



SONiX Technology Co., Ltd.

SN8F5900 Series

Datasheet

(Rev. 1.7)

8051-based Microcontroller

SN8F5907

SN8F5908

SN8F5909

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:

- 64 KB non-volatile flash memory (IROM) with in-system program support
- 256 bytes internal RAM (IRAM)
- 2 KB external RAM (XRAM)

I/O Pin: 20-Pins pure I/O and 24-Pins I/O share with LCD function.

Interrupt: 10 interrupt sources with priority

- levels control and unique interrupt vectors
- 8 internal interrupts: T0, TC0, TC1, TC2, ADC, UART, I2C, SPI
- 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, PWM, Buzzer output.(TC0~TC2)
- One Buzzer Pin (P15): 1KHz / 2KHz / 4KHz / 8KHz

SPI, UART, I2C interface with SMBus Support

One Comparator for Low Battery Detect 2.2V~3.6V (0.2V/step)

Power Supply: 2.0~6.5V (AVDD/DVDD)

Charge Pump Regulator:

- Input voltage: 2.0V ~ 6.5V
- Regulator AVDDR output for Analog Circuit (ADC, PGIA, OP..)
- AVDDR selectable: 2.7V, 3.0V, 3.3V, 3.6V
- ACM 1V for ADC common voltage.

PGIA: Programmable Gain Instrument Amplifier, High Input Impedance

- Gain: 1x, 4x, 8x, 16x, 32x, 64x, 128x

24-Bit Delta Sigma ADC:

- Single or Differential channel configure.
- 9 external Input channels: AI1~AI9
- 4 internal Input channels: 1/8*VDD detect, Temperature sensor, ACM, and Gnd.
- Internal or External reference voltages
- Conversion Rate: 2Hz~7.8KHz

OPA: 1 set Operational Amplifier

LCD Driver: C-Type LCD driver up to 252 dot

- 4 com or 6 com with 1/3 bias
- 4x44 or 6x42 dots, LCD pin share with I/O
- Adjustable VLCD output 2.6V ~ 4.5V

On-Chip Debug Support:

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

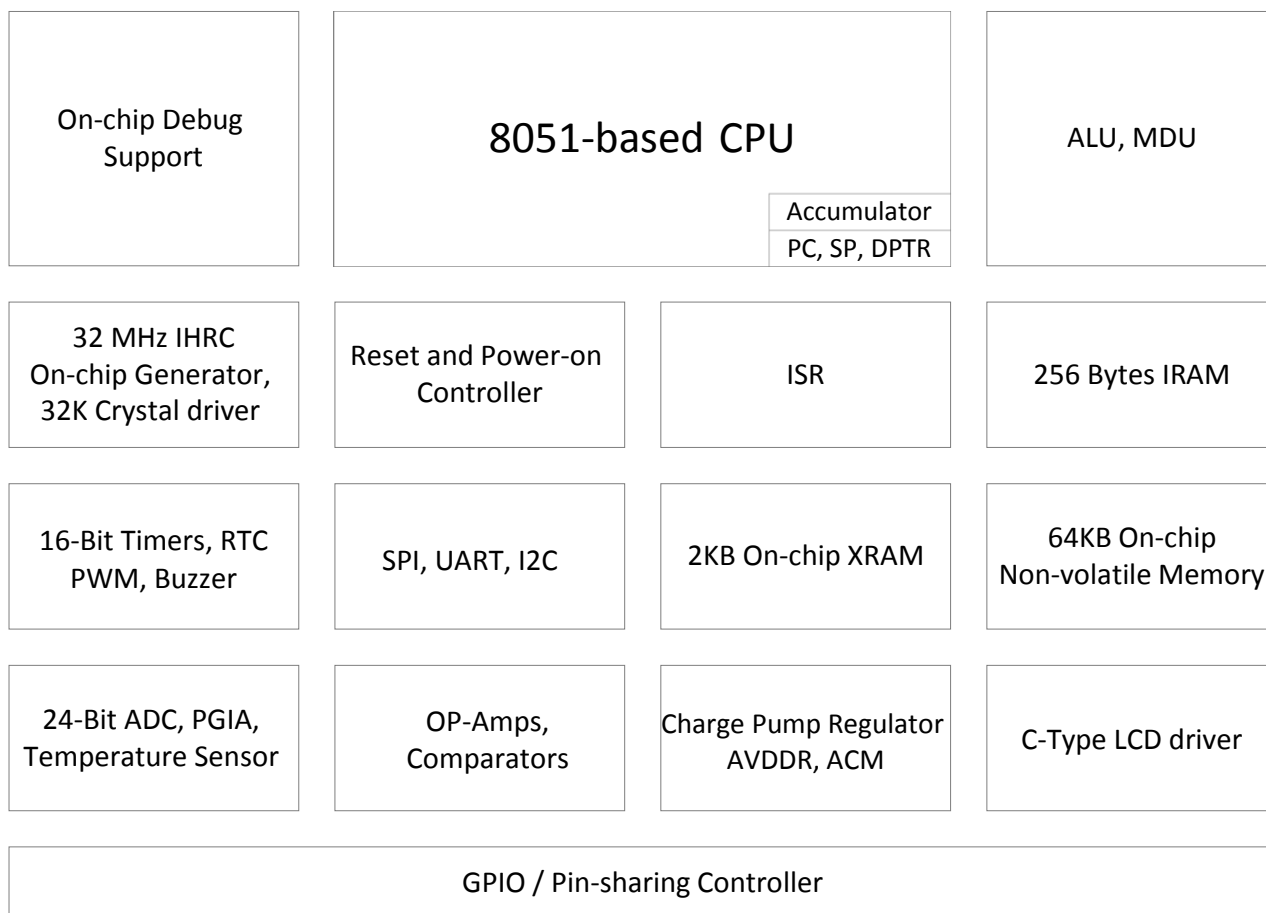
1.2 Applications

- Blood Pressure Meter
- Glucose Meter
- Other measurement application.
- Thermometer
- Weight scale

1.3 Features Selection Table

	I/O	LCD	RTC	16-Bit Timer / PWM	I2C	SPI	UART	ADC	OPA	CMP	Int. INT	Ext. INT	Package Types
SN8F5907	34	4x26, 6x24	V	3	V	V	V	5-Ch	1	2	8	2	LQFP64
SN8F5908	44	4x36, 6x34	V	3	V	V	V	9-Ch	1	2	8	2	LQFP80
SN8F5909	44	4x44, 6x42	V	3	V	V	V	9-Ch	1	2	8	2	LQFP100

1.4 Block Diagram



2 Table of Contents

1	Device Overview.....	2
2	Table of Contents	5
3	Revision History.....	6
4	Pin Assignments	8
5	CPU	15
6	Special Function Registers.....	20
7	Reset and Power-on Controller	27
8	System Clock and Power Management	30
9	Interrupt.....	34
10	MDU	40
11	GPIO	44
12	External Interrupt.....	48
13	TCx 16-BIT Timer/Counter	51
14	Timer 0	65
15	Buzzer Function.....	72
16	LCD Driver	73
17	ADC.....	80
18	Charge Pump Regulator	96
19	Comparator	99
20	OPA.....	102
21	UART.....	104
22	SPI.....	111
23	I2C	117
24	In-System Program.....	129
25	APPLICATION CIRCUIT	135
26	Electrical Characteristics	138
27	Instruction Set.....	144
28	Debug Interface.....	149
29	Analog Setting and Application.....	151
30	Overview of Packaging Information.....	152

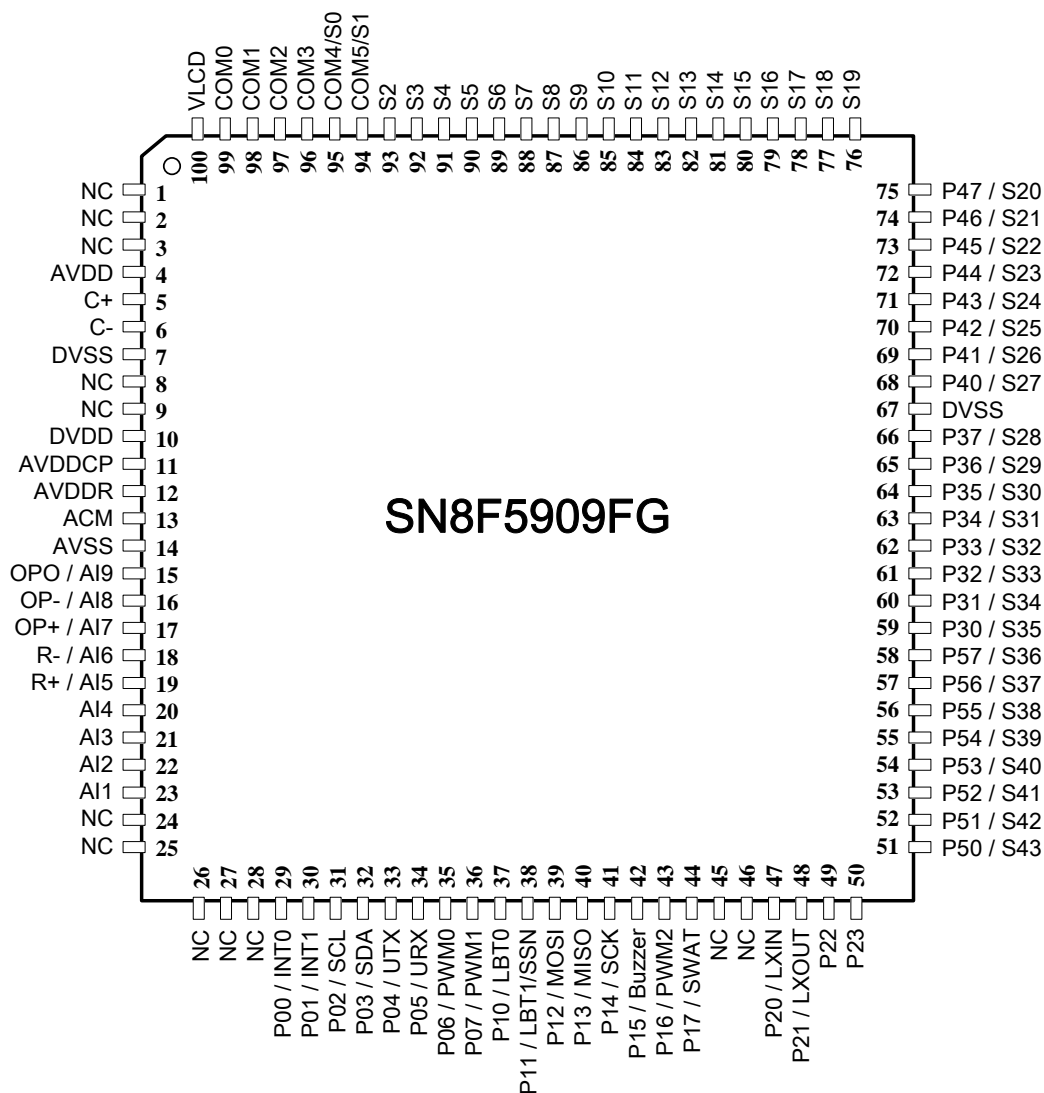
3 Revision History

Revision	Date	Description
1.0	May 2016	1. Modify PGA Register descriptions(P83)
1.1	May 2016	1. Modify Temperature Sensor Sensitivity descriptions -1.8mV/°C (P90) 2. Modify LQFP64 Pin Name (P10)
1.2	June 2016	1. Modify ADCM2 Register DRDY Bit descriptions. (P87) 2. Modify Example code SCL clock rate is from TC2 Timer overflow rate. (P120) 3. Remove Table 19-2 Recommended Setting TC2 Timer overflow period for Common UART 1M Baud Rates. (P107) 4. Modify SN8P5907 LQFP64 P4 Pin Name. (P10) 5. Modify Internal VDD voltage division setting value. (P101) 6. Modify T0 timer with 0.5 or 60 Sec wake up in stop mode. (P71) 7. Add T0 Timer RTC function use STOP&STOP_RTC Macros System into STOP MODE Sample code. (P69) 8. Modify CH19 Comparator Internal VDD voltage division value 2.1~3.5v. (P103)
1.3	Aug 2016	1. Modify P0~P5 Register Initial value. (P46) 2. Modify 24.2 Byte Program descriptions. (P130) 3. Modify 26.9 Flash Memory Characteristics descriptions. (P142) 4. Modify CH19 Comparator Internal VDD voltage division value 2.2~3.6v. (P99) 5. Modify 17.10 Temperature Sensor (TS) descriptions. (P90) 6. Add Charge Pump note descriptions. (P97) 7. Add Table 19-2 Recommended Setting TC2 Timer overflow period for Common UART Baud Rates descriptions.(P106) 8. Modify 19 comparator block diagram (P99) 9. Modify 17 ADC block diagram MUXP & MUXN setting value OP+/OPO(P80) 10. Add Debug interface P1.7 notice (P149) 11. Modify 4.4 Pin Descriptions-ACM Connect 0.1uF to AVDDR. (P14) 12. Modify LBTM Register (0xD7) LBTSEL3 descriptions.(P99) 13. Modify Overview of packaging information descriptions.(P152~156) 14. Delete T0 Timer RTC function use STOP&STOP_RTC Macros System into STOP MODE Sample code. (P69~P71) 15. Modify T0 timer with 0.5 or 60 Sec wake up in stop mode. (P71) 16. Modify Analog input signal channel selection table (P82) 17. Modify MUXP[3:0] & MUXN[3:0] descriptions. (P85) 18. Add CH29 Analog Setting and Application descriptions.(P151) 19. Add (*Note: Port P1.7 don't connect any loading@Debug Mode.) (P13) 20. Modify Comparator Characteristic LBTSEL [3:0] descriptions.(P100)(P103) 21. Add 26.10 Recommendations for special use environmental descriptions.(P142) 22. Modify LCDM1 Register (0xAD) descriptions.(P76) 23. Add Charge Pump Regulator Setting table. (P98)
1.4	Oct 2016	1. Modify baud rate control note descriptions.(P105) 2. Modify ADC Conversion Rate Table.(P84)
1.5	Oct 2016	1. Modify Ch9 Interrupt descriptions.(P34) 2. Modify Ch12 External Interrupt descriptions.(P48) 3. Modify 27.5 Boolean manipulation descriptions.(P147)
1.6	Nov 2016	1. Add 8.3 Power Management 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.(P30) 2. Modify T0 with 0.5 or 60 Sec wake up stop mode sample code.(P69) 3. Modify LCD Function keep running in stop mode.(P75&P78)
1.7	Jan 2017	1. Add Revision History descriptions.(P6)

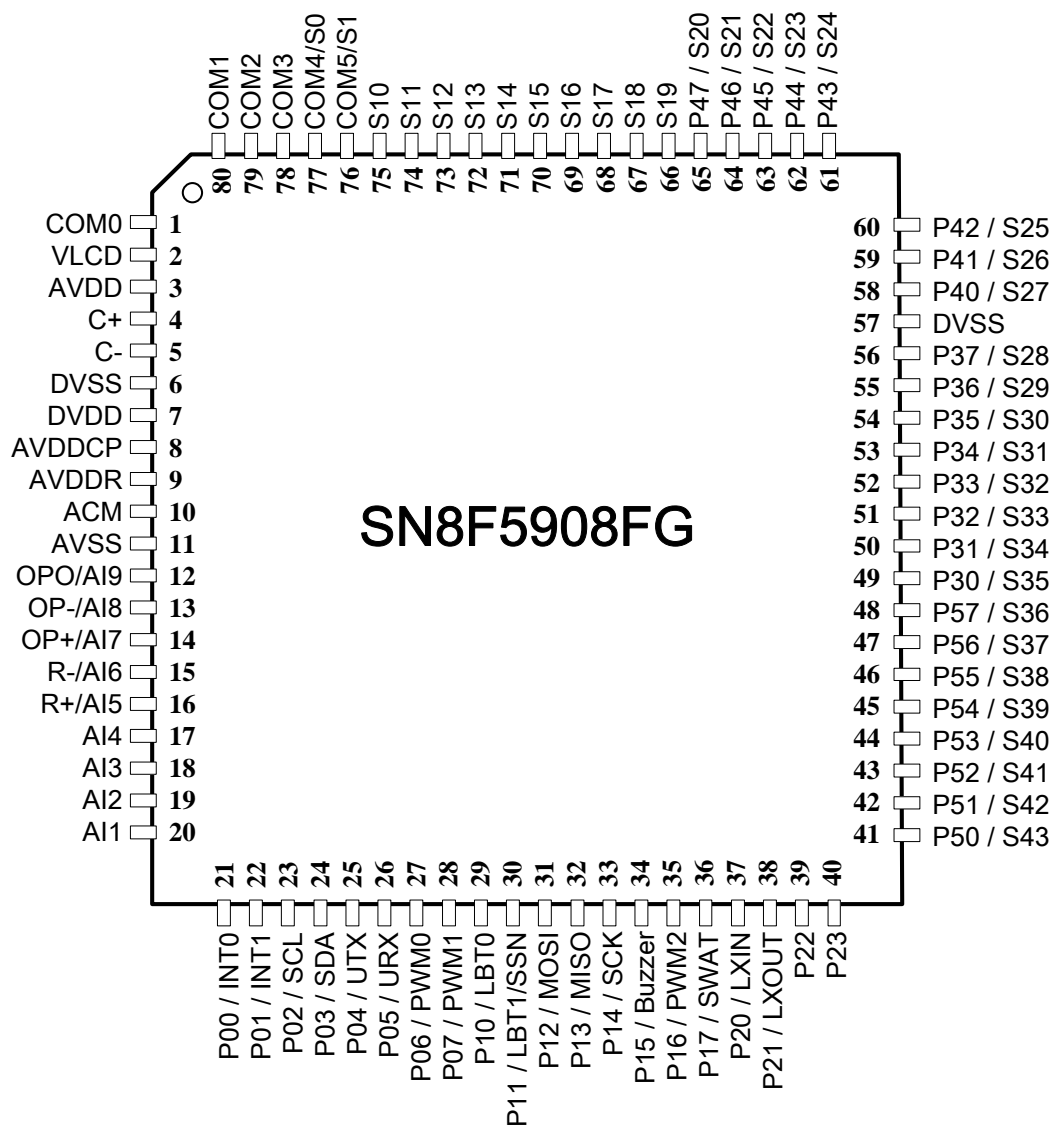
SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights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

4 Pin Assignments

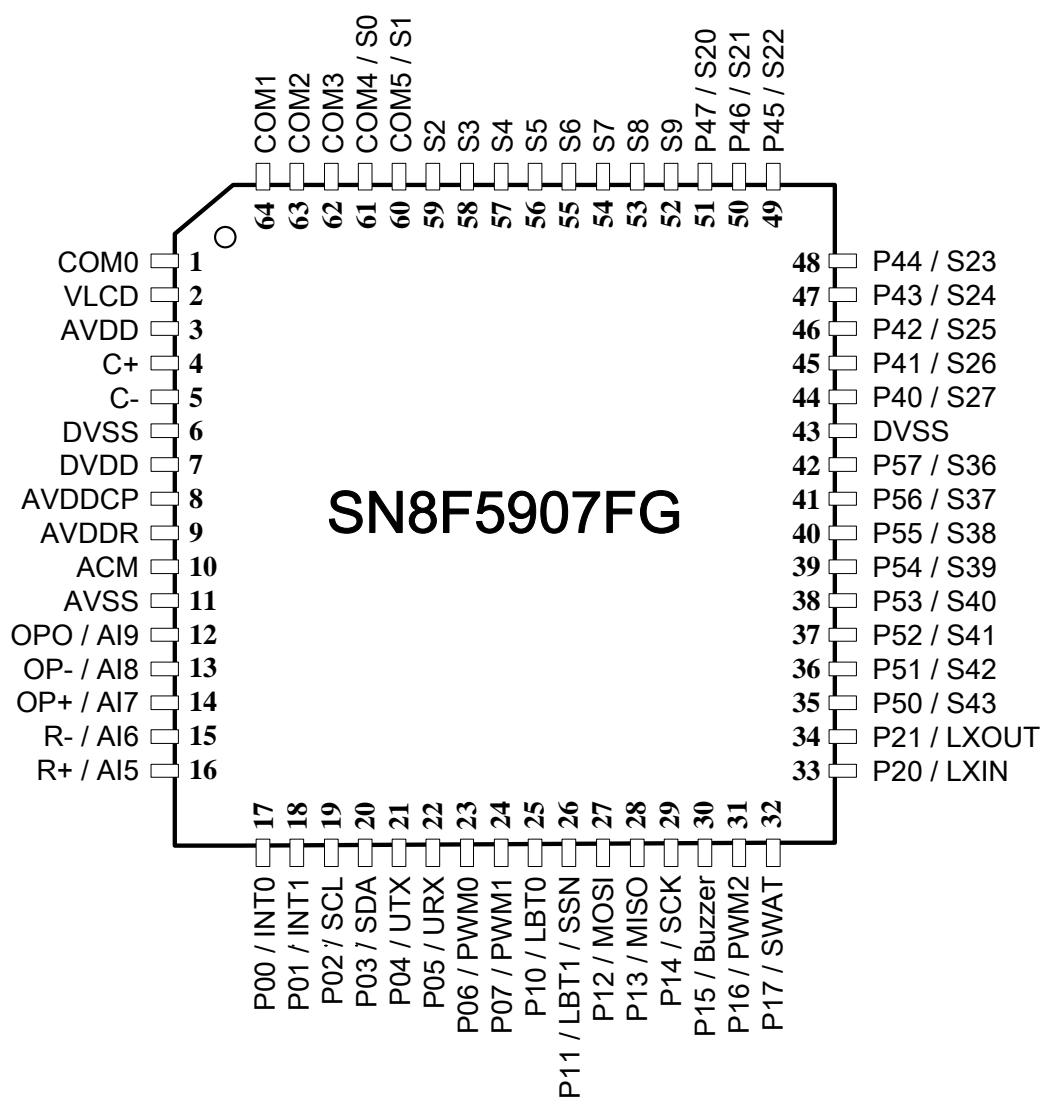
4.1 SN8F5909F (LQFP100)



4.2 SN8F5908F (LQFP80)



4.3 SN8F5907F (LQFP64)



4.4 Pin Descriptions

Power Pins

Pin	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 Output	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
LBT0	Analog Input	Comparator external Input
P1.1	Digital I/O	GPIO
LBT1	Analog Output	Ground control pin.
SSN	Digital Input	SPI: Slave selection pin
P1.2	Digital I/O	GPIO
MOSI	Digital I/O	SPI: transmission pin (master) reception pin (slave)
P1.3	Digital I/O	GPIO
MISO	Digital I/O	SPI: reception pin (master) transmission pin
P1.4	Digital I/O	GPIO
SCK	Digital I/O	SPI: clock output (master) clock input (slave)

P1.5 Buzzer	Digital I/O Digital Output	GPIO Buzzer output pin
P1.6 PWM2	Digital I/O Digital Output	GPIO PWM2 output pin
P1.7 SWAT	Digital I/O Digital I/O	GPIO Debug interface (*Note: Port P1.7 can't connect any loading @Debug Mode.)

