

# **SN8F29E4x Series**

## **USER'S MANUAL**

Version 1.4

SN8F29E47  
SN8F29E48  
SN8F29E49

# **SONiX 8-Bit Micro-Controller**

SONiX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONiX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONiX products are not designed, intended, or authorized for use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONiX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONiX products for any such unintended or unauthorized application, Buyer shall indemnify and hold SONiX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONiX was negligent regarding the design or manufacture of the part.

### AMENDENT HISTORY

Version	Date	Description
VER 1.0		First issue.
VER 1.1	2018/07/20	Update Interrupt priority/UART Diagram
VER 1.2	2018/08/16	Update Features UART(P10)/ PIN ASSIGNMENT(P13)/ADC, regulator, DAC and OPA internal Diagram(P130)/ Electrical characteristic(P161)/ PGIA control register AMPM(P142).
VER 1.3	2019/02/19	Addition SN8F29E49/SN8F29E48/SN8F29E47 in the title page.
VER 1.4	2021/05/28	Add OTP Programming. (P165)

### User Precautions

- ✧ User want to disable **P00IEN,P01IEN,T0IEN,TC0IEN,TC1IEN,ADCIEN,UTXIEN,URXIEN,MSPIEN,UTX2IEN,URX2IEN**  
✧ must add “**JMP \$+2**” instruction after **b0bclr** (**P00IEN,P01IEN,T0IEN,TC0IEN,TC1IEN,ADCIEN,UTXIEN,URXIEN,MSPIEN,UTX2IEN,URX2IEN**) command.

Example for disable INT0 interrupt:

Example for disable TC0 interrupt:

Example for disable ADCIEN interrupt:

☞ **IDE compiler**

```
b0bclr P00IEN ;
JMP $+2 ;
```

☞ **IDE compiler**

```
b0bclr TC0IEN ;
JMP $+2 ;
```

☞ **IDE compiler**

```
b0bclr ADCIEN ;
JMP $+2 ;
```

☞ **IDS compiler**

```
__ASM{
b0bclr P00IEN;
JMP $+2 ;}
```

☞ **IDS compiler**

```
__ASM{
b0bclr TC0IEN ;
JMP $+2 ;}
```

☞ **IDS compiler**

```
__ASM{
b0bclr ADCIEN;
JMP $+2 ;}
```

# Table of Content

AMENDMENT HISTORY .....	2
<b>1 PRODUCT OVERVIEW .....</b>	<b>10</b>
1.1 OVERVIEW .....	10
1.2 FEATURES .....	10
1.3 SYSTEM BLOCK DIAGRAM .....	12
1.4 PIN DESCRIPTIONS .....	16
1.5 PIN CIRCUIT DIAGRAMS .....	18
<b>2 CENTRAL PROCESSOR UNIT (CPU) .....</b>	<b>20</b>
2.1 MEMORY MAP .....	20
2.1.1 PROGRAM MEMORY (ROM) .....	20
2.2 RESET VECTOR (0000H) .....	21
2.2.1 INTERRUPT VECTOR (0008H~001CH) .....	22
2.2.2 INTERRUPT PRIORITY .....	22
2.2.3 LOOK-UP TABLE DESCRIPTION .....	24
2.2.4 JUMP TABLE DESCRIPTION .....	26
2.2.5 CHECKSUM CALCULATION .....	28
2.2.6 DATA MEMORY (RAM) .....	29
2.2.7 SYSTEM REGISTER .....	31
SYSTEM REGISTER DESCRIPTION .....	31
2.2.8 SYSTEM REGISTER DEFINITION .....	32
2.2.9 ACCUMULATOR .....	34
2.2.10 PROGRAM COUNTER .....	35
2.2.11 H, L REGISTERS .....	37
2.2.12 R REGISTER .....	38
2.2.13 Y, Z REGISTERS .....	38
2.2.14 X REGISTERS .....	39
2.2.15 PROGRAM FLAG .....	40
2.2.16 W REGISTERS .....	40
2.3 ADDRESSING MODE .....	42
2.3.1 IMMEDIATE ADDRESSING MODE .....	42
2.3.2 DIRECTLY ADDRESSING MODE .....	42
2.3.3 INDIRECTLY ADDRESSING MODE .....	42
2.4 STACK OPERATION .....	43
2.4.1 OVERVIEW .....	43
2.4.2 STACK POINTER .....	43

2.4.3	STACK BUFFER.....	44
2.5	STACK OPERATION .....	44
2.5.1	OVERVIEW.....	44
2.5.2	Fcpu Code Option.....	44
2.5.3	Security code option.....	44
2.5.4	Noise Filter code option.....	44
<b>3</b>	<b>RESET .....</b>	<b>45</b>
3.1	OVERVIEW.....	45
3.2	POWER ON RESET .....	46
3.3	WATCHDOG RESET .....	46
3.4	BROWN OUT RESET .....	46
3.4.1	OVERVIEW.....	46
3.4.2	The system operation voltage .....	47
3.4.3	LOW VOLTAGE DETECTOR (LVD) .....	47
3.4.4	Watchdog reset: .....	48
<b>4</b>	<b>SYSTEM CLOCK.....</b>	<b>49</b>
4.1	OVERVIEW.....	49
4.2	SYSTEM CLOCK .....	49
4.3	INTERNAL HIGH RC (IHRC) .....	49
4.4	SYSTEM CLOCK BLOCK DIAGRAM .....	50
4.5	OSCM REGISTER .....	50
4.6	SYSTEM CLOCK MEASUREMENT .....	51
<b>5</b>	<b>SYSTEM OPERATION MODE.....</b>	<b>52</b>
5.1	OVERVIEW.....	52
5.2	SYSTEM MODE SWITCHING.....	53
5.3	WAKE-UP.....	54
5.3.1	OVERVIEW.....	54
5.3.2	WAKEUP TIME.....	54
5.3.3	PIW WAKEUP CONTROL REGISTER.....	56
<b>6</b>	<b>INTERRUPT .....</b>	<b>57</b>
6.1	OVERVIEW.....	57
6.2	INTERRUPT OPERATION.....	57
6.3	INTEN INTERRUPT ENABLE REGISTER.....	58
6.4	INTRQ INTERRUPT REQUEST REGISTER .....	60
6.5	GIE GLOBAL INTERRUPT OPERATION .....	61
6.6	EXTERNAL INTERRUPT OPERATION (INT0~INT1) .....	63

6.7	T0 INTERRUPT OPERATION .....	64
6.8	TC0 INTERRUPT OPERATION .....	65
6.9	TC1 INTERRUPT OPERATION .....	66
6.10	UART/UART2 INTERRUPT OPERATION .....	66
6.11	MULTI-INTERRUPT OPERATION .....	67
<b>7</b>	<b>I/O PORT .....</b>	<b>68</b>
7.1	OVERVIEW .....	68
7.2	I/O PORT MODE .....	69
7.3	I/O PULL UP REGISTER .....	70
7.4	I/O PORT DATA REGISTER .....	71
<b>8</b>	<b>TIMER .....</b>	<b>72</b>
8.1	WATCHDOG TIMER .....	72
8.2	T0 8-BIT BASIC TIMER .....	74
8.2.1	OVERVIEW .....	74
	T0 TIMER OPERATION .....	74
8.3	T0 MODE REGISTER .....	76
8.3.1	T0C COUNTING REGISTER .....	76
8.4	TC0 16-BIT TIMER/COUNTER .....	78
8.4.1	OVERVIEW .....	78
8.4.2	TC0M MODE REGISTER .....	79
8.5	TC0 TIMER OPERATION .....	80
8.5.1	TC0C COUNTING REGISTER .....	81
8.5.2	TC0R AUTO-RELOAD REGISTER .....	82
8.5.3	TC0D PWM DUTY REGISTER .....	83
8.5.4	TC0 EVENT COUNTER .....	84
8.5.5	PULSE WIDTH MODULATION (PWM) .....	85
8.5.6	TC0 TIMER OPERATION EXAMPLE .....	86
8.6	TC1 16-BIT TIMER/COUNTER .....	88
8.6.1	OVERVIEW .....	88
8.6.2	TC1M MODE REGISTER .....	89
8.6.3	TC1C COUNTING REGISTER .....	90
8.6.4	TC1R AUTO-RELOAD REGISTER .....	91
8.6.5	TC1D PWM DUTY REGISTER .....	92
8.6.6	TC1 EVENT COUNTER .....	93
8.6.7	PULSE WIDTH MODULATION (PWM) .....	94
8.6.8	TC1 TIMER OPERATION EXAMPLE .....	95
<b>9</b>	<b>UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER(UART) .....</b>	<b>97</b>

9.1	OVERVIEW .....	97
9.2	UART OPERATION .....	98
9.3	UART BAUD RATE .....	99
9.4	UART TRANSFER FORMAT .....	100
9.5	ABNORMAL POCKET .....	101
9.6	UART RECEIVER CONTROL REGISTER .....	101
9.7	UART TRANSMITTER CONTROL REGISTER .....	102
9.8	UART DATA BUFFER .....	102
9.9	UART OPERATION EXAMLPE .....	103
<b>10</b>	<b>UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER (UART2) .....</b>	<b>106</b>
10.1	OVERVIEW .....	106
10.2	UART2 OPERATION .....	107
10.3	UART2 BAUD RATE .....	108
10.4	UART2 TRANSFER FORMAT .....	109
10.5	ABNORMAL POCKET .....	110
10.6	UART2 RECEIVER CONTROL REGISTER .....	110
10.7	UART2 TRANSMITTER CONTROL REGISTER .....	111
10.8	UART2 DATA BUFFER .....	111
10.9	UART2 OPERATION EXAMLPE .....	112
<b>11</b>	<b>MAIN SERIAL PORT(MSP) .....</b>	<b>113</b>
11.1	OVERVIEW .....	113
11.2	MSP STATUS REGISTER .....	113
11.3	MSP MODE REGISTER 1 .....	114
11.4	MSP MODE REGISTER 2 .....	115
11.5	MSP MSPBUF REGISTER .....	116
11.6	MSP MSPADR REGISTER .....	116
11.7	SLAVE MODE OPERATION .....	116
10.7.1	Addressing .....	116
10.7.2	Slave Receiving .....	117
10.7.3	Slave Transmission .....	117
10.7.4	General Call Address .....	118
10.7.5	Slave Wake up .....	119
10.8	MASTER MODE .....	120
10.8.1	Mater Mode Support .....	120
10.8.2	MSP Rate Generator .....	120
10.8.3	MSP Mater START Condition .....	121
10.8.3.1	WCOL Status Flag .....	121
10.8.4	MSP Master mode Repeat START Condition .....	121

10.8.4.1	WCOL Status Flag .....	121
10.8.5	Acknowledge Sequence Timing.....	122
10.8.5.1	WCOL Status Flag .....	122
10.8.6	STOP Condition Timing.....	122
10.8.6.1	WCOL Status Flag .....	122
10.8.7	Clock Arbitration .....	123
10.8.8	Master Mode Transmission .....	123
10.8.8.1	BF Status Flag .....	123
10.8.8.2	WCOL Flag.....	123
10.8.8.3	ACKSTAT Status Flag .....	123
10.8.9	Master Mode Receiving.....	124
10.8.9.1	BF Status Flag .....	124
10.8.9.2	MSPOV Flag.....	124
10.8.9.3	WCOL Flag.....	124
<b>12</b>	<b>LCD DRIVER.....</b>	<b>125</b>
12.1	OVERVIEW.....	125
12.2	LCD TIMING.....	125
12.3	LCD RAM LOCATION.....	127
12.4	LCDM1 REGISTER .....	128
12.5	LCDM2 REGISTER .....	129
12.6	C-TYPE LCD MODE .....	129
<b>13</b>	<b>ADC,REGULATOR,DAC AND OPA .....</b>	<b>130</b>
13.1	OVERVIEW.....	130
13.2	VOLTAGE REGULATOR.....	132
13.3	VREG-CHARGE PUMP MODE REGISTER.....	132
13.4	ADC INPUT CHANNEL SELECTION .....	133
13.4.1	VCHS-ADC input channel selection Register.....	134
13.4.2	ADC input signal channel selection table .....	134
13.5	ADC STRUCTURE AND CONTROL REGISTER.....	135
13.5.1	OVERVIEW.....	135
13.5.2	Analog Inputs and Voltage Operation Range.....	136
13.5.3	Reference Voltage .....	136
13.5.4	Input Buffer.....	136
13.5.5	ADC Gain .....	136
13.6	ADCM1 -ADC MODE1 CONTROL REGISTER .....	137
13.7	ADCM2 -ADC MODE1 CONTROL REGISTER .....	138
13.8	ADCM3-ADC MODE1 CONTROL REGISTER .....	139
13.9	ADC DATA OUTPUT .....	140

13.10	PGIA CONTROL REGISTER .....	142
13.11	LBTM: LOW BATTERY DETECT .....	143
13.11.1	OVERVIEW .....	143
13.11.2	LBTM: Low Battery Detect .....	143
13.12	OPM REGISTER .....	144
13.13	1-CHANNEL DIGITAL TO ANALOG CONVERTER .....	147
13.13.1	OVERVIEW .....	147
13.13.2	DAC Register .....	147
<b>14</b>	<b>LED DRIVER .....</b>	<b>149</b>
14.1	OVERVIEW .....	149
14.2	LED REGISTER .....	149
<b>15</b>	<b>IN SYSTEM PROGRAM FLASH ROM .....</b>	<b>150</b>
15.1	OVERVIEW .....	150
15.2	ISP FLASH ROM ERASE OPERATION .....	151
15.3	ISP FLASH ROM PROGRAM OPERATION .....	152
15.4	ISP PROGRAM/ERASE CONTROL REGISTER .....	155
15.5	ISP ROM ADDRESS REGISTER .....	155
15.6	ISP RAM ADDRESS REGISTER .....	155
15.7	ISP ROM PROGRAMMING LENGTH REGISTER .....	156
<b>16</b>	<b>INSTRUCTION TABLE .....</b>	<b>157</b>
<b>17</b>	<b>ELECTRICAL CHARACTERISTIC .....</b>	<b>159</b>
17.1	ABSOLUTE MAXIMUM RATING .....	159
17.2	ELECTRICAL CHARACTERISTIC .....	159
<b>18</b>	<b>DEVELOPMENT TOOL .....</b>	<b>163</b>
18.1	OVERVIEW .....	163
18.2	SMART DEVELOPMENT ADAPTER .....	163
18.3	OTP PROGRAMMING PIN TO TRANSITION BOARD MAPPING .....	165
<b>19</b>	<b>APPLICATION CIRCUIT ,START-KIT .....</b>	<b>166</b>
19.1	APPLICATION CIRCUIT .....	166
19.2	START KIT CIRCUIT .....	167
19.3	SN8F29E49 STARTER-KIT .....	168
19.4	PROGRAMMER INSTALLATION .....	170
<b>20</b>	<b>MARKING DEFINITION .....</b>	<b>172</b>
20.1	INTRODUCTION .....	172
20.2	MARKING INDETIFICATION SYSTEM .....	172



---

20.3	DATECODE SYSTEM.....	173
<b>21</b>	<b>PACKAGE INFORMATION .....</b>	<b>174</b>
21.1	LQFP 100 PIN .....	174
21.2	LQFP 80 PIN .....	175
21.3	LQFP 64 PIN .....	176

# 1 PRODUCT OVERVIEW

## 1.1 OVERVIEW

SN8F29E49 series 8-bit micro-controller is a new series production applied advanced semiconductor technology to implement flash ROM architecture. Under flash ROM platform, SN8F29E49 builds in in-system-programming (ISP) function extending to EEPROM emulation and Embedded ICE(EICE) function. It offers 2-set individual programmable PWMs, serial interfaces and flexible operating modes. Powerful functionality, high reliability and low power consumption can apply to AC power application and battery level application easily.

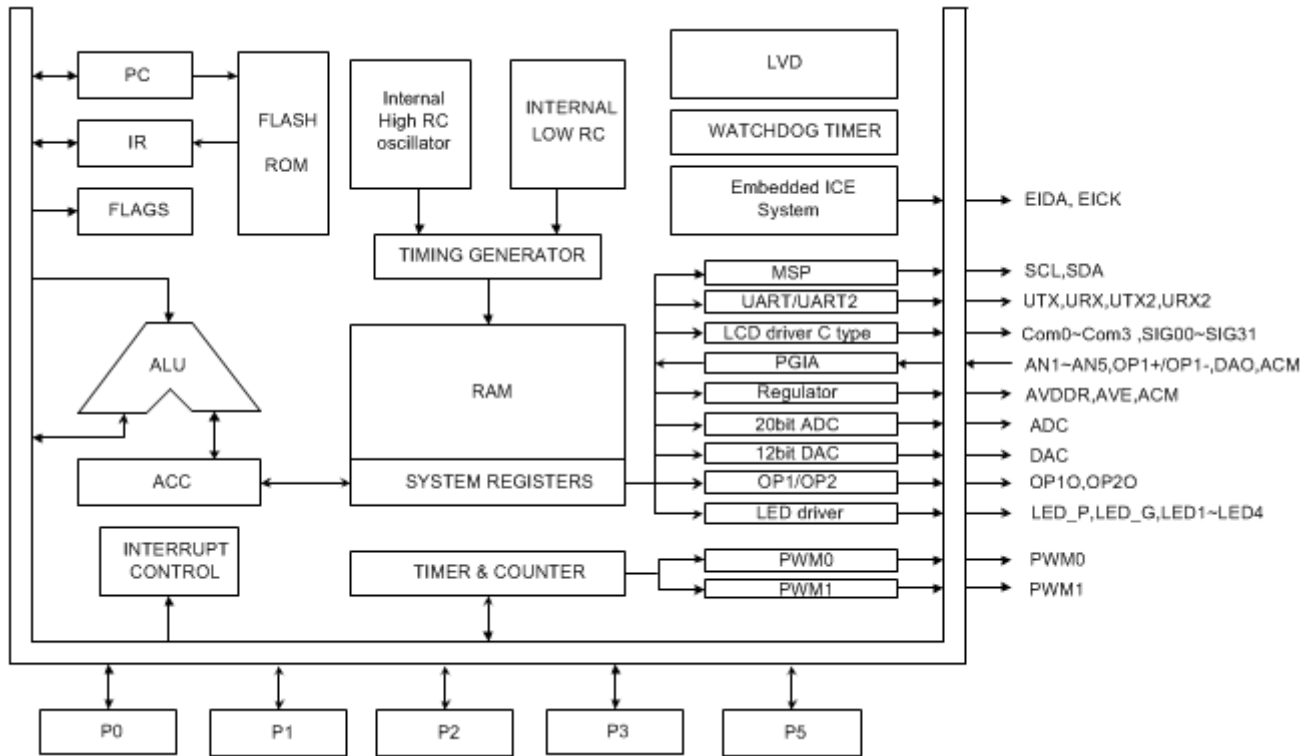
## 1.2 FEATURES

- ◆ **Memory configuration**  
Flash ROM size: 64K \* 16bits  
RAM size: 6144 \* 8 bits  
16-levels stack buffer  
LCD RAM size: 4\*32 bits, 6\*30 bits
- ◆ **I/O pin configuration (Total 24+10pins=34pins)**  
Bi-directional: P0, P1, P2, P3, P5  
Wakeup:  
P0, P1 level change trigger  
P0, P1 Green mode, Sleep mode wake up  
Pull-up resisters:P0, P1, P2, P3, P5  
P3 pins shared with SEG pins
- ◆ **Interrupt sources**  
11 internal interrupts:  
T0,TC0,TC1,ADC,UART,UART2,MSP  
2 external interrupts:P00, P01
- ◆ **Timer System**  
**T0:** 8-bit timer with green mode wakeup function  
Real Time Clock (RTC): 0.5 second  
(embedded hour/min/Sec registers)  
**TC0/TC1:** 16-bit timer Auto-reload timer  
/ PWM
- ◆ **On chip watchdog timer**
- ◆ **Fcpu(Instruction cycle)**  
 $F_{cpu} = F_{hosc}/2, F_{hosc}/4, F_{hosc}/8$
- ◆ **Powerful instructions**  
All instructions are one word length.  
Most of instructions are one cycle only  
Maximum instruction cycle is "2".  
JMP instruction jumps to all ROM area.  
All ROM area lookup table function (MOVC)  
Hardware multiplier (MUL)
- ◆ **20-Bit Delta Sigma ADC with 18-bit Noise Free**  
ADC Gain: 1x, 2x, 4x  
ADC Input Ch.: AN1~AN5, OP1+, OP1-, OPOUT1,  
OP2+, OP2-, OPOUT2, DAC, ACM, GND,  
VDD\_DET,  
Input channel Configuration: Single / Differential  
ADC Conversion Rate: 7.6Hz ~ 3.9KHz
- ◆ **Single power supply: 2.0V ~ 3.6V**  
Digital Circuit: 2.0V ~ 3.6V  
Analog Circuit: 2.4V ~ 3.6V
- ◆ **Built-in Regulator for analog circuit**  
AVDDR: 2.4V with current 10mA for Analog Part  
AVE: Output 2.0V for reference source  
ACM: 1.0V Analog Common voltage
- ◆ **Two channel UART Interface**
- ◆ **One channel MSP Interface**
- ◆ **One channel 12-bit DAC**  
Reference: External, Internal 0.2V / 0.35V / 0.65V / 1.2V
- ◆ **Two Operational Amplifier**
- ◆ **On chip voltage comparator**  
  
Built in low battery detect 2.2V~2.9V (LBT variation:100mV )  
External P10 comparator input.
- ◆ **LCD driver: 128 / 180 dots**  
1/4 and 1/6 duty with 1/3 bias  
(SEG26~31 Share Pins function with IO)  
4 common \* 32 segment  
1C type LCD
- ◆ **Build-in Internal High Clock 8MHz**
- ◆ **Build in Embedded ICE function.**
- ◆ **System clocks and Operating modes**  
Internal high clock: RC type up to 8MHz  
internal low clock: RC type 32KHz(3V)  
External low clock: Crystal type 32KHz.  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop  
Green mode: Periodical wakeup by T0/TC0 timer
- ◆ **PGIA: 1x, 16x, 32, 64x, 128x**
- ◆ **Package: LQFP 100/80/64, Dice**

 **Feature Selection table**

CHIP	Flash	RAM	Stack	LCD	Timer			I/O	ADC	DAC	OP	PWM	MSP	UART	Wakeup Pin	Package
					T0	TC0	TC1									
SN8F29E49	64K*16	6k*8	16	4*32	V	V	V	24+10	20-Bit 15-Ch.	12-Bit	2	2	1	2	8	LQFP100
SN8F29E48	64K*16	6k*8	16	4*32	V	V	V	21+6	20-Bit 12-Ch.	12-Bit	2	2	1	1	8	LQFP80
SN8F29E47	64K*16	6k*8	16	0	V	V	V	33	20-Bit 15-Ch.	12-Bit	2	2	1	1	8	LQFP64

## 1.3 SYSTEM BLOCK DIAGRAM



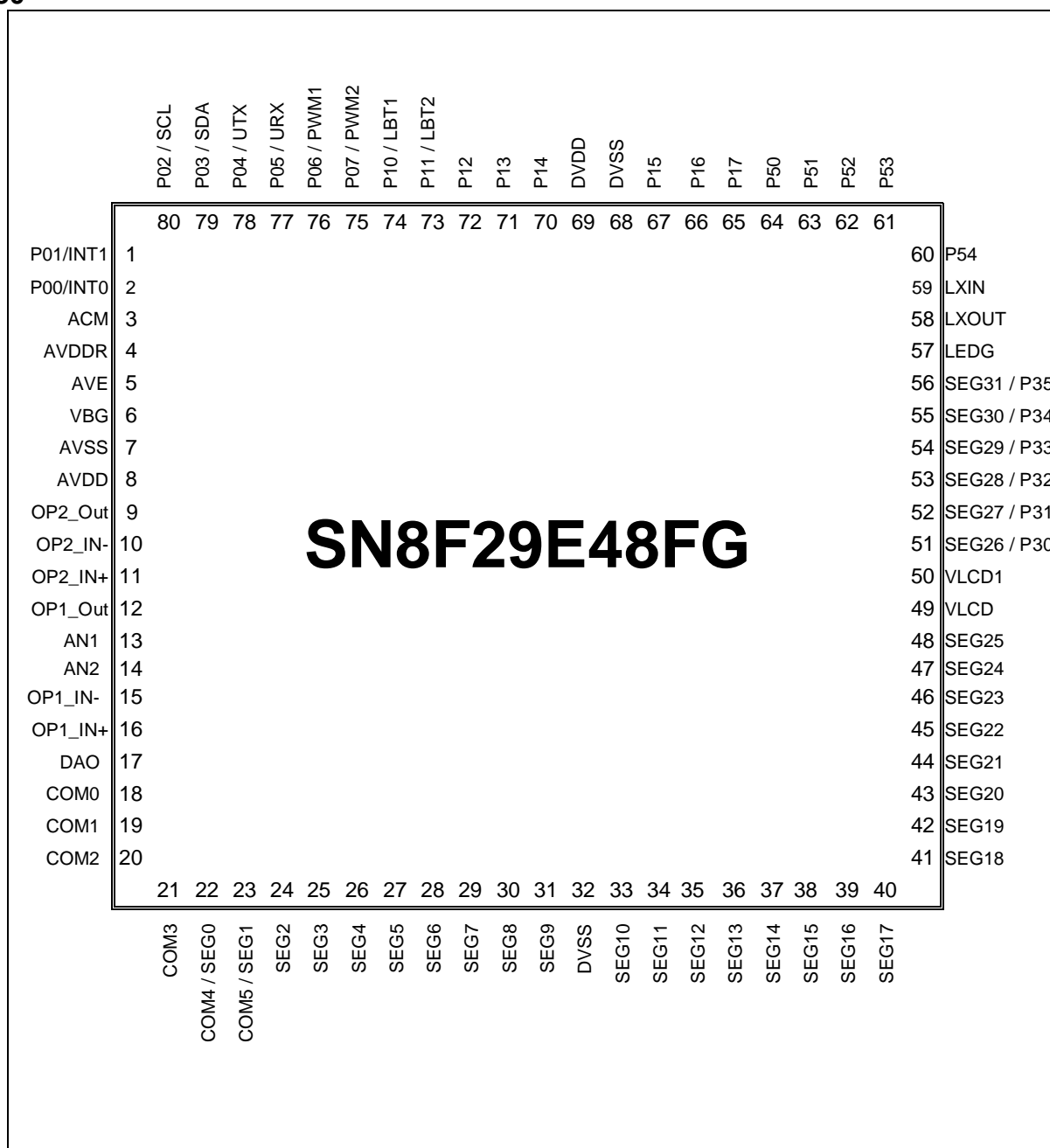
## PIN ASSIGNMENT

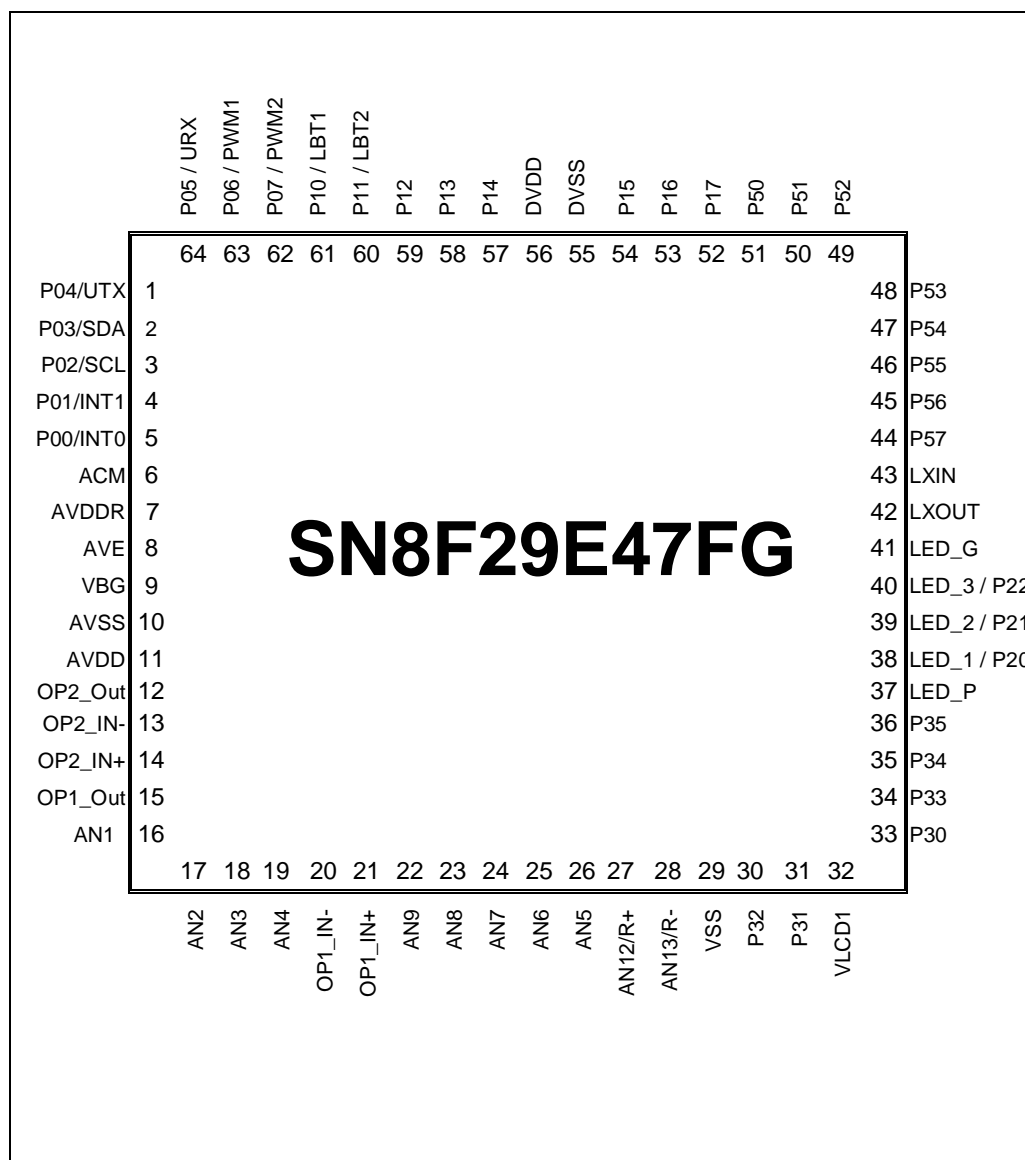
## LQFP100

		P36 / UTX2	P00 / INT0	P01 / INT1	P02 / SCL	P03 / SDA	P04 / UTX	P05 / URX	P06 / PWM1	P07 / PWM2	P10 / LBT1	P11 / LBT2	P12	P13	P14	DVDD	P37 / URX2	DVSS	P15	P16	P17	P50	P51	P52	P53	P54		
		100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76		
ACM	1																										75	P55
AVDDR	2																										74	P56
AVE	3																										73	P57
VBG	4																										72	LXIN
R-	5																										71	LXOUT
R+	6																										70	LED_G
AVSS	7																										69	LED4 / P23
AVDD	8																										68	LED3 / P22
OP2_Out	9																										67	LED2 / P21
OP2_IN-	10																										66	LED1 / P20
OP2_IN+	11																										65	LED_P
OP1_Out	12																										64	SEG31 / P35
AN1	13																										63	SEG30 / P34
AN2	14																										62	SEG29 / P33
AN3	15																										61	SEG28 / P32
AN4	16																										60	SEG27 / P31
OP1_IN-	17																										59	SEG26 / P30
OP1_IN+	18																										58	VLCD1
AN5	19																										57	VLCD
AN6	20																										56	SEG25
AN7	21																										55	SEG24
AN8	22																										54	SEG23
AN9	23																										53	SEG22
DA_Ref	24																										52	SEG21
DAO	25																										51	SEG20
		26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50		
		COM0	COM1	COM2	COM3	COM4 / SEG0	COM5 / SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	DVSS	SEG10	SEG11	SEG12	SEG13	SEG14	SEG15	SEG16	SEG17	SEG18	SEG19		

SN8F29E49FG

SN8F29E49FG

**LQFP80**


**LQFP64**


## 1.4 PIN DESCRIPTIONS

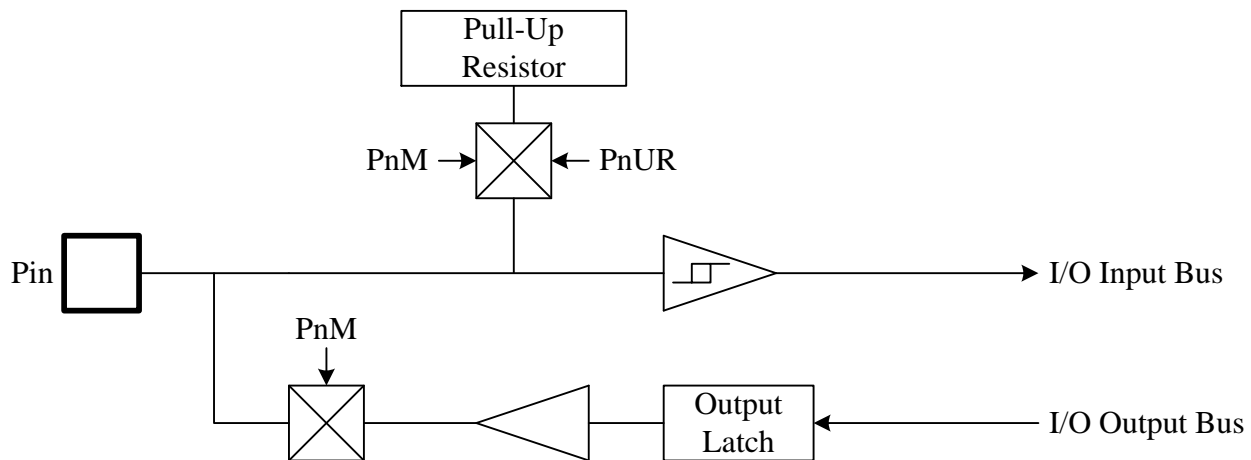
PIN NAME		DESCRIPTION
DVDD / DVSS	P	Digital function power Input pins
AVDD / AVSS	P	Analog function power Input pins
AVDDR	P	Regulator power output pin, Voltage = 2.4V / 2.7V / 3.0V. AVDDR is analog circuit power source.( ADC / OP /DAC)
AVE+	P	Regulator output = 2.0V for Sensor. Maximum output current = 5mA
VBG	P	Bad gap Voltage = 1.2V
ACM	P	ACM Voltage output = 1V
LXIN, LXOUT	I/O	Low speed (32768 Hz) oscillator pins.
P00 / INT0	I/O	Port 00 bi-direction pin / Built-in pull-up resistances. Interrupt INT0/ Wake up from Sleep and Green mode. TC0 event counter input.
P01 / INT1	I/O	Port 01 bi-direction pin / Built-in pull-up resistances. Interrupt INT1/ Wake up from Sleep and Green mode. TC1 event counter input.
P02 / SCL	I/O	Port 02 bi-direction pin and Built-in pull-up resistor / MSP SCL PIN.
P03 / SDA	I/O	Port 03 bi-direction pin and Built-in pull-up resistor / MSP SDA PIN.
P04 / UTX	I/O	Port 04 bi-direction pin and Built-in pull-up resistor / UART UTX PIN.
P05 / URX	I/O	Port 05 bi-direction pin and Built-in pull-up resistor / UART URX PIN.
P06 / PWM1	I/O	Port 06 bi-direction pin and Built-in pull-up resistor / PWM1 output PIN / TC0 signal output pin
P07 / PWM2	I/O	Port 07 bi-direction pin and Built-in pull-up resistor / PWM2 output PIN / TC1 signal output pin.
P10 / LBT1	I/O	Port 10 bi-direction pin and Built-in pull-up resistor / Low battery detect Input pin.
P11 / LBT2	I/O	Port 11 bi-direction pin and Built-in pull-up resistor / Low battery detect Gnd pin.
P12	I/O	Port 12 bi-direction pin and Built-in pull-up resistor.
P13	I/O	Port 13 bi-direction pin and Built-in pull-up resistor
P14	I/O	Port 14 bi-direction pins and Built-in pull-up resistors. Share with PGCLK.
P15	I/O	Port 15 bi-direction pins and Built-in pull-up resistors. Share with OTPCLK /SWCLK.
P16/EICK	I/O	Port 16 bi-direction pins and Built-in pull-up resistors. Share with EICK
P17/EIDA	I/O	Port 17 bi-direction pins and Built-in pull-up resistors. Share with EIDA..
P50 ~ P57	I/O	Port 50 ~ 57 bi-direction pins and Built-in pull-up resistors
DA_Ref	I	DAC External Vref input Pin.
DAO	O	DAC Output
R+	AI	ADC positive reference voltage input
R-	AI	ADC negative reference voltage input.
AN1 ~ AN9	I	Analog Input Pin
OP_IN+, OP_IN-	I	Operational Amplifier Input Channels.
OP_OUT	O	Operational Amplifier Output Channel.
LED1 ~ LED4 P20~P23	I/O	VLED Output pins. Share with P20~P23
LEDP, LEDG	P	VLED power pins



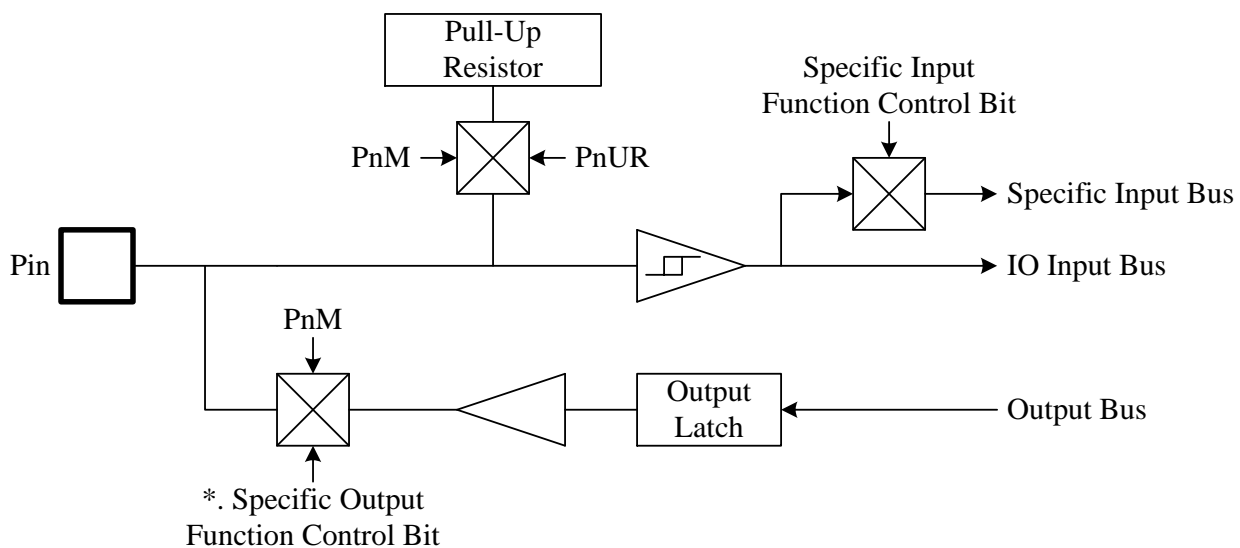
COM0 ~ COM3	O	LCD driver common pins.
COM4 / S0	O	LCD driver common pin. Share pin with segment.
COM5 / S1	O	LCD driver common pin. Share pin with segment.
S2 ~ S25	O	LCD driver segment pins.
S26 ~ S31	O	LCD driver segment pins. Share pins with P30~P35.
P36 / UTX2	I/O	Port 36 bi-direction pin and Built-in pull-up resister / UART UTX2 PIN.
P37 / URX2	I/O	Port 37 bi-direction pin and Built-in pull-up resister / UART URX2 PIN.
VLCD	P	VLCD voltage pin, connect 0.1uf capacitor.
VLCD1	P	S26~S31 for LCD function, VLCD1 connect to VLCD

## 1.5 PIN CIRCUIT DIAGRAMS

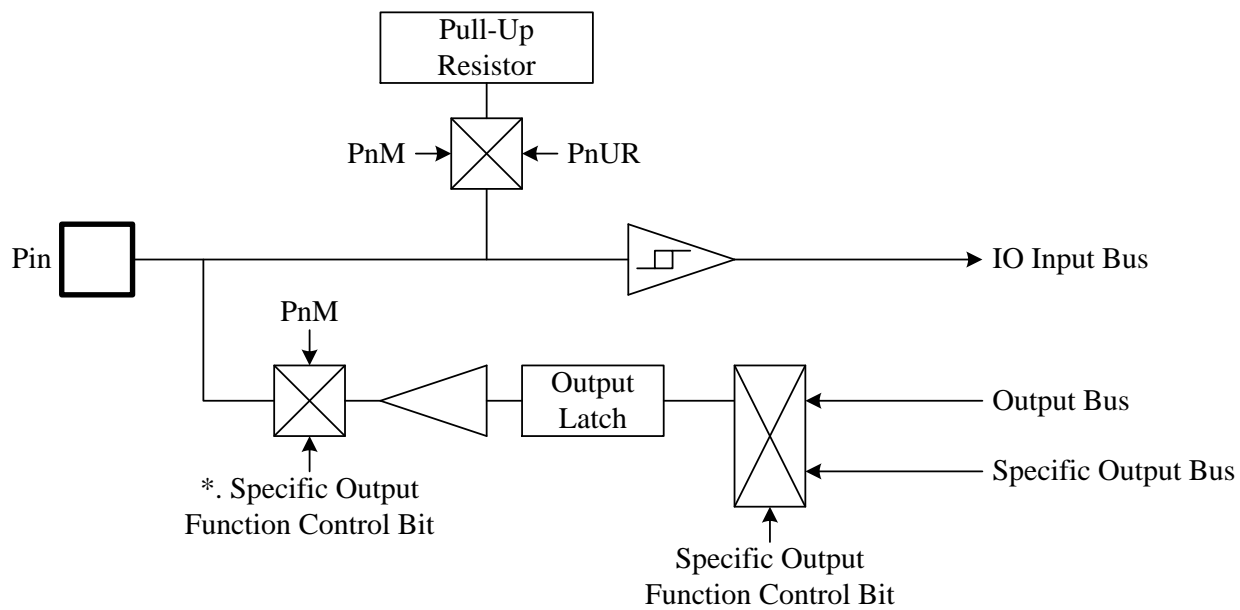
### Normal Bi-direction I/O Pin.



### Bi-direction I/O Pin Shared with Specific Digital Input Function, e.g. INT0, Event counter, UART.



\*. Some specific functions switch I/O direction directly, not through PnM register.

**Bi-direction I/O Pin Shared with Specific Digital Output Function, e.g. PWM, UART.**

**\*. Some specific functions switch I/O direction directly, not through PnM register.**

# 2 CENTRAL PROCESSOR UNIT (CPU)

## 2.1 MEMORY MAP

### 2.1.1 PROGRAM MEMORY (ROM)

64K words FLASH ROM

ROM			
0000H	<b><i>Reset vector</i></b>	Reset vector	
0001H	<b><i>General purpose area</i></b>		
...			
0007H			
0008H		<b><i>INT0</i></b>	Interrupt vector
000AH		<b><i>INT1</i></b>	
000CH		<b><i>T0</i></b>	
000EH	<b><i>TC0</i></b>		
0010H	<b><i>TC1</i></b>		
0012H	<b><i>ADC</i></b>		
0014H	<b><i>UART TX Interrupt Vector</i></b>		
0016H	<b><i>UART RX Interrupt Vector</i></b>		
0018H	<b><i>MSP Interrupt Vector</i></b>		
001AH	<b><i>UART2 TX Interrupt Vector</i></b>		
001CH	<b><i>UART2 RX Interrupt Vector</i></b>		
001DH			User program
...			
...		<b><i>General purpose area</i></b>	
...			
...			
...			
...			
...			
...			
...			
FFFBH		End of user program	
FFFCH		<b><i>Reserved</i></b>	
...			
...			
FFFFH			

## 2.2 RESET VECTOR (0000H)

A one-word vector address area is used to execute system reset.

- ☞ **Power On Reset (POR=1).**
- ☞ **Watchdog Reset (WDT=1).**

After power on reset, external reset or watchdog timer overflow reset, then the chip will restart the program from address 0000h and all system registers will be set as default values. The following example shows the way to define the reset vector in the program memory.

☞ **Example: Defining Reset Vector**

```

                                ORG      0          ; 0000H
                                JMP      START      ; Jump to user program address.
                                ...
START:                          ORG      1AH          ; 001AH, The head of user program.
                                ...                  ; User program
                                ...
                                ENDP                ; End of program
```

☞ **Note:** The head of user program should skip interrupt vector area to avoid program execution error.

## 2.2.1 INTERRUPT VECTOR (0008H~001CH)

A 11-word vector address area is used to execute interrupt request. If any interrupt service executes, the program counter (PC) value is stored in stack buffer and jump to 0008h~001Ch of program memory to execute the vectored interrupt. This interrupt is multi-vector and each of interrupts points to unique vector. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

**Note:** The “PUSH” and “POP” operations aren’t through instruction (PUSH, POP) and can executed save and load ACC and working registers (0x80~0x8F) by hardware automatically.

	ROM	Priority
0008H	INT0 Interrupt vector	1
000AH	INT1 Interrupt vector	2
000CH	T0 Interrupt vector	3
000EH	TC0 Interrupt vector	4
0010H	TC1 Interrupt vector	5
0012H	ADC Interrupt vector	6
0014H	TX Interrupt vector	7
0016H	RX Interrupt vector	8
001AH	TX2 Interrupt vector	9
001CH	RX2 Interrupt vector	10
0018H	MSP Interrupt vector	11

## 2.2.2 INTERRUPT PRIORITY

When one interrupt request occurs, and the program counter points to the correlative vector to execute interrupt service routine. If INT0 interrupt occurs, the program counter points to ORG 8. If INT1 interrupt occurs, the program counter points to ORG A. In normal condition, several interrupt requests happen at the same time. So the priority of interrupt sources is very important, or the system doesn’t know which interrupt is processed first. The interrupt priority is follow vector sequence. ORG 8 is priority 1. ORG A is priority 2. In the case, the interrupt processing priority is as following.

If INT0, T0, TC1 and MSP interrupt requests happen at the same time, the system processing interrupt sequence is INT0, T0, TC1 and then MSP. The system processes INT0 interrupt service routine first, and then processes T0 interrupt routine...Until finishing processing all interrupt requests.

### Example:

Interrupt Request Occurrence Sequence: (2~8 interrupt requests occur during RX interrupt service routine execution.)

1	2	3	4	5	6	7	8
UART RX	INT0	MSP	INT1	T0	TC0	TC1	ADC

Interrupt Processing Sequence:

1	2	3	4	5	6	7	8
UART RX	INT0	INT1	T0	TC0	TC1	ADC	MSP

**Example: Defining Interrupt Vector.** The interrupt service routine is following user program.

```
.CODE
    ORG      0          ; 0000H
    JMP      START      ; Jump to user program address.
    ...
    ORG      8          ; Interrupt vector, 0008H.
    JMP      ISR_INT0    ; Jump to interrupt service routine address.
    ORG      A
    JMP      ISR_INT1
    ORG      C
    JMP      ISR_T0
    ORG      E
    JMP      ISR_TC0
    ORG      10
    JMP      ISR_TC1
    ORG      12
    JMP      ISR_ADC
    ORG      14
    JMP      ISR_TX
    ORG      16
    JMP      ISR_RX
    ORG      18
    JMP      ISR_MSP
    ORG      1EH


START:
    ...                ; 001AH, The head of user program.
    ...                ; User program.
    JMP      START      ; End of user program.
    ...

ISR_INT0:
    ...                ; The head of interrupt service routine.
    ...                ; Save ACC and 0x80~0x8F register to buffers.
    ...
    ...                ; Load ACC and 0x80~0x8F register from buffers.
    RETI              ; End of interrupt service routine.

ISR_INT1:
    ...                ;
    ...                ; Save ACC and 0x80~0x8F register to buffers.
    ...
    ...                ; Load ACC and 0x80~0x8F register from buffers.
    RETI              ; End of interrupt service routine.
    ...
    ...
    ...

ISR_RX:
    ...                ;
    ...                ; Save ACC and 0x80~0x8F register to buffers.
    ...
    ...                ; Load ACC and 0x80~0x8F register from buffers.
    RETI              ; End of interrupt service routine.

    ENDP              ; End of program.
```

 **Note:** It is easy to understand the rules of SONiX program from demo programs given above. These points are as following:  
 The address 0000H is a “JMP” instruction to make the program starts from the beginning.  
 The address 0008H~001CH is interrupt vector.  
 User's program is a loop routine for main purpose application.

### 2.2.3 LOOK-UP TABLE DECRSIPTION

In the ROM's data lookup function, Y register is pointed to middle byte address (bit 8~bit 15) and Z register is pointed to low byte address (bit 0~bit 7) of ROM. After MOVC instruction executed, the low-byte data will be stored in ACC and high-byte data stored in R register.

**Example: To look up the ROM data located "TABLE1".**

```

B0MOV    Y, #TABLE1$M    ; To set lookup table1's middle address
B0MOV    Z, #TABLE1$L    ; To set lookup table1's low address.
MOVC     ; To lookup data, R = 00H, ACC = 35H

                ; Increment the index address for next address.
INCMS     Z              ; Z+1
JMP      @F             ; Z is not overflow.
INCMS     Y              ; Z overflow (FFH → 00), → Y=Y+1
NOP

@@:       MOVC           ; To lookup data, R = 51H, ACC = 05H.
...
TABLE1:   DW             ; To define a word (16 bits) data.
          DW             0035H
          DW             5105H
          DW             2012H
          ...

```

**Note:** The Y register will not increase automatically when Z register crosses boundary from 0xFF to 0x00. Therefore, user must be take care such situation to avoid look-up table errors. If Z register is overflow, Y register must be added one. The following INC\_YZ macro shows a simple method to process Y and Z registers automatically.

➤ **Example: INC\_YZ macro.**

```

INC_YZ     MACRO
            INCMS     Z          ; Z+1
            JMP      @F         ; Not overflow

            INCMS     Y          ; Y+1
            NOP        ; Not overflow

@@:
            ENDM

```



**Example: Modify above example by “INC\_YZ” macro.**

```

B0MOV    Y, #TABLE1$M    ; To set lookup table1's middle address
B0MOV    Z, #TABLE1$L    ; To set lookup table1's low address.
MOVC                                           ; To lookup data, R = 00H, ACC = 35H

    INC_YZ                                ; Increment the index address for next address.
;
@@:      MOVC                                ; To lookup data, R = 51H, ACC = 05H.
;
TABLE1:  ...                                ;
          DW      0035H                    ; To define a word (16 bits) data.
          DW      5105H
          DW      2012H
          ...

```

The other example of look-up table is to add Y or Z index register by accumulator. Please be careful if “carry” happen.

**Example: Increase Y and Z register by B0ADD/ADD instruction.**

```

B0MOV    Y, #TABLE1$M    ; To set lookup table's middle address.
B0MOV    Z, #TABLE1$L    ; To set lookup table's low address.

B0MOV    A, BUF          ; Z = Z + BUF.
B0ADD    Z, A

B0BTS1   FC              ; Check the carry flag.
JMP      GETDATA         ; FC = 0
INCMS    Y               ; FC = 1. Y+1.
NOP

GETDATA: ;
;
          MOVC            ; To lookup data. If BUF = 0, data is 0x0035
; If BUF = 1, data is 0x5105
; If BUF = 2, data is 0x2012
          ...

TABLE1:  DW      0035H    ; To define a word (16 bits) data.
          DW      5105H
          DW      2012H
          ...

```

## 2.2.4 JUMP TABLE DESCRIPTION

The jump table operation is one of multi-address jumping function. Add low-byte program counter (PCL) and ACC value to get one new PCL. If PCL is overflow after PCL+ACC, PCH adds one automatically. The new program counter (PC) points to a series jump instructions as a listing table. It is easy to make a multi-jump program depends on the value of the accumulator (A).

\* **Note:** PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL-ACC, PCH keeps value and not change.

