SONiX Technology Co., Ltd.

SN8F5840 Series Datasheet

(Rev. 1.9)

8051-based Microcontroller

SN8F5849
SN8F5848
SN8F5847
SN8F5844

1 Device Overview

1.1 Features

Enhanced 8051 microcontroller with reduced instruction cycle time (up to 12 times 80C51)
- Up to 32 MHz flexible CPU frequency
- Internal 32 MHz Clock Generator (IHRC)
- Real-time clock with 32.768 kHz crystal
- Fcpu : 16MIPs ~ 0.25MIPs

Memory:
- 32 KB non-volatile flash memory (IROM) with in-system program support
- 256 bytes internal RAM (IRAM)
- 3 KB external RAM (XRAM)
- 1.5KB ISP Data Flash Memory (ISP DATA ROM)

I/O Pin: 10-Pins pure I/O and 38-Pins I/O share with LCD function.

Interrupt: 11 interrupt sources with priority levels control and unique interrupt vectors
- 8 internal interrupts: T0, TC0, TC1, TC2, I2C, SAR ADC, UART-TX/RX, SPI
- 2 set serial Channel Selection.
- 2 external interrupts: INT0, INT1

Hardware Multiplication/Division Unit and 2 set of DPTR

Timer System :
- 1 set 8-Bit base Timer for RTC.(T0)
- 3 set 16-Bit Timer/Counter with Auto reload,
- 11 set PWM, Buzzer output.(TC0~TC2)
- One Buzzer Pin (P12): 1KHz / 2KHz / 4KHz / 8KHz / 16KHz / 32KHz

SPI, UART, I2C interface with SMBus Support

One Comparator for Low Battery Detect 2.2V~2.9V (0.1V/step)

Power Supply: 2.0~3.6V (AVDD/DVDD)

Regulator:
- Input voltage: 2.0V ~ 3.6V
- Regulator AVREFH output for Analog Circuit (SAR ADC, DAC, VBias)
- AVREFH selectable: 2.0V, 2.5V, 3.0V

Two 12-Bit DAC:
- Reference: Internal AVREFH*2/8& AVREFH*4/8 & AVREFH and External Vref.

12-Bit SAR ADC:
- Input channels: AI1~AI10
- Built-in one Battery Measurement: 1/2*AVDD, 1/4*AVDD
- Internal Vref: 2V/2.5V/3V/AVREFH/Ext Vref
- ADC conversion Rate (31.25k ~ 250kHz)

OPA: 2 set Operational Amplifier
- Rail to Rail OPA1, OPA2 with PGA embedded
- OPA2 Gain: 1x, 3x, 7x, 15x, 31x, 63x, 95x

LCD Driver: C-Type LCD driver up to 192 dot
- 4 com or 6 com with 1/3 bias
- 4x34 or 6x32 dots, LCD pin share with I/O
- Adjustable VLCD output 2.7V ~ 3.45V

On-Chip Debug Support:
- Single-wire debug interface 5 hardware breakpoints
  Unlimited software breakpoints ROM data security/protection
- Watchdog and programmable external reset

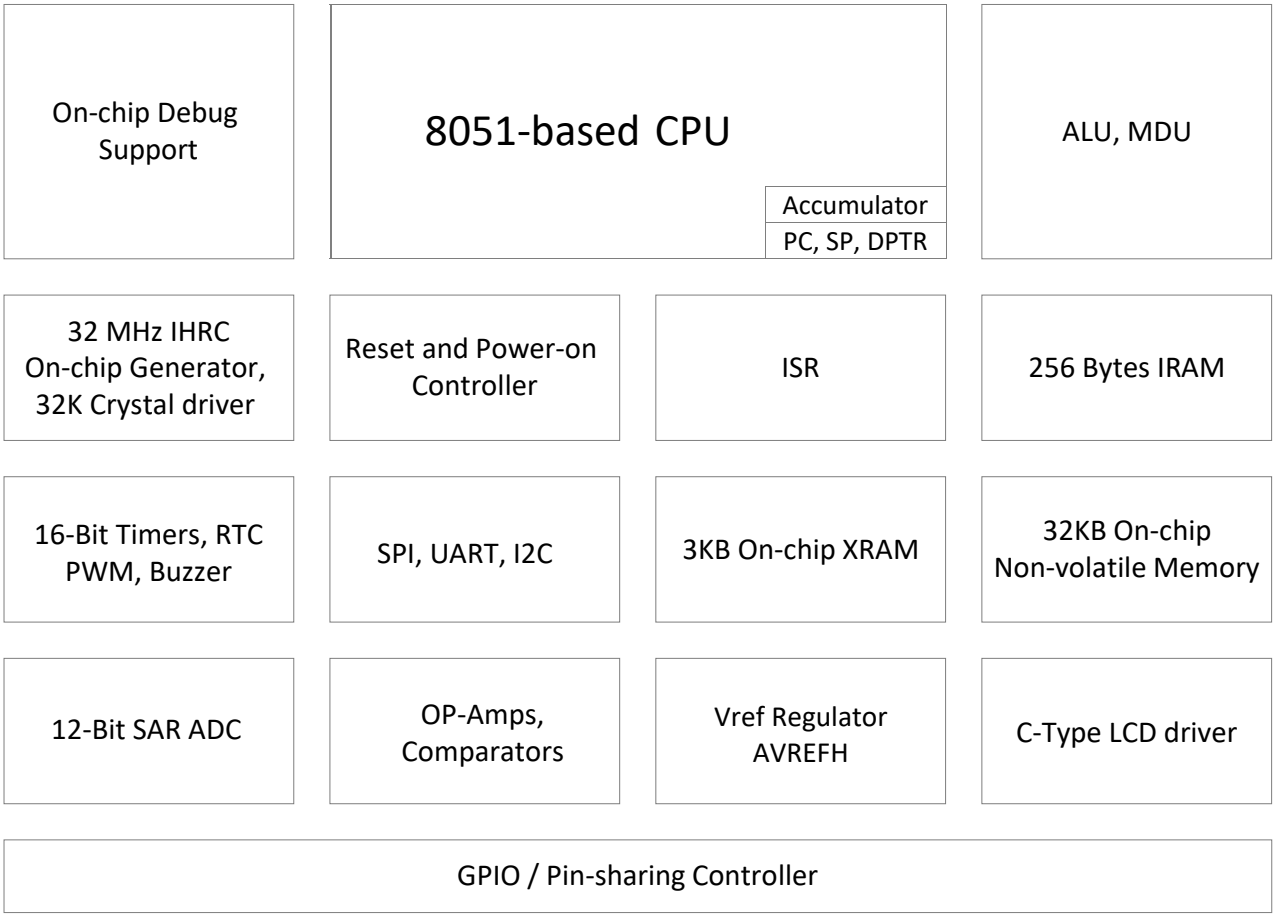
1.2 Applications

- Pulse Oximeter
- Glucose Meter
- Other measurement application.
- Blood Pressure Meter
- Weight scale

1.3 Features Selection Table

	I/O	LCD	RTC	16-Bit Timer / PWM	I2C	SPI	UART	SAR ADC	DAC	OPA	CMP	Buzzer	Int. INT	Ext. INT	Package Types
SN8F5849	48	4x34, 6x32	V	3/11	V	V	V	10-Ch	2	2	1	1	8	2	LQFP80
SN8F5848	45	4x31, 6x29	V	3/11	V	V	V	10-Ch	2	2	1	1	8	2	LQFP64
SN8F5847	31	4x17, 6x15	V	3/7	V	V	V	9-Ch	2	2	1	1	8	2	LQFP48
SN8F5844	17	-	V	3/3	V	-	V	7-Ch	2	2	1	1	8	2	TSSOP28

1.4 Block Diagram



2 Table of Contents

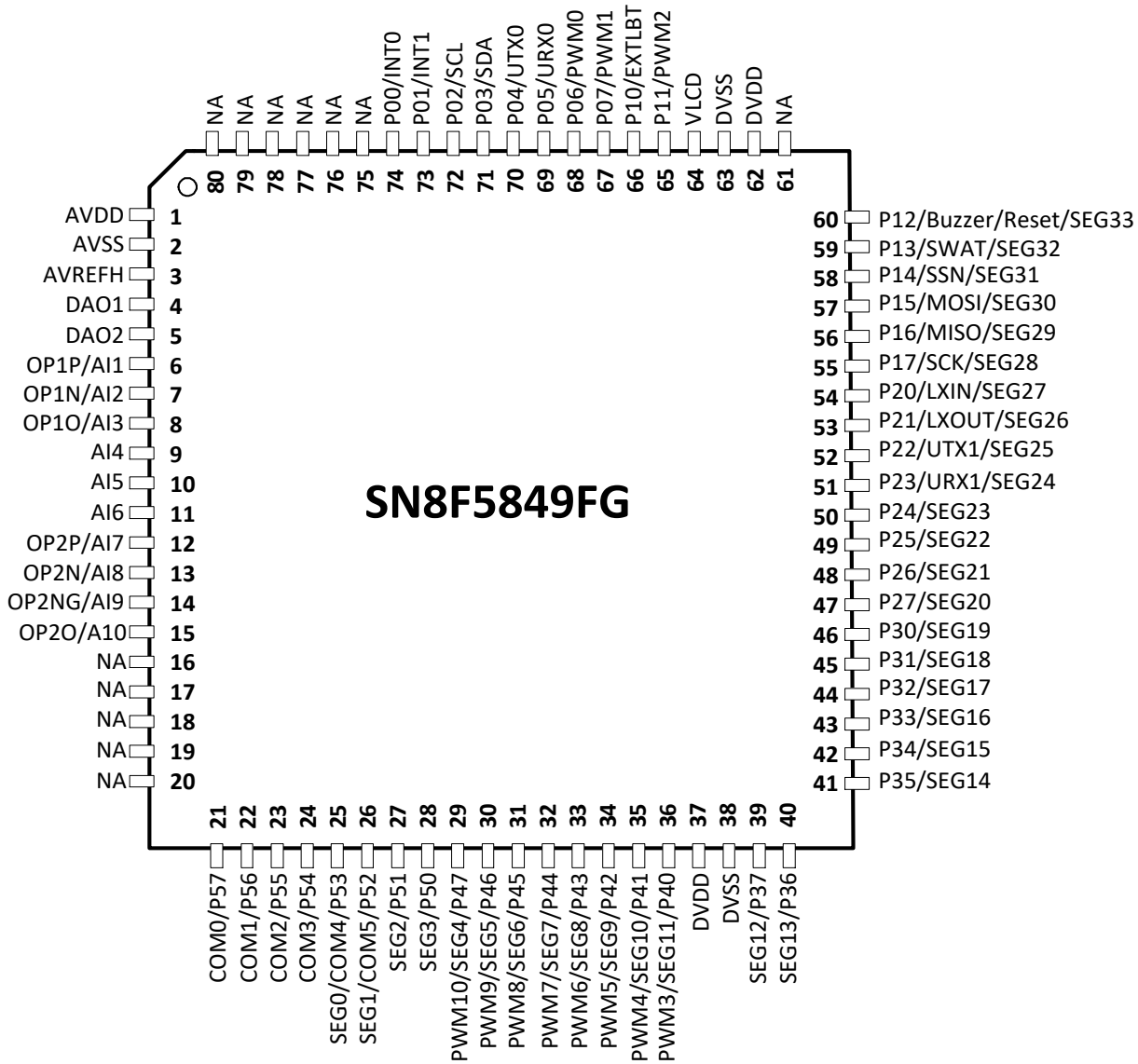
3 Revision History

Revision	Date	Description
1.0	Aug 2020	1. First Revision
1.1	Aug 2020	1.Update 7.1 Configuration of Reset and Power-on Controller (P82~P83) 2.Remove BP application circuit 3.Update 9.3 Interrupt Register- IRCON2 (P41&P44) 4.Modify 17.9 Sample Code (P94&P95) 5.Modify TC0M Register -0xC1 (P64) 6.Modify LBTM Register-0xD7 (P101) 7.Modify IEN2 Register-0X9A(P42) 8.Modify interrupt table (P38) 9.Modify Special Function Register Memory Map (P22) 10.Modify Features descriptions (P2) 11.Modify Operational amplifier 2 & internal gain function Sample code(P106) 12.Modify 17.9 Sample Code (P95) 13.Modify LCD Control Registers (P84~P87) 14.Modify Power Management(P35) 15.Modify Features Selection Table (P3) 16.Modify 1.1Features(P2) 17.Modify 22.5Sample Code(P122) 18.Modify 21.6 Sample Code(P116) 19.Modify VREFH Register -0xB3 (P98) 20.Modify SEC & MIN Register(P75)
1.2	Aug 2020	1.Modify TC2M Register (0xA1) (P66) 2.Modify OPA2 Gain: 1x, 3x, 7x, 15x, 31x, 63x, 95x (P2) 3.Modify Debug Interface Hardware (P154) 4.Modify Programming Pin Mapping (P160~P161)
1.3	Dec 2020	1.Modify VLCD output 2.7V ~ 3.45V (P84) 2.Modify ISP DATA Program Memory (ISP DATA ROM) (P16/P18) 3.Modify 24 In-System Program (P135~P139) 4.Modify 8.3 Power Management (P35~P36) 5.Modify 27.9 LCD Characteristic(P147)
1.4	Jul 2021	1.Modify VBIASEN Control Bit (P106) 2.Remove OPM3 Register please set “01” for all application (P105) 3.Modify 4.3 N8F5847F (P11)
1.5	Sep 2021	1.Add 4.4 SN8F5844T-TSSOP28 (P12) 2.Update 1.3 Features Selection Table (P3) 3.Add 32.5 TSSOP28 PIN Packaging Information (P168) 4.Add 25.1 CRC-CCITT Sample Code(P143)
1.6	Nov 2021	1.Modify Normal mode supply current at Fcpu set 16MHz. (P146) 2.Remove LVD16 and LVD12 detect voltage at -40℃ to 85℃. (P146) 3.Modify ESD and LU min and the typical result. (P146) 4.Remove SAR ADC Input offset voltage trimmed result. (P147) 5.Modify OP Buffer Mode Switch Ron from 50 to 350 ohm at typical. (P148) 6.Modify SW3~SW5 Ron from 20 to 40 ohm at typical. (P148) 7.Modify DAC Differential Nonlinearity(DNL) from ±2 to 4 at typical. (P148)
1.7	Sep 2022	1.Modify MP5 Writer Programming Pin Mapping of SN8F5847.(P161) 2.Modify SN-Link ISP Programming Pin Mapping of SN8F5847.(P162) 3.Modify 1.3Features Selection Table(P3) 4.Modify 31.4 MP5 Writer Programming Pin Mapping (P161) 5.Modify 31.6 SN-Link ISP Programming Pin Mapping(P162)
1.8	May 2023	1.Modify 27.2 System Operation Characteristics (P146)

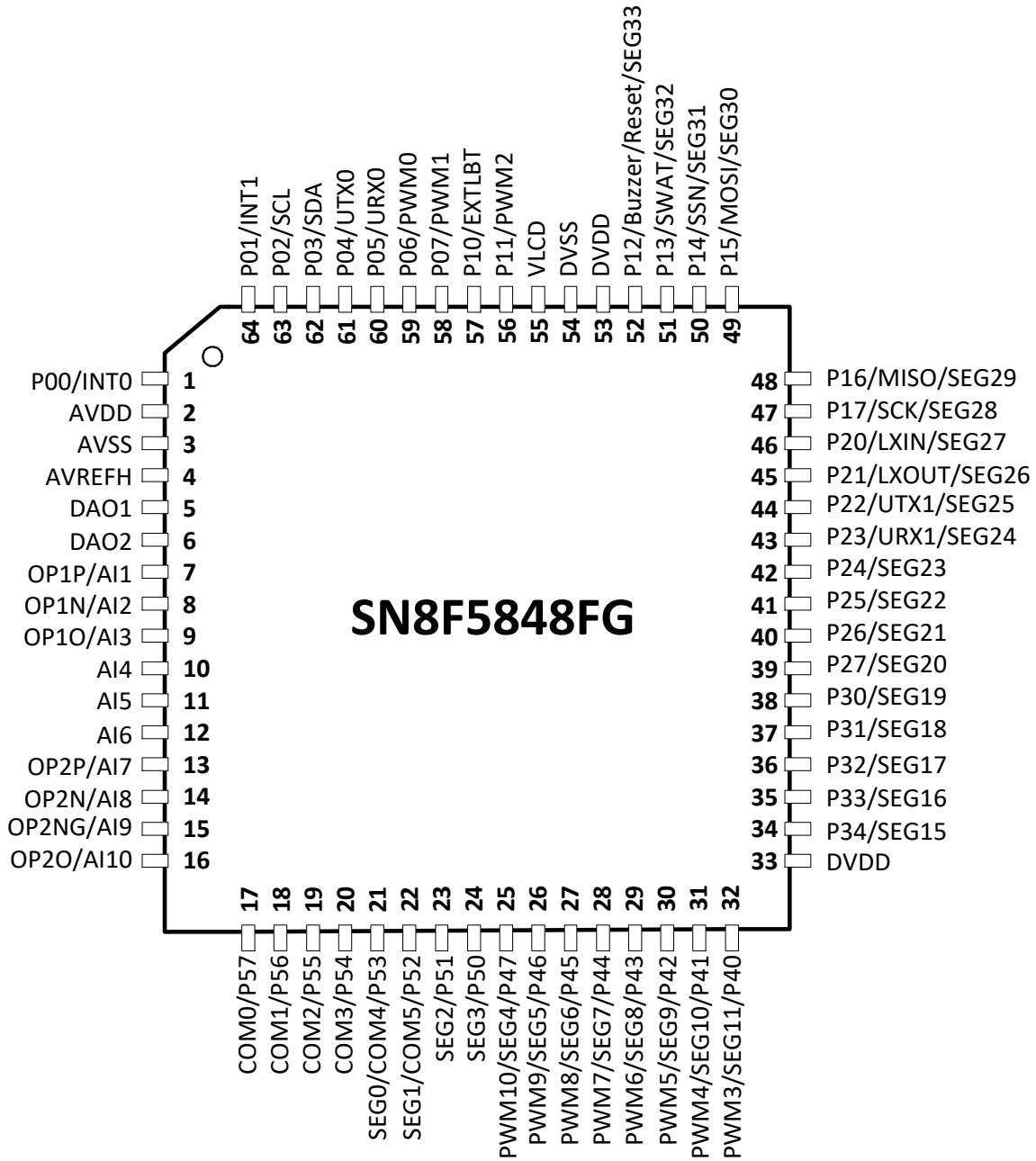
1.9	Jan 2024	1.Update 8.3 Power Management Characteristics (P36~37) 2.Update 8.4 System Clock and Power Management Registers(P38) 3.Update 24.4 Sample Code (P139)

4 Pin Assignments

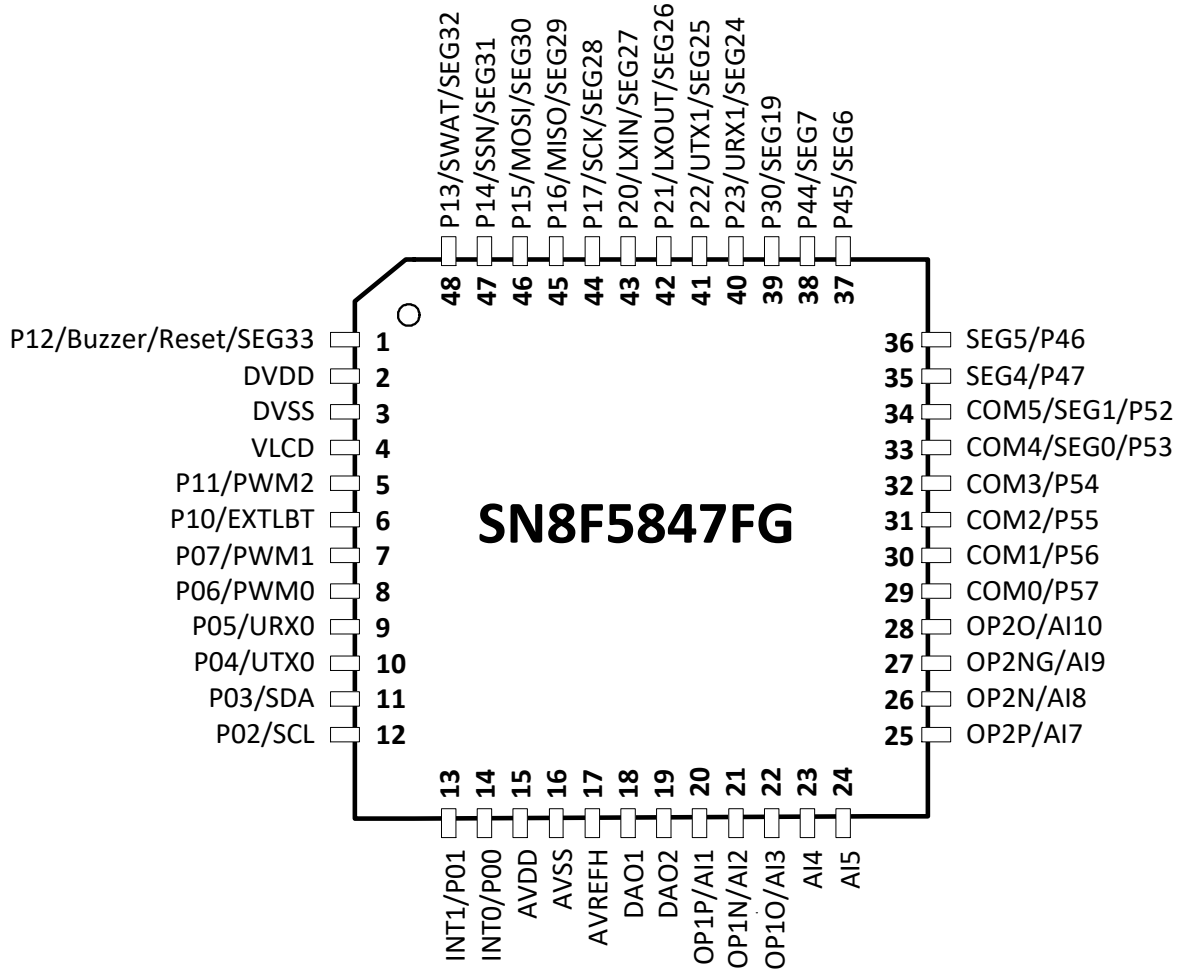
4.1 SN8F5849F (LQFP80)



4.2 SN8F5848F (LQFP64)



4.3 SN8F5847F (LQFP48)



4.4 SN8F5844T (TSSOP28)

AI5	1	28	AI6
DAO1	2	27	AI10/OP2O
AVREFH	3	26	AI9/OP2NG
VSS	4	25	AI3/OP1O
P21/LXOUT/SEG26	5	24	AI2/OP1N
P20/LXIN/SEG27	6	23	AI1/OP1P
P17/SCK/SEG28	7	22	P03/SDA
P15/MOSI/SEG30	8	21	P01/INT1
P14/SSN/SEG31	9	20	P00/INT0
P13/SWAT/SEG32	10	19	P02/SCL
P12/Buzzer/Reset/SEG33	11	18	P04/UTX0
VDD	12	17	P05/URX0
P11/PWM2	13	16	P06/PWM0
P10/EXTLBT	14	15	P07/PWM1

4.5 Pin Descriptions

Power Pins

Pin Name	Type	Description
DVDD	Power	Digital power supply
DVSS	Power	Digital Ground
AVDD	Power	Analog power supply
AVSS	Power	Analog Ground

Port 0

Pin Name	Type	Description
P0.0	Digital I/O	GPIO
INT0	Digital I/O	INT0 Interrupt input.
P0.1	Digital I/O	GPIO
INT1	Digital I/O	INT1 Interrupt input.
P0.2	Digital I/O	GPIO
SCL	Digital I/O	I2C: clock output (master) clock input (slave)
P0.3	Digital I/O	GPIO
SDA	Digital I/O	I2C: data pin
P0.4	Digital I/O	GPIO
UTX	Digital Output	UART: transmission pin
P0.5	Digital I/O	GPIO
URX	Digital Input	UART: reception pin
P0.6	Digital I/O	GPIO
PWM0	Digital Output	PWM0 output pin
P0.7	Digital I/O	GPIO
PWM1	Digital Output	PWM1 output pin

Port 1

Pin Name	Type	Description
P1.0	Digital I/O	GPIO
EXLBT	Analog Input	Comparator external Input
P1.1	Digital I/O	GPIO
PWM2	Digital Output	PWM2 output pin
P1.2	Digital I/O	GPIO
Buzzer	Digital Output	Buzzer output pin
Reset	Digital Input	Ext Reset pin.
SEG33	LCD Output	LCD: SEG33
P1.3	Digital I/O	GPIO
SWAT	Digital I/O	Debug interface
SEG32	LCD Output	LCD: SEG32

P1.4 SSN SEG31	Digital I/O Digital Input LCD Output	GPIO SPI: Slave selection pin LCD: SEG31
P1.5 MOSI SEG30	Digital I/O Digital Input LCD Output	GPIO SPI: transmission pin (master) reception pin (slave) LCD: SEG30
P1.6 MISO SEG29	Digital I/O Digital Input LCD Output	GPIO SPI: reception pin (master) transmission pin (slave) LCD: SEG29
P1.7 SCK SEG28	Digital I/O Digital Input LCD Output	GPIO SPI: clock output (master) clock input (slave) LCD: SEG28

Port 2

Pin Name	Type	Description
P2.0 LXIN SEG27	Digital I/O Analog Input LCD Output	GPIO System clock: external clock input LCD: SEG27
P2.1 LXOUT SEG26	Digital I/O Analog Output LCD Output	GPIO System clock: drive external crystal LCD: SEG26
P2.2 SEG25 UTX0	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG25 UART0: transmission pin
P2.3 SEG24 URX0	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG24 UART0: reception pin
P2.4 SEG23	Digital I/O LCD Output	GPIO LCD: SEG23
P2.5 SEG22	Digital I/O LCD Output	GPIO LCD: SEG22
P2.6 SEG21	Digital I/O LCD Output	GPIO LCD: SEG21
P2.7 SEG20	Digital I/O LCD Output	GPIO LCD: SEG20

Port 3

Pin Name	Type	Description
P3.0 SEG19	Digital I/O LCD Output	GPIO LCD: SEG19
P3.1 SEG18	Digital I/O LCD Output	GPIO LCD: SEG18
P3.2 SEG17	Digital I/O LCD Output	GPIO LCD: SEG17
P3.3 SEG16	Digital I/O LCD Output	GPIO LCD: SEG16
P3.4 SEG15	Digital I/O LCD Output	GPIO LCD: SEG15
P3.5 SEG14	Digital I/O LCD Output	GPIO LCD: SEG14
P3.6 SEG13	Digital I/O LCD Output	GPIO LCD: SEG13
P3.7 SEG12	Digital I/O LCD Output	GPIO LCD: SEG12

Port 4

Pin Name	Type	Description
P4.0 SEG11 PWM3	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG11 PWM3 output pin
P4.1 SEG10 PWM4	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG10 PWM4 output pin
P4.2 SEG09 PWM5	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG09 PWM5 output pin
P4.3 SEG08 PWM6	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG08 PWM6 output pin
P4.4 SEG07 PWM7	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG07 PWM7 output pin
P4.5 SEG06 PWM8	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG06 PWM8 output pin
P4.6 SEG05 PWM9	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG05 PWM9 output pin

P4.7 SEG04 PWM10	Digital I/O LCD Output Digital I/O	GPIO LCD: SEG04 PWM9 output pin

Port 5

Pin Name	Type	Description
P5.0 SEG03	Digital I/O LCD Output	GPIO LCD: SEG03
P5.1 SEG02	Digital I/O LCD Output	GPIO LCD: SEG02
P5.2 COM5 SEG01	Digital I/O LCD Output LCD Output	GPIO LCD:COM5 LCD: SEG01
P5.3 COM4 SEG00	Digital I/O LCD Output LCD Output	GPIO LCD:COM4 LCD: SEG00
P5.4 COM3	Digital I/O LCD Output	GPIO LCD: COM3
P5.5 COM2	Digital I/O LCD Output	GPIO LCD: COM2
P5.6 COM1	Digital I/O LCD Output	GPIO LCD: COM1
P5.7 COM0	Digital I/O LCD Output	GPIO LCD: COM0

LCD

Pin Name	Type	Description
VLCD	LCD Output	LCD: output voltage
COM0	LCD Output	LCD: COM0
COM1	LCD Output	LCD: COM1
COM2	LCD Output	LCD: COM2
COM3	LCD Output	LCD: COM3
COM4 SEG0	LCD Output LCD Output	LCD: COM4 LCD: SEG0
COM5 SEG1	LCD Output LCD Output	LCD: COM5 LCD: SEG1
SEG2~SEG33	LCD Output	LCD: SEG2 ~ SEG33

Analog Pins

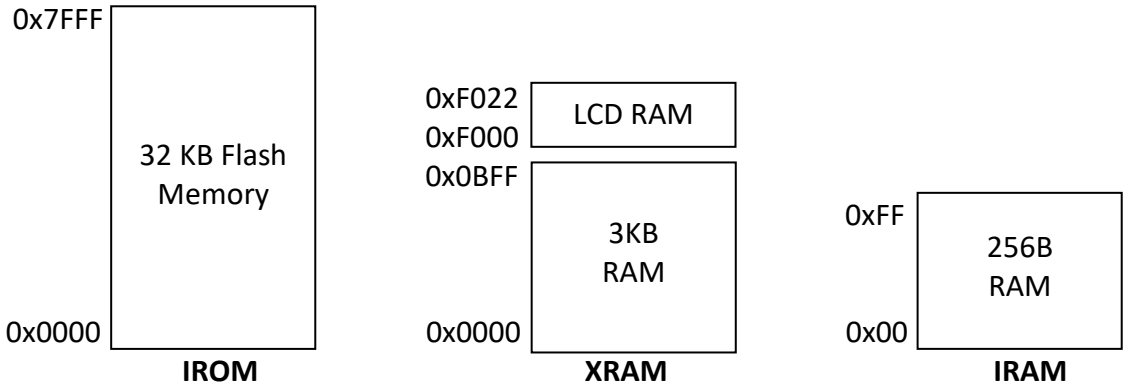
Pin Name	Type	Description
DAO2	Analog Output	DAC2: Output Pin
DAO1	Analog Output	DAC1: Output Pin
AVREFH	Analog Input	DAC: DAC External Vref input Pin.
AI10 OP2O	Analog Input Analog Output	ADC: Input channel 10 OPA2: output pin
AI9 OP2NG	Analog Input Analog Output	ADC: Input channel 9 OPA2: output pin
AI8 OP2N	Analog Input Analog Input	ADC: Input channel 8 OPA2: negative input pin
AI7 OP2P	Analog Input Analog Input	ADC: Input channel 7 OPA2: positive input pin
AI6	Analog Input	ADC: Input channel 6
AI5	Analog Input	ADC: Input channel 5
AI4	Analog Input	ADC: Input channel 4
AI3 OP1O	Analog Input Analog Output	ADC: Input channel 3 OPA1: output pin
AI2 OP1N	Analog Input Analog Input	ADC: Input channel 2 OPA1: negative input pin
AI1 OP1P	Analog Input Analog Input	ADC: Input channel 1 OPA1: positive input pin

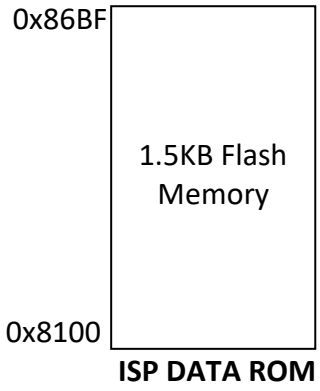
5 CPU

SN8F5000 family is an enhanced 8051 microcontroller (MCU). It is fully compatible with MCS-51 instructions, hence the ability to cooperate with modern development environment (e.g. Keil C51). Generally speaking, SN8F5000 CPU has 9.4 to 12.1 times faster than the original 8051 at the same frequency.

5.1 Memory Organization

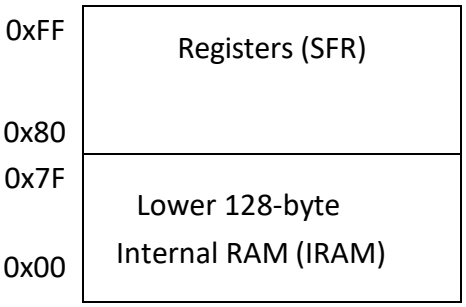
SN8F5840 builds in three on-chip memories: internal RAM (IRAM), external RAM (XRAM) , program memory (IROM) and ISP data program memory (ISP DATA ROM). The internal RAM is a 256-byte RAM which has higher access performance (direct and indirect addressing). By contrast, the external RAM has 3 KB of size, but it requires a longer access period. The program memory is a 32 KB non-volatile memory and has a maximum 16MHz speed limitation.





5.2 Direct Addressing: IRAM and SFR

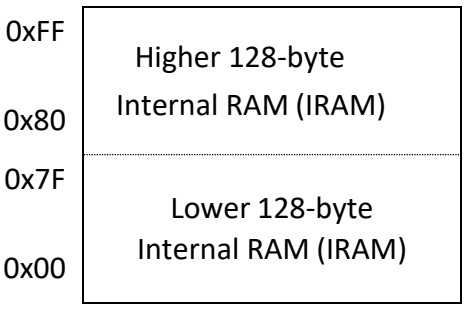
Direct addressing instructions (e.g. MOV A, direct) can access the lower 128-byte internal RAM (address range: 0x00 – 0x7F) and all registers (SFR, address range: 0x80 – 0xFF).



Moreover, the lowest 32 bytes of internal RAM (0x00 – 0x1F) can be seen as four set of R0 – R7 working registers which are addressable by fastest assembly instructions like MOV A, R0. Internal RAM from 0x20 to 0x2F and every 0x0/0x8-ending SFR addresses are bit-addressable.

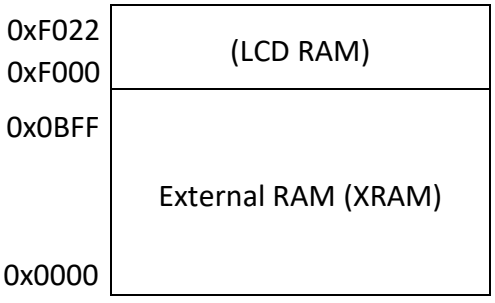
5.3 Indirect Addressing: IRAM

Although direct addressing instructions take fewer period to access internal RAM than indirect addressing, the second addressing type has the full range accessibility to internal RAM and is the only method to access the higher 128-byte internal RAM (0x80 – 0xFF).



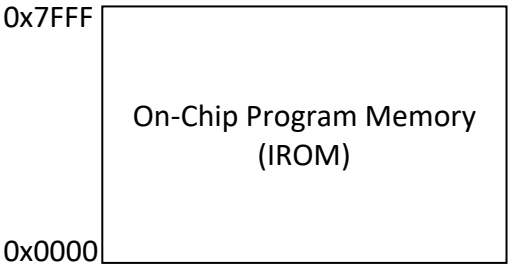
5.4 External RAM (XRAM)

The external RAM enlarges the capacity of variables, yet it is the lowest access performance in the contrast of internal RAM. Since frequently used variables and local variables are expected to store in internal RAM, the vast majority of external RAM usages are specific. It can be allocated as a variable storage area for lower priority tasks, or look-up table preloaded from ROM to speed up the access period. LCD RAM is also located in external RAM.



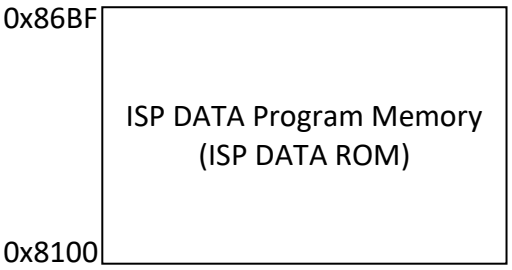
5.5 Program Memory (IROM)

The program memory is a non-volatile storage area where stores code, look-up ROM table, and other data with occasional modification. It can be updated by debug tools like SN-Link Adapter II, and a program can also self-update via in-system program process (refer to In-system Program).



5.6 ISP DATA Program Memory (ISP DATA ROM)

The ISP program memory is a non-volatile storage area where stores data occasional modification.



5.7 Program Memory Security

The SN8F5800 provides security options at the disposal of the designer to prevent unauthorized access to information stored in FLASH memory. When enable security option, the ROM code is secured and not dumped complete ROM contents. ROM security rule is all address ROM data protected and outputs 0x00.

5.8 Data Pointer

A data pointer helps to specify the XRAM and IROM address while performing MOVX and MOVC instructions. The microcontroller has two set of data pointer (DPH/DPL and DPH1/DPL1) which is selectable by DPS register. The DPC register controls two functions: next DPTR selection and automatically increase/decrease DPTR function.

The next DPTR selection can specify which DPTR is anticipated to use after perform MOVX @DPTR instruction. In other word, the DPS can automatically swap between the two data pointers. To enable this function: write 0 to DPSEL and fill 1 to NDPS firstly, then write 1 to DPSEL and fill 0 to NDPS register.

The automatically increase/decrease DPTR function can make an increment or decrement after perform MOVX @DPTR instruction. As a result, it enables a continuous external RAM access without re-specified DPTR value. Those functions are controlled by the DPC Register, where there are separate DPC register bits for each DPTR, to provide high flexibility in data transfers. The DPC Register address 0x93 points to the window where the actual DPC is selected using the DPS Register, same as for the DPTR.

5.9 Stack

Stack can be assigned to any area of internal RAM (IRAM). However, it requires manual assignment to ensure its area does not overlap other RAM's variables. An overflow and underflow stack could also mistakenly overwrite other RAM's variables; thus, these factors should be considered while arrange the size of stack.

By default, stack pointer (SP register) points to 0x07 which means the stack area begin at IRAM address 0x08. In other word, if a planned stack area is assigned from IRAM address 0xC0, the

appropriate SP register is anticipated to set at 0xBF after system reset.