Port 2

Pin Name	Type	Description
P2.0 LXIN	Digital I/O Analog Input	GPIO System clock: external clock input
P2.1 LXOUT	Digital I/O Analog Output	GPIO System clock: drive external crystal
P2.2	Digital I/O	GPIO
P2.3	Digital I/O	GPIO

Port 3

Pin Name	Type	Description
P3.0 SEG35	Digital I/O LCD Output	GPIO LCD: SEG35
P3.1 SEG34	Digital I/O LCD Output	GPIO LCD: SEG34
P3.2 SEG33	Digital I/O LCD Output	GPIO LCD: SEG33
P3.3 SEG32	Digital I/O LCD Output	GPIO LCD: SEG32
P3.4 SEG31	Digital I/O LCD Output	GPIO LCD: SEG31
P3.5 SEG30	Digital I/O LCD Output	GPIO LCD: SEG30
P3.6 SEG29	Digital I/O LCD Output	GPIO LCD: SEG29
P3.7 SEG28	Digital I/O LCD Output	GPIO LCD: SEG28

Port 4

Pin Name	Type	Description
P4.0 SEG27	Digital I/O LCD Output	GPIO LCD: SEG27
P4.1 SEG26	Digital I/O LCD Output	GPIO LCD: SEG26
P4.2 SEG25	Digital I/O LCD Output	GPIO LCD: SEG25
P4.3 SEG24	Digital I/O LCD Output	GPIO LCD: SEG24
P4.4 SEG23	Digital I/O LCD Output	GPIO LCD: SEG23
P4.5 SEG22	Digital I/O LCD Output	GPIO LCD: SEG22
P4.6 SEG21	Digital I/O LCD Output	GPIO LCD: SEG21
P4.7 SEG20	Digital I/O LCD Output	GPIO LCD: SEG20

Port 5

Pin Name	Type	Description
P5.0 SEG43	Digital I/O LCD Output	GPIO LCD: SEG43
P5.1 SEG42	Digital I/O LCD Output	GPIO LCD: SEG42
P5.2 SEG41	Digital I/O LCD Output	GPIO LCD: SEG41
P5.3 SEG40	Digital I/O LCD Output	GPIO LCD: SEG40
P5.4 SEG39	Digital I/O LCD Output	GPIO LCD: SEG39
P5.5 SEG38	Digital I/O LCD Output	GPIO LCD: SEG38
P5.6 SEG37	Digital I/O LCD Output	GPIO LCD: SEG37
P5.7 SEG36	Digital I/O LCD Output	GPIO LCD: SEG36

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	LCD Output	LCD: COM4
SEG0	LCD Output	LCD: SEG0
COM5	LCD Output	LCD: COM5
SEG1	LCD Output	LCD: SEG1
SEG2~SEG19	LCD Output	LCD: SEG2 ~ SEG19

Analog Pins

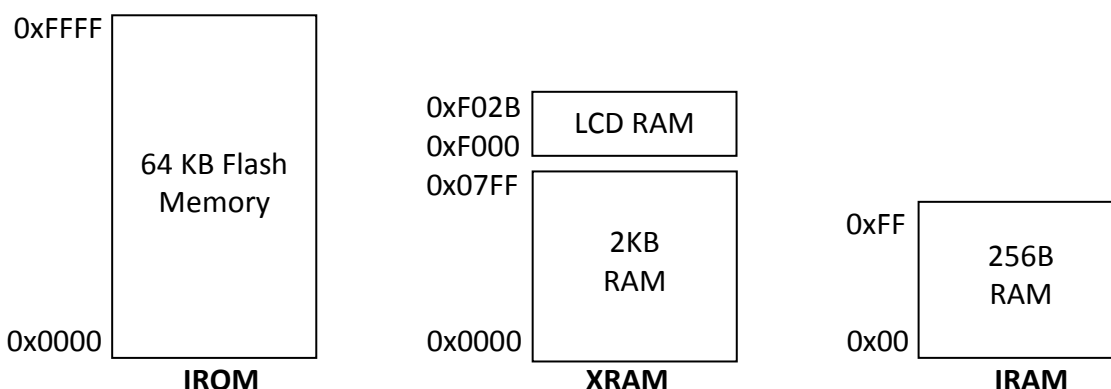
Pin Name	Type	Description
AVDD	Power	Analog power supply
AVSS	Power	Analog Ground
C+,C-	-	Charge pump Capacitor connection Pins Connect 1uF when charge pump enable
AVDDCP	Pump Output	Charge Pump output. Connect 2.2uF to DVSS when charge pump enable.
AVDDR	Analog Output	Regulator output for analog part. Connect 1uF to AVSS. Selectable: 2.7V, 3.0V, 3.3V, and 3.6V
ACM	Analog Output	ACM Voltage output 1V for ADC common voltage. (sink only) Connect 0.1uF to AVDDR.
AI1	Analog Input	ADC: Input channel 1
AI2	Analog Input	ADC: Input channel 2
AI3	Analog Input	ADC: Input channel 3
AI4	Analog Input	ADC: Input channel 4
AI5	Analog Input	ADC: Input channel 5
R+	Analog Input	ADC: external reference positive input pin
AI6	Analog Input	ADC: Input channel 6
R-	Analog Input	ADC: external reference negative input pin
AI7	Analog Input	ADC: Input channel 7
OP+	Analog Input	OPA: positive input
AI8	Analog Input	A ADC: Input channel 8
OP-	Analog Input	OPA: negative input pin
AI9	Analog Input	ADC: Input channel 9
OPO	Analog Output	OPA: output 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.

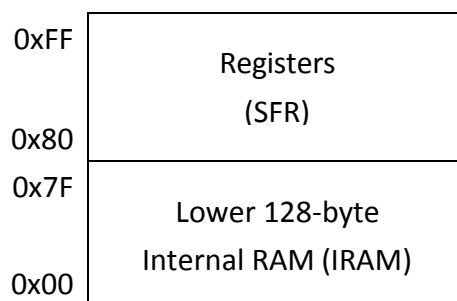
5.1 Memory Organization

SN8F5900 builds in three on-chip memories: internal RAM (IRAM), external RAM (XRAM), and program memory (IROM). The internal RAM is a 256-byte RAM which has higher access performance (direct and indirect addressing). By contrast, the external RAM has 2 KB of size, but it requires a longer access period. The program memory is a 64 KB non-volatile memory and has a maximum 16MHz speed limitation.



5.2 Direct Addressing: IRAM and SFR

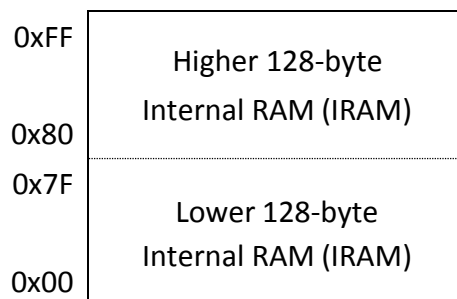
Direct addressing instructions (e.g. MOV A, direct) can access the lower 128-byte internal RAM (address range: 0x00 – 0x7F) and all registers (SFR, address range: 0x80 – 0xFF).



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.

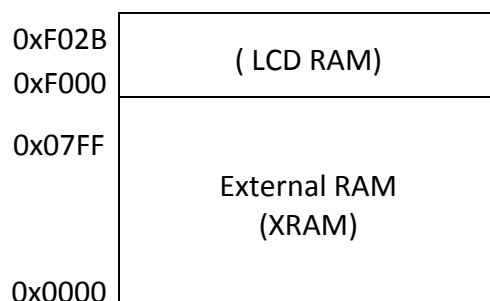
5.3 Indirect Addressing: IRAM

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).



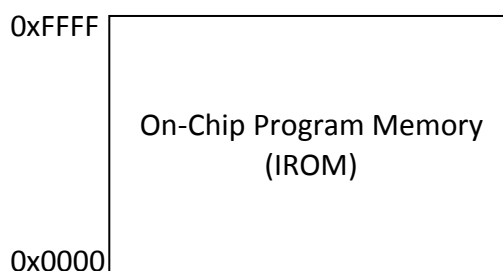
5.4 External RAM (XRAM)

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 Program Memory Security

The SN8F5900 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.

5.7 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.

5.8 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 IROM address 0x08. In other word, if a planned stack area is assigned from IROM 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.

5.9 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	-	-	I2CDAT	I2CADR	I2CCON	I2CSTA	SMBSEL	SMBDST
D0	PSW	CHS	VREG	AMPM	ADCM1	ADCM2	BZRM	LBTM
C8	-	TC1M	TC1RL	TC1RH	TC1CL	TC1CH	TC1DL	TC1DH
C0	IRCON	TC0M	TC0RL	TC0RH	TC0CL	TC0CH	TC0DL	TC0DH
B8	IEN1	IP1	SORELH	T0M	T0C	MIN	SEC	-
B0	P3	LCDM1	LCDM2			ADCDL	ADCDM	ADCDH
A8	IEN0	IPO	SORELL	-	-	-	-	-
A0	P2	TC2M	TC2RL	TC2RH	TC2CL	TC2CH	TC2DL	TC2DH
98	SOCON	SOBUF	IEN2	OPM	PEBYTE	P5COM	P3CON	P4CON
90	P1	P1W	DPS	DPC	PECMD	PEROML	PEROMH	PERAM
88	-	-	-	-	-	-	CKCON	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.
-	0x87~0x8D	-
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 lowbyte
PEROMH	0x96	In-System Program ROM address highbyte
PERAM	0x97	In-System Program RAM mapping address
S0CON	0x98	UART control register.
S0BUF	0x99	UART data buffer.
IEN2	0x9A	Interrupts enable register
OPM	0x9B	OP-AMP controls register.
PEBYTE	0x9C	In-System Program ROM Byte write address
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	PWM duty control low byte buffer
TC2DH	0xA7	PWM duty control high byte buffer
IEN0	0xA8	Interrupts enable register
IP0	0xA9	Interrupts priority register
SORELL	0xAA	UART reload low byte register
-	0xAB~0xAF	-
P3	0xB0	Port 3 data buffer.
LCDM1	0xB1	LCD Mode 1 control register.
LCDM2	0xB2	LCD Mode 2 control register.
-	0xB3	-
-	0xB4	-
ADCDL	0xB5	ADC output data Low byte.
ADCDM	0xB6	ADC output data medium byte.
ADCDH	0xB7	ADC output data high byte.
IEN1	0xB8	Interrupts enable register.
IP1	0xB9	Interrupts priority register.
SORELH	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.
-	0xBF	-

0xC0 - 0xDF Registers Description

Register	Address	Description
IRCON	0xC0	Interrupts request register.
TCOM	0xC1	TC0 control register.
TCORL	0xC2	TC0 reload buffer low byte.
TCORH	0xC3	TC0 reload buffer high byte.
TCOCL	0xC4	TC0 counter low byte.
TCOCH	0xC5	TC0 counter high byte.
TCODL	0xC6	PWM0 duty control low byte buffer.
TCODH	0xC7	PWM0 duty control high byte buffer.
-	0xC8	-
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.
CHS	0xD1	ADC input channel selection register.
VREG	0xD2	Voltage regulator control register.
AMPM	0xD3	PGIA control register.
ADCM1	0xD4	ADC control register1.
ADCM2	0xD5	ADC control register2.
BZRM	0xD6	Buzzer control register.
LBTM	0xD7	Comparator and Low battery detection control register.
(NA)	0xD8	-
(NA)	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	<p>Carry flag.</p> <p>0: Addition without carry, subtraction with borrowing signal, rotation with shifting out logic “0”, comparison result < 0.</p> <p>1: Addition with carry, subtraction without borrowing, rotation with shifting out logic “1”, comparison result ≥ 0.</p>
6	AC	R/W	0	<p>Auxiliary carry flag.</p> <p>0: If there is no a carry-out from 3rd bit of Accumulator in BCD operations.</p> <p>1: If there is a carry-out from 3rd bit of Accumulator in BCD operations.</p>
5	F0	R/W	0	General purpose flag 0. General purpose flag available for user.
4..3	RS[1:0]	R/W	00	<p>Register bank select control bit, used to select working register bank.</p> <p>00: 00H – 07H (Bnak0)</p> <p>01: 08H – 0FH (Bnak1)</p> <p>10: 10H – 17H (Bnak2)</p> <p>11: 18H – 1FH (Bnak3)</p>
2	OV	R/W	0	<p>Overflow flag.</p> <p>0: Non-overflow in Accumulator during arithmetic Operations.</p> <p>1: overflow in Accumulator during arithmetic Operations.</p>
1	F1	R/W	0	General purpose flag 1. General purpose flag available for user.
0	P	R	0	<p>Parity flag. Reflects the number of ‘1’s in the Accumulator.</p> <p>0: if Accumulator contains an even number of ‘1’s.</p> <p>1: Accumulator contains an odd number of ‘1’s.</p>

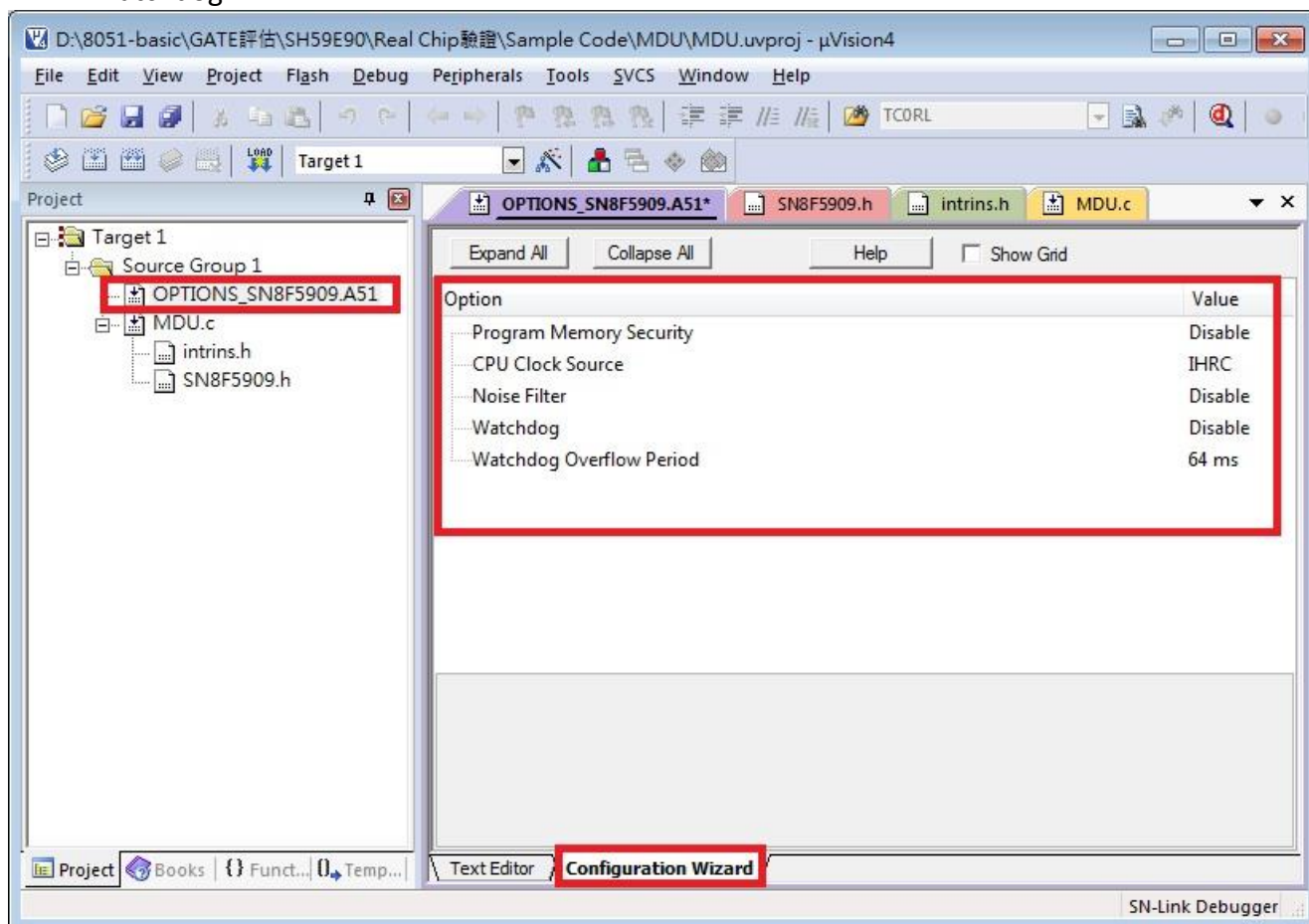
7 Reset and Power-on Controller

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 *SN8F590x_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
- Noise Filter
- Watchdog



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 intervaltime	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

In SN5900 series, there is no external reset pin.

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.

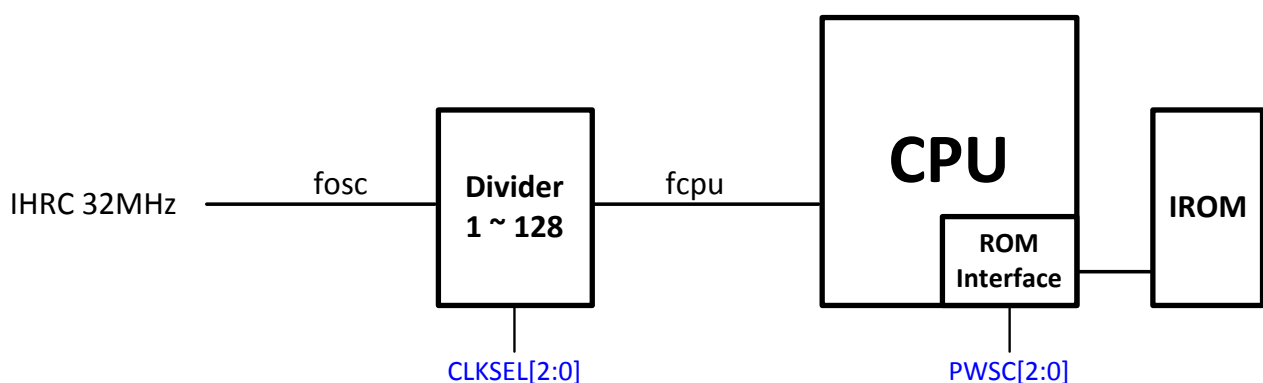
The normal mode means that CPU and peripheral functions are under normally execution. The system clock is based on the combination of source selection, clock divider, and program memory wait state. IDLE mode is the situation that temporarily suspends CPU clock and its execution, yet it remains peripherals' 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 1 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 = 0x04; // set fcpu = fosc / 8
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. For example if the CPU is anticipated to run at 32 MHz and the IROM has to run at 16 MHz, one extended cycle must be placed by CKCON register.

$$\text{IROM fetching cycle} = \frac{f_{\text{cpu}}}{\text{PWSC}[2:0] + 1} \leq 16\text{MHz}, \text{PWSC}[2:0] = 0 \sim 7$$

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

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
CKCON	-	PWSC2	PWSC1	PWSC0	ESYN	EWSC2	EWSC1	EWSC0
CLKSEL	-	-	-	-	-	CLKSEL2	CLKSEL1	CLKSEL0
CLKCMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W

CKCON Register (0x8E)

Bit	Field	Type	Initial	Description
7	Reserved	R	0	
6..4	PWSC[2:0]	R/W	111	Extended cycle(s) applied to reading program memory 000: non (set "000", if Fcpu ≤ 16MHz) 001: 1 cycle (set "001", if Fcpu > 16MHz) 010: 2 cycles (reserved) 011: 3 cycles (reserved) 100: 4 cycles (reserved) 101: 5 cycles (reserved) 110: 6 cycles (reserved) 111: 7 cycles (reserved)
3	ESYN	R/W	0	Extended extra cycles to write XRAM (user always set "0")
2..0	EWSC[2:0]	R/W	001	Extended cycle(s) applied to reading XRAM 000: non (user always set "0") 001: 1 cycle (reserved) 010: 2 cycles (reserved) 011: 3 cycles (reserved) 100: 4 cycles (reserved) 101: 5 cycles (reserved) 110: 6 cycles (reserved) 111: 7 cycles (reserved)

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: fcpu = fosc / 1 * 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..4	Reserved	R	0x00	
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

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	Request (IRQ)	IRQ Clearance	Priority / Vector
System Reset	-	-	-	0 / 0x0000
INT0	EX0	IE0	Automatically	1 / 0x0003
INT1	EX1	IE1	Automatically	2 / 0x000B
TC0	ETC0	TCF0	Automatically	3 / 0x0013
T0	ET0	TF0	Automatically	4 / 0x0093
TC1	ETC1	TCF1	Automatically	5 / 0x001B
TC2	ETC2	TCF2	Automatically	6 / 0x009B
UART	ES0	TIO / RIO	By firmware	7 / 0x0023
SPI	ESPI	SPIF / MODF	By firmware	8 / 0x00A3
ADC	EADC	ADCF	By firmware	9 / 0x002B
I2C	EI2C	SI	By firmware	10 / 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 activates. 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	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	-	-	-
Group 2	TC0	T0	-	-
Group 3	TC1	TC2	-	-
Group 4	UART	SPI	-	-
Group 5	ADC	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 prioritygroup.
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 prioritygroup.
Else	Reserved	R	0	

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

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

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

```
1  IP0 = 0x28;
2  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	-	-	ES0	-	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
IEN2	-	-	-	EADC	ETC2	ETC1	ETC0	-
IRCON	ADCF	TCF2	TCF1	TCF0	-	TF0	IE1	IE0
SOCON	SM0	SM1	SM20	RENO	TB80	RB80	TIO	RI0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CRO

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.
4	ES0	R/W	0	UART interrupt control bit. 0: Disable UART interrupt function. 1: Enable UART 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
4	EADC	R/W	0	ADC interrupt control bit 0: Disable ADC interrupt function. 1: Enable ADC interrupt function.
3	TC2	R/W	0	TC2 overflow interrupt control bit. 0: Disable TC2 interrupt function. 1: Enable TC2 interrupt function.
2	TC1	R/W	0	TC1 overflow interrupt control bit. 0: Disable TC1 interrupt function. 1: Enable TC1 interrupt function.
1	TC0	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	

S0CON Register (0X98)

Bit	Field	Type	Initial	Description
1	TIO	R/W	0	<p>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.</p> <p>0: None UART transmit interrupt request.</p> <p>1: UART transmit interrupt request.</p>
0	RIO	R/W	0	<p>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.</p> <p>0: None UART receive interrupt request.</p> <p>1: UART receive interrupt request.</p>
Else				Refer to other chapter(s)

SPSTA Register (0XE1)

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

I2CCON Register (0XDC)

Bit	Field	Type	Initial	Description
7	SI	R/W	0	<p>Serial interrupt flag</p> <p>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.</p>
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 <SN8F5909.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 shifttime 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.

```
1 #include <SN8F5909.H>
2 #pragma mdw_r515 //Keil C51 MDU command line
3
4 void main(void)
5 {
6     unsigned int Divisor; // 16-bit divisor
7     unsigned long Dividend; // 32-bit dividend
8     unsigned long Quotient; // 32-bit Quotient
9     unsigned int Remainder; // 16-bit Remainder
10
11     Divisor = 0x1234;
12     Dividend = 0x56789ABC;
13     Quotient = Dividend / Divisor; //0x0004C016
14     Remainder = Dividend % Divisor; //0x0A44
15
16     while(1);
17 }
18
```

11 GPIO

The microcontroller has up to 44 bidirectional general purpose I/O pin (GPIO). Unlike the original 8051 only has open-drain output, SN8F5909 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	-	-	-	-	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	-	-	-	P05OC	P04OC	P14OC	P13OC	P12OC

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
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	P14OC	R/W	0	P1.4 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
1	P13OC	R/W	0	P1.3 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
0	P12OC	R/W	0	P1.2 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 POM – 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	-	-	-	-	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 Resistors

The P0UR to P5UR registers are mapped to each pins' internal 100 kΩ (in typical value) pull-up resistor.

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	-	-	-	-	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 resistor control of P0.7 0: Disable* 1: Enable
6	P06UR	R/W	0	On-chip pull-up resistor control of P0.6 0: Disable* 1: Enable
5	P05UR	R/W	0	On-chip pull-up resistor control of P0.5 0: Disable* 1: Enable
4..0				et cetera

* Recommended disable pull-up resistor 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
P3CON	P3CON7	P3CON6	P3CON5	P3CON4	P3CON3	P3CON2	P3CON1	P3CON0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0

11.5 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.

```
1 #include <intrins.h>
2 #include <SN8F5909.h>
3
4 void Init_GPIO(void);
5 void Init_ISR(void);
6 void Delay1ms(unsigned int n);
7
8 void main(void)
9 {
10     Init_GPIO();           // Initial GPIO
11     Init_ISR();           // Initial interrupt setting
12
13     while (1) {
14         WDTR = 0x5A;      // clear watchdog if watchdog enable
15         // To Do...
16         Delay1ms(100);
17     }
18 }
19
20 void Init_GPIO(void)
21 {
22     P0 = 0x00;
23     P0UR = 0xFF;
24     POM = 0x00;           // P0.0, P0.1 as input mode
25 }
26
27 void Init_ISR(void)
28 {
29     PEDGE &= 0x00;        // Clear PEDGE
30     PEDGE |= 0x02;        // EX0G = 0x10 : INT0 Falling edge trigger (default).
31     EX0 = 1;              // INT0 isr enable
32
33     PEDGE |= 0x04;        // EX1G = 0x01 : INT1 Rising edge trigger
34     EX1 = 1;              // INT1 isr enable
35     EAL = 1;              // Interrupt enable
36 }
37
38 void INT0_ISR(void) interrupt ISRInt0 // Vector @0x03
39 {
40     // cleared int0 flag by hardware
41     // IE0 = 0;           // Clear INT0 flag
42
43     // To Do...
44     _nop_();
45 }
46
47
48
49
50
51
52
53
54
```

```
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) 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.