Example: Jump table.

```

ORG      0X0100      ; The jump table is from the head of the ROM boundary

B0ADD    PCL, A       ; PCL = PCL + ACC, PCH + 1 when PCL overflow occurs.
JMP      A0POINT     ; ACC = 0, jump to A0POINT
JMP      A1POINT     ; ACC = 1, jump to A1POINT
JMP      A2POINT     ; ACC = 2, jump to A2POINT
JMP      A3POINT     ; ACC = 3, jump to A3POINT

```

SONiX provides a macro for safe jump table function. This macro will check the ROM boundary and move the jump table to the right position automatically. The side effect of this macro maybe wastes some ROM size.

➤ **Example:** If “jump table” crosses over ROM boundary will cause errors.

```

@JMP_A    MACRO      VAL
IF        (($+1) !& 0XFF00) != (($+(VAL)) !& 0XFF00)
JMP       ($ | 0XFF)
ORG       ($ | 0XFF)
ENDIF
B0ADD     PCL, A
ENDM

```

\* **Note:** “VAL” is the number of the jump table listing number.

Example: “@JMP\_A” application in SONiX macro file called “MACRO3.H”.

```

B0MOV     A, BUF0     ; “BUF0” is from 0 to 4.
@JMP_A    5           ; The number of the jump table listing is five.
RLCM      ; JMP cost two program count, so add RLCM instruction
JMP       A0POINT     ; ACC = 0, jump to A0POINT
JMP       A1POINT     ; ACC = 1, jump to A1POINT
JMP       A2POINT     ; ACC = 2, jump to A2POINT
JMP       A3POINT     ; ACC = 3, jump to A3POINT
JMP       A4POINT     ; ACC = 4, jump to A4POINT

```

If the jump table position is across a ROM boundary (0x00FF~0x0100), the “@JMP\_A” macro will adjust the jump table routine begin from next RAM boundary (0x0100).

**Example: “@JMP\_A” operation.**

**; Before compiling program.**

ROM address	B0MOV	A, BUF0	; “BUF0” is from 0 to 4.
	@JMP_A	5	; The number of the jump table listing is five.
	RLCM		; JMP cost two program count, so add RLCM instruction
0X00FD	JMP	A0POINT	; ACC = 0, jump to A0POINT
0X00FE	JMP	A1POINT	; ACC = 1, jump to A1POINT
0X00FF	JMP	A2POINT	; ACC = 2, jump to A2POINT
0X0100	JMP	A3POINT	; ACC = 3, jump to A3POINT
0X0101	JMP	A4POINT	; ACC = 4, jump to A4POINT

**; After compiling program.**

ROM address	B0MOV	A, BUF0	; “BUF0” is from 0 to 4.
	@JMP_A	5	; The number of the jump table listing is five.
	RLCM		; JMP cost two program count, so add RLCM instruction
0X0100	JMP	A0POINT	; ACC = 0, jump to A0POINT
0X0101	JMP	A1POINT	; ACC = 1, jump to A1POINT
0X0102	JMP	A2POINT	; ACC = 2, jump to A2POINT
0X0103	JMP	A3POINT	; ACC = 3, jump to A3POINT
0X0104	JMP	A4POINT	; ACC = 4, jump to A4POINT

## 2.2.5 CHECKSUM CALCULATION

The last ROM address are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

**Example: The demo program shows how to calculated Checksum from 00H to the end of user's code.**

```

MOV      A,#END_USER_CODE$L
B0MOV    END_ADDR1, A      ; Save low end address to end_addr1
MOV      A,#END_USER_CODE$M
B0MOV    END_ADDR2, A      ; Save middle end address to end_addr2
CLR      Y                  ; Set Y to 00H
CLR      Z                  ; Set Z to 00H

@@:
MOVC
B0BCLR   FC                ; Clear C flag
ADD      DATA1, A         ; Add A to Data1
MOV      A, R
ADC      DATA2, A         ; Add R to Data2
JMP      END_CHECK         ; Check if the YZ address = the end of code

AAA:
INCMS    Z                  ; Z=Z+1
JMP      @B                 ; If Z != 00H calculate to next address
JMP      Y_ADD_1            ; If Z = 00H increase Y

END_CHECK:
MOV      A, END_ADDR1
CMPSR    A, Z               ; Check if Z = low end address
JMP      AAA                ; If Not jump to checksum calculate
MOV      A, END_ADDR2
CMPSR    A, Y               ; If Yes, check if Y = middle end address
JMP      AAA                ; If Not jump to checksum calculate
JMP      CHECKSUM_END       ; If Yes checksum calculated is done.

Y_ADD_1:
INCMS    Y                  ; Increase Y
NOP
JMP      @B                 ; Jump to checksum calculate

CHECKSUM_END:
...
...

END_USER_CODE:              ; Label of program end

```

## 2.2.6 DATA MEMORY (RAM)

**6144 X 8-bit RAM**

Bank	Address	RAM Location	Comment
<b>Bank 0</b>	000H	General purpose area	RAM Bank 0
	...		
	07FH		
	080H	System Register	
	...		
	0FFH		End of Bank 0
<b>Bank 1</b>	100H	General purpose area	RAM Bank 1
	...		
<b>Bank 2</b>	1FFH	General purpose area	End of Bank 1
	200H		RAM Bank 2
<b>Bank 3</b>	...	General purpose area	
	2FFH		End of Bank 2
<b>Bank 3</b>	300H	General purpose area	RAM Bank 3
	...		
	3FFH		End of Bank 3
<b>Bank 24</b>	.....	General purpose area	
	1800H		RAM Bank 24
	...		
<b>Bank31</b>	187FH	LCD RAM area	End of Bank 24
	1F00H		RAM Bank 31
	1F1FH		End of Bank 31

The 6144-byte general purpose RAM is separated into Bank0, Bank1, Bank2, Bank3 to Bank24 and LCD Ram Bank31. Accessing the three banks' RAM is controlled by "RBANK" register. When RBANK = 0, the program controls Bank 0 RAM directly. When RBANK = 1, the program controls Bank 1 RAM directly. When RBANK = 2, the program controls Bank 2 RAM directly. Under one bank condition and need to access the other bank RAM, setup the RBANK register is necessary. When interrupt occurs, RBANK register is saved, and RAM bank is still last condition. User can select RAM bank through setup RBANK register during processing interrupt service routine. When RETI is executed to leave interrupt operation, RBANK register is reloaded, and RAM bank returns to last condition. Sonix provides "Bank 0" type instructions (e.g. b0mov, b0add, b0bts1, b0bset...) to control Bank 0 RAM in non-zero RAM bank condition directly.

- Example: Access Bank 0 RAM in Bank 1 condition. Move Bank 0 RAM (WK00) value to Bank 1 RAM (WK01).

; Bank 1 (RBANK = 1)

B0MOV A, WK00  
MOV WK01, A

; Use Bank 0 type instruction to access Bank 0 RAM.

☞ **Note:** For multi-bank RAM program, it is not easy to control RAM Bank selection. Users have to take care the RBANK condition very carefully, especially for interrupt service routine. The system won't save the RBANK and switch RAM bank to Bank 0, so these controls must be through program. It is a good to use Bank 0 type instruction to process the situations.

## 2.2.7 SYSTEM REGISTER

**System Register table**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7												URTX2	URRX2	URCR2	UTX2D	URX2D
8	L	H	R	Z	Y	X	PFLAG	RBANK	W0	W1	W2	W3	W4	W5	W6	W7
9	VREG	CHS	ADCM1	ADCM2	ADCM3	LBTM	LEDM	DAM	DABH	DABL	OPM1	OPM2	ADCDH	ADCDM	ADCDL	AMPM
A	P0M	P1M	P3M	P5M	P0UR	P1UR	P3UR	P5UR	P0	P1	P3	P5	P1W	LCDM1	LCDM2	P2
B	INTEN0	INTEN1	INTRQ0	INTRQ1	P2UR	P2M	OSCM	WDTR	PCH	PCL	STKP	PEDGE	@HL	@YZ	SEC	MIN
C	T0M	T0C	TC0M	TC0RL	TC0RH	TC0CL	TC0CH	TC0DL	TC0DH	TC1M	TC1RL	TC1RH	TC1CL	TC1CH	TC1DL	TC1DH
D	MSPSTAT	MSPM1	MSPM2	MSPBUF	MSPADR	URTX	URRX	URCR	UTXD	URXD	PECMD	PEROML	PEROMH	PERAML	PERAMH	PERAMCNT
E	STK0L	STK0H	STK1L	STK1H	STK2L	STK2H	STK3L	STK3H	STK4L	STK4H	STK5L	STK5H	STK6L	STK6H	STK7L	STK7H
F	STK8L	STK8H	STK9L	STK9H	STK10L	STK10H	STK11L	STK11H	STK12L	STK12H	STK13L	STK13H	STK14L	STK14H	STK15L	STK15H

## SYSTEM REGISTER DESCRIPTION

H, L = Working, @HL addressing register.

R = Working register and ROM look-up data buffer.

X = Working and ROM address register

RBANK = RAM bank selection register.

VREG= Voltage Regulators control register

ADC1= ADC control register 1

ADC2= ADC control register 2

ADC3= ADC control register 3

LBTM= Low Battery control register

CHS= ADC input channel selection Register

DAM= DAC control register

INTEN0,1 = Interrupt enable register.

WDTR = Watchdog timer clear register.

OSCM = Oscillator mode register.

T0M = T0 mode register.

T0C = T0 counting register.

INTRQ0,1 = Interrupt request register.

TCnRL= TCN timer counter reload buffer low byte

TCnRH= TCN timer counter reload buffer high byte

TCnCL= TCN timer counter buffer low byte

TCnCH= TCN timer counter buffer high byte

TCnDL= TCn duty low byte control register.

TCnDH= TCn duty high byte control register.

MSPBUF= MSP buffer register.

MSPADR= MSP address register.

MSPSTAT= MSP status register

PECMD= ISP command register.

PEROM= ISP ROM address

@HL = RAM HL indirect addressing index pointer.

Y, Z = Working, @YZ and ROM addressing register.

PFLAG = Special flag register.

W0~W7= Working register

AMPM PGIA control register

ADCDH/M/L ADC Data register

OPM1/2 OP-amp control register and switch control register

DABH/L DAC Data register

P1W = Port 1 wakeup register.

LEDM= LED control register

PEDGE = P0.0 edge direction register.

LCDM1/2= LCD control register

Pn= Port N data buffer

PnM= Port N input/output mode register

PnUR = Port N pull-up resistor control register.

PCH, PCL = Program counter.

STKP = Stack pointer buffer.

TCnM= TCn mode register.

SEC= T0 sec counter

MIN= T0 min counter

URXD = UART receive data buffer.

UTXD = UART transmit data buffer.

URTX = UART transmit control register.

URRX = UART receive control register.

URCR = UART baud rate control register.

MSPM1= MSP mode register1

MSPM2= MSP mode register2

PERAM= ISP RAM mapping address

PERAMCNT= ISP RAM programming counter register.

@YZ = RAM YZ indirect addressing index pointer.

STK0~STK15 = Stack 0 ~ stack 15 buffer.

URX2D = UART2 receive data buffer.

UTX2D = UART2 transmit data buffer.

URTX2 = UART2 transmit control register.

URRX2 = UART2 receive control register.

URCR2 = UART2 baud rate control register.

## 2.2.8 SYSTEM REGISTER DEFINITION

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Name
07BH	UTX2EN	UTXP2EN	UTX2PS	UTX2BRK	URX2BZ	UTX2BZ	-	-	R/W	URTX2
07CH	URX2EN	URXP2EN	URX2PS	URX2PC	UFMER1	UR2S2	URS1	UR2S0	R/W	URRX2
07DH	UR2CR7	UR2CR6	UR2CR5	UR2CR4	UR2CR3	UR2CR2	UR2CR1	UR2CR0	R/W	URCR2
07EH	UTX2D7	UTX2D6	UTX2D5	UTX2D4	UTX2D3	UTX2D2	UTX2D1	UTX2D0	R/W	UTX2D
07FH	URX2D7	URX2D6	URX2D5	URX2D4	URX2D3	URX2D2	URX2D1	URX2D0	R/W	URX2D
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZISPBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	RBNKS4	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
088H	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0	R/W	W0
089H	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0	R/W	W1
08AH	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0	R/W	W2
08BH	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0	R/W	W3
08CH	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0	R/W	W4
08DH	W5BIT7	W5BIT6	W5BIT5	W5BIT4	W5BIT3	W5BIT2	W5BIT1	W5BIT0	R/W	W5
08EH	W6BIT7	W6BIT6	W6BIT5	W6BIT4	W6BIT3	W6BIT2	W6BIT1	W6BIT0	R/W	W6
08FH	W7BIT7	W7BIT6	W7BIT5	W7BIT4	W7BIT3	W7BIT2	W7BIT1	W7BIT0	R/W	W7
090H	BGEN	ACMEN	AVEN	AVDDREN	AVDDRS1	AVDDRS0	-	VREFS	R/W	VREG
091H	MUXP3	MUXP2	MUXP1	MUXP0	MUXN3	MUXN2	MUXN1	MUXN0	R/W	CHS
092H	OSR2	OSR1	OSR0	RVS	IRVS2	IRVS1	IRVS0	ADCEN	R/W	ADCM1
093H	UGBEH	VDTEN	-	GX	GR	ADGN1	ADGN0	DRDY	R/W	ADCM2
094H	CPCKEN	CPCSK1	CPCSK0	-	ADCKINV	ADCKS2	ADCKS1	ADCKS0	R/W	ADCM3
095H	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTEN	R/W	LBTM
096H	-	LEDIO	LEDPW1	LEDPW0	LED4EN	LED3EN	LED2EN	LED1EN	R/W	LEDM
097H	DAEN	-	-	-	-	DAF2	DAF1	DAF0	R/W	DAM
098H	-	-	-	-	DAB11	DAB10	DAB9	DAB8	R/W	DABH
099H	DAB7	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	R/W	DABL
09AH	SW5	SW4	SW3	SW2	SW1	SW0	OP2EN	OP1EN	R/W	OPM1
09BH	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	R/W	OPM2
09CH	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16	R/W	ADCDH
09DH	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R/W	ADCDM
09EH	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R/W	ADCDL
09FH	BGRCMP0	BGRCHPEN	SW14	GS2	GS1	GS0	PCHPEN	AMPEN	R/W	AMPM
0A0H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0A1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0A2H	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0A3H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0A4H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0A5H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0A6H	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R	W	P3UR
0A7H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0A8H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0A9H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0AAH	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0ABH	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0ACH	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	R/W	P1W
0ADH	LCDPENB	P3SEG	LCDBNK	-	LCDMODE	LCDENB	LCDRATE	-	R/W	LCDM1
0AEH	VAR1	VAR0	BGM	DUTY1	DUTY0	VCP2	VCP 1	VCP0	R/W	LCDM2
0AFH	-	-	-	-	P23	P22	P21	P20	R/W	P2
0B0H	ADCIEN	TC1IEN	TC0IEN	T0IEN	URXIEN	UTXIEN	P01IEN	P00IEN	R/W	INTEN0
0B1H	-	-	-	-	URX2IEN	UTX2IEN	T0MOD	MSPIEN	R/W	INTEN1
0B2H	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	URXIRQ	UTXIRQ	P01IRQ	P00IRQ	R/W	INTRQ0
0B3H	-	-	-	-	URX2IRQ	UTX2IRQ	-	MSPIRQ	R/W	INTRQ1
0B4H	-	-	-	-	P23R	P22R	P21R	P20R	W	P2UR
0B5H	-	-	-	-	P23M	P22M	P21M	P20M	R/W	P2M
0B6H	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0B7H	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR



0B8H	-	PC14	PC13	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0B9H	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0BAH	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0	R/W	STKP
0BBH	-	-	-	-	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0BCH	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0BDH	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0BEH	-	-	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0	R/W	SEC
0BFH	-	-	MIN5	MIN4	MIN3	MIN2	MIN1	MIN0	R/W	MIN
0C0H	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB	R/W	T0M
0C1H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0C2H	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS1	TC0CKS0	PWM0OUT	-	R/W	TC0M
0C3H	TC0RL7	TC0RL6	TC0RL5	TC0RL4	TC0RL3	TC0RL2	TC0RL1	TC0RL0	W	TC0RL
0C4H	TC0RH7	TC0RH6	TC0RH5	TC0RH4	TC0RH3	TC0RH2	TC0RH1	TC0RH0	W	TC0RH
0C5H	TC0CL7	TC0CL6	TC0CL5	TC0CL4	TC0CL3	TC0CL2	TC0CL1	TC0CL0	R/W	TC0CL
0C6H	TC0CH7	TC0CH6	TC0CH5	TC0CH4	TC0CH3	TC0CH2	TC0CH1	TC0CH0	R/W	TC0CH
0C7H	TC0DL7	TC0DL6	TC0DL5	TC0DL4	TC0DL3	TC0DL2	TC0DL1	TC0DL0	R/W	TC0DL
0C8H	TC0DH7	TC0DH6	TC0DH5	TC0DH4	TC0DH3	TC0DH2	TC0DH1	TC0DH0	R/W	TC0DH
0C9H	TC1ENB	TC1rate2	TC1rate1	TC10rate0	TC1CKS1	TC1CKS0	PWM1OUT	-	R/W	TC1M
0CAH	TC1RL7	TC1RL6	TC1RL5	TC1RL4	TC1RL3	TC1RL2	TC1RL1	TC1RL0	W	TC1RL
0CBH	TC1RH7	TC1RH6	TC1RH5	TC1RH4	TC1RH3	TC1RH2	TC1RH1	TC1RH0	W	TC1RH
0CCH	TC1CL7	TC1CL6	TC1CL5	TC1CL4	TC1CL3	TC1CL2	TC1CL1	TC1CL0	R/W	TC1CL
0CDH	TC1CH7	TC1CH6	TC1CH5	TC1CH4	TC1CH3	TC1CH2	TC1CH1	TC1CH0	R/W	TC1CH
0CEH	TC1DL7	TC1DL6	TC1DL5	TC1DL4	TC1DL3	TC1DL2	TC1DL1	TC1DL0	R/W	TC1DL
0CFH	TC1DH7	TC1DH6	TC1DH5	TC1DH4	TC1DH3	TC1DH2	TC1DH1	TC1DH0	R/W	TC1DH
0D0H	-	CKE	D_A	P	S	RED_WRT	-	BF	R/W	MSPSTAT
0D1H	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC	R/W	MSPM1
0D2H	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	R/W	MSPM2
0D3H	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
0D4H	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
0D5H	UTXEN	UTXPEN	UTXPS	UTXBRK	URXBZ	UTXBZ	-	-	R/W	URTX
0D6H	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0	R/W	URRX
0D7H	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0	R/W	URCR
0D8H	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0	R/W	UTXD
0D9H	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0	R/W	URXD
0DAH	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0	R/W	PECMD
0DBH	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0	R/W	PEROML
0DCH	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0	R/W	PEROMH
0DDH	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0	R/W	PERAML
0DEH				PERAMH4	PERAMH3	PERAMH2	PERAMH1	PERAMH0	R/W	PERAMH
0DFH	PERAMCNT4	PERAMCNT3	PERAMCNT2	PERAMCNT1	PERAMCNT0	-	-	-	R/W	PERAMCNT
0E0H	S8PC7	S8PC6	S8PC5	S8PC4	S8PC3	S8PC2	S8PC1	S8PC0	R/W	STK8L
0E1H	-	S8PC14	S8PC13	S8PC12	S8PC11	S8PC10	S8PC9	S8PC8	R/W	STK8H
0E2H	S9PC7	S9PC6	S9PC5	S9PC4	S9PC3	S9PC2	S9PC1	S9PC0	R/W	STK9L
0E3H	-	S9PC14	S9PC13	S9PC12	S9PC11	S9PC10	S9PC9	S9PC8	R/W	STK9H
0E4H	S10PC7	S10PC6	S10PC5	S10PC4	S10PC3	S10PC2	S10PC1	S10PC0	R/W	STK10L
0E5H	-	S10PC14	S10PC13	S10PC12	S10PC11	S10PC10	S10PC9	S10PC8	R/W	STK10H
0E6H	S11PC7	S11PC6	S11PC5	S11PC4	S11PC3	S11PC2	S11PC1	S11PC0	R/W	STK11L
0E7H	-	S11PC14	S11PC13	S11PC12	S11PC11	S11PC10	S11PC9	S11PC8	R/W	STK11H
0E8H	S12PC7	S12PC6	S12PC5	S12PC4	S12PC3	S12PC2	S12PC1	S12PC0	R/W	STK12L
0E9H	-	S12PC14	S12PC13	S12PC12	S12PC11	S12PC10	S12PC9	S12PC8	R/W	STK12H
0EAH	S13PC7	S13PC6	S13PC5	S13PC4	S13PC3	S13PC2	S13PC1	S13PC0	R/W	STK13L
0EBH	-	S13PC14	S13PC13	S13PC12	S13PC11	S13PC10	S13PC9	S13PC8	R/W	STK13H
0ECH	S14PC7	S14PC6	S14PC5	S14PC4	S14PC3	S14PC2	S14PC1	S14PC0	R/W	STK14L
0EDH	-	S14PC14	S14PC13	S14PC12	S14PC11	S14PC10	S14PC9	S14PC8	R/W	STK14H
0EEH	S15PC7	S15PC6	S15PC5	S15PC4	S15PC3	S15PC2	S15PC1	S15PC0	R/W	STK15L
0EFH	-	S15PC14	S15PC13	S15PC12	S15PC11	S15PC10	S15PC9	S15PC8	R/W	STK15H
0F0H	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0F1H	-	S0PC14	S0PC13	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

0F2H	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0F3H	-	S1PC14	S1PC13	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0F4H	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0F5H	-	S2PC14	S2PC13	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0F6H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F7H	-	S3PC14	S3PC13	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0F8H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F9H	-	S4PC14	S4PC13	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0FAH	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0FBH	-	S5PC14	S5PC13	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0FCH	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0FDH	-	S6PC14	S6PC13	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0FEH	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0FFH	-	S7PC14	S7PC13	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H

**Note:**

1. To avoid system error, make sure to put all the "0" and "1" as it indicates in the above table.
2. All of register names had been declared in SN8ASM assembler.
3. One-bit name had been declared in SN8ASM assembler with "F" prefix code.
4. "b0bset", "b0bclr", "bset", "bclr" instructions are only available to the "R/W" registers.

## 2.2.9 ACCUMULATOR

The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is zero (Z) or there is carry (C or DC) occurrence, then these flags will be set to PFLAG register. ACC is not in data memory (RAM), so ACC can't be access by "B0MOV" instruction during the instant addressing mode.

**Example: Read and write ACC value.**

; Read ACC data and store in BUF data memory

```
MOV      BUF, A
```

; Write a immediate data into ACC

```
MOV      A, #0FH
```

; Write ACC data from BUF data memory

```
MOV      A, BUF
```

The system will store ACC and working registers (0x80-0x8F) by hardware automatically when interrupt executed.

➤ **Example: Protect ACC and working registers.**

```
.CODE
INT_SERVICE:
```

```
; Save ACC to buffer.
; Save working registers to buffer.
```

```
...
...
```

```
; Load working registers form buffers.
; Load ACC form buffer.
```

```
RETI      ; Exit interrupt service vector
```

## 2.2.10 PROGRAM COUNTER

The program counter (PC) is a 16-bit binary counter separated into the high-byte 8 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 15.

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

### ☞ ONE ADDRESS SKIPPING

There are nine instructions (CMPRS, INCS, INCMS, DECS, DECMS, BTS0, BTS1, B0BTS0, B0BTS1) with one address skipping function. If the result of these instructions is true, the PC will add 2 steps to skip next instruction.

***If the condition of bit test instruction is true, the PC will add 2 steps to skip next instruction.***

```

        B0BTS1    FC          ; To skip, if Carry_flag = 1
        JMP      C0STEP      ; Else jump to C0STEP.
        ...
        ...
C0STEP:    NOP

        B0MOV     A, BUF0     ; Move BUF0 value to ACC.
        B0BTS0    FZ          ; To skip, if Zero flag = 0.
        JMP      C1STEP      ; Else jump to C1STEP.
        ...
        ...
C1STEP:    NOP

```

***If the ACC is equal to the immediate data or memory, the PC will add 2 steps to skip next instruction.***

```

        CMPRS     A, #12H     ; To skip, if ACC = 12H.
        JMP      C0STEP      ; Else jump to C0STEP.
        ...
        ...
C0STEP:    NOP

```

*If the destination increased by 1, which results overflow of 0xFF to 0x00, the PC will add 2 steps to skip next instruction.*

**INCS instruction:**

**INCS**      BUF0  
JMP      C0STEP      ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP:      NOP

**INCMS instruction:**

**INCMS**      BUF0  
JMP      C0STEP      ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP:      NOP

*If the destination decreased by 1, which results underflow of 0x01 to 0x00, the PC will add 2 steps to skip next instruction.*

**DECS instruction:**

**DECS**      BUF0  
JMP      C0STEP      ; Jump to C0STEP if ACC is not zero.

...

...

C0STEP:      NOP

**DECMS instruction:**

**DECMS**      BUF0  
JMP      C0STEP      ; Jump to C0STEP if BUF0 is not zero.

...

...

C0STEP:      NOP

## MULTI-ADDRESS JUMPING

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program Counter supports “**ADD M,A**”, “**ADC M,A**” and “**B0ADD M,A**” instructions for carry to PCH when PCL overflow automatically. For jump table or others applications, users can calculate PC value by the three instructions and don't care PCL overflow problem.

**Note:** PCH only support PC up counting result and doesn't support PC down counting. When PCL is carry after PCL+ACC, PCH adds one automatically. If PCL borrow after PCL-ACC, PCH keeps value and not change.

➤ **Example:** If PC = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
MOV      A, #28H
B0MOV    PCL, A          ; Jump to address 0328H
...

; PC = 0328H
MOV      A, #00H
B0MOV    PCL, A          ; Jump to address 0300H
...
```

**Example:** If PC = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
B0ADD    PCL, A          ; PCL = PCL + ACC, the PCH cannot be changed.
JMP      A0POINT        ; If ACC = 0, jump to A0POINT
JMP      A1POINT        ; ACC = 1, jump to A1POINT
JMP      A2POINT        ; ACC = 2, jump to A2POINT
JMP      A3POINT        ; ACC = 3, jump to A3POINT
...
...
```

## 2.2.11 H, L REGISTERS

The H and L registers are the 8-bit buffers. There are two major functions of these registers.

Can be used as general working registers

Can be used as RAM data pointers with @HL register

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

**Example:** If want to read a data from RAM address 20H of bank\_0, it can use indirectly addressing mode to access data as following.

```

B0MOV    H, #00H      ; To set RAM bank 0 for H register
B0MOV    L, #20H      ; To set location 20H for L register
B0MOV    A, @HL       ; To read a data into ACC

```

**Example: Clear general-purpose data memory area of bank 0 using @HL register.**

```

CLR      H             ; H = 0, bank 0
B0MOV    L, #07FH      ; L = 7FH, the last address of the data memory area

CLR_HL_BUF:
CLR      @HL           ; Clear @HL to be zero
DECMS    L             ; L - 1, if L = 0, finish the routine
JMP      CLR_HL_BUF    ; Not zero

CLR      @HL

END_CLR:                ; End of clear general purpose data memory area of bank 0
...

```

## 2.2.12 R REGISTER

R register is an 8-bit buffer. There are two major functions of the register.

Can be used as working register

For store high-byte data of look-up table

(MOVC instruction executed, the high-byte data of specified ROM address will be stored in R register and the low-byte data will be stored in ACC).

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>R</b>	<b>RBIT7</b>	<b>RBIT6</b>	<b>RBIT5</b>	<b>RBIT4</b>	<b>RBIT3</b>	<b>RBIT2</b>	<b>RBIT1</b>	<b>RBIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

**\* Note: Please refer to the “LOOK-UP TABLE DESCRIPTION” about R register look-up table application.**

## 2.2.13 Y, Z REGISTERS

The Y and Z registers are the 8-bit buffers. There are three major functions of these registers.

Can be used as general working registers

Can be used as RAM data pointers with @YZ register

Can be used as ROM data pointer with the MOVC instruction for look-up table

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>Y</b>	<b>YBIT7</b>	<b>YBIT6</b>	<b>YBIT5</b>	<b>YBIT4</b>	<b>YBIT3</b>	<b>YBIT2</b>	<b>YBIT1</b>	<b>YBIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>Z</b>	<b>ZBIT7</b>	<b>ZBIT6</b>	<b>ZBIT5</b>	<b>ZBIT4</b>	<b>ZBIT3</b>	<b>ZBIT2</b>	<b>ZBIT1</b>	<b>ZBIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

**Example: Uses Y, Z register as the data pointer to access data in the RAM address 025H of bank0.**

```

B0MOV    Y, #00H      ; To set RAM bank 0 for Y register
B0MOV    Z, #25H      ; To set location 25H for Z register
B0MOV    A, @YZ       ; To read a data into ACC

```

**Example: Uses the Y, Z register as data pointer to clear the RAM data.**

```

B0MOV    Y, #0           ; Y = 0, bank 0
B0MOV    Z, #07FH        ; Z = 7FH, the last address of the data memory area

```

CLR\_YZ\_BUF:

```

CLR      @YZ             ; Clear @YZ to be zero

DECMS    Z               ; Z – 1, if Z= 0, finish the routine
JMP      CLR_YZ_BUF      ; Not zero

```

```

CLR      @YZ
END_CLR:                               ; End of clear general purpose data memory area of bank 0
...

```

## 2.2.14 X REGISTERS

X register is an 8-bit buffer and only general working register purpose.  
Can be used as general working registers

085H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>X</b>	<b>XBIT7</b>	<b>XBIT6</b>	<b>XBIT5</b>	<b>XBIT4</b>	<b>XBIT3</b>	<b>XBIT2</b>	<b>XBIT1</b>	<b>XBIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

## 2.2.15 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. POR, WDT, and RST bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD33 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	-	-	-	-	-	C	DC	Z
Read/Write	-	-	-	-	-	R/W	R/W	R/W
After Reset	-	-	-	-	-	0	0	0

- Bit 2 **C**: Carry flag  
 1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic "1", comparison result  $\geq 0$ .  
 0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic "0", comparison result  $< 0$ .
- Bit 1 **DC**: Decimal carry flag  
 1 = Addition with carry from low nibble, subtraction without borrow from high nibble.  
 0 = Addition without carry from low nibble, subtraction with borrow from high nibble.
- Bit 0 **Z**: Zero flag  
 1 = The result of an arithmetic/logic/branch operation is zero.  
 0 = The result of an arithmetic/logic/branch operation is not zero.

 **Note:** Refer to instruction set table for detailed information of C, DC and Z flags.

## 2.2.16 W REGISTERS

W register includes W0~W7 8-bit buffers. There are two major functions of the register.

Can be used as general working registers in assembly language situation.

Can be used as program buffers in C-language situation.

088H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W0	W0BIT7	W0BIT6	W0BIT5	W0BIT4	W0BIT3	W0BIT2	W0BIT1	W0BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W1	W1BIT7	W1BIT6	W1BIT5	W1BIT4	W1BIT3	W1BIT2	W1BIT1	W1BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W2	W2BIT7	W2BIT6	W2BIT5	W2BIT4	W2BIT3	W2BIT2	W2BIT1	W2BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W3	W3BIT7	W3BIT6	W3BIT5	W3BIT4	W3BIT3	W3BIT2	W3BIT1	W3BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W4	W4BIT7	W4BIT6	W4BIT5	W4BIT4	W4BIT3	W4BIT2	W4BIT1	W4BIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
------	-------	-------	-------	-------	-------	-------	-------	-------



<b>W5</b>	<b>W5BIT7</b>	<b>W5BIT6</b>	<b>W5BIT5</b>	<b>W5BIT4</b>	<b>W5BIT3</b>	<b>W5BIT2</b>	<b>W5BIT1</b>	<b>W5BIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>W6</b>	<b>W6BIT7</b>	<b>W6BIT6</b>	<b>W6BIT5</b>	<b>W6BIT4</b>	<b>W6BIT3</b>	<b>W6BIT2</b>	<b>W6BIT1</b>	<b>W6BIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

08FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>W7</b>	<b>W7BIT7</b>	<b>W7BIT6</b>	<b>W7BIT5</b>	<b>W7BIT4</b>	<b>W7BIT3</b>	<b>W7BIT2</b>	<b>W7BIT1</b>	<b>W7BIT0</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

\* **Note:** In assembly language situation, W0~W7 can be used as general working registers.  
In C-language situation, W0~W7 are reserved for C-compiler, and recommend not to access W0~W7 by program strongly.

## 2.3 ADDRESSING MODE

### 2.3.1 IMMEDIATE ADDRESSING MODE

The immediate addressing mode uses an immediate data to set up the location in ACC or specific RAM.

**Example: Move the immediate data 12H to ACC.**

```
MOV      A, #12H      ; To set an immediate data 12H into ACC.
```

**Example: Move the immediate data 12H to R register.**

```
B0MOV    R, #12H      ; To set an immediate data 12H into R register.
```

**Note:** In immediate addressing mode application, the specific RAM must be 0x80~0x8F working register.

### 2.3.2 DIRECTLY ADDRESSING MODE

The directly addressing mode moves the content of RAM location in or out of ACC.

**Example: Move 0x12 RAM location data into ACC.**

```
B0MOV    A, 12H      ; To get a content of RAM location 0x12 of bank 0 and save in ACC.
```

**Example: Move ACC data into 0x12 RAM location.**

```
B0MOV    12H, A      ; To get a content of ACC and save in RAM location 12H of bank 0.
```

### 2.3.3 INDIRECTLY ADDRESSING MODE

The indirectly addressing mode is to access the memory by the data pointer registers (H/L, Y/Z).

**Example: Indirectly addressing mode with @HL register**

```
B0MOV    H, #0      ; To clear H register to access RAM bank 0.
B0MOV    L, #12H     ; To set an immediate data 12H into L register.
B0MOV    A, @HL      ; Use data pointer @HL reads a data from RAM location
                     ; 012H into ACC.
```

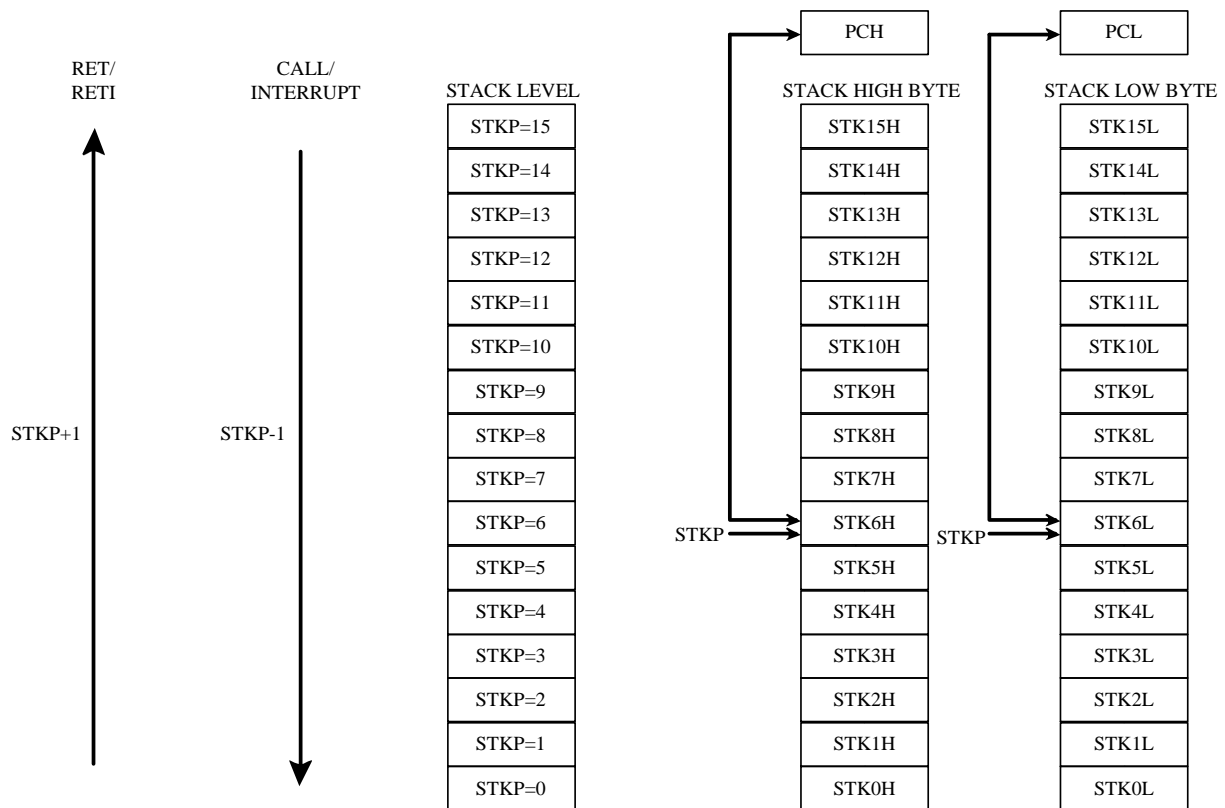
**Example: Indirectly addressing mode with @YZ register**

```
B0MOV    Y, #0      ; To clear Y register to access RAM bank 0.
B0MOV    Z, #12H     ; To set an immediate data 12H into Z register.
B0MOV    A, @YZ      ; Use data pointer @YZ reads a data from RAM location
                     ; 012H into ACC.
```

## 2.4 STACK OPERATION

### 2.4.1 OVERVIEW

The stack buffer has 16-level. These buffers are designed to push and pop up program counter's (PC) data when interrupt service routine and "CALL" instruction are executed. The STKP register is a pointer designed to point active level in order to push or pop up data from stack buffer. The STKnH and STKnL are the stack buffers to store program counter (PC) data.



### 2.4.2 STACK POINTER

The stack pointer (STKP) is a 4-bit register to store the address used to access the stack buffer, 15-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses. The two stack operations are writing to the top of the stack (push) and reading from the top of stack (pop). Push operation decrements the STKP and the pop operation increments each time. That makes the STKP always point to the top address of stack buffer and write the last program counter value (PC) into the stack buffer.

0EFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STKP</b>	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	1	1	1	1

Bit[3:0] **STKPBn**: Stack pointer (n = 0 ~ 3)

Bit 7 **GIE**: Global interrupt control bit.  
0 = Disable.  
1 = Enable. Please refer to the interrupt chapter.

➤ **Example: Stack pointer (STKP) reset, we strongly recommended to clear the stack pointers in the beginning of the program.**

```
MOV    A, #00000111B
B0MOV  STKP, A
```

## 2.4.3 STACK BUFFER

The program counter (PC) value is stored in the stack buffer before a CALL instruction executed or during interrupt service routine. Stack operation is a LIFO type (Last in and first out). The stack pointer (STKP) and stack buffer (STKnH and STKnL) are located in the system register area bank 0.

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	SnPC15	SnPC14	SnPC13	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

$STKn = STKnH, STKnL (n = 7 \sim 0)$

## 2.5 STACK OPERATION

### 2.5.1 OVERVIEW

The code option is the system hardware configurations including oscillator type, noise filter option, watchdog timer operation, LVD option, reset pin option and Flash ROM security control. The code option items are as following table:

Code Option	Content	Function Description
High_Clk	IHRC	High speed internal 8MHz RC. XIN/XOUT pins are bi-direction GPIO mode.
	IHRC_RTC	High speed internal 8MHz RC. XIN/XOUT pins are connected to external 32768Hz crystal.
High_Clk_Div	Fhosc/2	Normal mode instruction cycle is 2 high speed oscillator clocks.
	Fhosc/4	Normal mode instruction cycle is 4 high speed oscillator clocks.
	Fhosc/8	Normal mode instruction cycle is 8 high speed oscillator clocks.
NDT33/NDT18	Enable	Enable NDT function.
	Disable	Disable NDT function.
Hold time	2ms	NTD function option setting.
	4ms	
Filter	Enable	Enable 32K noise filter.
	Disable	Disable 32K noise filter.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
Security	Enable	Enable ROM code Security function.
	Disable	Disable ROM code Security function.

### 2.5.2 Fcpu Code Option

Fcpu means instruction cycle whose clock source includes high/low speed oscillator in different operating modes. High\_Fcpu and Low\_Fcpu code options select instruction cycle pre-scaler to decide instruction cycle rate. In normal mode (high speed clock), the system clock source is high speed oscillator, and Fcpu clock rate has eight options including Fhosc/2, Fhosc/4, Fhosc/8. In slow mode (low speed clock), the system clock source is internal low speed RC oscillator.

### 2.5.3 Security code option

Security code option is Flash ROM protection. When enable security code option, the ROM code is secured and not dumped complete ROM contents.

### 2.5.4 Noise Filter code option

Filter code option is a 32k RC noise filter. When enable filter code option, the filter to reduce noisy effect of system clock.

# 3 Reset

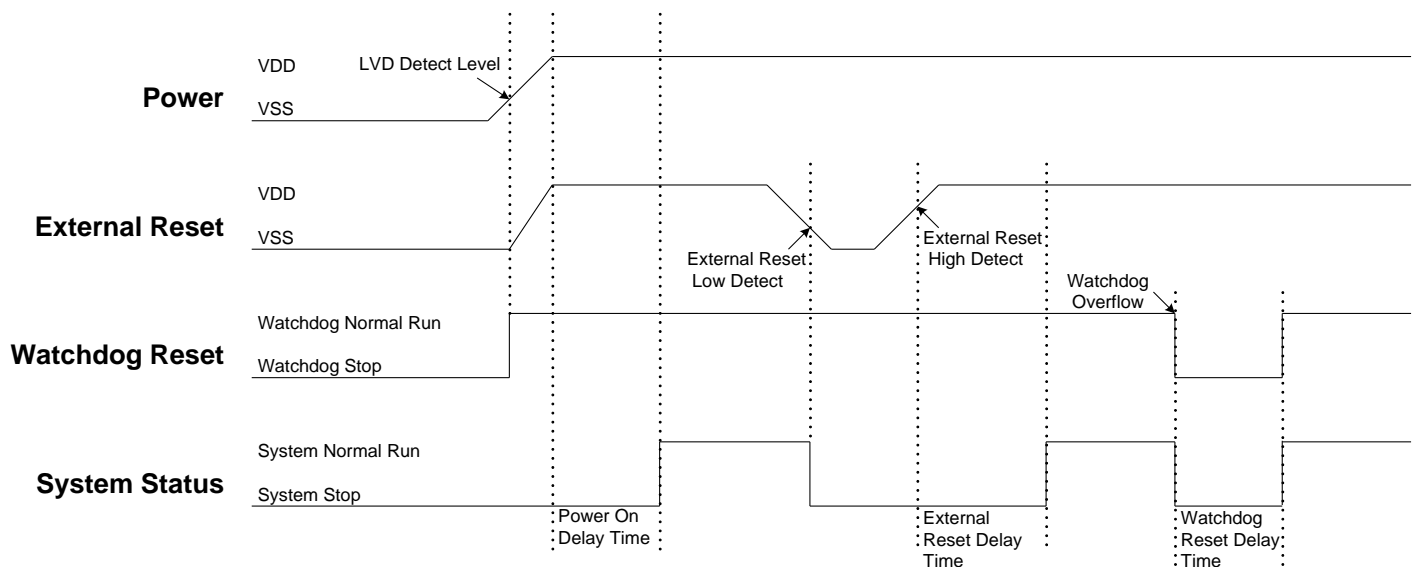
## 3.1 OVERVIEW

The system would be reset in three conditions as following.

- Power on reset
- Watchdog reset
- Brown out reset
- External reset (only supports external reset pin enable situation)

When any reset condition occurs, all system registers keep initial status, program stops and program counter is cleared. After reset status released, the system boots up and program starts to execute from ORG 0. The POR, WDT and RST flags indicate system reset status. [The system can depend on POR, WDT and RST status and go to different paths by program.](#)

Finishing any reset sequence needs some time. The system provides complete procedures to make the power on reset successful. For different oscillator types, the reset time is different. That causes the VDD rise rate and start-up time of different oscillator is not fixed. RC type oscillator's start-up time is very short, but the crystal type is longer. Under client terminal application, users have to take care the power on reset time for the master terminal requirement. The reset timing diagram is as following.



## 3.2 POWER ON RESET

The power on reset depend no LVD operation for most power-up situations. The power supplying to system is a rising curve and needs some time to achieve the normal voltage. Power on reset sequence is as following.

- **Power-up:** System detects the power voltage up and waits for power stable.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- **Program executing:** Power on sequence is finished and program executes from ORG 0.

## 3.3 WATCHDOG RESET

Watchdog reset is a system protection. In normal condition, system works well and clears watchdog timer by program. Under error condition, system is in unknown situation and watchdog can't be clear by program before watchdog timer overflow. Watchdog timer overflow occurs and the system is reset. After watchdog reset, the system restarts and returns normal mode. Watchdog reset sequence is as following.

- **Watchdog timer status:** System checks watchdog timer overflow status. If watchdog timer overflow occurs, the system is reset.
- **System initialization:** All system registers is set as initial conditions and system is ready.
- **Oscillator warm up:** Oscillator operation is successfully and supply to system clock.
- **Program executing:** Power on sequence is finished and program executes from ORG 0.

Watchdog timer application note is as following.

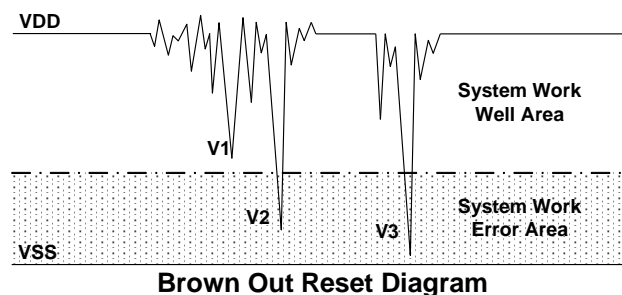
- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

\* **Note:** Please refer to the "WATCHDOG TIMER" about watchdog timer detail information.

## 3.4 BROWN OUT RESET

### 3.4.1 OVERVIEW

The brown out reset is a power dropping condition. The power drops from normal voltage to low voltage by external factors (e.g. EFT interference or external loading changed). The brown out reset would make the system not work well or executing program error.



The power dropping might through the voltage range that's the system dead-band. The dead-band means the power range can't offer the system minimum operation power requirement. The above diagram is a typical brown out reset diagram. There is a serious noise under the VDD, and VDD voltage drops very deep. There is a dotted line to separate the system working area. The above area is the system work well area. The below area is the system work error area called dead-band. V1 doesn't touch the below area and not effect the system operation. But the V2 and V3 is under the below area and may induce the system error occurrence. Let system under dead-band includes some conditions.

#### DC application:

The power source of DC application is usually using battery. When low battery condition and MCU drive any loading, the power drops and keeps in dead-band. Under the situation, the power won't drop deeper and not touch the system reset voltage. That makes the system under dead-band.

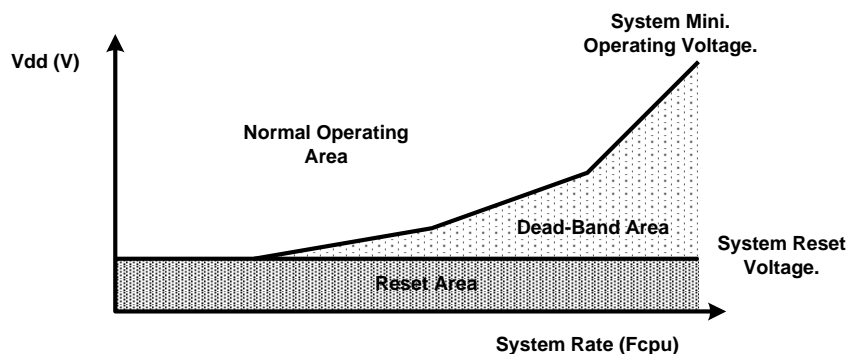
#### AC application:

In AC power application, the DC power is regulated from AC power source. This kind of power usually couples with AC noise that makes the DC power dirty. Or the external loading is very heavy, e.g. driving motor. The loading operating induces noise and overlaps with the DC power. VDD drops by the noise, and the system works under unstable power situation.

The power on duration and power down duration are longer in AC application. The system power on sequence protects the power on successful, but the power down situation is like DC low battery condition. When turn off the AC power, the VDD drops slowly and through the dead-band for a while.

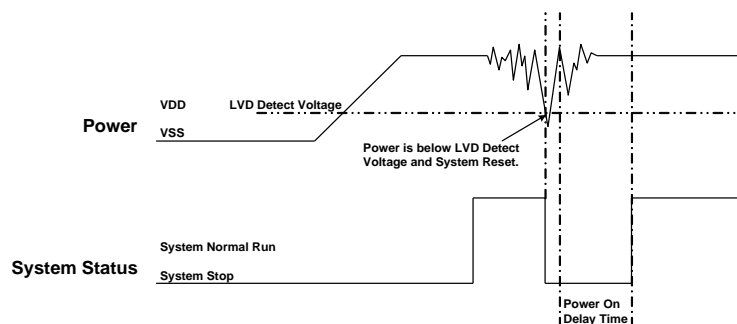
### 3.4.2 The system operation voltage

To improve the brown out reset needs to know the system minimum operating voltage which is depend on the system executing rate and power level. Different system executing rates have different system minimum operating voltage. The electrical characteristic section shows the system voltage to executing rate relationship.



Normally the system operation voltage area is higher than the system reset voltage to VDD, and the reset voltage is decided by LVD detect level. The system minimum operating voltage rises when the system executing rate upper even higher than system reset voltage. The dead-band definition is the system minimum operating voltage above the system reset voltage.

### 3.4.3 LOW VOLTAGE DETECTOR (LVD)



### **3.4.4 Watchdog reset:**

The watchdog timer is a protection to make sure the system executes well. Normally the watchdog timer would be clear at one point of program. Don't clear the watchdog timer in several addresses. The system executes normally and the watchdog won't reset system. When the system is under dead-band and the execution error, the watchdog timer can't be clear by program. The watchdog is continuously counting until overflow occurrence. The overflow signal of watchdog timer triggers the system to reset, and the system return to normal mode after reset sequence. This method also can improve brown out reset condition and make sure the system to return normal mode.

If the system reset by watchdog and the power is still in dead-band, the system reset sequence won't be successful and the system stays in reset status until the power return to normal range. Watchdog timer application note is as following.

#### **Reduce the system executing rate:**

If the system rate is fast and the dead-band exists, to reduce the system executing rate can improve the dead-band. The lower system rate is with lower minimum operating voltage. Select the power voltage that's no dead-band issue and find out the mapping system rate. Adjust the system rate to the value and the system exits the dead-band issue. This way needs to modify whole program timing to fit the application requirement.



# 4 SYSTEM CLOCK

## 4.1 OVERVIEW

The micro-controller is a dual clock system including high-speed and low-speed clocks. The high-speed clock includes internal high-speed oscillator and external oscillators selected by “High\_CLK” code option. The low-speed clock is from internal low-speed oscillator controlled by “CLKMD” bit of OSCM register. Both high-speed clock and low-speed clock can be system clock source through a divider to decide the system clock rate.

## 4.2 SYSTEM CLOCK

### **Clock Source:**

Fhosc: System high clock source is from internal high RC (IHRC) only.

Flosc: (1).System low clock source is from internal low RC (ILRC)

(2).System low clock source is from External 32768Hz Crystal when IHRC\_RTC selected in code option.

### **Instruction Cycle (Fcpu):**

Normal mode: Selected by High\_Clk\_Div code option from Fhosc/2, Fhosc/4 or Fhosc/8.

Slow mode: Flosc/4.