An assembly PUSH instruction costs one byte of stack. LCALL, ACALL instructions and interrupt respectively costs two bytes stack. POP-instruction decreases one count, and a RET/RETI subtract two counts of stack pointer.

5.10 Stack and Data Pointer Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SP	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
DPL	DPL7	DPL6	DPL5	DPL4	DPL3	DPL2	DPL1	DPL0
DPH	DPH7	DPH6	DPH5	DPH4	DPH3	DPH2	DPH1	DPH0
DPL1	DPL17	DPL16	DPL15	DPL14	DPL13	DPL12	DPL11	DPL10
DPH1	DPH17	DPH16	DPH15	DPH14	DPH13	DPH12	DPH11	DPH10
DPS	-	-	-	-	-	-	-	DPSEL
DPC	-	-	-	-	NDPS	ATMS	ATMD	ATME

SP Register (0x81)

Bit	Field	Type	Initial	Description
7..0	SP	R/W	0x07	Stack pointer

DPL Register (0x82)

Bit	Field	Type	Initial	Description
7..0	DPL[7:0]	R/W	0x00	Low byte of DPTR0

DPH Register (0x83)

Bit	Field	Type	Initial	Description
7..0	DPH[7:0]	R/W	0x00	High byte of DPTR0

DPL1 Register (0x84)

Bit	Field	Type	Initial	Description
7..0	DPL1[7:0]	R/W	0x00	Low byte of DPTR1

DPH1 Register (0x85)

Bit	Field	Type	Initial	Description
7..0	DPH1[7:0]	R/W	0x00	High byte of DPTR1

DPS Register (0x92)

Bit	Field	Type	Initial	Description
7..1	Reserved	R	0x00	
0	DPSEL	R/W	0	DPTR selection 0: DPH/DPL (DPTR0) is selected 1: DPH1/DPL1 (DPTR1) is selected

DPC Register (0x93)

Bit	Field	Type	Initial	Description
7..4	Reserved	R	0x0	
3	NDPS	R/W	0	Next DPTR selection The DPSEL loads this bit automatically after perform any MOVX @DPTR instruction.
2..1	ATMS/ATMD	R/W	00	Automatically increase/decrease DPTR (if ATME applied) 00: +1 after any MOVX @DPTR instruction 01: -1 after any MOVX @DPTR instruction 10: +2 after any MOVX @DPTR instruction 11: -2 after any MOVX @DPTR instruction
0	ATME	R/W	0	Automatically increase/decrease DPTR function 0: Disable 1: Enable

6 Special Function Registers

6.1 Special Function Register Memory Map

BIN / HEX	000	001	010	011	100	101	110	111
F8	P5	P0M	P1M	P2M	P3M	P4M	P5M	PFLAG
F0	B	P0UR	P1UR	P2UR	P3UR	P4UR	P5UR	SRST
E8	P4	MD0	MD1	MD2	MD3	MD4	MD5	ARCON
E0	ACC	SPSTA	SPCON	SPDAT	P1OC	CLKSEL	CLKCMD	-
D8	S0CON2	-	I2CDAT	I2CADR	I2CCON	I2CSTA	SMBSEL	SMBDST
D0	PSW	ADB	ADR	ADM2	LCDM3	PWMOUTM	BZRM	LBTM
C8	ADM	TC1M	TC1RL	TC1RH	TC1CL	TC1CH	TC1DL	TC1DH
C0	IRCON	TC0M	TC0RL	TC0RH	TC0CL	TC0CH	TC0DL	TC0DH
B8	IEN1	IP1	S0RELH	T0M	T0C	MIN	SEC	IRCON2
B0	P3	LCDM1	LCDM2	VREFH	ADT	DA2BL	DA2BH	OPM3
A8	IEN0	IP0	S0RELL	DAM	DA1BL	DA1BH	OPM1	OPM2
A0	P2	TC2M	TC2RL	TC2RH	TC2CL	TC2CH	TC2DL	TC2DH
98	S0CON	S0TBUF	IEN2	P2CON	P1CON	P5CON	P3CON	P4CON
90	P1	P1W	DPS	DPC	PECMD	PEROML	PEROMH	PERAM
88	CRCM	CRCDATL	CRCDATH	-	-	-	-	PEDGE
80	P0	SP	DPL	DPH	DPL1	DPH1	WDTR	PCON

6.2 Special Function register Description

0x80 - 0x9F Registers Description

Register	Address	Description
P0	0x80	Port 0 data buffer.
SP	0x81	Stack pointer register.
DPL	0x82	Data pointer 0 low byte register.
DPH	0x83	Data pointer 0 high byte register.
DPL1	0x84	Data pointer 1 low byte register.
DPH1	0x85	Data pointer 1 high byte register.
WDTR	0x86	Watchdog timer clear register.
CRCM	0x87	CRC mode controls register.
CRCDATL	0x88	CRC Low Byte counter.
CRCDATH	0x89	CRC High Byte counter.
-	0x8A~0x8D	-
CKCON	0x8E	Extended cycle controls register.
PEDGE	0x8F	External interrupt edge controls register.
P1	0x90	Port 1 data buffer.
P1W	0x91	Port 1 wake-up controls register.
DPS	0x92	Data pointer selects register.
DPC	0x93	Data pointer controls register.
PECMD	0x94	In-System Program command register
PEROML	0x95	In-System Program ROM address low byte
PEROMH	0x96	In-System Program ROM address high byte
PERAM	0x97	In-System Program RAM mapping address
S0CON	0x98	UART control register.
S0BUF	0x99	UART data buffer.
IEN2	0x9A	Interrupts enable register
P2CON	0x9B	Port 2 configuration controls register.
P1CON	0x9C	Port 1 configuration controls register.
P5CON	0x9D	Port 5 configuration controls register.
P3CON	0x9E	Port 3 configuration controls register.
P4CON	0x9F	Port 4 configuration controls register.

0xA0 - 0xBF Registers Description

Register	Address	Description
P2	0xA0	Port 2 data buffer
TC2M	0xA1	TC2 timer mode controls register
TC2RL	0xA2	TC2timer counter reload buffer low byte
TC2RH	0xA3	TC2 timer counter reload buffer high byte
TC2CL	0xA4	TC2 timer Low Byte counter
TC2CH	0xA5	TC2 timer High Byte counter
TC2DL	0xA6	PWM2 duty control low byte buffer
TC2DH	0xA7	PWM2 duty control high byte buffer
IEN0	0xA8	Interrupts enable register
IP0	0xA9	Interrupts priority register
S0RELL	0xAA	UART reload low byte register.
DAM	0xAB	DAC mode controls register.
DA1BL	0xAC	DAC1 control low byte buffer
DA1BH	0xAD	DAC1 control high byte buffer
OPM1	0xAE	OP1M mode controls register.
OPM2	0xAF	OP2M mode controls register.
P3	0xB0	Port 3 data buffer.
LCDM1	0xB1	LCD Mode 1 control register.
LCDM2	0xB2	LCD Mode 2 control register.
VREFH	0xB3	SAR ADC reference voltage controls register.
ADT	0xB4	SAR ADC Calibration controls register.
DA2BL	0xB5	DAC2 control low byte buffer
DA2BH	0xB6	DAC2 control high byte buffer
OPM3	0xB7	OP3M mode controls register.
IEN1	0xB8	Interrupts enable register.
IP1	0xB9	Interrupts priority register.
S0RELH	0xBA	UART reload high byte register.
T0M	0xBB	Timer0 Mode control register.
T0C	0xBC	Timer0 counter register.
MIN	0xBD	Timer0 minutes register.
SEC	0xBE	Timer0 second register.
IRCON2	0xBF	Interrupts request register.

0xC0 - 0xDF Registers Description

Register	Address	Description
IRCON	0xC0	Interrupts request register.
TC0M	0xC1	TC0 control register.
TC0RL	0xC2	TC0 reload buffer low byte.
TC0RH	0xC3	TC0 reload buffer high byte.
TC0CL	0xC4	TC0 counter low byte.
TC0CH	0xC5	TC0 counter high byte.
TC0DL	0xC6	PWM0 duty control low byte buffer.
TC0DH	0xC7	PWM0 duty control high byte buffer.
ADM	0xC8	SAR ADC control register.
TC1M	0xC9	TC1 control register.
TC1RL	0xCA	TC1 reload buffer low byte.
TC1RH	0xCB	TC1 reload buffer high byte.
TC1CL	0xCC	TC1 counter low byte.
TC1CH	0xCD	TC1 counter high byte.
TC1DL	0xCE	PWM1 duty control low byte buffer.
TC1DH	0xCF	PWM1 duty control high byte buffer.
PSW	0xD0	System flag register.
ADB	0xD1	SAR ADC data buffer.
ADR	0xD2	SAR ADC resolution selects register.
ADM2	0xD3	ADC Mode 2 control register
LCDM3	0xD4	LCD Mode 3 control register
PWMOUT	0xD5	PWM Out control register2.
BZRM	0xD6	Buzzer control register.
LBTM	0xD7	Comparator and Low battery detection control register.
S0CON2	0xD8	UART control register2.
-	0xD9	-
I2CDAT	0xDA	I2C data buffer.
I2CADR	0xDB	Own I2C slave address.
I2CCON	0xDC	I2C interface operation control register.
I2CSTA	0xDD	I2C Status Code.
SMBSEL	0xDE	SMBus mode controls register.
SMBDST	0xDF	SMBus internal timeout register.

0xE0 - 0xFF Registers Description

Register	Address	Description
ACC	0xE0	Accumulator register.
SPSTA	0xE1	SPI statuses register.
SPCON	0xE2	SPI control register.
SPDAT	0xE3	SPI data buffer.
P1OC	0xE4	Open drain controls register.
CLKSEL	0xE5	Clock switch selects register.
CLKCMD	0xE6	Clock switch controls Register.
-	0xE7	-
P4	0xE8	Port 4 data buffer.
MD0	0xE9	MDU controls register 0.
MD1	0xEA	MDU controls register 1.
MD2	0xEB	MDU controls register 2.
MD3	0xEC	MDU controls register 3.
MD4	0xED	MDU controls register 4.
MD5	0xEE	MDU controls register 5.
ARCON	0xEF	MDU Arithmetic control register.
B	0xF0	Multiplication/ division instruction data buffer.
P0UR	0xF1	Port 0 pull-up resister controls register.
P1UR	0xF2	Port 1 pull-up resister controls register.
P2UR	0xF3	Port 2 pull-up resister controls register.
P3UR	0xF4	Port 3 pull-up resister controls register.
P4UR	0xF5	Port 4 pull-up resister controls register.
P5UR	0xF6	Port 5 pull-up resister controls register.
SRST	0xF7	Software reset controls register.
P5	0xF8	Port 5 data buffer.
P0M	0xF9	Port 0 input/output mode register.
P1M	0xFA	Port 1 input/output mode register.
P2M	0xFB	Port 2 input/output mode register.
P3M	0xFC	Port 3 input/output mode register.
P4M	0xFD	Port 4 input/output mode register.
P5M	0xFE	Port 5 input/output mode register.
PFLAG	0xFF	Reset flag register.

6.3 System Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ACC	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0
B	B7	B6	B5	B4	B3	B2	B1	B0
PSW	CY	AC	F0	RS1	RS0	OV	F1	P

ACC Register (0xE0)

Bit	Field	Type	Initial	Description
7..0	ACC[7:0]	R/W	0x00	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 overflow (OV) or there is carry (C or AC) and parity (P) occurrence, then these flags will be set to PSW register.

B Register (0xF0)

Bit	Field	Type	Initial	Description
7..0	B[7:0]	R/W	0x00	The B register is used during multiplying and division instructions. It can also be used as a scratch-pad register to hold temporary data.

PSW Register (0xD0)

Bit	Field	Type	Initial	Description
7	CY	R/W	0	Carry flag. 0: Addition without carry, subtraction with borrowing signal, rotation with shifting out logic “0”, comparison result < 0. 1: Addition with carry, subtraction without borrowing, rotation with shifting out logic “1”, comparison result ≥ 0.
6	AC	R/W	0	Auxiliary carry flag. 0: If there is no a carry-out from 3rd bit of Accumulator in BCD operations. 1: If there is a carry-out from 3rd bit of Accumulator in BCD operations.
5	F0	R/W	0	General purpose flag 0. General purpose flag available for user.
4..3	RS[1:0]	R/W	00	Register bank select control bit, used to select working register bank. 00: 00H – 07H (Bnak0) 01: 08H – 0FH (Bnak1) 10: 10H – 17H (Bnak2) 11: 18H – 1FH (Bnak3)
2	OV	R/W	0	Overflow flag. 0: Non-overflow in Accumulator during arithmetic Operations. 1: overflow in Accumulator during arithmetic Operations.
1	F1	R/W	0	General purpose flag 1. General purpose flag available for user.
0	P	R	0	Parity flag. Reflects the number of ‘1’s in the Accumulator. 0: if Accumulator contains an even number of ‘1’s. 1: Accumulator contains an odd number of ‘1’s.

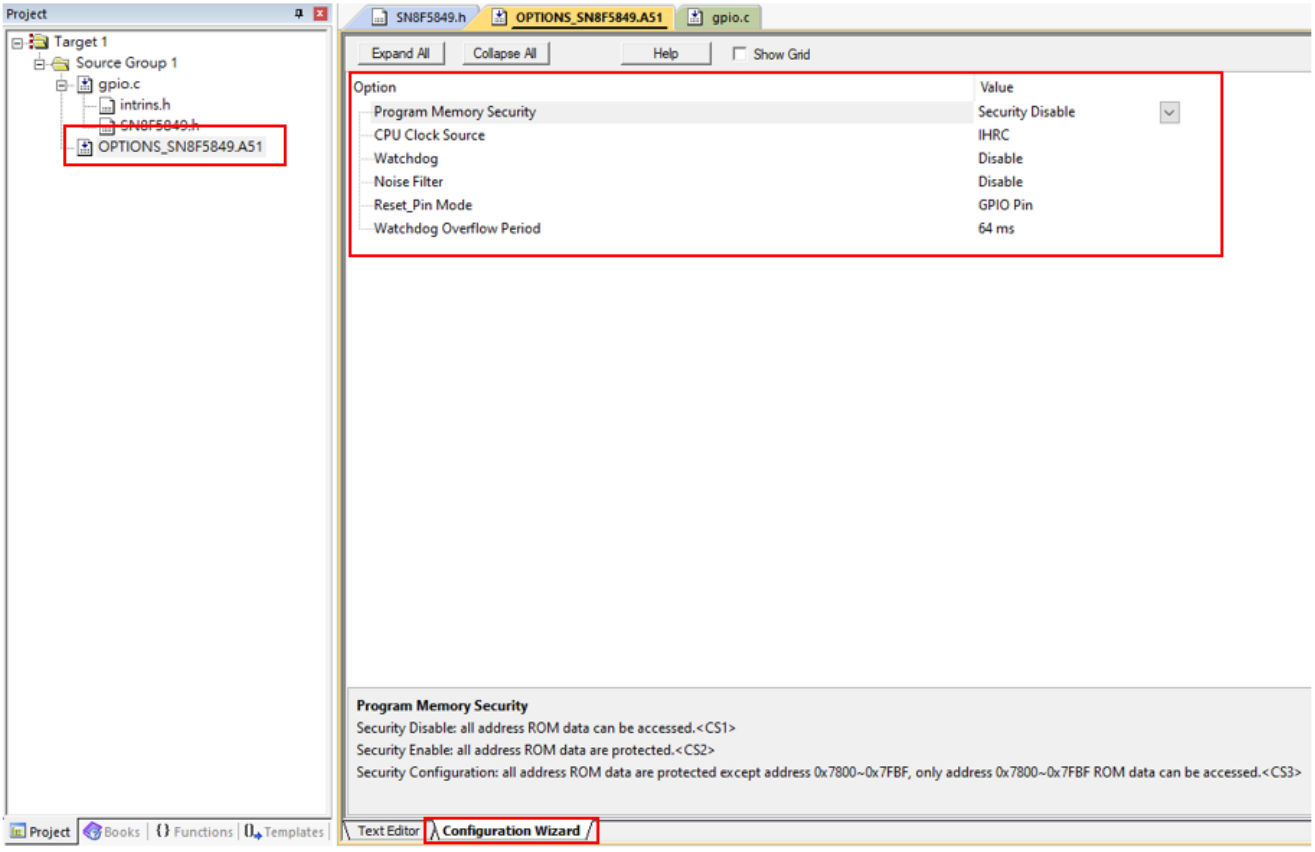
7 Reset and Power-on Controller

The reset and power-on controller has five reset sources: low voltage detectors (LVDs), watchdog, programmable external reset pin, and software reset. The first three sources would trigger an additional power-on sequence. Subsequently, the microcontroller initializes all registers and starts program execution with its reset vector (ROM address 0x0000).

7.1 Configuration of Reset and Power-on Controller

SONiX publishes a *SN8F58xx_OPTIONS.A51 file* in *SN-Link Driver for Keil C51.exe* (downloadable on cooperative website: www.sonix.com.tw). This *options file* contains appropriate parameters of reset sources and CPU clock source selection, and is strongly recommended to add to Keil project. *SN8F5000 Debug Tool Manual* provides the further detail of this configuration. The option items are as following:

- Program Memory Security
- CPU Clock Source
- Watchdog
- Noise Filter
- Reset Pin Mode
- Watchdog Overflow Period



The code option is the system hardware configurations including CPU Clock Source option, noise filter option, watchdog timer operation, ISP Program Area option, reset pin mode option and flash ROM security control. The code option items are as following table:

Code Option	Content	Function Description
Program Memory Security	Disable	Disable ROM code security function. Any address byte can read out.
	Configuration	Enable “especially ROM code security function”. All address always output 0x00, except address 0x7800~0x7FBF can read out.
	Enable	Enable ROM code security function. Any address byte always output 0x00.
CPU Clock Source	IHRC	High speed internal 32MHz RC. XIN/XOUT pins are bi-direction GPIO mode
	IHRC_RTC	High speed internal 32MHz RC. XIN/XOUT pins are oscillator mode for external low clock 32768Hz oscillator.
Watchdog Reset	Always	Watchdog timer is always on enable even in STOP mode and IDLE mode
	Enable	Enable watchdog timer. Watchdog timer stops in STOP mode and IDLE mode
	Disable	Disable Watchdog function
Noise Filter	Disable	Disable Noise Filter
	Enable	Enable Noise Filter
Reset Pin Mode	Reset with De-bounce	Enable External reset pin with De-bounce
	Reset without De-bounce	Enable External reset pin without De-bounce
	GPIO with P12	Enable P12 GPIO Mode
Watchdog Overflow Period	64ms	Watchdog timer clock source F_{ILRC} /4
	128ms	Watchdog timer clock source F_{ILRC} /8
	256ms	Watchdog timer clock source F_{ILRC} /16
	512ms	Watchdog timer clock source F_{ILRC} /32

7.2 Power-on Sequence

A power-on sequence would be triggered by LVD, watchdog, and external reset pin. It takes place between the end of reset signal and program execution. Overall, it includes two stages: power stabilization period, and clock stabilization period.

The power stabilization period spends 5ms in typical condition. Afterward the microcontroller fetches CPU Clock Source selection automatically. The selected clock source would be driven, and the system counts 4096 times of the clock period to ensure its reliability.

7.3 LVD Reset

The low voltage detectors monitor VDD pin's voltage at one level 1.8 V. when the VDD voltage is lower than 1.8V, the MCU system will be reset.

7.4 Watchdog Reset

Watchdog is a periodic reset signal generator for the purpose of monitoring the execution flow. Its internal timer is expected to be cleared in a check point of program flow; therefore, the actual reset signal would be generated only after a software problem occurs. Writing 0x5A to WDTR is the proper method to place a check point in program.

```
1     WDTR = 0x5A;
```

WDT clock pre-scaler	Clock/1	Clock/2	Clock/4	Clock/8
Watchdog interval time	64ms	128ms	256ms	512ms

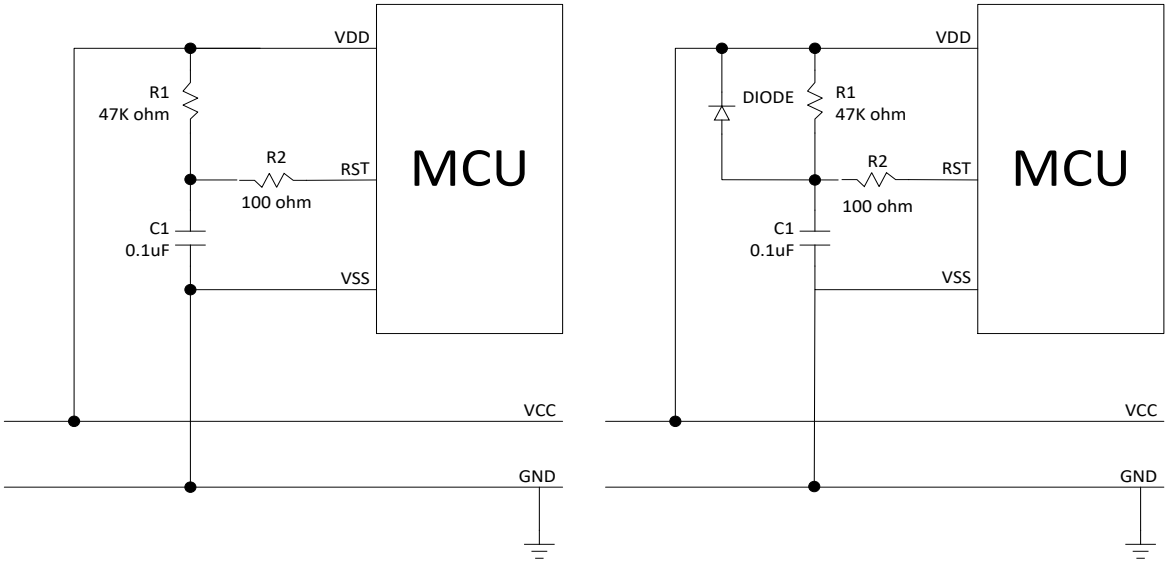
The operation mode of watchdog is configurable in options file:
Always mode counts its internal timer in all CPU operation modes (normal, IDLE, SLEEP);
Enable mode counts its internal timer during CPU stays in normal mode, and it would not trigger watchdog reset in IDLE and STOP modes;
Disable mode suspends its internal timer at all CPU modes, and the watchdog would not trigger in this condition.

When watchdog is operating in always mode, the system will consume additional power.

7.5 External Reset Pin

Programmable external reset pin is configurable in options file. Once it is enabled, it monitors its shared pin's logic level. A logical low (lower than 30% of VDD) would immediately trigger system reset until the input is recovered to high (lager than 70% of VDD).

An optional de-bounce period can improve reset signal's stability. Instead of immediate reset, the system reset requires an 8-ms-long logic low to avoid bouncing from a button key. Any signal lower than de-bounce period would not affect the CPU's execution.



✸ ***Note:***

1. The reset circuit is no any protection against unusual power or brown out reset on the left side of the figure.

2. The R2 100 ohm resistor of "Simply reset circuit" and "Diode & RC reset circuit" is necessary to limit any current flowing into reset pin from external capacitor C in the event of reset pin breakdown due to Electrostatic Discharge (ESD) or Electrical Over-stress (EOS) on the right side of the figure.

7.6 Software Reset

A software reset would be generated after consecutively set SRSTREQ register. As a result, this procedure enables firmware's ability to reset microcontroller (e.g. reset after firmware update). The following sample C code repeatedly set the least bit of SRST register to perform software reset.

```
1  SRST = 0x01;
2  SRST = 0x01;
```

7.7 Reset and Power-on Controller Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	-	-	-	-	-	-
SRST	-	-	-	-	-	-	-	SRSTREQ
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0

PFLAG Register

Bit	Field	Type	Initial	Description
7	POR	R	0	This bit is automatically set if the microcontroller has been reset by LVD.
6	WDT	R	0	This bit is automatically set if the microcontroller has been reset by watchdog.
5..0	Reserved	R	0	

SRST Register

Bit	Field	Type	Initial	Description
7..1	Reserved	R	0	
0	SRSTREQ	R/W	0	Consecutively set this bit for two times to trigger software reset.

WDTR Register (0x86)

Bit	Field	Type	Initial	Description
7..0	WDTR[7:0]	W	-	Watchdog clear is controlled by WDTR register. Moving 0x5A data into WDTR is to reset watchdog timer.

8 System Clock and Power Management

For power saving purpose, the microcontroller built in three different operation modes: normal, IDLE, and STOP mode.

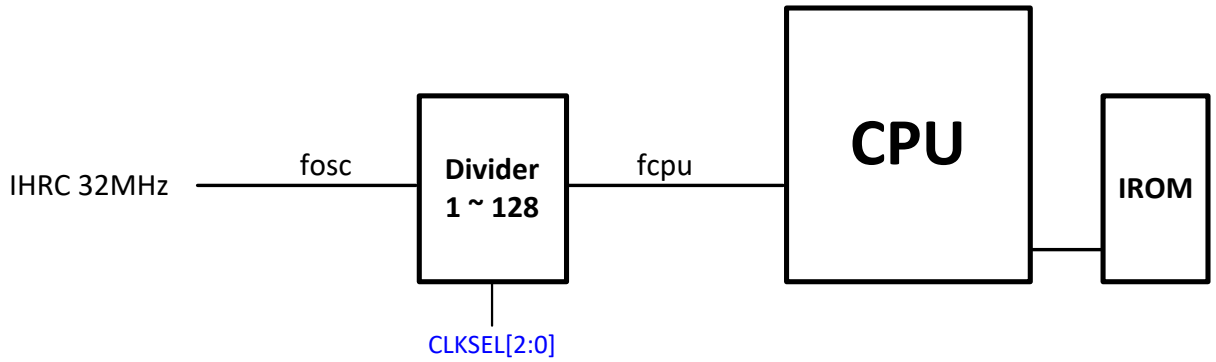
The normal mode means that CPU and peripheral functions are under normally execution. The system clock is based on the combination of source selection, clock divider, and program memory wait state. IDLE mode is the situation that temporarily suspends CPU clock and its execution, yet it remains peripherals' functionality (e.g. timers, SPI, UART, and I2C). By contrast, STOP mode disables all functions and clock generator until a wakeup signal to return normal mode. Additionally, the LCD function and T0-RTC function are also still can work in STOP Mode for low standby current application.

8.1 System Clock

The microcontroller has only one system clock source, on-chip clock generator (IHRC 32MHz). The reset and power-on controller automatically loads IHRC clock during power-on sequence. Therefore, the IHRC clock is as 'fosc' domain which is a fixed frequency at any time.

Subsequently, the fosc is divided by 2 to 128 times which is controlled by CLKSEL register. The CPU input the divided clock as its operation base (named fcpu). Applying CLKSEL's setting when CLKCMD register be written 0x69.

```
1   CLKSEL  =  0x06; // set fcpu = fosc / 2
2   CLKCMD  =  0x69; // Apply CLKSEL's setting
```



ROM interface is built in between CPU and IROM (program memory). It optionally extends the data fetching cycle in order to support lower speed program memory.

✸ ***Note: For user develop program in C language or assembly, the first line of the program set CLKSEL= 0x07~0x00, CLKMD= 0x69.***

System clock rate and program memory extended cycle limitation as follows.

Code Option CPU Clock Source	Fcpu = CLKSEL[2:0]
IHRC 32M	**Only Support** 000 = fosc / 128 001 = fosc / 64 010 = fosc / 32 011 = fosc / 16 100 = fosc / 8 101 = fosc / 4 110 = fosc / 2 111 = reserved

8.2 Noise Filter

The Noise Filter controlled by Noise Filter option is a low pass filter and supports crystal mode. The purpose is to filter high rate noise coupling on high clock signal from external oscillator. In high noisy environment, enable Noise Filter option is the strongly recommendation to reduce noise effect.

8.3 Power Management

After the end of reset signal and power-on sequence, the CPU starts program execution at the speed of fcpu. Overall, the CPU and all peripherals are functional in this situation (categorized as normal mode).

The least two bits of PCON register (IDLE at bit 0 and STOP at bit 1) control the microcontroller's power management unit.

If IDLE/IDLE+ bit is set by program, only CPU clock source would be gated. Consequently, peripheral functions (such as timers, PWM, SPI, UART, and I2C) and clock generator (IHRC 32 MHz) remain execution in this status. Any change from P0/P1 input and interrupt events can make the microcontroller turns back to normal mode, and the IDLE/IDLE+ bit would be cleared automatically.

If STOP bit is set, by contrast, CPU, peripheral functions, and clock generator are suspended. Data storage in registers and RAM would be kept in this mode. Any change from P0/P1 can wake up the microcontroller and resume system's execution. STOP bit would be cleared automatically.

***Note:** For user who is develop program in C language, IDLE and STOP macros is strongly recommended to control the microcontroller's system mode, instead of set IDLE and STOP bits directly.

```
1   IDLE();         Use C51 Macros into IDLE MODE
2   IDLE_PLUS();    Use C51 Macros into IDLE+ MODE
3   STOP();         Use C51 Macros into STOP MODE
```

***Note:** When System into Stop mode, Please Set All Function disable for power saving.

***Note:** If the system will stop mode or idle/idle+ mode, All IO will config as inputs pull-up or output high/low state. (SN8F5849 & SN8F5848 & SN8F5847 &SN8F5844)

***Sample:** Disable all functions (Timer, UART, ADC, AVDDR, OPA...) in stop mode. (SN8F5849 & SN8F5848 & SN8F5847 &SN8F5844)

```
ADM = 0x00;          // ADEN Disable
VREFH = 0x80;        // AVREFH & VIBASEN Disable
DAM = 0x00;          // DA1EN & DA2EN Disable
OPM1 = 0x00;         // OP2EN & OP1EN Disable
I2CCON = 0x00;       // ENS1 Disable
SPCON = 0x00;        // SPEN Disable
S0CON = 0x00;        // REN0 Disable
S0CON2 = 0X00;       // TEN0 Disable
BZRM = 0x00;         // BZRENB Disable
TC2M = 0x00;         // TC2ENB Disable
TC1M = 0x00;         // TC1ENB Disable
TC0M = 0x00;         // TC0ENB Disable
T0M = 0x00;          // T0ENB Disable
LCDM1 = 0x00;        // LCDPEN & LCDEN Disable

STOP(); // Use C51 Macros into STOP MODE
```

***Sample:** If the system will stop mode or idle/idle+ mode, All I/O will config as inputs pull-up state. (SN8F5849 & SN8F5848 & SN8F5847 &SN8F5844)

```
P0M=0x00;
P1M=0x00;
P2M=0x00;
P3M=0x00;
P4M=0x00;
P5M=0x00;
P0UR=0xFF;
P1UR=0xFF;
P2UR=0xFF;
P3UR=0xFF;
P4UR=0xFF;
P5UR=0xFF;

// Use C51 Macros into STOP/IDLE/IDLE+ MODE
STOP(); or IDLE(); or IDLE_PLUS();

```

***Sample:** If the system will stop mode or idle mode, All I/O will config as output High or Low state. (SN8F5849 & SN8F5848 & SN8F5847 &SN8F5844)

```
P0M=0xFF;
P1M=0xFF;
P2M=0xFF;
P3M=0xFF;
P4M=0xFF;
P5M=0xFF;
P0=0xFF or 0x00;
P1=0xFF or 0x00;
P2=0xFF or 0x00;
P3=0xFF or 0x00;
P4=0xFF or 0x00;
P5=0xFF or 0x00;

// Use C51 Macros into STOP/IDLE/IDLE+ MODE
STOP(); or IDLE(); or IDLE_PLUS();
```

8.4 System Clock and Power Management Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKSEL	-	-	-	-	-	CLKSEL2	CLKSEL1	CLKSEL0
CLKCMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PCON	SMOD	-	-	IDLE+	P2SEL	GF0	STOP	IDLE
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W

CLKSEL Register (0xE5)

Bit	Field	Type	Initial	Description
7..3	Reserved	R	0x00	
2..0	CLKSEL[2:0]	R/W	111	CLKSEL would be applied by writing CLKCMD. 000: fcpu = fosc / 128 001: fcpu = fosc / 64 010: fcpu = fosc / 32 011: fcpu = fosc / 16 100: fcpu = fosc / 8 101: fcpu = fosc / 4 110: fcpu = fosc / 2 111: reserved * After set CLKSEL [2:0], then must write 0x69 to apply CLKSEL's setting.

CLKCMD Register (0xE6)

Bit	Field	Type	Initial	Description
7..0	CMD[7:0]	W	0x00	Writing 0x69 to apply CLKSEL's setting.

PCON Register (0x87)

Bit	Field	Type	Initial	Description
7				Refer to other chapter(s)
6..3	Reserved	R	0x00	
4	IDLE+	R/W	0	1: Microcontroller switch to IDLE+ mode
3	P2SEL	R/W	1	High-order address byte configuration bit. Chooses the higher byte of address ("XRAM[15:8]") during MOVX @Ri operations 0: The "XRAM[15:8]" = "P2REG". The "P2REG" is the contents of Port2 output register. 1: The "XRAM[15:8]" = 0x00.
2	GF0	R/W	0	General Purpose Flag
1	STOP	R/W	0	1: Microcontroller switch to STOP mode
0	IDLE	R/W	0	1: Microcontroller switch to IDLE mode

Note: **If the system will stop/idle/idel+ mode, All I/O will config as inputs pull-up or output High/Low state.**

P1W Register (0x91)

Bit	Field	Type	Initial	Description
7..0	P1nW	R/W	0	0: Disable P1.n wakeup function 1: Enable P1.n wakeup function

9 Interrupt

The MCU provides 10 interrupt sources (2 external and 8 internal) with 4 priority levels. Each interrupt source includes one or more interrupt request flag(s). When interrupt event occurs, the associated interrupt flag is set to logic 1. If both interrupt enable bit and global interrupt (EAL=1) are enabled, the interrupt request is generated and interrupt service routine (ISR) will be started. Most interrupt request flags must be cleared by software. However, some interrupt request flags can be cleared by hardware automatically. In the end, ISR is finished after complete the RETI instruction. The summary of interrupt source, interrupt vector, priority order and control bit are shown as the table below.

Interrupt	Enable Interrupt	Request (IRQ)	IRQ Clearance	Priority / Vector
System Reset	-	-	-	0 / 0x0000
INT0	EX0	IE0	Automatically	1 / 0x0003
INT1	EX1	IE1	Automatically	2 / 0x000B
SAR ADC	ESAR	SARF	Automatically	3 / 0x008B
TC0	ETC0	TCF0	Automatically	4 / 0x0013
T0	ET0	TF0	Automatically	5 / 0x0093
TC1	ETC1	TCF1	Automatically	6 / 0x001B
TC2	ETC2	TCF2	Automatically	7 / 0x009B
UART-RX	ERS0	RI0	By firmware	8 / 0x0023
SPI	ESPI	SPIF / MODF	By firmware	9 / 0x00A3
UART-TX	ETS0	TI0	By firmware	10 / 0x0063
I2C	EI2C	SI	By firmware	11 / 0x00AB

9.1 Interrupt Operation

Interrupt operation is controlled by interrupt request flag and interrupt enable bits. Interrupt request flag is interrupt source event indicator, no matter what interrupt function status (enable or disable). Both interrupt enable bit and global interrupt (EAL=1) are enabled, the system executes interrupt operation when each of interrupt request flags actives. The program counter points to interrupt vector (0x03 – 0xEB) and execute ISR.

9.2 Interrupt Priority

Each interrupt source has its specific default priority order. If two interrupts occurs simultaneously, the higher priority ISR will be service first. The lower priority ISR will be serviced after the higher priority ISR completes. The next ISR will be service after the previous ISR complete, no matter the priority order.

For special priority needs, 4-level priority levels (Level 0 – Level 3) are used. All interrupt sources are classified into 6 priority groups (Group0 – Group5). Each group can be set one specific priority level. Priority level is selected by IP0/IP1 registers. Level 3 is the highest priority and Level 0 is the lowest. The interrupt sources inside the same group will share the same priority level. With the same priority level, the priority rule follows default priority.

Priority Level	IP1.x	IP0.x
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

The ISR with the higher priority level can be serviced first; even can break the on-going ISR with the lower priority level. The ISR with the lower priority level will be pending until the ISR with the higher priority level completes.