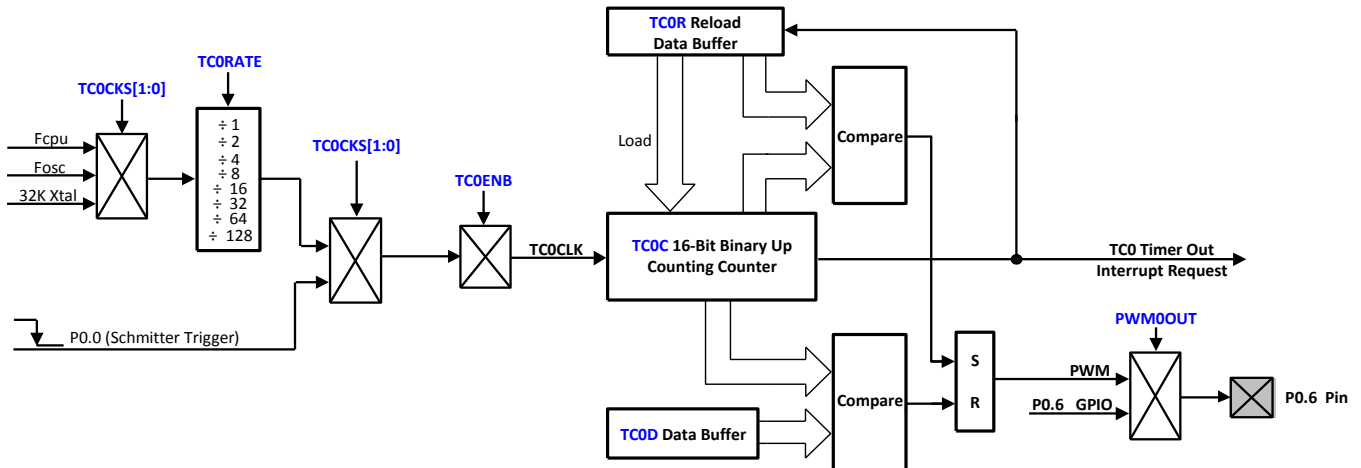
Note:

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 TC3 TATE	Timer Clock divider
x = 0, 1 or 2	

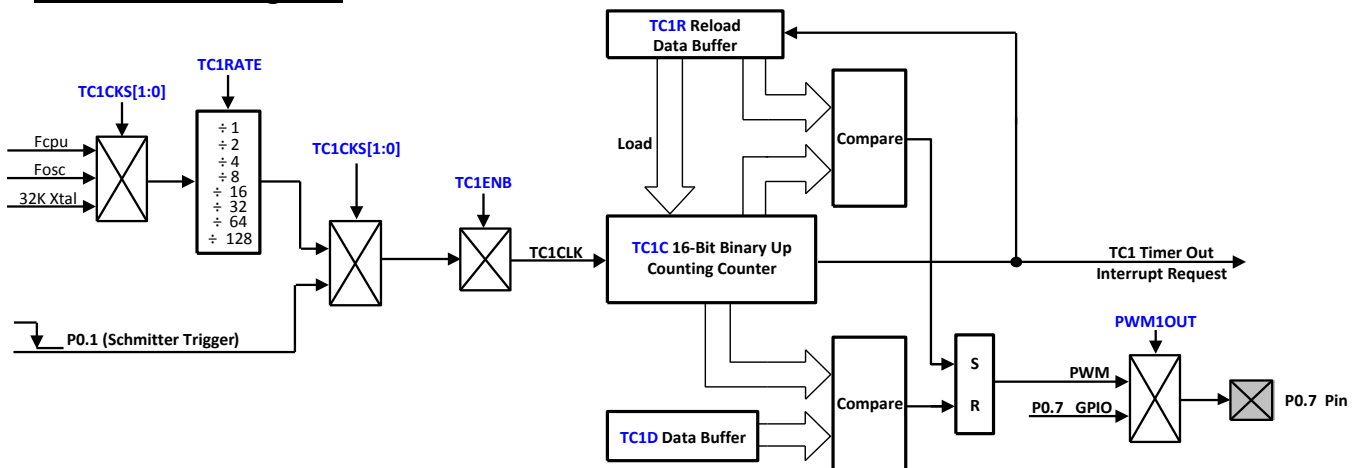
13.1 Timer Counter Diagram and Clock Source Selection

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

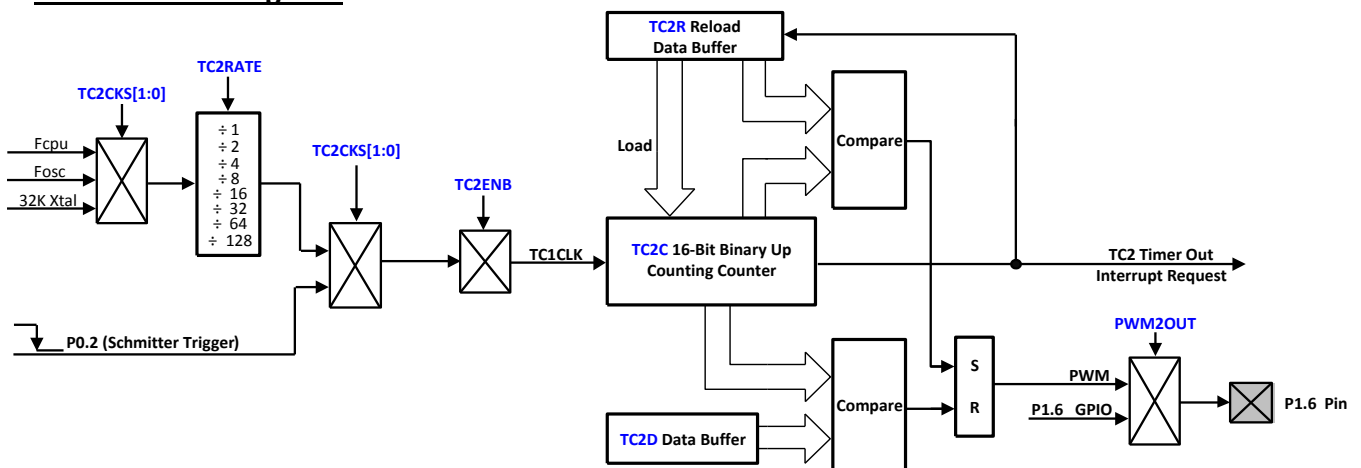
TC0 Structure Diagram:



TC1 Structure Diagram:

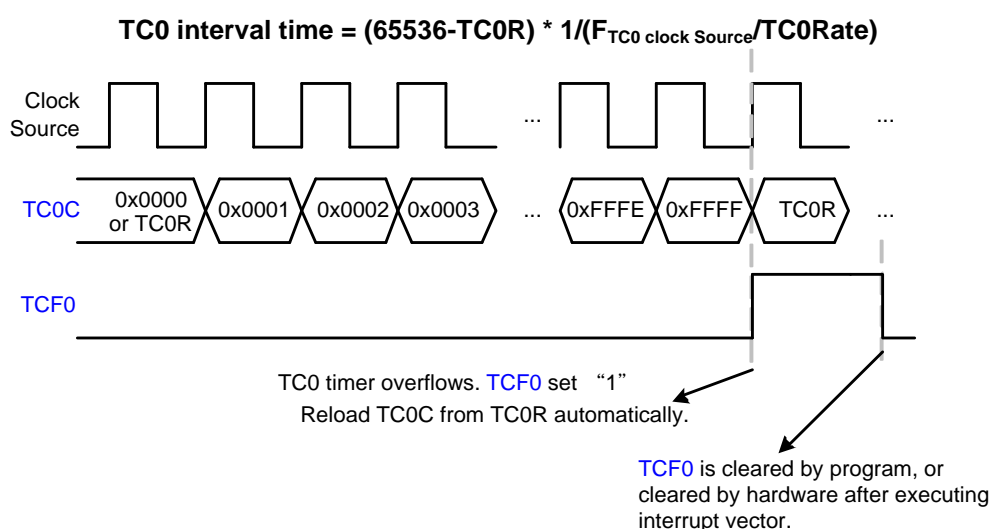


TC2 Structure Diagram:



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.



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:

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

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:

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

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

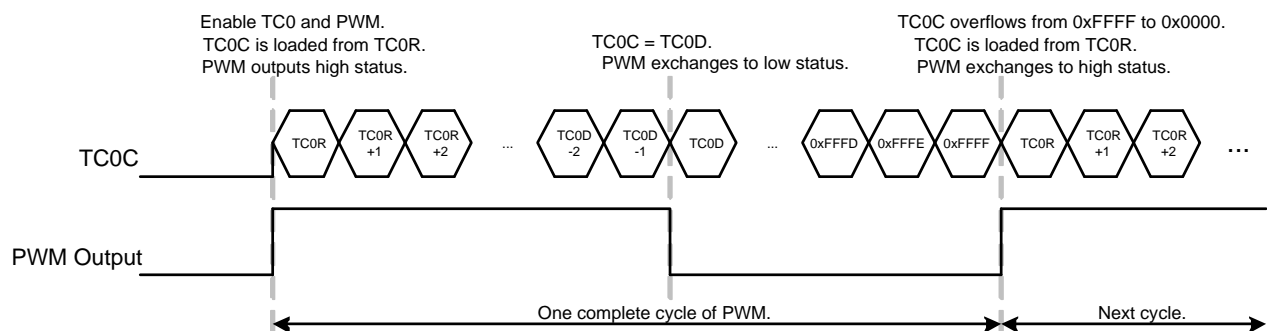
TC0 clock source is Fcpu = 32MHz/32 = 1MHz. Select TC0RATE=000 (Fcpu/128)

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

$$\begin{aligned}
 TC0C/TC0R \text{ initial value} &= 65536 - (TC0 \text{ interval time} * \text{input clock}) \\
 &= 65536 - (10\text{ms} * 32\text{MHz} / 32 / 128) \\
 &= 65536 - (10^{-2} * 32 * 10^6 / 32 / 128) \\
 &= \text{FFB2H}
 \end{aligned}$$

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:

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

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

The TC0clock source is $F_{\text{cpu}} = 32\text{MHz}/32 = 1\text{MHz}$. Select TC0RATE=000 ($F_{\text{cpu}}/128$).

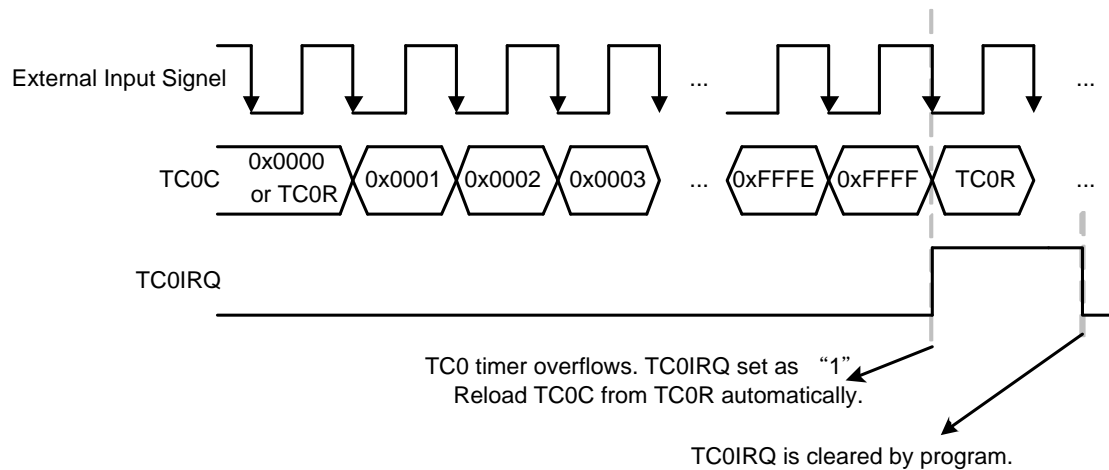
TC0R = FFB2H (TC0RH: FFH, TC0RL: B2H). TC0 interval time = 10ms. So the PWM cycle is 100Hz.

In 1/3 duty condition, the high pulse width is about 3.33ms.

$$\begin{aligned} \text{TC0D initial value} &= \text{FFB2H} + (\text{PWM high pulse width period} / \text{TC0 clock rate}) \\ &= \text{FFB2H} + (3.33\text{ms} * 32\text{MHz} / 32 / 128) \\ &= \text{FFB2H} + 1\text{AH} = \text{FFCCH} \end{aligned}$$

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	-
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

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: 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	Reserved	R	0	

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: 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: external 32K crystal 11: P0.2 (event counter)
1	PWM2OUT	R/W	0	PWM2 output control bit 0: Disable, P1.6 is GPIO mode. 1: Enable, P1.6 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

13.8 Sample Code

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

```

1 #include <intrins.h>      // for _nop_
2 #include <SN8F5909.h>
3
4 void TC0_Init(void);
5 void GPIO_Init(void);
6
7 void Init_SysCLK(void)
8 {
9     /* Clock Switch Select Register */
10    CLKSEL = 0x02;          // Fcpu = Fhosc/32
11    CLKCMD = 0x69;          // clock switch start
12
13    /* System Control Register */
14    CKCON &= 0x00;
15 }
16
17 void main(void)
18 {
19     Init_SysCLK();
20     GPIO_Init();
21     TC0_Init();
22
23     while (1) {
24         WDTR = 0x5A;        // clear watchdog if watchdog enable
25
26         // To Do...
27     }
28 }
29
30 void GPIO_Init(void)
31 {
32     POM |= 0xFF;            // P0 = Output Mode
33 }
34
35 void TC0_Init(void)
36 {
37     TCOM = 0X00;            // TC0 Clock = fcpu/128
38
39     /* TC0C/TC0R initial value =65536-(10ms*32MHz/32/128) */
40     TC0RL = 0XB2;           //AUTO-Reload Register
41     TC0RH = 0XFF;
42
43     TC0CL = 0XB2;           //TC0 COUNTING Register
44     TC0CH = 0XFF;
45
46     IEN2 |= 0x02;           // TC0IEN = 1
47     TCOM |= 0X80;           // TC0ENB = 1
48     IEN0 |= 0x80;           // EAL = 1
49 }
50
51
52
53
54

```

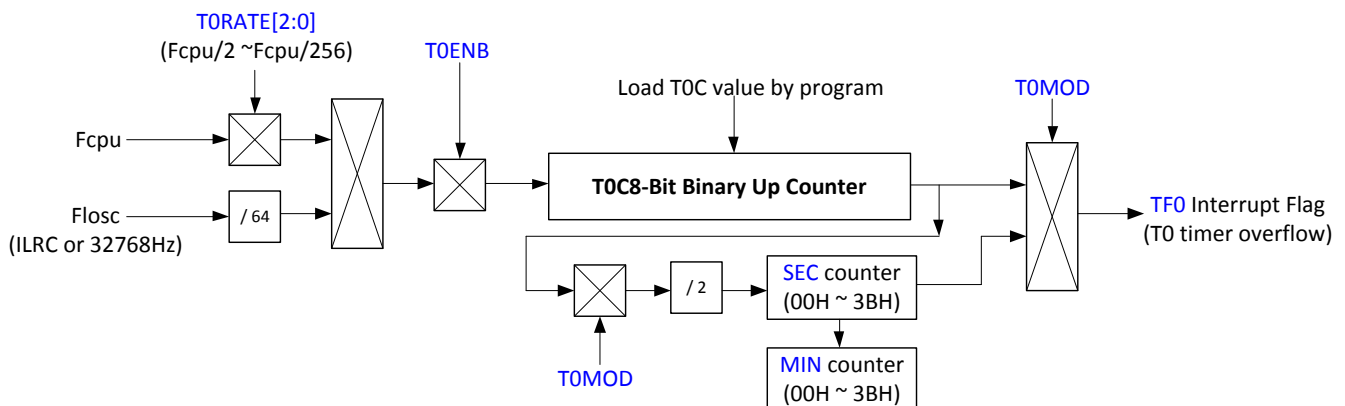


```
55 void TC0_ISR(void) interrupt ISRTC0 // Vector @ 0x13
56 {
57     // cleared TC0 interrupt flag by hardware
58     // TCF0 = 0;                // Clear TCF0 flag
59
60     // To Do...
61     P0 ^= 0xFF;                // P0 Toggle
62 }
63
```


14 Timer 0

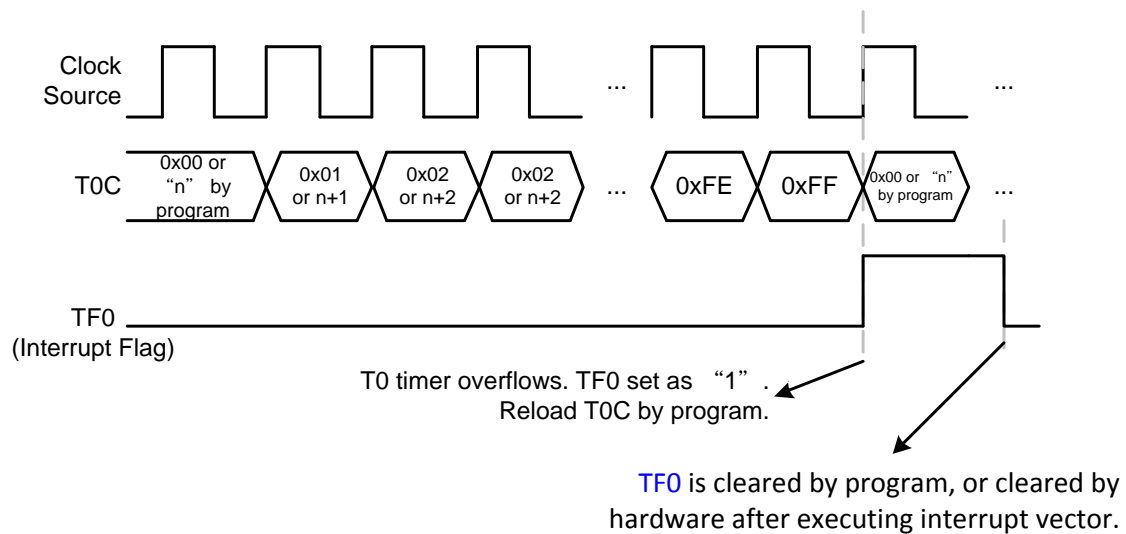
The T0 timer is an 8-bit binary up timer with basic timer function. The basic timer function supports flag indicator (TF0 bit) and interrupt operation (interrupt vector 0x93). The interval time is programmable through TOM, T0C registers and supports RTC function. The T0 builds in Stop mode wake-up function. 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 activates and the system points program counter to interrupt vector to do interrupt sequence.
- **RTC function:** T0 RTC function is controlled by T0TB bit. The RTC period is 0.5sec@32KHz when TOMOD=0. When TOMOD=1, T0 interrupt period is 60sec @32KHz.
- **Stop mode function:** T0 keeps running in stop mode and can wake-up from STOP mode as **TOENB = 1 & T0TB = 1 @IHRC_RTC** System will be Wake-up when TF0 activates 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 $\text{cpu}/2 \sim \text{Fcpu}/256$. 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	TOENB	R/W	0	T0 timer enable bit. 0 : Disable. 1 : Enable.
6..4	TORATE[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 Fcpu. 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:

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

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

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

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

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

$$= 256 - (10\text{ms} * 32\text{MHz} / 32 / 128) = 178 = \text{0xB2.}$$

14.4 Other Relative Register

SEC Register (0xBD)

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 (0xBE)

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 (0xBC)

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.

```

1
2 #include <intrins.h>          // for _nop_
3 #include <SN8F5909.h>
4
5 void T0_Init(void);
6
7 void Init_SysCLK(void)
8 {
9     /* Clock Switch Select Register */
10    CLKSEL = 0x02;           // Fcpu = Fhosc/32
11    CLKCMD = 0x69;          // clock switch start
12
13    /* System Control Register */
14    CKCON &= 0x00;
15 }
16
17 void main(void)
18 {
19     Init_SysCLK();
20     T0_Init();
21
22     while (1) {
23         WDTR = 0x5A;        // clear watchdog if watchdog enable
24
25         // To Do...
26     }
27 }
28
29 void T0_Init(void)
30 {
31     POM |= 0xFF;
32     TOM |= 0x10;            // Fcpu/128
33     T0C = 0xB2;            // T0C initial value = 256 - (T0 interal time * input clock)
34                             // T0C initial value = 256 - (10ms*32MHz/32/128) = 0xB2
35
36     IEN0 |= 0x02;          // T0IEN = 1
37     TOM |= 0x80;           // T0ENB = 1
38     IEN0 |= 0x80;          // EAL = 1
39 }
40
41 void T0_ISR(void) interrupt ISRTimer0 // Vector @0x93
42 {
43     // cleared TC0 interrupt flag by hardware
44     // TCF0 = 0;           // Clear TCF0 flag
45
46     T0C = 0xB2;            // Reload data to T0C
47     P0 ^= 0xFF;           // P0 Toggle
48 }
49
50
51
52
53
54

```

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

```
1
2 #include <intrins.h>      // for _nop_
3 #include <SN8F5909.h>
4
5 void T0_RTC_Init(void);
6 void GPIO_Init(void);
7
8 void Init_SysCLK(void)
9 {
10     /* Clock Switch Select Register */
11     CLKSEL = 0x02;        // Fcpu = Fhosc/32
12     CLKCMD = 0x69;        // clock switch start
13
14     /* System Control Register */
15     CKCON &= 0x00;
16 }
17
18 void main(void)
19 {
20     Init_SysCLK();
21     GPIO_Init();
22     T0_RTC_Init();
23
24     while (1) {
25         WDTR = 0x5A;        // clear watchdog if watchdog enable
26         STOP();              // System into STOP MODE
27     }
28 }
29
30 void GPIO_Init(void)
31 {
32     P0 = 0X00;
33     POM |= 0xFF;
34 }
35
36 void T0_RTC_Init(void)
37 {
38     TOM |= 0X01;            // T0TB = 1, Clock source must from 32768 Hz X'tal
39                             // Timer interrupt is occur when 0.5SEC overflow.
40
41     /* 60s Interrupt function */
42     //TOM |= 0X02;          // T0MOD = 1, Timer interrupt is occur when 60SEC overflow.
43                             // "SEC" = 00H~3BH(0~59)
44
45     IEN0 |= 0x02;           // T0IEN = 1
46     TOM |= 0XF0;           // T0ENB = 1
47     IEN0 |= 0x80;           // EAL = 1
48 }
49
50
51
52
53
54
```

```
55 void T0_ISR(void) interrupt ISRTimer0 // Vector @0x93
56 {
57     // cleared TC0 interrupt flag by hardware
58     // TCF0 = 0;           // Clear TCF0 flag
59
60     P0 ^= 0xFF;           // P0 Toggle
61 }
62
```

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 P1.5 Pin with 50% duty, and output frequency is controllable by setting the BZRCKS[1:0] register.

BZRM Register (0xD6)

Bit	Field	Type	Initial	Description
2..1	BZRCKS[1:0]	R/W	0	Buzzer output frequency 00 : 0.98KHz. 01 : 1.96KHz. 10 : 3.9KHz. 11 : 7.8KHz
0	BZRENB	R/W	0	Buzzer output control bit. 0 : GPIO Mode. 1 : Buzzer Mode. P15 output Buzzer signal.

