - ( M ) | N-----ZC |
| 22 | CMP dp | 45 | 2 | 3 | | |
| 23 | CMP dp + X | 46 | 2 | 4 | | |
| 24 | CMP !abs | 47 | 3 | 4 | | |
| 25 | CMP !abs + Y | 55 | 3 | 5 | | |
| 26 | CMP [ dp + X ] | 56 | 2 | 6 | | |
| 27 | CMP [ dp ] + Y | 57 | 2 | 6 | | |
| 28 | CMP { X } | 54 | 1 | 3 | | |
| 29 | CMPX #imm | 5E | 2 | 2 | Compare X contents with memory contents<br>( X ) - ( M ) | N-----ZC |
| 30 | CMPX dp | 6C | 2 | 3 | | |
| 31 | CMPX !abs | 7C | 3 | 4 | | |
| 32 | CMPY #imm | 7E | 2 | 2 | Compare Y contents with memory contents<br>( Y ) - ( M ) | N-----ZC |
| 33 | CMPY dp | 8C | 2 | 3 | | |
| 34 | CMPY !abs | 9C | 3 | 4 | | |
| 35 | COM dp | 2C | 2 | 4 | 1'S Complement : ( dp ) ← ~( dp ) | N-----Z- |
| 36 | DAA | DF | 1 | 3 | Decimal adjust for addition | N-----ZC |
| 37 | DAS | CF | 1 | 3 | Decimal adjust for subtraction | N-----ZC |
| 38 | DEC A | A8 | 1 | 2 | Decrement<br>M ← ( M ) - 1 | N-----Z- |
| 39 | DEC dp | A9 | 2 | 4 | | |
| 40 | DEC dp + X | B9 | 2 | 5 | | |
| 41 | DEC !abs | B8 | 3 | 5 | | |
| 42 | DEC X | AF | 1 | 2 | | |
| 43 | DEC Y | BE | 1 | 2 | | |
| 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 ) | N-----Z- |
| 46 | EOR dp | A5 | 2 | 3 | | |
| 47 | EOR dp + X | A6 | 2 | 4 | | |
| 48 | EOR !abs | A7 | 3 | 4 | | |
| 49 | EOR !abs + Y | B5 | 3 | 5 | | |
| 50 | EOR [ dp + X ] | B6 | 2 | 6 | | |
| 51 | EOR [ dp ] + Y | B7 | 2 | 6 | | |
| 52 | EOR { X } | B4 | 1 | 3 | | |
| 53 | INC A | 88 | 1 | 2 | Increment<br>M ← ( M ) + 1 | N-----Z- |
| 54 | INC dp | 89 | 2 | 4 | | |
| 55 | INC dp + X | 99 | 2 | 5 | | |
| 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<br> | N-----ZC |
| 60 | LSR dp | 49 | 2 | 4 | | |
| 61 | LSR dp + X | 59 | 2 | 5 | | |
| 62 | LSR !abs | 58 | 3 | 5 | | |
| 63 | MUL | 5B | 1 | 9 | Multiply : YA ← Y × A | N-----Z- |
| 64 | OR #imm | 64 | 2 | 2 | Logical OR<br>A ← ( A ) ∨ ( M ) | N-----Z- |
| 65 | OR dp | 65 | 2 | 3 | | |
| 66 | OR dp + X | 66 | 2 | 4 | | |
| 67 | OR !abs | 67 | 3 | 4 | | |
| 68 | OR !abs + Y | 75 | 3 | 5 | | |
| 69 | OR [ dp + X ] | 76 | 2 | 6 | | |
| 70 | OR [ dp ] + Y | 77 | 2 | 6 | | |
| 71 | OR { X } | 74 | 1 | 3 | | |
| 72 | ROL A | 28 | 1 | 2 | Rotate left through carry<br> | N-----ZC |
| 73 | ROL dp | 29 | 2 | 4 | | |
| 74 | ROL dp + X | 39 | 2 | 5 | | |
| 75 | ROL !abs | 38 | 3 | 5 | | |
| 76 | ROR A | 68 | 1 | 2 | Rotate right through carry<br> | N-----ZC |
| 77 | ROR dp | 69 | 2 | 4 | | |
| 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 ) | NV--HZC |
| 81 | SBC dp | 25 | 2 | 3 | | |
| 82 | SBC dp + X | 26 | 2 | 4 | | |
| 83 | SBC !abs | 27 | 3 | 4 | | |
| 84 | SBC !abs + Y | 35 | 3 | 5 | | |
| 85 | SBC [ dp + X ] | 36 | 2 | 6 | | |
| 86 | SBC [ dp ] + Y | 37 | 2 | 6 | | |
| 87 | SBC { X } | 34 | 1 | 3 | | |
| 88 | TST dp | 4C | 2 | 3 | Test memory contents for negative or zero<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- |

### Register / Memory Operation

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

## 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 : (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 |

## 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- |

## 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<br>pc ← jump address | -------- |
| 21 | JMP [!abs] | 1F | 3 | 5 | | |
| 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 ), $pc_H$ ← "0FF$_H$" . | -------- |
| 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) | -------- |

### Control Operation & Etc.

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