Group	Interrupt Source			
Group 0	INT0	-	-	-
Group 1	INT1	SAR ADC	-	-
Group 2	TC0	T0	-	-
Group 3	TC1	TC2	-	-
Group 4	UART-RX	SPI	-	UART-TX
Group 5	-	I2C	-	-

IP0, IP1 Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IP0	-	-	IP05	IP04	IP03	IP02	IP01	IP00
IP1	-	-	IP15	IP14	IP13	IP12	IP11	IP10

IP0 Register (0XA9)

Bit	Field	Type	Initial	Description
5..0	IP0[5:0]	R/W	0	Interrupt priority. Each bit together with corresponding bit from IP1 register specifies the priority level of the respective interrupt priority group.
Else	Reserved	R	0	

IP1 Register (0XB9)

Bit	Field	Type	Initial	Description
5..0	IP1[5:0]	R/W	0	Interrupt priority. Each bit together with corresponding bit from IP0 register specifies the priority level of the respective interrupt priority group.
Else	Reserved	R	0	

***Example**: Priority groups Level GP0>GP1>GP2>GP3=GP4=GP5.

```
IP0 = 0x05;
IP1 = 0x03;
```

***Example**: Priority groups Level GP5>GP4>GP3>GP2=GP1=GP0.

```
IP0 = 0x28;
IP1 = 0x30;
```

9.3 Interrupt Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	ESAR	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
IEN2	-	-	-	-	ETC2	ETC1	ETC0	-
IRCON	-	TCF2	TCF1	TCF0	-	TF0	IE1	IE0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
IRCON2	-	-	-	-	-	-	-	SARF

IEN0 Register (0XA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Enable all interrupt control bit. 0: Disable all interrupt function. 1: Enable all interrupt function.
5	ESAR	R/W	0	SAR ADC interrupt control bit. 0: Disable SAR ADC interrupt function. 1: Enable SAR ADC interrupt function.
4	ETS0	R/W	0	UART TX interrupt control bit. 0: Disable UART TX interrupt function. 1: Enable UART TX interrupt function.
3	ERS0	R/W	0	UART RX interrupt control bit. 0: Disable UART RX interrupt function. 1: Enable UART RX interrupt function.
2	EX1	R/W	0	External P0.1 interrupt (INT1) control bit. 0: Disable INT1 interrupt function. 1: Enable INT1 interrupt function.
1	ET0	R/W	0	T0 timer interrupt control bit. 0: Disable T0 interrupt function. 1: Enable T0 interrupt function
0	EX0	R/W	0	External P0.0 interrupt (INT0) control bit. 0: Disable INT0 interrupt function. 1: Enable INT0 interrupt function.
Else	Reserved	R	0	

IEN1 Register (0XB8)

Bit	Field	Type	Initial	Description
1	ESPI	R/W	0	SPI interrupt control bit 0: Disable SPI interrupt function. 1: Enable SPI interrupt function.
0	EI2C	R/W	0	I2C interrupt control bit. 0: Disable I2C interrupt function. 1: Enable I2C interrupt function.
Else	Reserved	R	0	

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
3	ETC2	R/W	0	TC2 overflow interrupt control bit. 0: Disable TC2 interrupt function. 1: Enable TC2 interrupt function.
2	ETC1	R/W	0	TC1 overflow interrupt control bit. 0: Disable TC1 interrupt function. 1: Enable TC1 interrupt function.
1	ETC0	R/W	0	TC0 overflow interrupt control bit. 0: Disable TC0 interrupt function. 1: Enable TC0 interrupt function.
Else	Reserved	R	0	

IRCON Register (0xC0)

Bit	Field	Type	Initial	Description
6	TCF2	R/W	0	TC2 timer interrupt request flag. 0: None TC2 interrupt request. 1: TC2 interrupt request.
5	TCF1	R/W	0	TC1 timer interrupt request flag. 0: None TC1 interrupt request. 1: TC1 interrupt request.
4	TCF0	R/W	0	TC0 timer interrupt request flag. 0: None TC0 interrupt request. 1: TC0 interrupt request.
2	TF0	R/W	0	T0 timer interrupt request flag. 0: None T0 interrupt request. 1: T0 interrupt request.
1	IE1	R/W	0	External P0.1 interrupt (INT1) request flag 0: None INT1 interrupt request. 1: INT1 interrupt request.

0	IE0	R/W	0	External P0.0 interrupt (INT0) request flag 0: None INT0 interrupt request. 1: INT0 interrupt request.
Else	Reserved	R	0	

S0CON Register (0X98)

Bit	Field	Type	Initial	Description
1	TI0	R/W	0	UART transmit interrupt request flag. It indicates completion of a serial transmission at UART. It is set by hardware at the end of bit 8 in mode 0 or at the beginning of a stop bit in other modes. It must be cleared by software. 0: None UART transmit interrupt request. 1: UART transmit interrupt request.
0	RI0	R/W	0	UART receive interrupt request flag. It is set by hardware after completion of a serial reception at UART. It is set by hardware at the end of bit 8 in mode 0 or in the middle of a stop bit in other modes. It must be cleared by software. 0: None UART receive interrupt request. 1: UART receive interrupt request.
Else				Refer to other chapter(s)

SPSTA Register (0XE1)

Bit	Field	Type	Initial	Description
7	SPIF	R	0	SPI complete communication flag Set automatically at the end of communication Cleared automatically by reading SPSTA, SPDAT registers
4	MODF	R	0	Mode fault flag
Else				Refer to other chapter(s)

I2CCON Register (0XDC)

Bit	Field	Type	Initial	Description
3	SI	R/W	0	Serial interrupt flag The SI is set by hardware when one of 25 out of 26 possible I2C states is entered. The only state that does not set the SI is state F8h, which indicates that no relevant state information is available. The SI flag must be cleared by software. In order to clear the SI bit, '0' must be written to this bit. Writing a '1' to SI bit does not change value of the SI.
Else				Refer to other chapter(s)

IRCON2 Register (0XBF)

Bit	Field	Type	Initial	Description
0	SARF	R/W	0	SAR interrupt request flag. 0: None SAR interrupt request 1: SAR interrupt request.
Else				Refer to other chapter(s)

10 MDU

The multiplication division unit is an on-chip arithmetic co-processor which enables the microcontroller to perform additional extended arithmetic operations. This unit provides 32-bit unsigned division, 16-bit unsigned multiplication, shift and normalize operations. These operations are identified by the different sequences of writing MD0 to MD5 registers.

10.1 Multiplication (16-bit x 16-bit)

The elements of a multiplication include three parts: multiplicand, multiplier and product. To start a multiplication requires following writing sequence: MD0 (low byte of multiplicand), MD4 (low byte of multiplier), MD1 (high byte of multiplicand), and MD5 (high byte of multiplier).

By the end of writing MD5 register, the multiplication is automatically started and takes 11 CPU cycles for its operation. The product of this term operation would be available to read by a specific sequence: MD0 (LSB), MD1, MD2, and MD3 (MSB) registers.

10.2 Division (32-bit/16-bit and 16-bit/16-bit)

The MDU supports two kind of division: 32-bit by 16-bit, and 16-bit by 16-bit. The first operation takes 17 CPU cycles to compute, whereas the second one takes 9 cycles only.

A 32-bit division started by a specific sequence of writing registers: MD0, MD1, MD2, MD3, MD4, and MD5. In this case, the 32-bit dividend is expected to store in MD3 (most significant bit) to MD0 registers, and 16-bit divisor is stored in MD5 and MD4 registers (MSB in MD5 register).

A 16-bit division operation cooperates with four registers only. The 16-bit dividend is stored in MD1 and MD0 registers, and the 16-bit divisor is stored in MD5 and MD4 registers (MD1 and MD5 for most signification bit). The appropriate performing sequence is ‘MD0, MD1, MD4, and MD5.’

The MDU starts computing from MD5 register is written. It spends 9 or 17 CPU cycles, depends on the length of dividend, before the outcome is generated. The quotient is stored in MD3 to MD0 registers for 32-bit division, and MD1 to MD0 registers for 16-bit division (LSB in MD0 register). The reminder would be placed in MD5 (MSB) and MD4 registers no matter which division is performed. However, reading MD5 register must be the last operation to indicate the full division is completed.

10.3 Shifting and Normalizing

The shifting and normalizing operations rotate the 32-bit registers (MD3 to MD0, MSB in MD3) for a certain or uncertain time.

In shift operation, the 32-bit unsigned integer is shifted left or right by a specified number of bits. The direction and shifting number is specified in ARCON register. A shift operation takes 3 to 18

CPU cycles depends on the shift time.

In normalizing operation, the 32-bit unsigned integer would be shifted left repeatedly until the most significant bit (7th bit of MD3 register) is 1. A normalizing operation takes 4 to 19 CPU cycles depends on the actual shift time.

Both shifting and normalizing operations are started by proper sequence of writing registers: MD0, MD1, MD2, MD3, and finally ARCON register. The result would be place in MD0 to MD3 registers which should be read in the sequence of MD0, MD1, MD2, and MD3.

10.4 Cooperate with Keil C51

Because Keil C51 supports both of hardware and software multiplication/division operators, a command line '#pragma mdu_r515' is required in C to enable the hardware MDU functionality for higher performance. Subsequently, Keil C51 would compile mathematic operators with MDU support.

```
1 #include <SN8F5849.H>
2 #pragma  mdu_r515 //Keil C51 MDU command line
```

10.5 The Error Flag (MDEF)

The "MDEF" error flag indicates an improperly performed operation (when one of the arithmetic operations has been restarted or interrupted by a new operation). The error flag mechanism is automatically enabled with the first write operation to "MD0" and disabled with the final read instruction from "MD3" (multiplication or shift/normalize) or "MD5" (division) in phase three.

The error flag is set when:
There is a write access to 'MDx' registers (any of 'MD0' to 'MD5' and ARCON) during phase two of MDU operation (restart or calculations interrupting)
There is a read access to one of MDx registers during phase two of MDU operation when the error flag mechanism is enabled. In such condition error flag is set but the calculation is not interrupted.
The error flag is reset only after read access to "ARCON" register. The error flag is read only.

10.6 The Overflow Flag (MDOV)

The MDOV overflow flag is set when one of the following conditions occurs:

Division by zero
Multiplication with a result greater than 0000 FFFFh
Start of normalizing if the most significant bit of MD3 is set (MD3.7= 1)
Any operation of the MDU that does not match the above conditions clears the overflow flag. Note that the overflow flag is exclusively controlled by hardware. It cannot be written.

10.7 MDU Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MD0	MD07	MD06	MD05	MD04	MD03	MD02	MD01	MD00
MD1	MD17	MD16	MD15	MD14	MD13	MD12	MD11	MD10
MD2	MD27	MD26	MD25	MD24	MD23	MD22	MD21	MD20
MD3	MD37	MD36	MD35	MD34	MD33	MD32	MD31	MD30
MD4	MD47	MD46	MD45	MD44	MD43	MD42	MD41	MD40
MD5	MD57	MD56	MD55	MD54	MD53	MD52	MD51	MD50
ARCON	MDEF	MDOV	SLR	SC4	SC3	SC2	SC1	SC0

MD Registers (MD0 – MD5: 0xE9 – 0xEE)

Bit	Field	Type	Initial	Description
7..0	MD[7:0]	R/W	0x00	Multiplication/Division Registers

ARCON Register (0xEF)

Bit	Field	Type	Initial	Description
7	MDEF	R/W	0	MDU error flag MDEF Indicates an improperly performed operation (when one of the arithmetic operations has been restarted or interrupted by a new operation).
6	MDOV	R/W	0	MDU overflow flag Overflow occurrence in the MDU operation.
5	SLR	R/W	0	Shift direction 0: Shift left operation 1: Shift right operation
4..0	SC[4:0]	R/W	0x00	Shift counter Write 0x00: Perform normalizing. The actual shift time would be readable after operation. Write else values: Specify the times of shift operation.

10.8 Sample Code

The following sample code demonstrates how to perform MDU 32 bit / 16 bit.

```
#include <SN8F5849.H>
#pragma  mdu_r515 //Keil C51 MDU command line

void main(void)
{
 unsigned int  Divisor; // 16-bit divisor
 unsigned long Dividend;     // 32-bit dividend
 unsigned long Quotient;     // 32-bit Quotient
 unsigned  int  Remainder;    // 16-bit Remainder

 Divisor = 0x1234; Dividend =
 0x56789ABC;
 Quotient = Dividend  / Divisor;     //0x0004C016
 Remainder = Dividend  % Divisor;
                     //0x0A44

 while(1);
}
```

11 GPIO

The microcontroller has up to 44 bidirectional general purpose I/O pin (GPIO). Unlike the original 8051 only has open-drain output, SN8F5849 builds in push-pull output structure to improve its driving performance.

11.1 Input and Output Control

The input and output direction control is configurable through P0M to P5M registers. These bits specify each pin that is either input mode or output mode.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
P3M	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
P1OC	-	P23OC	P22OC	P05OC	P04OC	P17OC	P16OC	P15OC

P0M: 0xF9, P1M: 0xFA, P2M: 0xFB, P3M: 0xFC, P4M: 0xFD, P5M:0xFE

Bit	Field	Type	Initial	Description
7	P07M	R/W	0	Mode selection of P0.7 0: Input mode 1: Output mode
6	P06M	R/W	0	Mode selection of P0.6 0: Input mode 1: Output mode
5	P05M	R/W	0	Mode selection of P0.5 0: Input mode 1: Output mode
4..0				et cetera

P1OC Register (0xE4)

Bit	Field	Type	Initial	Description
1	P23OC	R/W	0	P2.3 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
2	P22OC	R/W	0	P2.2 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
4	P05OC	R/W	0	P0.5 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
3	P04OC	R/W	0	P0.4 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
2	P17OC	R/W	0	P1.7 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
1	P16OC	R/W	0	P1.6 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
0	P15OC	R/W	0	P1.5 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode

11.2 Input Data and Output Data

By a read operation from any registers of P0 to P5, the current pin's logic level would be fetch to represent its external status. This operation remains functional even the pin is shared with other function like UART and I2C which can monitor the bus condition in some case.

A write P0 to P5 register value would be latched immediately, yet the value would be outputted until the mapped P0M – P5M is set to output mode. If the pin is currently in output mode, any value set to P0 to P5 register would be presented on the pin immediately.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
P1	P17	P16	P15	P14	P13	P12	P11	P10
P2	P27	P26	P25	P24	P23	P22	P21	P20
P3	P37	P36	P35	P34	P33	P32	P31	P30
P4	P47	P46	P45	P44	P43	P42	P41	P40
P5	P57	P56	P55	P54	P53	P52	P51	P50

P0: 0x80, P1: 0x90, P2: 0xA0, P3: 0xFC, P4: 0xE8, P5: 0xF8

Bit	Field	Type	Initial	Description
7	P07	R/W	1	Read: P0.7 pin's logic level Write 1/0: Output logic high or low (applied if P07M = 1)
6	P06	R/W	1	Read: P0.6 pin's logic level Write 1/0: Output logic high or low (applied if P06M = 1)
5	P05	R/W	1	Read: P0.5 pin's logic level Write 1/0: Output logic high or low (applied if P05M = 1)
4..0				et cetera

11.3 On-chip Pull-up Resisters

The P0UR to P5UR registers are mapped to each pins internal 200 kΩ (in typical value) pull-up resister.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07UR	P06UR	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR
P1UR	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR
P2UR	P27UR	P26UR	P25UR	P24UR	P23UR	P22UR	P21UR	P20UR
P3UR	P37UR	P36UR	P35UR	P34UR	P33UR	P32UR	P31UR	P30UR
P4UR	P47UR	P46UR	P45UR	P44UR	P43UR	P42UR	P41UR	P40UR
P5UR	P57UR	P56UR	P55UR	P54UR	P53UR	P52UR	P51UR	P50UR

P0UR: 0xF1, P1UR: 0xF2, P2UR: 0xF3, P3UR: 0xF4, P4UR: 0xF5, P5UR: 0xF6

Bit	Field	Type	Initial	Description
7	P07UR	R/W	0	On-chip pull-up resister control of P0.7 0: Disable* 1: Enable
6	P06UR	R/W	0	On-chip pull-up resister control of P0.6 0: Disable* 1: Enable
5	P05UR	R/W	0	On-chip pull-up resister control of P0.5 0: Disable* 1: Enable
4..0				et cetera

* Recommended disable pull-up resister if the pin is output mode or analog function

11.4 Pin Shared with LCD Function

The microcontroller builds in LCD functions. The LCD driver output pins share with GPIO function, which can be configured by setting PxCON registers.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5CON	P5CON7	P5CON6	P5CON5	P5CON4	P5CON3	P5CON2	P5CON1	P5CON0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
P3CON	P3CON7	P3CON6	P3CON5	P3CON4	P3CON3	P3CON2	P3CON1	P3CON0
P2CON	P2CON7	P2CON6	P2CON5	P2CON4	P2CON3	P2CON2	P2CON1	P1CON0
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	P1CON3	P1CON2	-	-

11.5 P1CON: 0x9C, P2CON: 0x9B, P3CON: 0x9E, P4CON: 0x9F, P5CON: 0x9D

Bit	Field	Type	Initial	Description
7	P3CON7	R/W	1	P37 function control bit 0: LCD function. 1: GPIO function.
6	P3CON6	R/W	1	P36 function control bit 0: LCD function. 1: GPIO function.
5	P3CON5	R/W	1	P35 function control bit 0: LCD function. 1: GPIO function.
4..0				et cetera

12 External Interrupt

INT0 and INT1 are external interrupt trigger sources. Build in edge trigger configuration function and edge direction is selected by PEDGE register. When both external interrupt (EX0/EX1) and global interrupt (EAL) are enabled, the external interrupt request flag (IE0/IE1) will be set to “1” as edge trigger event occurs. The program counter will jump to the interrupt vector (ORG 0x0003/0x000B) and execute interrupt service routine. Interrupt request flag will be cleared by hardware before ISR is executed.

12.1 External Interrupt Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	-	EX1G1	EX1G0	EX0G1	EX0G0
IEN0	EAL	-	-	ES0	-	EX1	ET0	EX0
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-

PEDGE Register (0X8F)

Bit	Field	Type	Initial	Description
3..2	EX1G[1:0]	R/W	10	External interrupt 1 trigger edge control register. 00: Reserved. 01: Rising edge trigger. 10: Falling edge trigger (default) 11: Both rising and falling edge trigger
1..0	EX0G[1:0]	R/W	10	External interrupt 0 trigger edge control register. 00: Reserved. 01: Rising edge trigger. 10: Falling edge trigger (default) 11: Both rising and falling edge trigger
Else	Reserved	R	0	

12.2 Sample Code

The following sample code demonstrates how to perform INT0/INT1 with interrupt.

```
#include <intrins.h>
#include <SN8F5849.h>

void Init_GPIO(void);
void Init_ISR(void);
void Delay1ms(unsigned int n);

void main(void)
{
   Init_GPIO();          // Initial GPIO
   Init_ISR();           // Initial interrupt setting

   while (1) {
       WDTR = 0x5A;     // clear watchdog if watchdog enable
       // To Do...
       Delay1ms(100);
   }
}

void Init_GPIO(void)
{
   P0 = 0x00;
   P0UR = 0xFF;
   P0M = 0x00;          // P0.0, P0.1 as input mode
}

void Init_ISR(void)
{
   PEDGE &= 0x00;       // Clear PEDGE
   PEDGE |= 0x02;       // EX0G = 0x10 : INT0 Falling edge trigger (default).
   EX0 = 1;             // INT0 isr enable

   PEDGE |= 0x04;       // EX1G = 0x01 : INT1 Rising edge trigger
   EX1 = 1;             // INT1 isr enable
   EAL = 1;             // Interrupt enable
}

void INT0_ISR(void) interrupt ISRInt0 // Vector @0x03
{
   // cleared int0 flag by hardware
   // IE0 = 0;          // Clear INT0 flag

  // To Do...
  _nop_();
}
```

```
55
56 void INT1_ISR(void) interrupt ISRInt1   // Vector @  0x0B
57 {
58   // cleared int1 flag by hardware
59   // IE1 = 0;          // Clear INT1 flag
60
61   // To Do...
62    _nop_();
63 }
64
65 void Delay1ms(unsigned int n)
66 {
67   unsigned int i, j;
68   // int value
69   i = 0;
70   j = 0;
71
72   for (i=0; i<n; i++) {
73        for (j=0; j<220; j++) {
74             _nop_();     _nop_();
75             _nop_();     _nop_();
76             _nop_();     _nop_();
77             _nop_();     _nop_();
78        }
79    }
80 }
81
82
83
```

13 TCx 16-BIT Timer/Counter

The TC0/TC1/TC2 timer is an 16-bit binary up timer with basic timer, event counter and PWM functions. The basic timer function supports flag indicator (TCxIRQ bit) and interrupt operation (interrupt vector). The interval time is programmable through TCxM, TCxC, TCxR registers. The event counter is changing TC0 clock source from system clock (Fcpu/Fosc/X'tal/32K) to external clock like signal (e.g. 32768Hz Crystal). TCx becomes a counter to count external clock number to implement measure application. TCx also builds in duty/cycle programmable PWM. The PWM cycle and resolution are controlled by TCx timer clock rate, TCxR and TCxD registers, so the PWM with good flexibility to implement IR carry signal, motor control and brightness adjuster…The main purposes of the TCx 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:**
  TCx timer function supports interrupt function. When TCx timer occurs overflow, the TCxIRQ actives 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 TCxR and TCxD registers.
- **Idle mode function:**
  All TCx functions (timer, PWM, event counter, auto-reload) works on normal or idle mode, stop working in STOP mode.

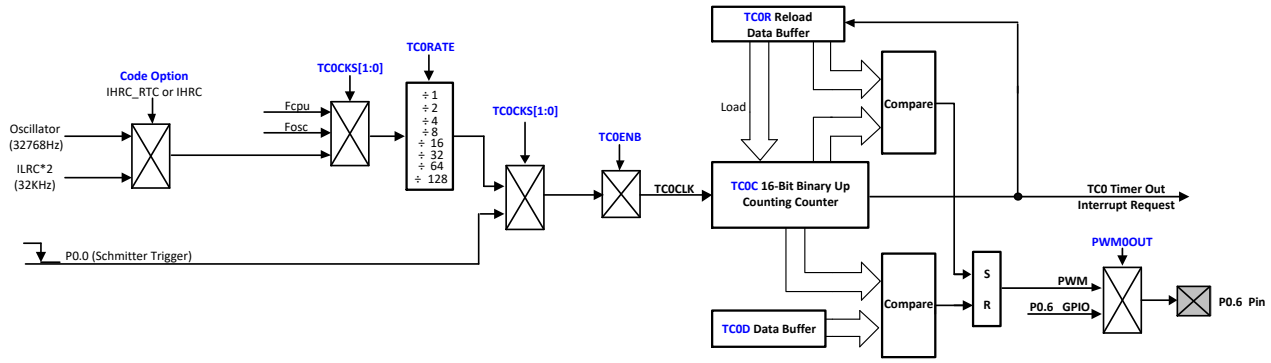
Note:

Register	Function description
TCx = TC0, TC1 or TC2.	Timer Counter
TCxC = TC0C, TC1C or TC2C	Timer Counter Counting data
TCxD = TCF0, TCF1 or TCF2	Timer PWM function duty data
TCxR = TCF0, TCF1 or TCF2	Timer Auto reload data
TCxENB = TC0ENB, TC1ENB or TC2ENB	Timer function enable bit
TCxM = TC0M, TC1M or TC2M	Timer control register
ETCx = ETC0, ETC1 or ETC2	Timer overflow interrupt control bit.
TCFx = TCF0, TCF1 or TCF2	Timer Overflow interrupt request flag.
TCxRATE = TC0RATE, TC1RATE or TC2 RATE	Timer Clock divider
x = 0, 1 or 2	

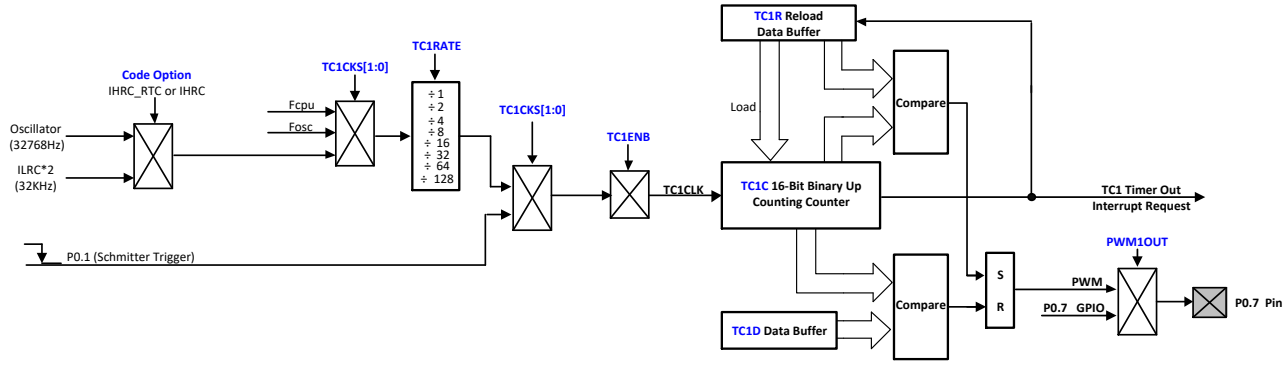
13.1 Timer Counter Diagram and Clock Source Selection

The figures below illustrate the clock selection circuit of TC0, TC1 and TC2. Each has three internal clock sources selection (fcpu, fosc, and 32K/X'tal) and one external signal input. The following block diagrams show the TCx internal structure.

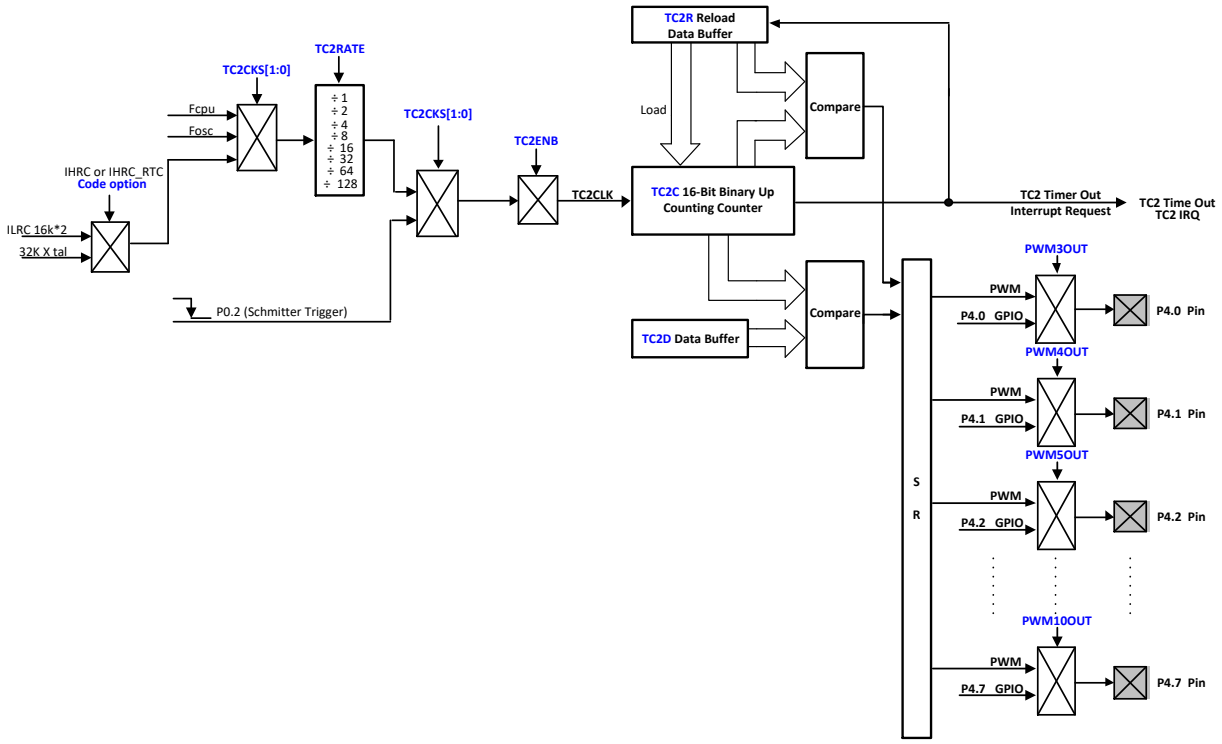
TC0 Structure Diagram:



TC1 Structure Diagram:



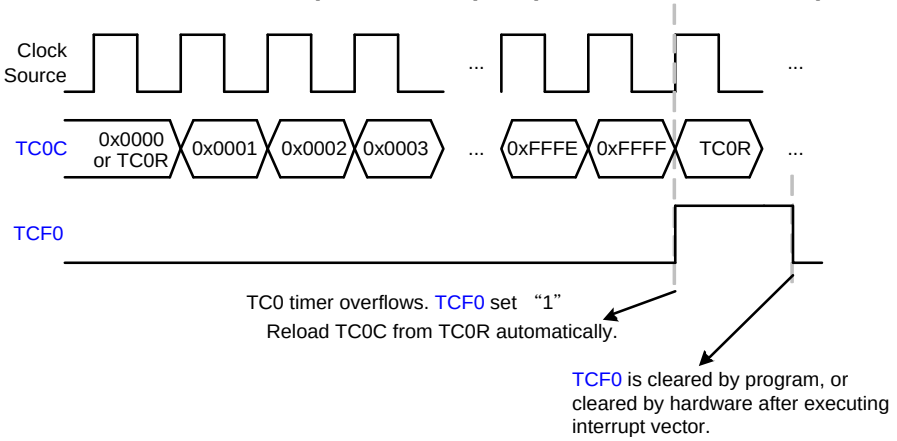
TC2 Structure Diagram:



13.2 TC0 Timer Operation (include TC1 and TC2)

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 TCF0 set as “1”. The interrupt flag TCF0 is clear by program or auto clear by hardware after executing interrupt vector. TCx 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 (ETC0=1), the program counter is pointed to interrupt vector to execute interrupt service routine after TC0 timer overflow occurrence.

TC0 interval time = (65536-TC0R) * 1/($F_{TC0\ clock\ Source}$/TC0Rate)



13.3 TC0 Counting Function and Register (include TC1 and TC2)

TC0C is TC0 16-bit counter. When TC0C overflow occurs, the TCF0 flag is set as “1” and cleared by hardware or firmware. 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.

TC0CH Register (TC0CH : 0xC5)

Bit	Field	Type	Initial	Description
7..0	TC0CH	R/W	0x00	High byte of TC0 counter

TC0CL Register (TC0CL : 0xC4)

Bit	Field	Type	Initial	Description
7..0	TC0CL	R/W	0x00	Low byte of TC0 counter

The equation of TC0C initial value is as following:

TC0C initial value = 65536 - (TC0 interrupt interval time * TC0 clock rate)

13.4 TC0 Auto Reload Function and Register (include TC1 and TC2)

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 1st 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.

TC0RH Register (TC0RH : 0xC3)

Bit	Field	Type	Initial	Description
7..0	TC0RH	R/W	0x00	High byte of TC0 reload buffer

TC0RL Register (TC0RL : 0xC2)

Bit	Field	Type	Initial	Description
7..0	TC0RL	R/W	0x00	Low byte of TC0 reload buffer

The equation of TC0R initial value is as following:

TC0R initial value = 65536 - (TC0 interrupt interval time * TC0 clock rate)

Example: To calculation TC0C and TC0R value to obtain 10ms TC0 interval time.

TC0 clock source is Fcpu = 32MHz/32 = 1MHz. Select TC0RATE=000 (Fcpu/128)

TC0 interval time = 10ms. TC0 clock rate = 32MHz/32/128

TC0C/TC0R initial value = 65536 - (TC0 interval time * input clock)

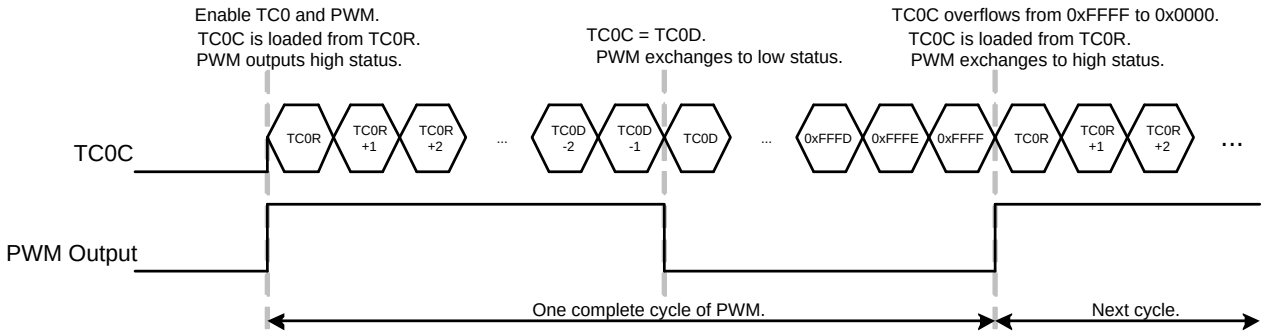
= 65536 - (10ms * 32MHz / 32 / 128)

= 65536 - ($10^{-2} * 32 * 10^{6}$ / 32 / 128)

= FFB2H

13.5 TC0 (TC1 / TC2) PWM Function

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.



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.

The equation of TC0D initial value is as following:

TC0D initial value = TC0R + (PWM high pulse width period / TC0 clock rate)

Example: To calculate TC0D value to obtain 1/3 duty PWM signal in 100Hz.

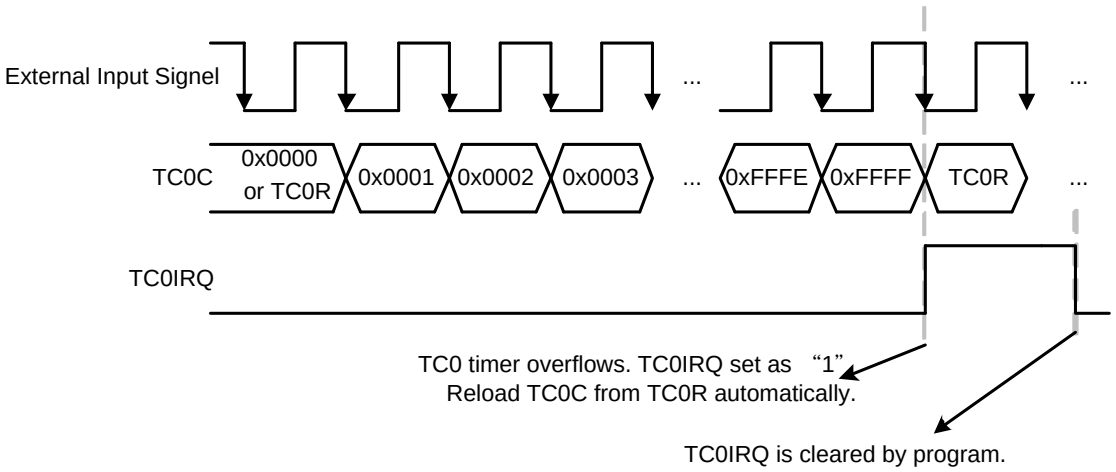
The TC0clock source is Fcpu = 32MHz/32= 1MHz. Select TC0RATE=000 (Fcpu/128).

TC0R = FFB2H (TC0RH: FFH,TC0RL: B2H). TC0 interval time = 10ms. So the PWM cycle is 100Hz. In 1/3 duty condition, the high pulse width is about 3.33ms.

TC0D initial value = FFB2H + (PWM high pulse width period / TC0 clock rate)
= FFB2H + (3.33ms * 32MHz / 32 / 128)
= FFB2H + 1AH = FFCCH

13.6 TC0 Event Counter Function (include TC1 and TC2)

TC0 event counter is set the TC0 clock source from external input pin (P0.0). When TC0CKS1=1, TC0 clock source is switch to external input pin (P0.0). 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.