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

```

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

```


16 LCD Driver

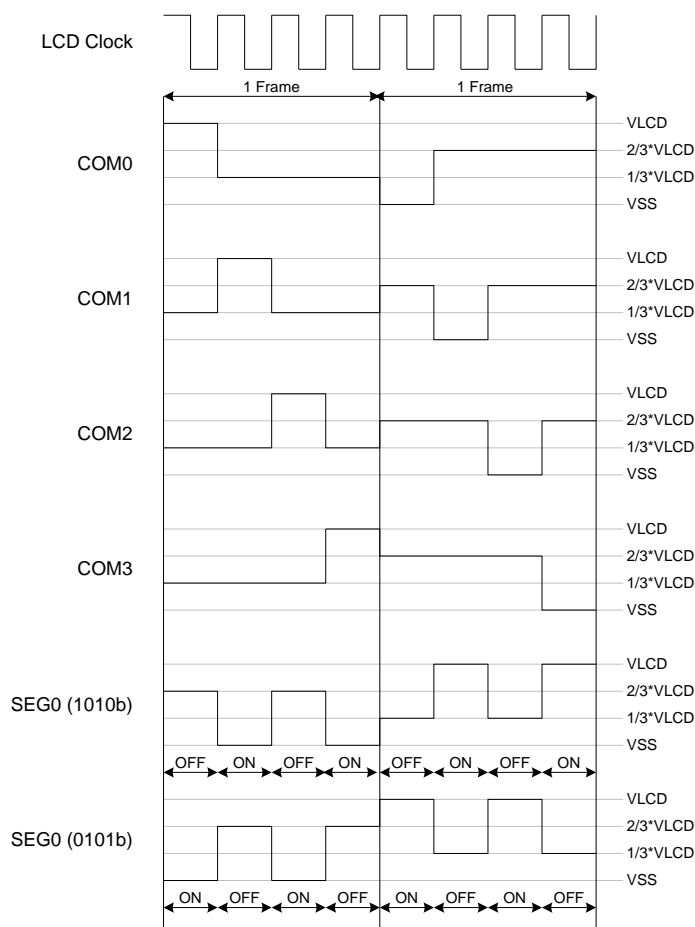
LCD driver is C-type structures with 4 x 44 or 6 x 42. The LCD scan timing is 1/4 or 1/6 duty with 1/3 bias only, to drive LCD of 176 dots or 252 dots. C-type LCD driver can output adjustable VLCD output voltage and adjustable driving power to support diversity of LCD panel. VLCD Pin must connect 0.1uF 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:

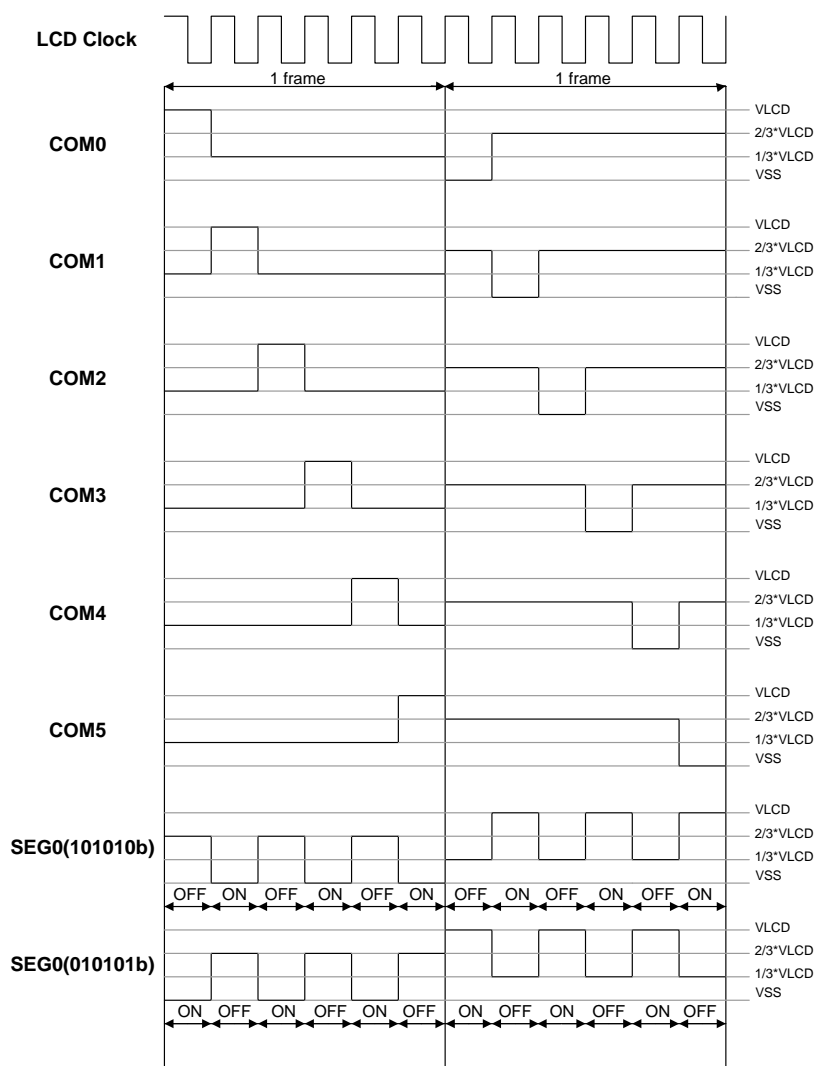
- **Stop mode function:** LCD function keeps running in stop mode as LCDEN = 1 & STOP = 1.
- **Low power mode function:** LCD low power function is controlled by BGM bit.

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 0xF02B.

LCD RAM location relate to COM0 ~ COM3 vs. SEG0 ~ SEG43 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 43	F02BH.0	F02B H.1	F02B H.2	F02B H.3	N/A	N/A	N/A	N/A

LCD RAM location relate to COM0 ~ COM5 vs. SEG2 ~ SEG43 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 43	F02BH.0	F02B H.1	F02B H.2	F02B H.3	F02B H.4	F02B 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	VAR1	VAR0	BGM	DUTY1	DUTY0	VCP2	VCP 1	VCP0

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
7..6	VAR[1:0]	R/W	0	VLCD Pump output ripple control bit. 00 : $\pm 30\text{mV}$. (please set "00") Others : reserve
5	BGM	R/W	1	LCD low power mode control bit. 0 : Enable low power mode. 1 : Normal operation mode.
4..3	DUTY[1:0]	R/W	0	LCD pump output driving capacity control bits. 00 : 25% 10 : 50% 01 : 37.5% 11 : 100% *Note: DUTY [1:0] is controllable when BGM=0, which is set for LCD low power consumption.
2..0	VCP[2:0]	R/W	011	VLCD output voltage control bits. 000 : 2.6V 011 : 3.2V 001 : 2.8V 100 : 3.6V 010 : 3.0V 101 : 4.5V Others: reserved. *Note: Battery voltage must be greater than 1.9V. @VLCD output voltage 2.6~3.6V

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

```

1    LCDM2 &= 0xE7;    // Driving capacity = 25%.
2    LCDM2 &= 0xDF;    // Enable Low power Mode.
```

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 (SEG35 ~ SEG28) 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 (SEG27 ~ SEG20) 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 (SE43 ~ SEG36) 1 : Set as GPIO function (P50 ~ P57)

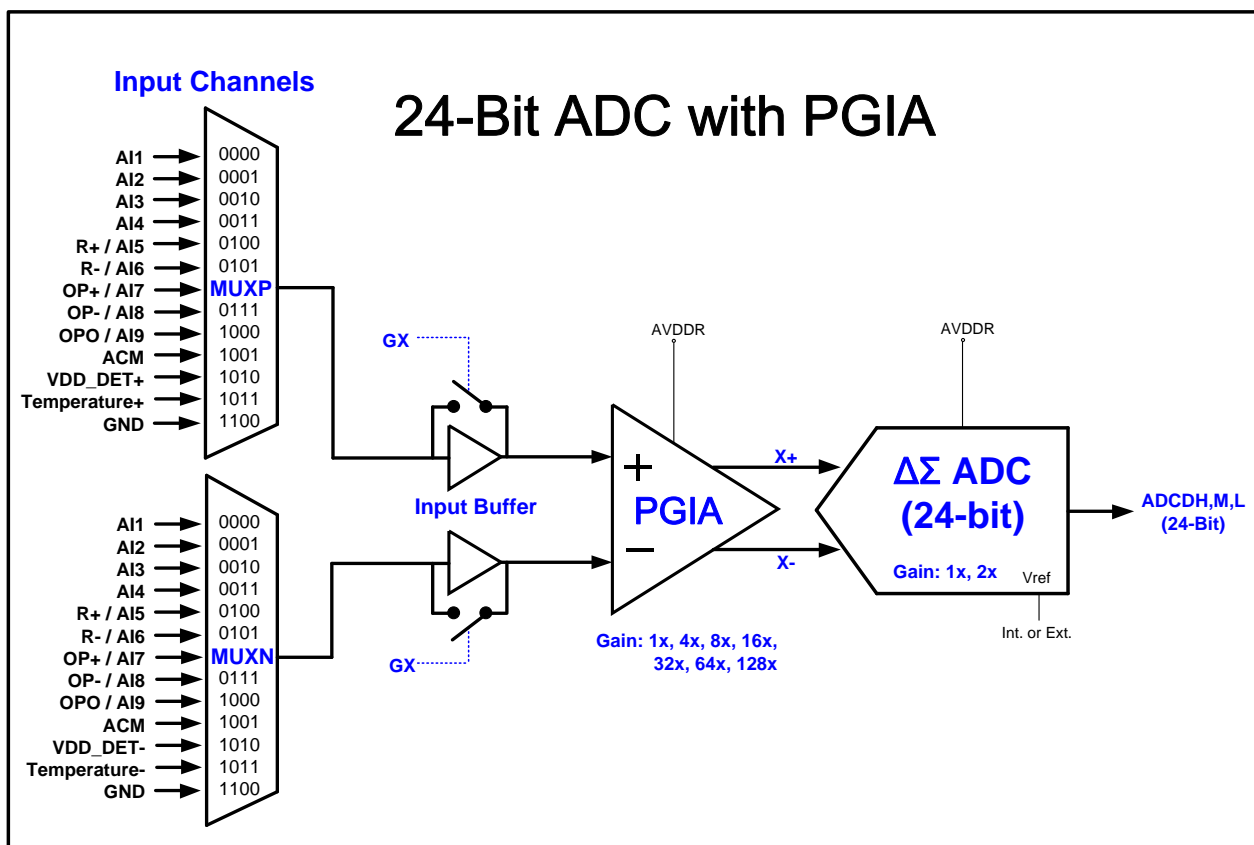
16.3 Sample Code

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

```
1
2 void Init_LCD(void)
3 {
4     LCDM1 &= 0x00;      // Clear LCDM1
5     LCDM1 |= 0x02;      // Enable LCD Pump
6     Delay1ms(1);        // Dealy 1m sec
7     LCDM1 &= 0xF3;      // LCD Clock=256HZ,Frame=64Hz @4-COM
8
9     LCDM2 &= 0x00;      // Clear LCDM2
10    LCDM2 &= 0xDF;      // Enable Low power Mode @STOP MODE
11    LCDM2 |= 0x03;      // VLCD=3.2V
12    LCDM2 &= 0xE7;      // Low power mode Driving capacity = 25%.
13
14    LCDM1 |= 0x01;      // Enable LCD
15 }
16
17 void main(void)
18 {
19     CLEAR_LCD_RAM();    // Clear LCD RAM
20     Init_LCD();         // Initial LCD
21     while (1)
22     {
23         // To Do...
24         STOP();         // Use C51 Macros into STOP MODE
25     }
26 }
27
28
```

17 ADC

The microcontroller integrates high resolution $\Delta \Sigma$ Analog-to-Digital Converter (ADC) and low noise programmable gain amplifier (PGA), which output 24-bit resolution with 18-bit noise free bit accuracy. ADC conversion rate can output from 2Hz to 7.8 kHz, and ADC reference voltage can set from internal or external (R+ / R-). The ADC operates in normal mode and idle mode, which works with interrupt or wakeup function after every ADC end of conversion if interrupt enable. ADC has internal Gain option with selective range of x1 and x2. The PGA input through configured 2 Multiplexer, MUXP and MUXN, provides differential-types input and single-end-type input. This ADC is optimized for measuring low-level unipolar or bipolar signals in sensory measurement and medical applications. A very low noise chopper-stabilized programmable gain amplifier (PGA) with selectable gains of 1x, 4x, 8x, 16x, 32x, 64x, and 128x in the ADC to accommodate these applications.



17.1 Configurations of Operation

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

1. Analog Power AVDDR, ACM and Bandgap must enable. (By AVDDREN, ACMEN, BGREN bits)
2. Choose the ADC input channels. (By MUXP[3:0] and MUXN[3:0] bits)
3. Input Buffer setting: Turn-off if PGIA is applied in the application. Turn-On if PGIA turn-off and sensor requires condition of higher input-impedance. (By GX bit)
4. PGIA enable if required, and Set PGIA Gain ratio (By PGAEN, GS[2:0] bits)
5. Turn on Chopper function and clock of PGIA and ADC (By PCHPEN, ACHPEN[1:0], AMPCKS[1:0] bits)
6. Set ADC reference voltage from internal voltage or external R+/R-. (By IRVS[2:0] bits)
7. Set ADC conversion rate through ADC clock and OSR Over-Sample Rate. (By ADCKS[1:0] and OSR[2:0] bits)
8. Set ADC interrupt function if necessary, and Enable ADC. (By EADC and ADCEN bits)
9. After enable ADENB bit, the ADC ready to convert analog signal to digital data. (24-bits ADCDH, ADCDL, ADCDL)

17.2 ADC Input Channels and PGIA

The ADC builds in 9 external channels input source and 4 internal channels input source, which (AI1 – AI9) to measure 9 different analog signal sources controlled by CHS[4:0] bits. 4 Internal channels including ACM, GND, VDD_DET (VDD detection), and Temperature channels. AI7~AI9 channels share with OPA input and output terminals. The ADC external reference voltage is input via AI5 and AI6. The VDD_DET channel is 1/8*VDD voltage input ADC to measure battery power by ADC. Temperature sensor is also embedded in the IC to measure temperature around the IC or room temperature roughly.

The microcontroller includes a very low noise chopper-stabilized programmable gain amplifier (PGIA) with selectable gains of 1x, 4x, 8x, 16x, 32x, 64x, and 128x, which inputs can be configured as a differential type or single-end type. The PGIA has high input-impedance characteristic (around 5G-Ohm), which is suitable applied or connect to sensor with higher input impedance.

Analog input signal channel selection table:

<u>MUXP[3:0]</u>	PGIA Positive input	<u>MUXN[3:0]</u>	PGIA Negative input
0000	AN1	0000	AN1
0001	AN2	0001	AN2
0010	AN3	0010	AN3
0011	AN4	0011	AN4
0100	AN5 / R+	0100	AN5 / R+
0101	AN6 / R-	0101	AN6 / R-
0110	AN7 / OP+	0110	AN7 / OP+
0111	AN8 / OP-	0111	AN8 / OP-
1000	AN9 / OPO	1000	AN9 / OPO
1001	ACM	1001	ACM
1010	VDD_DET+	1010	VDD_DET-
1011	Temperature+	1011	Temperature-
1100	AVss	1100	AVss
Others	NA.	Others	NA.

17.3 Analog Input Buffer

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

17.4 ADC Reference Voltage

ADC reference voltage can set from external R+ / R- input or from internal voltage source input, which control by setting IRVS[2:0] bits of ADCM1 register, and shows the Vref setting in following:

<u>IRVS[2:0]</u>		ADC Reference Voltage			
		AVDDR 2.7V	AVDDR 3.0V	AVDDR 3.3V	AVDDR 3.6V
0xx	External	R+ / R-			
100	0.2*AVDDR	0.54V	0.6V	0.66V	0.72V
101	0.3*AVDDR	0.81V	0.9V	0.99V	1.08V
110	0.4*AVDDR	1.08V	1.2V	1.32V	1.44V
111	0.5*AVDDR	1.35V	reserved	reserved	reserved

17.5 ADC Gain and ADC Output Code

The ADC builds in internal Gain Option with selective range of x1 and x2 for additional signal amplification expect PGIA. The ADC Gain setting is controlled by ADGN [2:0] bits in register ADCM1.

The ADC conversion output 24-Bit data which is combined with ADCDH, ADCDM, and ADCDL register in 2's compliment with sign bit numerical format, and Bit ADCB23 is the sign bit of ADC data. Refer to following formula to calculate ADC conversion data value. The following shows ADC output code calculation:

24-Bit format, ADC output code in ADCDH, ADCDM and ADCDL:

$$\frac{\Delta \text{Input} \times \text{PGIA_Gain} \times \text{ADC_Gain}}{\text{Vref}} \times 2^{(24-1)} = + (2^{(24-1)} - 1) \sim -2^{(24-1)}$$

PGIA_Gain : x1 ~ x128

ADC_Gain : x1 ~ x2

Vref : 0.3V ~ 1.5V

16-Bit format, ADC output code in ADCDH and ADCDM:

$$\frac{\Delta \text{Input} \times \text{PGIA_Gain} \times \text{ADC_Gain}}{\text{Vref}} \times 2^{(16-1)} = + 32767 \sim -32768$$

PGIA_Gain : x1 ~ x128

ADC_Gain : x1 ~ x2

Vref : 0.3V ~ 1.5V

17.6 ADC Interrupt Control

When the ADC converting successfully, the ADCF will be set to "1" no matter the EADC is enabled or not. If the EADC and the trigger event ADCF is set to be "1". As the result, the system will execute the interrupt vector. If the EADC = 0, the trigger event ADCF is still set to be "1". Moreover, the system won't execute interrupt vector even when the EADC is set to be "0". Users need to be cautious with the operation under multi-interrupt situation.

17.7 ADC Conversion Rate

The ADC provides variable output conversion rate from 2Hz up to 7.8 kHz, which conversion rate is decided by setting bits of ADCKS [1:0] and OSR [2:0] in ADCM2 register. Adjust ADC clock (ADCKS) and OSR can get suitable ADC output word rate. For High resolution application, OSR set maximum value of 32768 recommended. The ADC output code with slow output word rate is more stable than fast one. In ADC's application, that should be tradeoff between ADC's output word rate and stability (ENOB). The following table shows the ADC output word rate with setting:

$$\text{ADC Conversion Rate} = \text{ADC Clock} / \text{OSR}$$

ADC output stable data at the 3rd data after ADC enable or input channel switch, which the 1st and 2nd ADC output data are unstable data. The 3rd, 4th , 5th ... are stable data.

ADC Conversion Rate Table:

<u>ADCKS [1:0]</u>	<u>OSR [2:0]</u>	ADC clock	Conversion Rate	<u>ADCKS [1:0]</u>	<u>OSR [2:0]</u>	ADC clock	Conversion Rate
00	000	500KHz	7.8 kHz	01	000	125KHz	1.95 kHz
00	001		3.9 kHz	01	001		976 Hz
00	010		1.95 kHz	01	010		488 Hz
00	011		488 Hz	01	011		122 Hz
00	100		122 Hz	01	100		30.5 Hz
00	101		61 Hz	01	101		15.2 Hz
00	110		30.5 Hz	01	110		7.6 Hz
00	111		15.2 Hz	01	111		3.8 Hz
01	000	250KHz	3.9 kHz	11	000	62.5KHz	976 Hz
01	001		1.95 kHz	11	001		488 Hz
01	010		976 Hz	11	010		244 Hz
01	011		244 Hz	11	011		61 Hz
01	100		61 Hz	11	100		15.2 Hz
01	101		30.5 Hz	11	101		7.6 Hz
01	110		15.2 Hz	11	110		3.8 Hz
01	111		7.6 Hz	11	111		1.9 Hz

17.8 ADC Control Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CHS	MUXP3	MUXP2	MUXP1	MUXP0	MUXN3	MUXN2	MUXN1	MUXN0
VREG	BGREN	AVDDRS1	AVDDRS0	AVDDREN	ACMEN	CCKS	CPMOD1	CPMOD0
AMPM	GX	AMPCKS1	AMPCKS0	GS2	GS1	GS0	PCHPEN	PGIAEN
ADCM1	GR	IRVS2	IRVS1	IRVS0	ADGN	ACHPEN1	ACHPEN0	ADCEN
ADCM2	-	-	ADCKS1	ADCKS0	OSR2	OSR1	OSR0	DRDY
ADCDH	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16
ADCDM	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8
ADCDL	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0
IEN2	-	-	-	EADC	ETC2	ETC1	ETC0	-
IRCON	ADCF	TCF2	TCF1	TCF0	-	TF0	IE1	IE0

CHS Register (0xD1)

Bit	Field	Type	Initial	Description
7..4	MUXP[3:0]	R/W	0	ADC positive input channel selection bits. 0000 : AI1 0101 : AI6/R- 1010 : VDD_DET+ 0001 : AI2 0110 : AI7/OP+ 1011 : Temperature+ 0010 : AI3 0111 : AI8/OP- 1100 : AVSS 0011 : AI4 1000 : AI9/OPO others : N/A 0100 : AI5/R+ 1001 : ACM
3..0	MUXN[3:0]	R/W	0	ADC negative input channel selection bits. 0000 : AI1 0101 : AI6/R- 1010 : VDD_DET- 0001 : AI2 0110 : AI7/OP+ 1011 : Temperature- 0010 : AI3 0111 : AI8/OP- 1100 : AVSS 0011 : AI4 1000 : AI9/OPO others : N/A 0100 : AI5/R+ 1001 : ACM

VREG Register (0xD2) Refer to chapter of *“Charge pump regulator”*.

AMPM Register (0xD3)

Bit	Field	Type	Initial	Description
7	GX	R/W	0	ADC Input buffer enable bit. 0 : Disable 1 : Enable (When PGIA Gain set x1)
6..5	AMPCKS[1:0]	R/W	00	PGIA chopper frequency control bits. 00 : ADCKS / 128 10 : ADCKS / 16 01 : ADCKS / 32 11 : ADCKS / 8 (please set "01" for all application)
4..2	GS[2:0]..	R/W	000	PGIA Gain selection bits. 000 : 1x 100 : 32x 001 : 4x 101 : 64x 010 : 8x 110 : 128x 011 : 16x 111 : reserved
1	PCHPEN	R/W	0	PGIA chopper enable bit. 0 : Disable 1 : Enable. (must enable when PGIA enable)
0	PGIAEN	R/W	0	PGIA function enable bit. 0 : Disable 1 : Enable.

***Example:** When PGIA Gain set 1x application, the AI+/AI- signal will bypass PGIA and input ADC directly. PGIA can be disabled (PGIAEN=0) for power saving, and input buffer of ADC must be enabled (GX=1) for input high impedance characteristic of ADC.

```

1    AMPM |= 0x10;        // Enable X+/X- Unit Gain Buffer @GAIN 1x1.
2    AMPM &= 0xE3;        // PGIA x 1, GX Buff always set 1.
3    AMPM |= 0x01;        // Enable PGIA.
```

ADCM1 Register (0xD4)

Bit	Field	Type	Initial	Description
7	GR	R/W	0	ADC reference buffer enable bit. 0 : Disable 1 : Enable (When ADC Vref set from external pin R+ and R-.)

6..4	IRVS[2:0]	R/W	000	ADC reference voltage selection bits. 0xx : external R+/R- 100 : Internal 0.2 x AVDDR 101 : Internal 0.3 x AVDDR 110 : Internal 0.4 x AVDDR 111 : Internal 0.5 x AVDDR
3	ADGN	R/W	0	ADC Gain selection bit. 0 : 1x 1 : 2x
2..1	ACHPEN[1:0]	R/W	00	ADC chopper control bit2. 11 : Enable. (must set "11" for all applications)
0	ADCEN	R/W	0	ADC function enable bit. 0 : Disable 1 : Enable.

***Example:** When ADC Vref set from external pin R+ and R-, and R+/R- input buffer of ADC Vref must be enabled (GR=1) for input high impedance characteristic of ADC Vref.

```

1  ADCM1 &= 0x8F;    // EXT Vref
2  ADCM1 |= 0x80;    // Enable GR Unit Gain Buffer.@EXT Vref
3  ADCM1 |= 0x01;    // Enable ADC

```

ADCM2 Register (0xD5)

Bit	Field	Type	Initial	Description
5..4	ADCKS[1:0]	R/W	0	ADC clock selection bit. 00 : 500KHz 10 : 125KHz 01 : 250KHz 11 : 62.5KHz (set 250KHz recommend)
3..1	OSR[2:0]	R/W	000	ADC Over-Sampling-Rate selection bits. 000 : 64 100 : 4096 001 : 128 101 : 8192 010 : 256 110 : 16384 011 : 1024 111 : 32768
0	DRDY	R/W	0	ADC function enable bit. 0 : ADCDH, ADCDL, and ADCDLL conversion data are not ready. 1 : ADC output (update) new conversion data to ADCDH, ADCDL, and ADCDLL.

ADCDH Register (0xB7), ADC output high byte register.

ADCDM Register (0xB6), ADC output medium byte register.

ADCDL Register (0xB5), ADC output low byte register.

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.
	others	R/W	0	Refer to interrupt chapter.

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.
	others	R/W	0	Refer to interrupt chapter.

