

SN8F5701 Series Datasheet

8051-based Microcontroller

SN8F5701

SN8F57011

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)
- **4 KB non-volatile flash memory (IROM)** with in-system program support
- **256 bytes internal RAM (IRAM)**
- **8 interrupt sources with priority levels control and unique interrupt vectors**
- 5 internal interrupts
- 3 external interrupts: INTO/INT1/INT2
- 1 set of DPTR
- 2 set 8/16-bit timers with 4 operation modes
- **1 set 16-bit timers with PWM generator**
- each PWM generator has 6 output channels
- with individual duty, inverters and frequency control
- **12-bit SAR ADC** with 6 external and 3 internal channels, and 4 internal reference voltages
- **UART** interface
- **On-Chip Debug Support:**
 - Single-wire debug interface
 - 2 hardware breakpoints
 - Unlimited software breakpoints
 - ROM data security/protection
- Watchdog and programmable external reset
- 1.8/2.4/3.3-V low voltage detectors
- Wide supply voltage (1.8 V – 5.5 V) and temperature (-40 °C to 85 °C) range

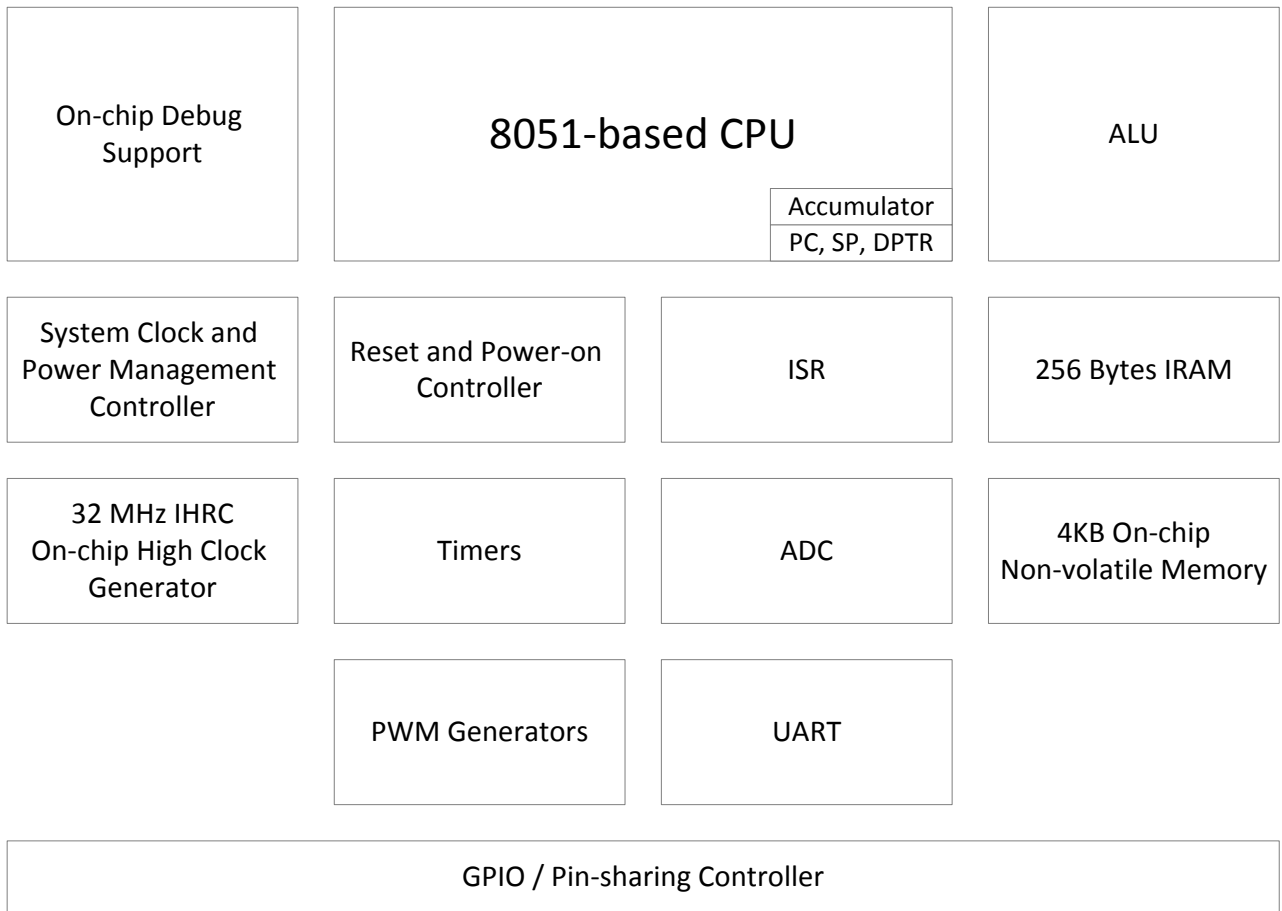
1.2 Applications

- Brushless DC motor
- Home automation
- Household
- Other

1.3 Features Selection Table

	I/O	PWM Channels	I2C	SPI	UART	ADC ext. Channels	OPA	CMP	Ext. INT	Package Types
SN8F5701	6	6	-	-	V	6	-	-	3	DIP8, SOP8, TSSOP8
SN8F57011	4	4	-	-	V	4	-	-	2	SOT23-6L

1.4 Block Diagram



2 Table of Contents

1	Device Overview.....	2
2	Table of Contents	4
3	Revision History.....	5
4	Pin Assignments	6
5	CPU.....	10
6	Special Function Registers.....	14
7	Reset and Power-on Controller.....	22
8	System Clock and Power Management.....	26
9	Interrupt.....	30
10	GPIO	36
11	External Interrupt.....	41
12	Timer 0 and Timer 1	43
13	Timer3	50
14	ADC.....	58
15	UART.....	68
16	In-System Program	75
17	Clock Fine-Tuning	79