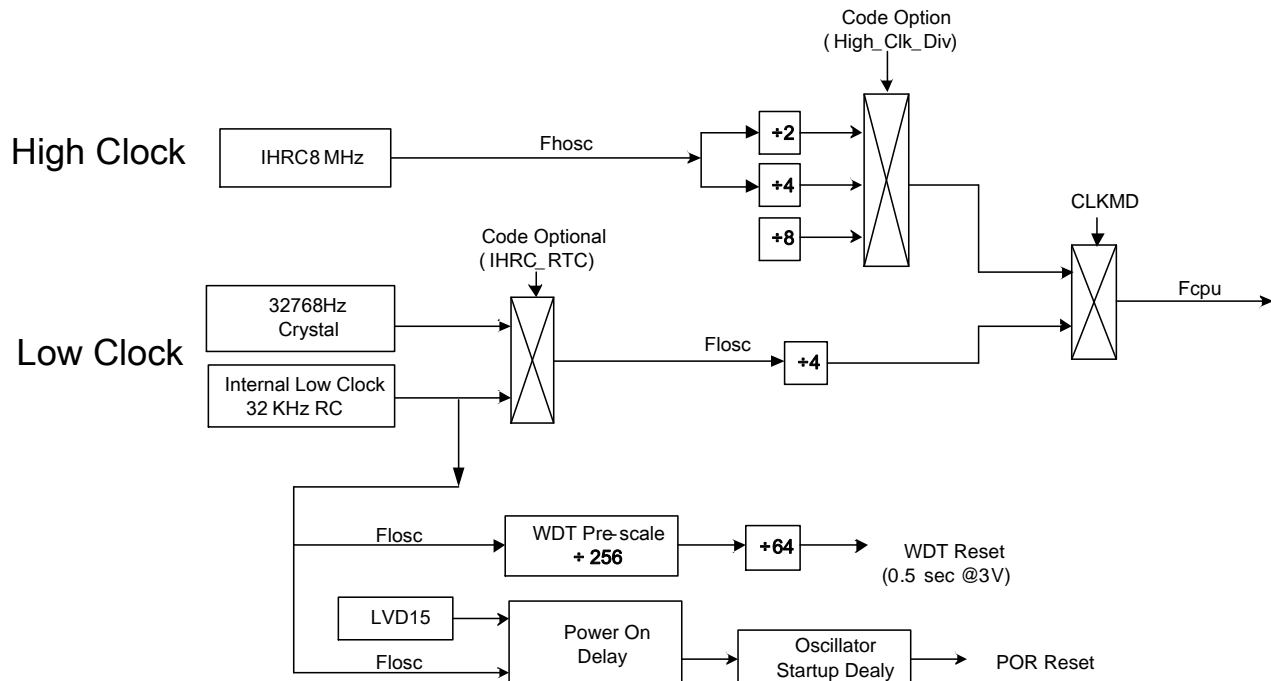
## 4.3 INTERNAL HIGH RC (IHRC)

Structure: 8MHz RC type.

Main Purpose: System high clock source.

Warm-up Time: Power on: 2048\*FIHRC. Wake-up from power down mode, green mode, slow mode: 32\*FIHRC.

## 4.4 SYSTEM CLOCK BLOCK DIAGRAM



## 4.5 OSCM REGISTER

The OSCM register is an oscillator control register. It controls oscillator status, system mode.

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>OSCM</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>CPUM1</b>	<b>CPUM0</b>	<b>CLKMD</b>	<b>STPHX</b>	<b>0</b>
Read/Write	-	-	-	R/W	R/W	R/W	R/W	-
After reset	-	-	-	0	0	0	0	-

- Bit 1     **STPHX**: External high-speed oscillator control bit.  
0 = External high-speed oscillator free run.  
1 = External high-speed oscillator free run stop. Internal low-speed RC oscillator is still running.
- Bit 2     **CLKMD**: System high/Low clock mode control bit.  
0 = Normal (dual) mode. System clock is high clock.  
1 = Slow mode. System clock is internal low clock.
- Bit[4:3]     **CPUM[1:0]**: CPU operating mode control bits.  
00 = normal.  
01 = sleep (power down) mode.  
10 = green mode.  
11 = reserved.

“STPHX” bit controls internal high speed RC type oscillator and external oscillator operations. When “STPHX=0”, the external oscillator or internal high speed RC type oscillator active. When “STPHX=1”, the external oscillator or internal high speed RC type oscillator are disabled. The STPHX function is depend on different high clock options to do different controls.

**IHRC\_RTC**: “STPHX=1” disables internal high speed RC type oscillator, and external 32768Hz crystal keeps oscillating.

## 4.6 SYSTEM CLOCK MEASUREMENT

Under design period, the users can measure system clock speed by software instruction cycle (Fcpu). This way is useful in RC mode.

**Example: Fcpu instruction cycle of external oscillator.**

	B0BSET	P0M.0	; Set P0.0 to be output mode for outputting Fcpu toggle signal.
@ @:			
	B0BSET	P0.0	; Output Fcpu toggle signal in low-speed clock mode.
	B0BCLR	P0.0	; Measure the Fcpu frequency by oscilloscope.
	JMP	@B	

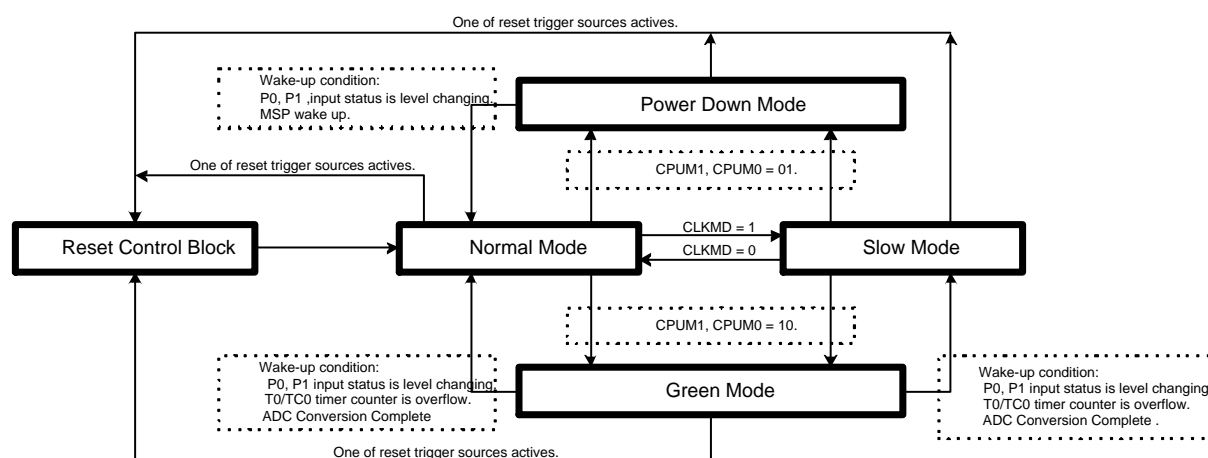
# 5 SYSTEM OPERATION MODE

## 5.1 OVERVIEW

The chip builds in four operating mode for difference clock rate and power saving reason. These modes control oscillators, op-code operation and analog peripheral devices' operation.

- Normal mode: System high-speed operating mode.
- Slow mode: System low-speed operating mode.
- Power down mode: System power saving mode (Sleep mode).
- Green mode: System ideal mode.

### Operating Mode Control Block



### Operating mode description

MODE	NORMAL	SLOW	GREEN	POWER DOWN	REMARK
Fhosc	Running	By STPHX	By STPHX	Stop	
Ffosc	Running	Running	Running	Stop	
CPU instruction	Executing	Executing	Stop	Stop	
T0 timer / TC0 timer / TC1 timer	* Active	*Active	*Active	Inactive	* Active if T0ENB=1 / TC0ENB/ TC1ENB
ADC	Active	*Active	*Active	Stop	*Active if high clock still running. (STPHX=0)
Watchdog timer	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	By Watch_Dog Code option	Refer to code option description
Internal interrupt	T0, TC0, TC1, ADC	T0, TC0, TC1, *ADC	T0, TC0, *ADC	All inactive	*Active if high clock still running. (STPHX=0)
External interrupt	P00, P01	P00, P01	P00, P01	P00, P01	
Wakeup source	-	-	P0, T0, TC0, *ADC **P1	P0, **P1, ***MSP	*Active if high clock still running. (STPHX=0) ** Active if (P1W = 0xFF) *** Active if address match.

## 5.2 SYSTEM MODE SWITCHING

- **Example: Switch normal/slow mode to power down (sleep) mode.**

```
B0BSET          FCPUM0          ; Set CPUM0 = 1.
```

\* **Note: During the sleep, only the wakeup pin and reset can wake up the system back to the normal mode.**

- Example: Switch normal mode to slow mode.**

```
B0BSET          FCLKMD          ;To set CLKMD = 1, Change the system into
                                slow mode
B0BSET          FSTPHX          ;To stop external high-speed oscillator for
                                power saving.
```

- **Example: Switch slow mode to normal mode (The IHRC oscillator is still running)**

```
B0BCLR          FCLKMD          ;To set CLKMD = 0
```

- Example: Switch slow mode to normal mode (The IHRC oscillator stops)**

If internal high clock stop and program want to switch back normal mode. It is necessary to delay at least 20ms for external clock stable.

```
                                B0BCLR          FSTPHX          ; Turn on the IHRC oscillator.
                                B0MOV          Z, #54          ; If VDD = 3.2V, ILRC =32KHz (typical) will
                                                                delay
@@:                            DECMS          Z              ; 0.125ms X 162 = 20.25ms for external
                                                                clock stable
                                JMP           @B
                                B0BCLR          FCLKMD          ; Change the system back to the normal
                                                                mode
```

- **Example: Switch normal/slow mode to green mode.**

```
B0BSET          FCPUM1          ; Set CPUM1 = 1.
```

\* **Note: If T0 timer wakeup function is disabled in the green mode, the wakeup pin P0 can wakeup the system backs to the previous operation mode.**

**Example: Switch normal/slow mode to Green mode and enable T0 wakeup function.**

; Set T0 timer wakeup function.

B0BCLR	FT0IEN	; To disable T0 interrupt service
B0BCLR	FT0ENB	; To disable T0 timer
MOV	A,#20H	;
B0MOV	T0M,A	; To set T0 clock = Fcpu / 64
MOV	A,#74H	;
B0MOV	T0C,A	; To set T0C initial value = 74H (To set T0 interval = 10 ms)
B0BCLR	FT0IEN	; To disable T0 interrupt service
B0BCLR	FT0IRQ	; To clear T0 interrupt request
B0BSET	FT0ENB	; To enable T0 timer
; Go into green mode		
B0BCLR	FCPUM0	;To set CPUMx = 10
B0BSET	FCPUM1	

\* **Note:** During the green mode with T0 wake-up function, the wakeup pins and T0 can wakeup the system back to the last mode. T0 wake-up period is controlled by program and T0ENB must be set.

**➤ Example: Switch normal/slow mode to Green mode and enable ADC wakeup function.**

; Set ADC timer wakeup function.

MOV	A,#11111111b	
B0MOV	VREG,A	; To Turn On all analog voltage regulators.
MOV	A,#00000111b	
B0MOV	AMPM1,A	; To Set PGIA function.
B0BSET	FADCENB	; To enable ADC Function
; Go into green mode with high clock running		
B0BSET	FCPUM1	;To set CPUM0 = 1

\* **Note\_1:** when system into green mode with conditions of ADC function enable and high clock still running, the system will be wakeup when ADC conversion complete.

\* **Note\_2:** The ADC green mode wakeup function is disable when ADCENB=0 or stop high clock (STPHX=1) is set before into green mode.

## 5.3 WAKE-UP

### 5.3.1 OVERVIEW

Under power down mode (sleep mode) or green mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode or slow mode. The wakeup trigger sources are external trigger (P0 level change) and internal trigger (T0 timer overflow and ADC conversion complete).

- Power down mode is waked up to normal mode. The wakeup trigger is only external trigger (P0 level change)
- Green mode is waked up to last mode (normal mode or slow mode). The wakeup triggers are external trigger (P0 level change) and internal trigger (T0 timer overflow and ADC conversion complete).

### 5.3.2 WAKEUP TIME

When the system is in power down mode (sleep mode), the high clock oscillator stops. When waked up from power down mode, MCU waits for 64 internal high-speed RC oscillator (IHRC) clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

\* **Note:** Wakeup from green mode is no wakeup time because the clock doesn't stop in green mode.

The value of the power down wakeup time is as the following.

$$\text{The Wakeup time} = 1/F_{\text{hosc}} * 64 \text{ (sec)} + 1\sim 2\text{ILRC start-up time}$$

\* **Note:** The high clock start-up time is depended on the VDD. In general, high clock start-up time will be several micro-second (us) at VDD=3V.

Example: The system is waked up from power down (sleep mode) by P0 level change. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

$$\begin{aligned}\text{The wakeup time} &= 1/F_{\text{hosc}} * 64 = 16\mu\text{s} \text{ (} F_{\text{hosc}} = 4\text{MHz)} \\ \text{The total wakeup time} &= 16\mu\text{s} + 1\sim 2\text{ILRC start-up time}(\sim 50\mu\text{s})\end{aligned}$$

### 5.3.3 P1W WAKEUP CONTROL REGISTER

Under power down mode (sleep mode) and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The Port 0 and Port 1 have wakeup function. Port 0 wakeup function always enables, but the Port 1 is controlled by the P1W register.

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1W</b>	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

Bit[7:0] **P10W~P17W:** Port 1 wakeup function control bits.

0 = Disable P1n wakeup function.

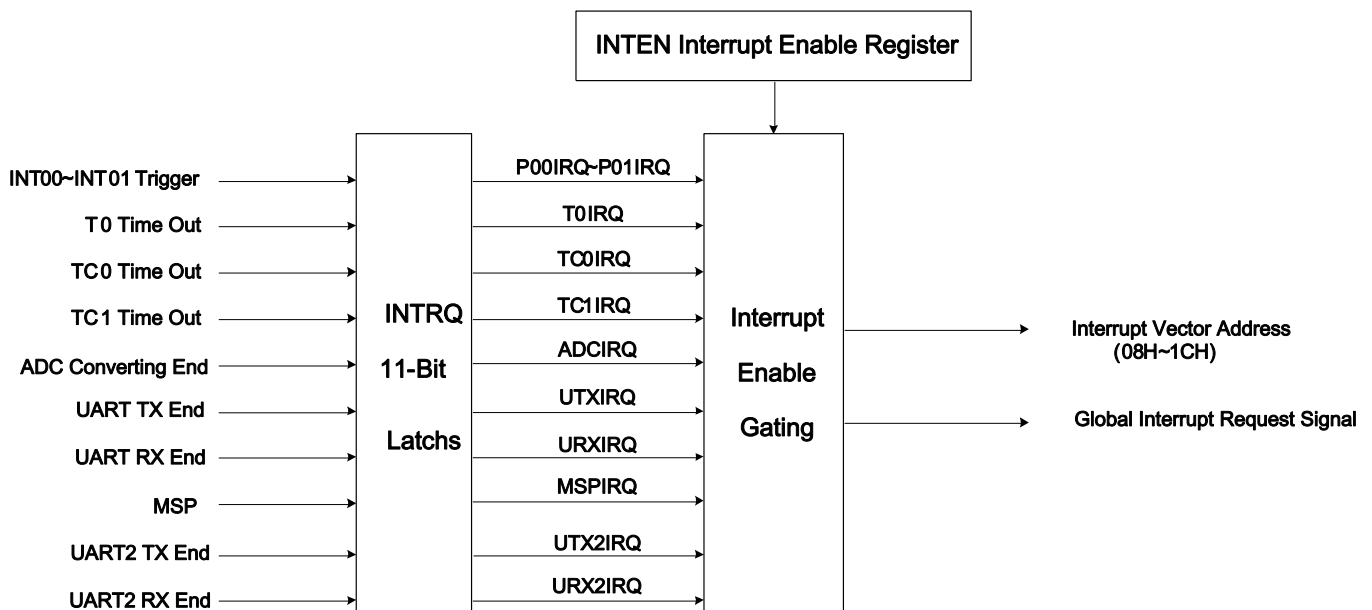
1 = Enable P1n wakeup function.



# 6 INTERRUPT

## 6.1 OVERVIEW

This MCU provides 11 interrupt sources, including 2 external interrupt (INT0/INT1) and 9 internal interrupt (T0/TC0/TC1/ADC/UTX/URX/UTX2/URX2/MSP). The external interrupt can wake up the chip while the system is switched from power down mode to high-speed normal mode, and interrupt request is latched until return to normal mode. Once interrupt service is executed, the GIE bit in STKP register will clear to “0” for stopping other interrupt request. On the contrast, when interrupt service exits, the GIE bit will set to “1” to accept the next interrupts’ request. The interrupt request signals are stored in INTRQ register.



**Note:** The GIE bit must enable during all interrupt operation.

## 6.2 INTERRUPT OPERATION

Interrupt operation is controlled by IRQ and IEN bits. The IRQ is interrupt source event indicator, no matter what interrupt function status (enable or disable). The IEN control the system interrupt execution. If IEN = 0, the system won't jump to interrupt vector to execute interrupt routine. If IEN = 1, the system executes interrupt operation when each of interrupt IRQ flags actives.

☞ **IEN = 1 and IRQ = 1, the program counter points to interrupt vector and execute interrupt service routine.**

When any interrupt requests occurs, the system provides to jump to interrupt vector and execute interrupt routine. The first procedure is “PUSH” operation. The end procedure after interrupt service routine execution is “POP” operation. The “PUSH” and “POP” operations aren't through instruction (PUSH, POP) and executed by hardware automatically.

☞ “PUSH” operation: PUSH operation saves the contents of ACC and working registers (0x80~0x8F) into hardware buffers. PUSH operation executes before program counter points to interrupt vector. The RAM bank keeps the status of main routine and doesn't switch to bank 0 automatically. The RAM bank is selected by program.

☞ “POP” operation: POP operation reloads the contents of ACC and working registers (0x80~0x8F) from hardware buffers. POP operation executes as RETI instruction executed. The RAM bank switches to last status of main routine after reloading RBANK content.

☞ 0x80~0x87 working registers include L, H, R, Z, Y, X, PFLAG, RBANK, W0~W7.

## 6.3 INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including eleven internal interrupts, two external interrupts enable control bits. One of the register to be set "1" is to enable the interrupt request function. Once of the interrupt occur, the stack is incremented and program jump to ORG 8~18H to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTEN0</b>	ADCIEN	TC1IEN	TC0IEN	T0IEN	URXIEN	UTXIEN	P01IEN	P00IEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 7     **ADCIEN**: ADC interrupt control bit.  
0 = Disable ADC interrupt function.  
1 = Enable ADC interrupt function.
- Bit 6     **TC1IEN**: TC1 timer interrupt control bit.  
0 = Disable TC1 interrupt function.  
1 = Enable TC1 interrupt function.
- Bit 5     **TC0IEN**: TC0 timer interrupt control bit.  
0 = Disable TC0 interrupt function.  
1 = Enable TC0 interrupt function.
- Bit 4     **T0IEN**: T0 timer interrupt control bit.  
0 = Disable T0 interrupt function.  
1 = Enable T0 interrupt function.
- Bit 3     **URXIEN**: UART receive interrupt control bit.  
0 = Disable UART receive interrupt function.  
1 = Enable UART receive interrupt function.
- Bit 2     **UTXIEN**: UART transmit interrupt control bit.  
0 = Disable UART transmit interrupt function.  
1 = Enable UART transmit interrupt function.
- Bit 1     **P01IEN**: External P01 interrupt (INT01) control bit.  
0 = Disable INT01 interrupt function.  
1 = Enable INT01 interrupt function.
- Bit 0     **P00IEN**: External P00 interrupt (INT00) control bit.  
0 = Disable INT00 interrupt function.  
1 = Enable INT00 interrupt function.

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTEN1</b>	-	-	-	-	URX2IEN	UTX2IEN	T0MOD	MSPIEN
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

- Bit 3     **URX1IEN**: UART2 receive interrupt control bit.  
0 = Disable UART2 receive interrupt function.  
1 = Enable UART2 receive interrupt function.
- Bit 2     **UTX1IEN**: UART2 transmit interrupt control bit.  
0 = Disable UART2 transmit interrupt function.  
1 = Enable UART2 transmit interrupt function.
- Bit 1     **T0MOD**: T0 timer control bit of interrupt and green mode wakeup mode.  
0 = Interrupt function and green mode wakeup enable when T0 overflow.  
1 = Interrupt function and green mode wakeup enable when "SEC" overflow.

- \* **Note:** If want to disable interrupt (xIEN), suggest add **jmp \$+2** instruction.
- \* **Note:** xIEN (P00IEN,P01IEN,UTXIEN,URXIEN,T0IEN,T0IEN,TC0IEN,TC1IEN,ADCIEN)

**Example: For disable P00IEN**

```

B0BCLR      FP00IEN      ; disable
JMP          $+2          ;

```

Setting				Function description		
T0ENB	T0TB	T0MOD	T0IEN	Interrupt	Green Mode Wakeup	Note
0	x	x	x	No	No	T0 disable
1	0	x	0	No	Yes	Green mode wakeup active when T0 overflow
1	0	x	1	Yes	Yes	Interrupt and Green mode wakeup function are active.
1	1	0	0	No	Yes	Green mode wakeup active when T0 overflow
1	1	0	1	Yes	Yes	Interrupt and Green mode wakeup function are active.
1	1	1	0	No	By "SEC" overflow	T0 is stilling counting. Green mode wakeup function is active when RTC register "SEC" overflow. (00~3BH)
1	1	1	1	By "SEC" overflow		T0 is stilling counting. Interrupt and Green mode wakeup function are active when RTC register "SEC" overflow occurring.

Bit 0 **MSPIEN:** MSP control bit of interrupt and green mode wakeup mode.  
 0 = Disable MSP interrupt function.  
 1 = Enable MSP interrupt function.

- \* **Note:** If P00/P01=input mode, P00IEN/P01IEN=0, then P00/P01 interrupt not active, but P00IRQ/P01IRQ still active

## 6.4 INTRQ INTERRUPT REQUEST REGISTER

INTRQ is the interrupt request flag register. The register includes all interrupt request indication flags. Each one of the interrupt requests occurs; the bit of the INTRQ register would be set "1". The INTRQ value needs to be clear by programming after detecting the flag. In the interrupt vector of program, users know the any interrupt requests occurring by the register and do the routine corresponding of the interrupt request.

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTRQ0</b>	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	URXIRQ	UTXIRQ	P01IRQ	P00IRQ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 7      **ADCIRQ**: ADC interrupt request flag.  
0 = None ADC interrupt request.  
1 = ADC interrupt request.
- Bit 6      **TC1IRQ**: TC1 timer interrupt request flag.  
0 = None TC1 interrupt request.  
1 = TC1 interrupt request.
- Bit 5      **TC0IRQ**: TC0 timer interrupt request flag.  
0 = None TC0 interrupt request.  
1 = TC0 interrupt request.
- Bit 4      **T0IRQ**: T0 timer interrupt request flag.  
0 = None T0 interrupt request.  
1 = T0 interrupt request.
- Bit 3      **URXIRQ**: UART receive interrupt request flag.  
0 = None UART receive interrupt request.  
1 = UART receive interrupt request.
- Bit 2      **UTXIRQ**: UART transmit interrupt request flag..  
0 = None UART transmit interrupt request.  
1 = UART transmit interrupt request.
- Bit 1      **P01IRQ**: External P01 interrupt (INT01) request flag.  
0 = None INT01 interrupt request.  
1 = INT01 interrupt request.
- Bit 0      **P00IRQ**: External P00 interrupt (INT00) request flag.  
0 = None INT00 interrupt request.  
1 = INT00 interrupt request.

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTRQ1</b>	-	-	-	-	URX2IRQ	UTX2IRQ	-	MSPIRQ
Read/Write	-	-	-	-	R/W	R/W	-	R/W
After reset	-	-	-	-	0	0	-	0

- Bit 2      **URX2IRQ**: RX2 interrupt request flag.  
0 = None RX2 interrupt request.  
1 = RX2 interrupt request.
- Bit 1      **UTX2IRQ**: TX2 interrupt request flag.  
0 = None TX2 interrupt request.  
1 = TX2 interrupt request.
- Bit 0      **MSPIRQ**: MSP interrupt request flag.  
0 = None MSP interrupt request.  
1 = MSP interrupt request.

## 6.5 GIE GLOBAL INTERRUPT OPERATION

GIE is the global interrupt control bit. All interrupts start work after the GIE = 1. It is necessary for interrupt service request. One of the interrupt requests occurs, and the program counter (PC) points to the interrupt vector (ORG 8~1C) and the stack add 1 level.

0BAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STKP</b>	GIE	-	-	-	STKPB3	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	-	R/W	R/W	R/W	R/W
After reset	0	-	-	-	1	1	1	1

Bit 7      **GIE:** Global interrupt control bit.  
 0 = Disable global interrupt.  
 1 = Enable global interrupt.

**Example: Set global interrupt control bit (GIE).**

```
BOBSET      FGIE      ; Enable GIE
```

**\* Note: The GIE bit must enable during all interrupt operation.**

---

**Example: INT0 interrupt service routine.**

---

	ORG	8	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	...		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FP00IRQ	; Check P00IRQ
	JMP	EXIT_INT	; P00IRQ = 0, exit interrupt vector
	B0BCLR	FP00IRQ	; Reset P00IRQ
	...		; INT0 interrupt service routine
	...		
EXIT_INT:			
	...		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

## 6.6 EXTERNAL INTERRUPT OPERATION (INT0~INT1)

Sonix provides 2 sets external interrupt sources in the micro-controller. INT0 and INT1 are external interrupt trigger sources and build in edge trigger configuration function. When the external edge trigger occurs, the external interrupt request flag will be set to "1" when the external interrupt control bit enabled. If the external interrupt control bit is disabled, the external interrupt request flag won't active when external edge trigger occurrence. When external interrupt control bit is enabled and external interrupt edge trigger is occurring, the program counter will jump to the interrupt vector (ORG 0x0008 ,0x000A) and execute interrupt service routine.

The external interrupt builds in wake-up latch function. That means when the system is triggered wake-up from power down mode, the wake-up source is external interrupt source (P0.0 or P0.1), and the trigger edge direction matches interrupt edge configuration, the trigger edge will be latched, and the system executes interrupt service routine first after wake-up.

0BBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PEDGE</b>	-	-	-	-	P01G1	P01G0	P00G1	P00G0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	1	0	1	0

Bit[3:2] **P01G[1:0]:** INT1 edge trigger select bits.

00 = reserved,  
01 = rising edge,  
10 = falling edge,  
11 = rising/falling bi-direction.

Bit[1:0] **P00G[1:0]:** INT0 edge trigger select bits.

00 = reserved,  
01 = rising edge,  
10 = falling edge,  
11 = rising/falling bi-direction.

**Example: Setup INT0 interrupt request and bi-direction edge trigger.**

```

MOV      A, #03H
B0MOV    PEDGE, A      ; Set INT0 interrupt trigger as bi-direction edge.

B0BSET   FP00IEN       ; Enable INT0 interrupt service
B0BCLR   FP00IRQ       ; Clear INT0 interrupt request flag
B0BSET   FGIE          ; Enable GIE

```

**Example: INT0 interrupt service routine.**

```

ORG      8              ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:
...              ; Push routine to save ACC and PFLAG to buffers.

B0BTS1   FP00IRQ       ; Check P00IRQ
JMP      EXIT_INT      ; P00IRQ = 0, exit interrupt vector

B0BCLR   FP00IRQ       ; Reset P00IRQ
...              ; INT0 interrupt service routine

EXIT_INT:
...              ; Pop routine to load ACC and PFLAG from buffers.
RETI        ; Exit interrupt vector

```

## 6.7 T0 INTERRUPT OPERATION

When the T0C counter occurs overflow, the T0IRQ will be set to “1” however the T0IEN is enable or disable. If the T0IEN = 1, the trigger event will make the T0IRQ to be “1” and the system enter interrupt vector. If the T0IEN = 0, the trigger event will make the T0IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

**\* Example: T0 interrupt request setup.**

B0BCLR	FT0IEN	; Disable T0 interrupt service
B0BCLR	FT0ENB	; Disable T0 timer
MOV	A, #20H	;
B0MOV	T0M, A	; Set T0 clock = Fcpu / 64
MOV	A, #74H	; Set T0C initial value = 74H
B0MOV	T0C, A	; Set T0 interval = 10 ms
B0BSET	FT0IEN	; Enable T0 interrupt service
B0BCLR	FT0IRQ	; Clear T0 interrupt request flag
B0BSET	FT0ENB	; Enable T0 timer
B0BSET	FGIE	; Enable GIE

**Example: T0 interrupt service routine.**

	ORG	0C	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
	...		; Push routine to save ACC and PFLAG to buffers.
	<b>B0BTS1</b>	<b>FT0IRQ</b>	<b>; Check T0IRQ</b>
	<b>JMP</b>	<b>EXIT_INT</b>	<b>; T0IRQ = 0, exit interrupt vector</b>
	<b>B0BCLR</b>	<b>FT0IRQ</b>	<b>; Reset T0IRQ</b>
	<b>MOV</b>	<b>A, #74H</b>	
	<b>B0MOV</b>	<b>T0C, A</b>	<b>; Reset T0C.</b>
	...		<b>; T0 interrupt service routine</b>
	...		
EXIT_INT:			
	...		; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

**\* Note: In RTC mode, don't reset T0C in interrupt service routine.**



## 6.8 TC0 INTERRUPT OPERATION

When the TC0C counter overflows, the TC0IRQ will be set to “1” no matter the TC0IEN is enable or disable. If the TC0IEN and the trigger event TC0IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the TC0IEN = 0, the trigger event TC0IRQ is still set to be “1”. Moreover, the system won't execute interrupt vector even when the TC0IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

### Example: TC0 interrupt request setup.

B0BCLR	FTC0IEN	; Disable TC0 interrupt service
B0BCLR	FTC0ENB	; Disable TC0 timer
MOV	A, #10H	;
B0MOV	TC0M, A	; Set TC0 clock = Fcpu / 64
MOV	A, #FFH	; Set TC0CH initial value = FFH
B0MOV	TC0CH, A	
MOV	A, #74H	; Set TC0CL initial value = 74H
B0MOV	TC0CL, A	
B0BSET	FTC0IEN	; Enable TC0 interrupt service
B0BCLR	FTC0IRQ	; Clear TC0 interrupt request flag
B0BSET	FTC0ENB	; Enable TC0 timer
B0BSET	FGIE	; Enable GIE

### Example: TC0 interrupt service routine.

	ORG	0CH	; Interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:			
...			; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FTC0IRQ	; Check TC0IRQ
	JMP	EXIT_INT	; TC0IRQ = 0, exit interrupt vector
	B0BCLR	FTC0IRQ	; Reset TC0IRQ
	MOV	A, #FFH	
	B0MOV	TC0CL, A	
	MOV	A, #74H	
	B0MOV	TC0CH, A	; Reset TC0C.
	...		; TC0 interrupt service routine
EXIT_INT:			
...			; Pop routine to load ACC and PFLAG from buffers.
	RETI		; Exit interrupt vector

\* **Note: In RTC mode, don't reset T0C in interrupt service routine.**

## 6.9 TC1 INTERRUPT OPERATION

When the TC1C counter overflows, the TC1IRQ will be set to “1” no matter the TC1IEN is enable or disable. If the TC1IEN and the trigger event TC1IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the TC1IEN = 0, the trigger event TC1IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the TC1IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

## 6.10 UART/UART2 INTERRUPT OPERATION

When the UART/UART2 transmitter successfully, the URXIRQ/UTXIRQ/URX2IRQ/UTX2IRQ will be set to “1” no matter the URXIEN/UTXIEN/URX2IEN/UTX2IEN is enable or disable. If the URXIEN/UTXIEN/URX2IEN/UTX2IEN and the trigger event URXIRQ/UTXIRQ/URX2IRQ/UTX2IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the URXIEN/UTXIEN/URX2IEN/UTX2IEN = 0, the trigger event URXIRQ/UTXIRQ/URX2IRQ/UTX2IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the URXIEN/UTXIEN/URX2IEN/UTX2IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

### ➤ Example: UART receive and transmit interrupt request setup.

B0BSET	FURXIEN	; Enable UART receive interrupt service
B0BCLR	FURXIRQ	; Clear UART receive interrupt request flag
B0BSET	FUTXIEN	; Enable UART transmit interrupt service
B0BCLR	FUTXIRQ	; Clear UART transmit interrupt request flag
B0BSET	FGIE	; Enable GIE

### ➤ Example: UART receive interrupt service routine.

	ORG	16	; RX interrupt vector
	JMP	INT_SERVICE	
INT_SERVICE:	...		; Push routine to save ACC and PFLAG to buffers.
	B0BTS1	FURXIRQ	; Check RXIRQ
	JMP	EXIT_INT	; RXIRQ = 0, exit interrupt vector
	B0BCLR	FURXIRQ	; Reset RXIRQ
	...		; UART receive interrupt service routine
	...		
EXIT_INT:	...		; Pop routine to load ACC and PFLAG from buffers.
	...		
	RETI		; Exit interrupt vector

## 6.11 MULTI-INTERRUPT OPERATION

Under certain condition, the software designer uses more than one interrupt requests. Processing multi-interrupt request requires setting the priority of the interrupt requests. The IRQ flags of interrupts are controlled by the interrupt event. Nevertheless, the IRQ flag “1” doesn’t mean the system will execute the interrupt vector. In addition, which means the IRQ flags can be set “1” by the events without enable the interrupt. Once the event occurs, the IRQ will be logic “1”. The IRQ and its trigger event relationship is as the below table.

<b>Interrupt Name</b>	<b>Trigger Event Description</b>
P00IRQ	P00trigger controlled by PEDGE
P01IRQ	P01 trigger controlled by PEDGE
T0IRQ	T0C overflow.
TC0IRQ	TC0C overflow.
TC1IRQ	TC1C overflow.
ADCIRQ	ADC transmitter successfully.
TXIRQ	UART receive successfully.
RXIRQ	UART transmit successfully.
TX2IRQ	UART2 receive successfully.
RX2IRQ	UART2 transmit successfully.
MSPIRQ	MSP interrupt request.

For multi-interrupt conditions, two things need to be taking care of. One is that it is multi-vector and each of interrupts points to unique vector. Two is using IEN and IRQ flags to decide which interrupt to be executed. The following example shows the way to define the interrupt vector in the program memory.

**Example: Check the interrupt request under multi-interrupt operation**

```

ORG      8                ; Interrupt vector
JMP      ISR_INT0
ORG      A
JMP      ISR_INT1
ORG      C
JMP      ISR_T0
ORG      E
JMP      ISR_T0C
ORG      10
JMP      ISR_T1C
ORG      12
JMP      ISR_ADC
ORG      14
JMP      ISR_TX
ORG      16
JMP      ISR_RX
ORG      18
JMP      ISR_MSP
ORG      1A
JMP      ISR_TX2
ORG      1C
JMP      ISR_RX

ISR_WAKE:                ; WAKE-UP interrupt service routine
    RETI                 ; Exit interrupt vector
ISR_INT0:                 ; INT0 interrupt service routine
    RETI                 ; Exit interrupt vector
ISR_INT1:                 ; INT1 interrupt service routine
    RETI                 ; Exit interrupt vector
...
...
ISR_UART_TX:             ; UART_TX interrupt service routine
    RETI                 ; Exit interrupt vector

```

# 7 I/O Port

## 7.1 OVERVIEW

The micro-controller builds in 36 pin I/O. Most of the I/O pins are mixed with analog pins and special function pins. The I/O shared pin list is as following.

I/O Pin		Share Pin		Share Pin control Condition
Name	Type	Name	Type	
P0.0	I/O	INT0	DC	P00IEN=1
P0.1	I/O	INT1		P01IEN=1
P0.2	I/O	SCL		MSPEN=1
		TC0		TC0CKS=1, TC0ENB=1
P0.3	I/O	SDA		MSPEN=1
		TC1		TC1CKS=1, TC1ENB=1
P0.4	I/O	UTX		UTXEN=1
P0.5	I/O	URX		URXEN=1
P0.6	I/O	PWM1		PWM1OUT=1
P0.7	I/O	PWM2	PWM2OUT=1	
P1.0	I/O	LBT1	DC	P10IO=1, Low battery detect input pin
P1.1	I/O	LBT2		P11IO=1, Low battery detect Gnd pin
P1.2	I/O	N.A		N.A
P1.3	I/O			
P1.4	I/O			
P1.5	I/O			
P1.6	I/O	EICK		EICE SDA connected
P1.7	I/O	EIDA		
P2.0	I/O	LED1	AC	LEDEN1=1 ,LEDIO=1
P2.1	I/O	LED2		LEDEN2=1,LEDIO=1
P2.2	I/O	LED3		LEDEN3=1,LEDIO=1
P2.3	I/O	LED4		LEDEN4=1,LEDIO=1
P3.0	I/O	SEG26	AC	VLCD1 connect VLCD0
P3.1	I/O	SEG27		
P3.2	I/O	SEG28		
P3.3	I/O	SEG29		
P3.4	I/O	SEG30		
P3.5	I/O	SEG31		
P3.6	I/O	URX2		URX2EN=1
P3.7	I/O	UTX2		UTX2EN=1
P5.0	I/O	N.A	DC	N.A
P5.1	I/O			
P5.2	I/O			
P5.3	I/O			
P5.4	I/O			
P5.5	I/O			
P5.6	I/O			
P5.7	I/O			

## 7.2 I/O PORT MODE

The port direction is programmed by PnM register. When the bit of PnM register is “0”, the pin is input mode. When the bit of PnM register is “1”, the pin is output mode.

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0M</b>	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1M</b>	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P2M</b>					P23M	P22M	P21M	P20M
Read/Write					R/W	R/W	R/W	R/W
After reset					0	0	0	0

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P3M</b>	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5M</b>	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PnM[7:0]**: Pn mode control bits. (n = 0~1).  
0 = Pn is input mode.  
1 = Pn is output mode.

**Note:** Users can program them by bit control instructions (**B0BSET**, **B0BCLR**).

**Example: I/O mode selecting**

CLR	P0M	; Set all ports to be input mode.
CLR	P1M	
MOV	A, #0FFH	; Set all ports to be output mode.
B0MOV	P0M, A	
B0MOV	P1M, A	
B0BCLR	P0M.0	; Set P0.0 to be input mode.
B0BSET	P0M.0	; Set P0.0 to be output mode.

## 7.3 I/O PULL UP REGISTER

The I/O pins build in internal pull-up resistors and only support I/O input mode. The port internal pull-up resistor is programmed by PnUR register. When the bit of PnUR register is “0”, the I/O pin’s pull-up is disabled. When the bit of PnUR register is “1”, the I/O pin’s pull-up is enabled.

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0UR</b>	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1UR</b>	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P2UR</b>	-	-	-	-	P23R	P22R	P21R	P20R
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0A6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P3UR</b>	P37R	P36R	P35R	P34R	P33R	P32R	P31R	P30R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5UR</b>	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

### Example: I/O Pull up Register

```

MOV      A, #0FFH      ; Enable Port0, 1 Pull-up register,
B0MOV    P0UR, A        ;
B0MOV    P1UR, A

```

## 7.4 I/O PORT DATA REGISTER

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0</b>	P07	P06	P05	P04	P03	P02	P01	P00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1</b>	P17	P16	P15	P14	P13	P12	P11	P10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P2</b>	-	-	-	-	P23	P22	P21	P20
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After reset	-	-	-	-	0	0	0	0

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P3</b>	P37	P36	P35	P34	P33	P32	P31	P30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5</b>	P57	P56	P55	P54	P53	P52	P51	P50
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

**Example: Read data from input port.**

```

B0MOV      A, P0          ; Read data from Port 0
B0MOV      A, P1          ; Read data from Port 1

```

**Example: Write data to output port.**

```

MOV        A, #0FFH      ; Write data FFH to all Port.
B0MOV      P0, A
B0MOV      P1, A

```

**Example: Write one bit data to output port.**

```

B0BSET     P0.0           ; Set P0.0 and P1.3 to be "1".
B0BSET     P1.3

B0BCLR     P0.0           ; Set P0.0 and P1.3 to be "0".
B0BCLR     P1.3

```

# 8 Timer

## 8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, watchdog timer overflow signal raises and resets MCU. Watchdog timer clock source is internal low-speed oscillator 32KHz RC type and through programmable pre-scalar controlled by WDT\_CLK code option.

**Watchdog timer interval time = 16384/ Internal Low-Speed oscillator (sec).**

VDD	Internal Low RC Freq.	Watchdog Overflow Time
3.3V	32KHz	512ms

The watchdog timer has three operating options controlled “WatchDog” code option.

- \* **Disable:** Disable watchdog timer function.
- \* **Enable:** Enable watchdog timer function. Watchdog timer actives in normal mode and slow mode. In power down mode and green mode, the watchdog timer stops.
- \* **Always\_On:** Enable watchdog timer function. The watchdog timer actives and not stop in power down mode and green mode.

- \* **Note:**
1. In high noisy environment, the “Always\_On” option of watchdog operations is the strongly recommendation to make the system reset under error situations and re-start again.
  2. If watchdog is “Always\_On” mode, it keeps running event under power down mode or green mode.

Watchdog clear is controlled by WDTR register. Moving **0x5A** data into WDTR is to reset watchdog timer.

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>WDTR</b>	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
Read/Write	W	W	W	W	W	W	W	W
After reset	0	0	0	0	0	0	0	0

- **Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.**

Main:

```

MOV      A, #5AH          ; Clear the watchdog timer.
B0MOV    WDTR, A
...
CALL     SUB1
CALL     SUB2
...
JMP      MAIN

```



➤ **Example: Clear watchdog timer by “@RST\_WDT” macro of Sonix IDE.**

Main:

```

    @RST_WDT                ; Clear the watchdog timer.
    ...
    CALL        SUB1
    CALL        SUB2
    ...
    JMP         MAIN

```

Watchdog timer application note is as following.

- Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.
- Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail.
- Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

☞ **Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.**

Main:

```

    ...                ; Check I/O.
    ...                ; Check RAM

```

```

Err:    JMP $          ; I/O or RAM error. Program jump here and don't
                        ; clear watchdog. Wait watchdog timer overflow to reset IC.

```

Correct:

```

    MOV        A, #5AH
    B0MOV     WDTR, A    ; I/O and RAM are correct. Clear watchdog timer and
                        ; execute program.
    ...
    CALL        SUB1
    CALL        SUB2
    ...
    ...
    JMP         MAIN    ; Clear the watchdog timer.

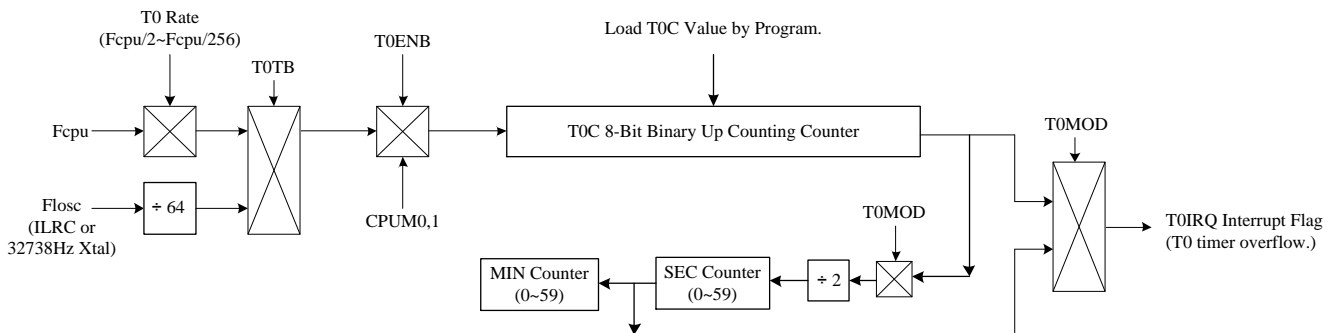
```

## 8.2 T0 8-BIT BASIC TIMER

### 8.2.1 OVERVIEW

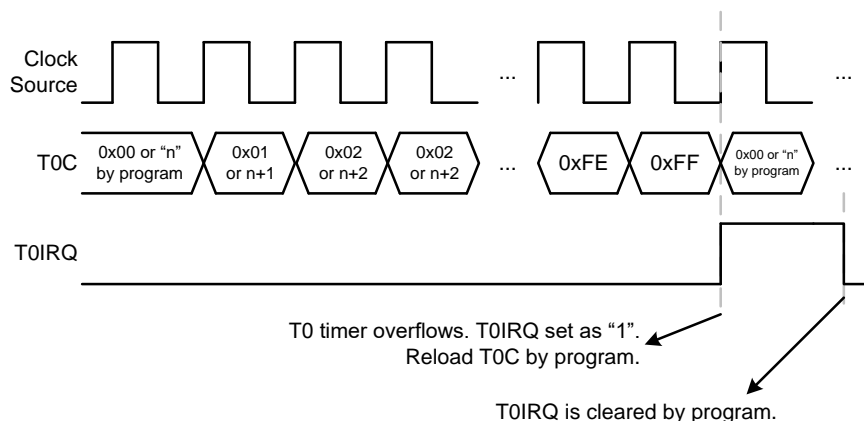
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (T0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through T0M, T0C registers and supports RTC function. The T0 builds in green mode wake-up function. When T0 timer overflow occurs under green mode, the system will be waked-up to last operating mode.

- ☞ **8-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** T0 timer function supports interrupt function. When T0 timer occurs overflow, the T0IRQ activates and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **RTC function:** T0 RTC function is controlled by T0TB bit. The RTC period is 0.5sec @32KHz when T0MOD=0. When T0MOD=1, T0 interrupt period is 60sec @32KHz.
- ☞ **Green mode function:** T0 keeps running in green mode and can wake-up from green mode as T0ENB = 1. System will be Wake-up when T0IRQ activates after T0 timer overflow occurrence.



## T0 TIMER OPERATION

T0 timer is controlled by T0ENB bit. When T0ENB=0, T0 timer stops. When T0ENB=1, T0 timer starts to count. T0C increases "1" by timer clock source. When T0 overflow event occurs, T0IRQ flag is set as "1" to indicate overflow and cleared by program. The overflow condition is T0C count from full scale (0xFF) to zero scale (0x00). T0 doesn't build in double buffer, so load T0C by program when T0 timer overflows to fix the correct interval time. If T0 timer interrupt function is enabled (T0IEN=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 000CH) and executes interrupt service routine after T0 overflow occurrence. Clear T0IRQ by program is necessary in interrupt procedure. T0 timer can works in normal mode, slow mode and green mode. In green mode, T0 keeps counting, set T0IRQ and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0rate[2:0] pre-scalar to decide  $F_{cpu}/2 \sim F_{cpu}/256$ . T0 length is 8-bit

(256 steps), and the one count period is each cycle of input clock.

T0RATE [2:0]	T0 Clock	High Speed Mode (Fcpu=1 MIP)		Low Speed Mode (Fcpu=32kHz / 4)	
		Max overflow Time	One Step = max/256	Max overflow Time	One Step = max/256
000	Fcpu / 256	65.563 ms	256 us	8192 ms	3200 us
001	Fcpu / 128	32.768 ms	128 us	4096 ms	1600 us
010	Fcpu / 64	16.384 ms	64 us	2048 ms	8000 us
011	Fcpu / 32	8.192 ms	32 us	1024 ms	4000 us
100	Fcpu / 16	4.096 ms	16 us	512 ms	2000 us
101	Fcpu / 8	2.048 ms	8 us	256 ms	1000 us
110	Fcpu / 4	1.024 ms	4 us	128 ms	500 us
111	Fcpu / 2	0.512 ms	2 us	64 ms	250 us

## 8.3 T0 MODE REGISTER

T0M is T0 timer mode control register to configure T0 operating mode including T0 pre-scaler, clock source... These configurations must be setup completely before enabling T0 timer.

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>T0M</b>	T0ENB	T0rate2	T0rate1	T0rate0	-	-	TC0GN	T0TB
Read/Write	R/W	R/W	R/W	R/W	-	-	R/W	R/W
After reset	0	0	0	0	-	-	0	0

Bit 7 **T0ENB**: T0 timer control bit.

0 = Disable.

1 = Enable.

Bit [6:4] **T0RATE [2:0]**: T0 timer clock source select bits.

000 = Fcpu/256, 001 = Fcpu/128, 010 = Fcpu/64, 011 = Fcpu/32, 100 = Fcpu/16, 101 = Fcpu/8,

110 = Fcpu/4, 111 = Fcpu/2.

Bit 1 **TC0GN**: Enable TC0 Green mode wake up function

0 = Disable.

1 = Enable.

Bit 0 **T0TB**: T0 timer RTC function control bit.

0 = Disable. T0 timer clock source is Fcpu.

1 = Enable. T0 timer clock source is external oscillator.

 **Note:** T0RATE is not available in RTC mode. The T0 interval time is fixed at 0.5 sec.

### 8.3.1 T0C COUNTING REGISTER

T0C is T0 8-bit counter. When T0C overflow occurs, the T0IRQ flag is set as "1" and cleared by program. The T0C decides T0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to T0C register, and then enable T0 timer to make sure the first cycle correct. After one T0 overflow occurs, the T0C register is loaded a correct value by program.

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>T0C</b>	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The equation of T0C initial value is as following.

$$T0C \text{ initial value} = 256 - (T0 \text{ interrupt interval time} * T0 \text{ clock rate})$$

**Example:** To calculation T0C to obtain 10ms T0 interval time. T0 clock source is Fcpu = 8MHz/4 = 2MHz. Select T0RATE=001 (Fcpu/128).

T0 interval time = 10ms. T0 clock rate = 8MHz/4/128

$$\begin{aligned}
 T0C \text{ initial value} &= 256 - (T0 \text{ interval time} * \text{input clock}) \\
 &= 256 - (10\text{ms} * 8\text{MHz} / 4 / 128) \\
 &= 256 - (10^{-2} * 8 * 10^6 / 4 / 128) \\
 &= 64H
 \end{aligned}$$

 **Note:** In RTC mode, T0C is 256 counts and generates T0 0.5 sec interval time. Don't change T0C value in RTC mode.

## T0 TIMER OPERATION EXPLAME

### ● T0 TIMER CONFIGURATION:

; Reset T0 timer.

```
MOV      A, #0x00      ; Clear T0M register.
B0MOV    T0M, A
```

; Set T0 clock source and T0 rate.

```
MOV      A, #0nnn0000b
B0MOV    T0M, A
```

; Set T0C register for T0 Interval time.

```
MOV      A, #value
B0MOV    T0C, A
```

; Clear T0IRQ

```
B0BCLR    FT0IRQ
```

; Enable T0 timer and interrupt function.

```
B0BSET    FT0IEN      ; Enable T0 interrupt function.
B0BSET    FT0ENB      ; Enable T0 timer.
```

● T0 works in RTC mode:

; Reset T0 timer.

```
MOV      A, #0x00      ; Clear T0M register.
B0MOV    T0M, A
```

; Set T0 RTC function.

```
B0BSET    FT0TB
```

; Clear T0C.

```
CLR      T0C
```

; Clear T0IRQ

```
B0BCLR    FT0IRQ
```

; Enable T0 timer and interrupt function.

```
B0BSET    FT0IEN      ; Enable T0 interrupt function.
B0BSET    FT0ENB      ; Enable T0 timer.
```

BEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>SEC</b>	-	-	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit [5:0]

**SEC[5:0]:** T0 timer Second counter.

SEC counter range = 00H ~ 3BH (0~59)

When SEC value of 3BH and increase "1", counter overflow occurring with reset counter value to 00H and MIN counter will increase "1".

BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MIN</b>	-	-	MIN5	MIN4	MIN3	MIN2	MIN1	MIN0
Read/Write	-	-	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	-	0	0	0	0	0	0

Bit [5:0]

**MIN[5:0]:** T0 timer Minute counter.

MIN counter range = 00H ~ 3BH (0~59)

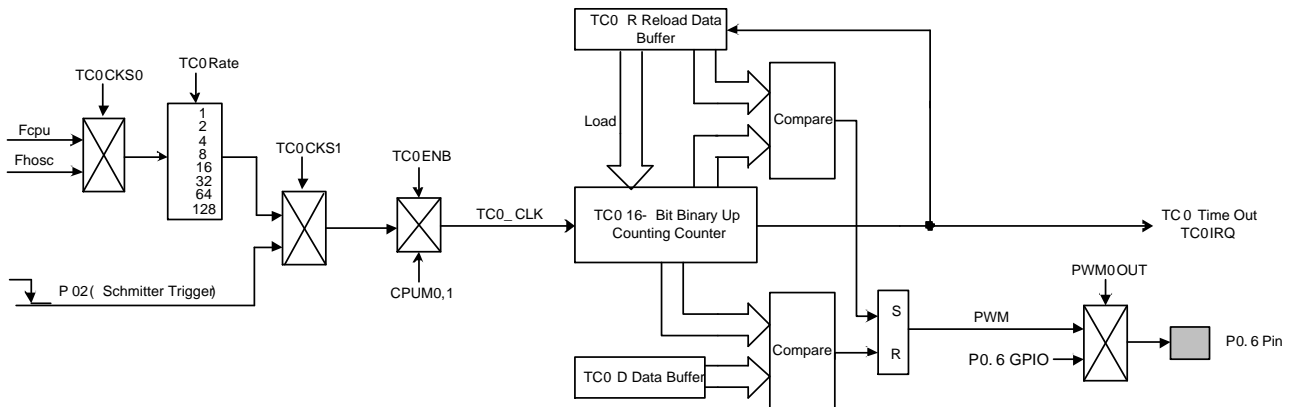
When MIN value of 3BH and increase "1", counter overflow occurring and reset counter value to 00H.

## 8.4 TC0 16-BIT TIMER/COUNTER

### 8.4.1 OVERVIEW

The TC0 timer is an 16-bit binary up timer with basic timer, event counter and PWM functions. The basic timer function supports flag indicator (TC0IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC0M, TC0C, TC0R registers. The event counter is changing TC0 clock source from system clock (Fcpu/Fhosc) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal...). TC0 becomes a counter to count external clock number to implement measure application. TC0 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC0 timer clock rate, TC0R and TC0D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster... The main purposes of the TC0 timer are as following.

- ☞ **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** TC0 timer function supports interrupt function. When TC0 timer occurs overflow, the TC0IRQ activates and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **Event Counter:** The event counter function counts the external clock counts.
- ☞ **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC0R and TC0D registers.
- ☞ **Green mode function:** All TC0 functions (timer, PWM, event counter, auto-reload) keeps running in green mode, and with wake-up function (TC0GN=1). Timer IRQ activates as any IRQ trigger occurrence, e.g. timer overflow...



## 8.4.2 TC0M MODE REGISTER

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0M</b>	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS1	TC0CKS0	PWM0OUT	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

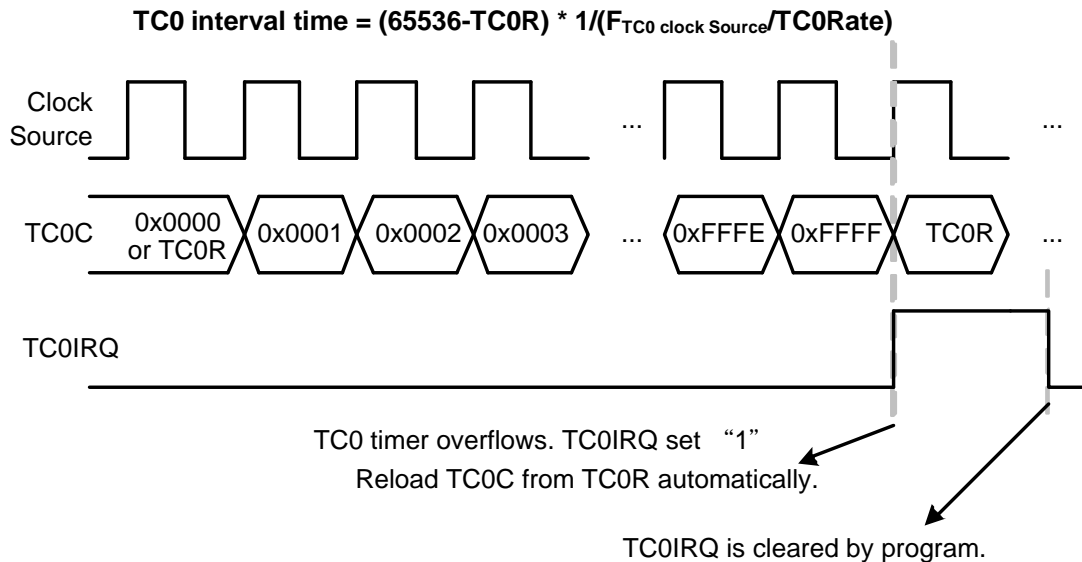
- Bit 1 **PWM0OUT**: PWM output control bit.  
 0 = Disable PWM output function, and P0.6 is GPIO mode.  
 1 = Enable PWM output function, and P0.6 outputs PWM signal.
- Bit 2 **TC0CKS0**: TC0 clock source select bit.  
 0 = Fcpu.  
 1 = Fhosc.
- Bit 3 **TC0CKS1**: TC0 clock source select bit.  
 0 = Internal clock (Fcpu and Fhosc controlled by TC0CKS0 bit).  
 1 = External input pin (P0.2) and enable event counter function. TC0rate[2:0] bits are useless.
- Bit [6:4] **TC0RATE[2:0]**: TC0 timer clock source select bits.

TC0CKS[1:0]	T0RATE[2:0]	TC0 Clock	TC0CKS[1:0]	T0RATE[2:0]	TC0 Clock
00	000	Fcpu / 128	01	000	Fhosc / 128
00	001	Fcpu / 64	01	001	Fhosc / 64
00	010	Fcpu / 32	01	010	Fhosc / 32
00	011	Fcpu / 16	01	011	Fhosc / 16
00	100	Fcpu / 8	01	100	Fhosc / 8
00	101	Fcpu / 4	01	101	Fhosc / 4
00	110	Fcpu / 2	01	110	Fhosc / 2
00	111	Fcpu / 1	01	111	Fhosc / 1
10	x	P0.2	11	x	P0.2

- Bit 7 **TC0ENB**: TC0 counter control bit.  
 0 = Disable TC0 timer.  
 1 = Enable TC0 timer.

## 8.5 TC0 TIMER OPERATION

TC0 timer is controlled by TC0ENB bit. When TC0ENB=1, TC0 timer starts to count. One count period is one clock source rate. TC0C is TC0 counter and up counting when TC0ENB=1. When TC0C counts from 0xFFFF to 0x0000, TC0 overflow condition is conformed and TC0IRQ set as "1". TC0 builds in auto-reload function and always enabled. When TC0 timer overflow occurs, the TC0C counter buffer will be reloaded from TC0R register automatically. TC0 is double buffer design. If the TC0R is changed by program, the new value will be loaded at next overflow occurrence, or the TC0 interval time is error. If TC0 interrupt function is enabled (TC0IEN=1), the program counter is pointed to interrupt vector to execute interrupt service routine after TC0 timer overflow occurrence.





### 8.5.1 TC0C COUNTING REGISTER

TC0C is TC0 16-bit counter. When TC0C overflow occurs, the TC0IRQ flag is set as “1” and cleared by program. The TC0C decides TC0 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC0C register and TC0R register first time, and then enable TC0 timer to make sure the first cycle correct. After one TC0 overflow occurs, the TC0C register is loaded a correct value from TC0R register automatically, not program.

C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0CL</b>	TC0CL7	TC0CL6	TC0CL5	TC0CL4	TC0CL3	TC0CL2	TC0CL1	TC0CL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC0CL [7:0]: TC0 timer Low Byte counter.

C6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0CH</b>	TC0CH7	TC0CH6	TC0CH5	TC0CH4	TC0CH3	TC0CH2	TC0CH1	TC0CH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC0CH [7:0]: TC0 timer High Byte counter.

The equation of TC0C initial value is as following.

$$TC0C \text{ initial value} = 65536 - (TC0 \text{ interrupt interval time} * TC0 \text{ clock rate})$$

## 8.5.2 TC0R AUTO-RELOAD REGISTER

TC0 timer builds in auto-reload function, and TC0R register stores reload data. When TC0C overflow occurs, TC0C register is loaded data from TC0R register automatically. Under TC0 timer counting status, to modify TC0 interval time is to modify TC0R register, not TC0C register. New TC0C data of TC0 interval time will be updated after TC0 timer overflow occurrence, TC0R loads new value to TC0C register. But at the first time to setup TC0M, TC0C and TC0R must be set the same value before enabling TC0 timer. TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1<sup>st</sup> buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer. This way can avoid any transitional condition to affect the correctness of TC0 interval time and PWM output signal.

C3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0RL</b>	TC0RL7	TC0RL6	TC0RL5	TC0RL4	TC0RL3	TC0RL2	TC0RL1	TC0RL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC0RL [7:0]: TC0 timer counter reload buffer low byte.

C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0RH</b>	TC0RH7	TC0RH6	TC0RH5	TC0RH4	TC0RH3	TC0RH2	TC0RH1	TC0RH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC0RH [7:0]: TC0 timer counter reload buffer high byte.

The equation of TC0R initial value is as following.

$$TC0R \text{ initial value} = 65536 - (TC0 \text{ interrupt interval time} * TC0 \text{ clock rate})$$

**Example:** To calculation TC0C and TC0R value to obtain 10ms TC0 interval time. TC0 clock source is  $F_{cpu} = 8MHz/8 = 1MHz$ . Select  $TC0RATE=000 (F_{cpu}/128)$ .

TC0 interval time = 10ms. TC0 clock rate =  $8MHz/8/128$

$$\begin{aligned}
 TC0C/TC0R \text{ initial value} &= 65536 - (TC0 \text{ interval time} * \text{input clock}) \\
 &= 65536 - (10ms * 8MHz / 8 / 128) \\
 &= 65536 - (10^{-2} * 8 * 10^6 / 8 / 128) \\
 &= FFB2H
 \end{aligned}$$

### 8.5.3 TC0D PWM DUTY REGISTER

TC0D register's purpose is to decide PWM duty. In PWM mode, TC0R controls PWM's cycle, and TC0D controls the duty of PWM. The operation is base on timer counter value. When TC0C = TC0D, the PWM high duty finished and exchange to low level. It is easy to configure TC0D to choose the right PWM's duty for application.

C7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0DL</b>	TC0DL7	TC0DL6	TC0DL5	TC0DL4	TC0DL3	TC0DL2	TC0DL1	TC0DL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0DH</b>	TC0DH7	TC0DH6	TC0DH5	TC0DH4	TC0DH3	TC0DH2	TC0DH1	TC0DH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **TC0DL [7:0]**: PWM duty control low byte buffer.

Bit [7:0] **TC0DH [7:0]**: PWM duty control high byte buffer.

The equation of TC0D initial value is as following.

$$TC0D \text{ initial value} = TC0R + (PWM \text{ high pulse width period} / TC0 \text{ clock rate})$$

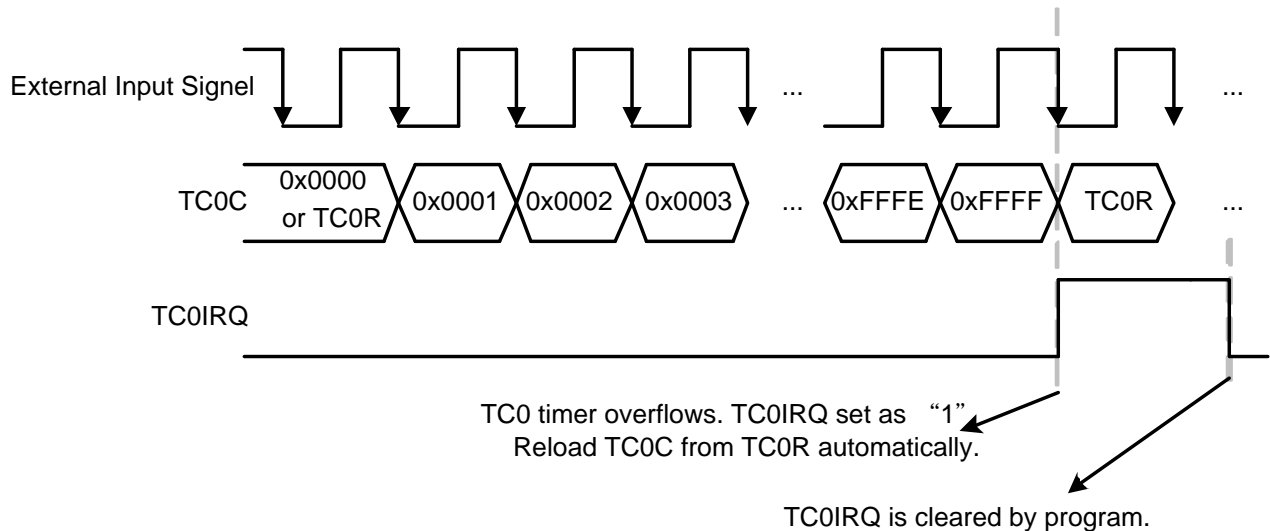
**Example: To calculate TC0D value to obtain 1/3 duty PWM signal. The TC0 clock source is Fcpu = 8MHz/8= 1MHz. Select TC0RATE=001 (Fcpu/64).**

TC0R = FF63H(TC0RH: FFH,TC0RL: 63H). TC0 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.

$$\begin{aligned}
 TC0D \text{ initial value} &= FF63H + (PWM \text{ high pulse width period} / TC0 \text{ clock rate}) \\
 &= FF63H + (3.33ms * 8MHz / 8 / 64) \\
 &= FF63H + 34H \\
 &= FF97H
 \end{aligned}$$

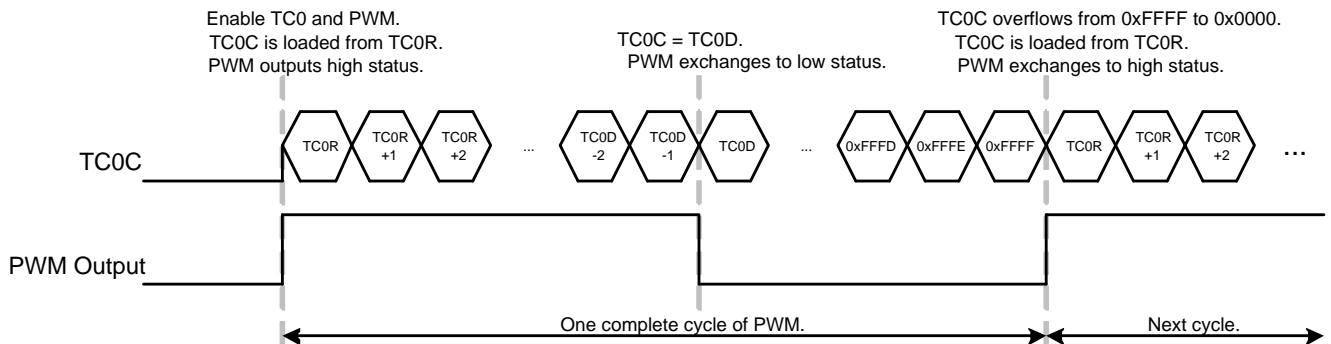
### 8.5.4 TC0 EVENT COUNTER

TC0 event counter is set the TC0 clock source from external input pin (P0.2). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.2). TC0 event counter trigger direction is falling edge. When one falling edge occurs, TC0C will up one count. When TC0C counts from 0xFFFF to 0x0000, TC0 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC0 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC0 event counter function enabled, and the P00IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal...These signal phase don't synchronize with MCU's main clock. Use TC0 event to measure it and calculate the signal rate in program for different applications.



### 8.5.5 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC0 timer enables and PWM0OUT bit sets as "1" (enable PWM output), the PWM output pin (P0.6) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC0R register controls the cycle of PWM, and TC0D decides the duty (high pulse width length) of PWM. TC0C initial value is TC0R reloaded when TC0 timer enables and TC0 timer overflows. When TC0C count is equal to TC0D, the PWM high pulse finishes and exchanges to low level. When TC0 overflows (TC0C counts from 0xFFFF to 0x0000), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC0C from TC0R automatically when TC0 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM cycle by program as PWM outputting, the new cycle occurs at next cycle when TC0C loaded from TC0R.



## 8.5.6 TC0 TIMER OPERATION EXAMPLE

### ● TC0 TIMER CONFIGURATION:

; Reset TC0 timer.

```
CLR          TC0M          ; Clear TC0M register.
```

; Set TC0 clock source and TC0 rate.

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

; Set TC0C and TC0R register for TC0 Interval time.

```
MOV          A, #value_A    ;TC0CL/TC0CH must be equal to TC0RH/TC0RL respectively.
B0MOV        TC0CH, A
B0MOV        TC0RH, A
MOV          A, #value_B
B0MOV        TC0CL, A
B0MOV        TC0RL, A
```

; Clear TC0IRQ

```
B0BCLR      FTC0IRQ
```

; Enable TC0 timer and interrupt function.

```
B0BSET      FTC0IEN        ; Enable TC0 interrupt function.
B0BSET      FTC0ENB        ; Enable TC0 timer.
```

### ● TC0 EVENT COUNTER CONFIGURATION:

; Reset TC0 timer.

```
CLR          TC0M          ; Clear TC0M register.
```

; Enable TC0 event counter.

```
B0BSET      FTC0CKS1        ; Set TC0 clock source from external input pin (P0.0).
```

; Set TC0C and TC0R register for TC0 Interval time.

```
MOV          A, # value_A    ; TC0CL/TC0CH must be equal to TC0RH/TC0RL respectively.
B0MOV        TC0C, A
B0MOV        TC0R, A
MOV          A, # value_B
B0MOV        TC0C, A
B0MOV        TC0R, A
```

; Clear TC0IRQ

```
B0BCLR      FTC0IRQ
```

; Enable TC0 timer and interrupt function.

```
B0BSET      FTC0IEN        ; Enable TC0 interrupt function.
B0BSET      FTC0ENB        ; Enable TC0 timer.
```

● **TC0 PWM CONFIGURATION:**

**; Reset TC0 timer.**

```
CLR          TC0M          ; Clear TC0M register.
```

**; Set TC0 clock source and TC0 rate.**

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

**; Set TC0C and TC0R register for PWM cycle.**

```
MOV          A, #value1_H          ; TC0CL/TC0CH must be equal to TC0RH/TC0RL respectively.
B0MOV        TC0CH, A
B0MOV        TC0RH, A
MOV          A, #value1_L
B0MOV        TC0CL, A
B0MOV        TC0RL, A
```

**; Set TC0D register for PWM duty.**

```
MOV          A, #value2_H          ; TC0DH/ TC0DL must be greater than TC0RH/ TC0RL.
B0MOV        TC0DH, A
MOV          A, #value2_L
B0MOV        TC0DL, A
```

**; Enable PWM and TC0 timer.**

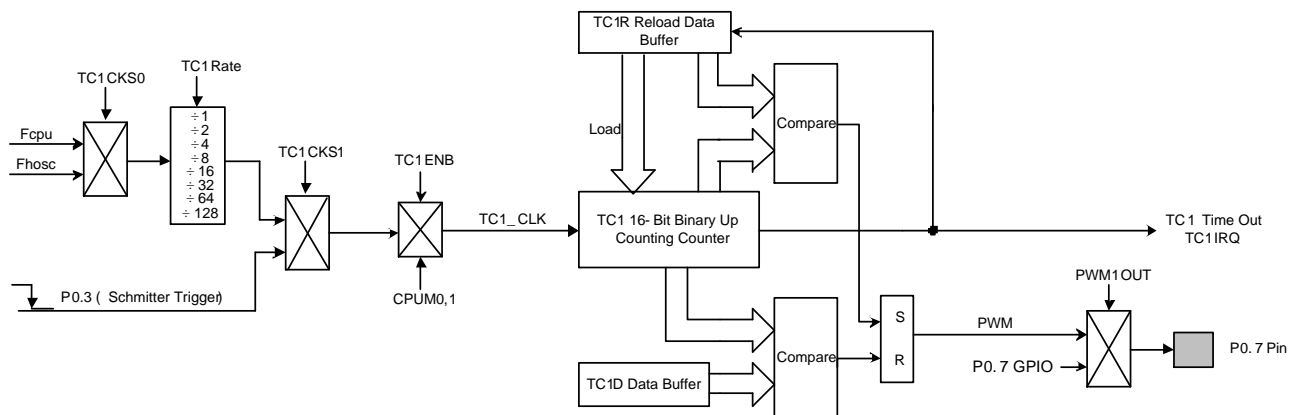
```
B0BSET       FPWM0OUT          ; Enable PWM.
B0BSET       FTC0ENB          ; Enable TC0 timer.
```

## 8.6 TC1 16-BIT TIMER/COUNTER

### 8.6.1 OVERVIEW

The TC1 timer is an 16-bit binary up timer with basic timer, event counter and PWM functions. The basic timer function supports flag indicator (TC1IRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TC1M, TC1C, TC1R registers. The event counter is changing TC1 clock source from system clock (Fcpu/Fhosc) to external clock like signal (e.g. continuous pulse, R/C type oscillating signal...). TC1 becomes a counter to count external clock number to implement measure application. TC1 also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TC1 timer clock rate, TC1R and TC1D registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster... The main purposes of the TC1 timer is as following.

- ☞ **16-bit programmable up counting timer:** Generate time-out at specific time intervals based on the selected clock frequency.
- ☞ **Interrupt function:** TC1 timer function supports interrupt function. When TC1 timer occurs overflow, the TC1IRQ activates and the system points program counter to interrupt vector to do interrupt sequence.
- ☞ **Event Counter:** The event counter function counts the external clock counts.
- ☞ **Duty/cycle programmable PWM:** The PWM is duty/cycle programmable controlled by TC1R and TC1D registers.
- ☞ **Green mode function:** All TC1 functions (timer, PWM, event counter, auto-reload) keep running in green mode and no wake-up function.





## 8.6.2 TC1M MODE REGISTER

TC1M is TC1 timer mode control register to configure TC1 operating mode including TC1 pre-scalar, clock source, PWM function...These configurations must be setup completely before enabling TC1 timer.

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1M</b>	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS1	TC1CKS0	PWM1OUT	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
After reset	0	0	0	0	0	0	0	-

- Bit 1     **PWM1OUT**: PWM output control bit.  
0 = Disable PWM output function, and P0.7 is GPIO mode.  
1 = Enable PWM output function, and P0.7 outputs PWM signal.
- Bit 2     **TC1CKS0**: TC1 clock source select bit.  
0 = Fcpu.  
1 = Fhosc.
- Bit 3     **TC1CKS1**: TC1 clock source select bit.  
0 = Internal clock (Fcpu and Fhosc controlled by TC1CKS0 bit).  
1 = External input pin (P0.3) and enable event counter function. **TC0rate[2:0] bits are useless.**
- Bit [6:4]     **TC1RATE[2:0]**: TC1 timer clock source select bits.  
TC1 clock source controlled by TC1RATE[2:0] and TC1CKS[1:0]

TC1CKS[1:0]	TC1RATE[2:0]	TC1 Clock	TC1CKS[1:0]	TC1RATE[2:0]	TC1 Clock
00	000	Fcpu / 128	01	000	Fhosc / 128
00	001	Fcpu / 64	01	001	Fhosc / 64
00	010	Fcpu / 32	01	010	Fhosc / 32
00	011	Fcpu / 16	01	011	Fhosc / 16
00	100	Fcpu / 8	01	100	Fhosc / 8
00	101	Fcpu / 4	01	101	Fhosc / 4
00	110	Fcpu / 2	01	110	Fhosc / 2
00	111	Fcpu / 1	01	111	Fhosc / 1
10	x	P0.3	11	x	P0.3

- Bit 7     **TC1ENB**: TC1 counter control bit.  
0 = Disable TC1 timer.  
1 = Enable TC1 timer.

### 8.6.3 TC1C COUNTING REGISTER

TC1C is TC1 16-bit counter. When TC1C overflow occurs, the TC1IRQ flag is set as “1” and cleared by program. The TC1C decides TC1 interval time through below equation to calculate a correct value. It is necessary to write the correct value to TC1C register and TC1R register first time, and then enable TC1 timer to make sure the first cycle correct. After one TC1 overflow occurs, the TC1C register is loaded a correct value from TC1R register automatically, not program.

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1CL</b>	TC1CL7	TC1CL6	TC1CL5	TC1CL4	TC1CL3	TC1CL2	TC1CL1	TC1CL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **TC1CL [7:0]**: TC1 timer Low Byte counter.

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1CH</b>	TC1CH7	TC1CH6	TC1CH5	TC1CH4	TC1CH3	TC1CH2	TC1CH1	TC1CH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **TC1CH [7:0]**: TC1 timer High Byte counter.

The equation of TC1C initial value is as following.

$$TC1C \text{ initial value} = 65536 - (TC1 \text{ interrupt interval time} * TC1 \text{ clock rate})$$

### 8.6.4 TC1R AUTO-RELOAD REGISTER

TC1 timer builds in auto-reload function, and TC1R register stores reload data. When TC1C overflow occurs, TC1C register is loaded data from TC1R register automatically. Under TC1 timer counting status, to modify TC1 interval time is to modify TC1R register, not TC1C register. New TC1C data of TC1 interval time will be updated after TC1 timer overflow occurrence, TC1R loads new value to TC1C register. But at the first time to setup T0M, TC1C and TC1R must be set the same value before enabling TC1 timer. TC1 is double buffer design. If new TC1R value is set by program, the new value is stored in 1<sup>st</sup> buffer. Until TC1 overflow occurs, the new value moves to real TC1R buffer. This way can avoid any transitional condition to affect the correctness of TC1 interval time and PWM output signal.

CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1RL</b>	TC1RL7	TC1RL6	TC1RL5	TC1RL4	TC1RL3	TC1RL2	TC1RL1	TC1RL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC1RL [7:0]: TC0 timer counter reload buffer low byte.

C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1RH</b>	TC1RH7	TC1RH6	TC1RH5	TC1RH4	TC1RH3	TC1RH2	TC1RH1	TC1RH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC1RH [7:0]: TC1 timer counter reload buffer high byte.

The equation of TC1R initial value is as following.

$$TC1R \text{ initial value} = 65536 - (TC1 \text{ interrupt interval time} * TC1 \text{ clock rate})$$

**Example: To calculation TC1C and TC1R value to obtain 10ms TC1 interval time. TC1 clock source is Fcpu = 8MHz/8 = 1MHz. Select TC1RATE=000 (Fcpu/128).**

TC1 interval time = 10ms. TC1 clock rate = 8MHz/16/128

$$\begin{aligned}
 TC1C/TC1R \text{ initial value} &= 65536 - (TC1 \text{ interval time} * \text{input clock}) \\
 &= 65536 - (10\text{ms} * 8\text{MHz} / 8 / 128) \\
 &= 65536 - (10^{-2} * 8 * 10^6 / 8 / 128) \\
 &= \text{FFB2H}
 \end{aligned}$$

## 8.6.5 TC1D PWM DUTY REGISTER

TC1D register's purpose is to decide PWM duty. In PWM mode, TC1R controls PWM's cycle, and TC1D controls the duty of PWM. The operation is based on timer counter value. When TC1C = TC1D, the PWM high duty finished and exchange to low level. It is easy to configure TC1D to choose the right PWM's duty for application.

CEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1DL</b>	TC1DL7	TC1DL6	TC1DL5	TC1DL4	TC1DL3	TC1DL2	TC1DL1	TC1DL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

CFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1DH</b>	TC1DH7	TC1DH6	TC1DH5	TC1DH4	TC1DH3	TC1DH2	TC1DH1	TC1DH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] TC1DL [7:0]: PWM duty control low byte buffer.

Bit [7:0] TC1DH [7:0]: PWM duty control high byte buffer.

The equation of TC1D initial value is as following.

$$TC1D \text{ initial value} = TC1R + (PWM \text{ high pulse width period} / TC1 \text{ clock rate})$$

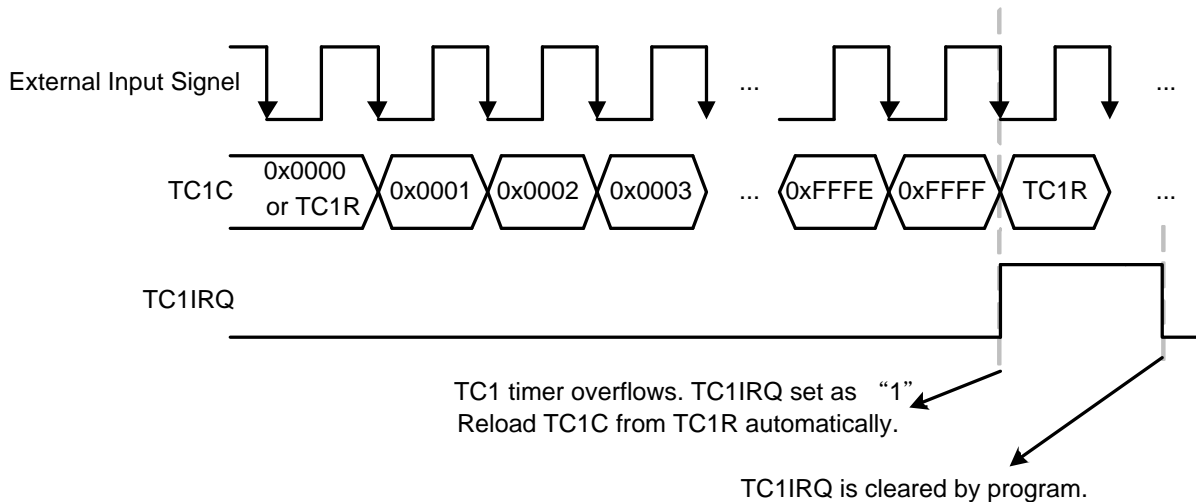
**Example: To calculate TC1D value to obtain 1/3 duty PWM signal. The TC1 clock source is Fcpu = 8MHz/8 = 1MHz. Select TC1RATE=000 (Fcpu/64).**

TC1R = 63H. TC1 interval time = 3.33ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 2ms.

$$\begin{aligned}
 TC1D \text{ initial value} &= FF63H + (PWM \text{ high pulse width period} / TC1 \text{ clock rate}) \\
 &= FF63H + (3.33ms * 8MHz / 8 / 128) \\
 &= FF63H + 34H \\
 &= FF97H
 \end{aligned}$$

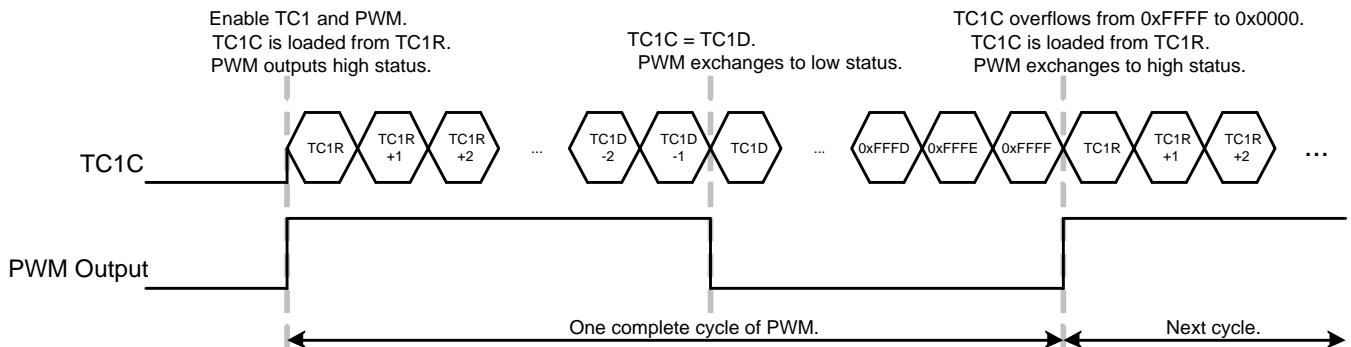
### 8.6.6 TC1 EVENT COUNTER

TC1 event counter is set the TC1 clock source from external input pin (P0.3). When TC1CKS1=1, TC1 clock source is switch to external input pin (P0.3). TC1 event counter trigger direction is falling edge. When one falling edge occurs, TC1C will up one count. When TC1C counts from 0xFFFF to 0x0000, TC1 triggers overflow event. The external event counter input pin's wake-up function of GPIO mode is disabled when TC1 event counter function enabled to avoid event counter signal trigger system wake-up and not keep in power saving mode. The external event counter input pin's external interrupt function is also disabled when TC1 event counter function enabled, and the P01IRQ bit keeps "0" status. The event counter usually is used to measure external continuous signal rate, e.g. continuous pulse, R/C type oscillating signal...These signal phase don't synchronize with MCU's main clock. Use TC1 event to measure it and calculate the signal rate in program for different applications.



### 8.6.7 PULSE WIDTH MODULATION (PWM)

The PWM is duty/cycle programmable design to offer various PWM signals. When TC1 timer enables and PWM1OUT bit sets as "1" (enable PWM output), the PWM output pin (P0.7) outputs PWM signal. One cycle of PWM signal is high pulse first, and then low pulse outputs. TC1R register controls the cycle of PWM, and TC1D decides the duty (high pulse width length) of PWM. TC1C initial value is TC1R reloaded when TC1 timer enables and TC1 timer overflows. When TC1C count is equal to TC1D, the PWM high pulse finishes and exchanges to low level. When TC1 overflows (TC1C counts from 0xFFFF to 0x0000), one complete PWM cycle finishes. The PWM exchanges to high level for next cycle. The PWM is auto-reload design to load TC1C from TC1R automatically when TC1 overflows and the end of PWM's cycle, to keeps PWM continuity. If modify the PWM cycle by program as PWM outputting, the new cycle occurs at next cycle when TC1C loaded from TC1R.



## 8.6.8 TC1 TIMER OPERATION EXAMPLE

### ● TC1 TIMER CONFIGURATION:

; Reset TC1 timer.

```
CLR          TC1M          ; Clear TC1M register.
```

; Set TC1 clock source and TC1 rate.

```
MOV          A, #0nnn0n00b
B0MOV        TC1M, A
```

; Set TC1C and TC1R register for TC1 Interval time.

```
MOV          A, #value_H    ; TC1CL/TC1CH must be equal to TC1RH/TC1RL respectively.
B0MOV        TC1CH, A
B0MOV        TC1RH, A
MOV          A, #value_L
B0MOV        TC1CL, A
B0MOV        TC1RL, A
```

; Clear TC1IRQ

```
B0BCLR      FTC1IRQ
```

; Enable TC1 timer and interrupt function.

```
B0BSET      FTC1IEN        ; Enable TC1 interrupt function.
B0BSET      FTC1ENB        ; Enable TC1 timer.
```

### ● TC1 EVENT COUNTER CONFIGURATION:

; Reset TC1 timer.

```
CLR          TC1M          ; Clear TC1M register.
```

; Enable TC1 event counter.

```
B0BSET      FTC1CKS1        ; Set TC1 clock source from external input pin (P0.3).
```

; Set TC1C and TC1R register for TC1 Interval time.

```
MOV          A, #value_H    TC0CL/TC0CH must be equal to TC0RH/TC0RL respectively.
B0MOV        TC1CH, A
B0MOV        TC1RH, A
MOV          A, #value_L
B0MOV        TC1CL, A
B0MOV        TC1RL, A
```

; Clear TC1IRQ

```
B0BCLR      FTC1IRQ
```

; Enable TC1 timer and interrupt function.

```
B0BSET      FTC1IEN        ; Enable TC1 interrupt function.
B0BSET      FTC1ENB        ; Enable TC1 timer.
```

● **TC1 PWM CONFIGURATION:**

**; Reset TC1 timer.**

```
CLR          TC1M          ; Clear TC1M register.
```

**; Set TC1 clock source and TC1 rate.**

```
MOV          A, #0nnn0n00b
B0MOV        TC1M, A
```

**; Set TC1C and TC1R register for PWM cycle.**

```
MOV          A, #value1_H
B0MOV        TC1CH, A
B0MOV        TC1RH, A
MOV          A, #value1_L
B0MOV        TC1CL, A
B0MOV        TC1RL, A
```

**TC0CH/TC0CL must be equal to TC0RH/TC0RL respectively.**

**; Set TC1D register for PWM duty.**

```
MOV          A, #value2_H
B0MOV        TC1DH, A
MOV          A, #value2_L
B0MOV        TC1DL, A
```

**; TC1DH/ TC1DL must be greater than TC1RH/ TC1RL.**

**; Enable PWM and TC1 timer.**

```
B0BSET       FPWM1OUT      ; Enable PWM.
B0BSET       FTC1ENB       ; Enable TC1 timer.
```



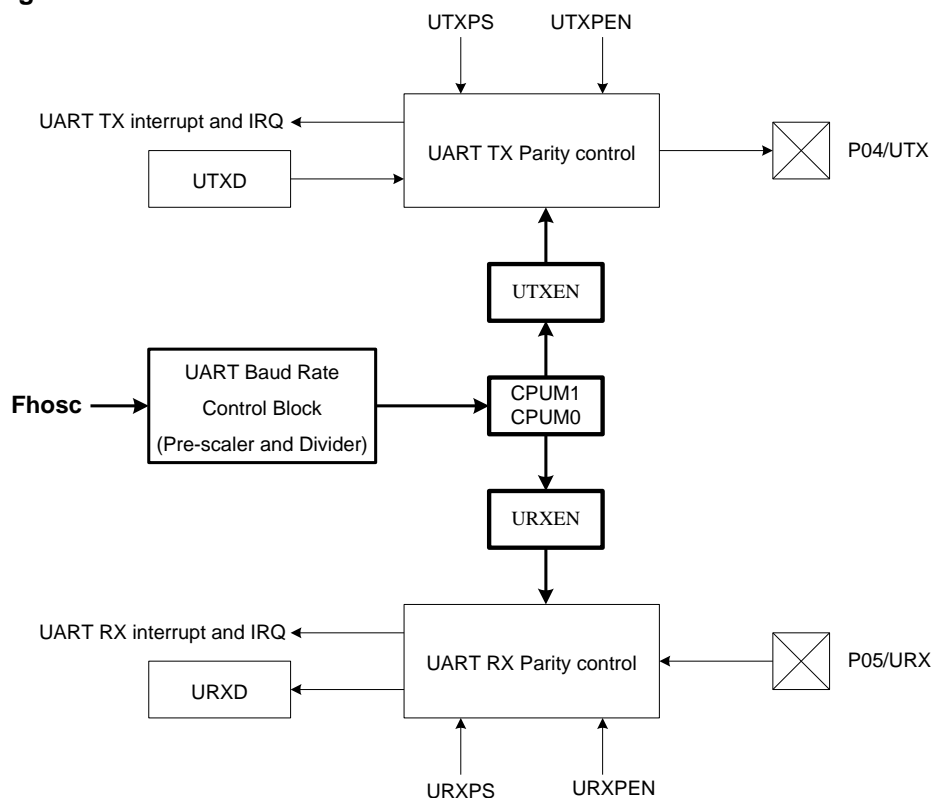
# 9 UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER(UART)

## 9.1 OVERVIEW

The UART interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART transceiver of Sonix 8-bit MCU allows RS232 standard and supports one byte data length. The transfer format has start bit, 8-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART I/O pins support push-pull and open-drain structures controlled by register.

The UART features include the following:

- ☞ Full-duplex, 2-wire asynchronous data transfer.
- ☞ Programmable baud rate.
- ☞ 8-bit data length.
- ☞ Odd and even parity bit.
- ☞ End-of-Transfer interrupt.
- ☞ Support break pocket function.
- ☞ Support wide range baud rate.



UART Interface Structure Diagram

## **9.2 UART OPERATION**

The UART RX and TX pins are shared with GPIO. When UART enables (URXEN=1, UTXEN=1), the UART shared pins transfers to UART purpose and disable GPIO function automatically. When UART disables, the UART pins returns to GPIO last status. The UART data buffer length supports 1-byte.

The UART supports interrupt function. URXIEN/UTXIEN are UART transfer interrupt function control bit. URXIEN=0, disable UART receiver interrupt function. UTXIEN=0, disable UART transmitter interrupt function. URXIEN=1, enable UART receiver interrupt function. UTXIEN=1, enable UART transmitter interrupt function. When UART interrupt function enable, the program counter points to interrupt vector (ORG 0014H/0016H) to do UART interrupt service routine after UART operating. URXIRQ/UTXIRQ is UART interrupt request flag, and also to be the UART operating status indicator when URXIEN=0 or UTXIEN=0, but cleared by program. When UART operation finished, the URXIRQ/UTXIRQ would be set to "1".

The UART also builds in "Busy Bit" to indicate UART bus status. URXBZ bit is UART RX operation indicator. UTXBZ bit is UART TX operation indicator. If bus is transmitting, the busy bit is "1" status. If bus is finishing operation or in idle status, the busy bit is "0" status.

UART TX operation is controlled by loading UTXD data buffer. After UART TX configuration, load transmitted data into UTXD 8-bit buffer, and then UART starts to transmit the packet following UART TX configuration.

UART RX operation is controlled by receiving the start bit from master terminal. After UART RX configuration, UART RX pin detects the falling edge of start bit, and then UART starts to receive the packet from master terminal.

UART provides URXPC bit and UFMER bit to check received packet. URXPC bit is received parity bit checker. If received parity is error, URXPC sets as "1". If URXPC bit is zero after receiving packet, the parity is correct. UFMER bit is received stream frame checker. The stream frame error definition includes "Start bit error", "Stop bit error", "Stream length error", "UART baud rate error"... Each of frame error conditions makes UFMER bit sets as "1" after receiving packet.

## 9.3 UART BAUD RATE

UART clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART clock source is generated from system oscillator called Fosc. Fosc passes through UART pre-scaler to get UART main clock called Fuart. UART pre-scaler has 8 selections (Fosc/1, Fosc/2, Fosc/4, Fosc/8, Fosc/16, Fosc/32, Fosc/64, Fosc/128) and 3-bit control bits (URS[2:0]). UART main clock (Fuart) purposes are the front-end clock and through UART 8-bit buffer (URCR) to obtain UART processing clock and decide UART baud rate.

UART Pre-scaler Selection, URS[2:0]	UART Main Clock Rate	Fuart (Fosc=8MHz)
000b	Fosc/1	8MHz
001b	Fosc/2	4MHz
010b	Fosc/4	2MHz
011b	Fosc/8	1MHz
100b	Fosc/16	0.5MHz
101b	Fosc/32	0.25MHz
110b	Fosc/64	0.125MHz
111b	Fosc/128	0.0625MHz

0D7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URCR	URCR7	URCR6	URCR5	URCR4	URCR3	URCR2	URCR1	URCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The UART baud rate clock source is Fosc and divided by pre-scaler. The equation is as following.

$$\text{UART Baud Rate} = 1/2 * (\text{Fuart} * 1/(256 - \text{URCR})) \dots \text{bps}$$

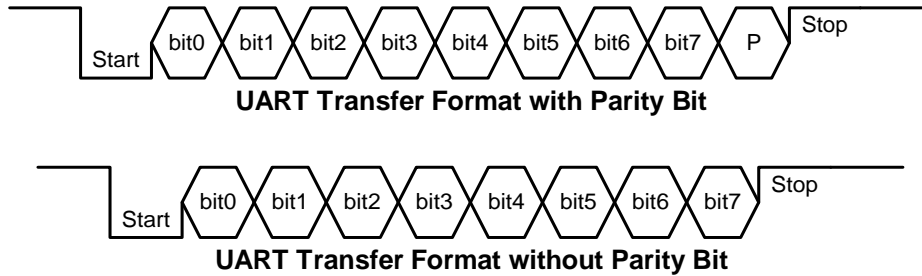
Fosc = 8MHz

Baud Rate	UART Pre-scaler	URS[2:0]	URCR (Hex)	Accuracy (%)
1200	Fosc/32	101b	98	0.16%
2400	Fosc/32	101b	CC	0.16%
4800	Fosc/32	101b	E6	0.16%
9600	Fosc/32	101b	F3	0.16%
19200	Fosc/32	101b	F9	0.16%
38400	Fosc/1	000b	98	0.16%
51200	Fosc/1	000b	B2	0.16%
57600	Fosc/1	000b	BB	0.64%
102400	Fosc/1	000b	DA	0.16%
115200	Fosc/1	000b	DD	-0.79%

**Note: We strongly recommend not to set URCR = 0xFF, or UART operation would be error.**

## 9.4 UART TRANSFER FORMAT

The UART transfer format includes “Bus idle status”, “Start bit”, “8-bit Data”, “Parity bit” and “Stop bit” as following.



**Bus Idle Status:** The bus idle status is the bus non-operating status. The UART receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART transmitter bus idle status of MCU is high status. The UART bus will be set when URXEN and UTXEN are enabled.

**Start Bit:** UART is an asynchronous type of communication and needs an attention bit to offer the receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

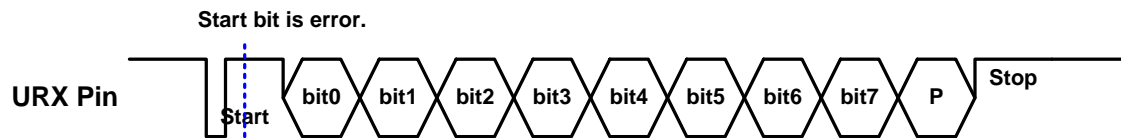
**8-bit Data:** The data format is 8-bit length, and LSB transfers first following the start bit. The one bit data duration is the unit of UART baud rate controlled by the register.

**Parity Bit:** The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URXPS/UTXPS bits. After receiving data and parity bit, the parity check executes automatically. The URXPC bit indicates the parity check result. The parity bit function is controlled by URXPEN/UTXPEN bits. If the parity bit function is disabled, the UART transfer contents remove the parity bit and the stop bit follows the data stream directly.

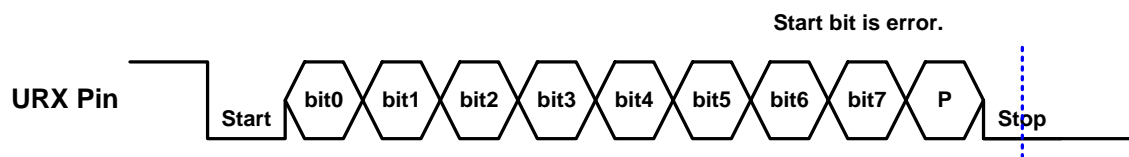
**Stop Bit:** The stop bit is like the start bit using a simple format to indicate the end of UART transfer. The stop bit format is low to high edge change and the duration is one bit period.

## 9.5 ABNORMAL POCKET

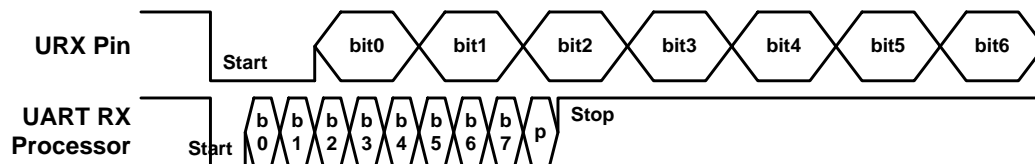
The abnormal pocket occurs in UART RX mode. Break pocket is one abnormal pocket of the UART architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error...When UART receives abnormal pocket, the UFMER bit will be set "1", and UART issues URXIRQ. The system finds the abnormal pocket through firmware. UART changes to initial status until detecting next start bit.



UART check the start bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



UART check the stop bit is error and issue UFMER flag, but the UART still finishes receiving the pocket.



If the host's UART baud rate isn't match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART rule, but the data is error. Use checking UFMER bit and URXPC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART will accept the pocket as correct one.

## 9.6 UART RECEIVER CONTROL REGISTER

0D6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URRX</b>	URXEN	URXPEN	URXPS	URXPC	UFMER	URS2	URS1	URS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 7 **URXEN**: UART RX control bit.  
0 = Disable UART RX. URX pin is GPIO mode or returns to GPIO status.  
1 = Enable UART RX. URX pin exchanges from GPIO mode to UART RX mode.
- Bit 6 **URXPEN**: UART RX parity bit control bit.  
0 = Disable UART RX parity bit function. The data stream doesn't include parity bit.  
1 = Enable UART RX parity bit function. The data stream includes parity bit.
- Bit 5 **UTXPS**: UART RX parity bit format control bit.  
0 = UART RX parity bit format is even parity.  
1 = UART RX parity bit format is odd parity.
- Bit 4 **URXPC**: UART RX parity bit checking flag.  
0 = Parity bit is correct or no parity function.  
1 = Parity bit is error.
- Bit 3 **UFMER**: UART RX stream frame error flag bit.  
0 = Collect UART frame.  
1 = UART frame is error including start/stop bit, stream length.
- Bit [2:0] **URS[2:0]**: UART per-scalar select bit.  
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32,  
110 = Fhosc/64, 111 = Fhosc/128.

## 9.7 UART TRANSMITTER CONTROL REGISTER

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URT<sub>X</sub></b>	UTXEN	UTXPEN	UTXPS	-	URXBZ	UTXBZ	-	-
Read/Write	R/W	R/W	R/W	-	R	R	-	-
After reset	0	0	0	-	0	0	-	-

- Bit 7 **UTXEN**: UART TX control bit.  
 0 = Disable UART TX. UTX pin is GPIO mode or returns to GPIO status.  
 1 = Enable UART TX. UTX pin exchanges from GPIO mode to UART TX mode and idle high status.
- Bit 6 **UTXPEN**: UART TX parity bit control bit.  
 0 = Disable UART TX parity bit function. The data stream doesn't include parity bit.  
 1 = Enable UART TX parity bit function. The data stream includes parity bit.
- Bit 5 **UTXPS**: UART TX parity bit format control bit.  
 0 = UART TX parity bit format is even parity.  
 1 = UART TX parity bit format is odd parity.
- Bit 3 **URXBZ**: UART RX operating status flag.  
 0 = UART RX is idle or the end of processing.  
 1 = UART RX is busy and processing.
- Bit 2 **UTXBZ**: UART TX operating status flag.  
 0 = UART TX is idle or the end of processing.  
 1 = UART TX is busy and processing.

**Note:** URXBZ and UTXBZ bits are UART operating indicators. After setting UART RX/TX operations, set (2\*Fcpu/Fuart)\*NOP instruction is necessary, and then check UART status through URXBZ and UTXBZ bits.

## 9.8 UART DATA BUFFER

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>UTXD</b>	UTXD7	UTXD6	UTXD5	UTXD4	UTXD3	UTXD2	UTXD1	UTXD0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **UTXD**: UART transmitted data buffer.

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URXD</b>	URXD7	URXD6	URXD5	URXD4	URXD3	URXD2	URXD1	URXD0
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **URXD**: UART received data buffer.

## 9.9 UART OPERATION EXAMLPE

### UART TX Configuration:

```

; Select parity bit function.
      B0BCLR      FUTXPEN      ; Disable UART TX parity bit function.
;or
      B0BSET      FUTXPEN      ; Enable UART TX parity bit function.

; Select parity bit format.
      B0BCLR      FUTXPS       ; UART TX parity bit format is even parity.
;or
      B0BSET      FUTXPS       ; UART TX parity bit format is odd parity.

; Set UART baud rate.
      MOV         A, #value1    ; Set UART pre-scaler URS[2:0].
      B0MOV       URRX, A
      MOV         A, #value2    ; Set UART baud rate 8-bit buffer.
      B0MOV       URCR, A

; Enable UART TX pin.
      B0BSET      FUTXEN       ; Enable UART TX function and UART TX pin.

; Enable UART TX interrupt function.
      B0BCLR      FUTXIRQ      ; Clear UART TX interrupt flag.
      B0BSET      FUTXIEN      ; Enable UART TX interrupt function.

; Load TX data buffer and execute TX transmitter.
      MOV         A, #value3    ; Load 8-bit data to UTXD data buffer.
      B0MOV       UTXD, A
                                   ;After loading UTXD, UART TX starts to transmit.
      NOP                                   ; One instruction delay for UTXBZ flag.

; Check TX operation.
      B0BTS0      FUTXBZ       ; Check UTXBZ bit.
      JMP         CHKTX        ; UTXBZ=1, TX is operating.
      JMP         ENDTX        ; UTXBZ=0, the end of TX.

```

\* **Note:** UART TX operation is started through loading UTXD data buffer.