17.9 ADC Electrical Characteristic

SYM.	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
I _{ADC}	Operating current	Run mode @ 2.4V	-	200	-	uA
I _{PDN}	Power down current	Stop mode @ 2.4V	-	0.1	-	μA
F _{SMP}	Conversion rate (WR)	ADC Clock=62.5KHz, OSR=32768	-	1.9	-	Hz
		ADC Clock=500KHz, OSR=64	-	7.8	-	KHz
T _{ADCSTL}	ADC settling Time	3*(1/WR), if WR=61Hz, T _{ADCSTL} = 3*16.4ms = 49.2ms	3			WR
V _{REF}	Reference Voltage Input Voltage	External V _{REF} Input Range (R+ - R-)	0.3		1.35	V
		Internal V _{REF} Input Range.	0.3		1.35	V
V _R	ADC Reference signal absolutely voltage	GR=1, R+ and R- absolutely input Voltage	0.4		AVDDR-1V	V
		GR=0, R+ and R- absolutely input Voltage	0.4		AVDDR-1V	V
V _{AI}	ADC Input signal absolutely voltage	GX=1, AI absolutely input Voltage	0.4		AVDDR-1V	V
		GX=0, AI absolutely input Voltage	0		AVDDR-1V	V
V _x	PGIA output signal absolutely voltage	X+ and X- absolutely output Voltage(GX=0)	0.4		AVDDR-1	
DNL	Differential non-linearity	ADC range ± 131072 × 0.9. (0.9 × Vref , 18-bits)		± 2		LSB
INL	Integral non-linearity	ADC range ± 131072 × 0.9. (0.9 × Vref , 18-bits)		± 4		LSB
NMC	No missing code	ADC range ± 131072 × 0.9. (0.9 × Vref , 24-bits)		18		bit
NFB	Noise free bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		18.5		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		15.5		bit
ENOB	Effective number of bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		21		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		18		bit
V _{AIN}	ADC Input differential range	ADC input signal, signal after PGIA application	0.3		1.44	V
T _{Drift}	ADC Temperature Drift	AVDDR = 2.7V, PGIA × 16, T= 0 ~ 50 °C		30		PPM/°C

SN8F5909 ADC Performance ENOB (Noise Free Bit) vs. Output Rate and Gain								
OSR	32768	16384	8192	4096	1024	256	128	64
WR	7.6Hz	15Hz	30.5Hz	61Hz	244Hz	976Hz	1.9KHz	3.9K
128x1	15.4	15	14.5	14	13.1	12.0	11.5	10.8
64x1	16.4	15.9	15.3	14.8	13.9	12.9	12.3	11.1
32x1	17.1	16.4	15.9	15.5	14.1	13.6	13.0	11.2
16x1	17.6	17.0	16.5	15.9	15.0	14.0	13.2	11.3
8x1	17.7	17.3	16.7	16.2	15.2	14.2	13.3	11.5
4x1	17.7	17.3	16.8	16.3	15.2	14.3	13.3	11.5
1x1	18.3	17.8	17.3	16.7	15.7	14.8	13.3	11.5

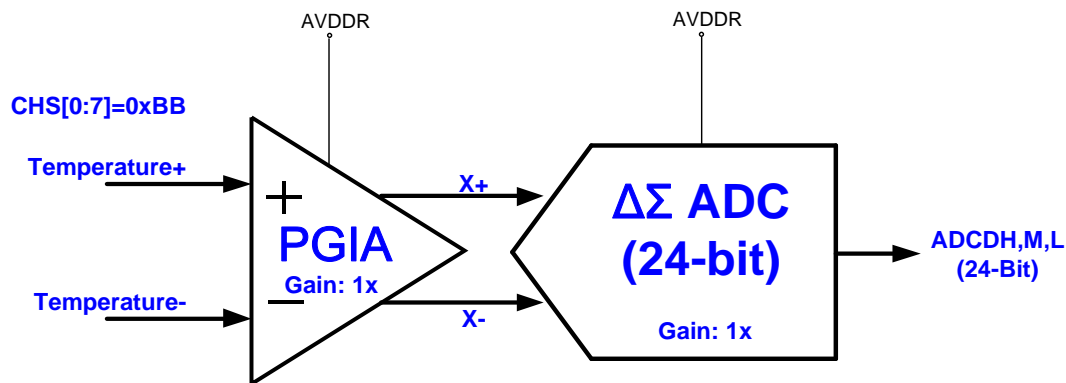
* Buffer off (GX=0, GR=0)

All Test condition: ADC 250kHz, Input-Short, Vref=0.81V, Gain = PGIA × ADC, Collect 1024 ADC date.

- Noise Free Resolution = Log2 (Full Scale Range / Peak-Peak Noise)
where Full Scale Range = 2 × Vref / Gain (ex. Vref=0.81V, Gain=1x~128x)
- Effective Resolution = Log2 (Full Scale Range / RMS_Noise)
- RMS Noise = $\sigma \times \text{LSB_Resolution}$ where $\text{LSB_Resolution} = \text{Full Scale Range} / 2^{\text{Bit}}$, Bit=24
 σ = standard deviation of 1024 ADC output data.
- Peak-Peak Noise = 6.6 × RMS Noise, or code variation range × LSB_Resolution where Code variation range = ADC counts max-min of 1024 data.

17.10 Temperature Sensor (TS)

In applications, sensor characteristic might change in different temperature also. To get the temperature information, SN8F5909 build in a temperature sensor (TS) for temperature measurement. Select the respective PGIA channel to access the Temperature Sensor ADC output.



- * **Note 1:** When selected Temperature Sensor, PGIA gain must set to 1x, or the result will be incorrect.
- * **Note 2:** The Temperature Sensor was just a reference data not real air temperature. For precision application, please use external thermistor sensor.

In 25°C, V(TS) will be about V_{TS} Offset typically, and if temperature rise 10°C, V(TS) will increase about 18mV ($V_{TS} = V_{TS} \text{ Offset} + 18\text{mV}$), if temperature drop 10°C, V(TS) will decrease about 18mV ($V_{TS} = V_{TS} \text{ Offset} - 18\text{mV}$).

Example:

Temperature	(Ts+) - (Ts-)	ADC Vref	ADC Output(16bit)
15°C	$V_{TS} \text{ Offset} + 0.018\text{V}$	0.81V	ADC offset + 738
25°C	$V_{TS} \text{ Offset}$	0.81V	ADC offset
35°C	$V_{TS} \text{ Offset} - 0.018\text{V}$	0.81V	ADC offset - 738

By ADC output of V(TS), can get temperature information and compensation the system.

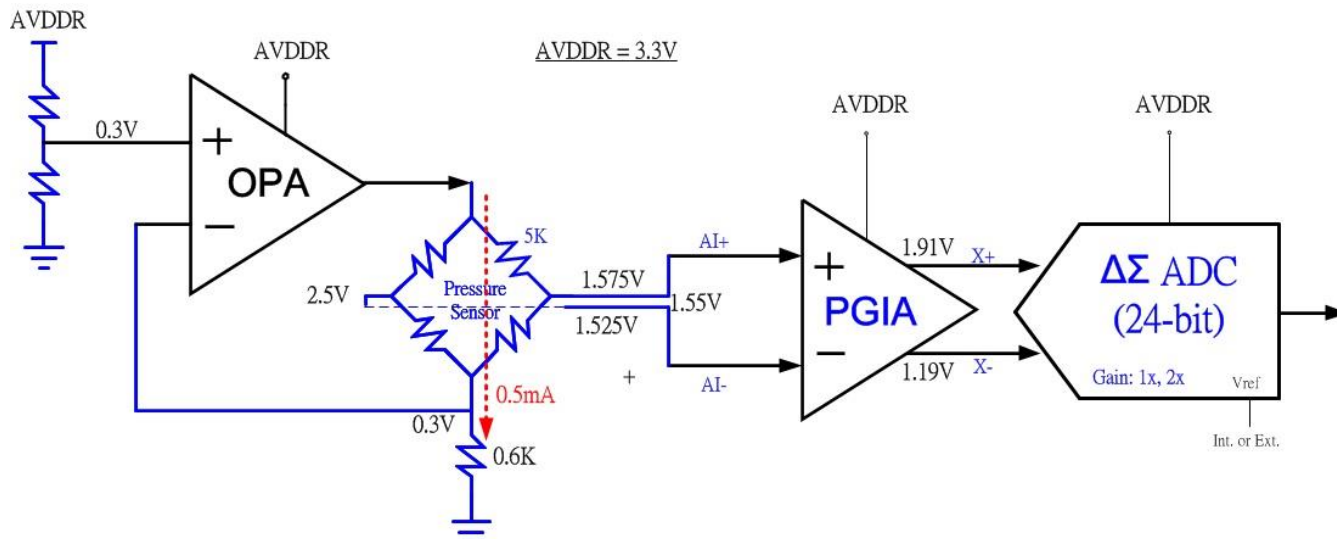
- * **Note 1:** The V(TS) voltage and temperature curve of each chip might different. Calibration in room temperature is necessary when application temperature sensor.
- * **Note 2:** -1.8mV/°C was typical temperature parameter only sensor, every single chip was different to each other. (-73.8 Cnt/°C)

SYM	DESCRIPTION	PARAMETER	MIN	TYP	MAX	UNIT
TR	Temperature Sensor Range	AVDDR=3V ,VDD=5V	-	-	-	°C
TS	Temperature Sensor Sensitivity	AVDDR=3V ,VDD=5V	-1.98	-1.8	-1.62	mV/°C
ETS	Temperature Sensor Accuracy	One Temperature point 25°C Calibration	-10	-	+10	%
		Two Temperature points Calibration.	-1	-	+1	%

17.11 ADC Application Notes

In applications, ADC design be attention to PGIA input signal(AI+,AI-) absolutely voltage and ADC Input signal (X+,X-)absolutely voltage. The PGIA input voltage must be between 0.4V~AVDDR-1V, the output signal must be between 0.4~AVDDR-1V. The ADC input signal (X+, X-) absolutely voltage range must be between 0~AVDDR-1V @GX Buff = OFF. The application circuit design as shown below.

*Example:



17.12 Sample Code

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

```

1
2 #include <intrins.h>      // for _nop_
3 #include <SN8F5909.h>
4
5 unsigned short ADCValue;
6
7 void Init_VREG(void);
8 void Init_PGIA(void);
9 void Init_ADC(void);
10 void ADC_Enable_ISR(void);
11 void ADC_ISR(void);
12
13 void main(void)
14 {
15     Init_VREG();           // Initial VREG
16     Init_PGIA();           // Initial PGIA
17     Init_ADC();            // Initial ADC
18     ADC_Enable_ISR();      // ADC ISR Enable
19
20     while (1) {
21         WDTR = 0x5A;       // clear watchdog if watchdog enable
22         // To Do ...
23     }
24 }
25
26
27 void Init_VREG(void)
28 {
29     VREG &= 0x00;          // Clear VREG
30     VREG |= 0x80;          // Enable Band gap
31     VREG &= 0x9F;          // AVDDR=2.7V
32
33     VREG |= 0x08;          // Enable ACM
34     VREG |= 0x10;          // Enable AVDDR
35 }
36
37
38 void Init_PGIA(void)
39 {
40     CHS &= 0x00;           // Clear CHS
41     CHS &= 0x0F;           // MUXP channel [AI1]
42     CHS |= 0x01;           // MUXP channel [AI2]
43
44     AMPM &= 0x00;          // Clear AMPM
45     AMPM |= 0x20;          // ADCLK=7.8k@ADCCLK=250K
46     AMPM |= 0x02;          // Enable PGIA Chopper
47     AMPM |= 0x18;          // PGIA x 128
48
49     AMPM |= 0x01;          // Enable PGIA
50 }
51
52
53
54

```

```

55
56 void Init_ADC(void)
57 {
58     ADCM1 &= 0x00;        // Clear ADCM1
59     ADCM1 |= 0x06;        // Enable ADC Chopper,
60                             // ACHPEN[1:0] always set 11
61     ADCM1 &= 0xF7;        // ADC Gain=1x
62     ADCM1 |= 0x50;        // ADC Vref = AVDDR*0.3
63
64     ADCM2 &= 0x00;        // Clear ADCM2
65     ADCM2 |= 0x10;        // ADC Clk=250kHz
66     ADCM2 |= 0x0E;        // ADC ODR=32768
67
68     ADCM2 &= 0xFE;        // Clear DRDY
69     ADCM1 |= 0x01;        // Enable ADC
70 }
71
72
73 void ADC_Enable_ISR(void)
74 {
75     ADCF = 0;             // Clear ADCF
76     IEN2 &= 0x00;         // Clear IEN2
77     IEN2 |= 0x10;         // ADC interrupt enable (EADC)
78     EAL = 1;             // Interrupt enable
79 }
80
81
82 void ADC_ISR(void) interrupt ISRAdc
83 {
84     // Get ADC value
85     if (ADCF) {
86         ADCF = 0;         // Clear ADC interrupt edge flag (ADCF)
87
88         /* Get16Bit ADC Data ( ADCDH,ADCDM ) */
89         ADCValue = ADCDH;
90         ADCValue = ADCValue<<8;
91         ADCValue = ADCValue+ADCDM;
92         _nop_();
93     }
94 }
95
96

```

The following sample code demonstrates how to perform ADC Temperature.

```

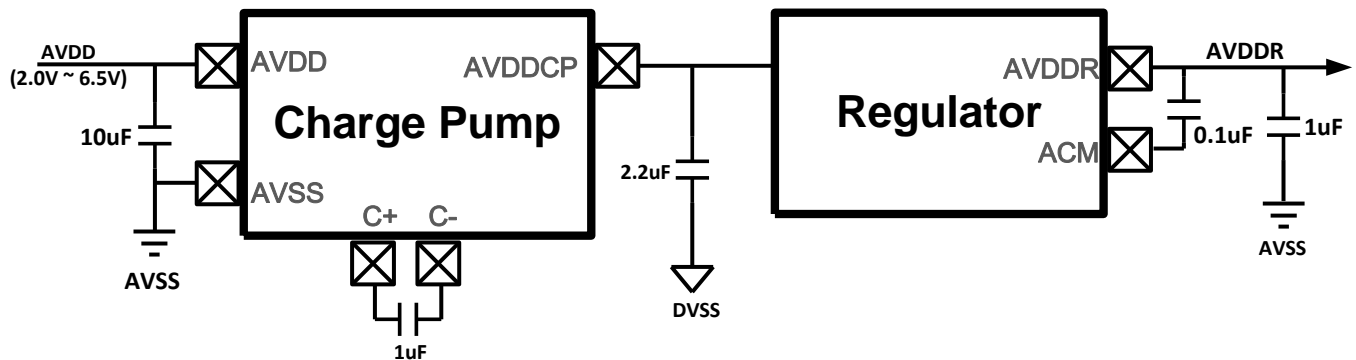
1  #include <intrins.h>      // for _nop_
2  #include <SN8F5909.h>
3
4  unsigned short ADCValue;
5
6  void Init_VREG(void);
7  void Init_PGIA(void);
8  void Init_ADC(void);
9
10 void main(void)
11 {
12     Init_VREG();           // Initial VREG
13     Init_PGIA();           // Initial PGIA
14     Init_ADC();            // Initial ADC
15
16     while (1) {
17         WDTR = 0x5A;       // clear watchdog if watchdog enable
18
19         // Get ADC value
20         if (ADCF) {
21             ADCF = 0;      // Clear ADC interrupt edge flag (ADCF)
22
23             /* Get16Bit ADC Data( ADCDH,ADCDM ) */
24             ADCValue = ADCDH;
25             ADCValue = ADCValue<<8;
26             ADCValue = ADCValue+ADCDM;
27             _nop_();
28         }
29     }
30 }
31
32 void Init_VREG(void)
33 {
34     VREG &= 0x00;          // Clear VREG
35     VREG |= 0x80;          // Enable Band gap
36     VREG &= 0x9F;          // AVDDR=2.7V
37
38     VREG |= 0x08;          // Enable ACM
39     VREG |= 0x10;          // Enable AVDDR
40 }
41
42 void Init_PGIA(void)
43 {
44     CHS &= 0x00;           // Clear CHS
45     CHS |= 0xB0;           // MUXP channel [Temp S+]
46     CHS |= 0x0B;           // MUXP channel [Temp S-]
47
48     AMPM &= 0x00;          // Clear AMPM
49     AMPM |= 0x20;          // ADCLK=7.8k@ADCCLK=250K
50     AMPM |= 0x02;          // Enable PGIA Chopper
51     AMPM &= 0xE3;          // PGIA x 1
52
53     AMPM |= 0x01;          // Enable PGIA
54 }

```

```
55 void Init_ADC(void)
56 {
57     ADCM1 &= 0x00;    // Clear ADCM1
58     ADCM1 |= 0x06;    // Enable ADC Chopper
59     ADCM1 &= 0xF7;    // ADC Gain=1x
60     ADCM1 |= 0x50;    // ADC Vref = AVDDR*0.3
61
62     ADCM2 &= 0x00;    // Clear ADCM2
63     ADCM2 |= 0x10;    // ADC Clk=250kHz
64     ADCM2 |= 0x0E;    // ADC ODR=32768
65
66     ADCM2 &= 0xFE;    // Clear DRDY
67     ADCM1 |= 0x01;    // Enable ADC
68 }
69
```

18 Charge Pump Regulator

The SN8F5900 IC has a built-in Voltage Charge-Pump Regulator (CPR) to support a stable analog voltage “AVDDR” for ADC, PGIA, OP, and external sensor use. Charge pump regulator is designed as efficient type of boost and buck mode, which apply in condition of wide VDD input range from 2.0V to 6.5V. AVDDR is adjustable output voltage 2.7V, 3.0V, 3.3V, and 3.6V with maximum 7.5mA current driving capacity. Another regulator ACM 1V is used as ADC common voltage, which is sink-type only.



18.1 Charge Pump Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREG	BGREN	AVDDRS1	AVDDRS0	AVDDREN	ACMEN	CPCKS	CPMOD1	CPMOD0

VREG Register (0xD2)

Bit	Field	Type	Initial	Description
7	BGREN	R/W	0	ADC Band Gap Enable bit. 0 : Disable. (must disable when system in stop mode) 1 : Enable. (must enable firstly when turn on AVDDR, ADC, PGIA, OPA, and LBT function)
6..5	AVDDRS[1:0]	R/W	00	AVDDR regulator output voltage control bits. 00 : 2.7V 10 : 3.3V 01 : 3.0V 11 : 3.6V
4	AVDDREN	R/W	0	AVDDR regulator function enable bit. 0 : Disable. 1 : Enable.

Note : AVDDR is source power of analog part.

3	ACMEN	R/W	0	ACM regulator function enable bit. 0 : Disable. 1 : Enable. (must enable when ADC turn on)
2	CPCKS	R/W	0	Charge pump clock control bit. 0 : 500KHz (please set 500KHz) 1 : 1MHz
1..0	CPMOD[1:0]	R/W	00	Charge pump mode control bits. 00 : Disable. 01 : Buck-Boost Mode. 10 : (reserved) 11 : 2x Pump Mode

The following table shows the pump working mode:

CPMOD[1:0]	Charge Pump Mode	Pump Behavior
00	Disable Mode	Pump Disable AVDDCP = VDD
01	Buck-Boost Mode	<u>VDD input range 2.0V ~ 6.5V</u> - Pump Enable. - AVDDCP = AVDDR + 0.5V - AVDDCP = 3.2V (when AVDDR disable)
10	Reserved	-
11	2x Pump Mode	<u>VDD input range 3.6V ~ 6.5V</u> - Pump Disable (when VDD > 3.6V must disable) - AVDDCP = VDD <u>VDD input range 2.0V ~ 3.4V</u> - Pump Enable - AVDDCP = 2x VDD (AVDDR enable or disable)
VDD : Pump input Voltage AVDDCP: Pump output voltage <u>Analog function turn on Sequence with Pump:</u> (1). BandGap Enable (2). Pump Enable (3). 1ms-delay for pump output stable (4). Enable AVDDR/ ACM (5). 0.5ms-Delay (6). Enable ADC, PGIA, OPA..		

* **Note : When battery voltage is less than AVDDR+0.3V, the pump mode can set buck-boost mode.**

18.2 Charge Pump Regulator Setting

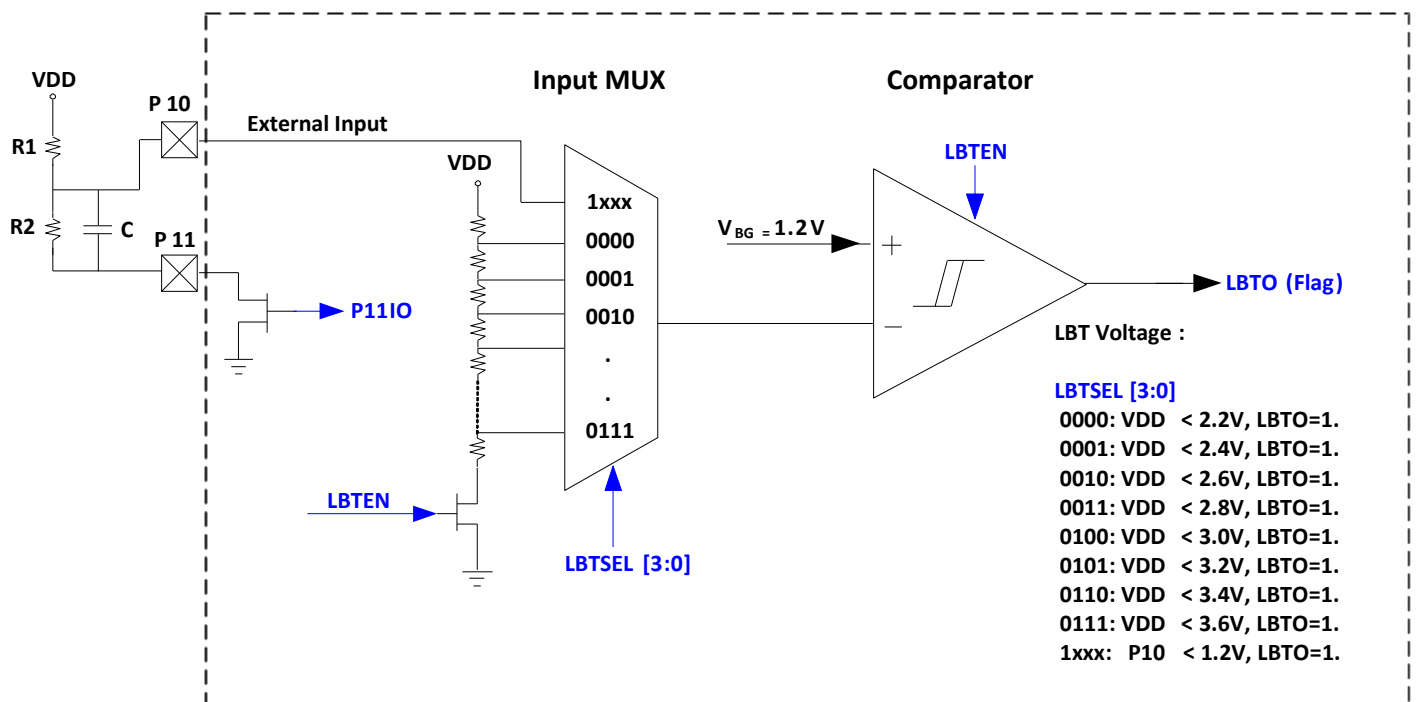
VDD Input Range	Charge Pump	AVDDCP	AVDDR	Note
6.5V ~ 4.0V	Disable	* No Capacitor C _{AVDDCP} & C _{Pump} * AVDDCP short to AVDD	* Connect 1uF to AVSS * 2.7V, 3V, 3.3V, 3.6V Available	No double current
6.5V ~ 3.3V	Disable		* Connect 1uF to AVSS * 2.7V, 3V Available	
6.5V ~ 3.0V	Disable		* Connect 1uF to AVSS * 2.7V Available	
6.5V ~ 2.0V	Enable	* Connect C _{AVDDCP} 2.2uF to DVSS (Connect C _{Pump} 1uF to C+/C-)	* Connect 1uF to AVSS * 2.7V, 3V, 3.3V, 3.6V Available	Current consumption from AVDDR will be double.
3.6V ~ 2.0V				

18.3 Charge Pump Regulator Characteristic

Charge Pump						
PARAMETER	SYM.	Condition	MIN.	TYP.	MAX.	UNIT
Operation Voltage (Charge Pump input)	V _{oper}	Charge Pump Input Voltage Range (AVDD)	2.0	-	6.5	V
Charge pump Output Range	V _{AVDDCP}	Charge Pump OFF, AVDD from 2V to 6.5V AVDDCP = AVDD	2.0	-	6.5	V
		Busk-boost Mode, AVDD from 2V to 6.5V AVDDCP = 3.2V ~ 4.1V (AVDDR + 0.5V)	AVDDR + 0.5V			V
		2x Pump Mode, AVDD from 2V ~ 3.3V AVDDCP = 4V ~ 6.6V	2x AVDD			V
		2x Pump Mode, AVDD from 3.3V ~ 6.5V AVDDCP = AVDD.	AVDD			V
Charge pump Output Current	I _{AVDDCP}	Charge Pump On or OFF, AVDD from 2V to 6.5V	-	5	7.5	mA
Charge pump intrinsic Current	I _{PUMP}	Busk-boost Mode, AVDD=3V		160		uA
ACM						
Analog common voltage	V _{ACM}	Output voltage	0.9	1	1.1	V
		Output Voltage drift vs. Temperature Δ10°C	-	±0.1	-	%
		Output Voltage drift vs. AVDD (2V~6.5V)	-	±0.1	-	%
VACM driving capacity	I _{SRC}	Only for ADC use	-	-	10	uA
VACM sinking capacity	I _{SNK}	Only for ADC use	-	-	1	mA
AVDDR						
Regulator output voltage AVDDR	V _{AVDDR}	AVDDR Output Range = 2.7V, 3.0V, 3.3V, 3.6V	2.7	-	3.6	V
		AVDDR set 2.7V	2.55	2.7	2.85	V
		AVDDR set 3.0V	2.85	3.0	3.15	V
		AVDDR set 3.3V	3.15	3.3	3.45	V
		AVDDR set 3.6V	3.45	3.6	3.75	V
		AVDDR Load regulation Δ5mA	V _{AVDDR} ± 0.05V			V
		Output Voltage drift vs. Temperature Δ10°C	-	±0.1	-	%
		Output Voltage drift vs. AVDD (2V~6.5V)	-	±0.1	-	%
AVDDR drive/Sink Current	I _{AVDDR}		-	-	7.5	mA
Quiescent current	I _{QI}	Pump + AVDDR + ACM		200		uA

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 1.2V. 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[2:0] register, from 2.1V to 3.5V with 0.2V 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	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTEN

LBTM Register (0xD7)

Bit	Field	Type	Initial	Description
6	P11IO	R/W	0	P1.1 Configuration bit. 0 : P1.1 set as GPIO function. 1 : Set P1.1 as LBT function, P1.1 connect to ground internally.