18	Electrical Characteristics	82
19	Instruction Set	85
20	Debug Interface.....	90
21	ROM Programming Pin.....	91
22	Ordering Information	93
23	Appendix: Reference Document	95

3 Revision History

Revision	Date	Description
1.0	Apr. 2017	First issue

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 SN8F5701P/S/T (DIP8/SOP8/TSSOP8)

VDD	1	U	8	VSS
PWM0/INT0/AIN0/P00	2		7	P05/AIN5/PWM5
PWM1/INT1/AIN1/P01	3		6	P04/AIN4/PWM4/SWAT
URX/PWM2/INT2/RST/AIN2/P02	4		5	P03/AIN3/PWM3/UTX

4.2 SN8F57011D (SOT23-6L)

SWAT/PWM4/AIN4/P04	1	U	6	VDD
VSS	2		5	P00/AIN0/INT0/PWM0
UTX/PWM3/AIN3/P03	3		4	P02/AIN2/RST/INT2/ PWM2/ URX

4.3 Pin Descriptions

Power Pins

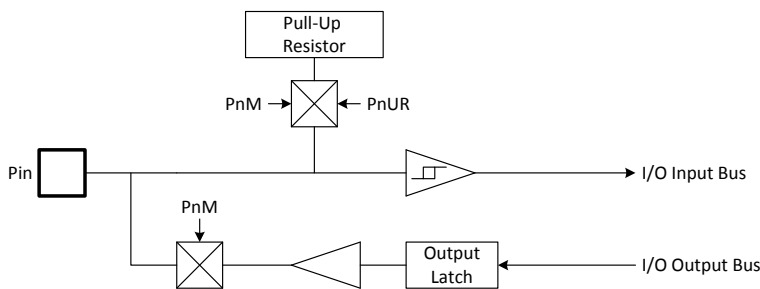
Pin Name	Type	Description
VDD	Power	Power supply
VSS	Power	Ground (0 V)

Port 0

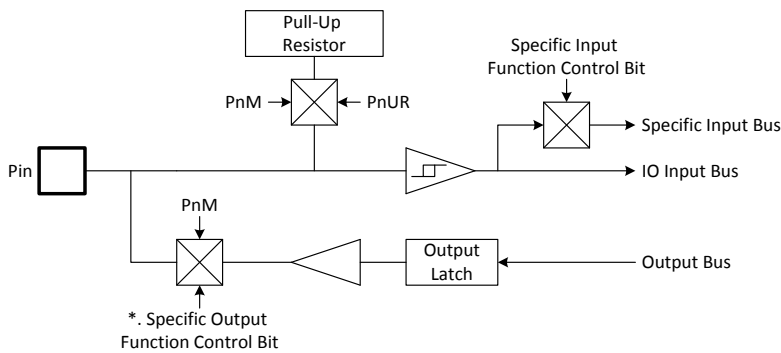
Pin Name	Type	Description
P0.0	Digital I/O	GPIO
AIN0	Analog Input	ADC: input channel 0
PWM0	Digital Output	PWM: programmable PWM output
INT0	Digital Input	INT0: external interrupt 0
P0.1	Digital I/O	GPIO
AIN1	Analog Input	ADC: input channel 1
PWM1	Digital Output	PWM: programmable PWM output
INT1	Digital Input	INT1: external interrupt 1
P0.2	Digital I/O	GPIO
AIN2	Analog Input	ADC: input channel 2
Reset	Digital Input	System reset (active low)
PWM2	Digital Output	PWM: programmable PWM output
INT2	Digital Input	INT2: external interrupt 2
URX	Digital Input	UART: reception pin
P0.3	Digital I/O	GPIO
AIN3	Analog Input	ADC: input channel 3
PWM3	Digital Output	PWM: programmable PWM output
UTX	Digital Output	UART: transmission pin
P0.4	Digital I/O	GPIO
AIN4	Analog Input	ADC: input channel 4
SWAT	Digital I/O	Debug interface
PWM4	Digital Output	PWM: programmable PWM output
P0.5	Digital I/O	GPIO
AIN5	Analog Input	ADC: input channel 5
PWM5	Digital Output	PWM: programmable PWM output