**Transmit Break Pocket:**
**; Select parity bit function.**

B0BCLR	FUTXPEN	; Disable UART TX parity bit function.
--------	---------	--

**;or**

B0BSET	FUTXPEN	; Enable UART TX parity bit function.
--------	---------	---------------------------------------

**; Select parity bit format.**

B0BCLR	FUTXPS	; UART TX parity bit format is even parity.
--------	--------	---

**;or**

B0BSET	FUTXPS	; UART TX parity bit format is odd parity.
--------	--------	--

**; Set UART baud rate.**

MOV	A, #value1	; Set UART pre-scaler URS[2:0].
-----	------------	---------------------------------

B0MOV	URRX, A	
-------	---------	--

MOV	A, #value2	; Set UART baud rate 8-bit buffer.
-----	------------	------------------------------------

B0MOV	URCR, A	
-------	---------	--

**; Enable UART TX pin.**

B0BSET	FUTXEN	; Enable UART TX function and UART TX pin.
--------	--------	--

**; Enable UART TX interrupt function.**

B0BCLR	FUTXIRQ	; Clear UART TX interrupt flag.
--------	---------	---------------------------------

B0BSET	FUTXIEN	; Enable UART TX interrupt function.
--------	---------	--------------------------------------

**; Start UART break pocket.**

B0BSET	FUTXBRK	; Transmit UART break pocket.
--------	---------	-------------------------------

NOP		; One instruction delay for UTXBZ flag.
-----	--	---

**; Check TX operation.**

B0BTS0	FUTXBZ	; Check UTXBZ bit.
--------	--------	--------------------

JMP	CHKTx	; UTXBZ=1, TX is operating.
-----	-------	-----------------------------

JMP	ENDTX	; UTXBZ=0, the end of TX.
-----	-------	---------------------------

**Note: UART TX break pocket is controlled by UTXBRK bit and needn't load UTXD buffer.**



**UART RX Configuration:****; Select parity bit function.**

B0BCLR	FURXPEN	; Disable UART RX parity bit function.
--------	---------	--

;or

B0BSET	FURXPEN	; Enable UART RX parity bit function.
--------	---------	---------------------------------------

**; Select parity bit format.**

B0BCLR	FURXPS	; UART RX parity bit format is even parity.
--------	--------	---

;or

B0BSET	FURXPS	; UART RX parity bit format is odd parity.
--------	--------	--

**; Set UART baud rate.**

MOV	A, #value1	; Set UART pre-scaler URS[2:0].
-----	------------	---------------------------------

B0MOV	URRX, A	
-------	---------	--

MOV	A, #value2	; Set UART baud rate 8-bit buffer.
-----	------------	------------------------------------

B0MOV	URCR, A	
-------	---------	--

**; Enable UART RX pin.**

B0BSET	FURXEN	; Enable UART RX function and UART RX pin.
--------	--------	--

**; Enable UART RX interrupt function.**

B0BCLR	FURXIRQ	; Clear UART RX interrupt flag.
--------	---------	---------------------------------

B0BSET	FURXIEN	; Enable UART RX interrupt function.
--------	---------	--------------------------------------

NOP		; One instruction delay for URXBZ flag.
-----	--	---

**; Check RX operation.**

B0BTS0	FURXBZ	; Check URXBZ bit.
--------	--------	--------------------

JMP	CHKRX	; URXBZ=1, RX is operating.
-----	-------	-----------------------------

JMP	ENDRX	; URXBZ=0, the end of RX.
-----	-------	---------------------------

**Note: UART RX operation is started as start bit transmitted from master terminal.**

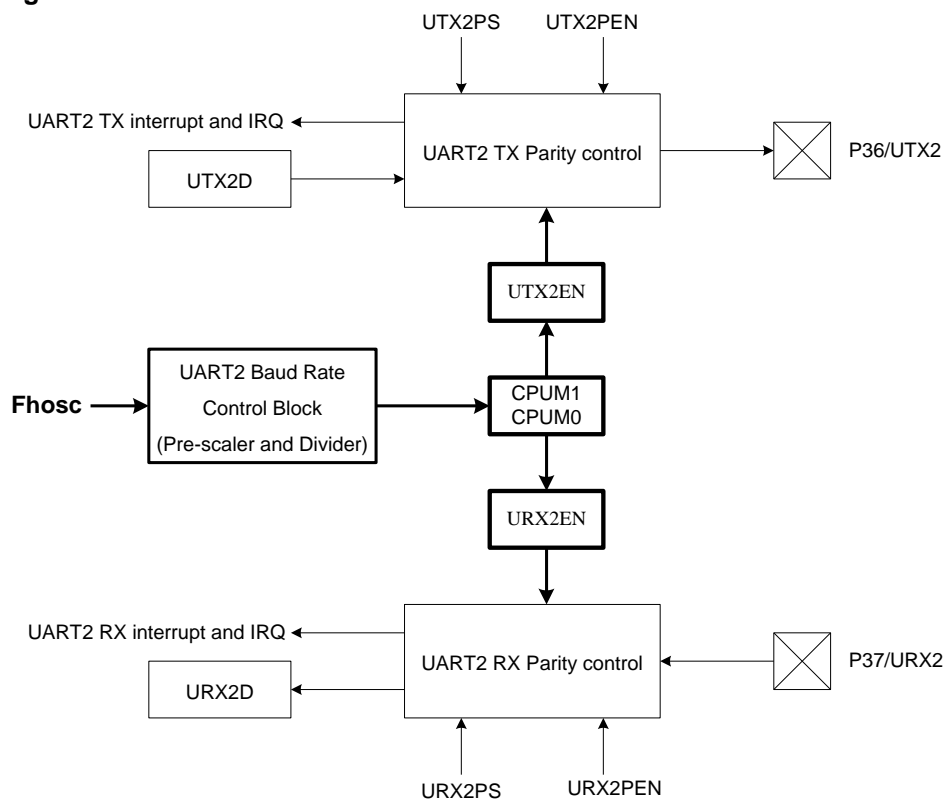
# 10 UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER (UART2)

## 10.1 OVERVIEW

The UART2 interface is an universal asynchronous receiver/transmitter method. The serial interface is applied to low speed data transfer and communicate with low speed peripheral devices. The UART2 transceiver of Sonix 8-bit MCU allows RS232 standard and supports one byte data length. The transfer format has start bit, 8-bit data, parity bit and stop bit. Programmable baud rate supports different speed peripheral devices. UART2 I/O pins support push-pull and open-drain structures controlled by register.

The UART2 features include the following:

- ☞ Full-duplex, 2-wire asynchronous data transfer.
- ☞ Programmable baud rate.
- ☞ 8-bit data length.
- ☞ Odd and even parity bit.
- ☞ End-of-Transfer interrupt.
- ☞ Support break pocket function.
- ☞ Support wide range baud rate.



UART2 Interface Structure Diagram

## 10.2 UART2 OPERATION

The UART2 RX and TX pins are shared with GPIO. When UART2 enables (URX2EN=1, UTX2EN=1), the UART2 shared pins transfers to UART2 purpose and disable GPIO function automatically. When UART2 disables, the UART2 pins returns to GPIO last status. The UART2 data buffer length supports 1-byte.

The UART2 supports interrupt function. URX2IEN/UTX2IEN are UART2 receive/transfer interrupt function control bit. URX2IEN=0, disable UART2 receiver interrupt function. UTX2IEN=0, disable UART2 transmitter interrupt function. URX2IEN=1, enable UART2 receiver interrupt function. UTX2IEN=1, enable UART2 transmitter interrupt function. When UART2 interrupt function enable, the program counter points to interrupt vector (ORG 001AH/001CH) to do UART2 interrupt service routine after UART2 operating. URX2IRQ/UTX2IRQ is UART2 interrupt request flag, and also to be the UART2 operating status indicator when URX2IEN=0 or UTX2IEN=0, but cleared by program. When UART2 operation finished, the URX2IRQ/UTX2IRQ would be set to "1".

The UART2 also builds in "Busy Bit" to indicate UART2 bus status. URX2BZ bit is UART2 RX operation indicator. UTX2BZ bit is UART2 TX operation indicator. If bus is transmitting, the busy bit is "1" status. If bus is finishing operation or in idle status, the busy bit is "0" status.

UART2 TX operation is controlled by loading UTX2D data buffer. After UART2 TX configuration, load transmitted data into UTX2D 8-bit buffer, and then UART2 starts to transmit the packet following UART2 TX configuration.

UART2 RX operation is controlled by receiving the start bit from master terminal. After UART2 RX configuration, UART2 RX pin detects the falling edge of start bit, and then UART2 starts to receive the packet from master terminal.

UART2 provides URX2PC bit and UFMER2 bit to check received packet. URX2PC bit is received parity bit checker. If received parity is error, URX2PC sets as "1". If URX2PC bit is zero after receiving packet, the parity is correct. UFMER2 bit is received stream frame checker. The stream frame error definition includes "Start bit error", "Stop bit error", "Stream length error", "UART2 baud rate error"... Each of frame error conditions makes UFMER2 bit sets as "1" after receiving packet.

## 10.3 UART2 BAUD RATE

UART2 clock is 2-stage structure including a pre-scaler and an 8-bit buffer. UART2 clock source is generated from system oscillator called Fhosc. Fhosc passes through UART2 pre-scaler to get UART2 main clock called Fuart2. UART2 pre-scaler has 8 selections (Fhosc/1, Fhosc/2, Fhosc/4, Fhosc/8, Fhosc/16, Fhosc/32, Fhosc/64, Fhosc/128) and 3-bit control bits (URS[2:0]). UART2 main clock (Fuart2) purposes are the front-end clock and through UART2 8-bit buffer (URCR2) to obtain UART2 processing clock and decide UART2 baud rate.

UART2 Pre-scaler Selection, UR2S[2:0]	UART2 Main Clock Rate	Fuart2 (Fhosc=8MHz)
000b	Fhosc/1	8MHz
001b	Fhosc/2	4MHz
010b	Fhosc/4	2MHz
011b	Fhosc/8	1MHz
100b	Fhosc/16	0.5MHz
101b	Fhosc/32	0.25MHz
110b	Fhosc/64	0.125MHz
111b	Fhosc/128	0.0625MHz

07DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URCR2</b>	UR2CR7	UR2CR6	UR2CR5	UR2CR4	UR2CR3	UR2CR2	UR2CR1	UR2CR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

The UART2 baud rate clock source is Fhosc and divided by pre-scaler. The equation is as following.

$$\text{UART2 Baud Rate} = 1/2 * (\text{Fuart2} * 1/(256 - \text{URCR2})) \dots \text{bps}$$

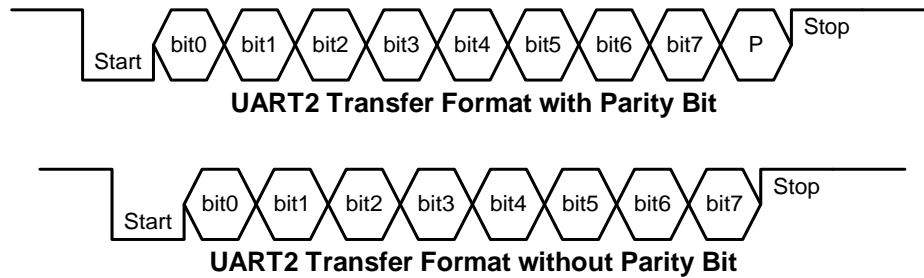
Fhosc = 8MHz

Baud Rate	UART2 Pre-scaler	URS[2:0]	URCR2 (Hex)	Accuracy (%)
1200	Fhosc/32	101b	98	0.16%
2400	Fhosc/32	101b	CC	0.16%
4800	Fhosc/32	101b	E6	0.16%
9600	Fhosc/32	101b	F3	0.16%
19200	Fhosc/32	101b	F9	0.16%
38400	Fhosc/1	000b	98	0.16%
51200	Fhosc/1	000b	B2	0.16%
57600	Fhosc/1	000b	BB	0.64%
102400	Fhosc/1	000b	DA	0.16%
115200	Fhosc/1	000b	DD	-0.79%

**Note: We strongly recommend not to set URCR2 = 0xFF, or UART2 operation would be error.**

## 10.4 UART2 TRANSFER FORMAT

The UART2 transfer format includes “Bus idle status”, “Start bit”, “8-bit Data”, “Parity bit” and “Stop bit” as following.



**Bus Idle Status:** The bus idle status is the bus non-operating status. The UART2 receiver bus idle status of MCU is floating status and tied high by the transmitter device terminal. The UART2 transmitter bus idle status of MCU is high status. The UART2 bus will be set when URX2EN and UTX2EN are enabled.

**Start Bit:** UART2 is an asynchronous type of communication and needs an attention bit to offer receiver the transfer starting. The start bit is a simple format which is high to low edge change and the duration is one bit period. The start bit is easily recognized by the receiver.

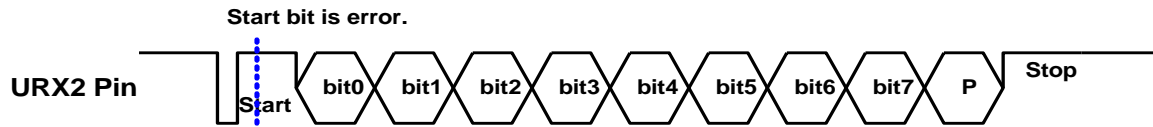
**8-bit Data:** The data format is 8-bit length, and LSB transfers first following start bit. The one bit data duration is the unit of UART2 baud rate controlled by register.

**Parity Bit:** The parity bit purpose is to detect data error condition. It is an extra bit following the data stream. The parity bit includes odd and even check methods controlled by URX2PS/UTX2PS bits. After receiving data and parity bit, the parity check executes automatically. The URX2PC bit indicates the parity check result. The parity bit function is controlled by URXP2EN/UTXP2EN bits. If the parity bit function is disabled, the UART2 transfer contents remove the parity bit and the stop bit follows the data stream directly.

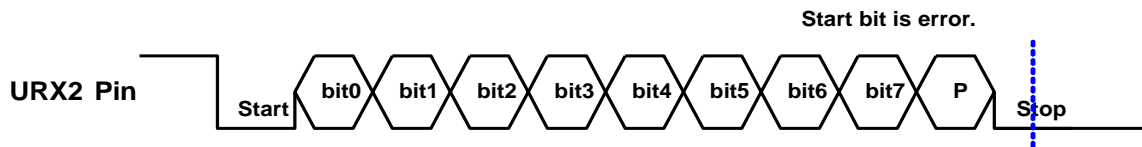
**Stop Bit:** The stop bit is like start bit using a simple format to indicate the end of UART2 transfer. The stop bit format is low to high edge change and the duration is one bit period.

## 10.5 ABNORMAL POCKET

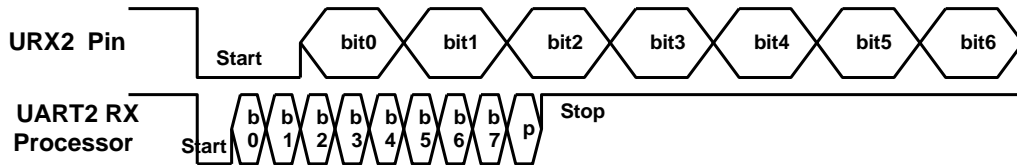
The abnormal pocket occurs in UART2 RX mode. Break pocket is one abnormal pocket of the UART2 architecture. The abnormal pocket includes Stream period error, start bit error, stop bit error...When UART2 receives abnormal pocket, the UFMER2 bit will be set "1", and UART2 issues URX2IRQ. The system finds the abnormal pocket through firmware. UART2 changes to initial status until detecting next start bit.



UART2 check the start bit is error and issue UFMER flag, but the UART2 still finishes receiving the pocket.



UART2 check the stop bit is error and issue U2FMER flag, but the UART2 still finishes receiving the pocket.



If the host's UART2 baud rate isn't match to receiver terminal, the received pocket is error. But it is not easy to differentiate the pocket is correct or not, because the received error pocket maybe match UART2 rule, but the data is error. Use checking UFMER2 bit and URX2PC bit status to decide the stream. If the two conditions seem like correct, but the pocket is abnormal, UART2 will accept the pocket as correct one.

## 10.6 UART2 RECEIVER CONTROL REGISTER

07CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URRX2</b>	URX2EN	URX2PEN	UTX2PS	URX2PC	URFMER2	UR2S2	UR2S1	UR2S0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

- Bit 7 **URX2EN**: UART2 RX control bit.  
0 = Disable UART2 RX. URX pin is GPIO mode or returns to GPIO status.  
1 = Enable UART2 RX. URX pin exchanges from GPIO mode to UART RX mode.
- Bit 6 **URX2PEN**: UART2 RX parity bit control bit.  
0 = Disable UART2 RX parity bit function. The data stream doesn't include parity bit.  
1 = Enable UART2 RX parity bit function. The data stream includes parity bit.
- Bit 5 **UTX2PS**: UART2 RX parity bit format control bit.  
0 = UART2 RX parity bit format is even parity.  
1 = UART2 RX parity bit format is odd parity.
- Bit 4 **URX2PC**: UART2 RX parity bit checking flag.  
0 = Parity bit is correct or no parity function.  
1 = Parity bit is error.
- Bit 3 **UFMER2**: UART2 RX stream frame error flag bit.  
0 = Collect UART frame.  
1 = UART2 frame is error including start/stop bit, stream length.
- Bit [2:0] **UR2S[2:0]**: UART2 per-scalar select bit.  
000 = Fhosc/1, 001 = Fhosc/2, 010 = Fhosc/4, 011 = Fhosc/8, 100 = Fhosc/16, 101 = Fhosc/32,  
110 = Fhosc/64, 111 = Fhosc/128.

## 10.7 UART2 TRANSMITTER CONTROL REGISTER

07BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URTX2</b>	UTX2EN	UTX2PEN	UTX2PS	-	URX2BZ	UTX2BZ	-	-
Read/Write	R/W	R/W	R/W	-	R	R	-	-
After reset	0	0	0	-	0	0	-	-

Bit 7 **UTX2EN**: UART2 TX control bit.

0 = Disable UART2 TX. UTX pin is GPIO mode or returns to GPIO status.

1 = Enable UART2 TX. UTX pin exchanges from GPIO mode to UART2 TX mode and idle high status.

Bit 6 **UTX2PEN**: UART2 TX parity bit control bit.

0 = Disable UART2 TX parity bit function. The data stream doesn't include parity bit.

1 = Enable UART2 TX parity bit function. The data stream includes parity bit.

Bit 5 **UTX2PS**: UART2 TX parity bit format control bit.

0 = UART2 TX parity bit format is even parity.

1 = UART2 TX parity bit format is odd parity.

Bit 3 **URX2BZ**: UART2 RX operating status flag.

0 = UART2 RX is idle or the end of processing.

1 = UART2 RX is busy and processing.

Bit 2 **UTX2BZ**: UART2 TX operating status flag.

0 = UART2 TX is idle or the end of processing.

1 = UART2 TX is busy and processing.

**Note:** *URX2BZ and UTX2BZ bits are UART2 operating indicators. After setting UART2 RX/TX operations, set (2\*Fcpu/Fuart2)\*NOP instruction is necessary, and then check UART2 status through URX2BZ and UTX2BZ bits.*

## 10.8 UART2 DATA BUFFER

07EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>UTX2D</b>	UTX2D7	UTX2D6	UTX2D5	UTX2D4	UTX2D3	UTX2D2	UTX2D1	UTX2D0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **UTX2D**: UART2 transmitted data buffer.

07FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>URX2D</b>	URX2D7	URX2D6	URX2D5	URX2D4	URX2D3	URX2D2	URX2D1	URX2D0
Read/Write	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **URX2D**: UART2 received data buffer.

## 10.9 UART2 OPERATION EXAMLPE

### UART2 TX Configuration:

```

; Select parity bit function.
      B0BCLR      FUTX2PEN      ; Disable UART2 TX parity bit function.
;or
      B0BSET      FUTX2PEN      ; Enable UART2 TX parity bit function.

; Select parity bit format.
      B0BCLR      FUTX2PS       ; UART2 TX parity bit format is even parity.
;or
      B0BSET      FUTX2PS       ; UART2 TX parity bit format is odd parity.

; Set UART2 baud rate.
      MOV         A, #value1     ; Set UART2 pre-scaler URS[2:0].
      B0MOV       URRX2, A
      MOV         A, #value2     ; Set UART2 baud rate 8-bit buffer.
      B0MOV       URCR2, A

; Enable UART2 TX pin.
      B0BSET      FUTX2EN       ; Enable UART2 TX function and UART2 TX pin.

; Enable UART2 TX interrupt function.
      B0BCLR      FUTX2IRQ       ; Clear UART2 TX interrupt flag.
      B0BSET      FUTX2IEN       ; Enable UART2 TX interrupt function.

; Load TX data buffer and execute TX transmitter.
      MOV         A, #value3     ; Load 8-bit data to UTX2D data buffer.
      B0MOV       UTX2D, A
                                   ;After loading UTX2D, UART2 TX starts to transmit.
      NOP                                   ; One instruction delay for UTX2BZ flag.

; Check TX operation.
      B0BTS0      FUTX2BZ       ; Check UTXBZ bit.
      JMP         CHKTX          ; UTXBZ=1, TX is operating.
      JMP         ENDTX          ; UTXBZ=0, the end of TX.

```

\* **Note:** UART2 TX operation is started through loading UTX2D data buffer.



# 11 MAIN SERIAL PORT(MSP)

## 11.1 OVERVIEW

The MSP (Main Serial Port) is a serial communication interface for data exchanging from one MCU to one MCU or other hardware peripherals. These peripheral devices may be serial EEPROM, A/D converters, Display device, etc. The MSP module can operate in one of two modes:

- ☞ **Master Tx,Rx Mode**
- ☞ **Slave Tx,Rx mode (with general address call) for multiplex slave in single master situation.**

The MSP features include the following:

- ☞ **2-wire synchronous data transfer/receiver.**
- ☞ **Master (SCL is clock output) or Slave (SCL is clock input) operation.**
- ☞ **SCL, SDA are programmable open-drain output pin for multiplex slave devices application.**
- ☞ **Support 400K clock rate @ Fcpu=4MIPs.**
- ☞ **End-of-Transfer/Receiver interrupt.**

## 11.2 MSP STATUS REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MSPSTAT</b>	-	CKE	D_A	P	S	RED_WRT	-	BF
Read/Write	-	R/W	R	R	R	R	-	R
After reset	-	0	0	0	0	0	-	0

- Bit 6     **CKE:** Slave Clock Edge Control bit  
 In Slave Mode: Receive Address or Data byte  
 0= Latch Data on SCL Rising Edge. **(Default)**  
 1= Latch Data on SCL Falling Edge.

**Note:**

**In Slave Transmit mode, Address Received depended on CKE setting. Data Transfer on SCL     Falling Edge.**

**In Slave Receiver mode, Address and Data Received depended on CKE setting.**

- Bit 5     **D\_A:** Data/Address\_ bit  
 0= Indicates the last byte received or transmitted was address.  
 1= Indicates the last byte received or transmitted was data.

- Bit 4     **P:** Stop bit  
 0 = Stop bit was not detected.  
 1 = Indicates that a stop bit has been detected last.

**Note: It will be cleared when Start bit was detected.**

- Bit 3     **S:** Start bit.  
 0 = Start bit was not detected.  
 1 = Indicates that a start bit has been detected last

**Note: It will be cleared when STOP bit was detected.**

- Bit 2 **RED\_WRT**: Read/Write bit information.  
 This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next start bit, stop bit, or not ACK bit.  
In slave mode:  
 0 = Write.  
 1 = Read.  
In master mode:  
 0 = Transmit is not in progress.  
 1 = Transmit is in progress.  
 Or this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSP is in IDLE mode.
- Bit 0 **BF**: Buffer Full Status bit  
Receive  
 1 = Receive complete, MSPBUF is full.  
 0 = Receive not complete, MSPBUF is empty.  
Transmit  
 1 = Data Transmit in progress (does not include the ACK and stop bits), MSPBUF is full.  
 0 = Data Transmit complete (does not include the ACK and stop bits), MSPBUF is empty.

## 11.3 MSP MODE REGISTER 1

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MSPM1</b>	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	-	MSPC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
After reset	0	0	0	0	0	0	-	0

- Bit 7 **WCOL**: Write Collision Detect bit  
Master Mode:  
 0 = No collision  
 1 = A write to the MSPBUF register was attempted while the MSP conditions were not valid for a transmission to be started  
Slave Mode:  
 0 = No collision  
 1 = The MSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
- Bit 6 **MSPOV**: Receive Overflow Indicator bit  
 0 = No overflow.  
 1 = A byte is received while the MSPBUF register is still holding the previous byte. MSPOV is a “don’t care” in transmit mode. MSPOV must be cleared in software in either mode. (must be cleared in software)
- Bit 5 **MSPENB**: MSP Communication Enable.  
 0 = Disables serial port and configures these pins as I/O port pins  
 1 = Enables serial port and configures SCL, SDA as the source of the serial port pins

**Note:** MSP status register will be clear after MSP Disable. So, user should setting MSP register again before MSP Enable.

Ex: **B0BCLR FMSPENB**  
**CALL MSP\_init\_setting**  
**B0BSET FMSPENB**

- Bit 4 **CKP**: SCL Clock Priority Control bit  
 In MSP Slave mode  
 0 = Hold SCL keeping Low. (Ensure data setup time and Slave device ready.)  
 1 = Release SCL Clock  
 (Slave Transistor mode CKP function always enables, Slave Receiver CPK function control by SLRXCKP)  
 In MSP Master mode Unused.

Bit 3 **SLRXCKP**: Slave Receiver mode SCL Clock Priority Control bit  
 In MSP Slave Receiver mode.  
 0 = Disable CKP function.  
 1 = Enable CKP function.  
 In MSP Slave and Slave Transistor mode Unused.

Bit 2 **MSPWK**: MSP Wake-up indication bit  
 0 = MCU NOT wake-up by MSP.  
 1 = MCU wake-up by MSP

\* **Note: Clear MSPWK before entering Power down mode for indication the wake-up source from MSP or not**

Bit 0 **MSPC**: MSP mode Control register  
 0 = MSP operated on Slave mode, 7-bit address  
 1 = MSP operated on Master mode.

## 11.4 MSP MODE REGISTER 2

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MSPM2</b>	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit 7 **GCEN**: General Call Enable bit (In Slave mode only)  
 0 = General call address disabled  
 1 = Enable interrupt when a general call address (0000h) is received.

Bit 6 **ACKSTAT**: Acknowledge Status bit (In master mode only)  
In master transmit mode:  
 0 = Acknowledge was received from slave  
 1 = Acknowledge was not received from slave

Bit 5 **ACKDT**: Acknowledge Data bit. (In master mode only)  
In master receive mode:  
 Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.  
 0 = Acknowledge  
 1 = Not Acknowledge

bit 4 **ACKEN**: Acknowledge Sequence Enable bit (In MSP master mode only)  
 In master receive mode:  
 0 = Acknowledge sequence idle  
 1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit AKDT data bit. Automatically cleared by hardware.

bit 3 **RCEN**: Receive Enable bit (In master mode only)  
 0 = Receive idle  
 1 = Enables Receive mode for MSP

bit 2 **PEN**: Stop Condition Enable bit (In master mode only)  
 0 = Stop condition idle  
 1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.

bit 1 **RSEN**: Repeated Start Condition Enabled bit (In master mode only)  
 0 = Repeated Start condition idle.  
 1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.

bit 0 **SEN**: Start Condition Enabled bit (In master mode only)  
 0 = Start condition idle  
 1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.

## 11.5 MSP MSPBUF REGISTER

**MSPBUF initial value = 0000 0000**

0D3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MSPBUF</b>	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

## 11.6 MSP MSPADR REGISTER

**MSPADR initial value = 0000 0000**

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>MSPADR</b>	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:1] 7-bit Address.

Bit 0 Tx/Rx mode control bit.  
0=Tx mode.  
1=Rx mode.

## 11.7 SLAVE MODE OPERATION

When an address is matched or data transfer after and address match is received, the hardware automatically will generate the acknowledge (ACK\_) signal, and load MSPBUF (MSP buffer register) with the received data from MSPSR.

There are some conditions that will cause MSP function will not reply ACK\_ signal:

- 1.Data Buffer already full: BF=1 (MSPSTAT bit 0), when another transfer was received.
- 2.Data Overflow: MSPOV=1 (MSPM1 bit 6), when another transfer was received

When BF=1, means MSPBUF data is still not read by MCU, so MSPSR will not load data into MSPBUF, but MSPIRQ and MSPOV bit will still set to 1. BF bit will be clear automatically when reading MSPBUF register. MSPOV bit must be clear through by Software.

### 10.7.1 Addressing

When MSP Slave function has been enabled, it will wait a START signal occur. Following the START signal, 8-bit address will shift into the MSPSR register. The data of MSPSR[7:1] is compare with MSPADDR register on the falling edge of eight SCL pulse, If the address are the same, the BF and MSPOV bit are both clear, the following event occur:

1. MSPSR register is loaded into MSPBUF on the falling edge of eight SCL pulse.
2. Buffer full bit (BF) is set to 1, on the falling edge of eight SCL pulse.
3. An ACK\_ signal is generated.
4. MSP interrupt request MSPIRQ is set on the falling edge of ninth SCL pulse.

Status when Data is Received		MSPSR → MSPBUF	Reply an ACK_ signal	Set MSPIRQ
BF	MSPOV			
0	0	Yes	Yes	Yes
*0	*1	Yes	No	Yes
1	0	No	No	Yes
1	1	No	No	Yes

**Data Received Action Table**

**Note:** BF=0, MSPOV=1 shows the software is not set properly to clear Overflow register.

## 10.7.2 Slave Receiving

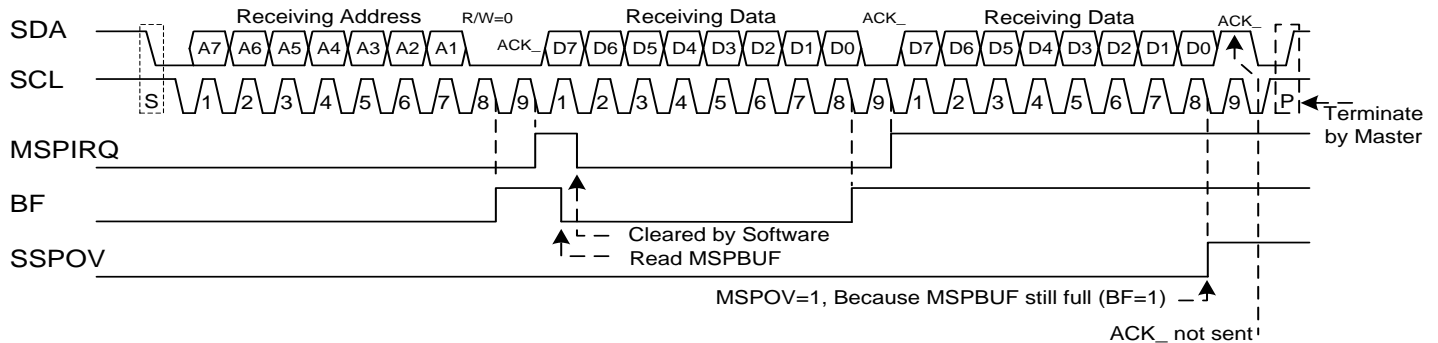
When the R/W bit of address byte =0 and address is matched, the RED\_WRT bit of MSPSTAT is cleared. The address will be load into MSPBUF. After reply an ACK\_ signal, MSP will receive data every 8 clock. The CKP function enable or disable (Default) is controlled by SLRXCKP bit and data latch edge -Rising edge (Default) or Falling edge is controlled by CPE bit.

When overflow occur, no acknowledge signal replied which either BF=1 or MSPOV=1.

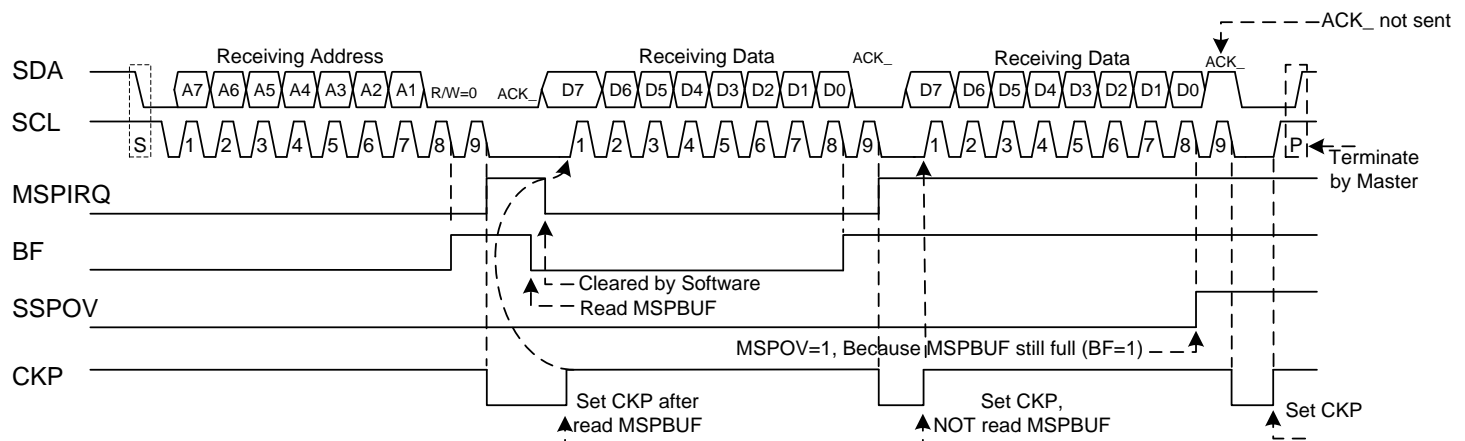
MSP interrupt is generated in every data transfer. The MSPIRQ bit must be clear by software.

Following is the Slave Receiving Diagram

SLRXCKP=0



SLRXCKP=1

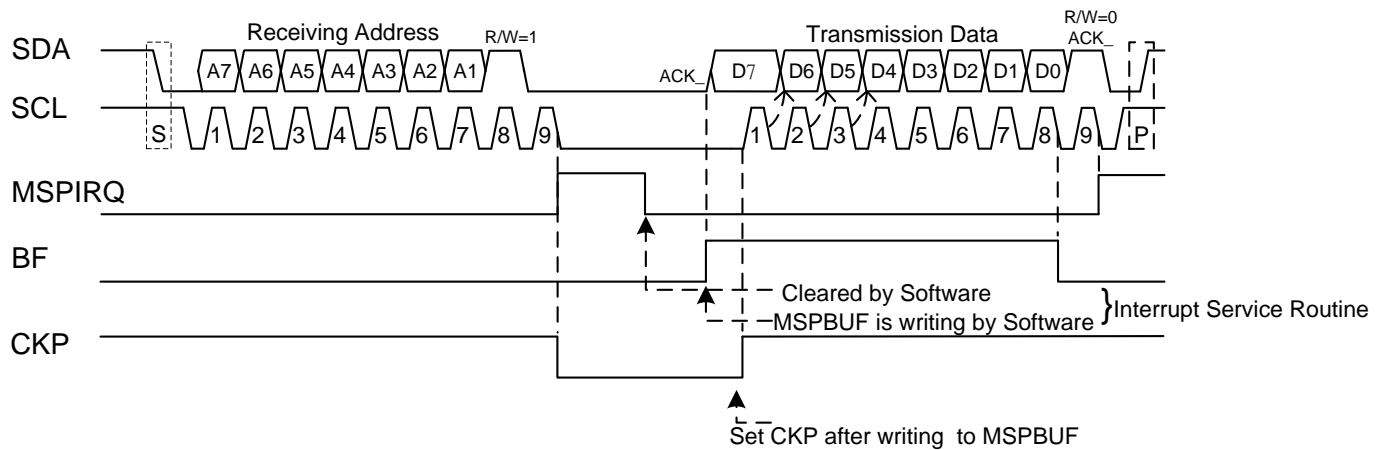


## 10.7.3 Slave Transmission

After address match, the following R/W bit is set, MSPSTAT bit 2 RED\_WRT will be set. The received address will be load to MSPBUF and ACK\_ will be sent at ninth clock then SCL will be hold low. Transmission data will be load into MSPBUF which also load to MSPSR register. The Master should monitor SCL pin signal. The slave device may hold on the master by keep CKP low. When set. After load MSPBUF, set CKP bit, MSPBUF data will shift out on the falling edge on SCL signal. This will ensure the SDA signal is valid on the SCL high duty.

An MSP interrupt is generated on every byte transmission. The MSPIRQ will be set on the ninth clock of SCL. Clear MSPIRQ by software. MSPSTAT register can monitor the status of data transmission.

In Slave transmission mode, an ACK\_ signal from master-receiver is latched on rising edge of ninth clock of SCL. If ACK\_ = high, transmission is complete. Slave device will reset logic and waiting another START signal. If ACK\_ = low, slave must load MSPBUF which also MSPSR, and set CKP=1 to start data transmission again.



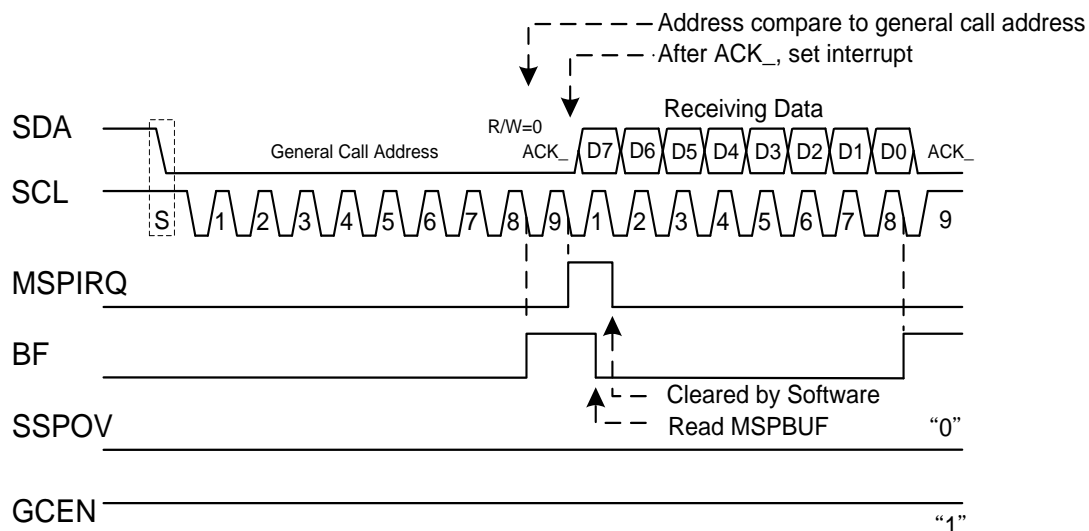
**MSP Slave Transmission Timing Diagram**

### 10.7.4 General Call Address

In MSP bus, the first 7-bit is the Slave address. Only the address match MSPADR the Slave will response an ACK\_. The exception is the general call address which can address all slave devices. When this address occur, all devices should response an acknowledge.

The general call address is a special address which is reserved as all "0" of 7-bit address. The general call address function is control by GCEN bit. Set this bit will enable general call address and clear it will disable. When GECN=1, following a START signal, 8-bit address will shift into MSPSR and the address is compared with MSPADR and also the general call address which fixed by hardware.

If the general call address matches, the MSPSR data is transferred into MSPBUF, the BF flag bit is set, and in the falling edge of the ninth clock (ACK\_) MSPIRQ flag set for interrupt request. In the interrupt service routine, reading MSPBUF can check if the address is the general call address or device specific.

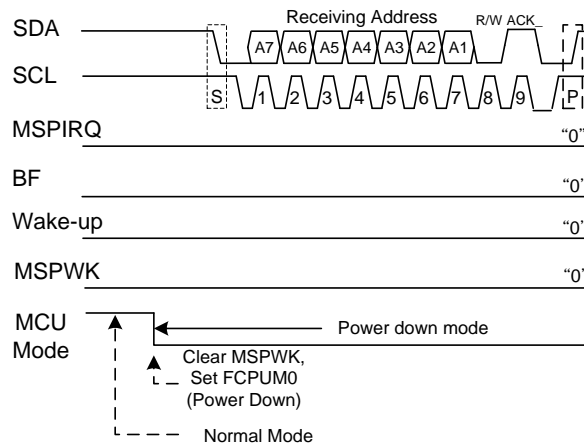


**General Call Address Timing Diagram**

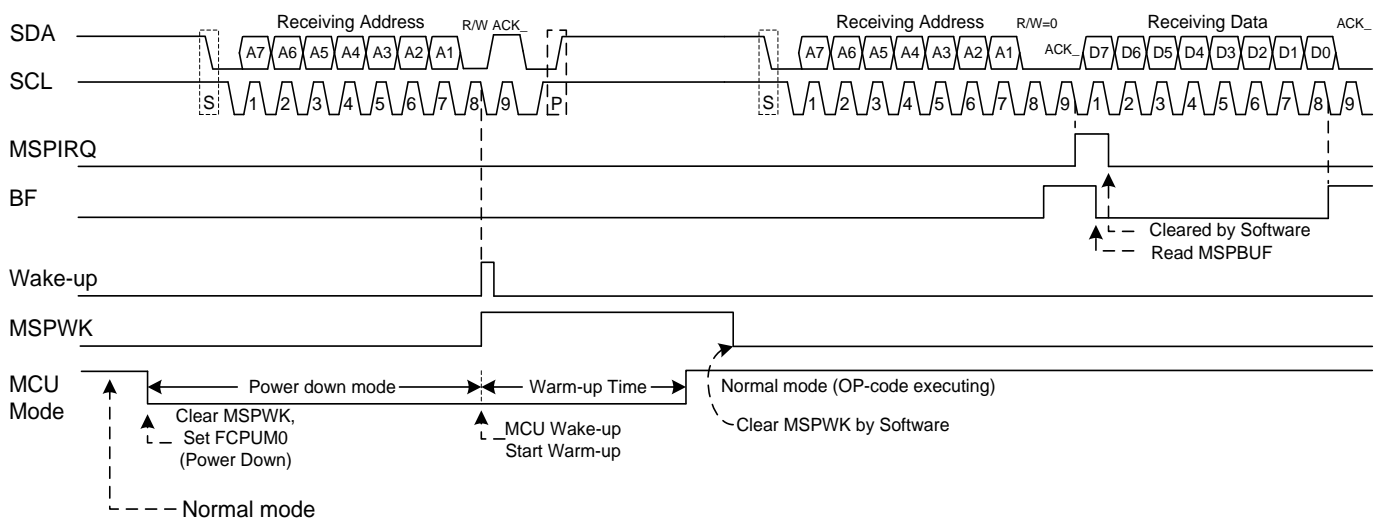
## 10.7.5 Slave Wake up

When MCU enter Power down mode, if MSPENB bit is still set, MCU can wake-up by matched device address. The address of MSP bus following START bit, 8-byte address will shift into MSPSR, if address matched, an NOT Acknowledge will response on the ninth clock of SCL and MCU will be wake-up, MSPWKset and start wake-up procedure but MSPIRQ will not set and MSPSR data will not load to MSPBUF. After MCU finish wake-up procedure, MSP will be in idle status and waiting master's START signal. Control register BF, MSPIRQ, MSPOV and MSPBUF will be the same status/data before power down.

If address not matches, a NOT acknowledge is still sent on the ninth clock of SCL, but MCU will be NOT wake-up and still keep in power down mode.



**MSP Wake-up Timing Diagram: Address NOT Matched**



**MSP Wake-up Timing Diagram: Address Matched**

**Note:**

1. **MSP function only can work on Normal mode, when wake-up from power down mode, MCU must operate in Normal mode before Master sent START signal.**
2. **In MSP wake-up, if the address not matches, MCU will keep in power down mode.**
3. **Clear MSPWK before enter power down mode by Software for wake-up indication.**

## 10.8 MASTER MODE

Master mode of MSP operation from a START signal and end by STOP signal. The START (S) and STOP (P) bit are clear when reset or MSP function disabled. In Master mode the SCL and SDA line are controlled by MSP hardware. Following events will set MSP interrupt request (MSPIRQ), if MSPIEN set, interrupt occurs.

- ☞ START condition
- ☞ STOP condition
- ☞ Data byte transmitted or received
- ☞ Acknowledge Transmit.
- ☞ Repeat START.

### 10.8.1 Mater Mode Support

Master mode enable when MSPC and MSPENB bit set. Once the Master mode enabled, the user had following six options.

- Send a START signal on SCL and SDA line.
- Send a Repeat START signal on SCL and SDA line.
- Write to MSPBUF register for Data or Address byte transmission
- Send a STOP signal on SCL and SDA line.
- Configuration MSP port for receive data
- Send an Acknowledge at the end of a received byte of data.

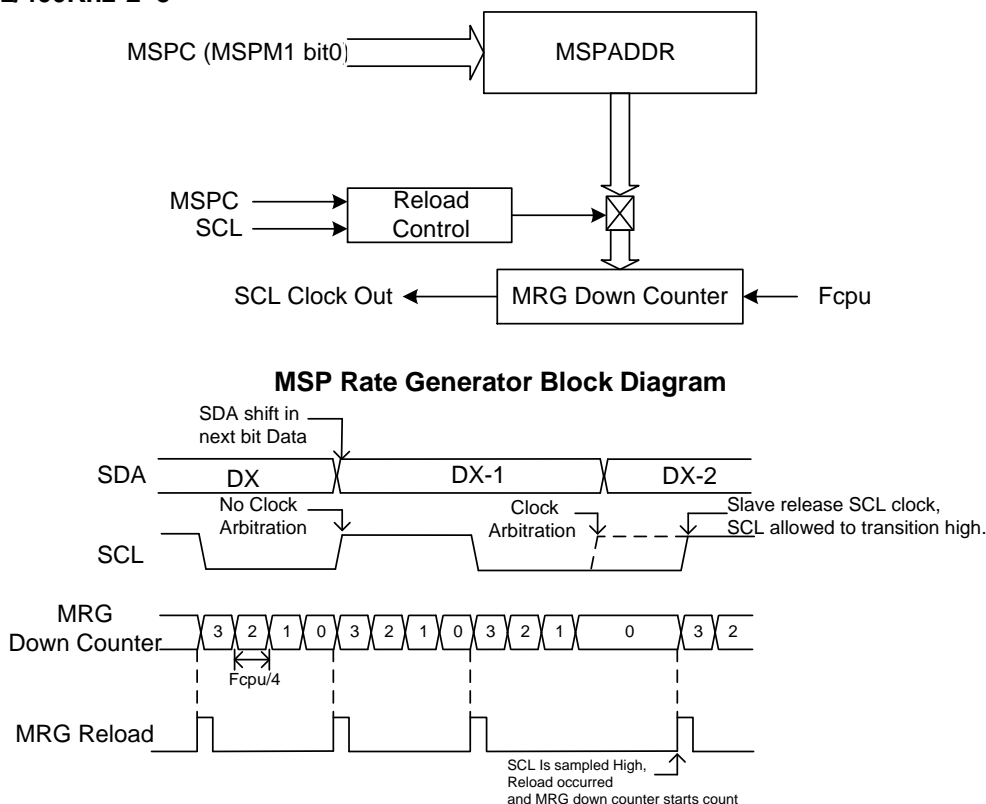
### 10.8.2 MSP Rate Generator

In MSP Mode, the MSP rate generator's reload value is located in the lower 7 bit of MSPADDR register. When MRG is loaded with the register, the MRG count down to 0 and stop until another reload has taken place. In MSP mater mode MRG reload from MSPADDR automatically. If Clock Arbitration occur for instance (SCL pin keep low by Slave device), the MRG will reload when SCL pin is detected High.

SCL clock rate =  $F_{cpu}/(MSPADDR)*2$

For example, if we want to set 400Khz in 4Mhz Fcpu, the MSPADDR have to set 0x05h.

$MSPADDR = 4Mhz/400Khz*2 = 5$





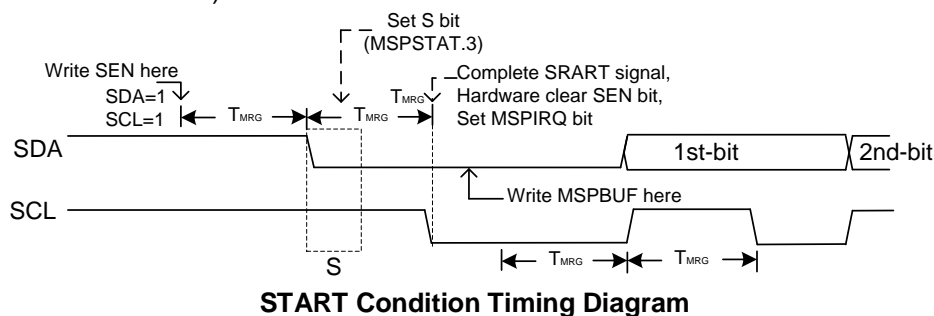
## MRG Timing Diagram with and without Clock Arbitration (MSPADRR=0x03)

## 10.8.3 MSP Master START Condition

To generate a START signal, user sets SEN bit (MSPM2.0). When SDA and SCL pin are both sampled High, MSP rate generator reload MSPADRR[6:0], and starts down counter. When SDA and SCL are both sampled high and MRG overflow, SDA pin is drive low. When SCL sampled high, and SDA transmitted from High to Low is the START signal and will set S bit (MSPSTAT.3). MRG reload again and start down counter. SEN bit will be clear automatically when MRG overflow, the MRG is suspend leaving SDA line held low, and START condition is complete.

## 10.8.3.1 WCOL Status Flag

If user write to MSPBUF when START condition processing, then WCOL is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



START Condition Timing Diagram

## 10.8.4 MSP Master mode Repeat START Condition

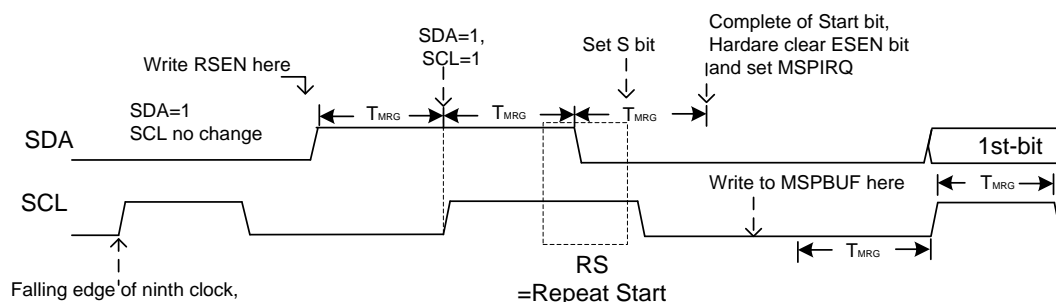
When MSP logic module is idle and RSEN set to 1, Repeat Start progress occurs. RSEN set and SCL pin is sampled low, MSPADRR[6:0] data reload to MSP rate generator and start down counter. The SDA pin is release to high in one MSP rate generate counter ( $T_{MRG}$ ). When the MRG is overflow, if SDA is sampled high. SCL will be brought high. When SCL is sampled high, MSPADRR reload to MRG and start down counter. SDA and SCL must keep high in one  $T_{MRG}$  period. In the next  $T_{MRG}$  period, SDA will be brought low when SCL is sampled high, then RSEN will clear automatically by hardware and MRG will not reload, leaving SDA pin held low. Once detect SDA and SCL occur START condition, the S bit will be set (MSPSTAT.3). MSPIRQ will not set until MRG overflow.

**Note:**

1. While any other event is progress, Set RSEN will take no effect.
2. A bus collision during the Repeat Start condition occurs: SDA is sampled low when SCL goes from low to high.