5	LBTSEL3	R/W	0	Comparator negative input source selection bit. 0 : P1.0 set as GPIO function 1 : External VDD voltage division.
4..2	LBTSEL[2:0]	R/W	010	Internal VDD voltage division control bits. 000 : VDD < 2.2V, LBTO=1. 001 : VDD < 2.4V, LBTO=1. 010 : VDD < 2.6V, LBTO=1. 011 : VDD < 2.8V, LBTO=1. 100 : VDD < 3.0V, LBTO=1. 101 : VDD < 3.2V, LBTO=1. 110 : VDD < 3.4V, LBTO=1. 111 : VDD < 3.6V, LBTO=1.
1	LBTO	R/W	0	Comparator Output flag. 0 : Comparator negative voltage great than 1.2V. 1 : Comparator negative voltage less than 1.2V.
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	2.04	2.15	2.26	V
		Condition: LBTSEL [3:0] = 0100, VLBT=3V	2.80	2.95	3.10	
		Condition: LBTSEL [3:0] = 0111, VLBT=3.6V	3.40	3.55	3.70	
V _{ELBT}	External Low-Battery detect voltage	Condition: LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.1	1.2	1.3	
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.

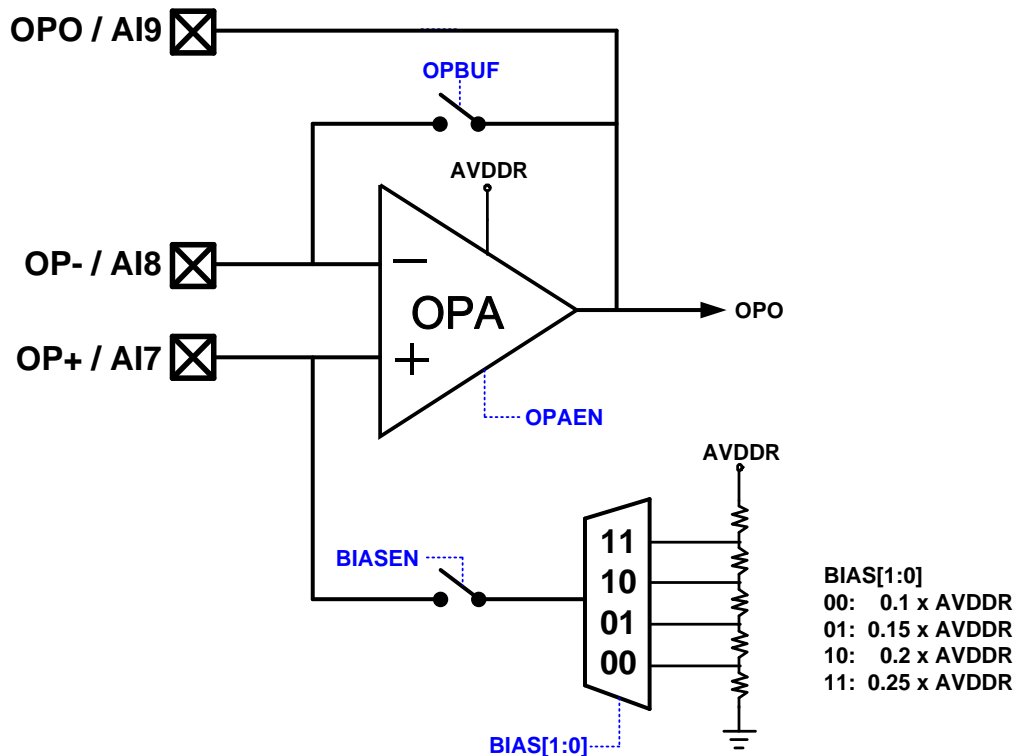
```

1
2 #include <intrins.h>      // for _nop_
3 #include <SN8F5909.h>
4
5 unsigned char LBTOValue;
6
7 void Init_VREG(void);
8 void Init_LBTM(void);
9
10 void main(void)
11 {
12     Init_VREG();          // Initial VREG
13     Init_LBTM();          // Initial LBT
14
15     while (1) {
16         LBTOValue = LBTM; //Get LBTO Flag
17         LBTOValue = LBTOValue>>1;
18         LBTOValue &= 0x01;
19
20         if(LBTOValue==1){
21             // To Do ...
22         }
23         Else{
24             // To Do ...
25         }
26     }
27 }
28
29
30 void Init_VREG(void)
31 {
32     VREG &= 0x00;          // Clear VREG
33     VREG |= 0x80;          // Enable Band gap
34     VREG &= 0x9F;          // AVDDR=2.7V
35
36     VREG |= 0x08;          // Enable ACM
37     VREG |= 0x10;          // Enable AVDDR
38 }
39
40 void Init_LBTM(void)
41 {
42     LBTM &= 0x00;          // Clear LBTM
43     /* Low Battery Detect Voltage as 2.6v */
44     LBTM |= 0x08;
45
46     LBTM |= 0x01;          // Comparator function enable
47 }
48
49

```

20 OPA

The microcontroller builds in one operational amplifier. The OP-Amp power source form AVDDR. OP-Amp input signal and output voltage are within range of 0V to AVDDR-0.1V. The OP-Amp input and output pin share with ADC input channel (AI7~AI9). OP_Amp can be configured as buffer mode when OPBUF=1. There are four internal bias-voltages can be shorted to positive end of OP-Amp through setting register BIAS[1:0] and BIASEN.



20.1 OP-Amp Control Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	BIAS1	BIAS0	BIASEN	OPBUF	OPAEN

OPM Register (0x9B)

Bit	Field	Type	Initial	Description
4..3	BIAS[1:0]	R/W	0	Bias output voltage control bits 00 : 0.1 x AVDDR. 01 : 0.15 x AVDDR. 10 : 0.2 x AVDDR 11 : 0.25 x AVDDR
2	BIASEN	R/W	0	Bias function enable bit. 0 : Disable. 1 : Enable, Vbias output connect to OP+.
1	OPBUF	R/W	0	OPA buffer mode control bit. 0 : Disable. 1 : Enable, OPO and OP- are shorted by MOS-SW.
0	OPAEN	R/W	0	OPA function enable bit. 0 : Disable. 1 : Enable.

***Example:** Operational amplifier & internal bias-voltages function Sample code.

```

1  VREG |= 0XB0; // Enable Band gap
2                // AVDDR=3.0V & Enable AVDDR
3  OPM |= 0x18;  // Set Bias 0.75V@AVDDR=3V.
4  OPM |= 0x04;  // Set Bias enable.
5
6  OPM |= 0X03;  // OP-Amp enables & OP-Buff enable

```

20.2 OP-Amp Electrical Characteristic

Operation Amplifier						
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Operation voltage	Voper	Power source from AVDDR	2.7	-	3.6	V
Input offset voltage	VOS			±3		mV
Operating Current	Ioper	Per OP-Amp	-	185	-	uA
Analog Input voltage range	VIN	OP+ / OP-	0.05		AVDDR-0.1	V
Analog Output voltage range	Vout	OPOUT	0.05		AVDDR-0.1	V
Output short circuit current	IOH/IOL	Unit Gain Buffer. Vo = 0V~3.2V, AVDDR=3.3V	-	±5	-	mA
Buffer Mode Switch Ron	RBUF	AVDDR 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	TON		-	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 and URX signal received via P0.5.

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

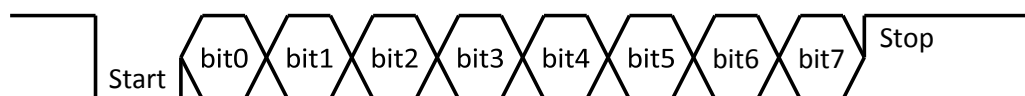
In mode 0, the UART transmits/receives an 8-bit-length data without start and stop bits. It handles the first bit of the bus as LSB (least significant bit), and its baud rate is fixed at $f_{cpu}/12$. The reception is started by setting RENO and clearing RIO bits, whereas the transmission is started 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 SORELH/SORELL 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 TIO bit is automatically set as a notification/interrupt.

The UART synchronizes with the start bit from master if the reception is enabled (RENO = 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, $f_{cpu}/32$ and $f_{cpu}/64$, but the mode 3 can control its baud rate by SORELH/SORELL 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 $f_{cpu}/12$, and the mode 2 has two baud rate selection which is chosen by SMOD register: $f_{cpu}/32$ (SMOD = 0) and $f_{cpu}/64$ (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) in mode 1 and 3, the baud rate is generated as following equation.

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{f_{cpu}}{64 \times (1024 - \text{S0REL})}$$

Table 19-1 Recommended Setting for Common UART Baud Rates ($f_{cpu} = 32 \text{ MHz}$)

Baud Rate	SMOD	S0RELH	S0RELL	Accuracy
4800 Hz	1	0x03	0x30	-0.16 %
9600 Hz	1	0x03	0x98	-0.16 %
19200 Hz	1	0x03	0xCC	-0.16 %
38400 Hz	1	0x03	0xE6	-0.16 %
56000 Hz	1	0x03	0xEE	0.79 %
57600 Hz	1	0x03	0xEF	-2.12 %
115200 Hz	1	0x03	0xF7	3.55 %
128 kHz	1	0x03	0xF8	2.34 %
250 kHz	1	0x03	0xFC	0 %
500 kHz	1	0x03	0xFE	0 %
1 MHz	1	0x03	0xFF	0 %

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.

$$\text{Baud Rate} = 2^{\text{SMOD}} \times \frac{1}{32 \times \text{Timer period}}$$

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

Baud Rate	SMOD	Timer Priod	TC2CH/ L	TC2RH/ L	Accuracy
4800 Hz	1	13.021 us	0xFE5F	0xFE5F	-0.08 %
9600 Hz	1	6.510 us	0xFF30	0xFF30	0.16 %
19200 Hz	1	3.255 us	0xFF98	0xFF98	0.16 %
38400 Hz	1	1.628 us	0xFFCC	0xFFCC	0.16 %
56000 Hz	1	1.116 us	0xFFDC	0xFFDC	-0.80 %
57600 Hz	1	1.085 us	0xFFDD	0xFFDD	-0.80 %
115200 Hz	1	0.543 us	0xFFEF	0xFFEF	2.08 %
128 kHz	1	0.488 us	0xFFFF0	0xFFFF0	-2.40 %
250 kHz	1	0.250 us	0xFFFF8	0xFFFF8	0 %
500 kHz	1	0.125 us	0xFFFFC	0xFFFFC	0 %

*** Note: System clock Fcpu minimum setting value follow Fcpu < Timer Priod Value. @ PWSC[2:0]=001**
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 = 32M Hz)

```

1  TC2CH = 0x0FF;      // baud rate 115200
2  TC2CL = 0x0EF;      // 0.543us Overflow
3  TC2RH = 0x0FF;      // auto-reload value
4  TC2RL = 0x0EF;      //
5  TC2M |= 0x0F0;      // TC2 ENABLE, TC2 timer clock= Fcpu/1
6
7  S0CON2 = 0x00;      // baud rate = TC2 Timer overflow period
8  PCON |= 0x80;       // SMOD=1, Fcpu*2 in model/mode3;
9  S0CON |= 0x50;      // UART Enable, UART Mode 1
10
```

21.5 UART Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SOCON	SM0	SM1	SM20	REN0	TB80	RB80	TIO	RIO
SOCON2	BD	-	-	-	-	-	-	-
S0BUF	S0BUF7	S0BUF6	S0BUF5	S0BUF4	S0BUF3	S0BUF2	S0BUF1	S0BUF0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
SORELH	-	-	-	-	-	-	SOREL9	SOREL8
SORELL	SOREL7	SOREL6	SOREL5	SOREL4	SOREL3	SOREL2	SOREL1	ROREL0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
P1OC	-	-	-	P05OC	P04OC	P14OC	P13OC	P12OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P0	P07	P06	P05	P04	P03	P02	P01	P00

SOCON 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 reception function 0: Disable 1: Enable
3	TB0	R/W	0	The 9 th bit transmission data (mode 2, 3)
2	RB0	R/W	0	The 9 th bit data from reception
1	TIO	R/W	0	UART interrupt flag of transmission
0	RIO	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 SORELH, SORELL registers
6..0	Reserved	R	0x00	

S0BUF Register (0x99)

Bit	Field	Type	Initial	Description
7..0	S0BUF	R/W	0x00	Action of writing data triggers UART 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)

SORELH/SORELL Registers (SORELH: 0xBA, SORELL: 0xAA)

Bit	Field	Type	Initial	Description
15..10	Reserved	R	0x00	
9..0	SOREL[9:0]	R/W	0x00	SORELH[1:0] & SORELL[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	ESO	R/W	0	Enable UART interrupt
Else				Refer to other chapter(s)

P1OC Register (0xE4)

Bit	Field	Type	Initial	Description
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 as high causes URX cannot received data.

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
5	P05M	R/W	0	0: Set P0.5 (URX) as input mode (required) 1: Set P0.5 (URX) as output mode*
4	P04M	R/W	0	0: Set P0.4 (UTX) as input mode* 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: Set P0.4 (UTX) always low* 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.

21.6 Sample Code

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

```
1 #define SYSUartSM0 (0 << 6)
2 #define SYSUartSM1 (1 << 6)
3 #define SYSUartSM2 (2 << 6)
4 #define SYSUartSM3 (3 << 6)
5 #define SYSUartREN (1 << 4)
6 #define SYSUartSMOD (1 << 7)
7 #define SYSUartES0 (1 << 4)
8
9 void SYSUartInit(void)
10 {
11     // set UTX, URX pins' mode at here or at GPIO initialization
12     P04 = 1;
13     P05 = 0;
14     // set P04 Output Mode and P05 Input Mode
15     P0M = (P0M | 0x10) & ~0x20;
16
17     // configure UART mode between SM0 and SM3, enable URX
18     S0CON = SYSUartSM1 | SYSUartREN;
19
20     // configure UART baud rate
21     PCON = SYSUartSMODE1;
22     S0CON2 = 0x80;
23     S0RELH = 0x03;
24     S0RELL = 0xFE;
25
26     // enable UART interrupt
27     IEN0 |= SYSUartES0;
28
29     // global interrupt enable
30     EAL = 1;
31
32     // send first UTX data
33     S0BUF = uartTxBuf;
34 }
35
36 void SYSUartInterrupt(void) interrupt ISRUart
37 {
38     if (TI0 == 1) {
39         S0BUF = uartTxBuf;
40         TI0 = 0;
41     } else if (RI0 == 1)
42     { uartRxBuf = S0BUF;
43       RI0 = 0;
44     }
45 }
```

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 P1.4) 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 P1.2) plays the role of data output, and MISO pin (shared with P1.3) 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 P1.1) 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.

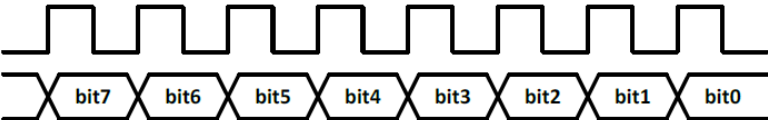
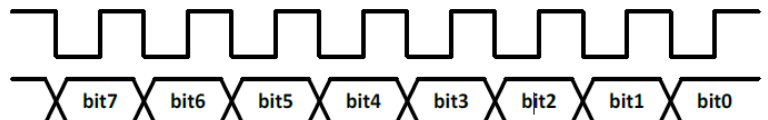
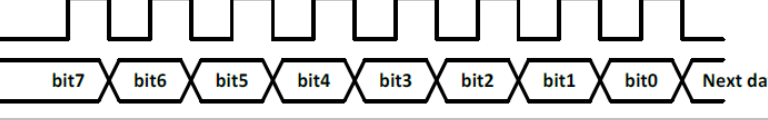
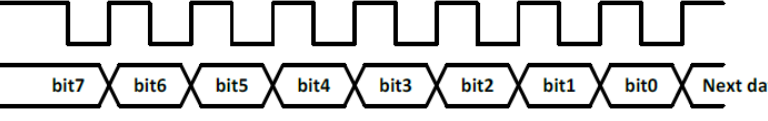
The slave mode has two status flags with interrupt function:

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.

C P O L	C P H A	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		SCK idle high Data latch at falling edges

22.4 SPI 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	-	-	ES0	-	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
P1OC	-	-	-	P05OC	P04OC	P14OC	P13OC	P12OC
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 clockedge 1: Data latched by the second of clockedge

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	P14OC	R/W	0	0: Switch P1.4 (SCK) to input or output mode 1: Switch P1.4 (SCK) to open-drain mod
1	P13OC	R/W	0	0: Switch P1.3 (MISO) to input or output mode 1: Switch P1.3 (MISO) to open-drain mod
0	P12OC	R/W	0	0: Switch P1.2 (MOSI) to input or output mode 1: Switch P1.2 (MOSI) to open-drain mode
Else				Refer to other chapter(s)

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
7	P11M	R/W	0	0: Set P1.1 (SSN) as input mode* 1: Set P1.1 (SSN) as output mode*
Else				Refer to other chapter(s)

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

P1M Register

Bit	Field	Type	Initial	Description
4	P14M	R/W	0	0: Set P1.4 (SCK) as input mode ^{slave mode} 1: Set P1.4 (SCK) as output mode ^{master mode}
3	P13M	R/W	0	0: Set P1.3 (MISO) as input mode ^{master mode} 1: Set P1.3 (MISO) as output mode ^{slave mode}
2	P12M	R/W	0	0: Set P1.2 (MOSI) as input mode ^{slave mode} 1: Set P1.2 (MOSI) as output mode ^{master mode}
Else				Refer to other chapter(s)

¹Setting SCK as input mode is essential in slave mode; setting as output mode is recommended in master mode.

²Setting MISO as input mode is essential in master mode; setting as output mode is recommended in slave mode.

³Setting MOSI as input mode is essential in slave mode; setting as output mode is recommended in master mode.

22.5 Sample Code

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

```

1 #define SpiMaster (1 << 4)           //SPI = Master mode
2 #define SpiSlave (1 << 4)           //SPI = Slave mode
3 #define SpiMode0 (0 << 2)           //SCK idle low, data latch at rising edge
4 #define SpiMode1 (1 << 2)           //SCK idle low, data latch at falling edge
5 #define SpiMode2 (2 << 2)           //SCK idle high, data latch at falling edge
6 #define SpiMode3 (3 << 2)           //SCK idle high, data latch at rising edge
7 #define SpiEn (1 << 6)              //Enable SPI
8 #define SpiSSNEn (0 << 5)           //SSN pin function enable
9 #define SpiSSNDis (1 << 5)          //SSN pin function disable
10
11
12 unsigned char u8SpiData = 0;         // data buffer
13 unsigned char u8TxCompleted = 0;
14
15
16 void SpiMaster(void)
17 {
18     //SCK & MOSI = output, MISO = input
19     P1M |= 0x14;
20     //Enable Spi, Master mode, SSN pin disable, Fclk/128
21     //SCK idle low, data latch at falling edge
22     SPCON = SpiEn | SpiMaster | SpiMode1 | SpiSSNDis | 0x82;
23     //Enable Global/SPI interrupt
24     while (1) {
25         SPDAT = 0x55;
26         while(!u8TxCompleted);        // wait end of transmition
27         u8TxCompleted = 0;             // clear sw flag
28         u8RcvData = u8SpiData;         // receive 0x66
29         SPDAT = 0x99;
30         while(!u8TxCompleted);        // wait end of transmition
31         u8TxCompleted = 0;             // clear sw flag
32         u8RcvData = u8SpiData;         // receive 0xAA
33     }
34 }
35
36
37 void SpiInterrupt(void) interrupt ISRSpi //0xA3
38 {
39     switch ( SPSTA )                  // Clear SPI flag (SPIF) by reading
40     {
41         case 0x80:
42             u8SpiData = SPDAT;
43             u8TxCompleted = 1;
44             break;
45
46         case 0x10:
47             // Mode Fault
48             break;
49     }
50 }
51
52
53
54

```

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).

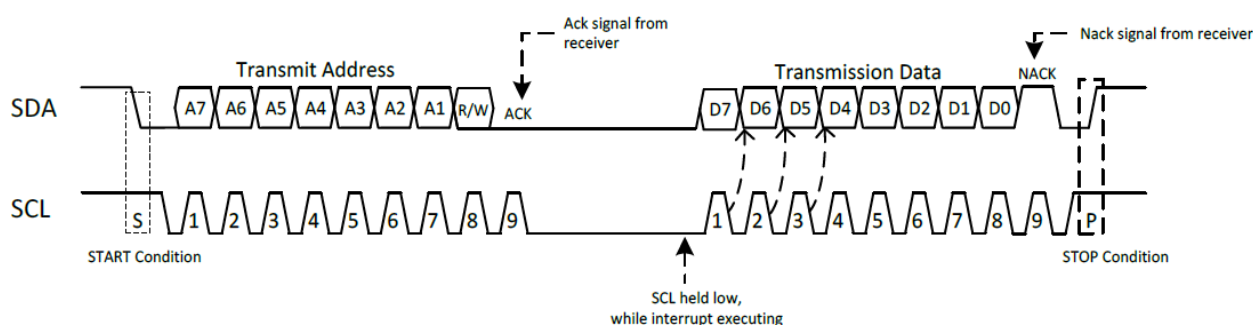
When a master transmit data to a slave, it's called "WRITE" operation; when a slave transmit data to a master, it's called "READ" operation. It also supports multi-master communication and keeps data transmission correctly by an arbitration method to decide one master has the control on bus and transmit 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 "WRITER" 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 I2CADDR 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.

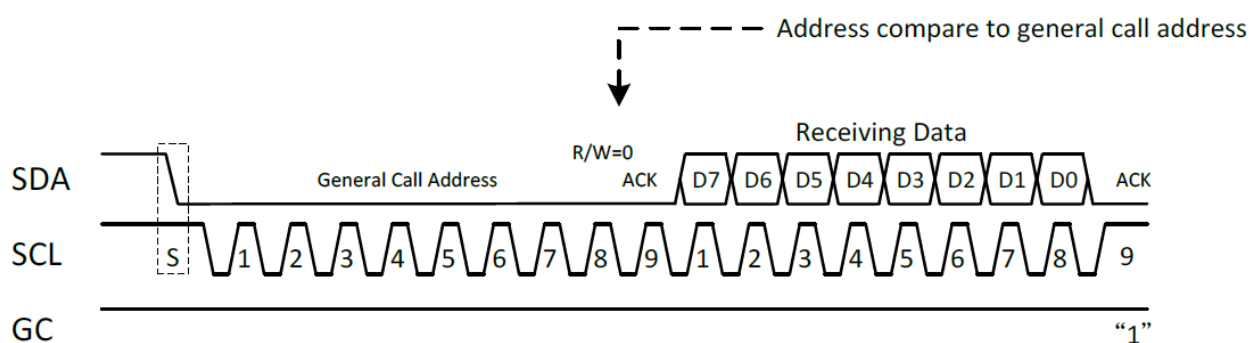
23.6 Slave Receiver Mode

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 I2CADDR 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.