4.4 Pin Circuit Diagrams

Normal Bi-direction I/O Pin.

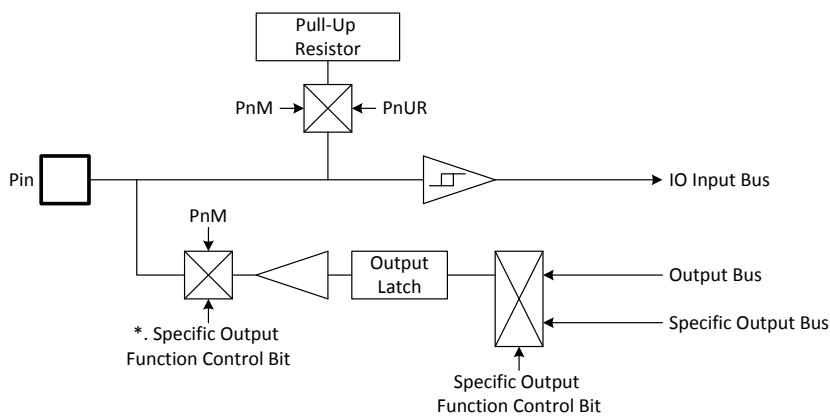


Bi-direction I/O Pin Shared with Specific Digital Input Function, e.g. INTO, UART.



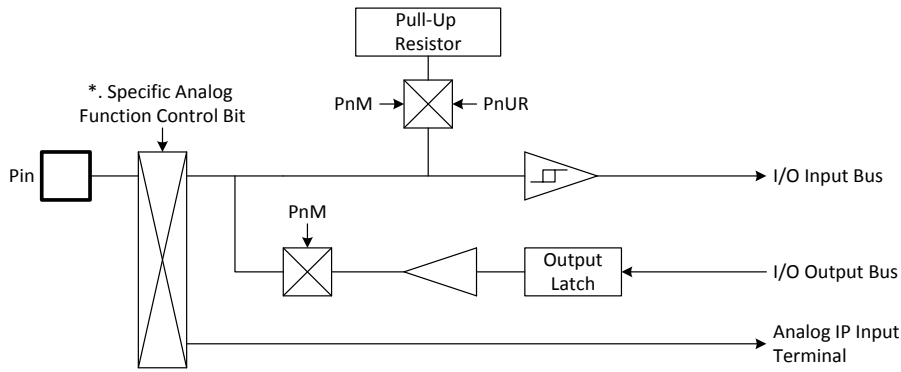
*. Some specific functions switch I/O direction directly, not through PnM register.

Bi-direction I/O Pin Shared with Specific Digital Output Function, e.g. T3, UART.



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

Bi-direction I/O Pin Shared with Specific Analog Input Function, e.g. ADC.



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

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.

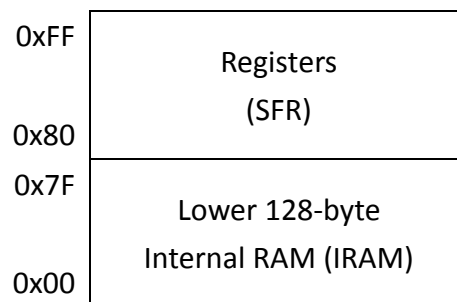
5.1 Memory Organization

SN8F5701 builds in two on-chip memories: internal RAM (IRAM), and program memory (IROM). The internal RAM is a 256-byte RAM which has higher access performance (direct and indirect addressing). The program memory is a 4 KB non-volatile memory and has a maximum 8 MHz speed limitation.