Note: TC0 External Input signal = P00
TC1 External Input signal = P01
TC2 External Input signal = P02

13.7 TC0, TC1 and TC2 Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS1	TC0CKS0	PWM0OUT	TC0GN
TC0RL	TC0RL7	TC0RL6	TC0RL5	TC0RL4	TC0RL3	TC0RL2	TC0RL1	TC0RL0
TC0RH	TC0RH7	TC0RH6	TC0RH5	TC0RH4	TC0RH3	TC0RH2	TC0RH1	TC0RH0
TC0CL	TC0CL7	TC0CL6	TC0CL5	TC0CL4	TC0CL3	TC0CL2	TC0CL1	TC0CL0
TC0CH	TC0CH7	TC0CH6	TC0CH5	TC0CH4	TC0CH3	TC0CH2	TC0CH1	TC0CH0
TC0DL	TC0DL7	TC0DL6	TC0DL5	TC0DL4	TC0DL3	TC0DL2	TC0DL1	TC0DL0
TC0DH	TC0DH7	TC0DH6	TC0DH5	TC0DH4	TC0DH3	TC0DH2	TC0DH1	TC0DH0
TC1M	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS1	TC1CKS0	PWM1OUT	-
TC1RL	TC1RL7	TC1RL6	TC1RL5	TC1RL4	TC1RL3	TC1RL2	TC1RL1	TC1RL0
TC1RH	TC1RH7	TC1RH6	TC1RH5	TC1RH4	TC1RH3	TC1RH2	TC1RH1	TC1RH0
TC1CL	TC1CL7	TC1CL6	TC1CL5	TC1CL4	TC1CL3	TC1CL2	TC1CL1	TC1CL0
TC1CH	TC1CH7	TC1CH6	TC1CH5	TC1CH4	TC1CH3	TC1CH2	TC1CH1	TC1CH0
TC1DL	TC1DL7	TC1DL6	TC1DL5	TC1DL4	TC1DL3	TC1DL2	TC1DL1	TC1DL0
TC1DH	TC1DH7	TC1DH6	TC1DH5	TC1DH4	TC1DH3	TC1DH2	TC1DH1	TC1DH0
TC2M	TC2ENB	TC2RATE0	TC2RATE1	TC2RATE0	TC2CKS1	TC2CKS0	PWM2OUT	-
TC2RL	TC2RL7	TC2RL6	TC2RL5	TC2RL4	TC2RL3	TC2RL2	TC2RL1	TC2RL0
TC2RH	TC2RH7	TC2RH6	TC2RH5	TC2RH4	TC2RH3	TC2RH2	TC2RH1	TC2RH0
TC2CL	TC2CL7	TC2CL6	TC2CL5	TC2CL4	TC2CL3	TC2CL2	TC2CL1	TC2CL0
TC2CH	TC2CH7	TC2CH6	TC2CH5	TC2CH4	TC2CH3	TC2CH2	TC2CH1	TC2CH0
TC2DL	TC2DL7	TC2DL6	TC2DL5	TC2DL4	TC2DL3	TC2DL2	TC2DL1	TC2DL0
TC2DH	TC2DH7	TC2DH6	TC2DH5	TC2DH4	TC2DH3	TC2DH2	TC2DH1	TC2DH0
PWMOUT	PWM10OUT	PWM9OUT	PWM8OUT	PWM7OUT	PWM6OUT	PWM5OUT	PWM4OUT	PWM3OUT

TC0M Register (0xC1)

Bit	Field	Type	Initial	Description
7	TC0ENB	R/W	0	TC0 function control bit. 0: Disable 1: Enable
6..4	TC0RATE[2:0]	R/W	0	TC0 timer clock divider. 000: clock/128 100: clock/8 001: clock/64 101: clock/4 010: clock/32 110: clock/2 011: clock/16 111: clock/1
3..2	TC0CKS[1:0]	R/W	0	TC0 timer clock source select bits 00: Fcpu 01: Fosc 10: Internal 32K or External 32K crystal 11: P0.0 (event counter)
1	PWM0OUT	R/W	0	PWM0 output control bit 0: Disable, P0.6 is GPIO mode. 1: Enable, P0.6 output PWM signal.
0	TC0GN	R/W	0	0: Disable TC0 wakeup STOP mode function 1: Enable TC0 wakeup STOP mode function

TC1M Register (0xC9)

Bit	Field	Type	Initial	Description
7	TC1ENB	R/W	0	TC1 function control bit. 0: Disable 1: Enable
6..4	TC1RATE[2:0]	R/W	0	TC1 timer clock divider. 000: clock/128 100: clock/8 001: clock/64 101: clock/4 010: clock/32 110: clock/2 011: clock/16 111: clock/1
3..2	TC1CKS[1:0]	R/W	0	TC1 timer clock source select bits 00: Fcpu 01: Fosc 10: Internal 32K or External 32K crystal 11: P0.1 (event counter)
1	PWM1OUT	R/W	0	PWM1 output control bit 0: Disable, P0.7 is GPIO mode. 1: Enable, P0.7 output PWM signal.
0	Reserved	R	0	

TC2M Register (0xA1)

Bit	Field	Type	Initial	Description
7	TC2ENB	R/W	0	TC2 function control bit. 0: Disable 1: Enable
6..4	TC2RATE[2:0]	R/W	0	TC2 timer clock divider. 000: clock/128 100: clock/8 001: clock/64 101: clock/4 010: clock/32 110: clock/2 011: clock/16 111: clock/1
3..2	TC2CKS[1:0]	R/W	0	TC2 timer clock source select bits 00: Fcpu 01: Fosc 10: Internal 32K or External 32K crystal 11: P0.2 (event counter)
1	PWM2OUT	R/W	0	PWM2 output control bit 0: Disable, P1.1 is GPIO mode. 1: Enable, P1.1 output PWM signal.
0	Reserved	R	0	

Timer clock control table:

TCxCKS[1:0]	TCxRATE[2:0]	TCx Clock	TCxCKS[1:0]	TxRATE[2:0]	TCx Clock
00	000	Fcpu / 128	01	000	Fosc / 128
	001	Fcpu / 64		001	Fosc / 64
	010	Fcpu / 32		010	Fosc / 32
	011	Fcpu / 16		011	Fosc / 16
	100	Fcpu / 8		100	Fosc / 8
	101	Fcpu / 4		101	Fosc / 4
	110	Fcpu / 2		110	Fosc / 2
	111	Fcpu / 1		111	Fosc / 1
10	000	32KHz / 128	11	NA	P0.0 event counter function.
	001	32KHz / 64			
	010	32KHz / 32			
	011	32KHz / 16			
	100	32KHz / 8			
	101	32KHz / 4			
	110	32KHz / 2			
	111	32KHz / 1			

Note: TCx = TC0, TC1 and TC2

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
4	EADC	R/W	0	ADC interrupt control bit 0: Disable ADC interrupt function. 1: Enable ADC interrupt function.
3	ETC2	R/W	0	TC2 overflow interrupt control bit. 0: Disable TC2 interrupt function. 1: Enable TC2 interrupt function.
2	ETC1	R/W	0	TC1 overflow interrupt control bit. 0: Disable TC1 interrupt function. 1: Enable TC1 interrupt function.
1	ETC0	R/W	0	TC0 overflow interrupt control bit. 0: Disable TC0 interrupt function. 1: Enable TC0 interrupt function.
Else	Reserved	R	0	

IRCON Register (0xC0)

Bit	Field	Type	Initial	Description
7	ADCF	R/W	0	ADC interrupt request flag. 0: None ADC interrupt request 1: ADC interrupt request.
6	TCF2	R/W	0	TC2 timer interrupt request flag. 0: None TC2 interrupt request. 1: TC2 interrupt request.
5	TCF1	R/W	0	TC1 timer interrupt request flag. 0: None TC1 interrupt request. 1: TC1 interrupt request.
4	TCF0	R/W	0	TC0 timer interrupt request flag. 0: None TC0 interrupt request. 1: TC0 interrupt request.
2	TF0	R/W	0	T0 timer interrupt request flag. 0: None T0 interrupt request. 1: T0 interrupt request.
1	IE1	R/W	0	External P0.1 interrupt (INT1) request flag 0: None INT1 interrupt request. 1: INT1 interrupt request.
0	IE0	R/W	0	External P0.0 interrupt (INT0) request flag 0: None INT0 interrupt request. 1: INT0 interrupt request.
Else	Reserved	R	0	

TC0CH / TC1CH / TC2CH Registers (TC0CH : 0xC5, TC1CH : 0xCD, TC2CH : 0xA5)

Bit	Field	Type	Initial	Description
7..0	TCxCH	R/W	0x00	High byte of TCx counter

TC0CL / TC1CL / TC2CL Registers (TC0CL : 0xC4, TC1CL : 0xCC, TC2CL : 0xA4)

Bit	Field	Type	Initial	Description
7..0	TCxCL	R/W	0x00	Low byte of TCx counter

TC0RH / TC1RH / TC2RH Registers (TC0RH : 0xC3, TC1RH : 0xCB, TC2RH : 0xA3)

Bit	Field	Type	Initial	Description
7..0	TCxRH	R/W	0x00	High byte of TCx reload buffer

TC0RL / TC1RL / TC2RL Registers (TC0RL : 0xC2, TC1RL : 0xCA, TC2RL : 0xA2)

Bit	Field	Type	Initial	Description
7..0	TCxRL	R/W	0x00	Low byte of TCx reload buffer

TC0DH / TC1DH / TC2DH Registers (TC0DH : 0xC7, TC1DH : 0xCF, TC2DH : 0xA7)

Bit	Field	Type	Initial	Description
7..0	TCxDH	R/W	0x00	High byte of TCx duty control

TC0DL / TC1DL / TC2DL Registers (TC0DL : 0xC6, TC1DL : 0xCE, TC2DL : 0xA6)

Bit	Field	Type	Initial	Description
7..0	TCxDL	R/W	0x00	Low byte of TCx duty control

PWMOUTM Register (0xD5)

Bit	Field	Type	Initial	Description
0	PWM3OUT	R/W	0	PWM3 output control bit 0: Disable, P4.0 is GPIO mode. 1: Enable, P4.0 output PWM3 signal.
1	PWM4OUT	R/W	0	PWM4 output control bit 0: Disable, P4.1 is GPIO mode. 1: Enable, P4.1 output PWM4 signal.
2	PWM5OUT	R/W	0	PWM5 output control bit 0: Disable, P4.2 is GPIO mode. 1: Enable, P4.2 output PWM5 signal.
3	PWM6OUT	R/W	0	PWM6 output control bit 0: Disable, P4.3 is GPIO mode. 1: Enable, P4.3 output PWM6 signal.

4	PWM7OUT	R/W	0	PWM7 output control bit 0: Disable, P4.4 is GPIO mode. 1: Enable, P4.4 output PWM7 signal.
5	PWM8OUT	R/W	0	PWM8 output control bit 0: Disable, P4.5 is GPIO mode. 1: Enable, P4.5 output PWM8 signal.
6	PWM9OUT	R/W	0	PWM9 output control bit 0: Disable, P4.6 is GPIO mode. 1: Enable, P4.6 output PWM9 signal.
7	PWM10OUT	R/W	0	PWM10 output control bit 0: Disable, P4.7 is GPIO mode. 1: Enable, P4.7 output PWM10 signal.

13.8 Sample Code

The following sample code demonstrates how to perform TC0/TC1/TC2 with interrupt.

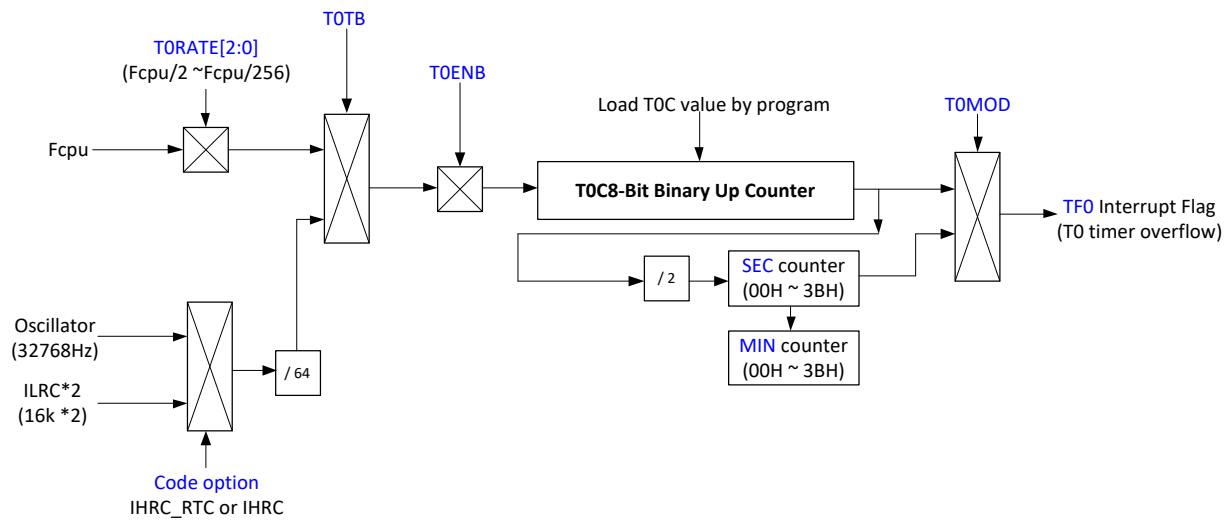
```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>

void TC0_Init(void);
void GPIO_Init(void);

void Init_SysCLK(void)
{

   /* Clock Switch Select Register */
   CLKSEL = 0x02;         // Fcpu = Fhosc/32
   CLKCMD = 0x69;         // clock switch start
}

void main(void)
{
   Init_SysCLK();
   GPIO_Init();
   TC0_Init();

   while (1) {
       WDTR = 0x5A;       // clear watchdog if watchdog enable

       // To Do...
   }
}

void GPIO_Init(void)
{
   P0M |= 0xFF;           // P0 = Output Mode
}

void TC0_Init(void)
{
   TC0M = 0X00;           // TC0 Clock = fcpu/128

   /* TC0C/TC0R initial value =65536-(10ms*32MHz/32/128) */
   TC0RL = 0XB2;          //AUTO-Reload Register
   TC0RH = 0XFF;

   TC0CL = 0XB2;          //TC0 COUNTING Register
   TC0CH = 0XFF;

   IEN2 |= 0x02;          // TC0IEN = 1
   TC0M |= 0X80;          // TC0ENB = 1
   IEN0 |= 0x80;          // EAL = 1
}
```

```
55 void TC0_ISR(void) interrupt ISRTC0 // Vector @  0x13
56 {
57   // cleared TC0 interrupt flag by hardware
58   // TCF0 = 0;           // Clear TCF0 flag
59
60   // To Do...
61   P0 ^= 0XFF;            // P0 Toggle
62 }
63
```

14 Timer 0

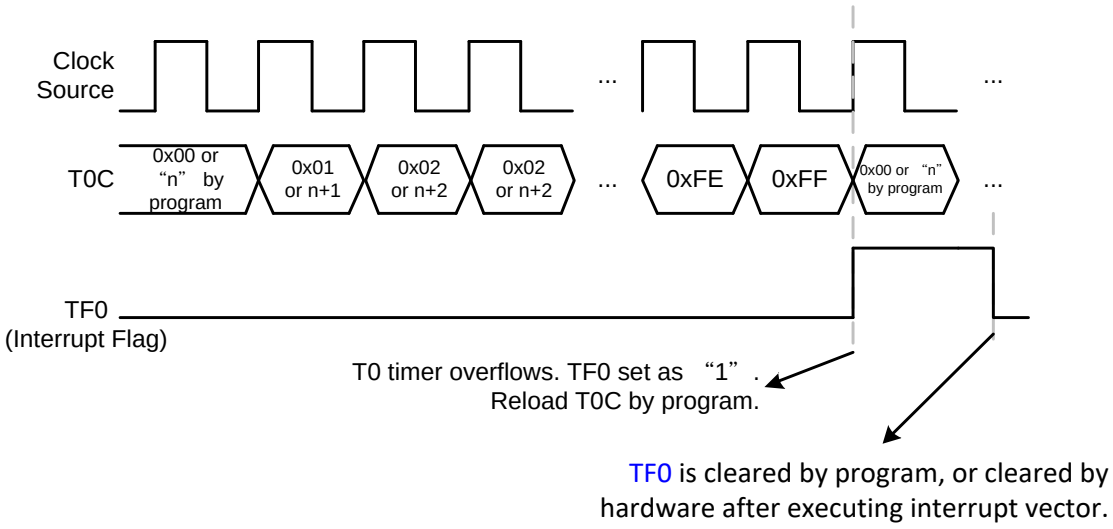
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (TF0 bit) and interrupt operation (interrupt vector 0x93). The interval time is programmable through T0M, T0C registers and supports RTC function. The T0 builds in Stop mode wake-up function. When T0 timer overflow occurs under stop mode, the system will be waked-up to normal 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 TF0 actives 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.
- **Stop mode function**: T0 keeps running in stop mode and can wake-up from STOP mode as **T0ENB = 1 & T0TB=1 @IHRC_RTC/IHRC** System will be Wake-up when TF0 actives after T0 timer overflow occurrence.



14.1 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, FT0 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 (ET0=1), the system will execute interrupt procedure. The interrupt procedure is system program counter points to interrupt vector (ORG 0093H) and executes interrupt service routine after T0 overflow occurrence. Clear TF0 by program is necessary in interrupt procedure. T0 timer can works in normal mode, idle mode and stop mode. In stop mode, T0 keeps counting, set TF0 and wakes up system when T0 timer overflows.



T0 clock source is Fcpu (instruction cycle) through T0RATE[2:0] pre-scalar to decide cpu/2~Fcpu/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)	
		Max overflow Time	One Step = max/256
000	Fcpu / 256	65.563 ms	256 us
001	Fcpu / 128	32.768 ms	128 us
010	Fcpu / 64	16.384 ms	64 us
011	Fcpu / 32	8.192 ms	32 us
100	Fcpu / 16	4.096 ms	16 us
101	Fcpu / 8	2.048 ms	8 us
110	Fcpu / 4	1.024 ms	4 us
111	Fcpu / 2	0.512 ms	2 us

14.2 T0 Mode Control 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.

T0M Register (0xBB)

Bit	Field	Type	Initial	Description
7	T0ENB	R/W	0	T0 timer enable bit. 0 : Disable. 1 : Enable.
6..4	T0RATE[2:0]	R/W	0	T0 timer clock source select bits. 000 : Fcpu/256. 100 : Fcpu/16. 001 : Fcpu/128. 101 : Fcpu/8. 010 : Fcpu/64. 110 : Fcpu/4. 011 : Fcpu/32. 111 : Fcpu/2.
1	T0MOD	R/W	0	T0 timer interrupt Mode control bit 0 : T0 interrupt occur when T0C register overflow. 1 : T0 interrupt occur when SEC register overflow.
0	T0TB	R/W	0	T0 timer RTC function control bit. 0 : Disable RTC function. T0 clock source from Fcup. 1 : Enable RTC function. T0 clock source from Low clock.

14.3 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.

T0C Register (0xBC)

Bit	Field	Type	Initial	Description
7..0	T0C[7:0]	R/W	0	Timer 0 counter register.

The equation of T0C initial value is as following:

T0C initial value = 256 - (T0 interrupt interval time * T0 clock rate)

Example: To calculation T0C to obtain 10ms T0 interval time.

T0 clock source is Fcpu = 32MHz/32 = 1MHz. Select T0RATE=001 (Fcpu/128)

T0 interval time = 10ms. T0 clock rate = 32MHz/32/128

T0C initial value = 256 - (T0 interval time * input clock)

= 256 - (10ms * 32MHz / 32 / 128) = 178 = **0xB2**.

14.4 Other Relative Register

SEC Register (0xBE)

Bit	Field	Type	Initial	Description
5..0	SEC[5:0]	R/W	0	Timer 0 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

MIN Register (0xBD)

Bit	Field	Type	Initial	Description
5..0	MIN[5:0]	R/W	0	Timer 0 Minute counter. MIN counter range = 00H ~ 3BH (0 ~ 59) When MIN value of 3BH and increase "1", counter overflow occurring with reset counter value to 00H

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
1	ET0	R/W	0	Timer 0 Interrupts enable. 0 : Disable Timer 0 interrupt. 1 : Enable Timer 0 interrupt.
Else				Refer to other chapter(s)

IRCON Register (0xC0)

Bit	**Field**	**Type**	**Initial**	**Description**
2	TF0	R/W	0	T0 timer interrupt request flag. 0 : None T0 interrupt request. 1 : T0 interrupt request.
Else				Refer to other chapter(s)

14.5 Sample Code

The following sample code demonstrates how to perform T0 with interrupt.

```
#include <intrins.h>     // for _nop_
#include <SN8F5849.h>

void T0_Init(void);

void Init_SysCLK(void)
{

   /* Clock Switch Select Register */
   CLKSEL = 0x02;       // Fcpu = Fhosc/32
   CLKCMD = 0x69;       // clock switch start

}

void main(void)
{
   Init_SysCLK();
   T0_Init();

   while (1) {
       WDTR = 0x5A;     // clear watchdog if watchdog enable

       // To Do...
   }
}

void T0_Init(void)
{
   P0M |= 0xFF;
   T0M |= 0X10;         // Fcpu/128
   T0C = 0XB2;          // T0C inital value = 256 - (T0 interal time * input clock)
                        // T0C inital value = 256 - (10ms*32MHz/32/128) = 0XB2

   IEN0 |= 0x02;        // T0IEN = 1
   T0M  |= 0X80;        // T0ENB = 1
   IEN0 |= 0x80;        // EAL = 1
}

void T0_ISR(void) interrupt ISRTimer0 // Vector @0x93
{
   // cleared TC0 interrupt flag by hardware
   // TCF0 = 0;          // Clear TCF0 flag

   T0C = 0XB2;          // Reload data to T0C
   P0 ^= 0XFF;          // P0 Toggle
}
```

The following sample code demonstrates how to perform T0 with 0.5 or 60 Sec wake up stop mode.

```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>

void T0_RTC_Init(void);
void GPIO_Init(void);

void Init_SysCLK(void)
{

   /* Clock Switch Select Register */
   CLKSEL = 0x02;       // Fcpu = Fhosc/32
   CLKCMD = 0x69;       // clock switch start

}

void main(void)
{
   Init_SysCLK();
   GPIO_Init();
   T0_RTC_Init();

   while (1) {
     WDTR = 0x5A;       // clear watchdog if watchdog enable
     STOP();            // System into STOP MODE
   }
}

void GPIO_Init(void)
{
   P0 = 0X00;
   P0M |= 0xFF;
}

void T0_RTC_Init(void)
{
   T0M |= 0X01;         // T0TB = 1,Clock source must from 32768 Hz X'tal
                        // Timer interrupt is occur when 0.5SEC overflow.

   /* 60s Interrupt function */
   //T0M |= 0X02;       // T0MOD = 1,Timer interrupt is occur when 60SEC overflow.
                        // "SEC" = 00H~3BH(0~59)

   IEN0 |= 0x02;        // T0IEN = 1
   T0M |= 0XF0;         // T0ENB = 1
   IEN0 |= 0x80;        // EAL = 1
}

void T0_ISR(void) interrupt ISRTimer0 // Vector @0x93
{
   // cleared TC0 interrupt flag by hardware
   // TCF0 = 0;           // Clear TCF0 flag

   P0 ^= 0XFF;            // P0 Toggle
}
```

```
void GPIO_Init(void)
{
   P1CON = 0xFF;        // P1 set IO Mode
   P2CON = 0xFF;        // P2 set IO Mode
   P3CON = 0xFF;        // P3 set IO Mode
   P4CON = 0xFF;        // P4 set IO Mode
   P5CON = 0xFF;        // P5 set IO Mode

   P0UR = 0xFF;         // P0 pull high for input mode
   P1UR = 0xFF;         // P1 pull high for input mode
   P2UR = 0xFF;         // P2 pull high for input mode
   P3UR = 0xFF;         // P3 pull high for input mode
   P4UR = 0xFF;         // P4 pull high for input mode
   P5UR = 0xFF;         // P5 pull high for input mode

   P0M = 0x00           // P0 as input mode
   P1M = 0x00           // P1 as input mode
   P2M = 0x00           // P2 as input mode
   P3M = 0x00           // P3 as input mode
   P4M = 0x00           // P4 as input mode
   P5M = 0x00           // P5 as input mode
}
```

15 Buzzer Function

Buzzer function is controlled by BZRENB bit, which be configured as GPIO mode or Buzzer mode. Buzzer output square wave signal through P1.2 Pin with 50% duty, and output frequency is controllable by setting the BZRCKS[2:0] register.

BZRM Register (0xD6)

Bit	Field	Type	Initial	Description
3..1	BZRCKS[2:0]	R/W	0	Buzzer output frequency 000 : 0.98KHz. 001 : 1.96KHz. 010 : 3.9KHz. 011 : 7.8KHz 100 : 15.6KHz. 101 : 31.25KHz other : reserved
0	BZRENB	R/W	0	Buzzer output control bit. 0 : GPIO Mode. 1 : Buzzer Mode. P12 output Buzzer signal.

***Example**: Buzzer Function Sample code.

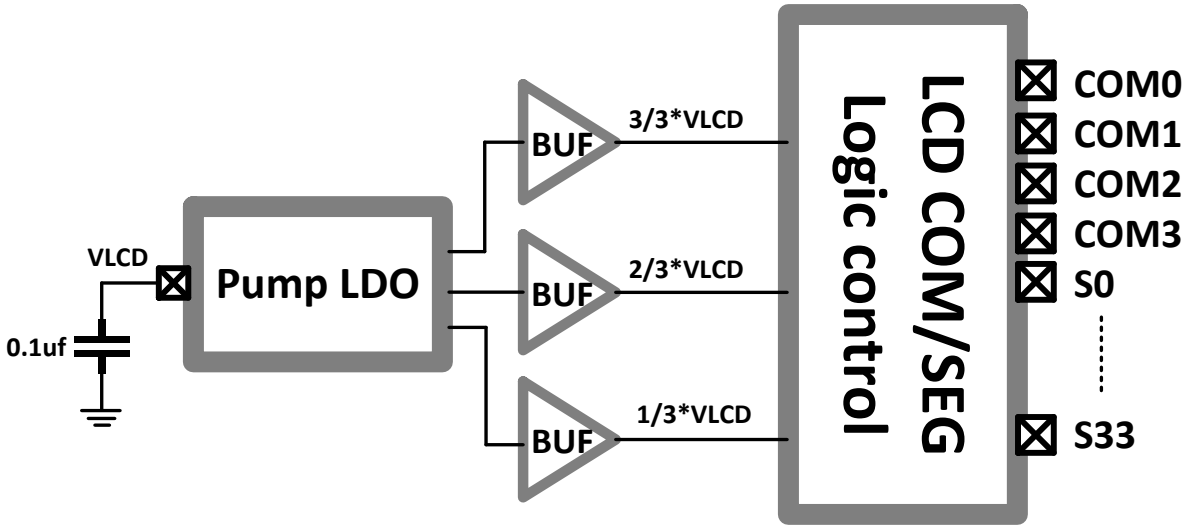
```
BZRM |= 0X02;  // Set Buzzer clock=1.95k Hz
BZRM |= 0X01;  // Buzzer enable
```

16 LCD Driver

LCD driver is C-type structures with 4 x 34 or 6 x 32. The LCD scan timing is 1/4 or 1/6 duty with 1/3 bias only, to drive LCD of 136 dots or 192 dots. C-type LCD driver can output adjustable VLCD output voltage and adjustable driving power to support diversity of LCD panel. VLCD Pin must connect 0.1uF capacitor to DVSS for C-Type LCD driver operation.

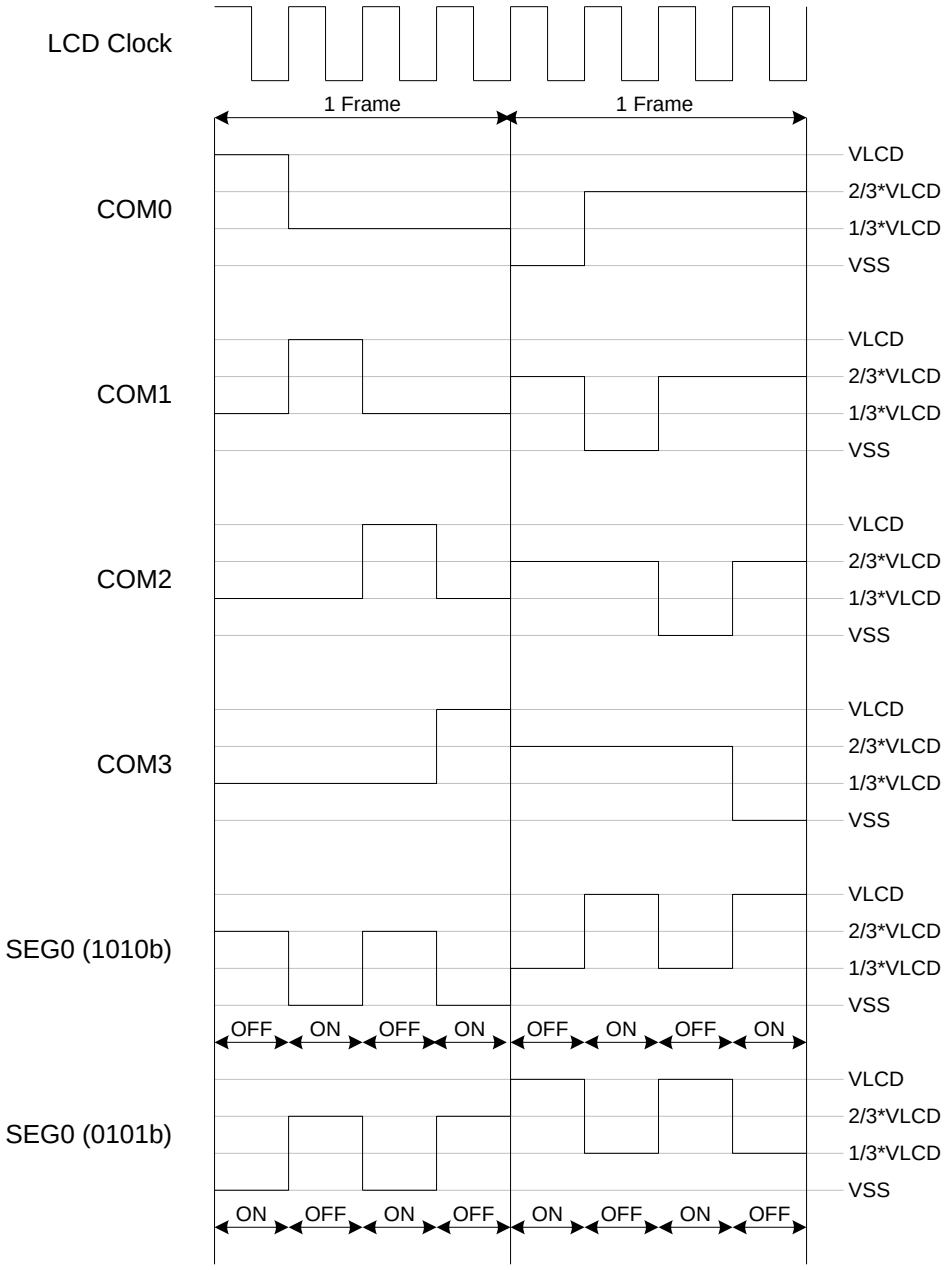
LCD driver output 4 or 6-com waveform controlled by LCDMODE bit, the output frame Rate controlled by LCDRATE bit, which shows in following table:

- **Stop mode function**: LCD function keeps running in stop mode as LCDEN = 1 & STOP = 1.
- **Low power mode function:** LCD low power function is controlled by BGM bit.

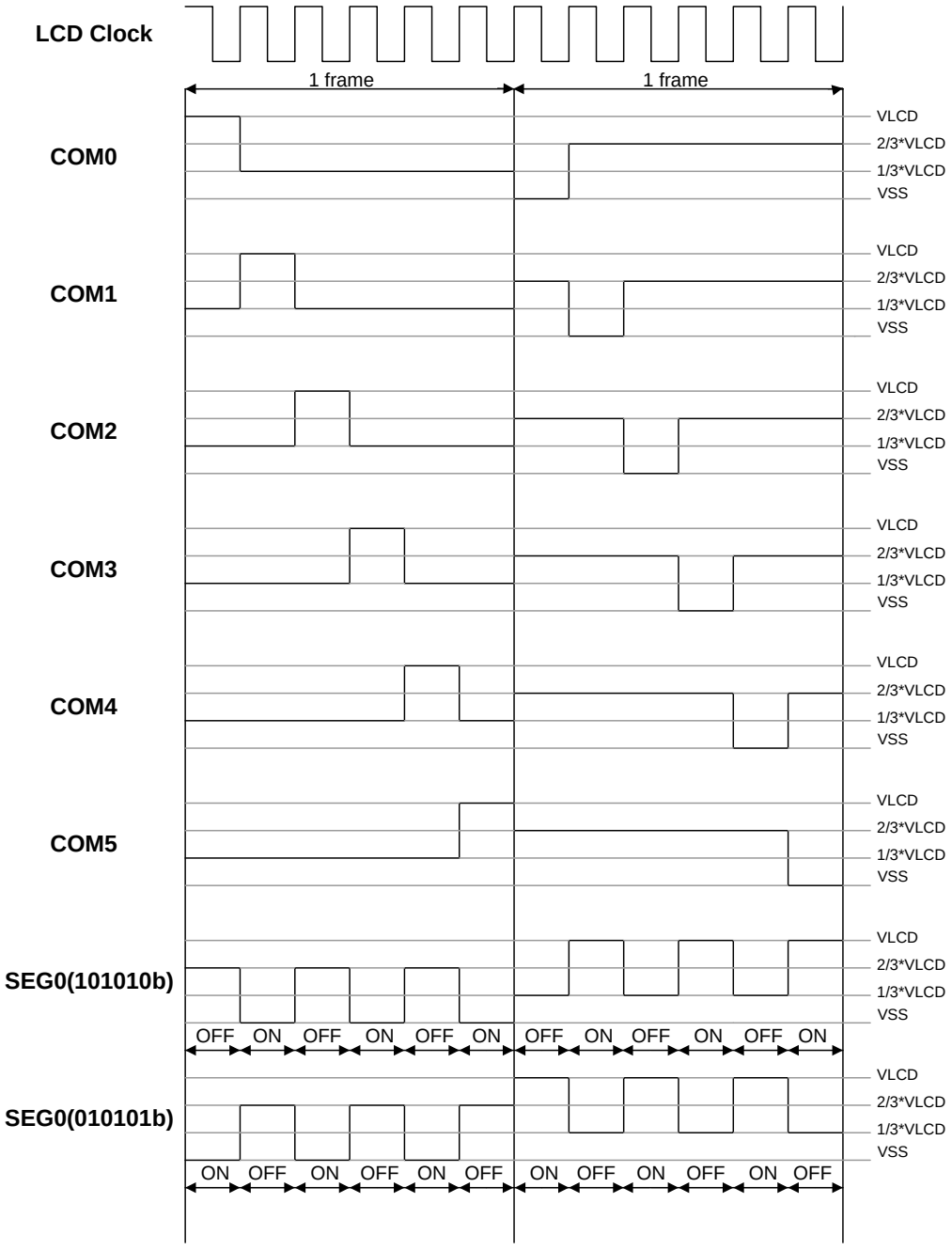


Control Register		LCD Clock (Hz)	Frame Rate (Hz)	Type
LCDRATE	LCDMODE			
0	0	32KHz / 128=256	64	4-COM (1/4 duty)
0	1		43	6-COM (1/6 duty)
1	0	32KHz / 64 =512	128	4-COM (1/4 duty)
1	1		85	6-COM (1/6 duty)

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

16.1 LCD RAM Location

LCD RAM locates in address from 0xF000 to 0xF022.

LCD RAM location relate to COM0 ~ COM3 vs. SEG0 ~ SEG34 LCD as show in following table.

	Bit0 COM0	Bit1 COM1	Bit2 COM2	Bit3 COM3	Bit4 -	Bit5 -	Bit6 -	Bit7 -
SEG 0	F000H.0	F000H.1	F000H.2	F000H.3	N/A	N/A	N/A	N/A
SEG 1	F001H.0	F001H.1	F001H.2	F001H.3	N/A	N/A	N/A	N/A
SEG 2	F002H.0	F002H.1	F002H.2	F002H.3	N/A	N/A	N/A	N/A
SEG 3	F003H.0	F003H.1	F003H.2	F003H.3	N/A	N/A	N/A	N/A
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 34	F022H.0	F022 H.1	F022 H.2	F022 H.3	N/A	N/A	N/A	N/A

LCD RAM location relate to COM0 ~ COM5 vs. SEG2 ~ SEG32 LCD as show in following table.

	Bit0 COM0	Bit1 COM1	Bit2 COM2	Bit3 COM3	Bit4 COM4	Bit5 COM5	Bit6 -	Bit7 -
SEG 0 (COM4)	N/A	N/A	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	N/A	N/A
SEG 2	F002H.0	F002H.1	F002H.2	F002H.3	F002H.4	F002H.5	N/A	N/A
SEG 3	F003H.0	F003H.1	F003H.2	F003H.3	F003H.4	F003H.5	N/A	N/A
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 32	F022H.0	F022 H.1	F022 H.2	F022 H.3	F022 H.4	F022 H.5	N/A	N/A

16.2 LCD Control Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	-	-	-	LCDBNK	LCDMODE	LCDRATE	LCDPEN	LCDEN
LCDM2	-	BGM	DUTY1	DUTY0	VCP3	VCP2	VCP 1	VCP0
LCDM3					ENLDOBIA	DUTYB1	DUTYB0	-

LCDM1 Register (0xAD)

Bit	Field	Type	Initial	Description
4	LCDBNK	R/W	0	LCD blank control bit. 0 : Normal display. 1 : All of the LCD dots turn off.
3	LCDMODE	R/W	0	LCD driver 4-COM or 6-COM control bit. 0 : 4-COM mode. 1 : 6-COM mode.
2	LCDRATE	R/W	0	LCD clock rate control bit. 0 : 256Hz. 1 : 512Hz.
1	LCDPEN	R/W	0	C-Type LCD pump enable bit. 0 : Pump disable. 1 : Pump enable.
0	LCDEN	R/W	0	LCD driver enable bit. 0 : LCD driver disable. 1 : LCD driver enable.