## 10.8.4.1 WCOL Status Flag

If user write to MSPBUF when Repeat START condition processing, then WCOL is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



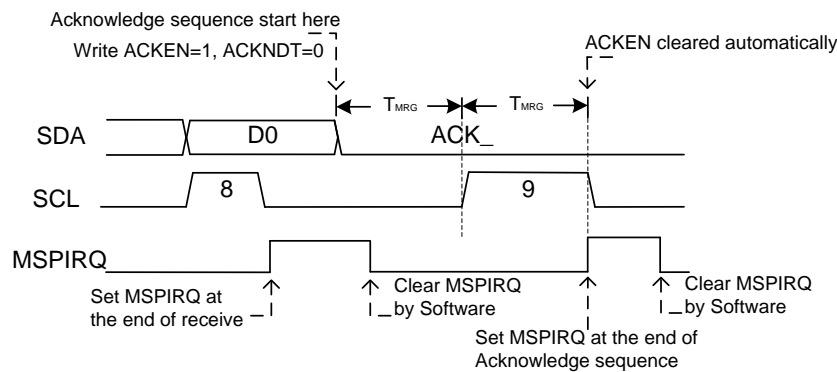
Repeat Start Condition Timing Diagram

### 10.8.5 Acknowledge Sequence Timing

An acknowledge sequence is enabled when set ACKEN (MSPM2.4). SCL is pulled low when set ACKEN and the content of the acknowledge data bit is present on SDA pin. If user wished to reply a acknowledge, ACKDT bit should be cleared. If not, set ACKDT bit before starting a acknowledge sequence. SCL pin will be release (brought high) when MSP rate generator overflow. MSP rate generator start a  $T_{MRG}$  period down counter, when SCL is sampled high. After this period, SCL is pulled low, and ACKEN bit is clear automatically by hardware. When next MRG overflow again, MSP goes into idle mode.

#### 10.8.5.1 WCOL Status Flag

If user write to MSPBUF when Acknowledge sequence processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



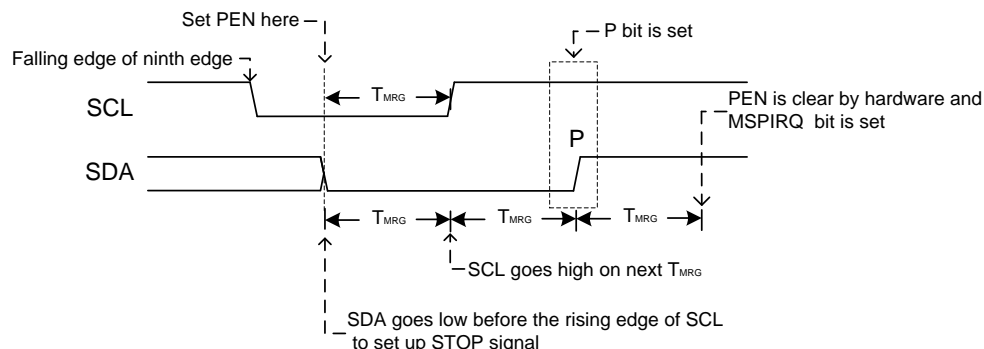
**Acknowledge Sequence Timing Diagram**

### 10.8.6 STOP Condition Timing

At the end of received/transmitted, a STOP signal present on SDA pin by setting the STOP bit register, PEN (MSPM2.1). At the end of receive/transmit, SCL goes low on the falling edge of ninth clock. Master will set SDA go low, when set PEN bit. When SDA is sampled low, MSP rate generator is reloaded and start count down to 0. When MRG overflow, SCL pin is pull high. After one  $T_{MRG}$  period, SDA goes High. When SDA is sampled high while SCL is high, bit P is set. PEN bit is clear after next one  $T_{MRG}$  period, and MSPIRQ is set.

#### 10.8.6.1 WCOL Status Flag

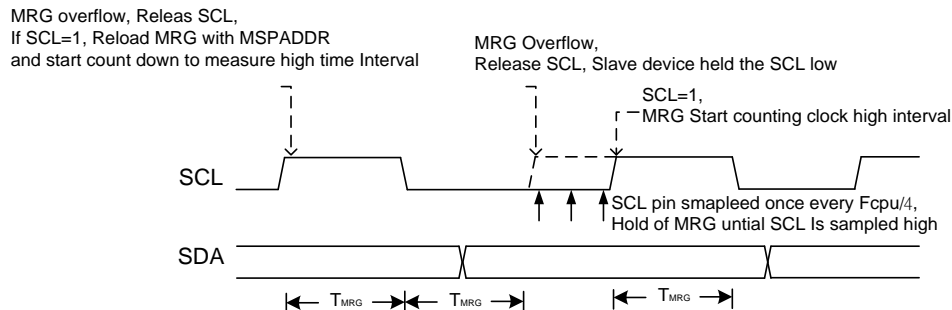
If user write to MSPBUF when a STOP condition is processing, then WCOL bit is set and the content of MSPBUF data is un-changed. (the writer doesn't occur)



**STOP condition sequence Timing Diagram**

### 10.8.7 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeat START, STOP condition that SCL pin allowed to float high. When SCL pin is allowed float high, the master rate generator (MRG) suspended from counting until the SCL pin is actually sampled high. When SCL is sampled high, the MRG is reloaded with the content of MSPADDR[6:0], and start down counter. This ensure that SCL high time will always be at least one MRG overflow time in the event that the clock is held low by an external device.



### Clock Arbitration sequence Timing Diagram

### 10.8.8 Master Mode Transmission

Transmission a data byte, 7-bit address or the eight bit data is accomplished by simply write to MSPBUF register. This operation will set the Buffer Full flag BF and allow MSP rate generator start counting. After write to MSPBUF, each bit of address will be shifted out on the falling edge of SCL until 7-bit address and R/W\_ bit are complete. On the falling edge of eighth clock, the master will pull low SDA for slave device respond with an acknowledge. On the ninth clock falling edge, SDA is sampled to indicate the address already accept by slave device. The status of the ACK bit is load into ACKSTAT status bit. Then MSPIRQ bit is set, the BF bit is clear and the MRG is hold off until another write to the MSPBUF occurs, holding SCL low and allow SDA floating.

### 10.8.8.1 BF Status Flag

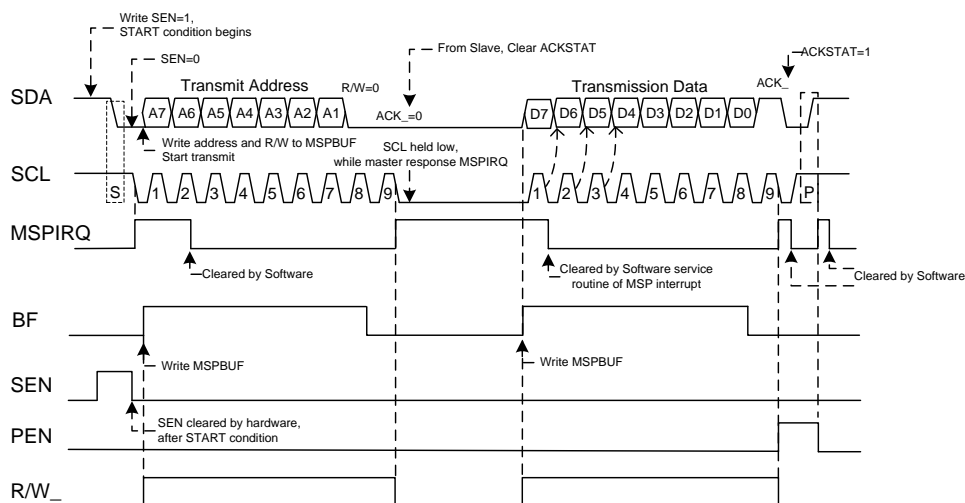
In transmission mode, the BF bit is set when user writes to MSPBUF and is cleared automatically when all 8 bit data are shift out.

### 10.8.8.2 WCOL Flag

If user write to MSPBUF during Transmission sequence in progress, the WCOL bit is set and the content of MSPBUF data will unchanged.

### 10.8.8.3 ACKSTAT Status Flag

In transmission mode, the ACKSTAT bit is cleared when the slave has sent an acknowledge (ACK\_=0), and is set when slave does not acknowledge (ACK\_=1). A slave send an acknowledge when it has recognized its address (including general call), or when the slave has properly received the data.



### MSP Master Transmission Mode Timing Diagram

### 10.8.9 Master Mode Receiving

Master receiving mode is enable by set RCEN bit.

The MRG start counting and when SCL change state from low to high, the data is shifted into MSPSR. After the falling edge of eighth clock, the receive enable bit (RCEN) is clear automatically, the contents of MSP are load into MSPBUF, the BF flag is set, the MSPIRQ flag is set and MRG counter is suspended fro, counting, holding SCL low. The MSP is now in IDLE mode and awaiting the next operation command. When the MSPBUF data is read by Software, the BF flag is cleat automatically. By setting ACKEN bit, user can send an acknowledge bit at the end of receiving.

#### 10.8.9.1 BF Status Flag

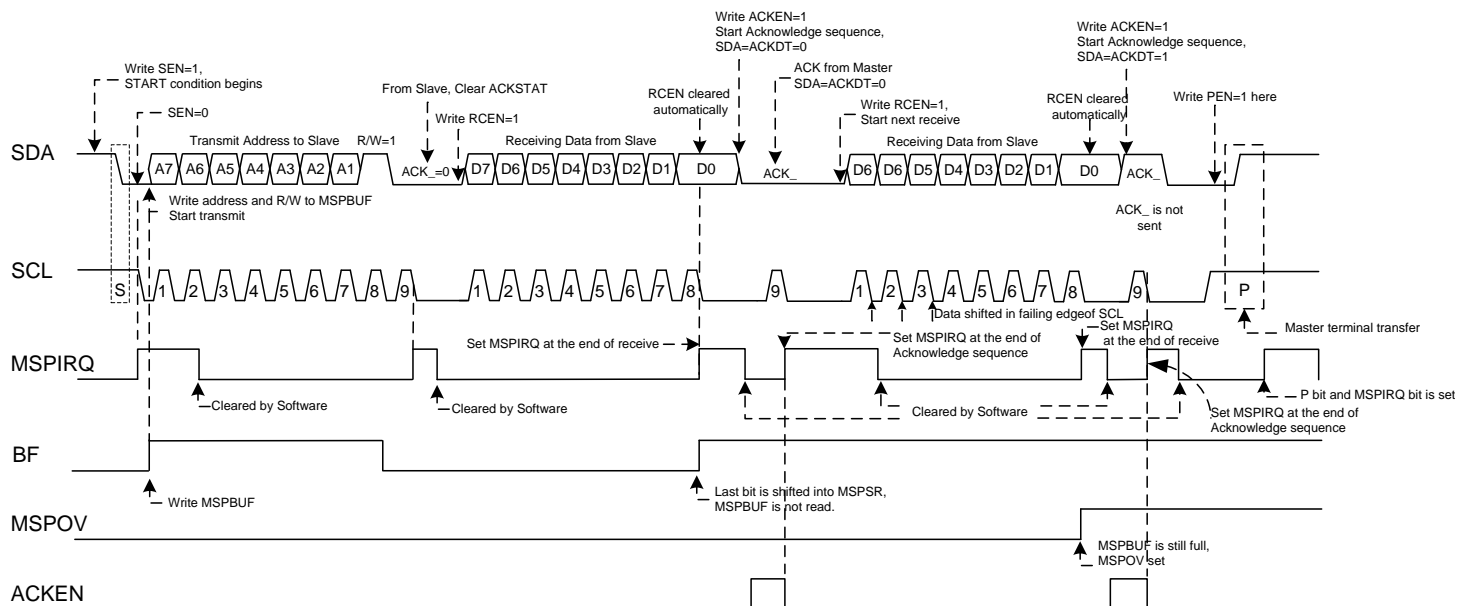
In Reception mode, the BF bit is set when an address or data byte is loaded into MSPBUF from MSPSR. It is cleared automatically when MSPBUF is read.

#### 10.8.9.2 MSPOV Flag

In receive operation, the MSPOV bit is set when another 8-bit are received into MSPSR, and the BF bit is already set from previous reception

#### 10.8.9.3 WCOL Flag

If user write to MSPBUF when a receive is already progress, the WCOL bit is set and the content of MSPBUF data will unchanged.



MSP Master Receiving Mode Timing Diagram

# 12 LCD DRIVER

## 12.1 OVERVIEW

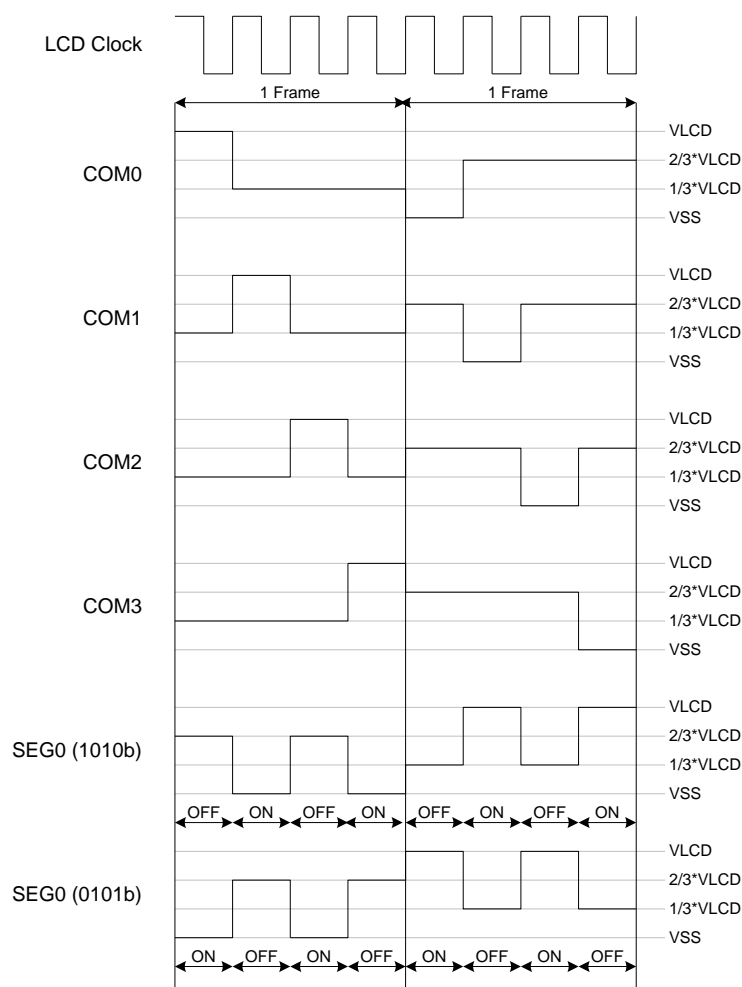
LCD driver is C type structures with 4 x 32 or 6 x 30. The LCD scan timing is 1/4 or 1/6 duty with 1/3 bias structure and C-type mode to yield 128 dots or 180 dots. C type is using internal charge pump to adjust LCD power and bias voltage.

## 12.2 LCD TIMING

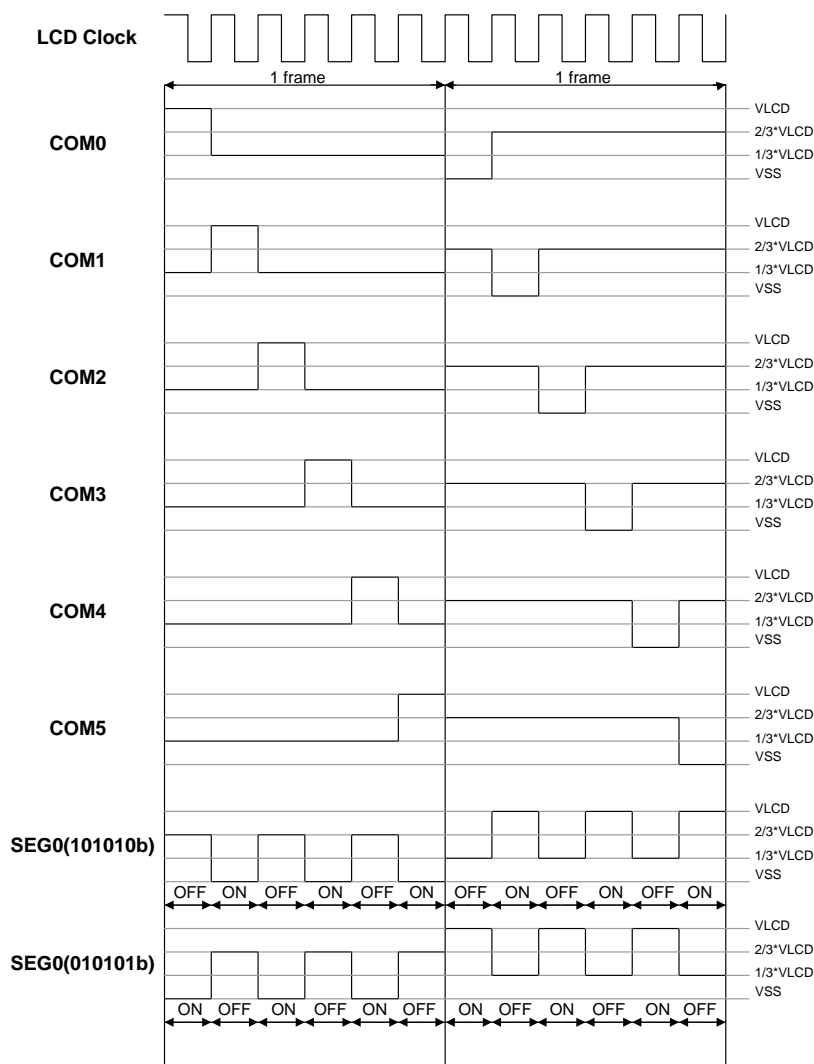
Register		LCD Clock Source	LCD Clock (Hz)	Frame Rate (Hz)	Type	Pump source
LCDRATE	LCDMODE					
0	0	Fosc	$F_{osc}/128=256$	64	4-COM (1/4 duty)	Embedded OSC
0	1	Fosc		43	6-COM (1/6 duty)	Embedded OSC
1	0	Fosc	$F_{osc}/64=512$	128	4-COM (1/4 duty)	Embedded OSC
1	1	Fosc		85	6-COM (1/6 duty)	Embedded OSC

Note:  $F_{osc}$  = ILRC or 32768Hz x'tal

### LCD Drive Waveform, 1/4 duty, 1/3 bias



### LCD Drive Waveform, 1/6 duty, 1/3 bias



LCD Drive Waveform, 1/6 duty, 1/3 bias

## 12.3 LCD RAM LOCATION

COM0~COM3 vs. SEG0~SEG31 LCD RAM Location

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
	COM0	COM1	COM2	COM3	-	-	-	-
SEG 0	00H.0	00H.1	00H.2	00H.3	N/A	N/A	-	-
SEG 1	01H.0	01H.1	01H.2	01H.3	N/A	N/A	-	-
SEG 2	02H.0	02H.1	02H.2	02H.3	N/A	N/A	-	-
SEG 3	03H.0	03H.1	03H.2	03H.3	N/A	N/A	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 31	1FH.0	1FH.1	1FH.2	1FH.3	N/A	N/A	-	-

COM0~COM5 vs. SEG2~SEG31 LCD RAM Location

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
	COM0	COM1	COM2	COM3	COM4	COM5	-	-
SEG 0 (COM4)	N/A	N/A	N/A	N/A	N/A	N/A	-	-
SEG 1 (COM5)	N/A	N/A	N/A	N/A	N/A	N/A	-	-
SEG 2	02H.0	02H.1	02H.2	02H.3	02H.4	02H.5	-	-
SEG 3	03H.0	03H.1	03H.2	03H.3	03H.4	03H.5	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 29	1DH.0	1D H.1	1D H.2	1D H.3	1D H.4	1D H.5	-	-
SEG 30	1EH.0	1E H.1	1E H.2	1E H.3	1E H.4	1E H.5	-	-
SEG 31	1FH.0	1F H.1	1F H.2	1F H.3	1F H.4	1F H.5	-	-

## 12.4 LCDM1 REGISTER

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	LCDPENB	P3SEG	LCDBNK	-	LCDMODE	LCDENB	LCDRATE	-
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	-
After Reset	0	0	0	-	0	0	1	-

- Bit1**     **LCDRATE:** LCD clock rate control  
**0** = LCD clock rate is 128Hz.  
**1** = LCD clock rate is 64Hz. (Recommend setting)
- Bit2**     **LCDENB:** LCD driver enable control bit.  
**0** = Disable  
**1** = Enable.
- Bit3**     **LCDMODE:** 4-COM or 6-COM control bit.  
**0** = 4-COM  
**1** = 6-COM (S0~S1 as COM4~COM5)
- Bit 5**     **LCDBNK:** LCD blank control bit.  
**0** = Normal display.  
**1** = All of the LCD dots off.
- Bit 6**     **P3SEG:** SEG26~31 and P30~P35 Selection bit.  
**0** = P30~P35 as IO Mode Enable, VLCD1 connect to VDD.  
**1** = P30~P35 as LCD Mode Enable, VLCD1 connect to VLCD.
- Bit7**     **LCDPENB :** C-Type LCD Pump Enable bit.  
**0** = Pump disable.  
**1** = Pump Enable.



## 12.5 LCDM2 REGISTER

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>LCDM2</b>	VAR1	VAR0	BGM	DUTY1	DUTY0	VCP2	VCP 1	VCP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	1	0	0	0	1	1

**Bit[2:0] VCP [2:0]:** LCD Charge pump output voltage

Register	Output Voltage	
VCP [2:0]	VLCD	Pump Mode
000	2.7V	C-Type, 4 or 6 COM
001	2.8V	
010	2.9V	
011	3.0V	
100	3.1V	
101	3.2V	
110	3.3V	
111	3.4V	

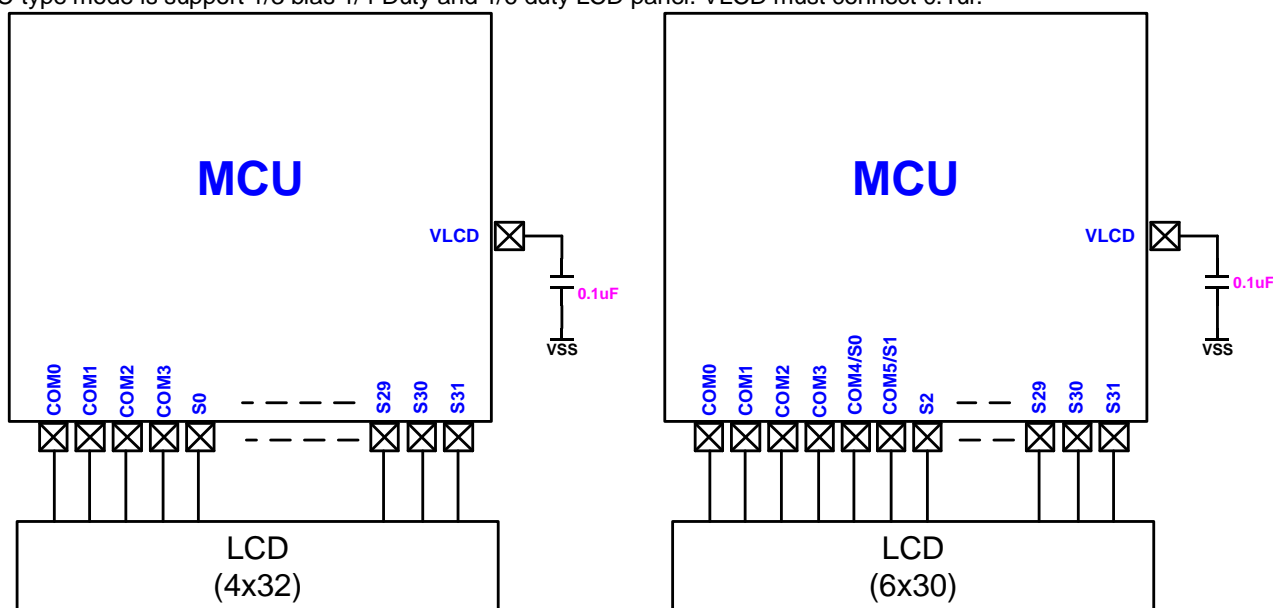
**Bit[4:3] Duty [1:0]:** LCD charge pump clock duty cycle selection.  
Note: Recommend user set duty register: Duty [1:0]: 11

**Bit5 BGM:** Low power mode control bit.  
0 = Enable LCD Low power mode.  
1 = Disable LCD low power mode.

**Bit[7:6] VAR [1:0]:** LCD power saving mode register setting  
Note: Recommend user set VAR register: VAR [1:0]: 00

## 12.6 C-TYPE LCD MODE

LCD C-type mode is support 1/3 bias 1/4 Duty and 1/6 duty LCD panel. VLCD must connect 0.1uF.



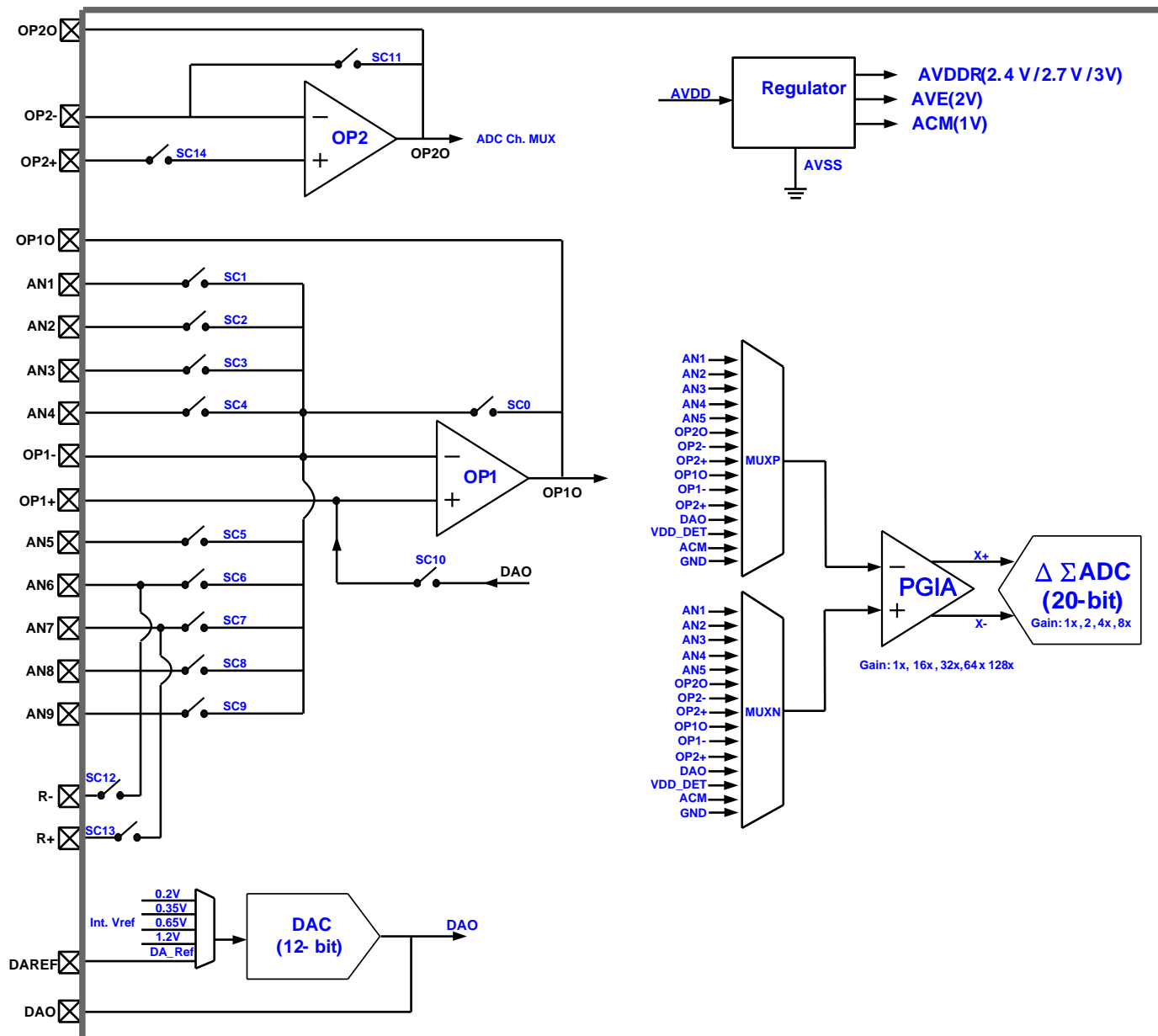
# 13 ADC, REGULATOR, DAC and OPA

## 13.1 OVERVIEW

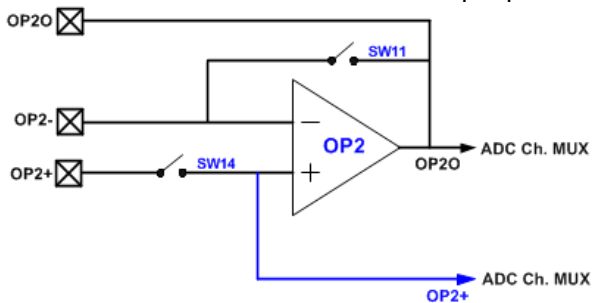
The SN8F29E49 has a built-in Voltage Regulator to support a stable voltage 2.4V from pin AVDDR and 2.0V from pin AVE+ with maximum 10mA current driving capacity. The AVDDR provides stable voltage for internal circuits (PGIA, ADC, DAC) and external sensor (load cell or thermistor). The SN8F29E49 integrated Analog-to-Digital Converters (ADC) to achieve 20-bit performance and up to  $2^{16}$ -step resolution. The ADC builds in internal Gain Option with selective range of x1, x2, x4 and x8 for additional signal amplification expect PGIA. The PGIA provides positive and negative channels. There are AN1, AN2, AN3, AN4, AN5, OP2O, OP2-, OP2+, OP1O, OP1-, OP1+, DAO, ACM, GND, and Voltage detection mode channel included MUXP and MUXN. Input channel can be selected individually for different measurement modes. This ADC is optimized for measuring low-level unipolar or bipolar signals in weight scale and Glucose measurement applications. A very low noise chopper-stabilized programmable gain instrumentation amplifier (PGIA) with selectable gains of 1x, 16x, 32x, 64x, and 128x in the ADC to accommodate these applications.

The SN8F29E49 also integrated two rail to rail OP AMP and R-2R structure DAC. The comparators are Rail-to-Rail structure. That means the input/output voltage is real from VDD to VSS. The DAC is 12bit resolution with 10-Bit precision.

## ADC, regulator, DAC and OPA internal Diagram



**Note:** The  $OP2+$  ADC Channel MUX input path as below figure.



## 13.2 VOLTAGE REGULATOR

- AVDDR output 2.4V / 2.7V / 3.0V regulated from VDD with 10mA current driving capacity.
- AVE+ provides 2V for ADC/DAC reference source with maximum 5mA current driving capacity.
- ACM provides 1.0V voltage output supply.

## 13.3 VREG-CHARGE PUMP MODE REGISTER

90H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREG	BGEN	ACMEN	AVEN	AVDDREN	AVDDRS1	AVDDRS0	-	VREFS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W
After Reset	0	0	1	0	0	0	-	0

Bit0: **VREFS: Internal Vref source** control bit.

0 = Internal Vref source from AVE.

1 = Internal Vref source from AVDDR.

Bit[3:2]: **AVDDRS[1:0]**: AVDDR output voltage selection.

AVDDRS[1:0]	AVDDR Output Voltage
00	2.4V
01	2.7V
1x	3.0V

Bit4: **AVDDREN**: Regulator (AVDDR) voltage Enable control bit.

0 = Disable Regulator and AVDDR Output voltage.

1 = Enable Regulator and AVDDR Output voltage.

Bit5: **AVEN**: AVE+ voltage output control bit.

0 = Disable AVE+ output Voltage

1 = Enable AVE+ output Voltage

Bit6: **ACMEN**: Analog Common Mode (ACM) voltage Enable control bit.

0 = Disable Analog Common Mode voltage 1V.

1 = Enable Analog Common Mode voltage 1V.

Bit7: **BGEN**: Band Gap Reference voltage enable control bit

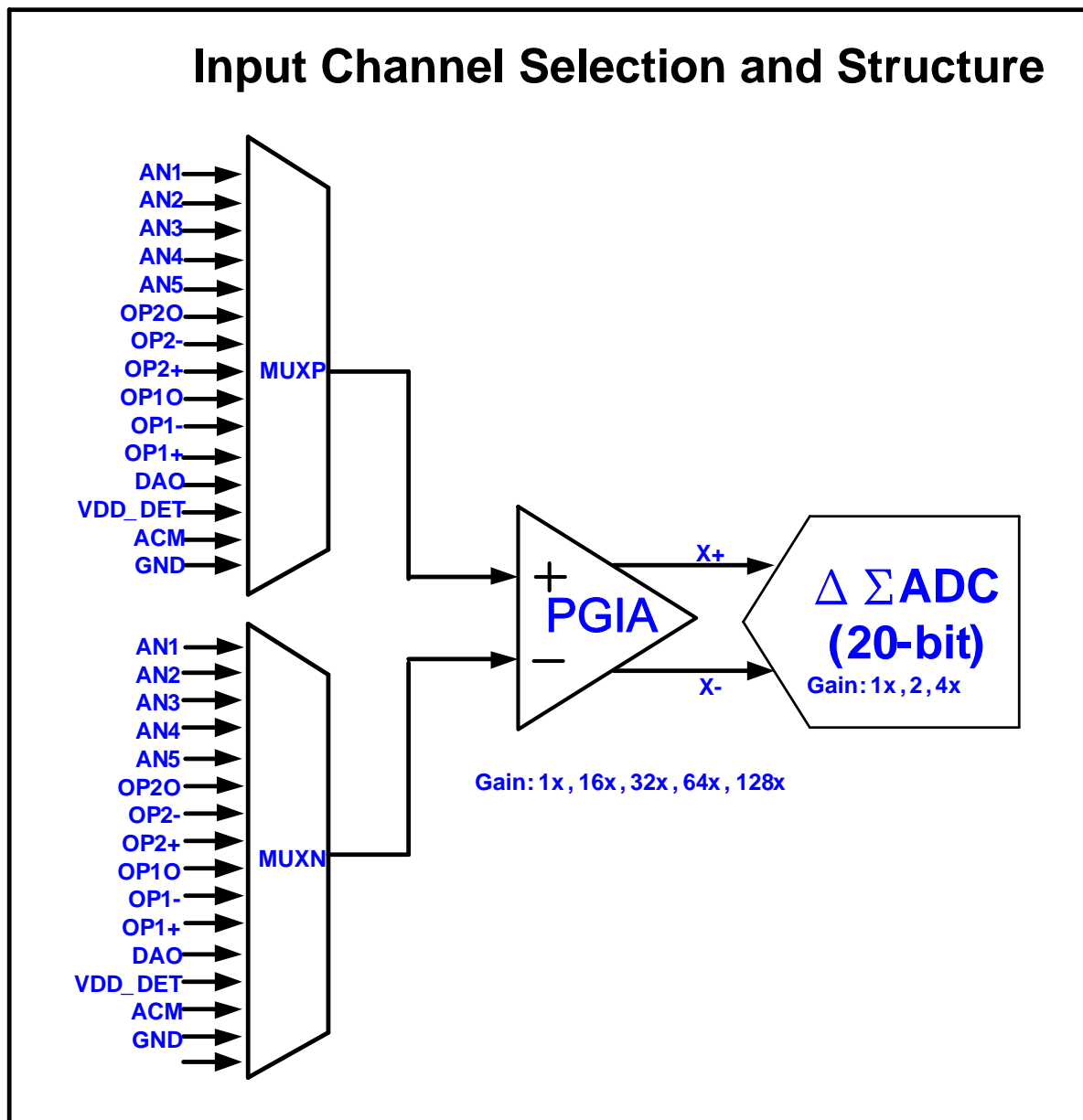
0 = Disable Band Gap Reference Voltage

1 = Enable Band Gap Reference Voltage

## 13.4 ADC INPUT CHANNEL SELECTION

The SN8F29E49 has a built-in channel selection to support ADC signal source. MUXP and MUXN have 15 channels select to input ADC. User can change difference ADC signal by setting CHS register. MUXP is connected to PGIA negative input channel, MUXN is connected to PGIA positive input channel.

ADC Input channels structure diagram



### 13.4.1 VCHS-ADC input channel selection Register

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
<b>CHS</b>	MUXP3	MUXP2	MUXP1	MUXP0	MUXN3	MUXN2	MUXN1	MUXN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

**Bit[3:0]:** MUXN[3:0]: ADC MUXN input Channel Selection

**Bit[7:4]:** MUXP[3:0]: ADC MUXP input Channel Selection

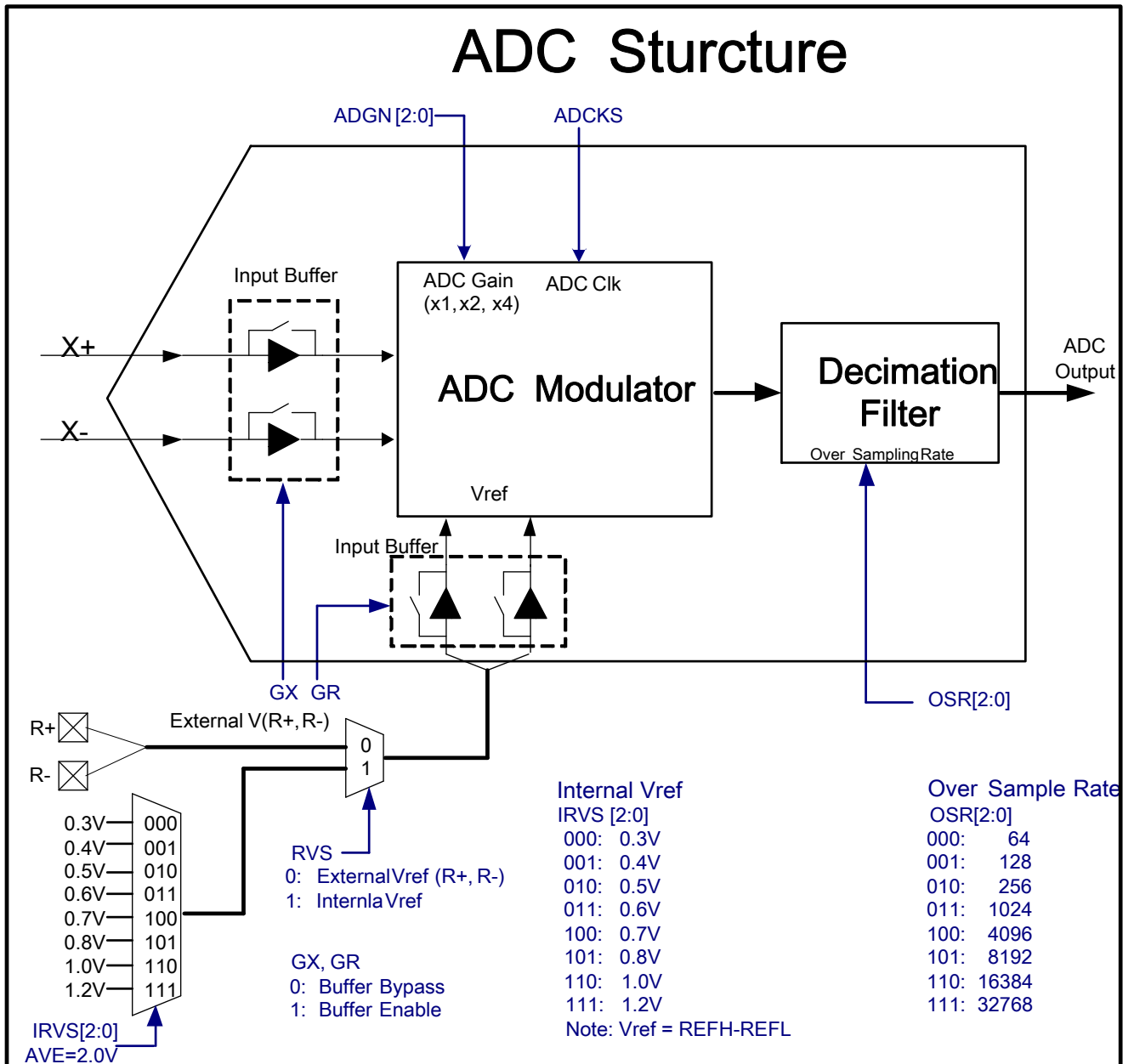
### 13.4.2 ADC input signal channel selection table

MUXP [3:0]	ADC X+ Input Channel	MUXN [3:0]	ADC X- Input Channel
0000	AN1	0000	AN1
0001	AN2	0001	AN2
0010	AN3	0010	AN3
0011	AN4	0011	AN4
0100	AN5	0100	AN5
0101	OP2O	0101	OP2O
0110	OP2-	0110	OP2-
0111	OP2+	0111	OP2+
1000	OP1O	1000	OP1O
1001	OP1-	1001	OP1-
1010	OP1+	1010	OP1+
1011	DAO	1011	DAO
1100	D+ (3/8*VDD)	1100	D- (2/8*VDD)
1101	ACM	1101	ACM
1110	GND	1110	GND
1111	X	1111	x

## 13.5 ADC STRUCTURE AND CONTROL REGISTER

### 13.5.1 OVERVIEW

The SN8F29E49 integrated a 20-bit  $\Delta\Sigma$  Analog-to-Digital Converters (ADC) with decimation filters can be set for variable throughputs range from 1Hz up to 3.9 kHz. A reference voltage (Vref) is built in internal with selective range from 0.3V to 0.8V in AVE=2V condition, or an external reference voltage can be used to adjust an adequate range via differential voltage between input pins of R+ and R-. The on-chip input buffers can be used to provide high input impedance for direct connection to sensitive transducers. The ADC builds in internal Gain Option with selective range of x1, x2 and x4 for additional signal amplification expect PGIA.



### 13.5.2 Analog Inputs and Voltage Operation Range

There are fifteen analog inputs for ADC and PGIA operation, including pins of AN1/AN2/AN3/AN4/AN5/OP2O/OP2-/OP2+/OP10/OP1-/OP1+/DAO/D+ or D-/ACM/GND. The analog inputs of PGIA, AN1/AN2/AN3/AN4/AN5/OP2O/OP2-/OP2+/OP10/OP1-/OP1+/DAO/D+ or D-/ACM/GND, are connected to external sensor's output signal, which can be configured as differential mode (AI+ to AI-) or single-end (AI± to ACM). External Vref for ADC is decided by differential voltage of input pin R+ and R-. All of analog inputs are restricted in absolute voltage range between 0.4V to 1.4V. Moreover, the output signals of PGIA, X+/X-, must also remain within the absolute voltage range.

### 13.5.3 Reference Voltage

There are two reference voltage (Vref) sources option for ADC operation. One is from internal Vref another is from external Vref. The ADC's Vref is selected using **RVS** and **IRVS[3:0]** bits in register **ADCM1**. When **RVS** bit is set to „1“, the ADC uses a internal Vref source which can be selected value from 0.3V~1.2V with voltage step via setting **IRVS[3:0]** bits. When **RVS** bit is cleared to „0“, the Vref is from external and the value is decided by differential voltage between Pins of **R+** and **R-**. Detail setting reference register **ADCM1**.

### 13.5.4 Input Buffer

Input Buffers are included ADC signal input buffer and ADC external reference input buffer R+/R-, which provide a high impedance of analog input, to minimized the input current of ADC for sensitive measurement and to avoid loading effect. When PGIA set 1x of application, the sensor output signal is bypass PGIA and direct connected to ADC's input. In that case, Input buffer function must be enabled by setting GX bit as „1“. If external Vref is selected for ADC, input buffer R+/R- also must be enabled by setting GR bit as „1“.

### 13.5.5 ADC Gain

The ADC builds in internal Gain Option with selective range of x1, x2, x4 and for additional signal amplification expect PGIA. The ADC Gain setting is controlled by **ADGN [1:0]** bits in register **ADCM1**. The analog signal after ADC Gain amplification, it can be adjusted offset level by subtraction or addition function, to increase the signal operation range of ADC in weigh-scales application. The following shows ADC output code calculation:

$$16bits : \frac{[(AI+) - (AI-) \times PGIA \times ADC\_Gain + V_{offset}}{Vref} \times 2^{(16-1)} = + 32768 \sim - 32768$$

$$18bits : \frac{[(AI+) - (AI-) \times PGIA \times ADC\_Gain + V_{offset}}{Vref} \times 2^{(18-1)} = + 131071 \sim - 131072$$

$$20bits : \frac{[(AI+) - (AI-) \times PGIA \times ADC\_Gain + V_{offset}}{Vref} \times 2^{(20-1)} = + 524287 \sim - 524288$$

PGIA : 1x ~ 128x

ADC\_Gain : 1x ,2x ,4x

Vref Source : Internal Vref or External Vref

Vref Range : 0.3V ~ 1.2V



## 13.6 ADCM1 -ADC MODE1 CONTROL REGISTER

92H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCM1</b>	OSR2	OSR1	OSR0	RVS	IRVS2	IRVS1	IRVS0	ADCEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	1	1	1	1	1	0	1	0

Bit[0]: **ADCEN**: ADC Enable Bit.

Bit[3:1]:**IRVS[2:0]**: ADC Internal Reference Voltage Selection.

Bit4: **RVS**: ADC Reference Voltage Internal/External Selection bit.

0 = Selection ADC Reference voltage from **External** reference R+, R-.

1 = Selection ADC Reference voltage from **Internal** reference with AVE.

RVS	VREFS	IRVS[2:0]	Reference source voltage					
			Vref source AVE			Vref source AVDDR 2.4V		
			Vref	REFH	REFL	Vref	REFH	REFL
0	x	x	(R+) – (R-)	R+ (Pad)	R- (Pad)	(R+) – (R-)	R+ (Pad)	R- (Pad)
1	0	000	0.3V	1.15V	0.85V	N.A		
		001	0.4V	1.20V	0.80V			
		010	0.5V	1.25V	0.75V			
		011	0.6V	1.30V	0.70V			
		100	0.7V	1.35V	0.65V			
		101	0.8V	1.40V	0.60V			
		110	1.0V	1.50V	0.50V			
		111	1.2V	1.60V	0.40V			
1	1	000	N.A			0.36V	1.38V	1.02V
		001				0.48V	1.44V	0.96V
		010				0.6V	1.5V	0.9V
		011				0.72V	1.56V	0.84V
		100				0.84V	1.62V	0.78V
		101				0.96V	1.68V	0.72V
		110				1.2V	1.8V	0.6V
		111				1.44V	1.92V	0.48V

Bit[7:5]: **ADC Over Sample Rate setting table.**

OSR [2:0]	OSR
000	64
001	128
010	256
011	1024
100	4096
101	8192
110	16384
111	32768

## 13.7 ADCM2 -ADC MODE1 CONTROL REGISTER

93H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCM2</b>	UGBEN	VDTEN	-	GX	GR	ADGN1	ADGN0	DRDY
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	-	0	0	0	0	0	0	0

Bit0: **DRDY**: ADC Conversion Ready bit:

"1" = ADC output (update) new conversion data to ADCDH, ADCDL, and ADCDLL

"0" = ADCDH, ADCDL, and ADCDLL conversion data are not ready.

Bit[2:1] **ADGN[1:0]**: ADC Gain Selection

ADGN[1:0]	ADC Gain
00	X1
01	X2
10	X4
11	Reserved

Bit3: **GR**: R+ R- Unit Gain Buffer Function control bit.

0 = Disable R+ R- UGB function.

1 = Enable R+ R- UGB function.

Bit4: **GX**: X+ X- Unit Gain Buffer Function control bit.

0 = Disable X+ X- UGB function.

1 = Enable X+ X- UGB function.

Bit6: **VDTEN**: VDD Voltage Detect function Enable bit.

0 = Disable VDD Voltage Detect function.

1 = Enable VDD Voltage Detect function.

Bit7: **UGBEN**: ADC offset voltage selection.

0 = ADC Gain x 1 offset voltage setting.

1 = ADC Gain x 4 offset voltage setting.

\* **Note:** Cancel VLCD Voltage Detect function, Only VDD Detection function

\* **Note:** ADC output stable data at the 3rd data after ADC enable. The 3rd, 4th, 5th ... are stable data after 1/WR later of each.

## 13.8 ADCM3-ADC MODE1 CONTROL REGISTER

94H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCM3</b>	CPCKEN	CPCSK1	CPCSK0	-	ADCKINV	ADCKS2	ADCKS1	ADCKS0
R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
After Reset	1	1	0	-	1	0	0	0

Bit[2:0]: **ADCKS[2:0]**: ADC Clock selection bit.

ADCKS[2:0]	ADC Clock	ADCKS[2:0]	ADC Clock
000	250kHz	100	333KHz
001	125KHz	101	166KHz
010	62.5KHz	110	83KHz
011	31.25KHz	111	41.6kHz

Bit[3]: ADCKINV: ADC Clock Inverse control bit.

**0: ADC Clock Non-Inverse.**

**1: ADC Clock Inverse.**

Bit[6:5] **CPCKS[1:0]**: ADC chopper selection.  
(Recommend **CPCKS[1:0]** setting "00" )

Bit[7]: CPCKEN: ADC Chopper Control bit.

**0: ADC Chopper Disable.**

**1: ADC Chopper Enable.**(Recommend Chopper enable)

CPCKEN	CPCKS[1:0]	ADC Chopper Freq.	Note
0	X	0	Chopper disable
1	00	ADCKS/32	For test
1	01	ADCKS/4	1. Sync. With ADCKS. 2. 1us delay after falling edge of ADCKS.
1	10	ADCKS/8	
1	11	ADCKS/16	

\* Note **ADC Output Rate = ADC Clock / OSR**

## 13.9 ADC DATA OUTPUT

9CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCDH</b>	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

9DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCDM</b>	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

9EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>ADCDL</b>	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0
R/W	R	R	R	R	R	R	R	R
After Reset	0	0	0	0	0	0	0	0

**ADCDH[7:0]:** Output high byte data of ADC conversion word.

**ADCDM[7:0]:** Output medium byte data of ADC conversion word.

**ADCDL [7:0]:** Output low byte data of ADC conversion word.

<b>ADC conversion data (2's compliment, Hexadecimal)</b>	<b>Decimal Value</b>
0x7FFFFH	524287
...	...
0x40000H	262144
...	...
0x10000H	65536
...	...
0x00002H	2
0x00001H	1
0x00000H	0
0xFFFFFH	-1
0xFFFFEH	-2
...	...
0xF0000H	-65536
...	...
0xC0000H	-262144
...	...
0x80000H	-524288

\* **Note 1:** ADCDH [7:0], ADCDM [7:0] and ADCDL [7:0] are read only registers.

\* **Note2:** For 16-Bit ADC resolution, please use registers of ADCDH and ADCDM (ADCB23~ADCB08).

**For 18-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.**  
(ADCB23~ADCB06).

**For 20-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.**  
(ADCB23~ADCB04).

**For 24-Bit ADC resolution, please use registers of ADCDH, ADCDM and ADCDL.**  
(ADCB23~ADCB00)

\* **Note3:** The ADC conversion data is combined with ADCDH, ADCDM, ADCDL in 2's compliment with sign bit numerical format, and Bit ADCB23 is the sign bit of ADC data.

\* **Note 3:** ADCB23=0 means data is Positive value, ADCB23=1 means data is Negative value.

\* **Note 4:** The Positive Full-Scale-Output value of ADC conversion is 0x7FFFF.

\* **Note 5:** The Negative Full-Scale-Output value of ADC conversion is 0x80000H.

\* **Note 6:** Because of the ADC design limitation, the ADC Linear range is +131071 ~ -131072 (18-bit).

Following table shows the Noise and ENOB (RMS and peak-to-peak) of the SN8F29E49 ADC with different output word rate rates and gain settings. These numbers are typical and are generated using a differential input-short condition, ADC Vref 0.8V and 1024-data of measurement.