5.2 Direct Addressing: IRAM and SFR

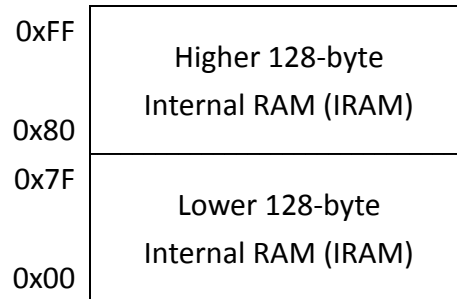
Direct addressing instructions (e.g. MOV A, direct) can access the lower 128-byte internal RAM (address range: 0x00 – 0x7F) and all registers (SFR, address range: 0x80 – 0xFF).



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.

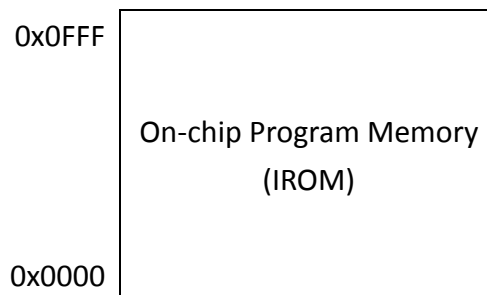
5.3 Indirect Addressing: IRAM

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 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).



5.5 Program Memory Security

The SN8F5701 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.

5.6 Data Pointer

A data pointer helps to specify the IROM address while performing MOVC instructions. The microcontroller has one set of data pointer (DPH/DPL).

5.7 Stack

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



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

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

5.8 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

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

6 Special Function Registers

6.1 Special Function Register Memory Map

BIN HEX	000	001	010	011	100	101	110	111
F8	-	POM	-	-	-	-	-	PFLAG
F0	B	POUR	FRQL	FRQH	FRQCMD	-	-	SRST
E8	-	-	-	-	-	-	-	-
E0	ACC	-	-	-	POOC	CLKSEL	CLKCMD	TCON0
D8	SOCON2	-	-	-	-	-	-	-
D0	PSW	-	ADM	ADB	ADR	VREFH	-	-
C8	-	-	-	-	-	-	-	-
C0	-	-	-	-	-	-	-	-
B8	-	IP1	SORELH	-	PWNV	PWO	PWCH	IRCON2
B0	-	PW3DL	PW3DH	PW4DL	PW4DH	PW5DL	PW5DH	-
A8	IEN0	IPO	SORELL	PW1DL	PW1DH	PW2DL	PW2DH	-
A0	-	T3M	T3CL	T3CH	T3YL	T3YH	PW0DL	PW0DH
98	SOCON	SOBUF	IEN2	-	-	-	POCON	-
90	-	-	-	-	PECMD	PEROML	PEROMH	PERAM
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PEDGE
80	P0	SP	DPL	DPH	-	-	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.
-	0x84	-
-	0x85	-
WDTR	0x86	Watchdog timer clear register.
PCON	0x87	System mode register.
TCON	0x88	Timer 0 / 1 controls register.
TMOD	0x89	Timer 0 / 1 mode register.
TL0	0x8A	Timer 0 counting low byte register.
TL1	0x8B	Timer 1 counting low byte register.
TH0	0x8C	Timer 0 counting high byte register.
TH1	0x8D	Timer 1 counting high byte register.
CKCON	0x8E	Extended cycle controls register.
PEDGE	0x8F	External interrupt edge controls register.
-	0x90	-
-	0x91	-
-	0x92	-
-	0x93	-
PECMD	0x94	In-System Program command register.
PEROML	0x95	In-System Program ROM address low byte
PEROMH	0x96	In-System Program ROM address high byte
PERAM	0x97	In-System Program RAM mapping address
SOCON	0x98	UART control register.
SOBUF	0x99	UART data buffer.
IEN2	0x9A	Interrupts enable register
-	0x9B	-
-	0x9C	-
-	0x9D	-
POCON	0x9E	Port 0 configuration controls register.
-	0x9F	-