***Example:** 1.LCD Pump must enable (LCDPEN=1) and wait 1ms firstly, then enable LCD driver (LCDEN=1).

```
1   LCDM1 |= 0x02;  // Enable LCD Pump.
2   Delay1ms(1);    // Dealy 1m sec.
```

***Example:** 2.LCD function keeps running in stop mode as LCDEN = 1 & STOP = 1.

```
1   LCDM1 |= 0x01;  // Enable LCD Function.
    ...
    ...
100 STOP();           // Use C51 Macros into STOP MODE
```

LCDM2 Register (0xB2)

Bit	Field	Type	Initial	Description
6	BGM	R/W	**1**	LCD low power mode control bit. 0 : Enable low power mode. 1 : Normal operation mode.
5..4	DUTY[1:0]	R/W	0	LCD pump output driving capacity control bits. 00 : reserved 10 : 50% 01 : reserved 11 : 100% ***Note:** DUTY [1:0] is controllable when BGM=0, which is set for LCD low power consumption.
3..0	VCP[3:0]	R/W	1000	VLCD output voltage control bits. 0000 : 2.7V 1000 : 3.1V 0001 : 2.75V 1001 : 3.15V 0010 : 2.8V 1010 : 3.2V 0011 : 2.85V 1011 : 3.25V 0100 : 2.9V 1100 : 3.3V 0101 : 2.95V 1101 : 3.35V 0110 : 3.0V 1110 : 3.4V 0111 : 3.05V 1111 : 3.45V

***Example:** 3. LCD low power consumption mode.

```
LCDM2 |= 0x20;    // Driving capacity = 50%.
LCDM2 &= 0xBF;    // Enable Low power Mode.
```

LCDM3 Register (0XD4)

Bit	Field	Type	Initial	Description
3	ENLDOBIAS	R/W	0	LCD low power + mode control bit. 0 : Disable low power + mode. 1 : Enable low power + mode.
2..1	DUTYB[1:0]	R/W	00	LCD low power duty b Pulse control bits. 00 : Disable 10 : 1kHz 01 : 0.5kHz 11 : 2kHz (please set “11” for all application)

P1CON Register (0x9C)

Bit	Field	Type	Initial	Description
7..2	P1CON[7:2]	R/W	0xFF	Port 1 function control bit. 0 : Set as LCD function (SEG28 ~ SEG33) 1 : Set as GPIO function (P12 ~ P17)

P2CON Register (0x9B)

Bit	Field	Type	Initial	Description
7..0	P2CON[7:0]	R/W	0xFF	Port 2 function control bit. 0 : Set as LCD function (SEG20 ~ SEG27) 1 : Set as GPIO function (P20 ~ P27)

P3CON Register (0x9E)

Bit	Field	Type	Initial	Description
7..0	P3CON[7:0]	R/W	0xFF	Port 3 function control bit. 0 : Set as LCD function (SEG12 ~ SEG19) 1 : Set as GPIO function (P30 ~ P37)

P4CON Register (0x9F)

Bit	Field	Type	Initial	Description
7..0	P4CON[7:0]	R/W	0xFF	Port 4 function control bit. 0 : Set as LCD function (SEG4 ~SEG11) 1 : Set as GPIO function (P40 ~ P47)

P5CON Register (0x9D)

Bit	Field	Type	Initial	Description
7..0	P5CON[7:0]	R/W	0xFF	Port 5 function control bit. 0 : Set as LCD function (COM0 ~SEG3) 1 : Set as GPIO function (P50 ~ P57)

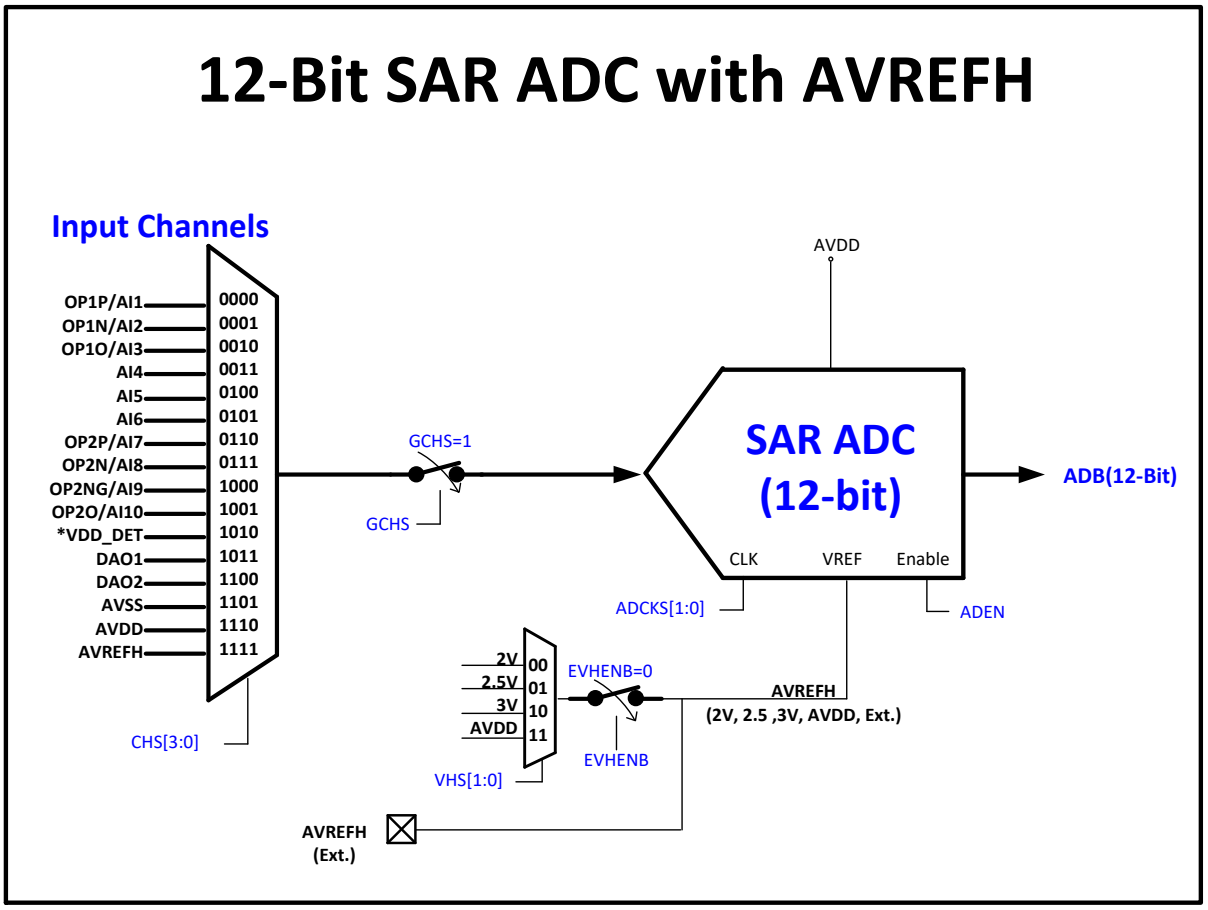
16.3 Sample Code

The following sample code demonstrates how to perform LCD Function keep running in stop mode.

```
void Init_LCD(void)
{
   LCDM1 &= 0x00;      // Clear LCDM1
   LCDM1 |= 0x02;      // Enable LCD Pump
   Delay1ms(1);        // Dealy 1m sec
   LCDM1 &= 0xF3;      // LCD Clock=256HZ,Frame=64Hz @4-COM

   LCDM2 &= 0x00;      // Clear LCDM2
   LCDM2 &= 0xBF;      // Enable Low power Mode @STOP MODE
   LCDM2 |= 0x08;      // VLCD=3.1V
   LCDM2 |= 0x20;      // Low power mode Driving capacity = 50%.

   LCDM1 |= 0x01;      // Enable LCD
}

void main(void)
{
   CLEAR_LCD_RAM();    // Clear LCD RAM
   Init_LCD();         // Initial LCD
   while (1)
   {
       // To Do...
       STOP();          // Use C51 Macros into STOP MODE
   }
}



void GPIO_Init(void)
{
   P1CON = 0xFF;        // P1 set IO Mode
   P2CON = 0xFF;        // P2 set IO Mode
   P3CON = 0xFF;        // P3 set IO Mode
   P4CON = 0xFF;        // P4 set IO Mode
   P5CON = 0xFF;        // P5 set IO Mode

   P0UR = 0xFF;         // P0 pull high for input mode
   P1UR = 0xFF;         // P1 pull high for input mode
   P2UR = 0xFF;         // P2 pull high for input mode
   P3UR = 0xFF;         // P3 pull high for input mode
   P4UR = 0xFF;         // P4 pull high for input mode
   P5UR = 0xFF;         // P5 pull high for input mode

   P0M = 0x00           // P0 as input mode
   P1M = 0x00           // P1 as input mode
   P2M = 0x00           // P2 as input mode
   P3M = 0x00           // P3 as input mode
   P4M = 0x00           // P4 as input mode
   P5M = 0x00           // P5 as input mode
}
```

17 SAR ADC

The analog to digital converter (ADC) is SAR structure with 10-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The SAR ADC builds in 10-channel input source to measure 10 different analog signal sources. The SAR ADC resolution is 12-bit. The ADC has four clock rates to decide SAR ADC converting rate. The SAR ADC reference high voltage includes 4 sources. Three internal power source including AVDD, 3V and 2V/2.5V. The other one is external reference voltage input pin from AVREFH pin. The ADC builds in CHS[3:0] SAR ADC input channel select registers to set pure analog input pin. After setup ADENB and ADS bits, the SAR ADC starts to convert analog signal to digital data. Besides ADS bit can start to convert analog signal. SAR ADC can work in idle mode. After SAR ADC operating, the system would be waked up from idle mode to normal mode if interrupt enable.



17.1 Configurations of Operation

These configurations must be setup completely before starting SAR ADC converting.SAR ADC is configured using the following steps:

1. Choose and enable the start of conversion SAR ADC input channel. (By CHS[3:0] bits & GCHS bit)
2. Choose SAR ADC high reference voltage. (By AVREFH register)
3. Choose SAR ADC Clock Source and Clock Rate. (By ADCKS[1:0] bits)
4. After setup ADENB bits, the SAR ADC ready to convert analog signal to digital data.

17.2 Start to Conversion

When SAR ADC IP is enabled by ADENB bit, it is necessary to make a SAR ADC start-up by program. Conversions may be initiated by the following:

- Writing a 1 to the ADS bit of register ADM

After setup ADENB and ADS bits, the SAR ADC starts to convert analog signal to digital data. The ADS bit is reset to logic 0 when the conversion is complete. When the conversion is complete, the SAR ADC circuit will set EOC and ADCF bits to “1” and the digital data outputs in ADB and ADR registers. If ADC interrupt function is enabled (EADC = 1), the SAR ADC interrupt request occurs and executes interrupt service routine when ADCF is “1” after SAR ADC converting. Clear ADCF by program is necessary in

17.3SAR ADC Input Channels

The SAR ADC builds in 10-channel input source (AI1 – AI10) to measure 10 different analog signal sources controlled by CHS[3:0] and GCHS bits. The VDD DET channel no any input pin from outside. In this time SAR ADC reference voltage must be internal 2V/2.5V or 3V or AVDD and External voltage, VDD DET can be a good battery detector for battery system. To select appropriate internal AVREFH level and compare value, a high performance and cheaper low battery detector is built in the system.

Analog input signal channel selection table:

MUXP[3:0]	**PGIA Positive input**	***Note***
0000	AI1 / OP1P	
0001	AI2 / OP1N	
0010	AI3 / OP1O	
0011	AI4	
0100	AI5	
0101	AI6	
0110	AI7 / OP2P	
0111	AI8 / OP2N	
1000	AI9 / OP2NG	
1001	AI10 / OP2O	
1010	VDD_DET	Battery detector channel (1/2*VDD or 1/4*VDD)
1011	DAO1	
1100	DAO2	
1101	AVSS	

1110	AVDD	
1111	AVREFH	

17.4 Reference Voltage

The SAR ADC builds in four high reference voltage source controlled through AVREFH register. There are one external voltage source and three internal voltage source (AVDD, 3V, 2V, 2.5V). When EVHENB bit is “1”, SAR ADC reference voltage is external voltage source from AVREFH In the condition. If EVHENB bit is “0”, SAR ADC reference high voltage is from internal voltage source selected by VHS[1:0] bits. If VHS[1:0] is “10”, SAR ADC reference high voltage is AVDD. If VHS[1:0] is “01”, SAR ADC reference high voltage is 3V. If VHS[1:0] is “00”, SAR ADC reference high voltage is 2V.

17.5 Signal Format

SAR ADC sampling voltage range is limited by high/low reference voltage. The SAR ADC low reference voltage is VSS. The SAR ADC high reference voltage includes internal AVREFH/3V/2V and external reference voltage source from AVREFH pin controlled by EVHENB bit. SAR ADC reference voltage range limitation is “(SAR ADC high reference voltage - low reference voltage)≥ 2V”. SAR ADC low reference voltage is VSS = 0V. So SAR ADC high reference voltage range is 2V to VDD. The range is SAR ADC external high reference voltage range.

- SAR ADC Internal Low Reference Voltage = 0V.
- SAR ADC Internal High Reference Voltage = AVREFH/3V/2V/2.5V. (EVHENB=0)
- SAR ADC External High Reference Voltage = 2V to AVREFH. (EVHENB=1)

SAR ADC sampled input signal voltage must be from SAR ADC low reference voltage to SAR ADC high reference. If the SAR ADC input signal voltage is over the range, the SAR ADC converting result is error (full scale or zero).

ADC Low Reference Voltage ≤ ADC Sampled Input Voltage ≤ ADC High Reference Voltage

17.6 Converting Time

The SAR ADC converting time is from ADS=1 (Start to SAR ADC convert) to EOC=1 (End of SAR ADC convert). The converting time duration is depend on SAR ADC clock rate. 12-bit SAR ADC’s converting time is 1/(SAR ADC clock /4)*16 sec. SAR ADC has one clock sources: fosc, ADCKS[1:0] bits: 00 = fosc/16, 01 = fosc/8, 10 = fosc/4, 11 = fosc/2.

The SAR ADC converting time affects SAR ADC performance. If input high rate analog signal, it is necessary to select a high SAR ADC converting rate. If the SAR ADC converting time is slower than analog signal variation rate, the SAR ADC result would be error. So to select a correct SAR ADC clock rate to decide a right SAR ADC converting rate is very important.

12 bits SAR ADC conversion time = 16/(SAR ADC Clock Rate/4)

ADCKS [1:0]	**ADC Clock Rate**	**Conversion Time**	**Conversion Rate**
00	Fosc/16	1/(32MHz/16/4)*16 = 32us	31.25k Hz
01	Fosc/8	1/(32MHz/8/4)*16 = 16us	62.5k Hz
10	Fosc/4	1/(32MHz/8/4)*16 = 8us	125k Hz
11	Fosc/2	1/(32MHz/2/2)*16 = 4us	250k Hz

17.7 Data Buffer

SAR ADC data buffer is 12-bit length to store SAR ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits. The ADB register is only 8-bit register including bit 4 – bit11 SAR ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit SAR ADC data buffer. The SAR ADC data buffer is a read-only register and the initial status is unknown after system reset.

AI n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
2/4096*VREFH	0	0	0	0	0	0	0	0	0	0	1	0
3/4096*VREFH	0	0	0	0	0	0	0	0	0	0	1	1
-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-
4096/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4096/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

17.8 SAR ADC Control Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	-	CHS3	CHS2	CHS1	CHS0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
ADR	LVBAT	GCHS	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
VREFH	EVHENB	VIBASEN	-	-	VBIAS1	VBIAS0	VHS1	VHS0
IEN0	EAL	-	ESAR	ETS0	ERS0	EX1	ET0	EX0
IRCON2	-	-	-	-	-	-	-	SARF

ADM Register (0xC8)

Bit	Field	Type	Initial	Description
7	ADENB	R/W	0	SAR ADC control bit. In stop mode, disable ADC to reduce power consumption. 0 : Disable 1 : Enable
6	ADS	R/W	0	SAR ADC conversion control. Write 1: Start SAR ADC conversion (automatically cleared by the end of conversion)
5	EOC	R/W	0	SAR ADC status bit. 0: SAR ADC progressing 1: End of conversion (automatically set by hardware; manually cleared by firmware)
3..0	CHS[3:0]	R/W	0x00	SAR ADC positive input channel selection bits. 0000 : AI1 0100 : AI5 1000 : AI9 1100 : DAO2 0001 : AI2 0101 : AI6 1001 : AI10 1101 : AVSS 0010 : AI3 0110 : AI7 1010 : VDET 1110 : AVDD 0011 : AI4 0111 : AI8 1011 : DAO1 1111: AVREFH

ADB Register (0xD1)

Bit	Field	Type	Initial	Description
7..0	ADB[11:4]	R	-	SAR ADC Result Bit [11:4]* in 12-bit ADC resolution mode.

* ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits.

ADR Register (0xD2)

Bit	Field	Type	Initial	Description
7	LVBAT	R/W	0	Low battery detector voltage select bit. 0: 1/4*VDD. 1: 1/2*VDD.
6	GCHS	R/W	0	SAR ADC global channel select bit. 0: Disable AIN channel. 1: Enable AIN channel.
5..4	ADCKS[1:0]	R/W	00	SAR ADC's clock rate select bit. 00 : Fosc/16 01 : Fosc/8 10 : Fosc/4 11 : Fosc/2
3..0	ADB[3:0]	R	-	SAR ADC Result Bit [3:0]* in 12-bit SAR ADC resolution mode.

* ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits.

VREFH Register (0xB3)

Bit	Field	Type	Initial	Description
7	EVHENB	R/W	1	SAR ADC/DAC internal reference voltage control bit. 0: Enable SAR ADC/DAC internal VREFH function. AVREFH pin is internal AVREFH output pin.. 1: Disable SAR ADC/DAC internal VREFH function. AVREFH pin is external AVREFH input pin.
2..0	VHS[2:0]	R/W	00	SAR ADC internal reference high voltage selects bits. 00 : VREFH = 2.0V. 01 : VREFH = 2.5V. 10 : AREFH = 3.0V 11 : AVDD
Else				Refer to other chapter(s)

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
5	ESAR	R/W	0	SAR ADC interrupt control bit. 0: Disable SAR ADC interrupt function. 1: Enable SAR ADC interrupt function.
Else				Refer to other chapter(s)

IRCON2 Register (0xBF)

Bit	Field	Type	Initial	Description
0	SARF	R/W*	0	SAR ADC interrupt request flag. 0 = None SAR ADC interrupt request. 1 = SAR ADC interrupt request.
Else				Refer to other chapter(s)

* This bit can't write '1' value.

✸ ***Note:***
1. AVREFH Pin must load capacitor 0.1uF.

17.9 Sample Code

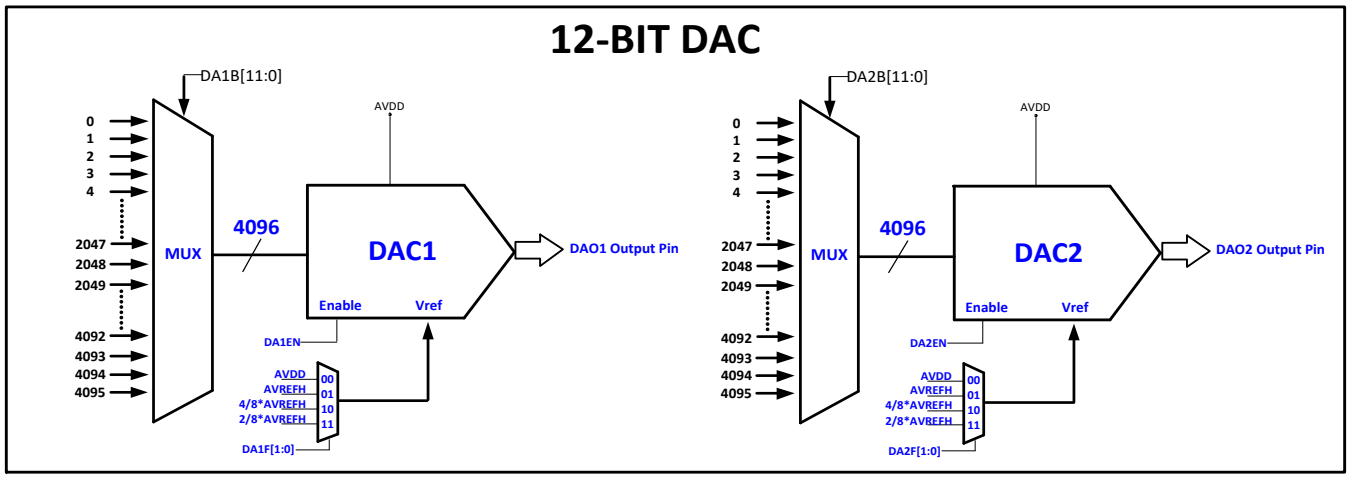
The following sample code demonstrates how to perform SAR ADC with interrupt.

```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>

unsigned short SARValue;

void Init_ADC(void);
void ADC_Enable_ISR(void);
void ADC_ISR(void);

void main(void)
{
   Init_SAR();          // Initial SAR ADC
   SAR_Enable_ISR();    // SAR ADC ISR Enable
   ADS =1 ;             // SAR ADC START Enable
   while (1) {
       WDTR = 0x5A;     // clear watchdog if watchdog enable
       // To Do ...
   }
}

void Init_SAR(void)
{
   VREFH &= 0x7C;       // Int. AVREFH Voltage=2V

   ADR &= 0x0F;         // Clear ADR Register
   ADR |= 0x10;         // SAR Clk=Fhosc/8
   ADR |= 0x40;         // Enable SAR channel

   ADM &= 0x00;         // Clear ADM
   ADM &= 0xF0;         // Select AIN1 channel
   ADM |= 0x80;         // Enable SAR

}

void SAR_Enable_ISR(void)
{
   IRCON2 &= 0xFE;      // Clear SARF
   IEN0 &= 0x00;        // Clear IEN0
   IEN0 |= 0x20;        // SAR interrupt enable (ESAR)
   EAL = 1;             // Interrupt enable
}
```

```
55 void SAR_ISR(void) interrupt ISRSar
56 {
57    // Get ADC value
58     if (EOC) {
59        EOC=0;           // Clear SAR End Flag
60        /* Get SAR ADC Data ( ADB,ADR ) */
61        SARValue = ADB; // Get SAR High Data
62        SARValue = SARValue<<4;
63        SARValue = SARValue+(ADR&0x0F);
64        _nop_();
65     }
66 }
67
68
69
70
71
```

18 DAC

The DAC structure is 12-bit resolution. DAC output circuit builds in one OP amp to enhance analog output driving current. When DAxEN=0, disable DAC function. When DAxEN=1, enable DACx function. DACx output signal connect to IO pad “DAOx”.



18.1 DAC Control Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAM	DA1EN	DA2EN	-	DA2F1	DA2F0	-	DA1F1	DA1F0
DA1BH	-	-	-	-	DA1B11	DA1B10	DA1B9	DA1B8
DA1BL	DA1B7	DA1B6	DA1B5	DA1B4	DA1B3	DA1B2	DA1B1	DA1B0
DA2BH	-	-	-	-	DA2B11	DA2B10	DA2B9	DA2B8
DA2BL	DA2B7	DA2B6	DA2B5	DA2B4	DA2B3	DA2B2	DA2B1	DA2B0
VREFH	EVHENB	VIBASEN	-	-	VIBAS1	VIBAS0	VHS1	VHS0

DAM Register (0xAB)

Bit	Field	Type	Initial	Description
7	DA1EN	R/W	0	DAC1 control bits 0 : Disable DAC1 function. 1 : Enable DAC1 function.
6	DA2EN	R/W	0	DAC2 control bits 0 : Disable DAC2 function. 1 : Enable DAC2 function.
4..3	DA2F[1:0]	R/W	0	DAC2 VREF setting bit. 00 : AVDD 01 : AVREFH 10 : VREF = 4/8 * AVREFH. 11 : VREF = 2/8 * AVREFH. **Note:** External VREF voltage (AVREFH PIN)

1..0	DA1F[1:0]	R/W	0	DAC1 VREF setting bit. 00 : AVDD 01 : AVREFH 10 : VREF = 4/8 * AVREFH. 11 : VREF = 2/8 * AVREFH. **Note:** External VREF voltage (AVREFH PIN)

DA1BH Register (0xAD)

Bit	Field	Type	Initial	Description
3..0	AD1B[11:8]	R/W	0	DAC1 setup data bit [11:8] in 12-bit resolution.

DA1BL Register (0xAC)

Bit	Field	Type	Initial	Description
7..0	AD1B[7:0]	R/W	0	DAC1 setup data bit [7:0] in 12-bit resolution.

Note: DA1B[11:0] written sequence is write DA1B[11:8] first, and then write DA1B[7:0]. End of writing DA1B[7:0] signal means the end of DA1B written.

DA2BH Register (0xB6)

Bit	Field	Type	Initial	Description
3..0	AD2B[11:8]	R/W	0	DAC2 setup data bit [11:8] in 12-bit resolution.

DA2BL Register (0xB5)

Bit	Field	Type	Initial	Description
7..0	AD2B[7:0]	R/W	0	DAC2 setup data bit [7:0] in 12-bit resolution.

Note: DA2B[11:0] written sequence is write DA2B[11:8] first, and then write DA2B[7:0]. End of writing DA2B[7:0] signal means the end of DA2B written.

VREFH Register (0xB3)

Bit	Field	Type	Initial	Description
7	EVHENB	R/W	1	SAR ADC/DAC internal reference voltage control bit. 0: Enable SAR ADC/DAC internal AVREFH function. AVREFH pin is internal AVREFH output pin.. 1: Disable SAR ADC/DAC internal AVREFH function. AVREFH pin is external AVREFH*(1) input pin. Note: EVHENB = 0 Enable SAR ADC/DAC internal AVREFH function.

1..0	VHS[1:0]	R/W	00	SAR ADC/DAC internal reference high voltage selects bits. 00 : VREFH = 2.0V. 01 : VREFH = 2.5V. 10 : VREFH = 3.0V. 11 : AVDD
Else				Refer to other chapter(s)

✸ ***Note:***
1. AVREFH Pin must load capacitor 0.1uF.

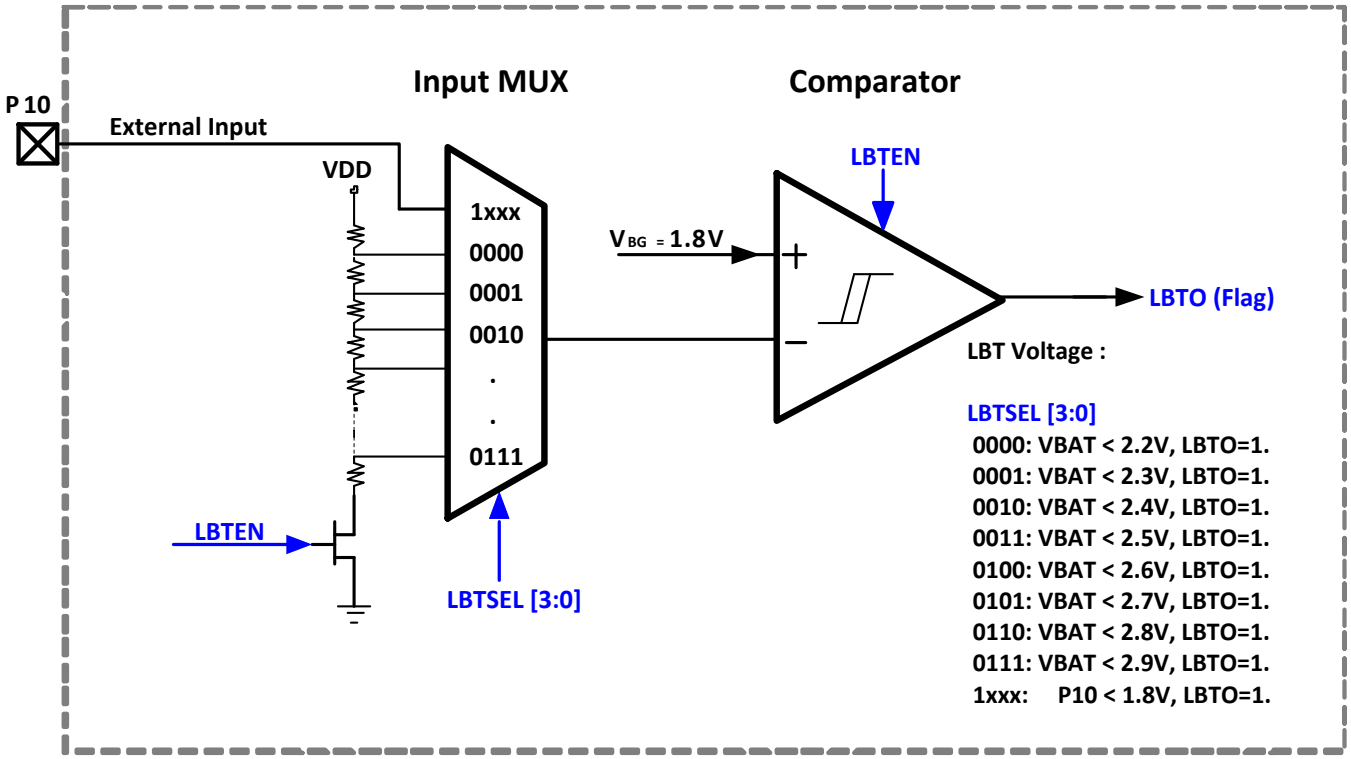
18.2Sample Code

The following sample code demonstrates how to perform DAC1

```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>

unsigned short ADCValue;

void Init_DAC(void);

void main(void)
{
   Init_VREG();         // Initial VREG
   Init_DAC();          // Initial DAC

   while (1) {
       WDTR = 0x5A;    // clear watchdog if watchdog enable
       // To Do ...
   }
}

void Init_DAC(void)
{
   VREFH &= 0xEF;       // Enable VREFH LDO
   VREFH |= 0x01;       // AVREFH = 2.5V

   DA1BH = 0x07;         // setting DAC output data
   DA1BL = 0xFF;         //

   DAM &= 0x00;         // Clear DAM
   DAM |= 0x03;         // DAC1 Vref = AVREFH*2/8
   DAM |= 0x80;         // Enable DAC1
}
```

19 Comparator

The microcontroller builds in comparator functions. When the positive input voltage is greater than the negative input voltage, the comparator output is high (LBTO=1). When the positive input voltage is smaller than the negative input voltage, the comparator output is low (LBTO=0). Comparator positive voltage is always from internal 1.8V. Comparator negative voltage is controllable from external P10 and Internal VDD voltage division. The comparator has flag indicator (LBTO) and use for different application, like low battery detection. Internal VDD voltage division of comparator negative input has eight voltage levels, which is selectable via LBTSEL[3:0] register, from 2.2V to 2.9V with 0.1V per step.



19.1 Comparator Control Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	-	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTEN

LBTM Register (0xD7)

Bit	Field	Type	Initial	Description
5..2	LBTSEL[3:0]	R/W	010	Internal VDD voltage division control bits. 0000 : VDD < 2.2V, LBTO=1. 0001 : VDD < 2.3V, LBTO=1. 0010 : VDD < 2.4V, LBTO=1. 0011 : VDD < 2.5V, LBTO=1. 0100 : VDD < 2.6V, LBTO=1. 0101 : VDD < 2.7V, LBTO=1. 0110 : VDD < 2.8V, LBTO=1. 0111 : VDD < 2.9V, LBTO=1. 1xxx : P10 < 1.8V, LBTO=1.
1	LBTO	R/W	0	Comparator Output flag. 0 : Comparator negative voltage great than 1.8V. 1 : Comparator negative voltage less than 1.8V.
0	LBTEN	R/W	00	Comparator function enable bit. 0 : Disable. 1 : Enable.

19.2 Comparator Characteristic

SYM.	*DESCRIPTION*	*PARAMETER*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
V_{ILBT}	Internal Low-Battery detect voltage	Condition: LBTSEL [3:0] = 0000, VLBT=2.2V	2.15	2.2	2.25	V
		Condition: LBTSEL [3:0] = 0100, VLBT=2.6V	2.55	2.6	2.65	
		Condition: LBTSEL [3:0] = 0111, VLBT=2.9V	2.85	2.9	2.95	
V_{ELBT}	External Low-Battery detect voltage	Condition: LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.75	1.8	1.85	
V_{HY}	Comparator Hysteresis Window			50	-	mV
I_{COMP}	Current consumption	AVDD = 3V		50	-	uA

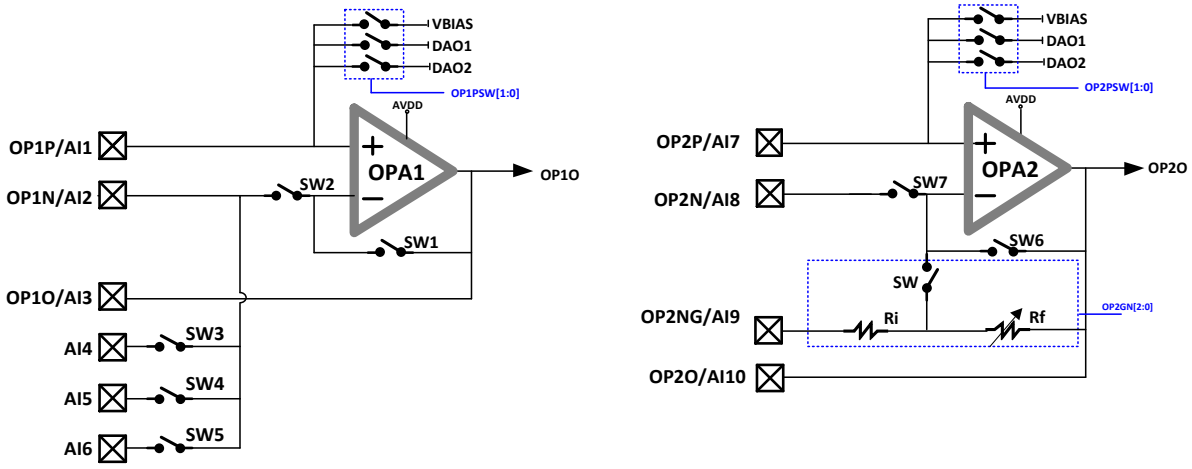
19.3 Sample Code

The following sample code demonstrates how to perform comparator functions.

```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>

unsigned char LBTOValue;

void Init_VREG(void);
void Init_LBTM(void);

void main(void)
{
   Init_LBTM();         // Initial LBT

   while (1) {
       LBTOValue = LBTM; //Get LBTO Flag
       LBTOValue = LBTOValue>>1;
       LBTOValue &= 0x01;

       if(LBTOValue==1){
         // To Do ...
       }
       Else{
         // To Do ...
       }
    }
}


void Init_LBTM(void)
{
   LBTM &= 0x00;        // Clear LBTM
   /* Low Battery Detect Voltage as 2.6v */
   LBTM |= 0x10;

   LBTM |= 0x01;        // Comparator function enable
}
```

20 OPA

The microcontroller builds in two operational amplifiers. The OP-Amp power source form AVDD. OP-Amp input signal and output voltage are within range of 0V to AVDD-0.1V. The OP-Amp input and output pin share with ADC input channel (AI1~AI3 & AI7~AI10). OP_Amp can be configured as buffer mode when SW1=1 & SW6=1. OPA2 has internal Gain option with selective range of x1 ~x95.



20.1 OP-Amp Control Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM1	SW6	SW5	SW4	SW3	SW2	SW1	OP2EN	OP1EN
OPM2	OP1PSW1	OP1PSW0	OP2PSW1	OP2PSW0	OP2GN2	OP2SN1	OP2GN0	SW7
OPM3	-	-	-	-	CHP2SEL1	CHP2SEL0	CHP1SEL1	CHP1SEL0
VREFH	EVHENB	VBIASEN	-	-	VBIAS1	VBIAS0	VHS1	VHS0

OPM1 Register (0xAE)

Bit	Field	Type	Initial	Description
7	SW6	R/W	0	OPA2 buffer mode control bit. 0 : Disable. 1 : Enable, OP2O and OP2- are shorted by MOS-SW.
6..3	SW[5:2]	R/W	0	CMOS switch control bits 0 : Disable. 1 : Enable, AIx and OP1- are shorted by MOS-SW.
2	SW1	R/W	0	OPA1 buffer mode control bit. 0 : Disable. 1 : Enable, OP1O and OP1- are shorted by MOS-SW.

1	OP2EN	R/W	0	OPA2 function enable bit. 0 : Disable. 1 : Enable.
0	OP1EN	R/W	0	OPA1 function enable bit. 0 : Disable. 1 : Enable.

OPM2 Register (0xAF)

Bit	Field	Type	Initial	Description
7..6	OP1PSW[1:0]	R/W	00	OP1PSW reference voltage connects OP1+ selection bit 00 : Disable. 01 : VBIAS. 10 : DAO1. 11 : DAO2.
5..4	OP2PSW[1:0]	R/W	00	OP2PSW reference voltage connects OP2+ selection bit 00 : Disable. 01 : VBIAS. 10 : DAO1. 11 : DAO2.
3..1	OP2GN[2:0]	R/W	0	OPA2 Gain Switch Selection. 000 : Diasble. 100 : Rf/Ri=15. 001 : Rf/Ri=1. 101 : Rf/Ri=31. 010 : Rf/Ri=3. 110 : Rf/Ri=63. 011 : Rf/Ri=7. 111 : Rf/Ri=95.
0	SW7	R/W	0	CMOS switch control bits 0 : Disable. 1 : Enable, AI8 and OP2- are shorted by MOS-SW.

OPM3 Register (0xB7)

Bit	Field	Type	Initial	Description
3..2	CHP2SEL[1:0]	R/W	00	OP2 chopper control bits 00 : Disable. 10 : 62.5K 01 : 31.25K. 11 : 125K
1..0	CHP1SEL[1:0]	R/W	00	OP1 chopper control bits 00 : Disable. 10 : 62.5K 01 : 31.25K. 11 : 125K

VREFH Register (0xB3)

Bit	Field	Type	Initial	Description
7	EVHENB	R/W	1	SAR ADC/DAC internal reference voltage control bit. 0: Enable SAR ADC/DAC internal AVREFH function. AVREFH pin is internal AVREFH output pin.. 1: Disable SAR ADC/DAC internal AVREFH function. AVREFH pin is external AVREFH*(1) input pin. Note: EVHENB = 0 Enable SAR ADC/DAC internal AVREFH function.
6	VBIASEN	R/W	0	OPAx + internal reference voltage control bit. 0: Disable OPAx + internal VBIAS function. 1: Enable OPAx + internal VBIAS function.
3..2	VBIAS[1:0]	R/W	0	OPAx + internal reference voltage selects bit. 00 : VBIAS = AVREFH 01 : VBIAS = AVREFH/2 10 : VBIAS = AVREFH/4 11 : VBIAS = AVREFH/16
1..0	VHS[1:0]	R/W	00	SAR ADC/DAC internal reference voltage selects bits. 00 : VREFH = 2.0V. 01 : VREFH = 2.5V. 10 : VREFH = 3.0V. 11 : AVDD
Else				Refer to other chapter(s)

***Example**: Operational amplifier 2 & internal gain function Sample code.

```
OPM3 |= 0x04;    // Set Chopper 31.25k Hz.
OPM2 |= 0x0C;     // Set Gain x64.

OPM1 |= 0X02;    // OP-Amp2 enable.

```

20.2 OP-Amp Electrical Characteristic

Operation Amplifier						
PARAMETER	**SYM.**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
Operation voltage	V_{oper}	Power source from AVDD	2.0	-	3.6	V
Input offset voltage	V_{OS}			±3		mV
Operating Current	I_{oper}	Per OP-Amp	-	100	-	uA
Analog Input voltage range	V_{IN}	OP+ / OP-	0.05		AVDD-0.1	V
Analog Output voltage range	V_{out}	OPOUT	0.05		AVDD-0.1	V
Output short circuit current	I_{OH}/I_{OL}	Unit Gain Buffer. Vo = 0V~3.2V, AVDD=3.3V	-	±5	-	mA
Buffer Mode Switch Ron	R_{BUF}	AVDD 3.3V, VOUT=1.65V.	-	50	--	Ω
Gain Bandwidth	GB	RL=300KΩ, CL=50pF		1.4	-	MHz
Slew Rate	SR	10% to 90%	-	0.8	-	V/uS
Turn on time	T_{ON}		-	20	-	uS

21 UART

The universal synchronous/asynchronous receiver/transmitter, or UART, provides an up to 1 MHz flexible full-duplex transmission. It has four operate modes: one synchronous, and three asynchronous. The synchronous mode transmits 8 bits data without start/stop bits, whereas the other modes respectively support 8 and 9 bits data transmission with start/stop bits appended. UTX signal output via P0.4 or P2.2 and URX signal received via P0.5 or P2.3.