Gain (PGIA x ADC)	OSR	WR	Noise Free Resolution (1)	Effective Resolution (2)	Pk-Pk Noise(4)	RMS Noise(3)
1 x 1	256	977 Hz	14.20	16.92	84.81	16.92
16 x 1	256	977 Hz	13.59	16.31	8.1	16.31
32 x 1	256	977 Hz	13.10	15.82	5.69	15.82
64 x 1	256	977 Hz	12.31	15.03	4.93	15.03
128 x 1	256	977 Hz	11.31	14.03	4.93	14.03
1 x 1	32768	7.6Hz	18.22	20.94	5.24	0.79
1 x 1	16384	15.3Hz	17.30	20.02	9.92	1.50
1 x 1	8192	31Hz	16.37	19.09	18.89	2.86
1 x 1	4096	61Hz	15.78	18.5	28.44	4.31
1 x 1	1024	244Hz	15.04	17.76	47.49	7.20
1 x 1	256	977 Hz	14.20	16.92	85.01	12.88
1 x 1	128	2.0 kHz	13.63	16.35	126.21	19.12
1 x 1	64	3.9 kHz	11.86	14.58	430.43	65.22
<b>All Test condition: ADC 250kHz, Input-Short, Vref=0.8V, Gain = PGIA x ADC, Collect 1024 ADC date.</b> <b>(1).Noise Free Resolution = <math>\text{Log}_2</math> (Full Scale Range / Peak-Peak Noise)</b> where Full Scale Range = $2 \times \text{Vref} / \text{Gain}$ (ex. Vref=0.8V, Gain=128x) <b>(2).Effective Resolution = <math>\text{Log}_2</math> (Full Scale Range / RMS_Noise)</b> <b>(3).RMS Noise = <math>\sigma \times \text{LSB\_Resolution}</math></b> where $\text{LSB\_Resolution} = \text{Full Scale Range} / 2^{\text{Bit}}$ , Bit=20 $\sigma$ = standard deviation of 1024 ADC output data. <b>(4). Peak-Peak Noise = <math>6.6 \times \text{RMS Noise}</math>, or code variation range x <math>\text{LSB\_Resolution}</math></b> where Code variation range = ADC counts max-min of 1024 data.						

## 13.10 PGIA CONTROL REGISTER

9FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>AMPM</b>	BGRCMP	BGRCHPEN	SW14	GS2	GS1	GS0	PCHPEN	AMPEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	1	0	0	0	0	0	0

**Bit[0]: AMPEN:** PGIA Enable Control bit.  
 0: PGIA disable. (PGIA Chopper auto off)  
 1: PGIA Enable. (PGIA Chopper controlled by PCHPEN)

**Bit[1]: PCHPEN:** PGIA Chopper control bit.  
 0: PGIA chopper disable.  
 1: PGIA chopper Enable.

**Bit[4:2]: GS[2:0]:** PGIA Gain selection.

GS[2:0]	PGIA Gain
<b>000</b>	16x
<b>001</b>	32x
<b>010</b>	64x
<b>011</b>	128x
<b>1xx</b>	1x

Note: 1x bypass PGIA.

**Bit5: SW14:** SC14 Enable bit.  
 0 = Disable SC14 function.  
 1 = Enable SC14 function.

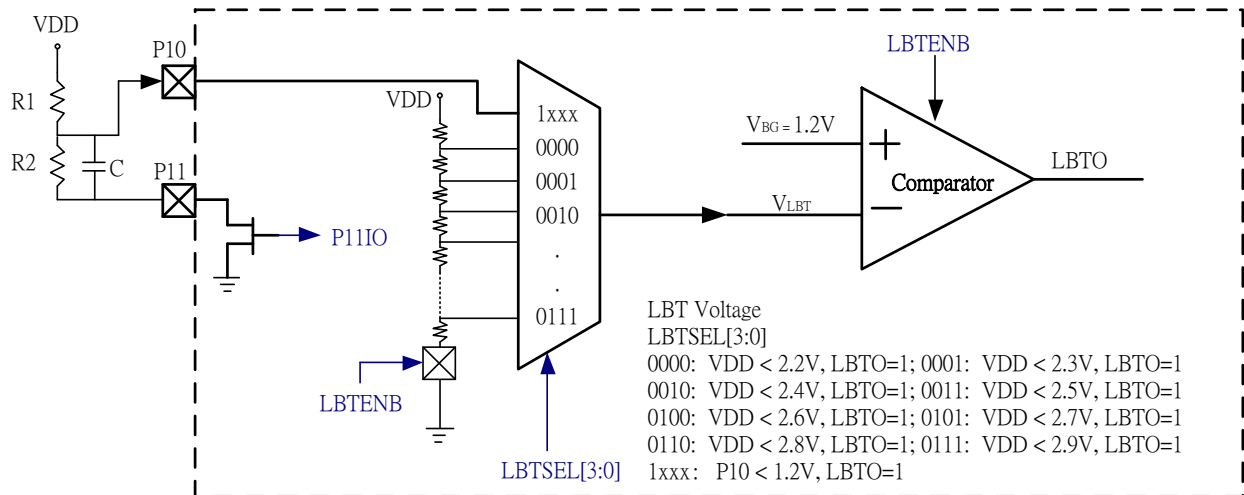
**Bit6: BGRCHPEN:** Band Gap chopper control bit.  
 0 = Band Gap chopper disable.  
 1 = Band Gap chopper enable.

**Bit7: BGRCMP:** chopper clock source selection  
 0 = 62.5K chopper clock. (Recommend selection)  
 1 = 500K chopper clock.

## 13.11 LBTM: LOW BATTERY DETECT

### 13.11.1 OVERVIEW

SN8F29E49 provided two different ways to measure VDD Voltage. One is from ADC reference voltage selection. It will be more precise but take more time and a little bit complex. Another way is using build in Voltage Comparator via internal or external input path to detect VDD voltage level. There are eight internal levels, every 0.1V one level from 2.2V~2.9V. The Function can be set for low.



### 13.11.2 LBTM: Low Battery Detect

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>LBTM</b>	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTEN	LBTEN
R/W	-	R/W	R/W	R/W	R/W	R/W	R	R/W
After Reset	-	0	0	0	0	0	0	0

Bit0: **LBTENB**: Low Battery Detect mode control Bit.  
0 = Disable Low Battery Detect function,  
1 = Enable Low Battery Detect function

Bit1: **LBTEN**: Low Battery Detect Output Bit.  
0 = LBT voltage ( $V_{LBT}$ ) Higher than Band Gap Reference Voltage 1.2V.  
1 = LBT voltage ( $V_{LBT}$ ) Lower than Band Gap Reference Voltage 1.2V.

Bit[5:2]: **LBTSEL[3:0]**: Low Battery Detect threshold voltage selection bit.

LBTENB	LBTSEL [3:0]	LBTEN = 1	Note
0	-	-	LBT Function disable
1	0000	VDD < 2.2V	Internal Input
1	0001	VDD < 2.3V	Internal Input
1	0010	VDD < 2.4V	Internal Input
1	0011	VDD < 2.5V	Internal Input
1	0100	VDD < 2.6V	Internal Input
1	0101	VDD < 2.7V	Internal Input
1	0110	VDD < 2.8V	Internal Input
1	0111	VDD < 2.9V	Internal Input
1	1xxx	P10 < 1.2V, LBTEN=1	External Input

## 13.12 OPM REGISTER

SN8F29E49 provided two OP-Amps user application. There are 11 switches for OP1 and 1 switch for OP2. When SC0 turn on condition; the OP1 became a unit gain buffer. This circuit structure is in following figure.

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>OPM1</b>	SW5	SW4	SW3	SW2	SW1	SW0	OP2EN	OP1EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
After Reset	0	0	0	0	0	0	0	0

Bit0: **OP1EN:** OP1 Enable Bit.  
0 = Disable OP1 function,  
1 = Enable OP1 function

Bit1: **OP2EN:** OP2 Enable Bit.  
0 = Disable OP2 function,  
1 = Enable OP2 function

Bit2: **SW0:** SC0 Enable bit  
0 = Disable SC0 function,  
1 = Enable SC0 function

Bit3: **SW1:** SC1 Enable bit  
0 = Disable SC1 function,  
1 = Enable SC1 function

Bit4: **SW2:** SC2 Enable bit  
0 = Disable SC2 function,  
1 = Enable SC2 function

Bit5: **SW3:** SC3 Enable bit  
0 = Disable SC3 function,  
1 = Enable SC3 function

Bit6: **SW4:** SC4 Enable bit  
0 = Disable SC4 function,  
1 = Enable SC4 function

Bit7: **SW5:** SC5 Enable bit  
0 = Disable SC5 function,  
1 = Enable SC5 function



09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>OPM2</b>	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit0: **SW6:** SC6 Enable Bit.  
 0 = Disable SC6 function  
 1 = Enable SC6 function

Bit1: **SW7:** SC7 Enable bit  
 0 = Disable SC7 function  
 1 = Enable SC7 function

Bit2: **SW8:** SC8 Enable bit  
 0 = Disable SC8 function  
 1 = Enable SC8 function

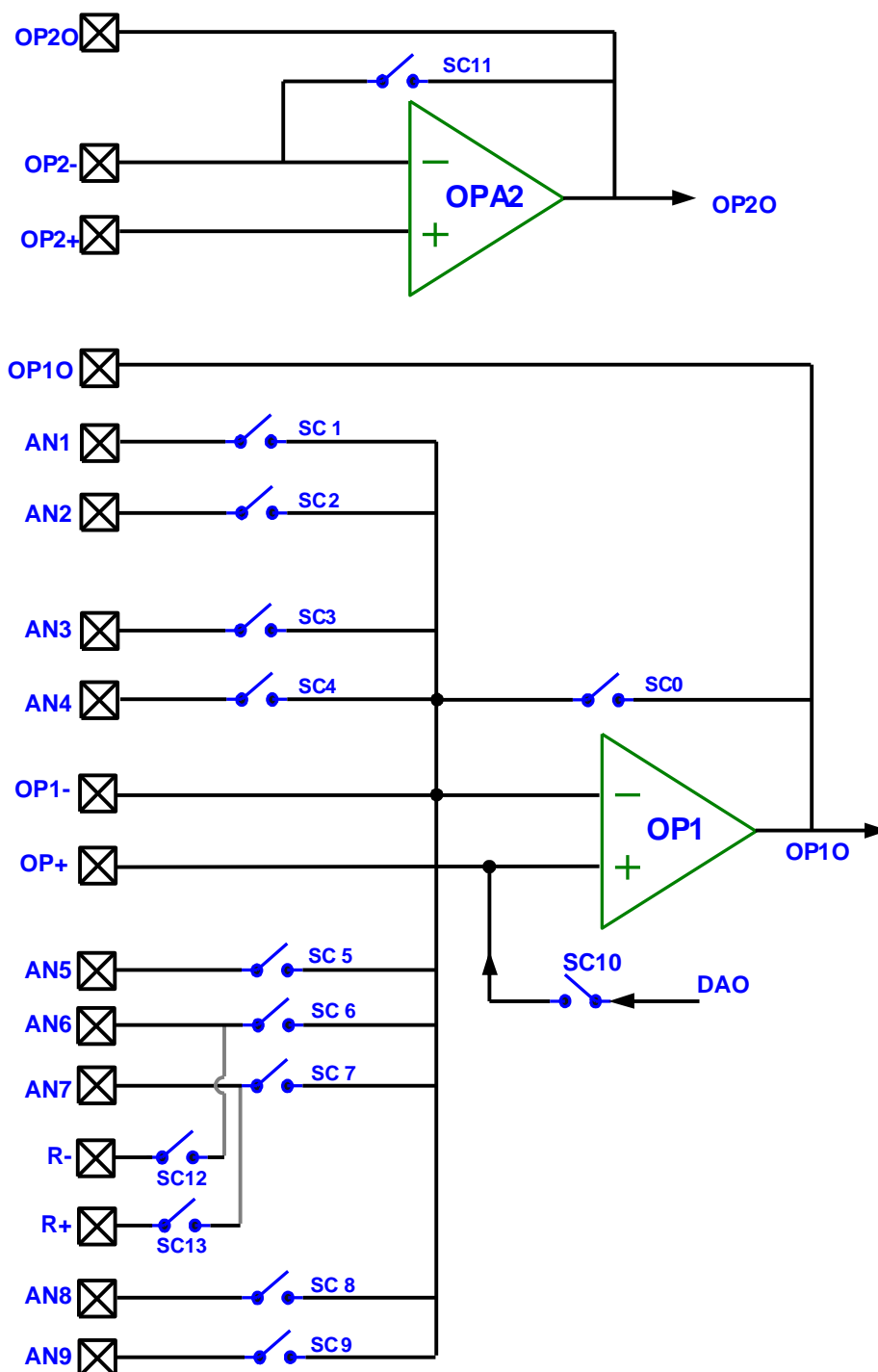
Bit3: **SW9:** SC9 Enable bit  
 0 = Disable SC9 function  
 1 = Enable SC9 function

Bit4: **SW10:** SC10 Enable bit  
 0 = Disable SC10 function  
 1 = Enable SC10 function

Bit5: **SW11:** SC11 Enable bit  
 0 = Disable SC11 function  
 1 = Enable SC11 function

Bit6: **SW12:** SC12 Enable bit  
 0 = Disable SC12 function  
 1 = Enable SC12 function

Bit7: **SW13:** SC13 Enable bit  
 0 = Disable SC13 function  
 1 = Enable SC13 function



## 13.13 1-CHANNEL DIGITAL TO ANALOG CONVERTER

### 13.13.1 OVERVIEW

The DAC structure is 12-bit resolution; it can generate analog signal on DAO pin. DAO output voltage range= 0V ~ DAREF.

### 13.13.2 DAC Register

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DAM</b>	DAEN	-	-	-	-	DAF2	DAF1	DAF0
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After Reset	0	-	-	-	-	0	0	0

Bit 7: **DAEN:** DAC control bit.  
0 = Disable DAC function.  
1 = Enable DAC function.

Bit [2:0]: **DAF [1:0]** = DAC VREF setting bit.

DAF [2:0]	Note
0xx	Vref = external voltage (DAREF pad input)
100	Vref = internal 0.2V
101	Vref = internal 0.35V
110	Vref = internal 0.65V
111	Vref = internal 1.20V

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DABH</b>	-	-	-	-	DAB11	DAB10	DAB9	DAB8
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
After Reset	-	-	-	-	0	0	0	0

Bit [3:0] **DAB [3:0]** = The low-nibble byte is DAC [8:11] data.

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>DABL</b>	DAB7	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After Reset	0	0	0	0	0	0	0	0

Bit [7:0] **DABL [7:0]** = DAC low byte data.

- \* **Note:** DAB[11:0] written sequence is write DABL[8:0] first, and then write DABH[4:0].
- \* **Note:** Max DAREF input voltage=1.2V.

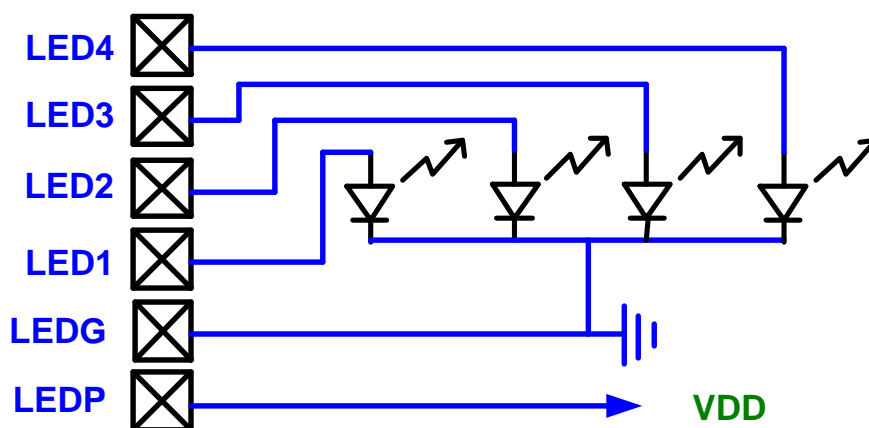
The DAB register, from bit0 to bit11, to generate analog signal on DAO pin. The DAO output signal voltage range is 0V ~ VREFH-0.1V. The DAB's data V.S DAO's output voltage as following:

DABH[3:0]				DABL[7:0]								DAO Output Voltage (V)
DAB11	DAB10	DAB9	DAB8	DAB7	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	
0	0	0	0	0	0	0	0	0	0	0	0	VSS
0	0	0	1	0	0	0	0	0	0	0	0	$1/4095 \cdot \text{DAREF}$
0	0	1	0	0	0	0	0	0	0	0	0	$2/4095 \cdot \text{DAREF}$
0	0	1	1	0	0	0	0	0	0	0	0	$3/4095 \cdot \text{DAREF}$
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	1	0	1	1	1	1	1	1	1	1	$4094/4095 \cdot \text{DAREF}$
1	1	1	1	1	1	1	1	1	1	1	1	DAREF

# 14 LED DRIVER

## 14.1 OVERVIEW

SN8F29E49 provided fixed current output 4 pins, which is for LED driving purpose. Its driving current is 4mA~10mA. LED constant current IP has independent VDD/GND Pads, named LEDP and LEDG. SN8F29E49 support four LED backlight design for LCD application. LED luminance is controlled by LEDPW register setting.



## 14.2 LED REGISTER

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>LEDM</b>	-	LEDIO	LEDPW1	LEDPW0	LED4EN	LED3EN	LED2EN	LED1EN
R/W	-	R/W	R/W	R/W	R/W	R/W	R	R/W
After Reset	-	0	0	1	0	0	0	0

Bit[3:0] **LEDxEN: LED Current source pin enable bitt.**

0 = Disable LED function,  
1 = Enable LED function

Bit[5:4] **LEDPW[1:0]: LED Output Current control bitt.**

00 = Output current 4mA  
01 = Output current 5mA  
10 = Output current 6mA  
11 = Output current 10mA

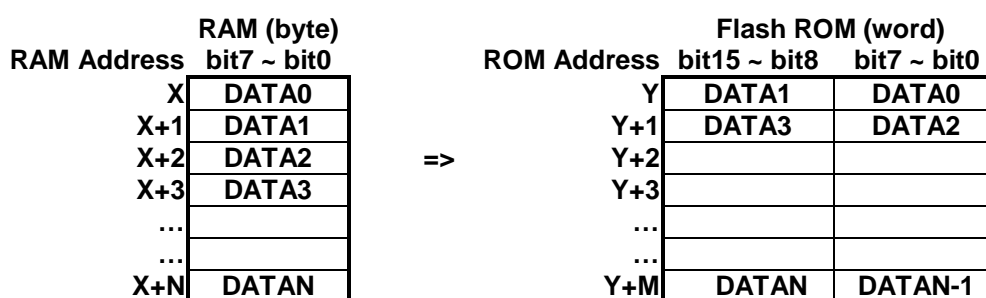
Bit[6] **LEDIO: P2 function selection**

0 = Selection I/O function  
1 = Selection LED function

# 15 IN SYSTEM PROGRAM FLASH ROM

## 15.1 OVERVIEW

The SN8F29E49 MCU integrated device feature in-system programmable (ISP) FLASH memory for convenient, upgradeable code storage. The FLASH memory may be programmed via the SONiX 8 bit MCU programming interface or by application code. The SN8F29E49 provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory. ISP Flash ROM provided user an easy way to storage data into Flash ROM. The ISP concept is memory mapping idea that is to move RAM buffer to flash ROM. Choice ROM/RAM address and executing ROM programming command – PECMD, after programming words which controlled by PERAMCNT, PERAML/PERAMCNT data will be programmed into address PEROML/PEROMH.



During Flash program or erase operation, the MCU is stalled, although peripherals (Timers, WDT, I/O, PWM, etc.) remain active. When PECMD register is set to execute ISP program and erase operations, the program counter stops, op-code can't be dumped from flash ROM, instruction stops operating, and program execution is hold not to active. At this time hardware depends on ISP operation configuration to do flash ROM erasing and flash ROM programming automatically. After ISP operation is finished, hardware releases system clock to make program counter running, system returns to last operating mode, and the next instruction is executed. Recommend to add two "NOP" instructions after ISP operations.

ISP flash ROM erase time = 25ms.....1-page, 256-word.

ISP flash ROM program time = 28us.....1-word.

ISP flash ROM program time = 56us.....2-word.

...

ISP flash ROM program time = 448us.....16-word.

...

ISP flash ROM program time = 896us.....32-word.

**Note:**

*Watch dog timer should be clear before the Flash write (program) or erase operation, or watchdog timer would overflow and reset system during ISP operating.*

*Besides program execution, all functions keep operating during ISP operating, e.g. timer, ADC, UART, MSP... All interrupt events still active and latch interrupt flags automatically. If any interrupt request occurs during ISP operating, the interrupt request will be process by program after ISP finishing.*

## 15.2 ISP FLASH ROM ERASE OPERATION

ISP flash ROM erase operation is to clear flash ROM contents to blank status “1”. Erasing ROM length is 256-word and has ROM page limitation. ISP flash ROM erase ROM map is as following:

has ROM page limitation. ISP flash ROM erase ROM map is as following.

ISP ROM MAP		ROM address bit0~bit6 (hex)														
		0000	0001	0002	...	0010	0011	...	0050	0051	...	00E0	00E1	...	00FE	00FF
ROM address bit7~bit15 (hex)	0000	This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.														
	0100	One ISP Erase Page														
	0200	One ISP Erase Page														
	0300	One ISP Erase Page														
	0400	One ISP Erase Page														
	0500	One ISP Erase Page														
	...	One ISP Erase Page														
	0F00	One ISP Erase Page														
	1000	One ISP Erase Page														
	1100	One ISP Erase Page														
	1200	One ISP Erase Page														
	1300	One ISP Erase Page														
	1400	One ISP Erase Page														
	...	One ISP Erase Page														
	FC00	One ISP Erase Page														
	FD00	One ISP Erase Page														
	FE00	One ISP Erase Page														
	FF00	This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.														

ISP flash ROM erase density is 256-word which limits erase page boundary. The first 256-word of flash ROM (0x0000~0x00FF) includes reset vector and interrupt vectors related essential program operation, and the last page 256-word of flash ROM (0xFE00~0xFFFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM erase operation in the two pages. Flash ROM area 0x0000~0xFF00 includes 256-page for ISP flash ROM erase operation.

The first step to do ISP flash ROM erase is to address ROM-page location. The address must be the head location of a page area, e.g. 0x0100, 0x0200, 0x0300...0xFD00, 0xFE00 and 0xFF00. PEROML [7:0] and PEROMH [7:0] define the target starting address [15:0] of flash ROM. Write the start address into PEROML and PEROMH registers, set PECMD register to “0xC3”, and the system start to execute ISP flash ROM erase operation.

**Example : Use ISP flash ROM erase to clear 0x0080~0x00FF contents of flash ROM.**

**; Set erased start address 0x0080.**

```
MOV      A, #0x80
B0MOV    PEROML, A           ; Move low byte address 0x80 to PEROML.
MOV      A, #0x00
B0MOV    PEROMH, A           ; Move high byte address 0x00 to PEROMH
```

**; Clear watchdog timer.**

```
MOV      A, #0X5A
B0MOV    WDTR, A
```

**; Start to execute ISP flash ROM erase operation.**

```
MOV      A, #0XC3           ; Start to page erase.
B0MOV    PECMD, A
NOP                                     ; NOP Delay
NOP                                     ; The end of ISP flash ROM erase operation.
```

The two “NOP” instructions make a short delay to let system stable after ISP flash ROM erase operation.

**Note: Don't execute ISP flash ROM erase operation for the first page and the last page, or affect program operation.**

## 15.3 ISP FLASH ROM PROGRAM OPERATION

ISP flash ROM program operation is to write data into flash ROM by program. Program ROM doesn't limit written ROM address and length, but limits 32-word density of one page. The number of ISP flash ROM program operation can be 1-word ~ 32-word at one time, but these words must be in the same page. ISP flash ROM program ROM map is as following:

ISP ROM MAP		ROM address bit0~bit4 (hex)								
		0000	0001	0002	...	000F	0010	...	001E	001F
ROM address bit5~bit15 (hex)	0000	This page includes reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP program.								
	0020	One ISP Program Page								
	0040	One ISP Program Page								
	0060	One ISP Program Page								
	0080	One ISP Program Page								
	00A0	One ISP Program Page								
	00C0	One ISP Program Page								
	00E0	One ISP Program Page								
	0100	One ISP Program Page								
	0120	One ISP Program Page								
	...	One ISP Program Page								
	1000	One ISP Program Page								
	1020	One ISP Program Page								
	...	One ISP Program Page								
	1F00	One ISP Program Page								
	1F20	One ISP Program Page								
	...	One ISP Program Page								
	FF80	This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP program.								

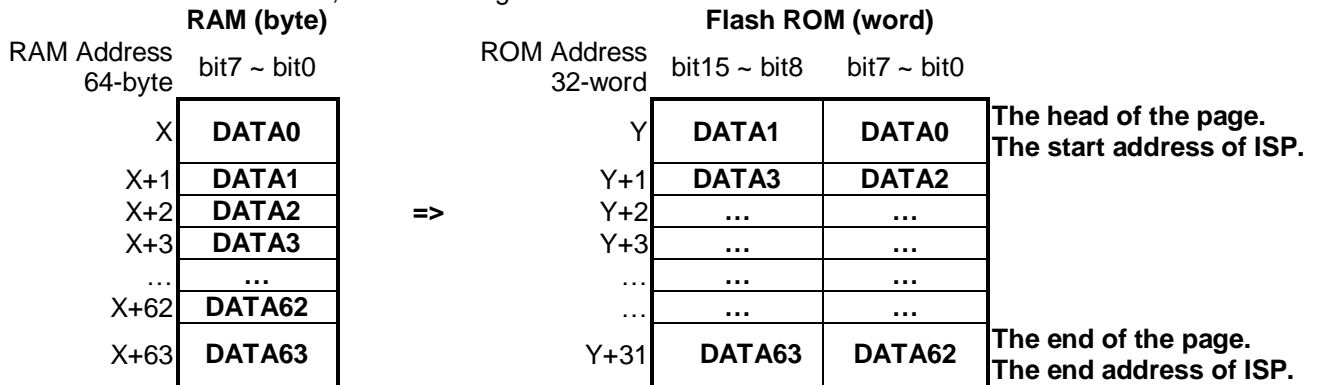
ISP flash ROM program page density is 32-word which limits program page boundary. The first 32-word of flash ROM (0x0000~0x001F) includes reset vector and interrupt vectors related essential program operation, and the last page 32-word of flash ROM (0xFF80~0xFFFF) includes system reserved ROM area, we strongly recommend do not execute ISP flash ROM program operation in the two pages. Flash ROM area 0x0020~0xFF7F includes 2044-page for ISP flash ROM program operation.

ISP flash ROM program operation is a simple memory mapping operation. The first step is to plan a RAM area to store programmed data and keeps the RAM address for ISP RAM addressing. The second step is to plan a ROM area will be programmed from RAM area in ISP flash ROM program operation. The RAM addressing is through PERAML[12:0] 13-bit buffer to configure the start RAM address. The RAM data storage sequence is down-up structure. The first RAM data is the low byte data of the first word of flash ROM. The second RAM data is the high byte data of the first word of ROM, and so on.

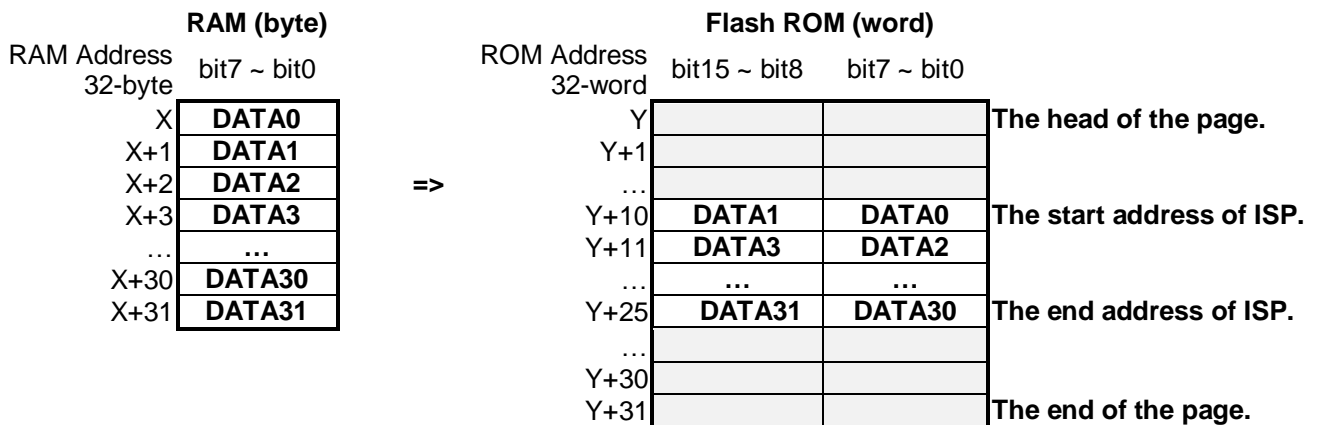
ISP programming length is 1-word~32-word. ISP flash ROM programming length is controlled by PERAMCNT[7:3] bits which is 5-bit format. Before ISP ROM programming execution, set the length by program. PEROML [7:0] and PEROMH [7:0] define the target starting address [15:0] of flash ROM. Write the start address into PEROML and PEROMH registers, set PECMD register to "0x5A", and the system start to execute ISP flash ROM program operation. If the programming length is over ISP flash ROM program page boundary, the hardware immediately stops programming flash ROM after finishing programming the last word of the ROM page. So it is very important to plan right ROM address and programming length.



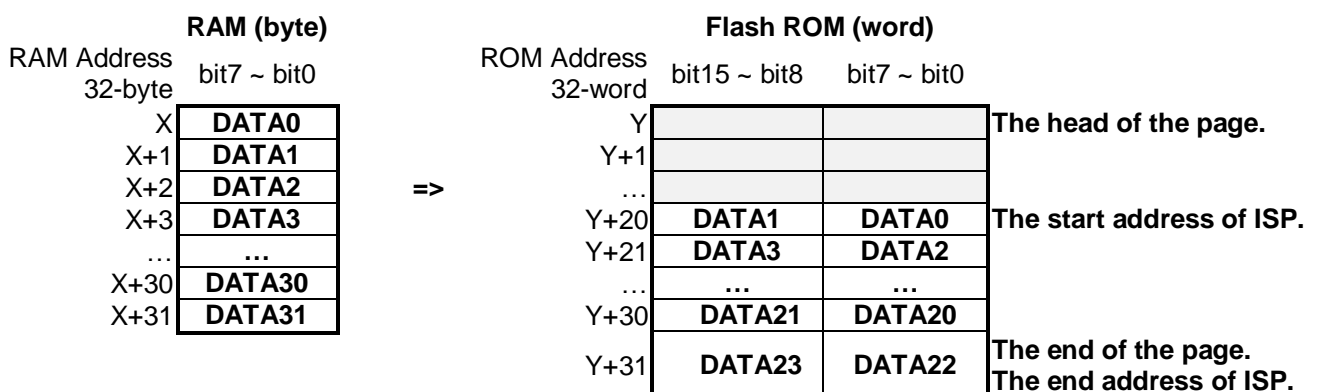
**Case 1:** 32-word ISP program. RAM buffer length is 64-byte and RAM address is X ~ X+63. PERAMCNT[7:3] = 11111b meets a complete one page 32-word of flash ROM. The page address of flash ROM is Y ~ Y+31. The Y is the start address and set to PEROML, PEROMH registers.



**Case 2:** 16-word ISP program: RAM buffer length is 32-byte. PERAMCNT [7:3] = 01111b meets 16-word of flash ROM. The page address of flash ROM is Y ~ Y+31, but the start address isn't the head of the page. Define the start address is Y+10 and set to PEROML, PEROMH registers. The programmed flash ROM area is Y+10~Y+25 addresses.



**Case 3:** Follow above case and change the ROM start address to Y+20. The programmed flash ROM area is Y+20~Y+35 addresses. The ROM range is out of the page boundary. After ISP flash ROM operation, the last 4-word data can't be written into flash ROM successfully. The programming length is over ISP flash ROM program page boundary, the hardware immediately stops programming flash ROM after finishing programming the last word (Y+31) of the ROM page.



**Example: Use ISP flash ROM program to program 32-word data to flash ROM as case 1. Set RAM buffer start address is 0x010. Set flash ROM programmed start address is 0x0020.**

**; Load data into 64-byte RAM buffer.**

...  
...

**; Set RAM start address of 64-byte buffer.**

```
MOV      A, #0x10
B0MOV    PERAML, A
MOV      A, #0x00
B0MOV    PERAMH, A
```

; Set PERAML [7:0] to 0x10.

; Set PERAMH [12:8] to 000b.

**; Set ISP program length to 32-word.**

```
MOV      A, #11111000b
B0MOV    PERAMCNT, A
```

; Set PERAMCNT[7:3] to 11111b.

**; Set programmed start address of flash ROM to 0x0020.**

```
MOV      A, #0x20
B0MOV    PEROML, A
MOV      A, #0x00
B0MOV    PEROMH, A
```

; Move low byte address 0x20 to PEROML.

; Move high byte address 0x00 to PEROMH

**; Clear watchdog timer.**

```
MOV      A, #0X5A
B0MOV    WDTR, A
```

**; Start to execute ISP flash ROM program operation.**

```
MOV      A, #0X5A
B0MOV    PECMD, A
NOP
NOP
```

; Start to program flash ROM.

; NOP Delay

; The end of ISP flash ROM program operation.

The two "NOP" instructions make a short delay to let system stable after ISP flash ROM program operation.

**Note: Don't execute ISP flash ROM program operation for the first page and the last page, or affect program operation.**

## 15.4 ISP PROGRAM/ERASE CONTROL REGISTER

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PECMD</b>	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0
Read/Write	W	W	W	W	W	W	W	W
After reset	-	-	-	-	-	-	-	-

Bit [7:0] **PECMD [7:0]**: ISP operation control register.  
 0x5A: Page Program (32 words / page).  
 0xC3: Page Erase (256words / page).  
 Others: Reserved.

**Note:** Before executing ISP program and erase operations, clear PECMD register is necessary. After ISP configuration, set ISP operation code in “MOV A,I” and “B0MOV M,A” instructions to start ISP operations.

## 15.5 ISP ROM ADDRESS REGISTER

ISP ROM address length is 16-bit and separated into PEROML and PEROMH registers. Before ISP execution, set the head address of ISP ROM by program.

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PEROML</b>	PEROML7	PEROML6	PEROML5	PEROML4	PEROML3	PEROML2	PEROML1	PEROML0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PEROML[7:0]**: The low byte buffer of ISP ROM address.

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PEROMH</b>	PEROMH7	PEROMH6	PEROMH5	PEROMH4	PEROMH3	PEROMH2	PEROMH1	PEROMH0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PEROMH[7:0]**: The high byte buffer of ISP ROM address.

## 15.6 ISP RAM ADDRESS REGISTER

ISP RAM address length is 13-bit and separated into PERAML register and PERAMH register. Before ISP execution, set the head address of ISP RAM by program.

0DDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PERAML</b>	PERAML7	PERAML6	PERAML5	PERAML4	PERAML3	PERAML2	PERAML1	PERAML0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

Bit [7:0] **PERAML[7:0]**: ISP RAM address [7:0].

0DEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PERAMH</b>	-	-	-	PERAMH4	PERAMH3	PERAMH2	PERAMH1	PERAMH0
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	0	0	0	0	0

Bit [4:0] **PERAMH[4:0]**: ISP RAM address [12:8].

## 15.7 ISP ROM PROGRAMMING LENGTH REGISTER

ISP programming length is 1-word ~ 32-word. ISP ROM programming length is controlled by PERAMCNT[7:3] bits which is 5-bit format. Before ISP ROM programming execution, set the length by program.

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PERAMCNT</b>	PERAMCNT4	PERAMCNT3	PERAMCNT2	PERAMCNT1	PERAMCNT0	-	-	-
Read/Write	R/W	R/W	R/W	R/W	R/W	-	-	-
After reset	0	0	0	0	0	-	-	-

Bit [7:3] **PERAMCNT[7:3]**: ISP ROM programming length control register.

$$\text{ISP programming length} = \text{PERAMCNT}[7:3] + 1$$

PERAMCNT[7:3]=0: ISP programming length is 1-word.

PERAMCNT[7:3]=1: ISP programming length is 2-word.

...

...

PERAMCNT[7:3]=30: ISP programming length is 31-word.

PERAMCNT[7:3]=31: ISP programming length is 32-word.

**Note:** Defines the number of words wanted to be programmed. The maximum PERAMCNT [7:3] is 01FH, which program 32 words (64 bytes RAM) to the Flash. The minimum PERAMCNT [7:3] is 00H, which program only 1 word to the Flash.

# 16 INSTRUCTION TABLE

Field	Mnemonic	Description	C	DC	Z	Cycle
MOV O V E	MOV A,M	$A \leftarrow M$	-	-	√	1
	MOV M,A	$M \leftarrow A$	-	-	-	1
	B0MOV A,M	$A \leftarrow M$ (bank 0)	-	-	√	1
	B0MOV M,A	$M$ (bank 0) $\leftarrow A$	-	-	-	1
	MOV A,I	$A \leftarrow I$	-	-	-	1
	B0MOV M,I	$M \leftarrow I$ , "M" only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z,...)	-	-	-	1
	XCH A,M	$A \leftrightarrow M$	-	-	-	1+N
	B0XCH A,M	$A \leftrightarrow M$ (bank 0)	-	-	-	1+N
	MOVC	$R, A \leftarrow ROM[Y,Z]$	-	-	-	2
A R I T H M E T I C	ADC A,M	$A \leftarrow A + M + C$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1
	ADC M,A	$M \leftarrow A + M + C$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1+N
	ADD A,M	$A \leftarrow A + M$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1
	ADD M,A	$M \leftarrow A + M$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1+N
	B0ADD M,A	$M$ (bank 0) $\leftarrow M$ (bank 0) + $A$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1+N
	ADD A,I	$A \leftarrow A + I$ , if occur carry, then $C=1$ , else $C=0$	√	√	√	1
	SBC A,M	$A \leftarrow A - M - /C$ , if occur borrow, then $C=0$ , else $C=1$	√	√	√	1
	SBC M,A	$M \leftarrow A - M - /C$ , if occur borrow, then $C=0$ , else $C=1$	√	√	√	1+N
	SUB A,M	$A \leftarrow A - M$ , if occur borrow, then $C=0$ , else $C=1$	√	√	√	1
	SUB M,A	$M \leftarrow A - M$ , if occur borrow, then $C=0$ , else $C=1$	√	√	√	1+N
	SUB A,I	$A \leftarrow A - I$ , if occur borrow, then $C=0$ , else $C=1$	√	√	√	1
	DAA	To adjust ACC's data format from HEX to DEC.	√	-	-	1
L O G I C	MUL A,M	$R, A \leftarrow A * M$ , The LB of product stored in Acc and HB stored in R register. ZF affected by Acc.	-	-	√	2
	AND A,M	$A \leftarrow A$ and $M$	-	-	√	1
	AND M,A	$M \leftarrow A$ and $M$	-	-	√	1+N
	AND A,I	$A \leftarrow A$ and $I$	-	-	√	1
	OR A,M	$A \leftarrow A$ or $M$	-	-	√	1
	OR M,A	$M \leftarrow A$ or $M$	-	-	√	1+N
	OR A,I	$A \leftarrow A$ or $I$	-	-	√	1
	XOR A,M	$A \leftarrow A$ xor $M$	-	-	√	1
	XOR M,A	$M \leftarrow A$ xor $M$	-	-	√	1+N
	XOR A,I	$A \leftarrow A$ xor $I$	-	-	√	1
	COM M	$A \leftarrow M$ (1's complement).	-	-	√	1
	COMM M	$M \leftarrow M$ (1's complement).	-	-	√	1
P R O C E S S	SWAP M	$A(b3 \sim b0, b7 \sim b4) \leftarrow M(b7 \sim b4, b3 \sim b0)$	-	-	-	1
	SWAPM M	$M(b3 \sim b0, b7 \sim b4) \leftarrow M(b7 \sim b4, b3 \sim b0)$	-	-	-	1+N
	RRC M	$A \leftarrow RRC M$	√	-	-	1
	RRCM M	$M \leftarrow RRC M$	√	-	-	1+N
	RLC M	$A \leftarrow RLC M$	√	-	-	1
	RLCM M	$M \leftarrow RLC M$	√	-	-	1+N
	CLR M	$M \leftarrow 0$	-	-	-	1
	BCLR M.b	$M.b \leftarrow 0$	-	-	-	1+N
	BSET M.b	$M.b \leftarrow 1$	-	-	-	1+N
	B0BCLR M.b	$M(bank\ 0).b \leftarrow 0$	-	-	-	1+N
	B0BSET M.b	$M(bank\ 0).b \leftarrow 1$	-	-	-	1+N
B R A N C H	CMPRS A,I	$ZF, C \leftarrow A - I$ , If $A = I$ , then skip next instruction	√	-	√	1 + S
	CMPRS A,M	$ZF, C \leftarrow A - M$ , If $A = M$ , then skip next instruction	√	-	√	1 + S
	INCS M	$A \leftarrow M + 1$ , If $A = 0$ , then skip next instruction	-	-	-	1 + S
	INCMS M	$M \leftarrow M + 1$ , If $M = 0$ , then skip next instruction	-	-	-	1+N+S
	INC M	$A \leftarrow M + 1$ .	-	-	√	1
	INCM M	$M \leftarrow M + 1$ .	-	-	√	1+N
	DECS M	$A \leftarrow M - 1$ , If $A = 0$ , then skip next instruction	-	-	-	1 + S
	DECMS M	$M \leftarrow M - 1$ , If $M = 0$ , then skip next instruction	-	-	-	1+N+S
	DEC M	$A \leftarrow M - 1$ .	-	-	√	1
	DECM M	$M \leftarrow M - 1$ .	-	-	√	1+N
	BTS0 M.b	If $M.b = 0$ , then skip next instruction	-	-	-	1 + S
	BTS1 M.b	If $M.b = 1$ , then skip next instruction	-	-	-	1 + S
	B0BTS0 M.b	If $M(bank\ 0).b = 0$ , then skip next instruction	-	-	-	1 + S
	B0BTS1 M.b	If $M(bank\ 0).b = 1$ , then skip next instruction	-	-	-	1 + S
	TS0M M	If $M = 0$ , $Z = 1$ . Else $Z = 0$ .	-	-	√	1
	JMP d	$PC15/14 \leftarrow RomPages1/0, PC13 \sim PC0 \leftarrow d$	-	-	-	2
	CALL d	$Stack \leftarrow PC15 \sim PC0, PC15/14 \leftarrow RomPages1/0, PC13 \sim PC0 \leftarrow d$	-	-	-	2

	CALLHL	Stack $\leftarrow$ PC15~PC0, PC15~PC8 $\leftarrow$ H register, PC7~PC0 $\leftarrow$ L register	-	-	-	2
	CALLYZ	Stack $\leftarrow$ PC15~PC0, PC15~PC8 $\leftarrow$ Y register, PC7~PC0 $\leftarrow$ Z register	-	-	-	2
M	RET	PC $\leftarrow$ Stack	-	-	-	2
I	RETI	PC $\leftarrow$ Stack, and to enable global interrupt	-	-	-	2
S	RETLW I	PC $\leftarrow$ Stack, and load I to ACC.	-	-	-	2
C	NOP	No operation	-	-	-	1

**Note:** 1. "M" is system register or RAM. If "M" is system registers then "N" = 0, otherwise "N" = 1.  
 2. If branch condition is true then "S = 1", otherwise "S = 0".

# 17 ELECTRICAL CHARACTERISTIC

## 17.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd).....	- 0.3V ~ 3.6V
Input in voltage (Vin).....	Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr).....	0°C ~ + 50°C
Storage ambient temperature (Tstor) .....	–40°C ~ + 125°C

## 17.2 ELECTRICAL CHARACTERISTIC

### SN8F29E49 DC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 3.0V, Fosc = 8MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	VDD	Normal mode		2.4	3	3.6	V
RAM Data Retention voltage	VDR			1.4	-	-	V
VDD rise rate	VPOR	VDD rise rate to ensure power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		VSS	-	0.3VDD	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
I/O port pull-up resistor	RUP	VIN = VSS, VDD = 3V		100	200	300	KΩ
I/O port input leakage current	ILEKG	Pull-up resistor disable, VIN = VDD		-	-	1	uA
I/O output source current	IOH	VOP = VDD - 0.5V		7	10		mA
sink current	IOL	VOP = VSS + 0.5V		7	10		mA
INTN trigger pulse width	TINT0	INT0 ~ INT1 interrupt request pulse width		2/FCPU	-	-	Cycle
Supply Current	Idd1	Normal Mode	Vdd=3V, Low-Power Mode @ Fcpu run 2-MIPS	-	0.8	1.5	mA
	Idd2		Vdd=3V, Analog (AD/DA/OP) @ Fcpu run 2-MIPS	-	2.2	3	mA
	Idd3	Slow Mode	Vdd=3V, Analog (AD/DA/OP/IHRC) C- LCD	-	1.8	2.5	mA
	Idd4		Vdd=3V, ILRC,IHRC off, C-LCD on	-	150	300	uA
	Idd6		Vdd=3V, ILRC	-	130	260	uA
	Idd7	Green Mode	Vdd=3V, ILRC off, 32K_X'tal on C-Type LCD On (code option:IHRC_RTC)	-	28	40	uA
	Idd8		Vdd=3V , ILRC off, 32K_X'tal on (code option:IHRC_RTC)	-	5	10	uA
	Idd9		Vdd=3V , ILRC on (code option:IHRC)	-	125	150	uA
	Idd10	Sleep Mode	Vdd= 3V	-	3.3	4.5	uA
Internal High Oscillator Freq.	F <sub>IHRC</sub>	Internal High RC (IHRC), 0~50C.		7.84	8	8.16	MHz
LVD 15 detect level	V <sub>LVD15</sub>	POR detect level.		1.38	1.45	1.52	V
LVD 16 detect level	V <sub>LVD16</sub>	Core domain.		1.60	1.65	1.72	V
LDO18	V <sub>LDO18</sub>	Core Power regulator output voltage		1.8V	1.85	1.92	V
		Low power mode output Voltage.		1.65	1.7	1.75	V
ADC							
Operation voltage	V <sub>oper</sub>	Power source from AVDDR		2.35	2.4	2.45	V
Operating current	I <sub>DD_ADC</sub>	Run mode @ 2.35V		-	350	-	uA
Power down current	I <sub>PDN</sub>	Stop mode @ 2.35V		-	0.1	-	uA

Conversion rate (Word Rate, WR)	$F_{WR}$	ADC Clock=250KHz, OSR=32768	-	7.6	-	Sps
		ADC Clock=333KHz, OSR=64	-	5.2	-	KSps
Reference Voltage Input absolutely Voltage	$V_{R+}$	GR=1, R+, R- Input absolutely Voltage	0.3	-	1.4	V
		GR=0, R+, R- Input absolutely Voltage	0.3	-	1.7	V
Reference Voltage Range	$V_{ref}$	ADC Reference voltage range. (R+) - (R-)	0.3	-	1.2	V
ADC Input signal absolutely Voltage	$V_{AIN}$	Input Buffer on (GX=1), AVDDR=2.35V	0.4	-	1.4	V
		Input Buffer off (GX=0), AVDDR=2.35V	0	-	1.7	V
Integral non-linearity	INL	ADC Gain x 8, ADC Input Range $\pm 0.9 \times V_{ref}$	-	-	0.01	%FS R
No missing code	NMC	ADC range $\pm 0.9 \times V_{ref}$	-	18	-	bit
ADC Noise free bits	NFB	Gain=1, Vref:0.8V, WR: 1KHz, Input-short Buffer Off. (GR=GX=0)	-	14.2	-	bit
		Gain=128, Vref: 0.8V, OSR: 1KHz, Input-short	-	11.3	-	bit
Unit Gain Buffer Current	$I_{GX}$	ADC signal input buffer operation current.	-	80	100	$\mu A$
	$I_{GR}$	ADC reference input buffer operation current.	-	80	100	$\mu A$
ADC Settling time		Ready to start convert after set ADENB = "1"	-	3's	-	ADC data
Unit Gain Buffer Input/output Range	$V_{GX}$	Absolutely voltage range. (AVDDR 2.4V)	0.4	-	1.4	V
	$V_{GR}$	Min = 0 + 0.4V, Max = AVDDR-1V.	0.4	-	1.4	V
<b>PGIA</b>						
PGIA Current consumption	$I_{DD\_PGIA}$	Run mode @2.4V	-	140	280	$\mu A$
Power down current	$I_{PDN}$	VDD = 2.4V, PGIA x 200	-	-	0.1	$\mu A$
PGIA Gain Range	Gain	AI+, AI+ signal input range (AVDDR =2.4V)	190	200	220	Gain
PGIA Input range	$V_{opin}$	AI+, AI- signal input range. (AVDDR = 2.4V)	0.4	-	1.4	V
PGIA Output Range	$V_{opout}$	Signal output range. (AVDDR = 2.4V)	0.4	-	1.4	V
<b>ADC Regulator AVDDR / AVE / ACM</b>						
Operation voltage	$V_{oper}$	Power source from AVDD.	2.35	2.4	2.45	V
Analog common voltage	$V_{ACM}$	Output voltage	0.9	1	1.1	V
Regulator output voltage AVE+	$V_{AVE+}$	AVE+ = 2V, @25°C, VDD = 3V(metal option)	1.95	2	2.05	V
Regulator output voltage AVDDR	$V_{AVDDR}$	AVDDR=2.35V, @25°C, Vdd = 3V. (4-Bit Code-option Trim)	2.35	2.4	2.45	V
AVE+ output current capacity	$I_{VA+}$	AVE+ output current ability	-	-	5	mA
AVDDR output current capacity	$I_{VADR+}$	AVDDR output current ability	-	-	10	mA
Quiescent current	$I_{QI}$	ACM + AVDDR + AVE	-	130	150	$\mu A$
VACM driving capacity	$I_{SRC}$		-	-	10	$\mu A$
VACM sinking capacity	$I_{SNK}$		-	-	1	mA
<b>Digital to Analog Converter</b>						
Operation voltage	$V_{oper}$	Power source from AVDDR	2.35	2.4	2.45	V
DAREF internal input voltage	$V_{DRIN}$	DAREF from source internal 0.18V	-	0.18	-	V
		DAREF from source internal 0.35V	-	0.35	-	V
		DAREF from source internal 0.65V	-	0.65	-	V
		DAREF from source internal 1.20V (BG source)	-	1.20	-	V
		DAREF from external source	0.1	-	1.2	V
offset voltage			-	3	-	mV
Positive Supply Current	$I_{DAC}$	Supply current @ no loading	-	300	600	$\mu A$
Relative Accuracy	INL		-	-	$\pm 8$	LSB
Differential Nonlinearity	DNL	Monotonic	-	-	4	LSB
Output voltage range	$V_{DAO}$		0.03	-	$V_{REFH-0.01}$	V
DAC load current		Buffer stage output (Drive / Sink )	-1	-	1	mA



Maximum DAC load Capacitance			-	-	100	pF
Full-scale settling time			-	40	-	uS
DAC turn on time			-	15	-	uS
Operation Amplifier						
Operation voltage	V <sub>oper</sub>	Power source from AVDDR	2.35	2.4	2.45	V
Input offset voltage	V <sub>OS</sub>		-	±2	±3	mV
Input offset drift w/temp.			-	10	-	uV/°C
Operating Current	I <sub>oper</sub>	per OP-Amp	-	130	-	uA
Analog Input voltage range	V <sub>in</sub>	OP+ / OP-	0.05	-	AVDDR-0.1	V
Analog Output voltage range	V <sub>out</sub>	OPOUT	0.05	-	AVDDR-0.1	V
Output short circuit current	I <sub>OH</sub> /I <sub>OL</sub>	Unit Gain Buffer. AVDDR=2.35V	-	±6	-	mA
Power Supply Rejection Ratio	PSRR	V <sub>cm</sub> =V <sub>ss</sub>	-	80	-	dB
Common Mode Rejection Ratio	CMRR	V <sub>cm</sub> =-0.3V~1.7V. AVDDR=2.35V.	50	70	80	dB
Open loop voltage gain		V <sub>out</sub> =0.2V~1.7V. V <sub>cm</sub> =V <sub>ss</sub> .	90	100	-	dB
Phase margin	§ m	C <sub>L</sub> =50pF	-	60	-	degree
Gain Bandwidth	GB	R <sub>L</sub> =300KΩ, C <sub>L</sub> =50pF	-	1.4	-	MHz
Slew Rate	SR	10% to 90%	-	0.8	-	V/uS
High level output voltage	V <sub>OH</sub>		AVDD-0.4	-	AVDD	V
Low level output voltage	V <sub>OL</sub>		AVSS	-	0.4	V
Turn on time	T <sub>ON</sub>		-	10	20	uS
Turn off time	T <sub>OFF</sub>		-	-	1	uS
Analog Switch						
Operation voltage	V <sub>oper</sub>	Power source from AVDDR	2.35	2.4	2.45	V
SC0 CMOS On-State switch resistance	R <sub>on</sub>	OP <sub>out</sub> =0.1V~1.7V, OP <sub>-</sub> =0.1V~0.6V	-	380	-	Ω
SC1~SC4 NMOS On-State switch resistance	R <sub>on</sub>	OP <sub>out</sub> =0.1V~1.7V, OP <sub>-</sub> =0.1V~0.6V	-	30	-	Ω
SC5~SC7 NMOS On-State switch resistance	R <sub>on</sub>	OP <sub>-</sub> =0.1V~0.6V	-	10	-	Ω
SC8~SC9 NMOS On-State switch resistance	R <sub>on</sub>	OP <sub>-</sub> =0.1V~0.6V	-	70	-	Ω
SC10 NMOS On-State switch resistance	R <sub>on</sub>	OP <sub>-</sub> =0.1V~0.6V	-	20	-	Ω
SC11 CMOS On-State switch resistance	R <sub>on</sub>	OP <sub>out</sub> =0.1V~1.7V, OP <sub>-</sub> =0.1V~0.6V	-	420	-	Ω
SC12~ SC13 CMOS On-State switch resistance	R <sub>on</sub>	OP <sub>-</sub> =0.1V~0.6V	-	20	-	Ω
SC14 CMOS On-State switch resistance	R <sub>on</sub>	OP <sub>out</sub> =0.1V~1.7V, OP <sub>-</sub> =0.1V~0.6V	-	420	-	Ω
LCD Driver						
Operation voltage	V <sub>oper</sub>		2.4	3	3.6	V
C-Type LCD Operation Current	I <sub>CLCD</sub>		-	70	-	uA
External capacitor for LCD driver			-	0.1	-	uF
LCD leakage current			-	-	1	uA
LCD driver frequency			-	64	128	Hz
C-Type VLCD output Voltage	V <sub>LCD</sub>	VLCD set 3V	2.85	3.05	3.25	V
		Condition: VLCD=2.6V ~ 3.3V, 0.1V step	-0.2		+0.2	V
VLED Driver						
Operation voltage	V <sub>oper</sub>		2.5	3	3.6	V
VLED Output Current Capacity	I <sub>LED</sub>	VDD=3.0V, I <sub>LED</sub> =5mA	4.5	5	5.5	mA
		VDD=3.0V, I <sub>LED</sub> =10mA	9	10	11	mA

## LBT Driver

Internal Low-Battery detect voltage	V <sub>ILBT</sub>	Condition: VLBT=2.5V	2.35	2.5	2.65	V
		Condition: VLBT=2.2 ~ 2.9 V	-0.1		+0.1	V
External Low-Battery	V <sub>ELBT</sub>	Condition: VDD =2.2~3.6V, P10 input comparator	1.05	1.2	1.35	V

“\*” These parameters are for simulation data.