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

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	4MHz	8MHz	16MHz	32MHz
0	0	0	256	15.6	31.3	62.5	125
0	0	1	224	17.9	35.7	71.4	142.9
0	1	0	192	20.8	41.7	83.3	166.7
0	1	1	160	25	50	100	200
1	0	0	960	4.2	83.3	16.7	33.3
1	0	1	120	33.3	66.7	133.3	266.7
1	1	0	60	66.7	133.3	266.7	533.3
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 = 32M Hz)

```

1  TC2CH = 0xFF;      // clock rate 400k Hz = 1/0.3125us/8
2  TC2CL = 0xF6;      // 0.3125us Overflow
3  TC2RH = 0xFF;      // auto-reload value
4  TC2RL = 0xF6;      //
5  TC2M |= 0xF0;      // TC2 ENABLE, TC2 timer clock=Fcpu/1
6
7  I2CCON |= 0x83;     // SPR[2:0]:111,SCL source from TC2 Timer
8  I2CCON |= 0x40;     // I2C enable (ENSL)
9

```

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 another 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 another 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 another 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 master 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) Tout 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.

$$T_{mext}/T_{sext}/T_{out} = \frac{\text{Timer Period(sec)} * F_{CPU}(\text{Hz})}{1024}$$

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	Descriptio
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	Tsxt timeout error.
XXXX X1XX	Tmxt 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: reserved
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 busin
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 hasreceived 1: An ACK will be returned when a byte hasreceived

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
	88H	Previously addressed with own SLA; DATA byte 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

Slave Transmitter							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	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
	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.

```

1 unsigned int I2CAddr;
2 unsigned int I2C_TXData0;
3 unsigned int I2C_TXDatan;
4 unsigned int I2C_RXData0;
5 unsigned int I2C_RXDatan;
6
7
8 void I2C_Init(void)
9 {
10     P02 = 0;
11     P03 = 0;
12     P0M |= 0x00;           // P02 & P03 as input
13
14     I2CCON |= 0x40;         // I2C enable (ENS1)
15     I2CCON |= 0x82;         // Clock rate bit : 110b ; 533.3KHz at 32MHz
16
17     EI2C = 1;              // I2C interrupt enable
18     EAL = 1;              // Interrupt enable
19 }
20
21
22 void I2C_ISR(void) interrupt ISRI2c    // Vector @ 0xAB
23 {
24     switch (I2CSTA)
25     {
26         // tx mode
27         case 0x08:
28             I2CCON &= 0xDF;    // START (STA) = 0
29             I2CDAT = I2CAddr;  // Tx/Rx addr
30             break;
31
32         case 0x18:             // write first byte
33             I2CDAT = I2C_TXData0;
34             break;
35
36         case 0x28:             // write n byte
37             I2CDAT = I2C_TXDatan;
38             break;
39
40         case 0x30:             // STOP (STO)
41             I2CCON |= 0x10;
42             break;
43
44         // rx mode
45         case 0x40:             // get slave addr
46             I2CCON |= 0x04;    // AA = 1
47             break;
48
49         case 0x50:             // read n byte
50             I2C_RXData0 = I2CDAT;
51             I2CCON &= 0xFB;    // AA = 0
52             break;
53
54     }

```

```
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

SN8F5900 builds in an on-chip 64 KB program memory, aka IROM, which is equally divided to 1024 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.

0xFFFF	Page 1023
0xFFC0	
0xFFBF	
0xFF80	Page 1022
	...
0x00BF	Page 2
0x0080	
0x007F	
0x0040	Page 1
0x003F	
0x0000	

Program memory (IROM)

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 1022th page of program memory (IROM, 0xFF80 – 0xFFBF) 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 0xFF80 to EPROMH/EPROML registers, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0xA5A' into PECMD [11:0] registers to duplicate the buffer's data to 1022th page of IROM.

In general, every page has the capability to be modified by in-system program procedure. However, since the first and least pages (page 0 and 1023) respectively stores reset vector and information for power-on controller, incorrectly perform page program (such as turn off power while programming) may cause faulty power-on sequence /reset.

24.2 Byte Program

SN8F5900 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 1022th page of program memory (IROM, 0xFF82) 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 0xFF80 to EPROMH/EPROML registers, writer byte address 0x02 to PEBYTE register, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0xA1E' into PECMD [11:0] registers to duplicate the buffer's data to 3rd byte of 1022th 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	-	-	CMD11	CMD10	CMD9	CMD8
PECMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PEBYTE			PEBYTE5	PEBYTE4	PEBYTE	PEBYTE2	PEBYTE1	PEBYTE0

PECMD Register (0x94)

Bit	Field	Type	Initial	Description
7..0	PECMD[7:0]	W	-	0x5A : Start page-program procedure 0x1E : Start byte-program procedure.

* 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..4	Reserved	R	0	
3..0	PECMD[11:8]	W	-	0xA: Enable in-system program Else values: Disable in-system program*

* Disabling in-system program can avoid mistakenly trigger ISP function.

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)

PEBYTE Register (0x9C)

Bit	Field	Type	Initial	Description
5..0	PEBYTE[5:0]	R/W	0x00	The Byte address of a page.

24.4 Sample Code

The following sample code demonstrates how to perform ISP.

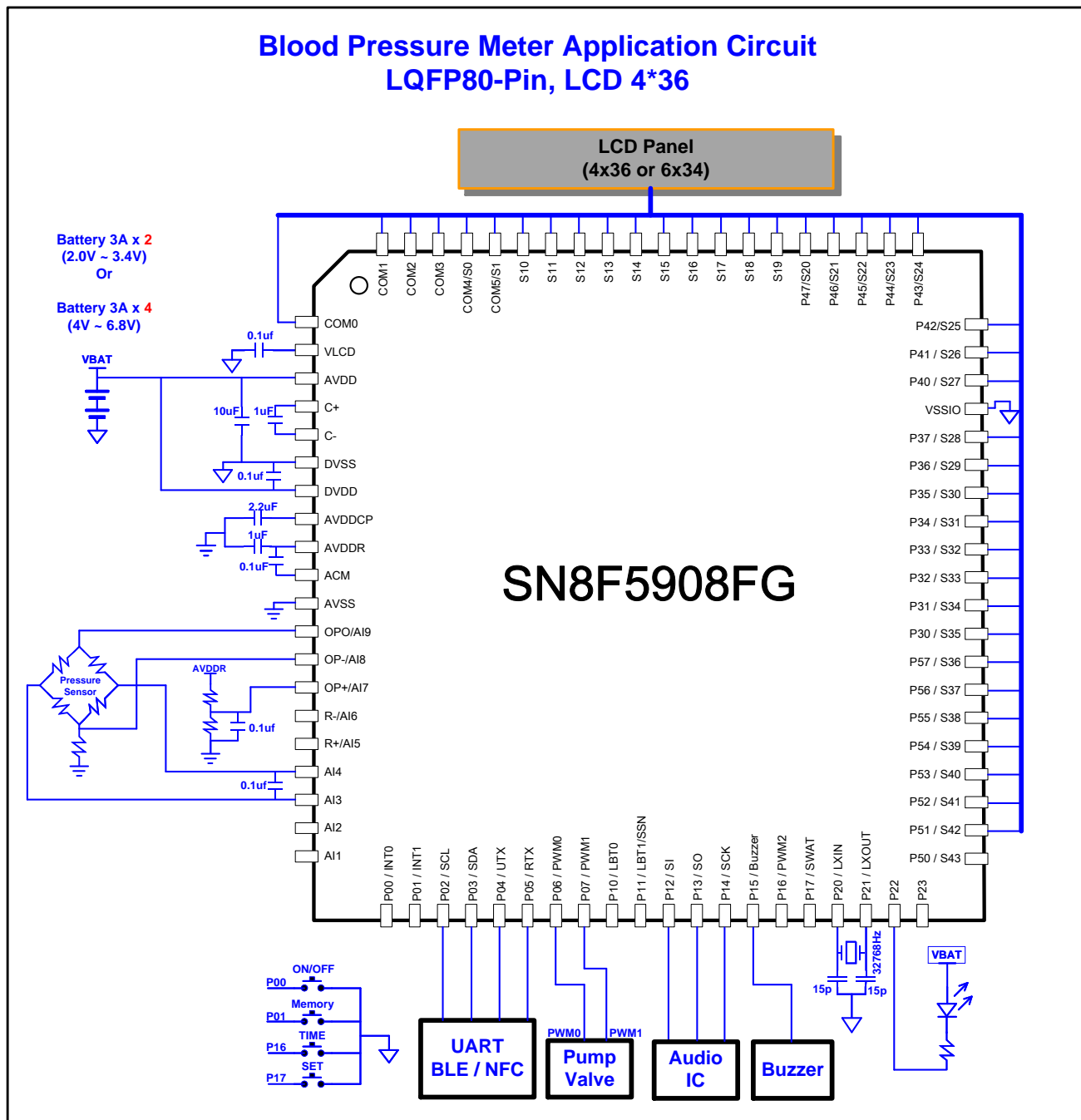
```
1  #include <intrins.h>      // for _nop_
2  #include <SN8F5909.h>
3
4  #include "GenericTypeDefs.h"
5
6  void ISPSetROMAddr(UINT16 u16addr);
7  void ISPSetRAMAddr(UINT8 u8addr);
8  void ISPWritePage(void);
9  void ISPWriteByte(void);
10 void ISPEecute_PAGE(void);
11 void ISPEecute_Byte(void);
12
13 void main(void)
14 {
15     WDTR = 0x5A;           // clear watchdog if watchdog enable
16
17     ISPEecute_PAGE();      // Program one page by ISP
18
19     ISPEecute_Byte();      // Program one Byte by ISP
20
21     while (1) {
22         WDTR = 0x5A;       // clear watchdog if watchdog enable
23
24         // To Do ...
25     }
26 }
27 void ISPEecute_PAGE(void)
28 {
29     UINT8 idata u8data[64] = {0};
30
31     UINT8 idata i = 0;
32
33     // step 1 : Get data
34     for (i = 1; i < 65; i++)
35         u8data[i-1] = i;    // write data for test
36
37     // step 2 : Set RAM addr of data
38     i = u8data;             // get start addr
39
40     ISPSetRAMAddr(i);
41     // step 3 : Set ROM start addr (Range is 0x0000~0xFFFF)
42     ISPSetROMAddr(0x8880);
43
44     // step 4 : Program one page (64 bytes)
45     ISPWritePage();
46
47     //erase all USER ROM
48     //ISPEraseAllROM();
49 }
50
51
52
53
54
```

```
55
56 void ISPExecute_Byte(void)
57 {
58     UINT8 idata u8data_byte[1] = {0};
59     UINT8 idata j =0;
60
61     // step 1 : Get data
62     u8data_byte[0] = 0XAA;    // write data for test
63     // step 2 : Set RAM addr of data
64     j = u8data_byte;          // get start addr
65     ISPSetRAMAddr(j);
66     // step 3 : Set ROM start addr (Range is 0x0000~0xFFFF)
67     ISPSetROMAddr(0x8880);
68
69     // step 4 : Set ROM Byte addr(Range is 0x00~0x3F)
70     PEBYTE = 0X20;
71
72     // step 5 : Progarm one byte (1 byte)
73     ISPWriteByte();
74
75     //erase all USER ROM
76     //ISPEraseAllROM();
77 }
78
79 void ISPSetROMAddr(UINT16 u16addr)
80 {
81     // set ROM addr
82     PEROML &= 0x0F;
83     //ISP Target Start ROM address Low byte
84     PEROML |= ((u16addr & 0x00F0));
85     //ISP Target Start ROM address High byte
86     PEROMH |= ((u16addr & 0xFF00)>>8);
87     _nop_();
88
89 }
90
91 void ISPSetRAMAddr(UINT8 u8addr)
92 {
93     // set RAM addr
94     PERAM = u8addr;
95 }
96
97 /*void ISPEraseAllROM(void)
98 {
99     // erase whole USER ROM
100     PECMD = 0x96;
101     PEROML |= 0x0A;
102 }*/
103
104
105
106
107
108
109
```