21.1 UART Mode 0: Synchronous 8-bit Receiver/Transmitter

In mode 0, the UART transmits/receives an 8-bit-length data without start and stop bits. It handles the first bit of the bus as LSB (least significant bit), and its baud rate is fixed at fcpu/12. The reception is started by setting REN0 and clearing RI0 bits, whereas the transmission is started by writing data to S0BUF.

21.2 UART Mode 1: Asynchronous 8-bit Receiver/Transmitter

In mode 1, the UART operates typical format protocol which has a start bit, 8 data bits, and one stop bit. Its baud rate is controlled by S0RELH/S0RELL registers, TC2 Timer overflow period depending on BD register.

A Transmission is started by writing S0BUF register. The first bit of transmission is a start bit (always 0), then data from S0BUF proceed (LSB first). After which a stop bit (always 1) is transmitted, and TI0 bit is automatically set as a notification/interrupt.

The UART synchronizes with the start bit from master if the reception is enabled (REN0 = 1). The first data bit would be seen as LSB (least significant bit), and S0BUF would be updated after the completion of a reception.

21.3 UART Mode 2/3: Asynchronous 9-bit Receiver/Transmitter

Both mode 2 and 3 have a start bit, 9 data bits, and one stop bit. The mode 2 has only two baud rate selections, fcpu/32 and fcpu/64, but the mode 3 can control its baud rate by S0RELH/S0RELL registers or TC2 Timer overflow period.

The 9th data bit (after the MSB of S0BUF register proceed) is writable in TB80 register for its transmission, and readable in RB80 for a reception. By always setting TB80 register, it is an alternative method to transmit two stop bits with 8 data bits. Another common application of 9th bit is parity check.

Furthermore, the 9th bit can also be seen as identification for multiprocessor communication. If SM20 register is set, the receive interrupt is generated only when the 9th received bit is high. To utilize this feature to multiprocessor communication, on the other word, all slave processors set SM20 to 1. The master processor transmits the slave's address, with the 9th bit set to 1, causing reception interrupt in all of the slaves. The slave processors' software compares the received byte with their network address. If there is a match, the addressed slave clears its SM20 flag and the rest of the message is transmitted from the master with the 9th bit set to 0. The other slaves keep their SM20 to 1 so that they ignore the rest of the message sent by the master.

21.4 Baud Rate Control

The UART mode 0 has a fixed baud rate at fcpu/12, and the mode 2 has two baud rate selection which is chosen by SMOD register: fcpu/32 (SMOD = 0) and fcpu/64 (SMOD = 1).

The baud rate of UART mode 1 and mode 3 is generated by either S0RELH/S0RELL registers (BD = 1) or TC2 Timer overflow period (BD = 0). The SMOD bit doubles the frequency from the generator. The SMOD bit doubles the frequency from the generator.

If the S0RELH/S0RELL is selected (BD = 1 & CD = 1/0) in mode 1 and 3, the baud rate is generated as following equation.

$$Baud\ rate = 2^{SMOD} \times \frac{\text{fcpu}}{(64 - 8 \times \text{CD0}) \times (1024 - \text{S0REL})}$$

Table 19-1 Recommended Setting for Common UART Baud Rates (fcpu = 16 MHz)

Baud Rate	SMOD	CD	S0RELH	S0RELL	Accuracy
4800 bps	1	0	0x03	0x98	-0.16 %
9600 bps	1	0	0x03	0xCC	-0.16 %
19200 bps	1	0	0x03	0xE6	-0.16 %
38400 bps	1	0	0x03	0xF3	-0.16 %
56000 bps	1	0	0x03	0xF7	0.79 %
57600 bps	1	1	0x03	0xF6	0.79 %
115200 bps	1	1	0x03	0xFB	0.79 %
250k bps	1	0	0x03	0xFE	0 %
500k bps	1	0	0x03	0xFF	0 %

Note: The Baud rate 57600, 115200 bps CD bit have been setting 1

If the TC2 Timer overflow period is selected (BD = 0) in mode 1 and 3, the baud rate is generated as following equation. The TC2 Timer must be in 16-bit auto-reload mode which can generate periodically overflow signals.

$$\text{Baud Rate} = 2^{SMOD} \times \frac{1}{(32 - 4 \times \text{CD0}) \times \text{Timer period}}$$

Table 19-2 Recommended Setting TC2 Timer overflow period for Common UART Baud Rates (fcpu = 32 MHz)

Baud Rate	SMOD	CD0	Timer Period	TC2CH/ L	TC2RH/ L	Accuracy
4800 bps	1	0	13.021 us	0xFE30	0xFE30	0.16 %
9600 bps	1	0	6.510 us	0xFF98	0xFF98	0.16 %
19200 bps	1	0	3.255 us	0xFFCC	0xFFCC	0.16 %
38400 bps	1	0	1.628 us	0xFFE6	0xFFE6	0.16 %
56000 bps	1	0	1.116 us	0xFFEE	0xFFEE	-0.80 %
57600 bps	1	0	1.085 us	0xFFEF	0xFFEF	2.80 %
115200	1	0	0.543 us	0xFFF7	0xFFF7	-3.68 %
128k bps	1	0	0.488 us	0xFFF8	0xFFF8	-2.40 %
250k bps	1	0	0.250 us	0xFFFC	0xFFFC	0 %
500k bps	1	0	0.125 us	0xFFFE	0xFFFE	0 %

✸ ***Note: System clock Fcpu minimum setting value follow Fcpu < Timer Priod Value.***
Ex: If TC2 Timer overflow period is 6.510us system clock Fcpu can setting 32M/64 or 32M/128.

***Example**: TC2 Timer overflow period is selected (BD = 0) in mode 1 and 3, the baud rate 115200 is setting as following equation.(Fcpu = 16M Hz)

```
TC2CH = 0x0FF;      // baud rate 9600
TC2CL = 0x098;      // 6.510us Overflow
TC2RH = 0x0FF;      // auto-reload value
TC2RL = 0x098;      //
TC2M |= 0x0F0;      // TC2 ENABLE, TC2 timer clock= Fpcu/1

S0CON2 = 0x00;      // baud rate = TC2 Timer overflow period
PCON |= 0x80;       // SMOD=1,Fcpu*2 in mode1/mode3;
S0CON |=0x50;       // UART Enable,UART Mode 1
```

21.5 UART Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
S0CON2	BD	-	CD	TEN0	-	-	-	URCHS
S0TBUF	S0TBUF7	S0TBUF6	S0TBUF5	S0TBUF4	S0TBUF3	S0TBUF2	S0TBUF1	S0TBUF0
S0RBUF	S0RBUF7	S0RBUF6	S0RBUF5	S0RBUF4	S0RBUF3	S0RBUF2	S0RBUF1	S0RBUF0
PCON	SMOD	-	-	IDEL+	P2SEL	GF0	STOP	IDLE
S0RELH	-	-	-	-	-	-	S0REL9	S0REL8
S0RELL	S0REL7	S0REL6	S0REL5	S0REL4	S0REL3	S0REL2	S0REL1	R0REL0
IEN0	EAL	-	ESAR	ETS0	ERS0	EX1	ET0	EX0
P1OC	-	P23OC	P22OC	P05OC	P04OC	P17OC	P16OC	P15OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
P0	P07	P06	P05	P04	P03	P02	P01	P00
P2	P27	P26	P25	P24	P23	P22	P21	P20

S0CON Register (0x98)

Bit	Field	Type	Initial	Description
7..6	SM[0:1]	R/W	00	UART mode selection 00: Mode 0 01: Mode 1 10: Mode 2 11: Mode 3
5	SM20	R/W	0	Multiprocessor communication (mode 2, 3) 0: Disable 1: Enable
4	REN0	R/W	0	UART-RX reception function 0: Disable 1: Enable
3	TB0	R/W	0	The 9th bit transmission data (mode 2, 3)
2	RB0	R/W	0	The 9th bit data from reception
1	TI0	R/W	0	UART interrupt flag of transmission
0	RI0	R/W	0	UART interrupt flag of reception

S0CON2 Register (0xD8)

Bit	Field	Type	Initial	Description
7	BD	R/W	1	Baud rate generators selection (mode 1, 3) 0: TC2 Timer overflow period 1: Controlled by S0RELH, S0RELL registers
5	CD	R/W	0	UART baud rate 57600~115200 bps < 1% accuracy adjust bit. 1: UART accuracy functions enable. 0: UART accuracy functions disable.
4	TEN0	R/W	0	UART-TX reception function 0: Disable 1: Enable
0	URCHS	R/W	0	UART Channel Selection. 0: Set P04/P05 for UART Function. 1: Set P22/P23 for UART Function.

S0TBUF Register (0x99)

Bit	Field	Type	Initial	Description
7..0	S0TBUF	R/W	0x00	Action of writing data triggers UART TX communication (LSB first). Reception data is available to read by the end of packages.

S0RBUF Register (0xD9)

Bit	Field	Type	Initial	Description
7..0	S0RBUF	R/W	0x00	Action of writing data triggers UART RX communication (LSB first). Reception data is available to read by the end of packages.

PCON Register (0x87)

Bit	Field	Type	Initial	Description
7	SMOD	R/W	0	UART baud rate control **(UART mode 2)** 0: fcpu/64 1: fcpu/32 **(UART mode 1 , 3)** 0: fcpu*1 1: fcpu*2
6..0				Refer to other chapter(s)

S0RELH/S0RELL Registers (S0RELH: 0xBA, S0RELL: 0xAA)

Bit	Field	Type	Initial	Description
15..10	Reserved	R	0x00	
9..0	S0REL[9:0]	R/W	0x00	S0RELH[1:0] & S0RELL[7:0]. UART Reload Register is used for UART baud rate generation.

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
4	ETS0	R/W	0	Enable UART TX interrupt
3	ERS0	R/W	0	Enable UART RX interrupt
Else				Refer to other chapter(s)

P1OC Register (0xE4)

Bit	Field	Type	Initial	Description
6	P23OC	R/W	0	0: Switch P2.3 (URX) to input mode (required) ~~1: Switch P2.3 (URX) to open-drain mode~~*
5	P22OC	R/W	0	0: Switch P2.2 (UTX) to push-pull mode 1: Switch P2.2 (UTX) to open-drain mode
4	P05OC	R/W	0	0: Switch P0.5 (URX) to input mode (required) ~~1: Switch P0.5 (URX) to open-drain mode~~*
3	P04OC	R/W	0	0: Switch P0.5 (UTX) to push-pull mode 1: Switch P0.5 (UTX) to open-drain mode
Else				Refer to other chapter(s)

* Setting P05OC/P23OC as high causes URX cannot receive data.

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
5	P05M	R/W	0	0: Set P0.5 (URX) as input mode (required) 1: Reserved
4	P04M	R/W	0	0: Reserved 1: Set P0.4 (UTX) as output mode (required)
Else				Refer to other chapter(s)

* The URX and UTX respectively require input and output mode selection to receive/transmit data appropriately.

P0 Register (0x80)

Bit	Field	Type	Initial	Description
5	P05	R/W	0	This bit is available to read at anytime for monitoring the bus statue.
4	P04	R/W	0	0: Reserved 1: Make P0.4 (UTX) can output UART data (required)
Else				Refer to other chapter(s)

* Setting P04 initially high because UART block drive the shared pin low signal only.

P2M Register (0xFB)

Bit	Field	Type	Initial	Description
3	P23M	R/W	0	0: Set P2.3 (URX) as input mode (required) 1: Reserved
2	P22M	R/W	0	0: Reserved 1: Set P2.2 (UTX) as output mode (required)
Else				Refer to other chapter(s)

* The URX and UTX respectively require input and output mode selection to receive/transmit data appropriately.

P2 Register (0xA0)

Bit	Field	Type	Initial	Description
3	P23	R/W	0	This bit is available to read at anytime for monitoring the bus statue.
2	P22	R/W	0	0: Reserved 1: Make P2.2 (UTX) can output UART data (required)
Else				Refer to other chapter(s)

* Setting P04 initially high because UART block drive the shared pin low signal only.

21.6 Sample Code

The following sample code demonstrates how to perform UART mode 1 with interrupt.

```
#define  SYSUartSM0     (0 << 6)
#define  SYSUartSM1     (1 << 6)
#define  SYSUartSM2     (2 << 6)
#define  SYSUartSM3     (3 << 6)
#define  SYSUartREN     (1 << 4)
#define  SYSUartSMOD    (1 << 7)
#define  SYSUartES0     (1 << 4)

void SYSUartInit(void)
{
 // set UTX, URX pins' mode at here or at GPIO initialization
 P04 = 1;
 P05 = 0;
 // set P04 Output Mode and P05 Input Mode
 P0M = (P0M | 0x10) & ~0x20;

 // configure UART mode between SM0 and SM3, enable URX S0CON
 = SYSUartSM1 | SYSUartREN;

 // configure UART baud rate PCON =
 SYSUartSMODE1;
 S0RELH = 0x03; S0RELL = 0xFB;
 S0CON2 = 0xA0;
 TEN0 = 1;
 REN0 = 1;

 // enable UART interrupt IEN0 |=
 SYSUartES0;

  // global interrupt enable
 EAL = 1;

 // send first UTX data ST0BUF =
 uartTxBuf;
}

void SYSUartInterruptTx(void) interrupt ISRUartTx
{
 if (TI0) {
  ST0BUF = uartTxBuf;
  TI0 = 0;
 }
}

void SYSUartInterruptRx(void) interrupt ISRUartRx
{
 if (RI0) {
  uartRxBuf = S0RBUF; RI0 = 0;
 }
}
```

22 SPI

The serial peripheral interface, aka SPI, has two operation modes: master and slave. A master proactively outputs bus clock through SCK pin, whereas slave device(s) handle communication based on SCK pin's clock input. An optional slave select pin (SSN) can be enabled by register in slave mode.

22.1 SPI Master

The SPI master mode has seven types of clock generator from fcpu/2 to fcpu/128. Generated clock is outputted through SCK pin (shared with P1.7) and its idle status is controlled by CPOL.

The phase of data input and output is automatically specified by CPHA register. In master mode MOSI pin (shared with P1.5) plays the role of data output, and MISO pin (shared with P1.6) fetches data from slave device. A SPI communication is started by writing SPDAT register; the received data from MISO is available to read after the end of data transmission.

The master mode has two status flags with interrupt function:

SPIF register indicates the end of one byte data communication. An interrupt would be issued at the same time if ESPI bit is enabled.

MODF is issued by SSN (shared with P1.4) low status while transmission. This interrupt source can be masked by setting SSDIS bit.

22.2 SPI Slave

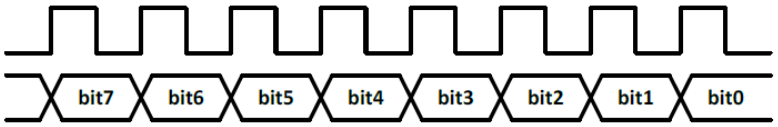
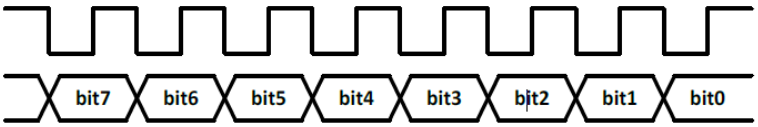
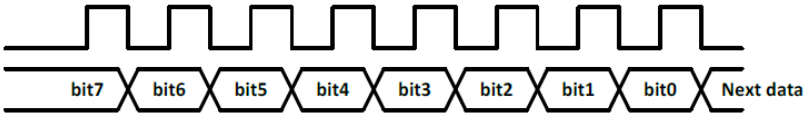
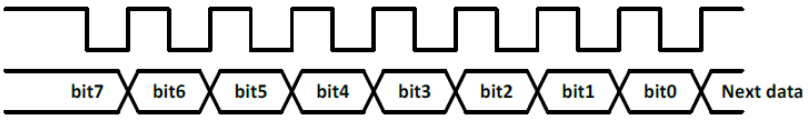
The SPI slave mode monitors SCK pin to control its MISO and MOSI communication. However, the maximum clock rate is limited at fcpu/8. Slave device(s) are expected to specify its CPOL and CPHA setting as the same configuration of the connected SPI bus.

The slave mode treats MOSI pin as its data input, and MISO pin as its data transmission. By default, the SSDIS register is low which means the slave select pin (SSN) is functional. A SPI communication would be processed if the SSN is low status. Thus, a slave device is suspended if its SSN is high status.

The slave mode has two status flags with interrupt function:
SPIF indicates the end of one byte data communication. The original SPDAT's value has been transmitted, and the received data from MOSI is ready to be read on SPDAT.
MODF indicates that the slave select pin (SSN) has turned high before a completion of one byte communication. In other word, the last time of SPI communication is broken.

22.3 SPI Operation

The table below illustrates four different setting of CPOL and CPHA, and the bus operation in each combination.

CPOL	CPHA	Diagrams	Description
0	1		SCK idle low Data latch at falling edges
1	1		SCK idle high Data latch at rising edges
0	0		SCK idle low Data latch at rising edges
1	0		SCL idle high Data latch at falling edges

22.4SPI Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPCON	SPR2	SPEN	SSDIS	MATR	CPOL	CPHA	SPR1	SPR0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
SPDAT	SPDAT7	SPDAT6	SPDAT5	SPDAT4	SPDAT3	SPDAT2	SPDAT1	SPDAT0
IEN0	EAL	-	ESAR	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
P1OC	-	-	-	P05OC	P04OC	P17OC	P16OC	P15OC
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P1	P17	P16	P15	P14	P13	P12	P11	P10

SPCON Register (0xE2)

Bit	Field	Type	Initial	Description
7,1,0	SM[2:0]	R/W	000	SPI baud rate generator (master mode only) 000: fcpu/2 001: fcpu/4 010: fcpu/8 011: fcpu/16 100: fcpu/32 101: fcpu/64 110: fcpu/128 ~~111: reserved~~
6	SPEN	R/W	0	SPI communication function 0: Disable 1: Enable
5	SSDIS	R/W	0	Slave select pin function (MSTR = 0, CPHA = 0 only) 0: Enable slave selection pin (SSN) function 1: Disable slave select pin (SSN) function
4	MSTR	R/W	1	SPI mode 0: Slave mode 1: Master mode
3	CPOL	R/W	0	SCK pin idle status 0: SCK idle low 1: SCK idle high
2	CPHA	R/W	1	Clock phase of data latch control 0: Data latched by the first of clock edge 1: Data latched by the second of clock edge

SPSTA Register (0xE1)

Bit	Field	Type	Initial	Description
7	SPIF	R	0	SPI complete communication flag Set automatically at the end of communication Cleared automatically by reading SPSTA, SPDAT registers
6	WCOL	R	0	Write collision flag Set automatically if write SPDAT during communication Cleared automatically by reading SPSTA, SPDAT registers
5	SSERR	R	0	Synchronous slave select pin error Set automatically if SSN error controlling Cleared automatically by clear SPEN
4	MODF	R	0	Mode fault flag
3..0	Reserved	R	0x00	

SPDAT Register (0xE3)

Bit	Field	Type	Initial	Description
7..0	SPDAT	R/W	0x00	Master mode: action of writing data triggers SPI communication; reception data is readable after the end of one byte communication (SPIF automatically set). Slave mode: written data would be transmitted by SCK input; reception data is available to read after the end of one byte communication (SPIF automatically set).

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
Else				Refer to other chapter(s)

IEN1 Register (0xB8)

Bit	Field	Type	Initial	Description
1	ESPI	R/W	0	Enable SPI interrupt
Else				Refer to other chapter(s)

P1OC Register (0xE4)

Bit	Field	Type	Initial	Description
2	P17OC	R/W	0	0: Switch P1.7 (SCK) to input or output mode 1: Switch P1.7 (SCK) to open-drain mod
1	P16OC	R/W	0	0: Switch P1.6 (MISO) to input or output mode 1: Switch P1.6 (MISO) to open-drain mod
0	P15OC	R/W	0	0: Switch P1.5 (MOSI) to input or output mode 1: Switch P1.5 (MOSI) to open-drain mode
Else				Refer to other chapter(s)

P1M Register

Bit	Field	Type	Initial	Description
7	P17M	R/W	0	0: Set P1.7 (SCK) as input mode [1] slave mode 1: Set P1.7 (SCK) as output mode [1] master mode
6	P16M	R/W	0	0: Set P1.6 (MISO) as input mode [2] master mode 1: Set P1.6 (MISO) as output mode [2] slave mode
5	P15M	R/W	0	0: Set P1.5 (MOSI) as input mode [3] slave mode 1: Set P1.5 (MOSI) as output mode [3] master mode
4	P14M	R/W	0	0: Set P1.4 (SSN) as input mode* 1: Set P1.4 (SSN) as output mode
Else				Refer to other chapter(s)

[1]Setting SCK as input mode is essential in slave mode; setting as output mode is recommended in master mode. [2]Setting MISO as input mode is essential in master mode; setting as output mode is recommended in slave mode. [3]Setting MOSI as input mode is essential is slave mode; setting as output mode is recommended in master mode.

*If slave mode with SSN function: essentially to set SSN as input mode.

22.5Sample Code

The following sample code demonstrates how to perform SPI Master with interrupt.

```
#define SpiMaster (1 << 4)          //SPI = Master mode
#define SpiSlave (0 << 4)           //SPI = Slave mode
#define SpiMode0 (0 << 2)           //SCK idle low, data latch at rising edge
#define SpiMode1 (1 << 2)           //SCK idle low, data latch at falling edge
#define SpiMode2 (2 << 2)           //SCK idle high, data latch at falling edge
#define SpiMode3 (3 << 2)           //SCK idle high, data latch at rising edge
#define SpiEn (1 << 6)              //Enable SPI
#define SpiSSNEn (0 << 5)           //SSN pin function enable
#define SpiSSNDis (1 << 5)          //SSN pin function disable


unsigned char u8SpiData = 0;        // data buffer
unsigned char u8TxCompleted = 0;


void SpiMaster(void)
{
   //SCK & MOSI = output, MISO = input
   P1M |= 0x14;
   //Enable Spi, Master mode, SSN pin disable, Fclk/128
   //SCK idle low, data latch at falling edge
   SPCON = SpiEn | SpiMaster | SpiMode1 | SpiSSNDis | 0x82;
   //Enable Global/SPI interrupt
   while (1) {
     SPDAT = 0x55;
     while(!u8TxCompleted);         // wait end of transmition
     u8TxCompleted = 0;             // clear sw flag
     u8RcvData = u8SpiData;         // receive 0x66
     SPDAT = 0x99;
     while(!u8TxCompleted);         // wait end of transmition
     u8TxCompleted = 0;             // clear sw flag
     u8RcvData = u8SpiData;         // receive 0xAA
   }
}


void SpiInterrupt(void) interrupt ISRSpi //0xA3
{
   switch ( SPSTA )                  // Clear SPI flag (SPIF) by reading
   {
     case 0x80:
     u8SpiData = SPDAT;
     u8TxCompleted = 1;
     break;

     case 0x10:
     // Mode Fault
     break;
   }
}
```

23 I2C

The I2C is a serial communication interface for data exchanging from one MCU to one MCU or other hardware peripherals. The device can transmit data as a master or a slave with two bi-directional IO, SDA (Serial data output) and SCL (Serial clock input).

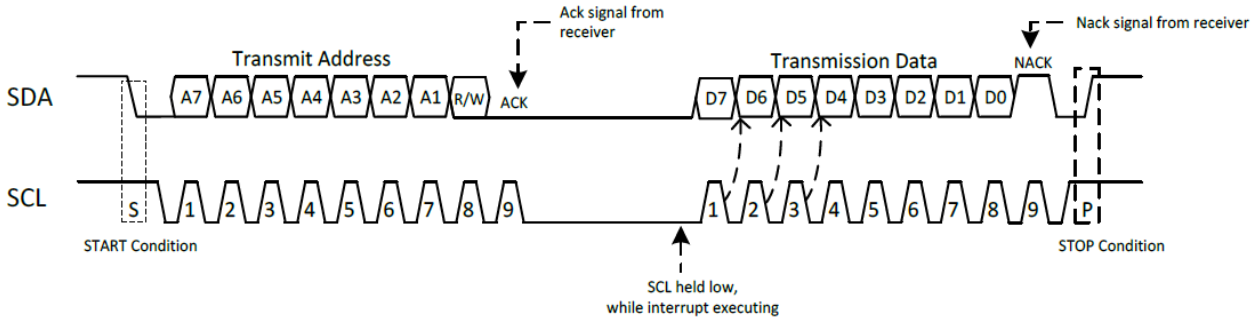
When a master transmit data to a slave, it's called "WRITE" operation; when a slave transmit data to a master, it's called "READ" operation. It also supports multi-master communication and keeps data transmission correctly by an arbitration method to decide one master has the control on bus and transmit its data.

23.1 I2C Protocol

I2C transmission structure includes a START(S) condition, 8-bit address byte, one or more data byte and a STOP (P) condition. START condition is generated by master to initial any transmission.

Data is transmitted with the Most Significant Bit (MSB) first. In address byte, the higher 7-bit is address bit and the lowest bit is data direction (R/W) bit. When R/W=0, it assigns a "WRITR" operation. When R/W=1, it assigns a "READ" operation.

After each byte is received, the receiver (a master or a slave) must send an acknowledge (ACK). If transmitter can't receive an ACK, it will recognize a not acknowledge (NACK). In WRITE operation, the master will transmit data to the slave and then waits for ACK from slave. In READ operation, the slave will transmit data to the master and then waits for ACK from master. In the end, the master will generate a STOP condition to finish transmission.



23.2 I2C Transfer Modes

The I2C can operate as a master/slave to execute the 8-bit serial data transmission/reception operation. Thus, the module can operate in one of four modes: Master Transmitter, Master Receiver, Slave Transmitter and Slave Receiver.

23.3 Master Transmitter Mode

The master transmits information to the slave. The serial data is output via SDA while the serial clock is output on SCL. Data transmission starts via generate a START(S) signal. After the START signal, the specific address byte of slave device is sent. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “0” to enable the master transmission. In the following, the master transmits one or more data byte to the slaver. After each data is transmitted, the master waits for the acknowledge (ACK) from the slave. In the end, the master generates a STOP (P) signal to terminate the data transmission.

23.4 Master Receiver Mode

The master receives the information from the slave. The serial data input via SDA while the serial clock output on SCL. Data reception starts via generate a START(S) signal. After the START signal, the specific address byte of slave device is sent. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “1” to enable the master reception. In the following, the master receives one or more data byte from the slaver. After each data is received, the master generates the acknowledge (ACK) or not acknowledge (NACK) to the slave via the status of AA bit. In the end, the master generates a STOP (P) signal to terminate the data transmission.

23.5 Slave Transmitter Mode

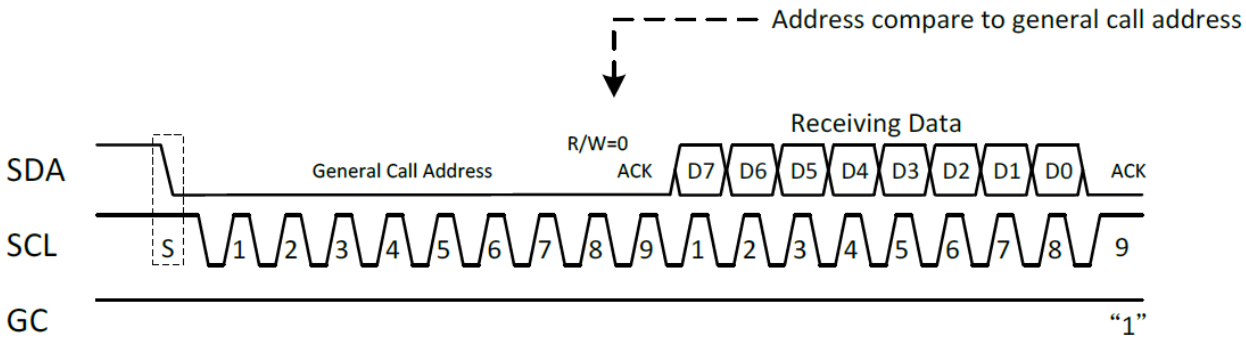
The slave transmits information to the master. The serial data output via SDA while the serial clock input on SCL. Data transmission starts via receive a START(S) signal from the master. After the START signal, the specific address byte of slave device is received. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “1” to enable the slave transmission. If the received address byte match the address in I2CADR register, the slave generate an acknowledge (ACK). Otherwise, if general call address condition is set (GC=1), the slave also generate an acknowledge (ACK) after general call address (0x00) is received. In the following, the slave transmits one or more data byte to the master. After each data is transmitted, the slave waits for the acknowledge (ACK) from the master. In the end, the slave receives a STOP (P) signal from the master to terminate the data transmission.

23.6 Slave Receiver Mode

The slave receives information from the master. Both the serial data and the serial clock are input on SDA and SCL. Data reception starts via receive a START(S) signal from the master. After the START signal, the specific address byte of slave device is received. The address byte includes 7-bit address bit and an 8th data direction (R/W) bit. The R/W is set “0” to enable the slave reception. If the received address byte match the address in I2CADR register, the slave generate an acknowledge (ACK). Otherwise, if general call address condition is set (GC=1), the slave also generate an acknowledge (ACK) after general call address (0x00) is received. In the following, the slave receives one or more data byte from the master. After each data is receives, the slave generates the acknowledge (ACK) or not acknowledge (NACK) to the master via the status of AA bit. In the end, the slave receives a STOP (P) signal from the master to terminate the data transmission.

23.7 General Call Address

In I2C bus, the first 7-bit is the slave address. Only the address matches slave address, 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 (ACK). 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 GC bit. Set this bit will enable general call address and clear it will disable. When GC=1, the general call address will be recognized. When GC=0, the general call address will be ignored.



23.8 Serial Clock Generator

In master mode, the SCL clock rate generator’s is controlled by CR[2:0] bit of I2CCON register. When CR[2:0]=000~110, SCL clock rate is from internal clock generator.

$$\text{SCL Clock Rate} = \frac{F_{CPU}}{\text{Prescaler}} \ (\text{Prescaler} = 256 \sim 60)$$

When CR[2:0]=111, SCL clock rate is from TC2 Timer overflow rate .

$$\text{SCL Clock Rate} = \frac{\text{Timer Overflow}}{8}$$

The table below shows the clock rate under different setting.

CR2	CR1	CR0	I2C	Bit Frequency (kHz)			
			Prescaler	2MHz	4MHz	8MHz	16MHz
0	0	0	256	7.8	15.6	31.3	62.5
0	0	1	224	8.95	17.9	35.7	71.4
0	1	0	192	10.4	20.8	41.7	83.3
0	1	1	160	12.5	25	50	100
1	0	0	960	2.1	4.2	83.3	16.7
1	0	1	120	16.7	33.3	66.7	133.3
1	1	0	60	33.3	66.7	133.3	266.7
1	1	1	(TC2 Timer overflow rate)/8				

***Example**: When CR[2:0]=111, SCL clock rate is from TC2 Timer overflow rate . The clock rate 400k Hz is setting as following equation. (Fcpu = 16M Hz)

```
TC2CH = 0x0FF;      // clock rate 400k Hz = 1/0.3125us/8
TC2CL = 0x0FB;      // 0.3125us Overflow
TC2RH = 0x0FF;      // auto-reload value
TC2RL = 0x0FB;      //
TC2M |= 0x0F0;      // TC2 ENABLE, TC2 timer clock= Fpcu/1

I2CCON |= 0x83;     // SPR[2:0]:111,SCL source from TC2 Timer
I2CCON |= 0x40;     // I2C enable (ENS1)
```

23.9 Synchronization and Arbitration

In multi-master condition, more than one master may transmit on bus in the same time. It must be decided which master has the control of bus and complete its transmission. Clock synchronization and arbitration are used to configure multi-master transmission. Clock synchronization is executed by synchronizing the SCL signal with anther devices.

When two masters want to transmit data in the same, the clock synchronization will start by the High to Low transition on the SCL. If master 1 clock set LOW first, it holds the SCL in LOW status until the clock transit to HIGH status. However, if anther master clock still keep LOW status, the Low to High transition of master 1 may not change SCL status (SCL keep LOW). In the other word, SCL keep LOW by the master with the longest clock time in LOW status. The SCL will transit from LOW to HIGH when the all devices clock transit to HIGH status. In the duration, the master1 will keep in HIGH status and wait for SCL transition (from LOW to HIGH), then continue its transmission. After clock synchronization, all devices clock and SCL clock are the same. Arbitration is used to decide which master can complete its transmission by SDA signal. Two masters may send out a START condition and transmit data on bus in the same time. They may influence by each other. Arbitration will force one master to lose the control on bus. Data transmission will keep until master output different data signal. If one master transmits HIGH status and anther master transmits LOW status, the SDA

will be pull low. The master output High will detect the different with SDA and lose the control on bus. The mater with LOW status wins the bus control and continues its transmission. There is no data miss during arbitration.

23.10 System Management Bus Extension

The optional System Management Bus (SMBus) protocol hardware supports 3 types timeout detection: (1) Tmext Timeout Detection: The cumulative stretch clock cycles within one byte. (2)Tsext Timeout Detection: The cumulative stretch clock cycles between start and stop condition. (3)Timeout Detection: The clock low measurement.

Timeout detection is controlled by SMBSEL and SMBDST registers. The SMBEXE bit of SMBSEL is SMBus extension function enable bit. When SMBEXE=1, SMBus extension function is enabled. Otherwise, Disable SMBus extension function. Timeout type and period setting is controlled by SMBTOP[2:0] and SMBDST. The period of SMBus timeout is controlled by three 16-bit buffers of Tmex, Tsext and Tout. The equation is as following.

$$\text{Tmext/Tsex t/Tout} = \frac{\text{Timer Peri od(sec)} * F_{CPU}(\text{Hz})}{1024}$$

Tmext is support by two 8-bit register of Tmext_L and Tmext_H . Tmext_L hold the low byte and Tmext_H hold high byte. Tsext is support by two 8-bit register of Tsext_L and Tsext_H . Tsext_L hold the low byte and Tsext_H hold high byte. Tout is support by two 8-bit register of Tout_L and Tout_H . Tout_L hold the low byte and Tout_H hold high byte.

Type	Time out period	Fcpu=32MHz	
		DEC	HEX
Tmext	5ms	157	9D
Tsext	25ms	782	30E
Tout	35ms	1094	446

By the setting of SMBTOP[2:0] to choose register type (as the table below), and write to register by write data to SMBDST register.

SMBTOP[2:0]	SMBDST	Description
000	Tmext_L	Select the low byte of Tmext register.
001	Tmext_H	Select the high byte of Tmext register.
010	Tsext_L	Select the low byte of Tsext register.
011	Tsext_H	Select the high byte of Tsext register.
100	Tout_L	Select the low byte of Tout register.
101	Tout_H	Select the high byte of Tout register.

When the SMBus extension function is enabled the lower 3-bit of I2CSTA hold the information about time out as the table below.

I2CSTA	Description
XXXX X000	No timeout errors.
XXXX XXX1	Tout timeout error.
XXXX XX1X	Tsext timeout error.
XXXX X1XX	Tmext timeout error.

23.11 I2C Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
I2CDAT	I2CDAT7	I2CDAT6	I2CDAT5	I2CDAT4	I2CDAT3	I2CDAT2	I2CDAT1	I2CDAT0
I2CADR	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	GC
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0
I2CSTA	I2CSTA7	I2CSTA6	I2CSTA5	I2CSTA4	I2CSTA3	I2CSTA2	I2CSTA1	I2CSTA0
SMBSEL	SMBEXE	-	-	-	-	SMBSTP2	SMBSTP1	SMBSTP0
SMBDST	SMBD7	SMBD6	SMBD5	SMBD4	SMBD3	SMBD2	SMBD1	SMBD0
IEN0	EAL	-	-	ES0	-	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P0	P07	P06	P05	P04	P03	P02	P01	P00

I2CDAT Register (0xDA)

Bit	Field	Type	Initial	Description
7:0	I2CDAT[7:0]	R/W	0x00	The I2CDAT register contains a byte to be transmitted through I2C bus or a byte which has just been received through I2C bus. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of byte shifting. The I2CDAT register is not shadowed or double buffered so the user should only read I2CDAT when an I2C interrupt occurs.