## FLASH MEMORY CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 1.98V, Fosc = 4MHz, Fcpu=1MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	Vdd1	Read mode	1.62		Vdd	V
		Erase/Program	1.62		Vdd	V
Endurance time	Ten1	Erase + Program, -10°C~85°C	20K	*100K	-	Cycle
	Ten2	Erase + Program, -40°C~-10°C	20K	*70K	-	Cycle
Page erase current	I <sub>er</sub>		-	-	-	mA
Program current	I <sub>pg</sub>	Vdd1=1.98V	-	3.5	-	mA
Page erase time	T <sub>er</sub>	Vdd = 1.98V, 1-page (128-word).	-	-	30	ms
Program time	T <sub>pg1</sub>	Vdd = 1.98V, ISP setup time.	-	-	380	us
	T <sub>pg2</sub>	Vdd = 1.98V, 1-word program.	-	-	30	us

“\*” These parameters are for design reference, not tested.

## FLASH MEMORY POWER MEASUREMENT

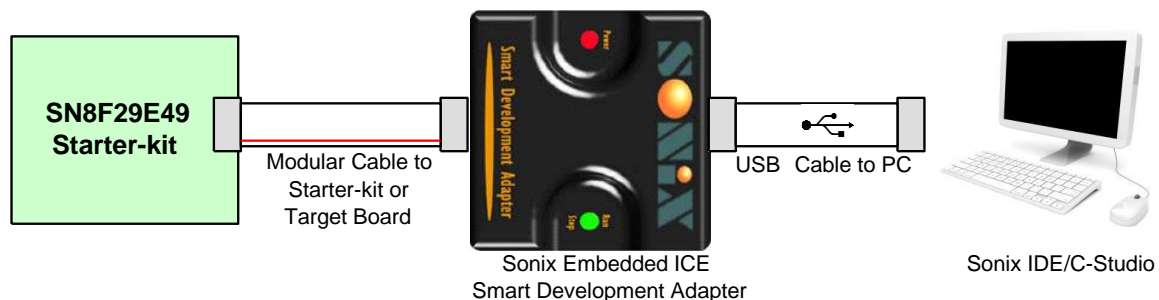
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Supply Voltage	Vdd2	Read mode	1.62		Vdd	V
		Erase/Program	1.62		Vdd	V
Page erase current measurement	I <sub>erm</sub>	Vdd2=1.82V, 0°C~70°C	-	2	-	mA
Program current measurement	I <sub>pgm</sub>	Vdd2=1.82V, 0°C~70°C	-	3.8	-	mA

# 18 DEVELOPMENT TOOL

## 18.1 OVERVIEW

SONiX provides an Embedded ICE emulator system to offer SN8F29E49 firmware development. The platform is an in-circuit debugger and controlled by SONiX M2IDE software on Microsoft Windows platform. The platform includes Smart Development Adapter, SN8F29E49 Starter-kit and M2IDE software to build a high-speed, low cost, powerful and multi-task development environment including emulator, debugger and programmer. To execute emulation is like run real chip because the emulator circuit integrated in SN8F29E49 to offer a real development environment.

### SN8F29E49 Embedded ICE Emulator System:



### SN8F29E49 Embedded ICE Emulator includes:

- Smart Development Adapter.
- USB cable to provide communications between the Smart Development Adapter and a PC.
- SN8F29E49 Starter-Kit.
- Modular cable to connect the Smart Development Adapter and SN8F29E49 Starter-Kit or target board.
- CD-ROM with M2IDE software (M2IDE V147).

### SN8F29E49 Embedded ICE Emulator Feature:

- Target's Operating Voltage: 2.4V~3.6V.
- Up to 6 hardware break points.
- System clock rate up to 4MHz (Fcpu=4mips).
- Oscillator supports internal high speed RC, internal low speed RC, external crystal/resonator.

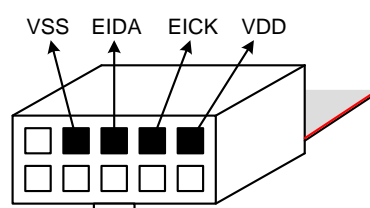
### SN8F29E49 Embedded ICE Emulator Limitation:

EIDA and EICK pins are shared with GPIO pins. In embedded ICE mode. We strongly recommend planning the two pins as simple function which can be verified without debugger platform.

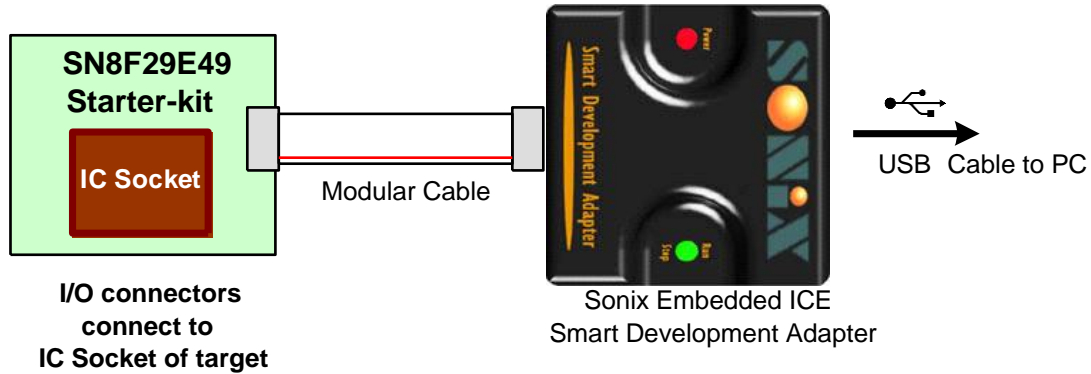
## 18.2 SMART DEVELOPMENT ADAPTER

Smart Development Adapter is a high speed emulator for Sonix Embedded ICE type flash MCU. It debugs and programs Sonix flash MCU and transfers MCU's system status, RAM data and system register between M2IDE and Sonix flash MCU through USB interface. The other terminal connected to SN8F29E49 Starter-kit or Target board is a 4-wire serial interface. In addition to debugger functions, the Smart Starter-Kit system also may be used as a programmer to load firmware from PC to MCU for engineering production, even mass production.

Smart Development Adapter communication with SN8F29E49 flash MCU is through a 4-wire bus. The pin definition of the Modular cable is as following:



- ☞ The modular cable can be inserted into SN8F29E49 Starter-Kit plugged into the target board or inserted into a matching socket at the target device on the target board.
- ☞ If the target board of application is designed and ready, the modular cable can be inserted into the target directly to replace SN8F29E49 Starter-Kit. Design the 4-wire interface connected with SN8F29E49 IC to build a real application environment. In the mode, set SN8F29E49 IC on the target is necessary, or the emulation would be error without MCU.



EIDA and EICK share with P1.6/P1.7 GPIO. In emulation mode, EIDA and EICK are Embedded ICE interface and not execute GPIO functions. The P1.6/P1.7 GPIO status still display on M2IDE window to simulate P1.6/P1.7 program execution.

## 18.3 OTP PROGRAMMING PIN TO TRANSITION BOARD MAPPING

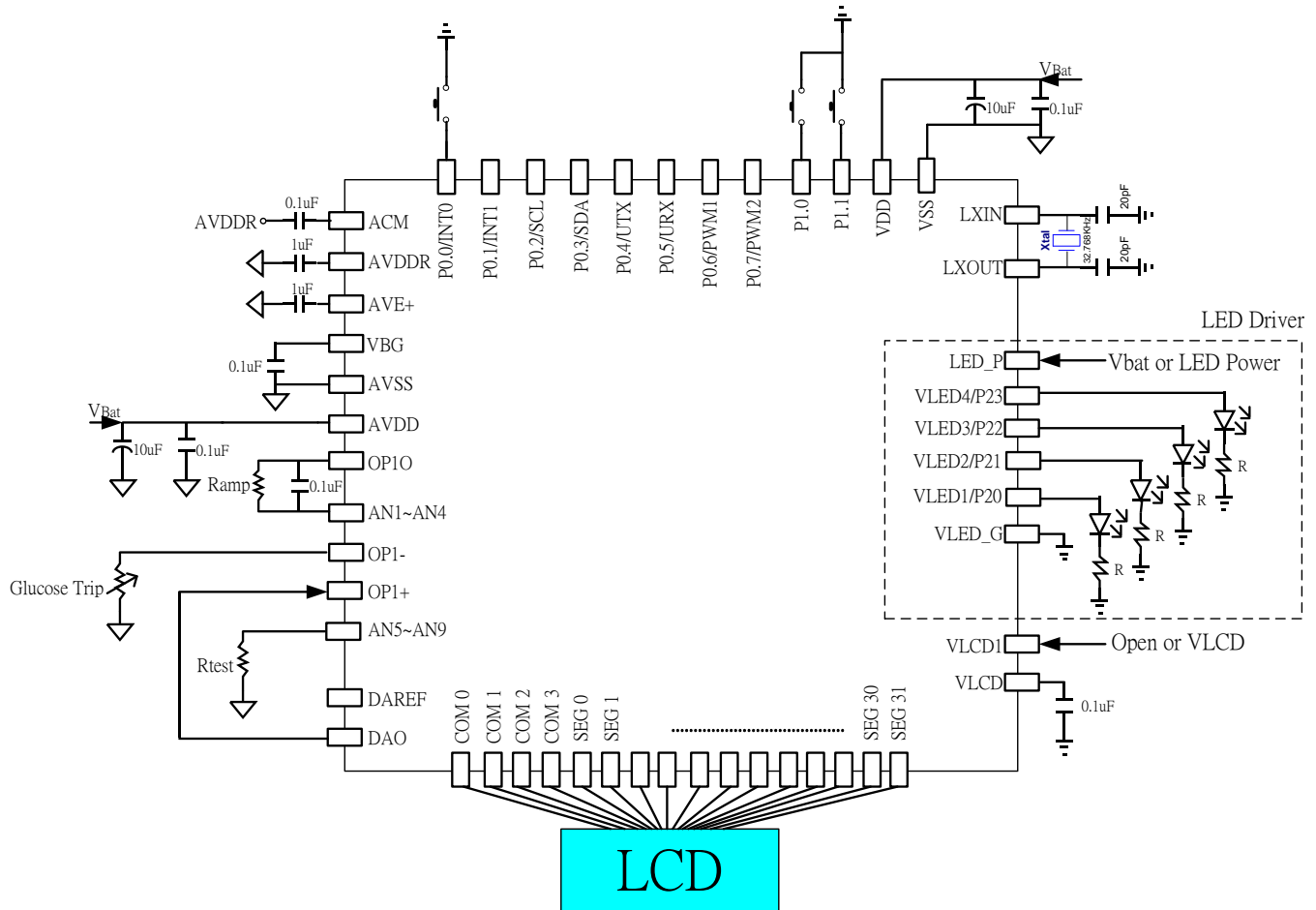
OTP Programming Pin of SN8F29E49				
MP PRO Writer		SN8F29E49		
JP3 of 40 PIN Adapter Board		Pin Assignment		
Number	Pin Name	Pad Name	LQFP100	LQFP80
PIN Number				
1	VDD	VDD	86	69
2	GND	VSS	40	32
3	CLK / PGCLK	P16	82	66
4	CE	-	-	-
5	PGM / OTPCLK	P17	81	65
6	OE / ShiftData	-		
7	D1	-	-	
8	D0	-	-	
9	D3	-	-	
10	D2	-	-	
11	D5	-	-	
12	D4	-	-	
13	D7	-	-	
14	D6	-	-	
15	VDD	-	-	
16	VPP	-	-	
17	HLS	-	-	
18	RST	-	-	
19	-	-	-	
20	ALSB/PDB	P50	80	64

### SN8F29E49 Package Type Programming with 20 PIN adapter board and with writer board:

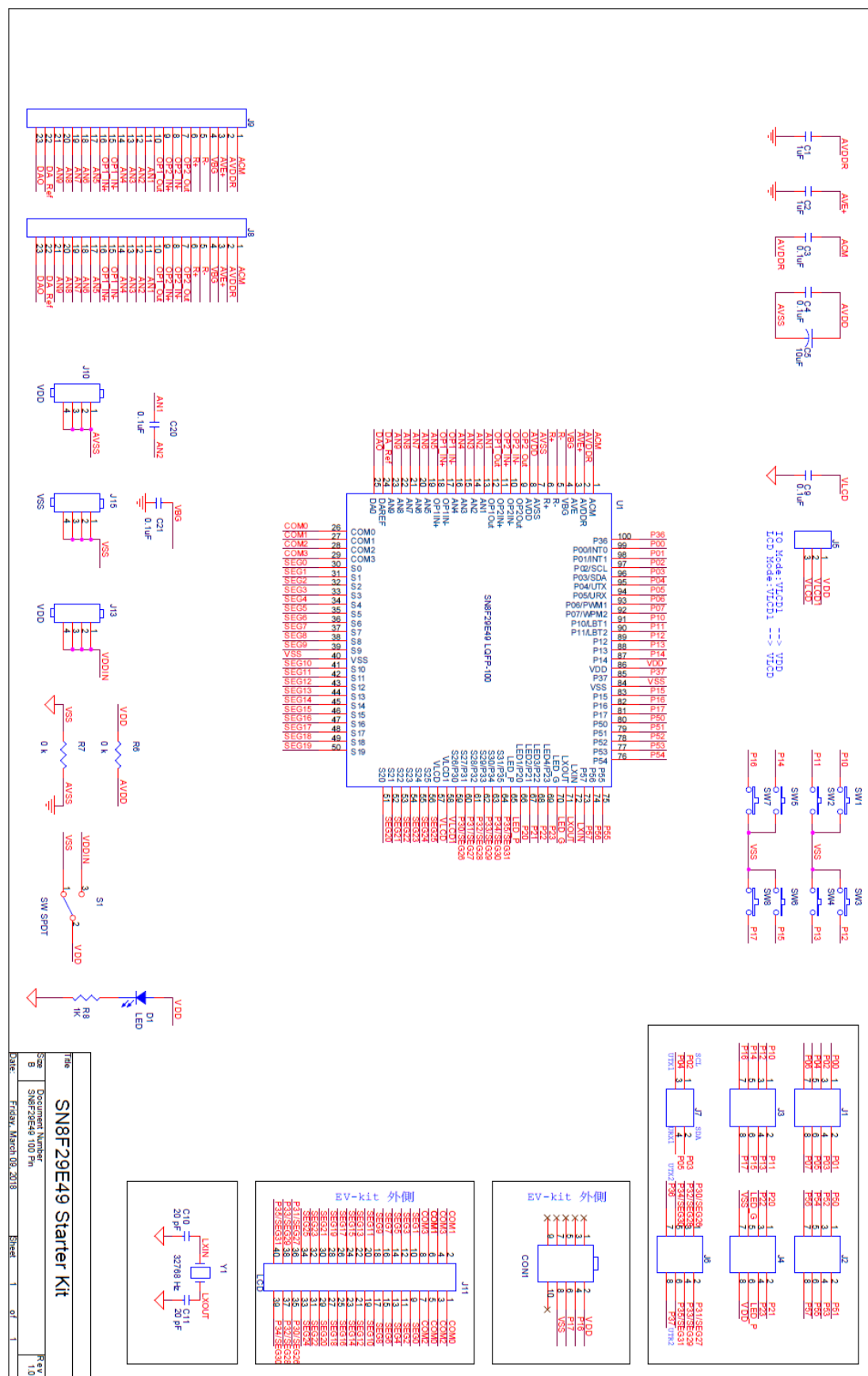
1. 20 PIN adapter board connect to MP PRO Writer.
2. Connect CON1 of SN8F29E49 Writer Board to the adapter board (JP3).
3. Program Kit is needed for package type of LQFP-100 IC.

# 19 Application circuit ,START-KIT

## 19.1 APPLICATION CIRCUIT



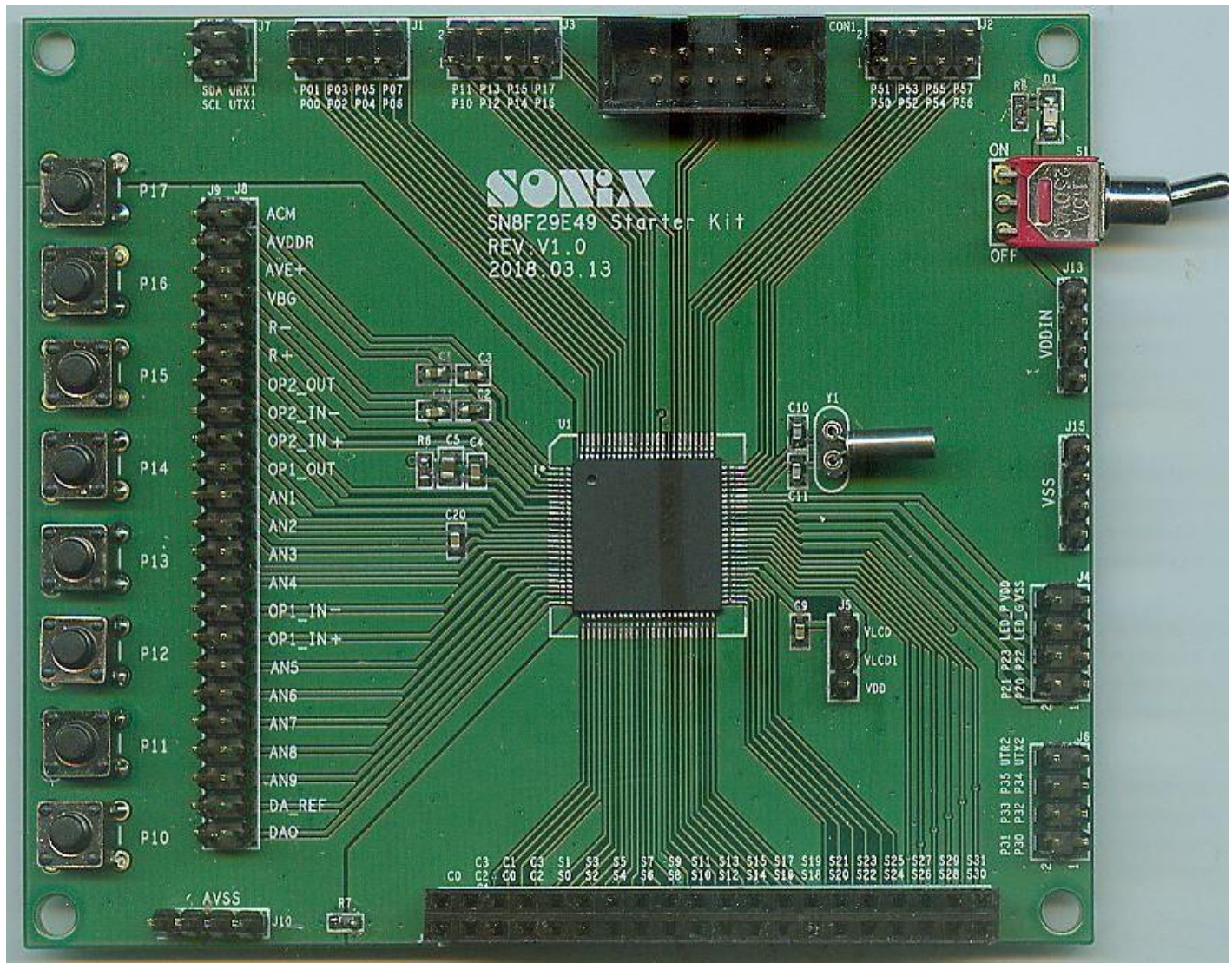
## 19.2 START KIT CIRCUIT





## 19.3 SN8F29E49 STARTER-KIT

SN8F29E49 Starter-kit is an easy-development platform. It includes SN8F29E49 real chip and I/O connectors to input signal or drive extra device of user's application. It is a simple platform to develop application as target board not ready. The starter-kit can be replaced by target board, because SN8F29E49 integrates embedded ICE in-circuit debugger circuitry. The schematic and outline of SN8F29E49 Starter-Kit is as following:



### EV-kit BOARD SETTING

1. J13: DC 3.0V power adapter.
2. S1: Target power switch.
3. U1: SN8F29E49 real chip.
4. D1: Power LED.
5. J1/J2/J3/J4/J6 : I/O connector.
6. Y1, C10, C11: External crystal/resonator oscillator components.
7. J5:P3 LCD SEG/IO switch. P3 is LCD SEG mode when LCD1 pin short LCD pin,
8. CON1: SDA connector.
9. J11 COM/SEG connect pin.
10. J8/J9 analog pin out.



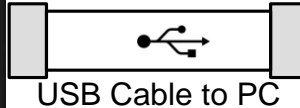
## Emulator/debugger InSTALLATION

Install the M2IDE Software (V147).

Connect Smart Development Adapter with PC plugging in USB cable.



Sonix Embedded ICE  
Smart Development Adapter

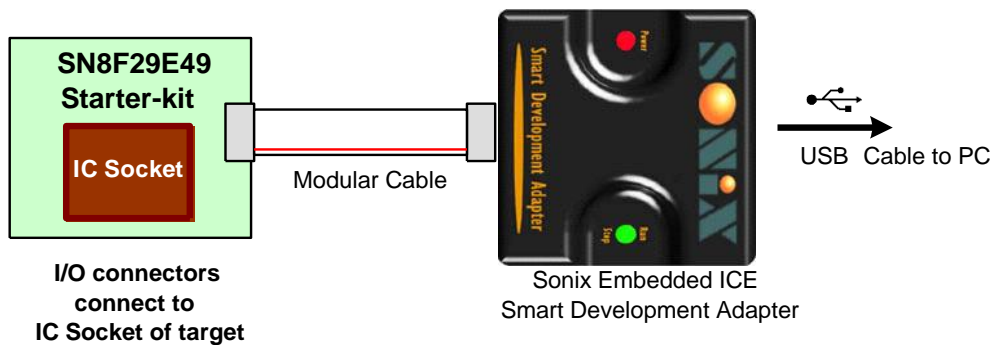


## USB Cable to PC

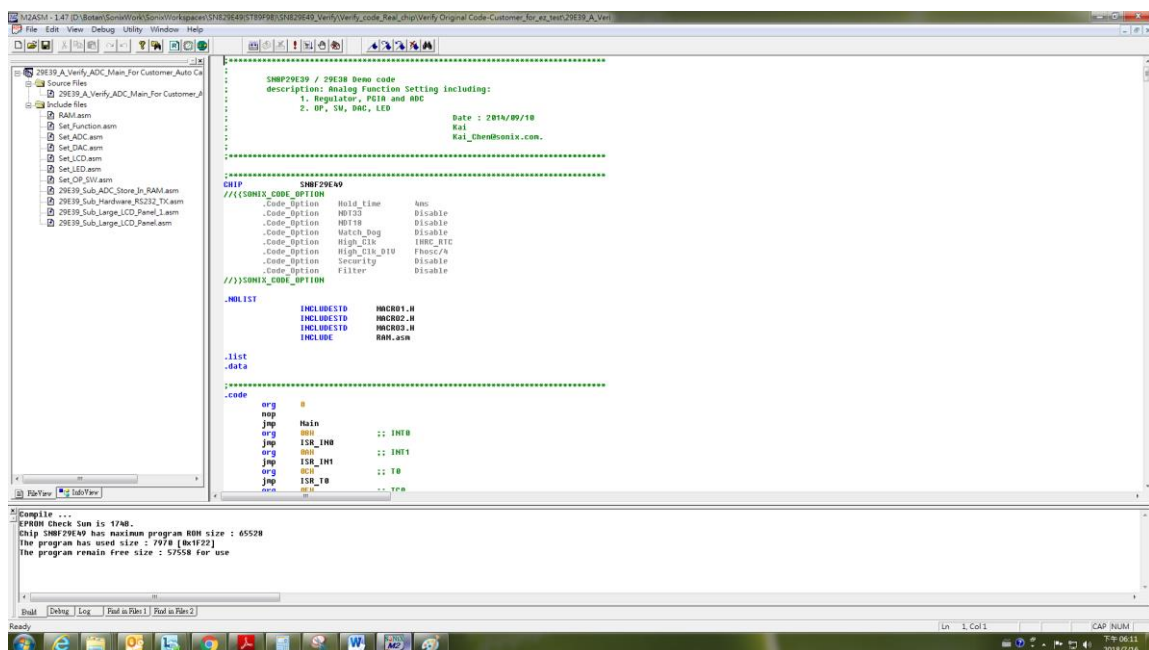


## Sonix IDE/C-Studio

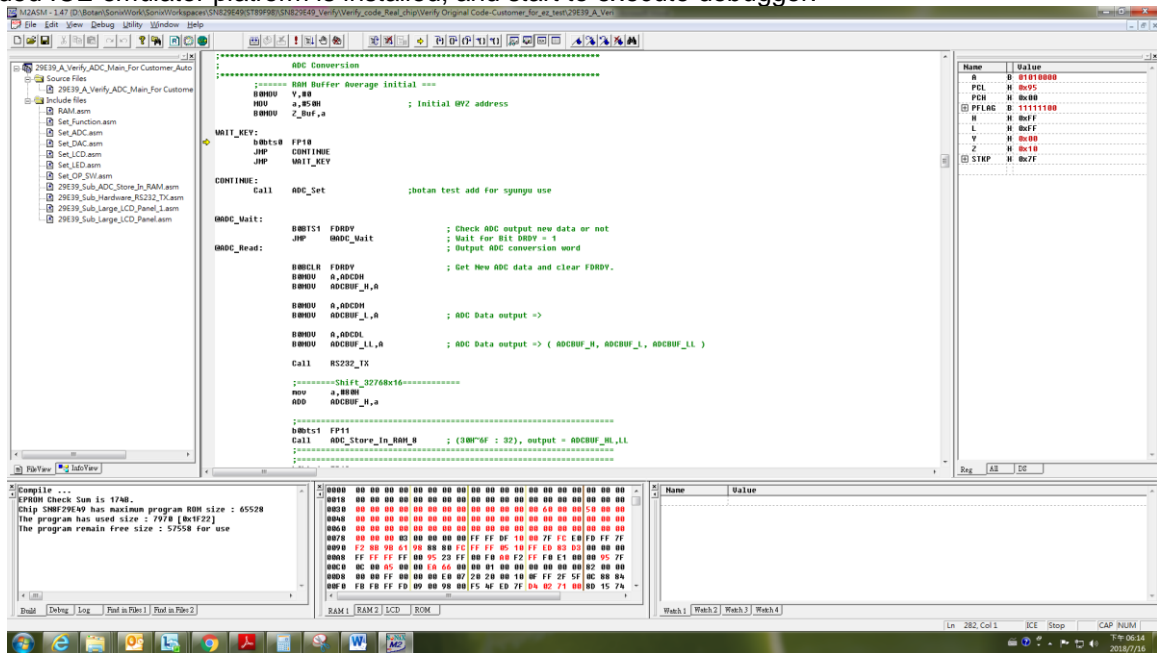
Attach the modular cable between Smart Development Adapter and SN8F29E49 Starter-kit or target.



Connect the power supplier to SN8F29E49 Starter-kit or target, and turn off the power. Open M2IDE software and load firmware program (A project or a “.ASM” file).

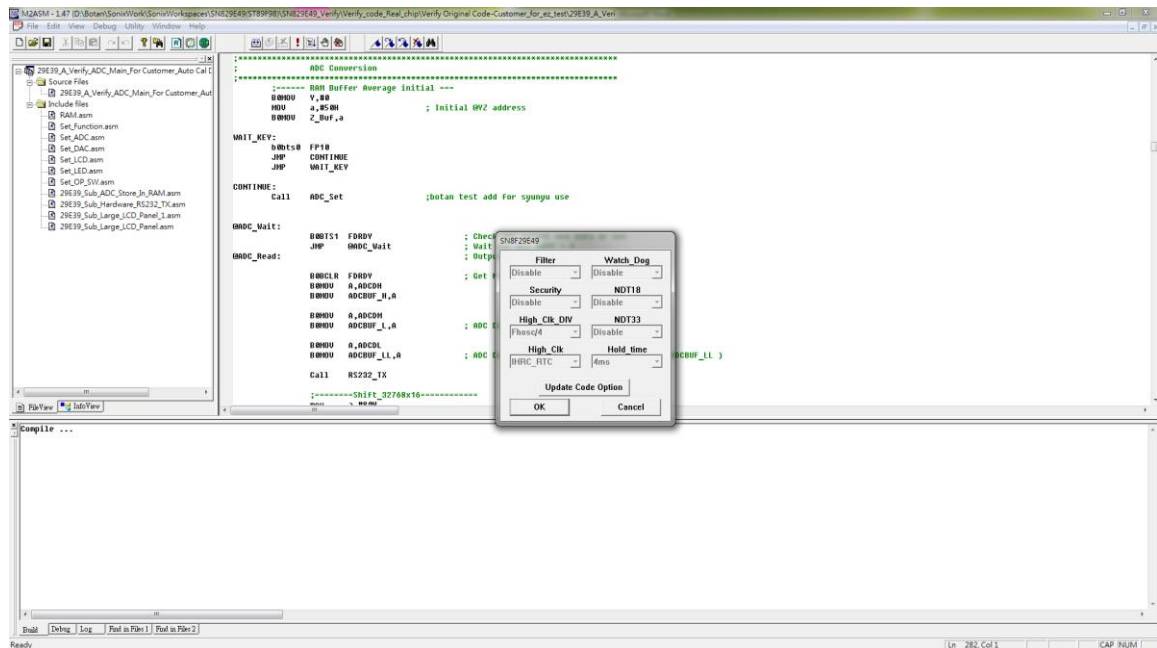


Turn on the power switch of SN8F29E49 Starter-kit or target.  
Embedded ICE emulator platform is installed, and start to execute debugger.



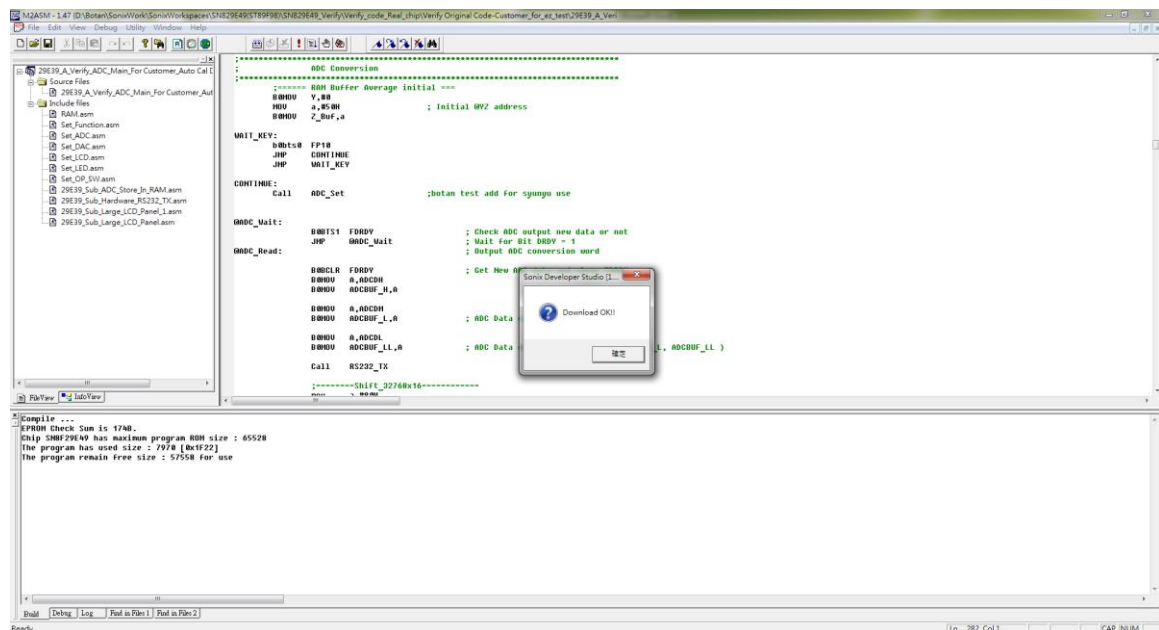
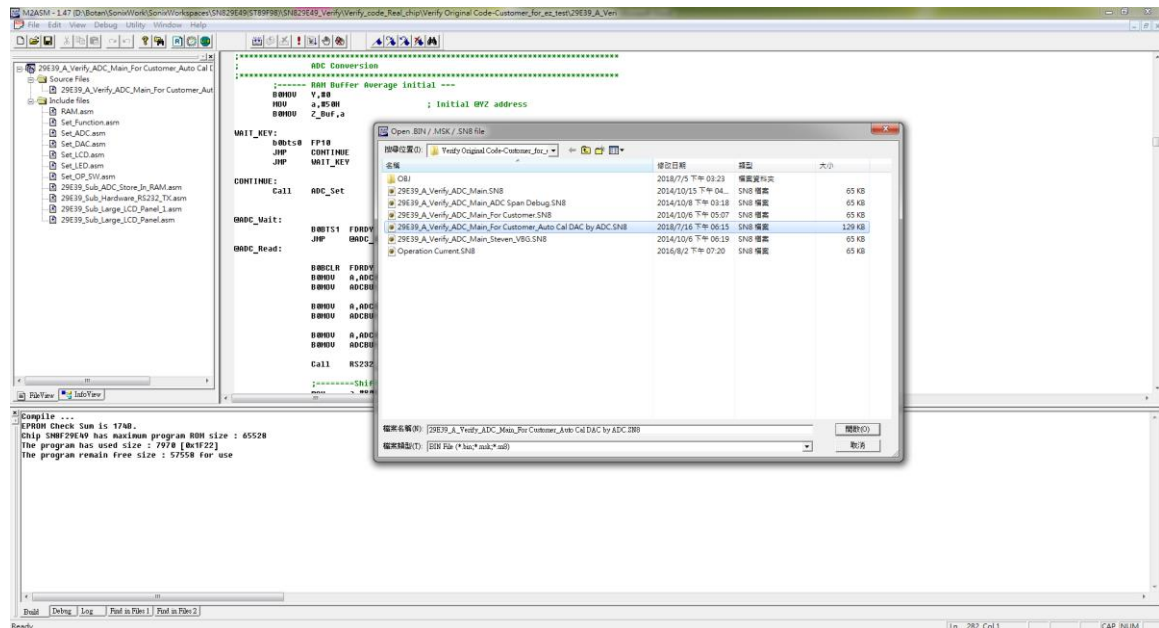
## 19.4 PROGRAMMER INSTALLATION

Setup emulator/debugger environment first.  
Compile the firmware program and generate a ".SN8" file.



Execute download (F8) function of M2IDE.

Open a ".SN8" file and press "Enter" to download firmware to SN8F29E49 Starter-kit or target.



Turn off the power of SN8F29E49 Starter-kit or target.

Disconnect SN8F29E49 Starter-kit or target from Smart Development Adapter.

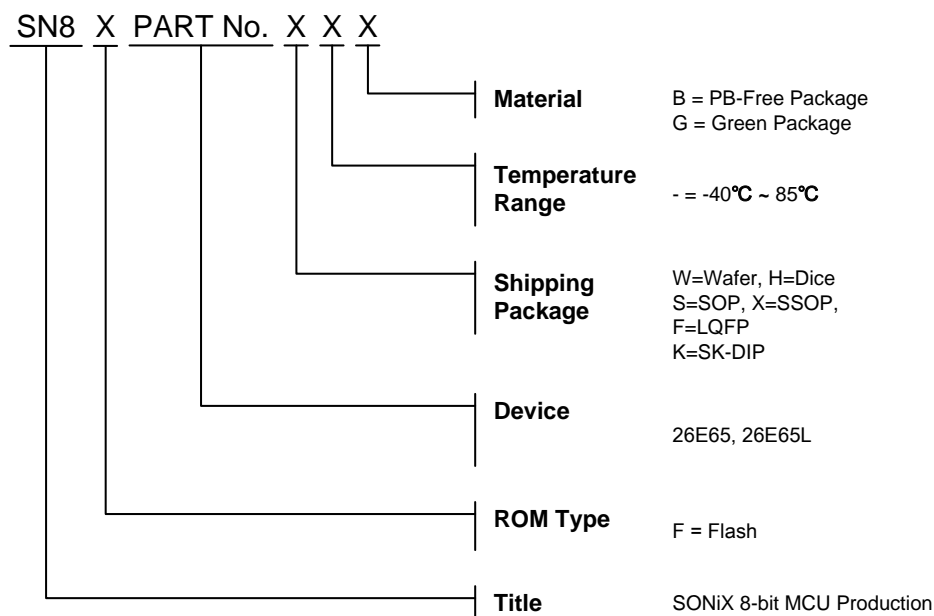
Turn on the power of SN8F29E49 Starter-kit or target, and MCU works independently.

# 20 Marking Definition

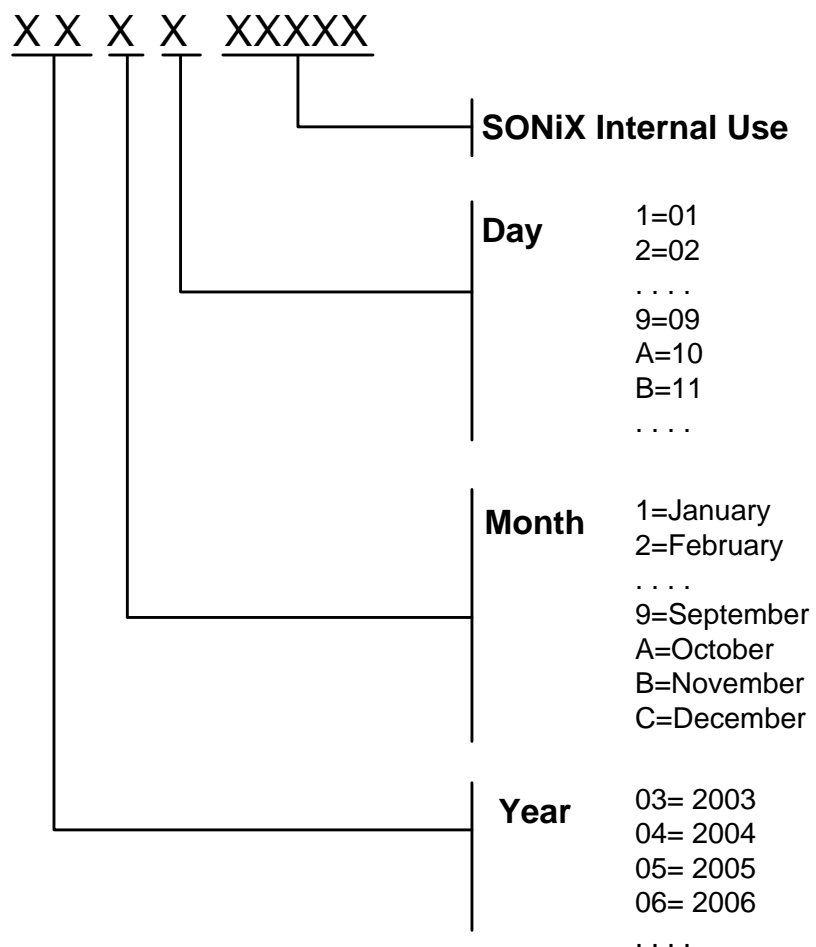
## 20.1 INTRODUCTION

There are many different types in Sonix 8-bit MCU production line. This note listed the production definition of all 8-bit MCU for order or obtain information. This definition is only for Blank Flash ROM MCU.

## 20.2 MARKING INDETIFICATION SYSTEM

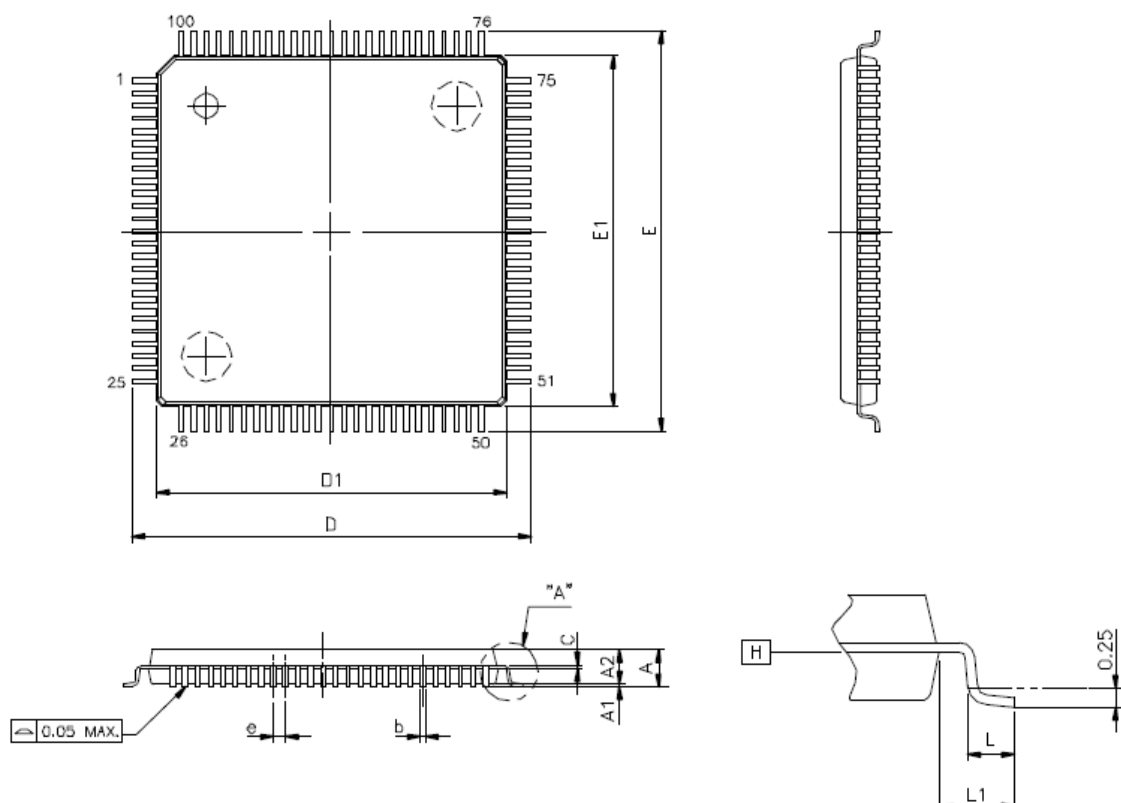


## 20.3 DATECODE SYSTEM



# 21 PACKAGE INFORMATION

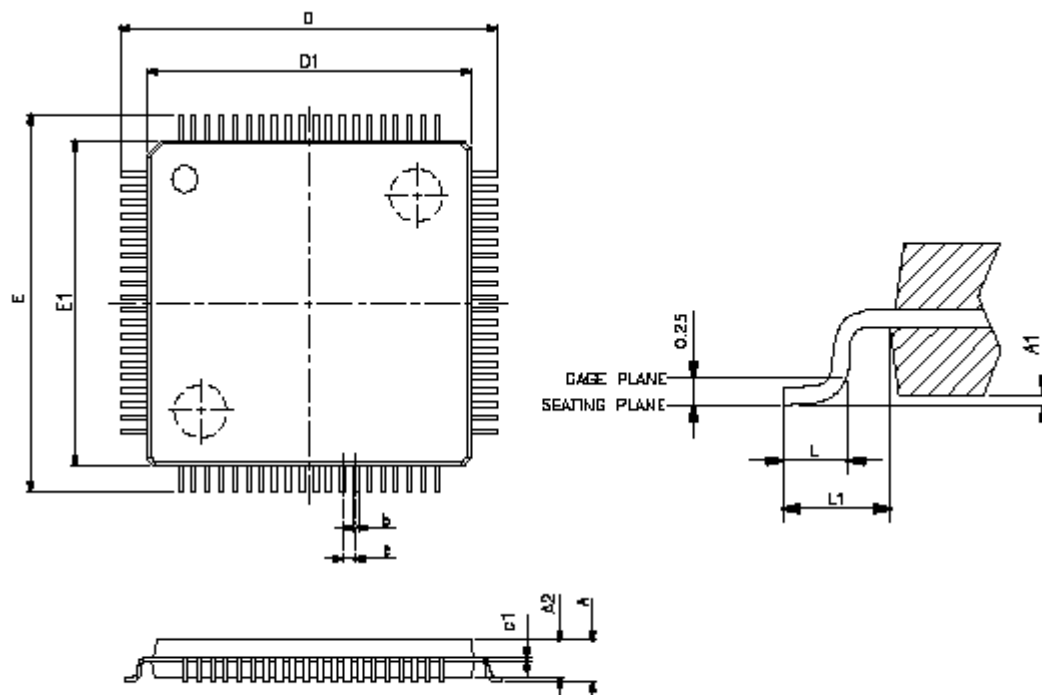
## 21.1 LQFP 100 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.20	0.27
c	0.09	0.127	0.20
D	16.00 BSC		
D1	14.00 BSC		
E	16.00 BSC		
E1	14.00 BSC		
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		

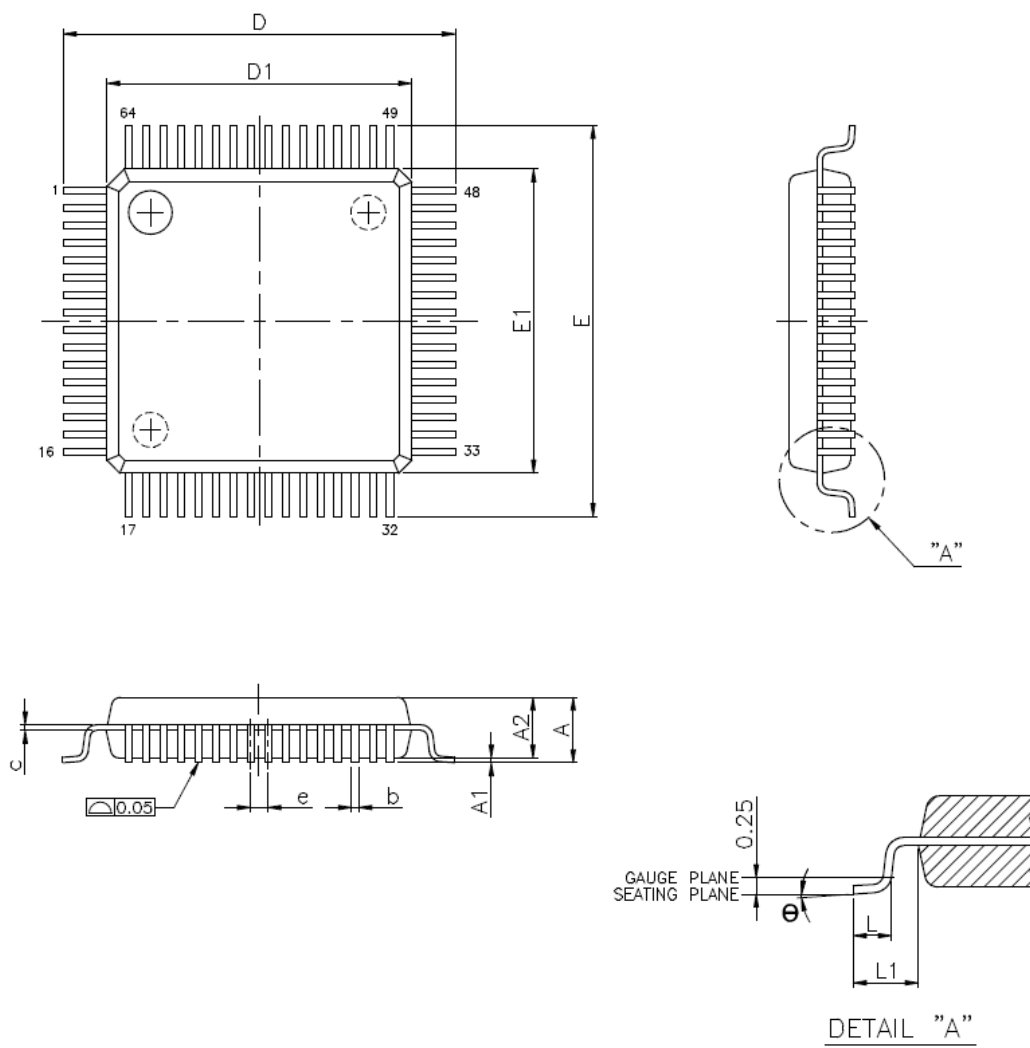
## 21.2 LQFP 80 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
a1	0.09	0.16
D	12 BSC	
D1	10 BSC	
E	12 BSC	
E1	10 BSC	
e	0.4 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

## 21.3 LQFP 64 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
b	0.13	0.18	0.23
c	0.09	—	0.20
D	9.00 BSC		
D1	7.00 BSC		
e	0.40 BSC		
E	9.00 BSC		
E1	7.00 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
$\theta$	0°	3.5°	7°



SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONIX products are not designed, intended, or authorized for use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

**Main Office:**

Address: 10F-1, NO.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan R.O.C.

Tel: 886-3-560 0888

Fax: 886-3-560 0889

**Taipei Office:**

Address: 15F-2, NO.171, Song Ted Road, Taipei, Taiwan R.O.C.

Tel: 886-2-2759 1980

Fax: 886-2-2759 8180

**Hong Kong Office:**

Unit 1519, Chevalier Commercial Centre, NO.8 Wang Hoi Road, Kowloon Bay, Hong Kong.

Tel: 852-2723-8086

Fax: 852-2723-9179

**Technical Support by Email:**

Sn8fae@sonix.com.tw