```
110
111 void ISPWritePage(void)
112 {
113     // execute Page Erase and Write
114     PEROML |= 0x0A;
115
116     if(EAL == 0) {
117         PECMD = 0x5A;
118         _nop_(); _nop_();
119     }
120     else {
121         EAL = 0;
122         PECMD = 0x5A;
123         _nop_(); _nop_();
124         EAL = 1;
125     }
126 }
127 void ISPWriteByte(void)
128 {
129     // execute Page Erase and Write
130     PEROML |= 0x0A;
131
132     if(EAL == 0) {
133         PECMD = 0x1E;
134         _nop_(); _nop_();
135     }
136     else {
137         EAL = 0;
138         PECMD = 0x1E;
139         _nop_(); _nop_();
140         EAL = 1;
141     }
142 }
```

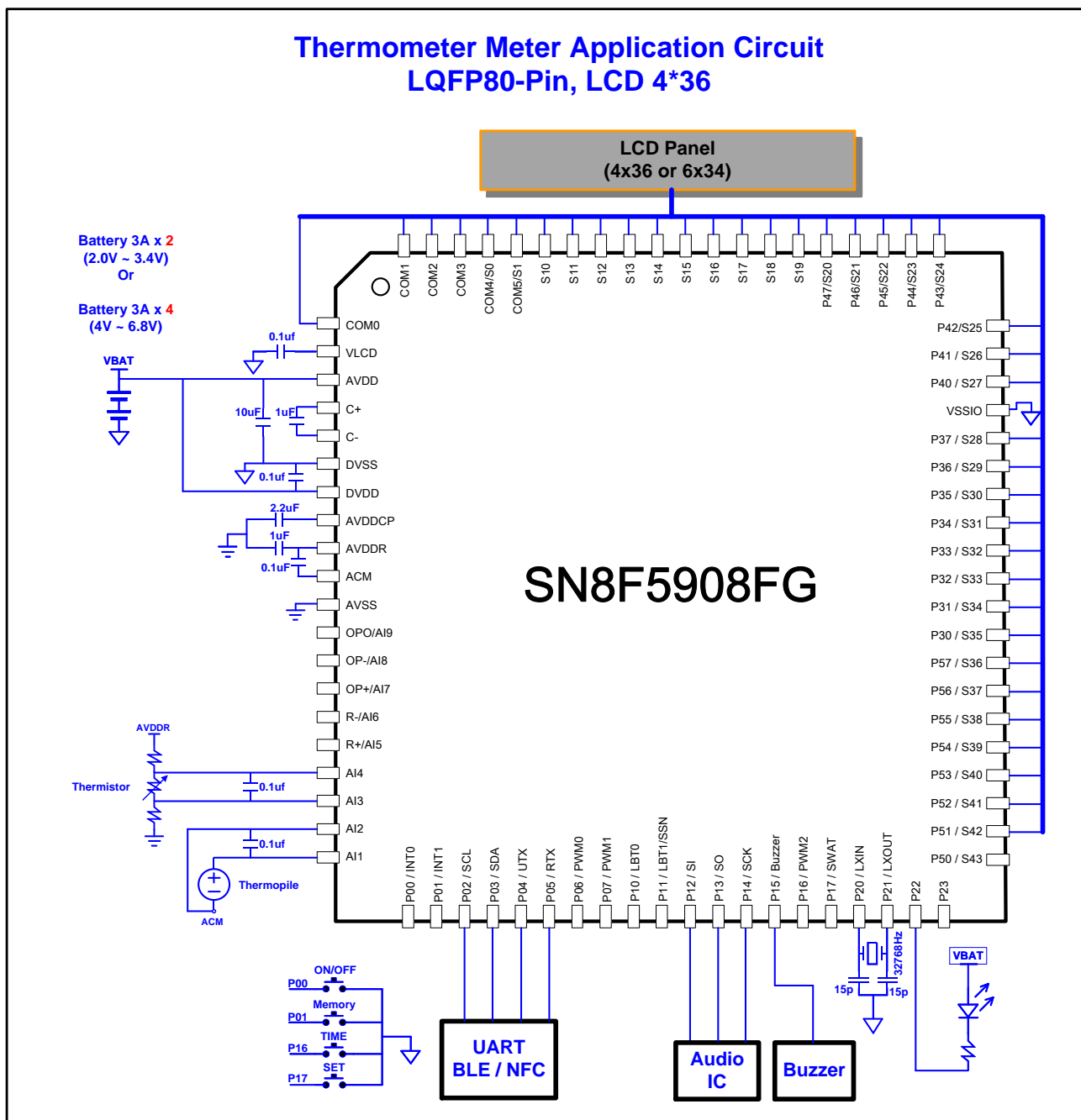
25 APPLICATION CIRCUIT

25.1 BP application circuit



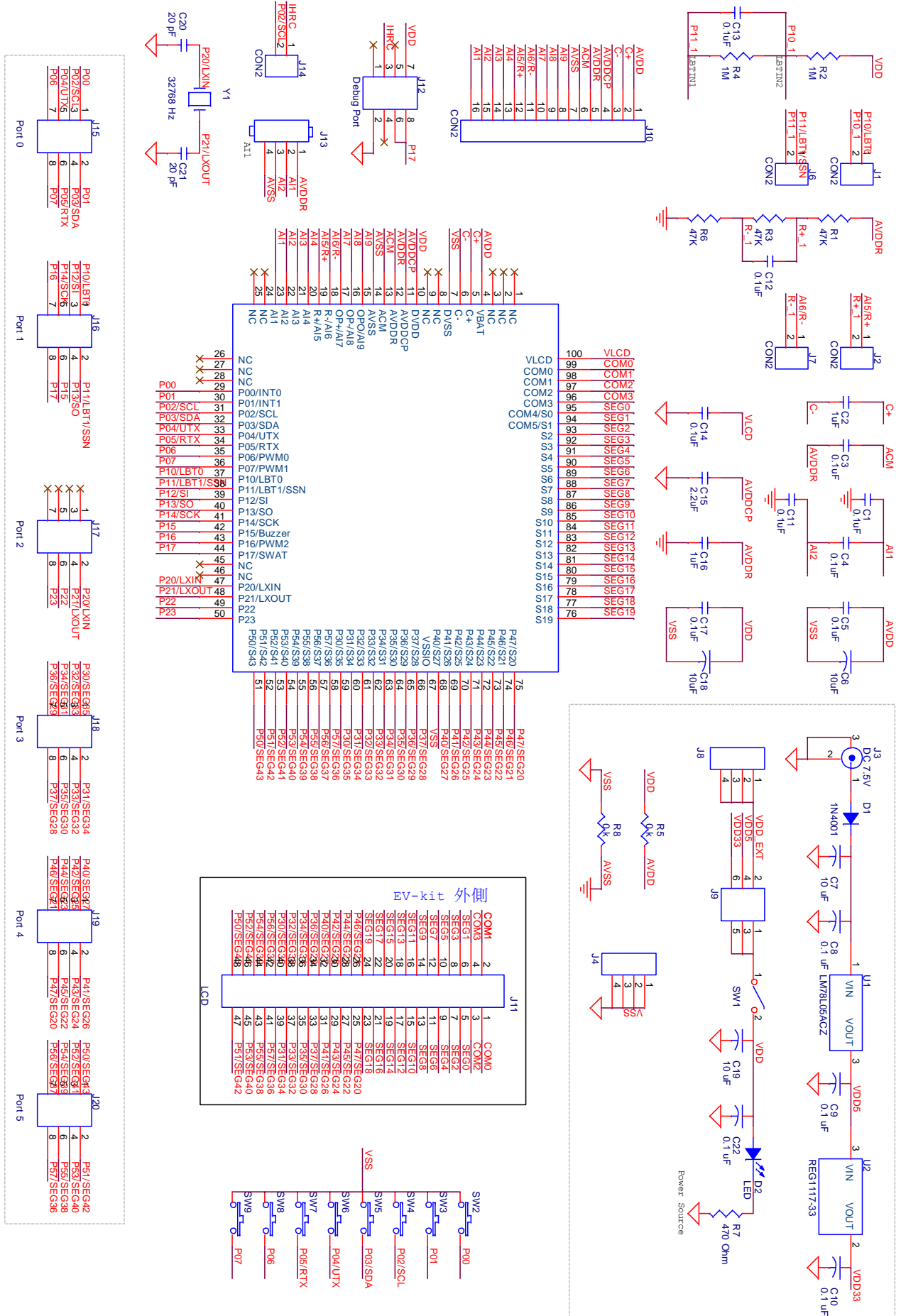
★ **Note:** DVDD/AVDD capacitors should be as close as possible with pins of IC

25.2 Thermometer application circuit



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

25.3 EV-KIT BOARD CIRCUIT



26 Electrical Characteristics

26.1 Absolute Maximum Ratings

Voltage applied at VDD to VSS	- 0.3V to 6.5V
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

26.2 System Operation Characteristics

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	Operating voltage	fcpu = 1MHz	2.0		6.5	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 = 3V ~ 5V, fcpu = 0.25MHz		2.8		mA
		VDD = 3V ~ 5V, fcpu = 1MHz		3.0		mA
		VDD = 3V ~ 5V, fcpu = 2MHz		3.3		mA
		VDD = 3V ~ 5V, fcpu = 4MHz		3.6		mA
		VDD = 3V ~ 5V, fcpu = 8MHz		4.3		mA
		VDD = 3V ~ 5V, fcpu = 16MHz		5.7		mA
		VDD = 3V ~ 5V, fcpu = 32MHz		8.4		mA
I _{DD2}	STOP mode supply current	VDD = 3V		3.5		μA
		VDD = 5V		4.0	6.0	μA
		VDD = 3V, RTC enable (32768Hz Crystal)		5.0		μA
I _{DD3}	IDLE mode supply current (fcpu = 0.25MHz)	VDD = 3V		0.56		mA
		VDD = 5V		0.57		mA
f _{IHRC}	Internal high clock generator	VDD = 2.0V ~ 6.5V, Temp. 25°C	-1%	32	+1%	MHz
		VDD = 2.0V to 6.5V, Temp. -40°C to 85°C	-2%	32	+2%	MHz
V _{LVD18}	LVD16 detect voltage	25°C	1.58	1.65	1.72	V
		-40°C to 85°C	1.60	1.65	1.80	V
V _{LVD15}	LVD15 detect voltage	25°C	1.40	1.45	1.52	V
		-40°C to 85°C	1.30	1.45	1.60	V
ESD	Human body mode (HBM)			2	2.5	KV
	Machine mode (MM)			200	300	V
LU	Latch Up	VDD < 6V	-400		+400	mA
		6V < VDD < 6.5V	-50		+400	mA
		6V < VDD < 6.5V, add 10Ω resistor in DVDD path.	-400		+400	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.

26.3 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	50	100	150	kΩ
		VDD = 5V	25	50	75	kΩ
I _{OH}	I/O output source current	VDD = 3V, V _O = VDD-0.5V	8	10		mA
I _{OL}	I/O sink current	VDD = 3V, V _O = VSS+0.5V	9	10		mA

26.4 ADC Characteristics

SYM.	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
I _{ADC}	Operating current	Run mode @ 2.4V	-	200	-	uA
I _{PDN}	Power down current	Stop mode @ 2.4V	-	0.1	-	μA
F _{SMP}	Conversion rate (WR)	ADC Clock=62.5KHz, OSR=32768	-	1.9	-	Hz
		ADC Clock=500KHz, OSR=64	-	7.8	-	kHz
T _{ADCSTL}	ADC settling Time	3*(1/WR), if WR=61Hz, T _{ADCSTL} = 3*16.4ms = 49.2ms		3		WR
V _{REF}	Reference Voltage Input Voltage	External V _{REF} Input Range (R+ - R-)	0.3		1.35	V
		Internal V _{REF} Input Range.	0.3		1.35	V
V _R	ADC Reference signal absolutely voltage	GR=1, R+ and R- absolutely input Voltage	0.4		AVDDR-1V	V
		GR=0, R+ and R- absolutely input Voltage	0.4		AVDDR-1V	V
V _{AI}	ADC Input signal absolutely voltage	GX=1, AI absolutely input Voltage	0.3		AVDDR-1V	V
		GX=0, AI absolutely input Voltage	0		AVDDR-1V	V
V _x	PGIA output signal absolutely voltage	X+ and X- absolutely output Voltage	0.3		AVDDR-1V	
	PGIA Gain Ratio	VDD=5V , PGIA=x128	-3		+3	%
DNL	Differential non-linearity	ADC range ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 2		LSB
INL	Integral non-linearity	ADC range ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 4		LSB
NMC	No missing code	ADC range ± 131072 x 0.9. (0.9 x Vref , 18-bits)		18		bit
NFB	Noise free bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		18.5		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		15.5		bit
ENOB	Effective number of bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		21		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		18		bit
V _{AIN}	ADC Input differential range	ADC input signal, signal after PGIA application	0.3		1.44	V
T _{Drift}	ADC/PGIA Temperature Drift	AVDDR = 2.7V		30		PPM/°C

26.5 Charge Pump Regulator Characteristic

Charge Pump						
SYM.	PARAMETER	Condition	MIN.	TYP.	MAX.	UNIT
V _{oper}	Operation Voltage (Charge Pump input)	Charge Pump Input Voltage Range (AVDD)	2.0	-	6.5	V
V _{AVDDCP}	Charge pump Output Range	Charge Pump OFF, AVDD from 2V to 6.5V AVDDCP = AVDD	2.0	-	6.5	V
		Busk-boost Mode, AVDD from 2V to 6.5V AVDDCP = 3.2V ~ 4.1V (AVDDR + 0.5V)	AVDDR + 0.5V			V
		2x Pump Mode, AVDD from 2V ~ 3.3V AVDDCP = 4V ~ 6.6V	2x AVDD			V
		2x Pump Mode, AVDD from 3.3V ~ 6.5V AVDDCP = AVDD.	AVDD			V
I _{AVDDCP}	Charge pump Output Current	Charge Pump On or OFF, AVDD from 2V to 6.5V	-	5	7.5	mA
I _{PUMP}	Charge pump intrinsic Current	Busk-boost Mode, AVDD=3V		160		uA
ACM						
V _{ACM}	Analog common voltage	Output voltage	0.9	1	1.1	V
		Output Voltage drift vs. Temperature $\Delta 10^{\circ}\text{C}$	-	± 0.1	-	%
		Output Voltage drift vs. AVDD (2V~6.5V)	-	± 0.1	-	%
I _{SRC}	VACM driving capacity	Only for ADC use	-	-	10	uA
I _{SNK}	VACM sinking capacity	Only for ADC use	-	-	1	mA
AVDDR						
V _{AVDDR}	Regulator output voltage AVDDR	AVDDR Output Range = 2.7V, 3.0V, 3.3V, 3.6V	2.7	-	3.6	V
		AVDDR set 2.7V	2.55	2.7	2.85	V
		AVDDR set 3.0V	2.85	3.0	3.15	V
		AVDDR set 3.3V	3.15	3.3	3.45	V
		AVDDR set 3.6V	3.45	3.6	3.75	V
		AVDDR Load regulation $\Delta 5\text{mA}$	V _{AVDDR} $\pm 0.05\text{V}$			V
		Output Voltage drift vs. Temperature $\Delta 10^{\circ}\text{C}$	-	± 0.1	-	%
		Output Voltage drift vs. AVDD (2V~6.5V)	-	± 0.1	-	%
I _{AVDDR}	AVDDR drive/Sink Current		-	-	7.5	mA
I _{QI}	Quiescent current	Pump + AVDDR + ACM		200		uA

26.6 OP-Amp Electrical Characteristic

Operation Amplifier						
SYM.	PARAMETER	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
V _{oper}	Operation voltage	Power source from AVDDR	2.7	-	3.6	V
V _{OS}	Input offset voltage			± 3		mV
I _{oper}	Operating Current	Per OP-Amp	-	185	-	uA
V _{IN}	Analog Input voltage range	OP+ / OP-	0.05		AVDDR-0.1	V
V _{out}	Analog Output voltage range	OPOUT	0.05		AVDDR-0.1	V
I _{OH} /I _{OL}	Output short circuit current	Unit Gain Buffer. Vo = 0V~3.2V, AVDDR=3.3V	-	± 5	-	mA
R _{BUF}	Buffer Mode Switch Ron	AVDDR 3.3V, VOUT=1.65V.	-	50	--	Ω
GB	Gain Bandwidth	RL=300K Ω , CL=50pF		1.4	-	MHz
SR	Slew Rate	10% to 90%	-	0.8	-	V/uS
T _{ON}	Turn on time		-	20	-	uS

26.7 Comparator Characteristic

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
V _{ILBT}	Internal Low-Battery detect voltage	Condition: LBTSEL [3:0] = 0000, VLBT=2.2V	2.04	2.15	2.26	V
		Condition: LBTSEL [3:0] = 0100, VLBT=3V	2.80	2.95	3.10	
		Condition: LBTSEL [3:0] = 0111, VLBT=3.6V	3.40	3.55	3.70	
V _{ELBT}	External Low-Battery detect voltage	Condition: LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.1	1.2	1.3	
V _{HY}	Comparator Hysteresis Window			50	-	mV
I _{COMP}	Current consumption	AVDD = 3V		50	-	uA

26.8 LCD Characteristic

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
I _{CLCD}	C-Type LCD Operation Current	VDD:3~5V, No panel, Low power Mode (BGM=0, DUTY[1:0]=00)		9		uA
		VDD:3~5V, 1/3 bias, No panel, Low power Mode (BGM=0, DUTY[1:0]=01)		10		
		VDD:3~5V, 1/3 bias, No panel, Low power Mode (BGM=0, DUTY[1:0]=10)		12		
		VDD:3~5V, 1/3 bias, No panel, Low power Mode (BGM=0, DUTY[1:0]=11)		17		
		VDD:3~5V, 1/3 bias, No panel, Normal Mode (BGM=1, DUTY[1:0]=xx)		127		
V _{LCD}	C-Type VLCD output Voltage	VDD: 2.0 ~ 6.5V, VLCD set 3V (VCP [2:0]=010)	2.8	3.0	3.2	V
		VDD: 2.5 ~ 6.5V, VLCD set 4.5V (VCP [2:0]=101)	3.4	4.5	4.7	

26.9 Flash Memory Characteristics

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
V _{dd}	Supply voltage		2.0		6.5	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.

26.10 Recommendations for using in special condition or environment

Condition / Environment	Suggestion
Condition (1) When MCU I/O connect to other circuit, and it has negative transient voltage appear in I/O, peak voltage over -3V, the MCU might occur latch-up phenomena.	Connecting 10-Ohm resistor before the power input the MCU's DVDD pin, that will solve the latch-put occurring by abnormal transient voltage appear in I/O.
Condition (2) When the power supply to MCU with high noise interference and the MCU function must work well. (MCU with High EFT ability)	MCU must set "Noise filter Enable" to enhance ability in High EFT environment.

27 Instruction Set

This chapter categorizes the SN8F5900 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.

27.1 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

27.2 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

27.3 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

27.4 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

27.5 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

27.6 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

28 Debug Interface

The debug interface, aka SWAT, shares one pin with GPIO P1.7 which can update full range of the on-chip program memory (IROM), and cooperate with development environment. The shared pin is automatically reserved for debug interface if a SN-Link is connected before microcontroller's power-on, whereas the pin remains other function(s) if it does not detect any handshake signal during microcontroller's power-on sequence.

***Note:** Port P1.7 can't connect any loading @Debug Mode.

28.1 Minimum Requirement

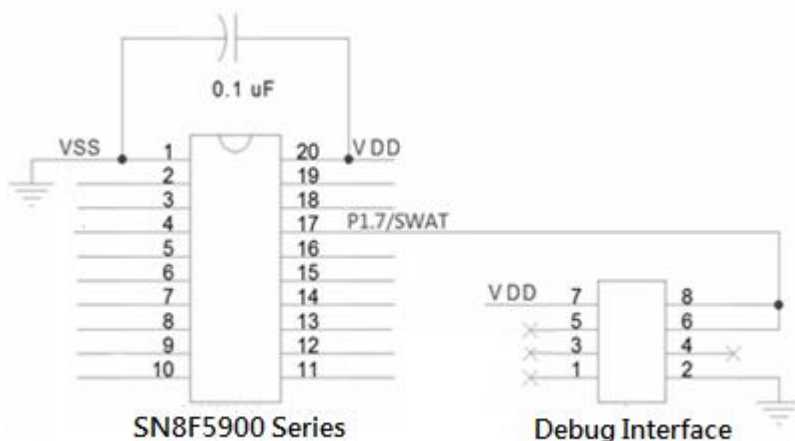
The following items are essential to build up an appropriate development environment. The compatibility is verified on listed versions, and is expected to execute perfectly on later version. SN-Link related information is available to download on SONiX website (www.sonix.com.tw); Keil C51 is downloadable on www.keil.com/c51.

- **SN-Link Adapter II** with updated firmware version ---
- **SN-Link Driver for Keil C51** version---
- **Keil C51** version ----

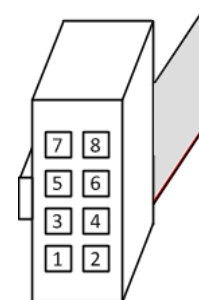
28.2 Debug Interface Hardware

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

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 red LED (Run) indicates the success of connection (refer *SN8F5000 Debug Tool Manual* for further detail).



example circuit



SN-Link header

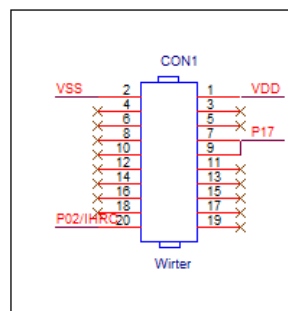
28.3 IHRC Calibration

MCU Internal 32 MHz Clock Generator (IHRC), whatever package or dice form, always be calibrated within 1% accuracy after IC mass production. In Customer's COB (Chip on Board) process, it is possible with few probabilities that the IHRC frequency is shifted over 1% after die encapsulation, cause by stress change on dice. We strong recommend that P02 and P17 pin must exist and reserve for writer re-programming and IHRC re-calibration, to solve few IC with IHRC frequency shift after COB processing if necessary.

However SN8F5900 Series provide IHRC calibration function for IHRC frequency drift. SN8F5900 Series provide debug interface 20th pin of Writer for IHRC calibration.

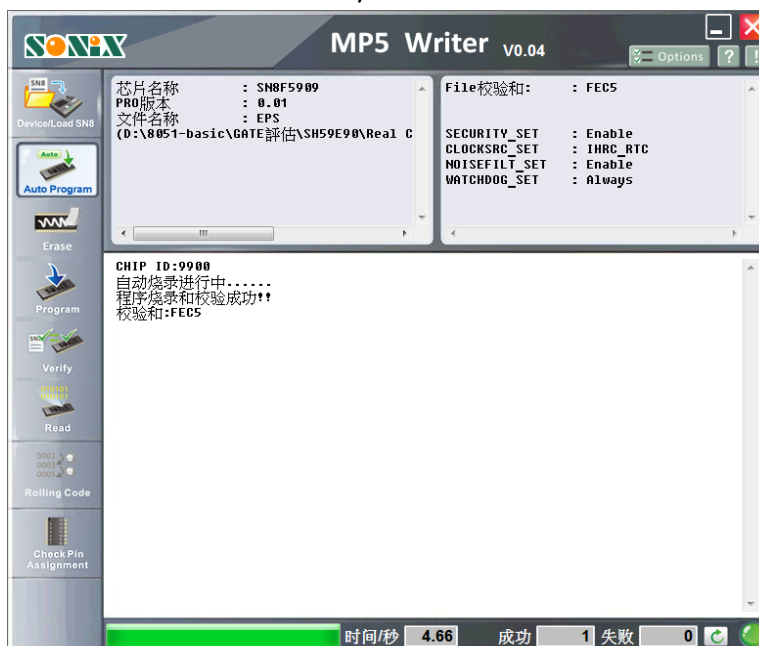
Following is IHRC calibration step:

- Step1: Download MP5-Writer from SONiX Web (<http://www.sonix.com.tw/article-tw-433-21531>)
- Step2: **Connect P02** pin with 20th pin of Writer.



Writer Interface

- Step3: Open MP5-Writer.exe → Device/Load SN8 → Select hex file format → Auto Program



* Note: IHRC calibration function was included Auto Program function.

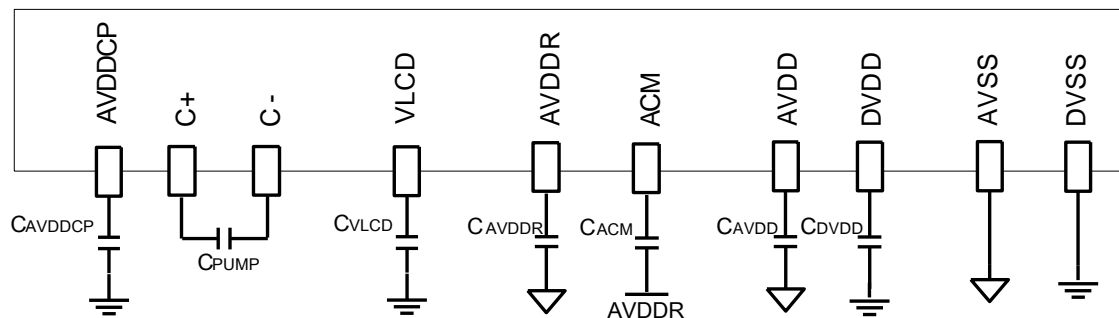
29 Analog Setting and Application

SN8F5900 is applied in many DC measurement applications, like weight Scale, pressure measure, blood pressure meter and thermometer. The following table indicate different applications setting which MCU power source come from CR2032 battery, AA/AAA dry battery or external Regulator.

Charge pump enable Capacitor Table:

Power	AVDDR	ACM	AVDDCP	C+/-	VLCD	AVDD	DVDD
	C _{AVDDR}	C _{ACM}	C _{AVDDCP}	C _{PUMP}	C _{VLCD}	C _{AVDD}	C _{DVDD}
CR2032 (2.4 ~ 3.6V)	0.47uf	0.1uf	2.2uf	1uf	0.1uf	10uf/104	10uf/104
2*AA/AAA Bat (2.4 ~ 3.6V)	1uf	0.1uf	2.2uf	1uf	0.1uf	10uf/104	10uf/104
4*AA/AAA Bat (4 ~ 6.5V)							

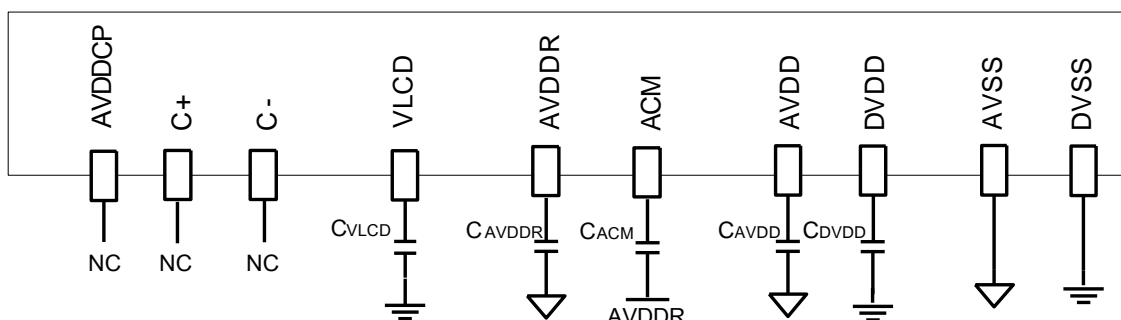
Analog Capacitor Connection (Charge pump enable):



Charge pump disable Capacitor Table (Charge pump disable):

Power type	AVDDR	ACM	AVDDCP	C+/-	VLCD	AVDD	DVDD
	C _{AVDDR}	C _{ACM}	C _{AVDDCP}	C _{PUMP}	C _{VLCD}	C _{AVDD}	C _{DVDD}
4*AA/AAA Bat (4V ~6.5V)	1uf	0.1uf	NC	NC	0.1uf	10uf/104	10uf/104

Analog Capacitor Connection (Charge pump disable):



30 Overview of Packaging Information

This documentation introduces SN8F5900 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.

SONiX Logo



Full Name

SN8F	5909	FG
8-bit MCU	Device Series	Package

Date Code

15	5	J	AEB11
Year	Mo.	Date	Internal Usage



Date Code

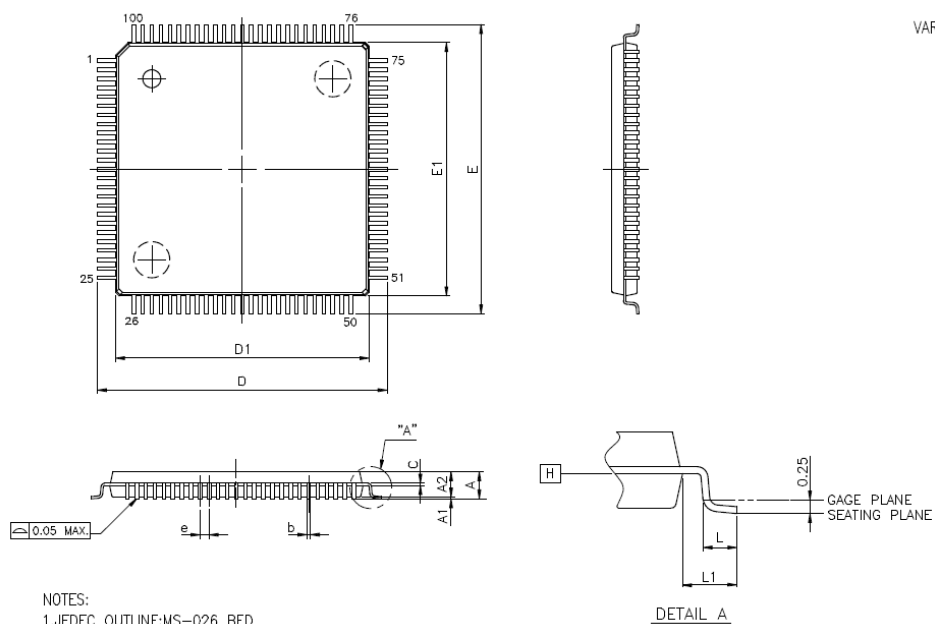
Year	15: 2015 16: 2016 17: 2017 et cetera
------	---

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
------	--

30.1 Device Nomenclature

Full Name	Packing Type
S8F5909W	Wafer
SN8F5909H	Dice
SN8F5909FG	LQFP, 100 pins, Green package
SN8F5908FG	LQFP, 80 pins, Green package
SN8F5907FG	LQFP, 64 pins, Green package

30.2 LQFP 100 PIN

NOTES:

1. JEDEC OUTLINE: MS-026 BCD

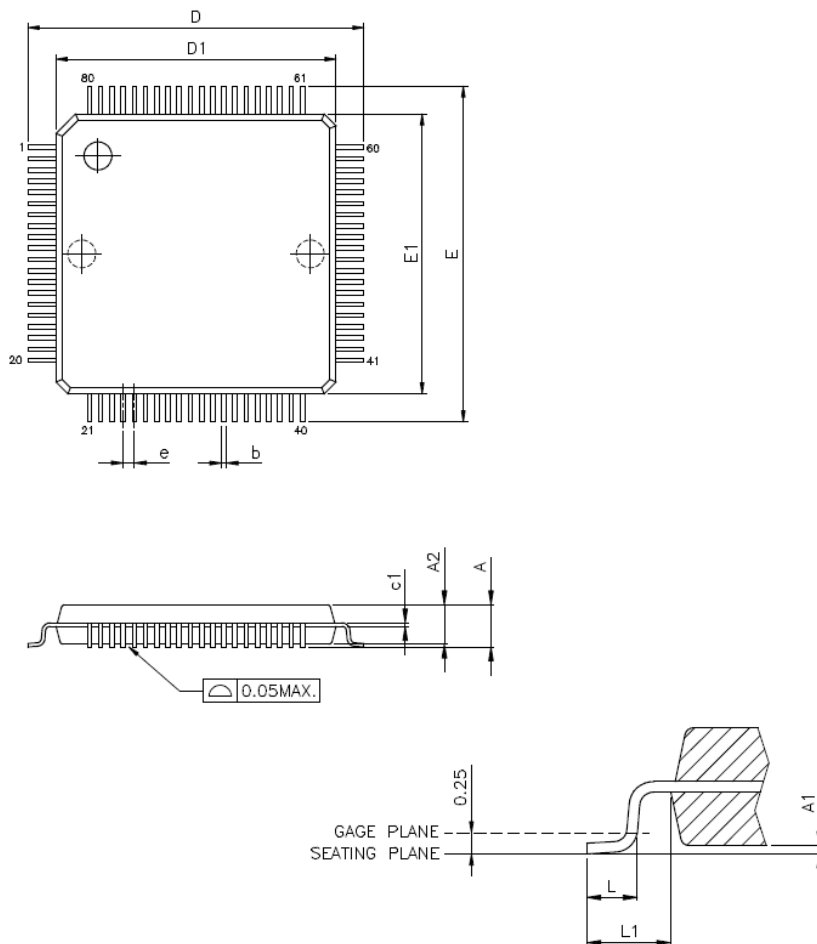
2. DATUM PLANE [H] IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.

3. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE [H].

4. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

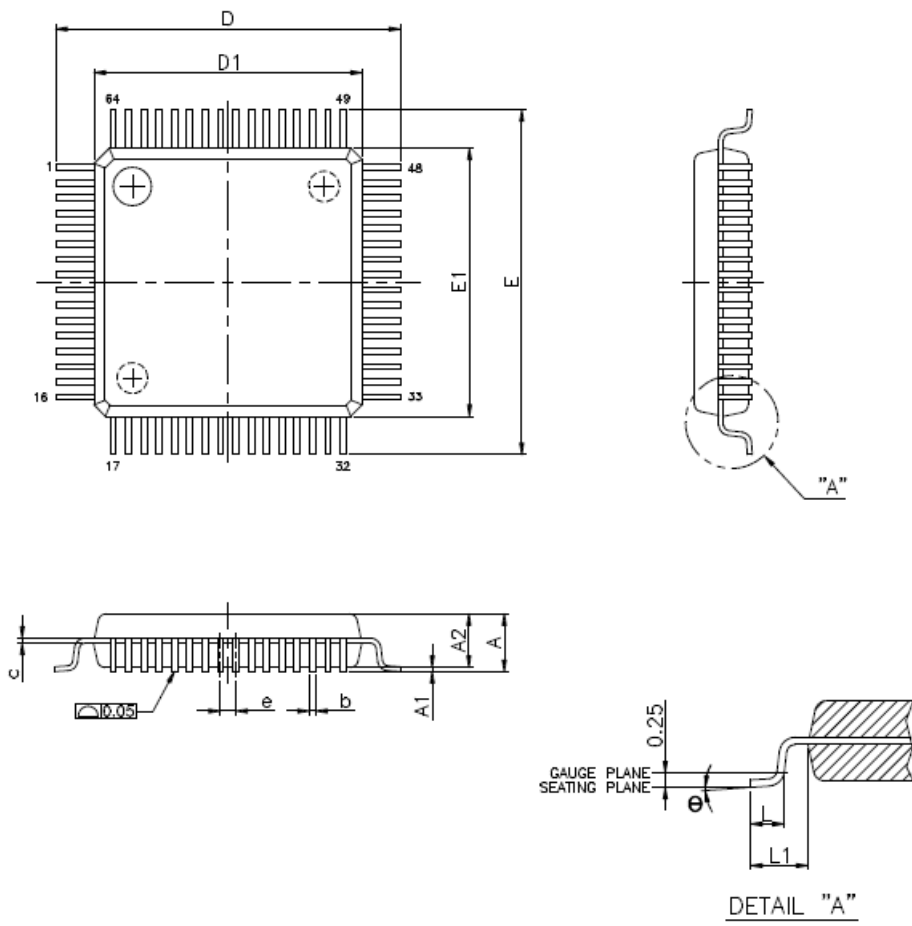
	Min	Typical (inch)	Max	Min	Typical (mm)	Max
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
C	0.004		0.008	0.09		0.20
D		0.630 BSC			16 BSC	
D1		0.551 BSC			14 BSC	
E		0.630 BSC			16 BSC	
E1		0.551 BSC			14 BSC	
e		0.020 BSC			0.5 BSC	
B	0.007		0.011	0.17		0.27
L	0.018		0.030	0.45		0.75
L1		0.039 REF			1.00 REF	

30.3 LQFP 80 PIN



	Min	Typical (inch)	Max	Min	Typical (mm)	Max
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
c1	0.004		0.006	0.09		0.16
D		0.473 BSC			12 BSC	
D1		0.393 BSC			10 BSC	
E		0.473 BSC			12 BSC	
E1		0.393 BSC			10 BSC	
e		0.016 BSC			0.4 BSC	
B	0.005		0.009	0.13		0.23
L	0.018		0.030	0.45		0.75
L1		0.039 REF			1.00 REF	

30.4 LQFP 64 PIN



	Min	Typical (inch)	Max	Min	Typical (mm)	Max
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
c	0.004		0.008	0.09		0.20
D		0.354 BSC			9 BSC	
D1		0.276 BSC			7 BSC	
E		0.354 BSC			9 BSC	
E1		0.276 BSC			7 BSC	
e		0.016 BSC			0.4 BSC	
b	0.005		0.009	0.13		0.23
L	0.018		0.030	0.45		0.75
L1		0.039 REF			1.00 REF	

SN8F5900 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