I2CADR Register (0xDB)

Bit	Field	Type	Initial	Description
7:1	I2CADR[6:0]	R/W	0x00	I2C slave address
0	GC	R/W	0	General call address (0X00) acknowledgment 0: ignored 1: recognized

I2CCON Register (0xDC)

Bit	Field	Type	Initial	Description
7,1,0	CR[2:0]	R/W	0	I2C clock rate 000: fcpu/256 001: fcpu/224 010: fcpu/192 011: fcpu/160 100: fcpu/960 101: fcpu/120 110: fcpu/60 111: (TC2 Timer overflow rate)/8
6	ENS1	R/W	0	I2C functionality 0: Disable 1: Enable
5	STA	R/W	0	START flag 0: No START condition is transmitted. 1: A START condition is transmitted if the bus is free.
4	STO	R/W	0	STOP flag 0: No STOP condition is transmitted. 1: A STOP condition is transmitted to the I2C bus in
3	SI	R/W	0	Serial interrupt flag The SI is set by hardware when one of 25 out of 26 possible I2C states is entered. The only state that does not set the SI is state F8h, which indicates that no relevant state information is available. The SI flag must be cleared by software. In order to clear the SI bit, '0' must be written to this bit. Writing a '1' to SI bit does not change value of the SI.
2	AA	R/W	0	Assert acknowledge flag 0: A NACK will be returned when a byte has received 1: An ACK will be returned when a byte has received

I2CSTA Register (0xDD)

Bit	Field	Type	Initial	Description
7:3	I2CSTA[7:3]	R	11111	I2C Status Code
2..0	I2CSTA[2:0]	R	000	SMBus Status Code

I2C status code and status

Mode	Status Code	Status of the I2C	Application software response					Next action taken by I2C hardware
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
Master Transmitter/ Receiver	08H	A START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R/W will be transmitted; ACK will be received
	10H	A repeated START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R/W will be transmitted; ACK will be received
			Load SLA+W					SLA+W will be transmitted; I2C will be switched to MST/TRX mode.
Master Transmitter	18H	SLA+W has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
			No action	1	0	0	X	Repeated START will be transmitted.
			No action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
			No action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
	20H	SLA+W has been transmitted; not ACK has been received	Load data byte*	0	0	0	X	Data byte will be transmitted; ACK will be received.
			No action	1	0	0	X	Repeated START will be transmitted.
			No action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
			No action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
	28H	Data byte in I2CDAT has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
			No action	1	0	0	X	Repeated START will be transmitted.
			No action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
			No action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
	30H	Data byte in I2CDAT has been transmitted; not ACK has been received	Load data byte*	0	0	0	X	Data byte will be transmitted; ACK will be received.
			No action	1	0	0	X	Repeated START will be transmitted.
			No action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
			No action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
Master Receiver	40H	SLA+R has been transmitted; ACK has been received	No action	0	0	0	0	Data byte will be received; not ACK will be returned
			No action	0	0	0	1	Data byte will be received; ACK will be returned
	48H	SLA+R has been transmitted; not ACK has been received	No action	1	0	0	X	Repeated START condition will be transmitted
			No action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset
			No action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset
	50H	Data byte has been received; ACK has been returned	Read data byte	0	0	0	0	Data byte will be received; not ACK will be returned
			Read data byte	0	0	0	1	Data byte will be received; ACK will be returned
	58H	Data byte has been received; not ACK has been returned	Read data byte	1	0	0	X	Repeated START condition will be transmitted
			Read data byte	0	1	0	X	STOP condition will be transmitted; STO flag will be reset
			Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset

Mode	Status Code	Status of the I2C	Application software response					Next action taken by I2C hardware
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
Slave Receiver	60H	Own SLA+W has been received; ACK has been returned	No action	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
	68H	Arbitration lost in SLA+R/W as master; own SLA+W has been received, ACK returned	No action	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
	70H	General call address (00H) has been received; ACK has been returned	No action	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
	78H	Arbitration lost in SLA+R/W as master; general call address has been received, ACK returned	No action	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
	80H	Previously addressed with own SLV address; DATA has been received; ACK returned	Read data byte	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
		Previously addressed with own	Read data byte	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address
			Read data byte	0	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized

	88H	SLA; DATA byte has been received; not ACK returned	Read data byte	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free
			Read data byte	1	0	0	1	Switched to not addressed SLV mode; own SLA or general

								call address will be recognized; START condition will be transmitted when the bus becomes free
	90H	Previously addressed with general call address; DATA has been received; ACK returned	Read data byte	X	0	0	0/1	Data byte will be received and not ACK/ACK will be returned
	98H	Previously addressed with general call address; DATA has been received; not ACK returned	Read data byte	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address
			Read data byte	0	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized
			Read data byte	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free
			Read data byte	1	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free
	A0H	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX	No action	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address
			No action	0	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized
			No action	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free
			No action	1	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free
Slave Transmitter	A8H	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
			Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
	B0H	Arbitration lost in SLA+R/W as master; own SLA+R has been received, ACK has been returned.	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
			Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
	B8H	Data byte has been transmitted; ACK will be received.	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
			Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
	C0H	Data byte has been transmitted; not ACK has been received.	No action	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address.
			No action	0	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized.
			No action	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
			No action	1	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
	C8H	Last data byte has been transmitted; ACK has been received.	No action	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address.
			No action	0	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized.
			No action	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
			No action	1	0	0	1	Switched to not addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
Miscellaneous	F8H	No relevant state information available; SI=0	No action	No action				Wait or proceed current transfer
	38H	Arbitration lost	No action	0	0	0	X	I2C will be released; A start condition will be transmitted.
			No action	1	0	0	X	When the bus becomes free. (enter to a master mode)
	00H	Bus error during MST or selected slave modes	No action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and I2C is switched to the not addressed SLV mode. STO flag is reset.

"SLA" means slave address, "R" means R/W=1, "W" means R/W=0

*For applications where NACK doesn't mean the end of communication.

SMBSEL Register (0xDE)

Bit	Field	Type	Initial	Description
7	SMBEXE	R/W	0	SMBus extension functionality 0: Disable 1: Enable
2..0	SMBSTP[2:0]	R/W	000	SMBus timeout register

SMBDST Register (0xDF)

Bit	Field	Type	Initial	Description
7..0	SMBD[7:0]	R/W	0x00	This register is used to provide a read/write access port to the SMBus timeout registers. Data read or written to that register is actually read or written to the Timeout Register which is pointed by the SMBSEL register.

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
Else				Refer to other chapter(s)

IEN1 Register (0xB8)

Bit	Field	Type	Initial	Description
0	EI2C	R/W	0	Interrupts enable. Refer to Chapter Interrupt
Else				Refer to other chapter(s)

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
3	P03M	R/W	0	0: Set P0.3 (SDA) as input mode (required) ~~1: Set P0.3 (SDA) as output mode~~*
2	P02M	R/W	0	0: Set P0.2 (SCL) as input mode (required) ~~1: Set P0.2 (SCL) as output mode~~*
Else				Refer to other chapter(s)

* The P02M and P03M respectively require be set input mode.

23.12 Sample Code

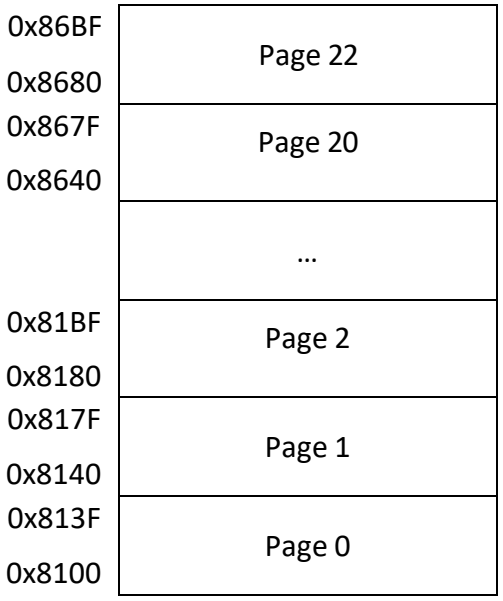
The following sample code demonstrates how to perform I2C with interrupt.

```
unsigned int I2CAddr;
unsigned int I2C_TXData0;
unsigned int I2C_TXDatan;
unsigned int I2C_RXData0;
unsigned int I2C_RXDatan;


void I2C_Init(void)
{
    P02 = 0;
    P03 = 0;
    P0M |= 0x00;            // P02 & P03 as input

    I2CCON |= 0x40;         // I2C enable (ENS1)
    I2CCON |= 0x82;         // Clock rate bit : 110b ; 266.7KHz at 16MHz

    EI2C = 1;               // I2C interrupt enable
    EAL = 1;                // Interrupt enable
}


void I2C_ISR(void) interrupt ISRI2c     // Vector @  0xAB
{
  switch (I2CSTA)
  {
    // tx mode
    case 0x08:
        I2CCON &= 0xDF;     // START (STA) = 0
        I2CDAT = I2CAddr;   // Tx/Rx addr
        break;

    case 0x18:              // write first byte
        I2CDAT = I2C_TXData0;
        break;

    case 0x28:              // write n byte
        I2CDAT = I2C_TXDatan;
        break;

    case 0x30:              // STOP (STO)
        I2CCON |= 0x10;
        break;

    // rx mode
    case 0x40:              // get slave addr
        I2CCON |= 0x04;     // AA = 1
        break;

    case 0x50:              // read n byte
        I2C_RXData0 = I2CDAT;
        I2CCON &= 0xFB;     // AA = 0
        break;


```

```
55
56      case 0x58:              // read last byte & stop
57          I2C_RXDatan = I2CDAT;
58          I2CCON |= 0x10;     // STOP (STO)
59          break;
60
61      default:
62          I2CCON |= 0x10;     // STOP (STO)
63    }
64    I2CCON &= 0xF7;           // Clear I2C flag (SI)
65  }
66
67
```

24 In-System Program

SN8F5840 builds in an on-chip 1.5KB ISP Data program memory, The ISP DATA ROM which is equally divided to 23 pages (64 bytes per page), The in-system program is a procedure that enables a firmware to freely modify every page's data or each byte data of page; in other word, it is the way to store value(s) into the non-volatile memory and/or live update firmware.



ISP Data Program memory
(ISP Data ROM)

24.1 Page Program

Because each page of the program memory has 64 bytes in length, a page program procedure requires 64 bytes IRAM as its data buffer.

As an example, assume the 0th page of program memory (ISP DATA ROM, 0x8100 – 0x813F) is the anticipated update area; the content is already stored in IRAM address 0x60 – 0x9F. To perform the in-system program, simply write starting IROM address 0x8100 to EPROMH/EPROML registers, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0x5A' into PECMD [7:0] registers to duplicate the buffer's data to 0th page of ISP DATA ROM.

24.2 Byte Program

SN8F5840 series support byte program function of all program memory area, a byte program procedure requires 1-bytes IRAM as its data buffer.

As an example, assume the 3rd byte of 0th page of program memory (IROM, 0x8102) is the anticipated update byte; the content is already stored in IRAM address 0x60. To perform the in-system program, simply write starting IROM page address 0x8100 to EPROMH/EPROML registers, writer byte address 0x02 to PEBYTE register, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0x1E' into PECMD [7:0] registers to duplicate the buffer's data to 3rd byte of 0th page of IROM.

24.3 In-system Program Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAM	RAM7	RAM6	RAM5	RAM4	RAM3	RAM2	RAM1	RAM0
PEROMH	ROM15	ROM14	ROM13	ROM12	ROM11	ROM10	ROM9	ROM8
PEROML	ROM7	ROM6	PEBYTE5	PEBYTE4	PEBYTE	PEBYTE2	PEBYTE1	PEBYTE0
PECMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0

PECMD Register (0x94)

Bit	Field	Type	Initial	Description
7..0	PECMD[7:0]	W	-	0x5A : Start page-program procedure 0x1E : Start byte-program procedure. 0x4E : Start four byte-program procedure. * After set PECMD[7:0], then must write 'NOP' instruction twice. Please reference24.4 sample code.

* Not permitted to write any other to PECMD register.

PEROML Register (0x95)

Bit	Field	Type	Initial	Description
7..6	PEROM[7:6]	R/W	00	The first address (7^{th} – 6^{th}) of program page (IROM)
5..0	PEBYTE[5:0]	R/W	0x00	The Byte address of a page.

PEROMH Register (0x96)

Bit	Field	Type	Initial	Description
7..0	PEROM[15:8]	R/W	0x00	The first address (15^{th}– 8^{th} bit) of program page

PERAM Register (0x97)

Bit	Field	Type	Initial	Description
7..0	PERAM[7:0]	R/W	0x00	The first address of data buffer (IRAM)

24.4 Sample Code

The following sample code demonstrates how to perform ISP .

```
#include <intrins.h>    // for _nop_
#include <SN8F5849.h>
#include "GenericTypeDefs.h"
#define CBYTE ((unsigned char volatile code  *) 0)
void ISPSetROMAddr(UINT16 u16addr);
void ISPSetRAMAddr(UINT8 u8addr);
void ISPWritePage(void);
void ISPWriteByte(void);
void ISPExecute_PAGE(int Start_addr);
void ISPExecute_Byte(int Start_addr,unsigned char data_byte);
UINT8 ISPExecute_Flash_Byte_Read(UINT32 Read_addr);
void main(void)
{

    UINT8 Main_Read_Data;

    ISPExecute_PAGE(0x4700);           // Program one page by ISP
    ISPExecute_PAGE(0x3700);           // Program one page by ISP

    ISPExecute_Byte(0x4701,0xA5);      // Program one Byte by ISP
    ISPExecute_Byte(0x3781,0x5A);      // Program one Byte by ISP

    Main_Read_Data = ISPExecute_Flash_Byte_Read(0x4701);
    Main_Read_Data = ISPExecute_Flash_Byte_Read(0x3781);

    while (1) {
        WDTR = 0x5A;       // clear watchdog if watchdog enable

        // To Do ...
    }
}
void ISPExecute_PAGE(int Start_addr)
{
    UINT8 idata u8data[64] = {0};
    UINT8 idata i =0;

    // step 1 : Get data
    for (i =1; i<65; i++)
      u8data[i-1] = i;     // write data for test

    // step 2 : Set RAM addr of data
    i = u8data;            // get start addr

    ISPSetRAMAddr(i);
    // step 3 : Set ROM start addr (Range is 0x0000~0x7FC0)
    ISPSetROMAddr(Start_addr);

    // step 4 : Progarm one page (64 bytes)
    ISPWritePage();
}
```

```
55 void ISPExecute_Byte(int Start_addr,unsigned char data_byte)
56 {
57     UINT8 idata u8data_byte[1] = {0};
58     UINT8 idata j =0;
59
60     // step 1 : Get data
61     u8data_byte[0] = data_byte; // write data for test
62
63     // step 2 : Set RAM addr of data
64     j = u8data_byte;            // get start addr
65     ISPSetRAMAddr(j);
66
67     // step 3 : Set ROM start addr (Range is 0x8100~0x86BF)
68     ISPSetROMAddr(Start_addr);
69
70     // step 4 : Progarm one byte (1 byte)
71     ISPWriteByte();
72 }
73
74 void ISPSetROMAddr(UINT16 u16addr)
75 {
76     // set ROM addr
77     // ISP Target Start ROM address Low byte
78     PEROML |= ((u16addr & 0x00FF));
79     // ISP Target Start ROM address High byte
80     PEROMH |= ((u16addr & 0xFF00)>>8);
81     _nop_();
82
83 }
84
85 void ISPSetRAMAddr(UINT8 u8addr)
86 {
87     // set RAM addr
88     PERAM = u8addr;
89 }
90
91 void ISPWritePage(void)
92 {
93     // execute Page Write
94
95     if(EAL == 0) {
96       PECMD = 0x5A;
97       _nop_(); _nop_();
98     }
99     else {
100      EAL = 0;
101      PECMD = 0x5A;
102      _nop_(); _nop_();
103      EAL = 1;
104    }
105 }
106
107
108
109
```

```
110
111 void ISPWriteByte(void)
112 {
113     // execute Byte Write
114
115     if(EAL == 0) {
116       PECMD = 0x1E;
117       _nop_(); _nop_();
118     }
119     else {
120       EAL = 0;
121       PECMD = 0x1E;
122       _nop_(); _nop_();
123       EAL = 1;
124     }
125 }
126
127 UINT8 ISPExecute_Flash_Byte_Read(UINT32 Read_addr)
128 {
129     UINT8 Read_Data = 0xFF;
130     Read_Data = CBYTE[Read_addr];
131
132     return Read_Data;
133 }
```

25 CRC

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-bit data word and a generator polynomial. Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. The CRC calculation circuit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location

- Support
  CRC-16 polynomial: X16+X15+X2+1
  CRC-16-CCITT polynomial: X16+X12+X5+1
- Handles 8-bit data size
- Single input/output 8-bit data register
- Input buffer to avoid bus stall during calculation
- 32KB ROM calculating time is 32ms.

25.1CRC Calculation Procedure

The CRC generator supports 2 calculation modes to obtain CRC code including EEPROM auto-calculation mode and data calculation mode.

EEPROM auto-calculation mode procedure:

1. SupportSelect CRC execute range by CRCROM bit. If CRCROM = 0, the CRC executes range is 0x0000~0x3FFF. Otherwise, the CRC executes range is 0x0000~0x7FFF.
2. Select CRC polynomial by CRCPOL bit. When CRCPOL=0, CRC-16-CCITT (X16+X12+X5+1) is selected. Otherwise, the CRC-16(X16+X15+X2+1) is selected.
3. Set CRCRST bit to reset initial seed value/ CRC data buffer and BUSY bit to 0.
4. Set URCRCEN bit to start CRC calculation and check BUSY bit.
5. CRC calculation is finished until BUSY bit from 1 to 0. Read CRC code from CRCDATH/L.

Data calculation mode procedure:

1. Select CRC polynomial by CRCPOL bit. When CRCPOL=0, CRC-16-CCITT (X16+X12+X5+1) is selected. Otherwise, the CRC-16(X16+X15+X2+1) is selected.
2. Set CRCRST bit to reset initial seed value/ CRC result and BUSY bit to 0.
3. Key in data to CRCDATL.
4. Set URCRCEN bit to start CRC calculation and check BUSY bit.
5. CRC calculation is finished until BUSY bit from 1 to 0. Read CRC code from CRCDATH/L

25.2 CRC REGISTERS

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCM	URCRCSTA	BUSY	CRCRST	CRCPOL	CRCROM1	CRCROM0	-	CRCEN
CRCDATL	CRCDAT7	CRCDAT6	CRCDAT5	CRCDAT4	CRCDAT3	CRCDAT2	CRCDAT1	CRCDAT0
CRCDATH	CRCDAT15	CRCDAT14	CRCDAT13	CRCDAT12	CRCDAT11	CRCDAT10	CRCDAT9	CRCDAT8

CRCM Register (0x88)

Bit	Field	Type	Initial	Description
7	URCRCEN	R/W	0	Enable bit of the CRC calculation for the 16KB/32KB User ROM 0: Disable 1: Enable
6	BUSY	R/W	0	Disable. Stop/Finish operation 0: CRC calculation finished. 1: CRC calculation is in process.
5	CRCRST	R/W	0	Reset bit. 0: No effect. 1: Reset the CRC calculation circuit.
4	CRCPOL	R/W	0	CRC Polynomial 0: CRC-16-CCITT 1: CRC-16
3..2	CRCROM [1:0]	R/W	0x00	CRC EEPROM calculation range. 00: 0x0000~0x3FFF (1/2 ROM). 01: 0x0000~0x1FFF (1/4 ROM). 10: 0x0000~0x7FBD 11:0x0000~0x77FF
0	CRCEN	R/W	0	Enable bit of the CRC IP 0: Disable. 1: Enable.

CRCDATL Register (0x89)

Bit	Field	Type	Initial	Description
7..0	CRCDAT[7:0]	R/W	0x00	Low byte of 16-bit CRC data.

CRCDATH Register (0x8A)

Bit	Field	Type	Initial	Description
7..0	CRCDAT[15:8]	R/W	0x00	High byte of 16-bit CRC data.

25.1Sample Code

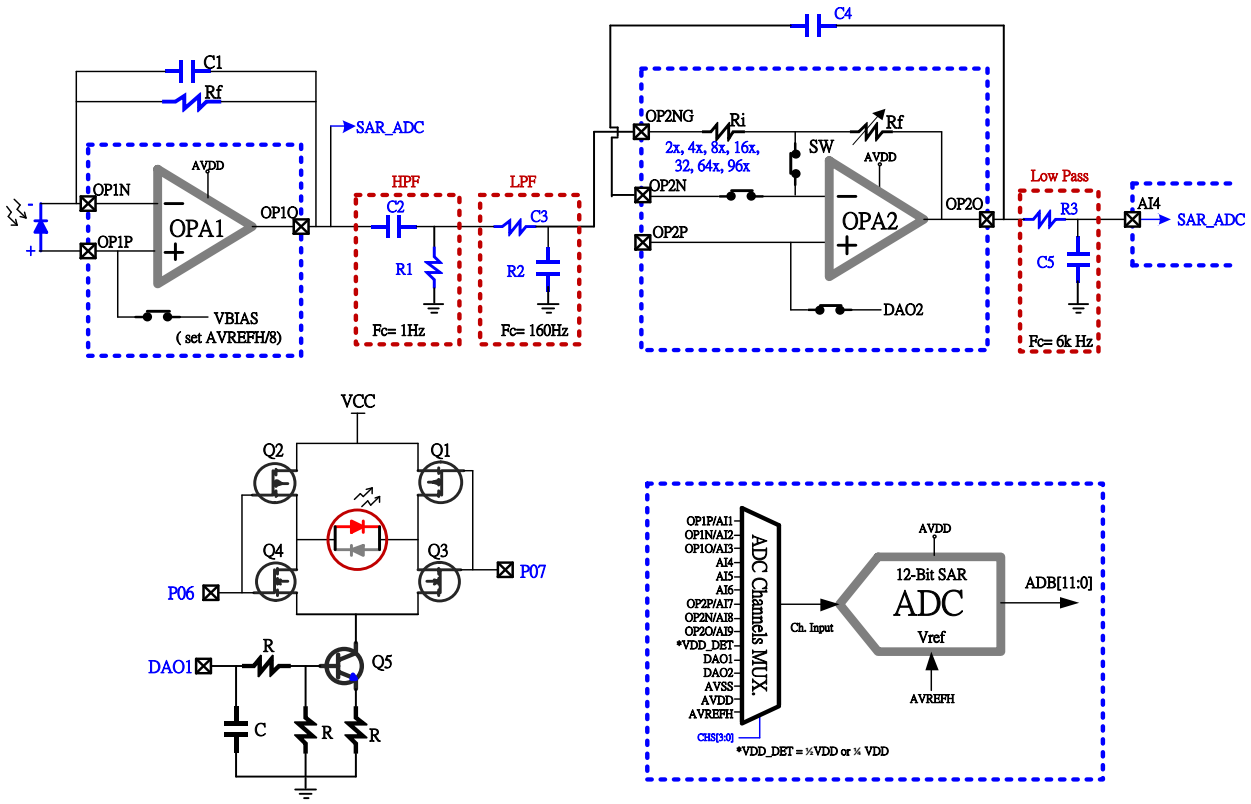
The following sample code demonstrates how to perform CRC-CCITT.

```
#include <intrins.h>          //for _nop_
#include <SN8F5849.h>

void CRCInit(void)
{
  CRCM = CRCM & 0x00;          //CRC area 0x0000~0x3FFF (1/2 ROM).
  CRCM = CRCM | 0x01;          //CRCEN = 1 Enable CRC function
  CRCM = CRCM & 0xEF;          //CRCPOL = 0 CCITT Mode
  CRCM = CRCM | 0x20;          //CRCRST = 1 initial Seed value and BUSY
}

void main(void)
{
  CRC_init();
  CRCM = CRCM | 0x80;           //URCRCSTA = 1 Start CRC function
   while((CRCM && 0x40)== 0x40){ //if BUSY==1 the CRC function is finished
   }

  CRC_result_data = ((0X00FF & CRCDATH)<<8) | CRCDATL;
  nop_();                       //CRC-CCITT 1/2 ROM Data finished

   while (1) {
         WDTR = 0x5A;            // clear watchdog if watchdog enable
         // To Do ...
     }
}
```

26 APPLICATION CIRCUIT

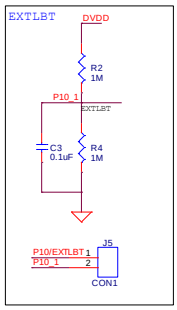
26.1 Pulse oximeter application circuit

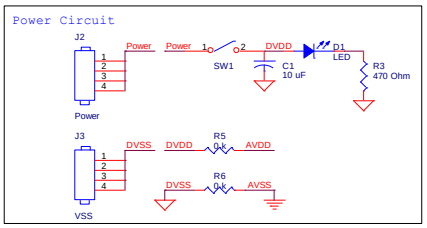
Pulse Oximeter Application Circuit

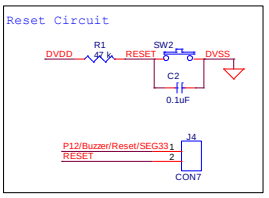


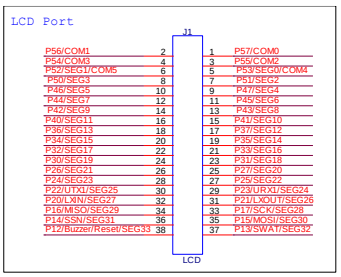
✸ ***Note: DVDD/AVDD capacitors should be as close as possible with pins of IC***

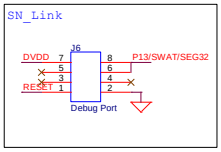
26.2 Starter-Kit BOARD CIRCUIT

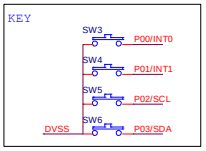


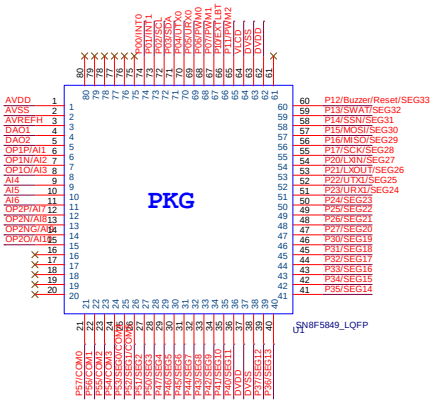


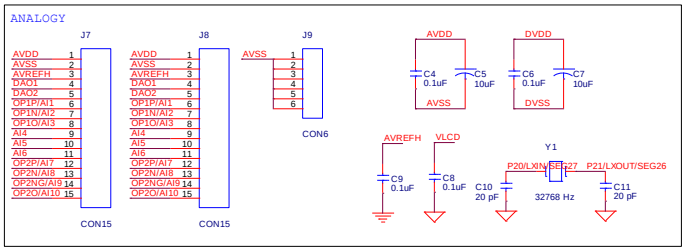


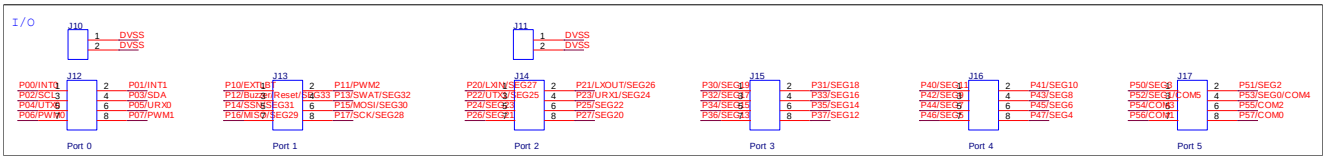












27 Electrical Characteristics

27.1 Absolute Maximum Ratings

Voltage applied at VDD to VSS ..... - 0.3V to 3.6V
Voltage applied at any pin to VSS..... - 0.3V to VDD+0.3V
Operating ambient temperature..... -40°C to 85°C
Storage ambient temperature ..... -40°C to 125°C

27.2 System Operation Characteristics

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	Operating voltage	fcpu = 1MHz	2.0		3.6	V
V_{DR}	RAM data retention Voltage		1.5			V
V_{POR}	VDD rising rate*		0.05			V/ms
I_{DD1}	Normal mode supply current	VDD = 2V ~ 3.6V, fcpu = 0.25MHz		1.3		mA
		VDD = 2V ~ 3.6V, fcpu = 1MHz		1.7		mA
		VDD = 2V ~ 3.6V, fcpu = 4MHz		1.9		mA
		VDD = 2V ~ 3.6V, fcpu = 8MHz		2.1		mA
		VDD = 2V ~ 3.6V, fcpu = 16MHz		5.5		mA
I_{DD2}	STOP mode supply current	VDD =3V		2.0		µA
		VDD = 3V, RTC enable (32768Hz Crystal)		3.0		µA
I_{DD3}	IDLE mode supply current (fcpu = 0.25MHz)	VDD = 3V		0.62		mA
f_{IHRC}	Internal high clock generator	VDD = 2.0V ~ 3.6V, Temp. 25°C	-1%	32	+1%	MHz
		VDD = 2.0V to 3.6V, Temp. -40°C to 85°C		±2%		MHz
f_{ILRC}	Internal Low clock generator	VDD = 3.0V @ 20°C	12	16	24	KHz
V_{LVD16}	LVD16 detect voltage	25°C	1.58	1.65	1.72	V
V_{LVD12}	LVD12 detect voltage	25°C	1.20	1.25	1.32	V
ESD	Human body mode (HBM)		2	8		KV
	Machine mode (MM)		200	400		V
LU	Latch Up		±400	±1000		mA
EFT	Electrical Fast Transient (Fcpu = 1mips)	V_{L+}			4500	V
		V_{L-}			4500	
		V_{N+}			4500	
		V_{N-}			4500	
		V_{LN+}			4500	
		V_{LN-}			4500	

* Parameter(s) with star mark are non-verified design reference. Ambient temperature is 25°C.

27.3 GPIO Characteristics

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
V_{IL}	Low-level input voltage		VSS		0.3VSS	V
V_{IH}	High-level input voltage		0.7VDD		VDD	V
I_{LEKG}	I/O port input leakage current	V_{IN} = VDD			2	μA
R_{UP}	Pull-up resister	VDD = 3V	100	200	300	kΩ
I_{OH}	I/O output source current	VDD = 3V, V_O = VDD-0.5V		20		mA
I_{OL}	I/O sink current	VDD = 3V, V_O = VSS+0.5V		80		mA

27.412b-SAR ADC Electrical Characteristics

SYM.	*DESCRIPTION*	*Condition*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
V_{SAR}	Operating Voltage	Power source from AVDD	2.0	-	3.6	V
V_{AI}	Input signal voltage		0		V_{REFH}	V
V_{REFH}	External reference voltage	AVREFH pin input voltage	2		V_{AVDD}	V
V_{IREF}	Internal 3V reference voltage	AVDD must set ≥ 3.1V	2.988	3	3.013	V
	Internal 2.5V reference voltage	AVDD must set ≥ 2.6V	2.488	2.5	2.513	V
	Internal 2V reference voltage	AVDD must set ≥ 2.1V	1.988	2	2.013	V
	AVDD reference voltage			AVDD		
I_{SAR}	ADC current consumption	AVDD=3.0V, Int. VREF=ON		300		uA
		AVDD=3.0V, Int. VREF=OFF		250		μA
I_{INTREF}	Internal Reference Voltage power consumption			50		uA
F_{ADCLK}	ADC clock				16	MHz
F_{ADSAP}	ADC sampling rate		31.25		250	KHz
T_{ADEN}	ADC function enable period		100			uS
DNL	Differential nonlinearity*	F_{ADSAMP} = 31.25kHz		±1		LSB
		F_{ADSAMP} = 62.5kHz		±1		LSB
		F_{ADSAMP} = 250kHz		±1		LSB
INL	Integral Nonlinearity*	F_{ADSAMP} = 31.25kHz		±2		LSB
		F_{ADSAMP} = 62.5kHz		±2		LSB
		F_{ADSAMP} = 250kHz		±2		LSB
NMC	No missing code*	F_{ADSAMP} = 31.25kHz		11		Bit
		F_{ADSAMP} = 62.5kHz		11		Bit
		F_{ADSAMP} = 250kHz		10		Bit
IPDN	Power down current	Stop mode	-	0.1	-	μA
V_{OFFSET}	Input offset voltage	Non-trimmed	-10		10	mV

* Parameters with star mark: VDD = 3V, VREFH = 2.0V, 25°C.

27.5OP-Amp Electrical Characteristic

Operation Amplifier						
SYM.	**PARAMETER**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
V_{oper}	Operation voltage	Power source from AVDD	2.0	-	3.6	V
V_{OS}	Input offset voltage			±3		mV
I_{oper}	Operating Current	Per OP-Amp	-	100	-	uA
V_{IN}	Analog Input voltage range	OP+ / OP-	0.05		AVDD-0.1	V
V_{out}	Analog Output voltage range	OPOUT	0.05		AVDD-0.1	V
I_{OH}/I_{OL}	Output short circuit current	Unit Gain Buffer. Vo = 0V~3.2V, AVDD=3.3V	-	±5	-	mA
R_{BUF}	Buffer Mode Switch Ron	AVDD 3.3V, VOUT=1.65V.	-	350	--	Ω
GB	Gain Bandwidth	RL=300KΩ, CL=50pF		1.4	-	MHz
SR	Slew Rate	10% to 90%	-	0.8	-	V/uS
T_{ON}	Turn on time		-	20	-	uS

27.6 Analog Switch Characteristic

Analog Switch						
SYM.	**PARAMETER**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
V_{oper}	Operation voltage	Power source from AVDD	2.0	-	3.6	V
$R1\sim R2_{on}$	SW1~2 CMOS On-State switch resistance		-	420	-	Ω
$R3\sim R5_{on}$	SW3~SW5 CMOS On-State switch resistance		-	40	-	Ω
$R6\sim R7_{on}$	SW6~7 CMOS On-State switch resistance		-	420	-	Ω

27.7 12b-DAC Electrical Characteristics

Operation Amplifier						
SYM.	**PARAMETER**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
V_{oper}	Operation voltage	Power source from AVDD	2.0	-	3.6	V
V_{DRIN}	DAREF internal input voltage	DAC reference from source internal 4/8*VREFH@VREFH=2	-	1.0	-	V
		DAC reference from source internal 2/8*VREFH@VREFH=2	-	0.5	-	V
		DAC reference from source external source	-	-	AVDD-0.1	V
V_{OFFSET}	Offset voltage		-	3	-	mV
I_{DAC}	Operating Current	Supply current @ no loading	-	300	-	uA
INL	Relative Accuracy		-	±8	-	LSB
DNL	Differential Nonlinearity		-	4	-	LSB
V_{DAO}	Output voltage range		0.03*		VREFH-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

* Parameter(s) with star mark are non-verified design reference.

27.8 Comparator Characteristic

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
V_{ILBT}	Internal Low-Battery detect voltage	Condition: LBTSEL [3:0] = 0000, VLBT=2.2V	2.15	2.2	2.25	V
		Condition: LBTSEL [3:0] = 0100, VLBT=2.6V	2.55	2.6	2.65	
		Condition: LBTSEL [3:0] = 0111, VLBT=2.9V	2.85	2.9	2.95	
V_{ELBT}	External Low-Battery detect voltage	Condition: LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.75	1.8	1.85	
V_{HY}	Comparator Hysteresis Window			50	-	mV
I_{COMP}	Current consumption	AVDD = 3V		50	-	uA

27.9 LCD Characteristic

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
I_{CLCD}	C-Type LCD Operation Current	VDD:2~3.6V, No panel, Low power Mode (BGM=0,DUTY[1:0]=10)		10		uA
		VDD:2~3.6V, No panel, Normal Mode (BGM=1)		100		
V_{LCD}	C-Type VLCD output Voltage	VDD: 2.0 ~ 3.6V, VLCD set 3.0V (VCP [3:0]=0110)	2.80	3.0	3.20	V

27.10 Flash Memory Characteristics

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
V_{dd}	Supply voltage		2.0		3.6	V
T_{en}	Endurance time	25°C		*100K		cycle
I_{wrt}	Write current	25°C		3	4	mA
T_{wrt}	Write time	Write 1 page=64 bytes, 25°C		5	8	ms

* Parameters with star mark are non-verified design reference.

28 Instruction Set

This chapter categorizes the SN8F5800 microcontroller's comprehensive assembly instructions. It includes five categories—arithmetic operation, logic operation, data transfer operation, Boolean manipulation, and program branch—which are fully compatible with standard 8051.

28.1 Symbol description

Symbol	Description
Rn	Working register R0 - R7
direct	One of 128 internal RAM locations or any Special Function Register
@Ri	Indirect internal or external RAM location addressed by register R0 or R1
#data	8-bit constant (immediate operand)
#data16	16-bit constant (immediate operand)
bit	One of 128 software flags located in internal RAM, or any flag of bit-addressable Special Function Registers
addr16	Destination address for LCALL or LJMP, can be anywhere within the 64-Kbyte page of program memory address space
addr11	Destination address for ACALL or AJMP, within the same 2-Kbyte page of program memory as the first byte of the following instruction
rel	SJMP and all conditional jumps include an 8-bit offset byte. Its range is +127/-128 bytes relative to the first byte of the following instruction
A	Accumulator

28.2 Arithmetic operations

Mnemonic	Description
ADD A, Rn	Add register to accumulator
ADD A, direct	Add directly addressed data to accumulator
ADD A, @Ri	Add indirectly addressed data to accumulator
ADD A, #data	Add immediate data to accumulator
ADDC A, Rn	Add register to accumulator with carry
ADDC A, direct	Add directly addressed data to accumulator with carry
ADDC A, @Ri	Add indirectly addressed data to accumulator with carry
ADDC A, #data	Add immediate data to accumulator with carry
SUBB A, Rn	Subtract register from accumulator with borrow
SUBB A, direct	Subtract directly addressed data from accumulator with borrow
SUBB A, @Ri	Subtract indirectly addressed data from accumulator with borrow
SUBB A, #data	Subtract immediate data from accumulator with borrow
INC A	Increment accumulator
INC Rn	Increment register
INC direct	Increment directly addressed location
INC @Ri	Increment indirectly addressed location
INC DPTR	Increment data pointer
DEC A	Decrement accumulator
DEC Rn	Decrement register
DEC direct	Decrement directly addressed location
DEC @Ri	Decrement indirectly addressed location
MUL AB	Multiply A and B
DIV	Divide A by B
DA A	Decimally adjust accumulator

28.3 Logic operations

Mnemonic	Description
ANL A, Rn	AND register to accumulator
ANL A, direct	AND directly addressed data to accumulator
ANL A, @Ri	AND indirectly addressed data to accumulator
ANL A, #data	AND immediate data to accumulator
ANL direct, A	AND accumulator to directly addressed location
ANL direct, #data	AND immediate data to directly addressed location
ORL A, Rn	OR register to accumulator

ORL A, direct	OR directly addressed data to accumulator
ORL A, @Ri	OR indirectly addressed data to accumulator
ORL A, #data	OR immediate data to accumulator
ORL direct, A	OR accumulator to directly addressed location
ORL direct, #data	OR immediate data to directly addressed location
XRL A, Rn	Exclusive OR (XOR) register to accumulator
XRL A, direct	XOR directly addressed data to accumulator
XRL A, @Ri	XOR indirectly addressed data to accumulator
XRL A, #data	XOR immediate data to accumulator
XRL direct, A	XOR accumulator to directly addressed location
XRL direct, #data	XOR immediate data to directly addressed location
CLR A	Clear accumulator
CPL A	Complement accumulator
RL A	Rotate accumulator left
RLC A	Rotate accumulator left through carry
RR A	Rotate accumulator right
RRC A	Rotate accumulator right through carry
SWAP A	Swap nibbles within the accumulator

28.4 Data transfer operations

Mnemonic	Description
MOV A, Rn	Move register to accumulator
MOV A, direct	Move directly addressed data to accumulator
MOV A, @Ri	Move indirectly addressed data to accumulator
MOV A, #data	Move immediate data to accumulator
MOV Rn, A	Move accumulator to register
MOV Rn, direct	Move directly addressed data to register
MOV Rn, #data	Move immediate data to register
MOV direct, A	Move accumulator to direct
MOV direct, Rn	Move register to direct
MOV direct1, direct2	Move directly addressed data to directly addressed location
MOV direct, @Ri	Move indirectly addressed data to directly addressed location
MOV direct, #data	Move immediate data to directly addressed location
MOV @Ri, A	Move accumulator to indirectly addressed location
MOV @Ri, direct	Move directly addressed data to indirectly addressed location
MOV @Ri, #data	Move immediate data to in directly addressed location

MOV DPTR, #data16	Load data pointer with a 16-bit immediate
MOVC A, @A+DPTR	Load accumulator with a code byte relative to DPTR
MOVC A, @A+PC	Load accumulator with a code byte relative to PC
MOVX A, @Ri	Move external RAM (8-bit address) to accumulator
MOVX A, @DPTR	Move external RAM (16-bit address) to accumulator
MOVX @Ri, A	Move accumulator to external RAM (8-bit address)
MOVX @DPTR, A	Move accumulator to external RAM (16-bit address)
PUSH direct	Push directly addressed data onto stack
POP direct	Pop directly addressed location from stack
XCH A, Rn	Exchange register with accumulator
XCH A, direct	Exchange directly addressed location with accumulator
XCH A, @Ri	Exchange indirect RAM with accumulator
XCHD A, @Ri	Exchange low-order nibbles of indirect and accumulator

28.5 Boolean manipulation

Mnemonic	Description
CLR C	Clear carry flag
CLR bit	Clear directly addressed bit
SETB C	Set carry flag
SETB bit	Set directly addressed bit
CPL C	Complement carry flag
CPL bit	Complement directly addressed bit
ANL C, bit	AND directly addressed bit to carry flag
ANL C, /bit	AND complement of directly addressed bit to carry
ORL C, bit	OR directly addressed bit to carry flag
ORL C, /bit	OR complement of directly addressed bit to carry
MOV C, bit	Move directly addressed bit to carry flag
MOV bit, C	Move carry flag to directly addressed bit

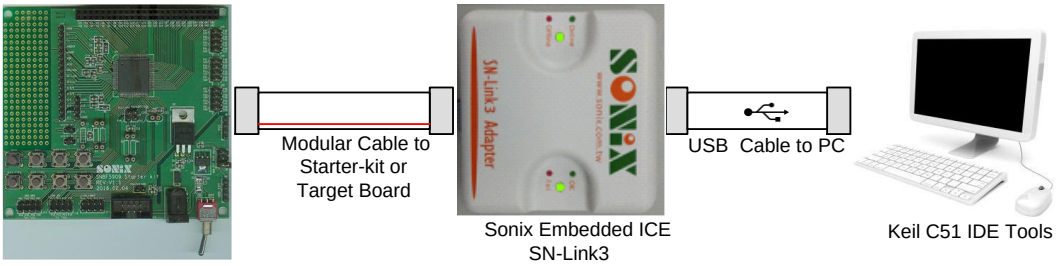
28.6 Program branches

Mnemonic	Description
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump

SJMP rel	Short jump (relative address)
JMP @A+DPTR	Jump indirect relative to the DPTR
JZ rel	Jump if accumulator is zero
JNZ rel	Jump if accumulator is not zero
JC rel	Jump if carry flag is set
JNC rel	Jump if carry flag is not set
JB bit, rel	Jump if directly addressed bit is set
JNB bit, rel	Jump if directly addressed bit is not set
JBC bit, rel	Jump if directly addressed bit is set and clear bit
CJNE A, direct, rel	Compare directly addressed data to accumulator and jump if not equal
CJNE A, #data, rel	Compare immediate data to accumulator and jump if not equal
CJNE Rn, #data, rel	Compare immediate data to register and jump if not equal
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal
DJNZ Rn, rel	Decrement register and jump if not zero
DJNZ direct, rel	Decrement directly addressed location and jump if not zero
NOP	No operation for one cycle

29 Development Environment

SONIX provides an Embedded ICE emulator system to offer SN8F5840 firmware development. The platform is an in-circuit debugger and controlled by Keil C51 IDE software on Microsoft Windows platform. The platform includes SN-Link3, SN8F5840 Starter-kit and Keil C51 IDE 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 SN8F5840 to offer a real development environment.



29.1 Minimum Requirement

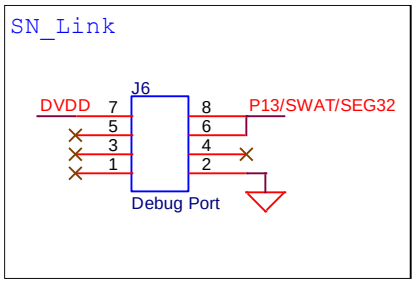
The following items are essential to build up an appropriate development environment. The compatibility is verified on listed versions, and is expected to execute perfectly on later version. SN-Link related information is available to download on SONiX website (www.sonix.com.tw);

- **SN-Link3 Adapter** with updated firmware version 1.04
- **SN-Link Driver for Keil C51** version 2.00.35
- **Keil C51** version 9.50a and 9.54a or greater.

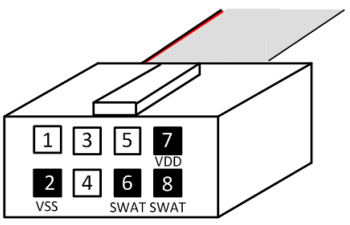
29.2 Debug Interface Hardware

The circuit below demonstrates the appropriate method to connect microcontroller's SWAT pin and SN-Link3 Adapter.

Before starting debug, microcontroller's power (VDD) must be switched off. Connect the SWAT to both 6th and 8th pins of SN-Link, and respectively link VDD and VSS to 7th pin and 2nd pin. A handshake procedure would be automatically started by turn on the microcontroller, and SN-Link's green LED (Run) indicates the success of connection (refer *SN8F5000 Debug Tool Manual* for further detail).



example circuit



SN-Link header

29.3 Development Tool

SN-Link3 Adapter

MP5 Writer

30 SN8F5840 Starter-Kit

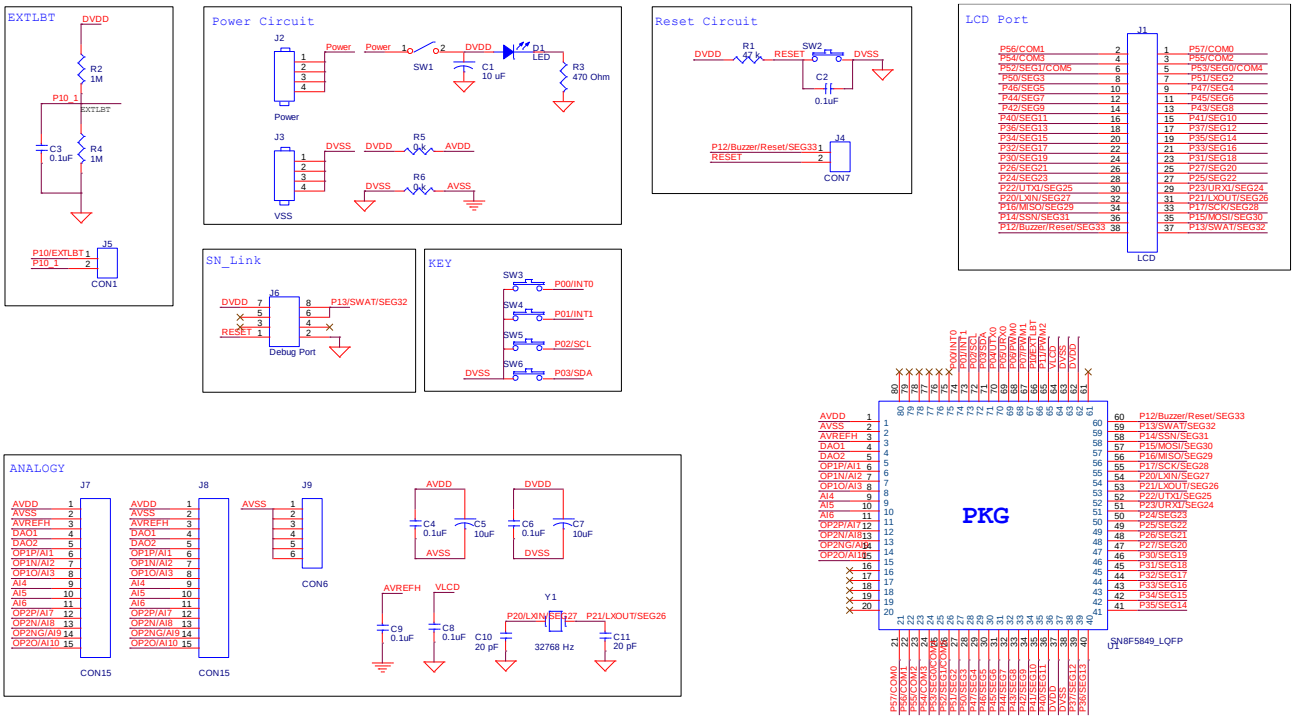
SN8F5000 Starter-Kit provides easy-development platform. It includes SN8F5000 family real chip and I/O connectors to input signal or drive device of user's application. It is a simple platform to develop application as target board not ready. The Starter-Kit can be replaced by target board, because SN8F5000 family integrates embedded ICE in-circuit debugger circuitry.

30.1Configurations of Circuit

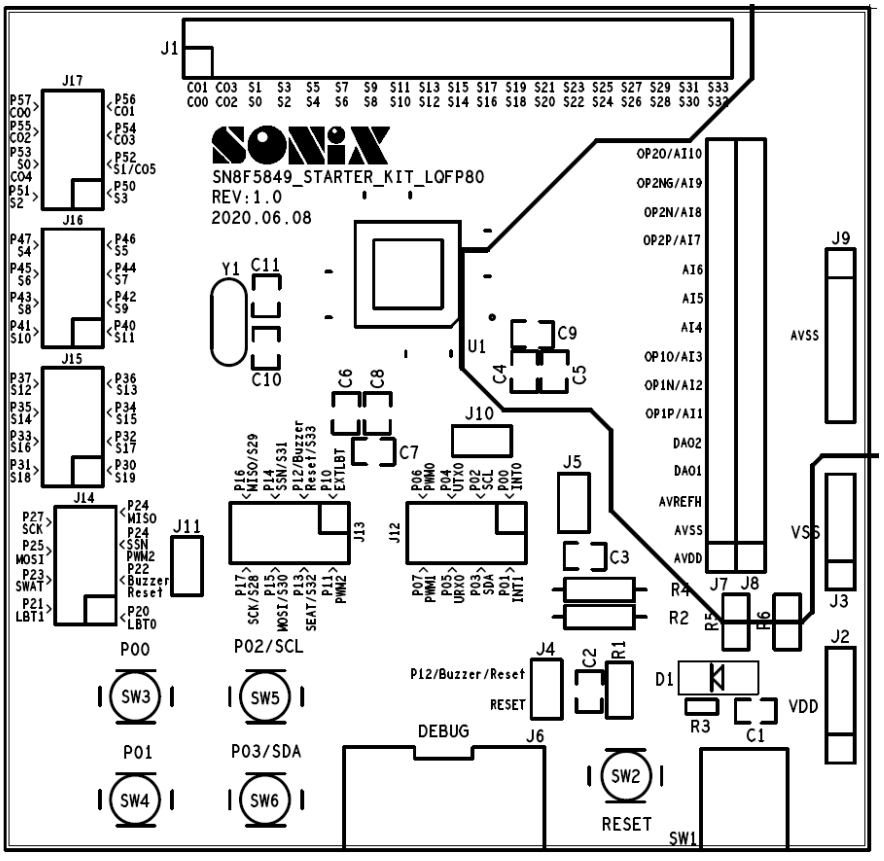
These configurations must be setup completely before starting Starter-Kit developing.

1. Confirm to the circuit board whether elements are complete.
2. The power source of Starter-Kit circuit is 2.0~3.6V.
3. If the power source is chosen from external power, then external power source connects to EXT pin.
4. The "XIN" pin and the "XOUT" pin need to connect crystal/resonator oscillator components when system clock is setting crystal or RTC mode.
5. The "XIN" pin needs to connect external clock source when system clock is setting external clock input mode.
6. The Debug Port can connect SN-LINK Adapter for emulation or download code.
7. The MCU LED will light up and SN8F5000 family chip will be connected to power when power (VDD) is switched on.

30.2Schematic



30.3 Floor Plan of PCB layout



30.4Component Description

Number	Description
C8	VLCD function capacitor.
C9	AVREFH function capacitor.
J5	LBT function connector.
R2,R4,C3	LBT capacitor and resisters.
C4,C5	AVDD power regulators capacitance.
C6,C7	DVDD power regulators capacitance.
J2,J3	power source.
J7,J8	ADC function pin.
J9	AVSS pin.
J10-J11	DVSS pin.
D1,R3,C1	MCU LED and resister and capacitor.
J12-J17	I/O connector.
SW3-SW6	I/O button.
R5,R6	0 ohm resisters.
J1	LCD function connector.
SW2,R1,C2	External reset trigger source
J4	External reset trigger function connector.
Y1,C10,C11	External crystal/resonator oscillator components.
SW1	Target power (VDD) switch.
J6	Debug Port
U1	SN8F5849 real chip (Sonix standard option).

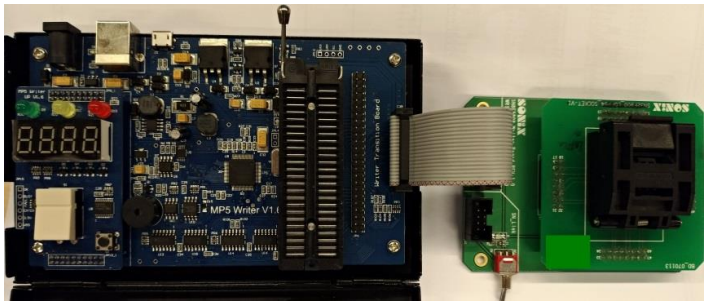
31 ROM Programming Pin

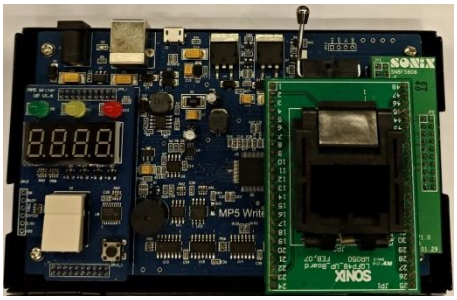
SN8F5840 Series Flash ROM erase/program/verify support SN-Link and MP5 Writer

- SN-Link: Debug interface and on board programming.
- MP5 Writer: For SN8F5840 series version mass programming.

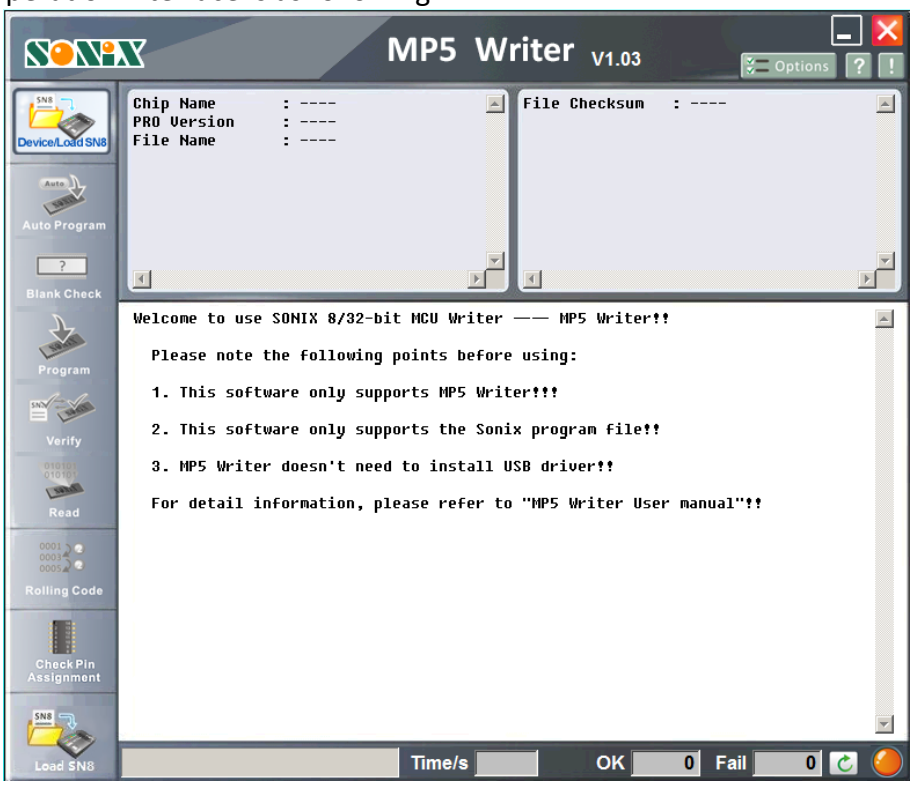
31.1 MP5 Hardware Connecting

Different package type with MCU programming connecting is as following, LQFP80 and LQFP64/48 Illustration.





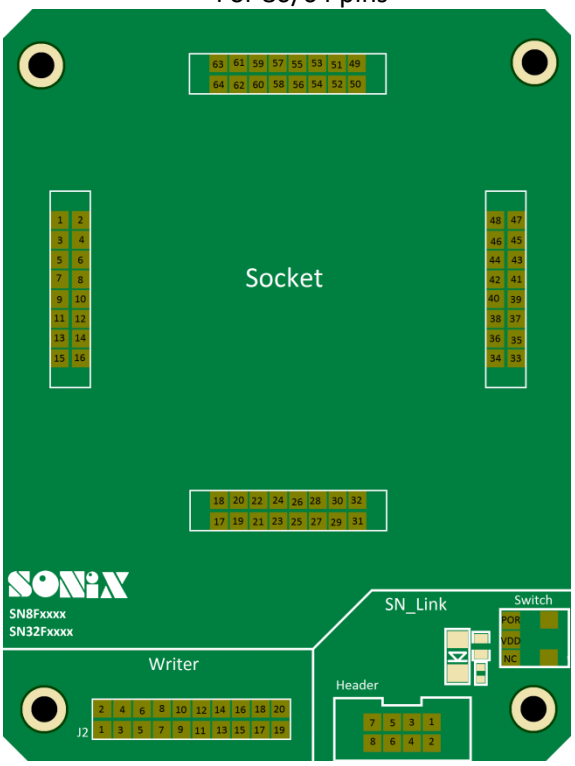
MP5 Software operation interface is as following.



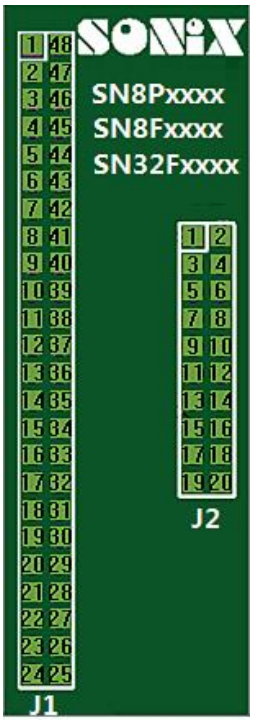
31.2 MP5 Writer Transition Board

MP5 Writer Transition Board:

For 80/64 pins



For 48 pins



31.3 MP5 Writer Programming Pin Mapping

There are two modes of MP5 writer programming: normal mode and high speed mode. Normal mode requires four pins to program the code. The high speed mode requires eight pins to program the code for fast programming.

Normal mode can meet most programming needs. However, if you want to shorten the programming time, you can use the high speed mode. High speed mode requires a more stable connection environment. Please confirm whether the environment can meet the requirements before use.

31.4 MP5 Writer Programming Pin Mapping

Writer Connector		MCU Pin Number	SN8F5849F	SN8F5848F	SN8F5847F	SN8F5844T
J2 Pin Number	J2 Pin Name		MCU Pin Number	MCU Pin Number	MCU Pin Number	MCU Pin Number
1	VDD	AVDD/DVDD	1/62,37	2/53,33	2/15	12
2	GND	AVSS/DVSS	2/63,38	3/54	3/16	4
7,9	SWAT	P1.3	59	51	48	10
20	PDB	P0.2	72	63	12	19

31.5 SN-Link ISP Programming

SN-Link ISP programming hardware and software are as following.

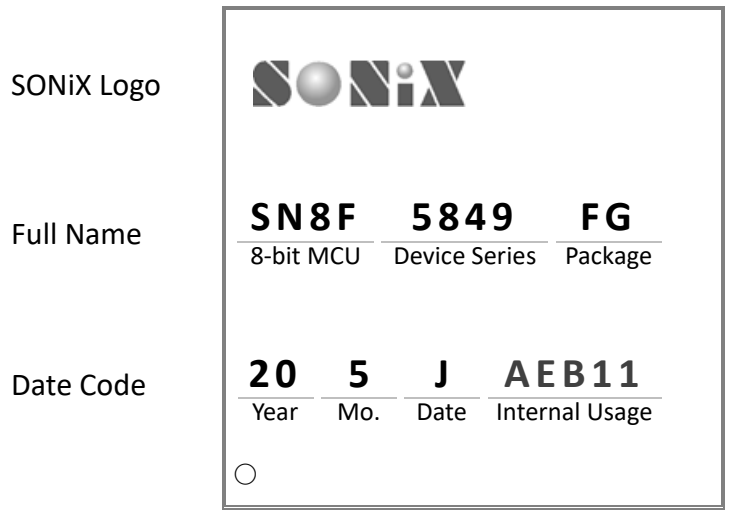
31.6 SN-Link ISP Programming Pin Mapping

SN-Link Connector		MCU Pin Number	SN8F5849F	SN8F5848F	SN8F5847F	SN8F5844T
Pin Number	Pin Name		Pin Number	Pin Number	Pin Number	Pin Number
7	VDD	AVDD/DVDD	1/62,37	2/53,33	2/15	12
2	GND	AVSS/DVSS	2/63,38	3/54	3/16	4
6,8	SWAT	P1.3	59	51	48	10

32 Overview of Packaging Information

This documentation introduces SN8F5840 family microcontrollers' mechanical data, such as height, width and pitch information.

In general, every SONiX microcontroller displays three columns: logo, device's full name and date code. The full name's package field maps to its package type; whereas the date code records the date of manufacture.



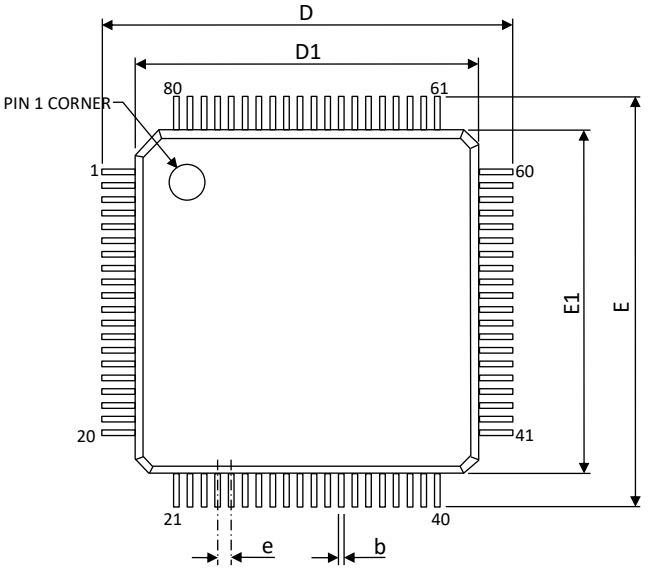
Date Code

Year	18: 2018 19: 2019 20: 2020 et cetera
Month	1: January 2: February 3: March A: October B: November C: December et cetera
Date	1: 01 2: 02 3: 03 A: 10 B: 11 et cetera

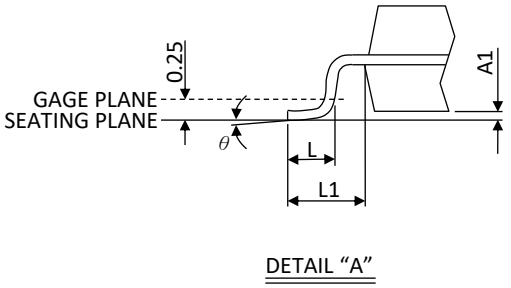
32.1 Device Nomenclature

Full Name	Packing Type
S8F5849W	Wafer
SN8F5849H	Dice
SN8F5849FG	LQFP, 80 pins, Green package
SN8F5848FG	LQFP, 64 pins, Green package
SN8F5847FG	LQFP, 48 pins, Green package
SN8F5844TG	TSSOP, 28 pins, Green package

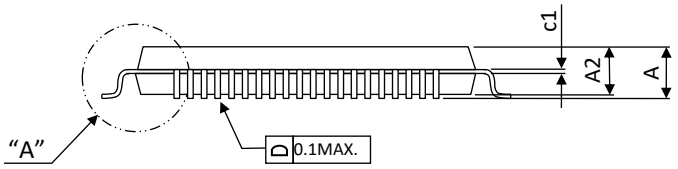
32.2LQFP 80 PIN



TOP VIEW



DETAIL "A"



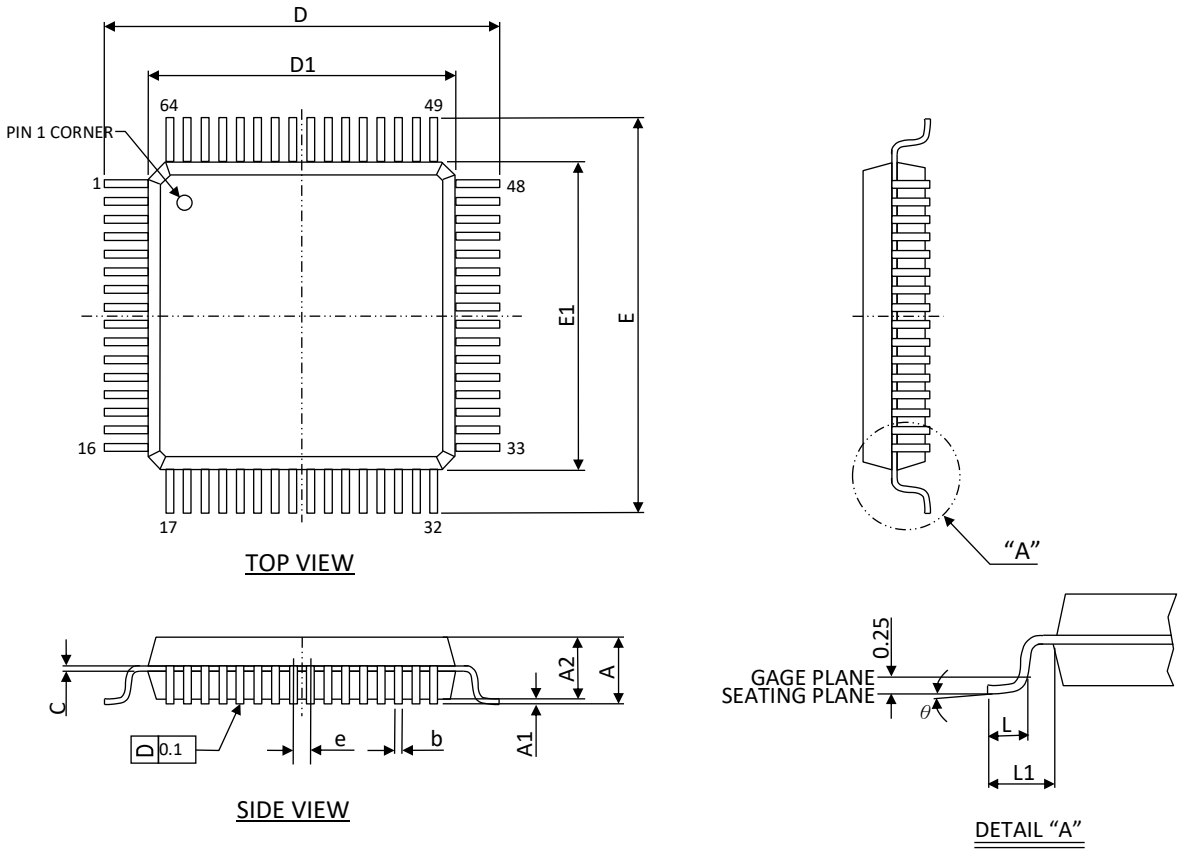
SIDE VIEW

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.60	--	--	0.063
A1	0.05	--	0.2	0.002	--	0.008
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.13	0.18	0.23	0.005	0.007	0.009
c1	0.09	--	0.18	0.004	--	0.007
D	12 BSC			0.472 BSC		
D1	10 BSC			0.394 BSC		
e	0.4 BSC			0.016 BSC		
E	12 BSC			0.472 BSC		
E1	10 BSC			0.394 BSC		
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	1.0 REF			0.039 REF		
θ	0˚	3.5˚	7˚	0˚	3.5˚	7˚

Notes:

1. CONTROLLING DIMENSION：MILLIMETER (mm)
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

32.3LQFP 64 PIN

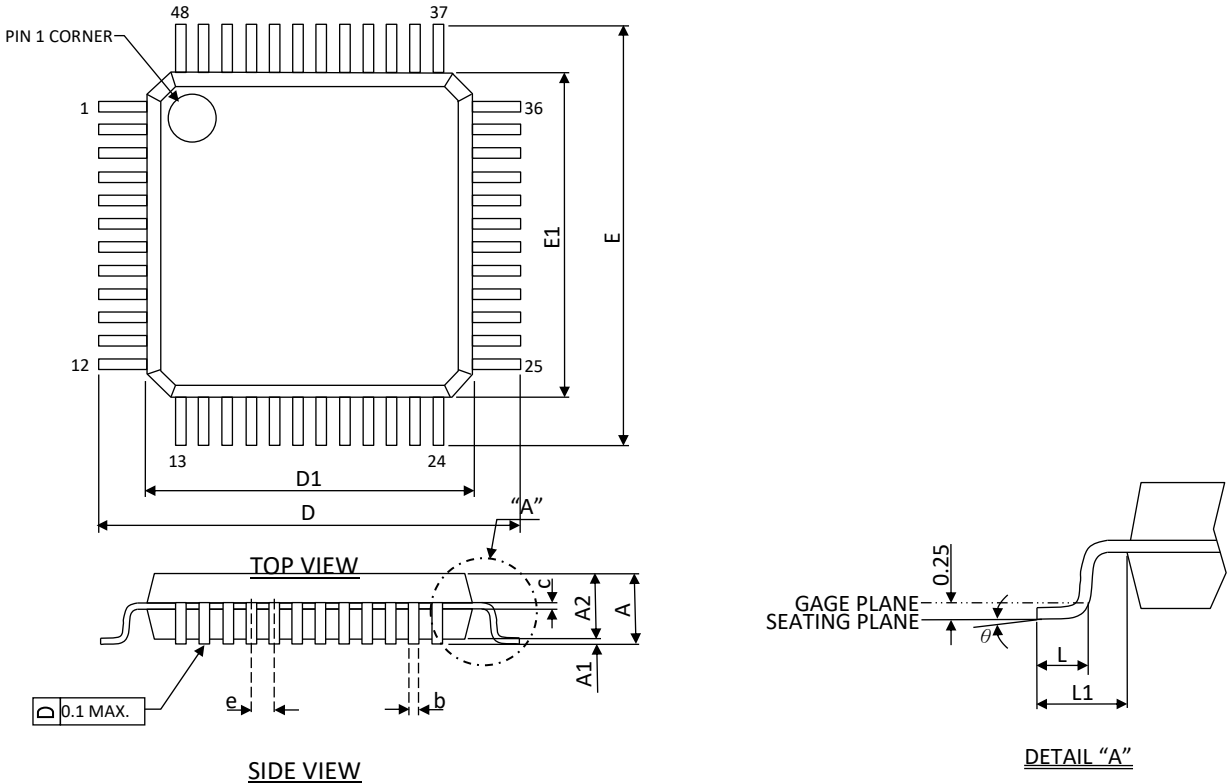


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.60	--	--	0.063
A1	0.05	--	0.25	0.002	--	0.01
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.13	0.19	0.25	0.005	0.007	0.010
c	0.09	--	0.20	0.004	--	0.008
D	9.00 BSC			0.354 BSC		
D1	7.00 BSC			0.276 BSC		
e	0.40 BSC			0.016 BSC		
E	9.00 BSC			0.354 BSC		
E1	7.00 BSC			0.276 BSC		
L	0.4	0.60	0.8	0.016	0.024	0.032
L1	1.00 REF			0.039 REF		
θ	0°	3.5°	7°	0°	3.5°	7°

Notes:

1. CONTROLLING DIMENSION：MILLIMETER (mm
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

32.4LQFP 48 PIN

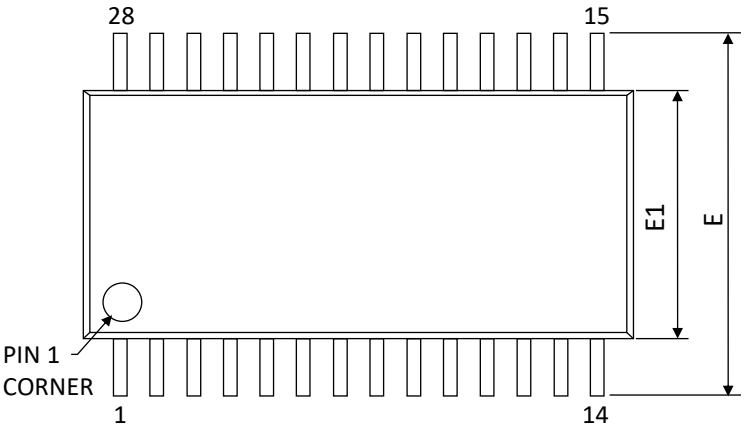


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.60	--	--	0.063
A1	0.05	--	0.15	0.002	--	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
c	0.09	--	0.20	0.004	--	0.008
D	9.00 BSC			0.354 BSC		
D1	7.00 BSC			0.276 BSC		
E	9.00 BSC			0.354 BSC		
E1	7.00 BSC			0.276 BSC		
e	0.50 BSC			0.020 BSC		
L	0.40	0.60	0.80	0.016	0.024	0.031
L1	1.00 REF			0.039 REF		
θ	0°	3.5°	7°	0°	3.5°	7°

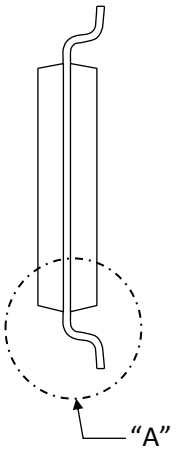
Notes：

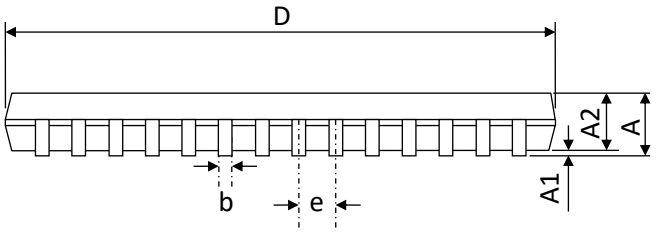
1. CONTROLLING DIMENSION：MILLIMETER (mm)
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

32.5TSSOP28 PIN

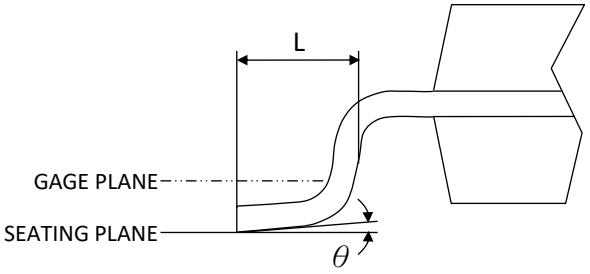


TOP VIEW





SIDE VIEW



DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.20	--	--	0.047
A1	0.00	--	0.15	0.000	--	0.006
A2	0.80	1.00	1.05	0.031	0.039	0.041
b	0.19	--	0.30	0.007	--	0.012
D	9.60	9.70	9.80	0.378	0.382	0.386
E		6.40 BSC.			0.252 BSC.	
E1	4.30	4.40	4.50	0.169	0.173	0.177
e		0.65 BSC.			0.026 BSC.	
L	0.45	0.60	0.75	0.018	0.024	0.030
θ	0°	--	8°	0°	--	8°

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE : MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

SN8F5840 Series Datasheet

8051-based Microcontroller

Corporate Headquarters 10F-1, No. 36, Taiyuan St. Chupei City, Hsinchu, Taiwan TEL: +886-3-5600888
FAX: +886-3-5600889

Taipei Sales Office
15F-2, No. 171, Songde Rd. Taipei City, Taiwan
TEL: +886-2-27591980
FAX: +886-2-27598180
mkt@sonix.com.tw
sales@sonix.com.tw

Hong Kong Sales Office
Unit 2603, No. 11, Wo Shing St. Fo Tan, Hong Kong
TEL: +852-2723-8086
FAX: +852-2723-9179
hk@sonix.com.tw

Shenzhen Contact Office
High Tech Industrial Park, Shenzhen, China
TEL: +86-755-2671-9666
FAX: +86-755-2671-9786
mkt@sonix.com.tw
sales@sonix.com.tw

USA Office
TEL: +1-714-3309877
TEL: +1-949-4686539
tlightbody@earthlink.net

Japan Office
2F, 4 Chome-8-27 Kudanminami Chiyoda-ku, Tokyo, Japan
TEL: +81-3-6272-6070
FAX: +81-3-6272-6165
jpsales@sonix.com.tw

FAE Support via email
8-bit Microcontroller Products:
sa1fae@sonix.com.tw
All Products: fae@sonix.com.tw