0xA0 - 0xBF Registers Description

Register	Address	Description
-	0xA0	-
T3M	0xA1	Timer 3 controls register.
T3CL	0xA2	Timer 3 counting low byte register.
T3CH	0xA3	Timer 3 counting high byte register.
T3YL	0xA4	Timer 3 cycle controls buffer low byte.
T3YH	0xA5	Timer 3 cycle controls buffer high byte.
PW0DL	0xA6	PW0 duty controls buffer low byte.
PW0DH	0xA7	PW0 duty controls buffer high byte.
IEN0	0xA8	Interrupts enable register
IPO	0xA9	Interrupts priority register.
SORELL	0xAA	UART reload low byte register.
PW1DL	0xAB	PW1 duty controls buffer low byte.
PW1DH	0xAC	PW1 duty controls buffer high byte.
PW2DL	0xAD	PW2 duty controls buffer low byte.
PW2DH	0xAE	PW2 duty controls buffer high byte.
-	0xAF	-
-	0xB0	-
PW3DL	0xB1	PW3 duty controls buffer low byte.
PW3DH	0xB2	PW3 duty controls buffer high byte.
PW4DL	0xB3	PW4 duty controls buffer low byte.
PW4DH	0xB4	PW4 duty controls buffer high byte.
PW5DL	0xB5	PW5 duty controls buffer low byte.
PW5DH	0xB6	PW5 duty controls buffer high byte.
-	0xB7	-
-	0xB8	-
IP1	0xB9	Interrupts priority register.
SORELH	0xBA	UART reload high byte register.
-	0xBB	-
PWNV	0xBC	PWM inverse control register.
PWO	0xBD	PWM frequency control register.
PWCH	0xBE	PWM channel control buffer.
IRCON2	0xBF	Interrupts request register.

0xC0 - 0xDF Registers Description

Register	Address	Description
-	0xC0	-
-	0xC1	-
-	0xC2	-
-	0xC3	-
-	0xC4	-
-	0xC5	-
-	0xC6	-
-	0xC7	-
-	0xC8	--
-	0xC9	-
-	0xCA	-
-	0xCB	-
-	0xCC	-
-	0xCD	-
-	0xCE	-
-	0xCF	-
PSW	0xD0	System flag register.
-	0xD1	-
ADM	0xD2	ADC controls register.
ADB	0xD3	ADC data buffer.
ADR	0xD4	ADC resolution selects register.
VREFH	0xD5	ADC reference voltage controls register.
-	0xD6	-
-	0xD7	-
SOCON2	0xD8	UART baud rate controls register.
-	0xD9	-
-	0xDA	-
-	0xDB	-
-	0xDC	-
-	0xDD	-
-	0xDE	-
-	0xDF	-

0xE0 - 0xFF Registers Description

Register	Address	Description
ACC	0xE0	Accumulator register.
-	0xE1	-
-	0xE2	-
-	0xE3	-
P0OC	0xE4	Open drain controls register.
CLKSEL	0xE5	Clock switch selects register.
CLKCMD	0xE6	Clock switch controls Register.
TCON0	0xE7	Timer 0 / 1 clock controls register.
-	0xE8	-
-	0xE9	-
-	0xEA	-
-	0xEB	-
-	0xEC	-
-	0xED	-
-	0xEE	-
-	0xEF	-
B	0xF0	Multiplication/ division instruction data buffer.
POUR	0xF1	Port 0 pull-up resister controls register.
FRQL	0xF2	Clock fine tuning controls buffer low byte
FRQH	0xF3	Clock fine tuning controls buffer high byte
FRQCMD	0xF4	Clock fine tuning command register.
-	0xF5	-
-	0xF6	-
SRST	0xF7	Software reset controls register.
-	0xF8	-
POM	0xF9	Port 0 input/output mode register.
-	0xFA	-
-	0xFB	-
-	0xFC	-
-	0xFD	-
-	0xFE	-
PFLAG	0xFF	Reset flag register.

6.3 System Registers

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

ACC Register (0xE0)

Bit	Field	Type	Initial	Description
7..0	ACC[7:0]	R/W	0x00	The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is overflow (OV) or there is carry (C or AC) and parity (P) occurrence, then these flags will be set to PSW register.

B Register (0xF0)

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

PSW Register (0xD0)

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

6.4 Register Declaration

SN8F5701 has many registers to control various functions, but SFR name is not predefined in the C51 / A51 compiler. To make programming easier and therefore need to add header files to declare SFR name.

When using the assembly code programs, please add the following sentence.

```
1 $NOMOD51 ;Do not recognize the 8051-specific predefined special register.
2 #include <SN8F5701.H>
```

When using the C code programs, please add the following sentence.

```
1 #include <SN8F5701.H>
```

After adding the header file, user can use name of registers to program. During compilation, the compiler will register name translate into register position through the header file.

Different devices need to use a different header file to declare, but the option file is to use the same.

Device	Header file	Options file
SN8F5701	SN8F5701.h	OPTIONS_SN8F5701.A51
SN8F57011	SN8F57011.h	

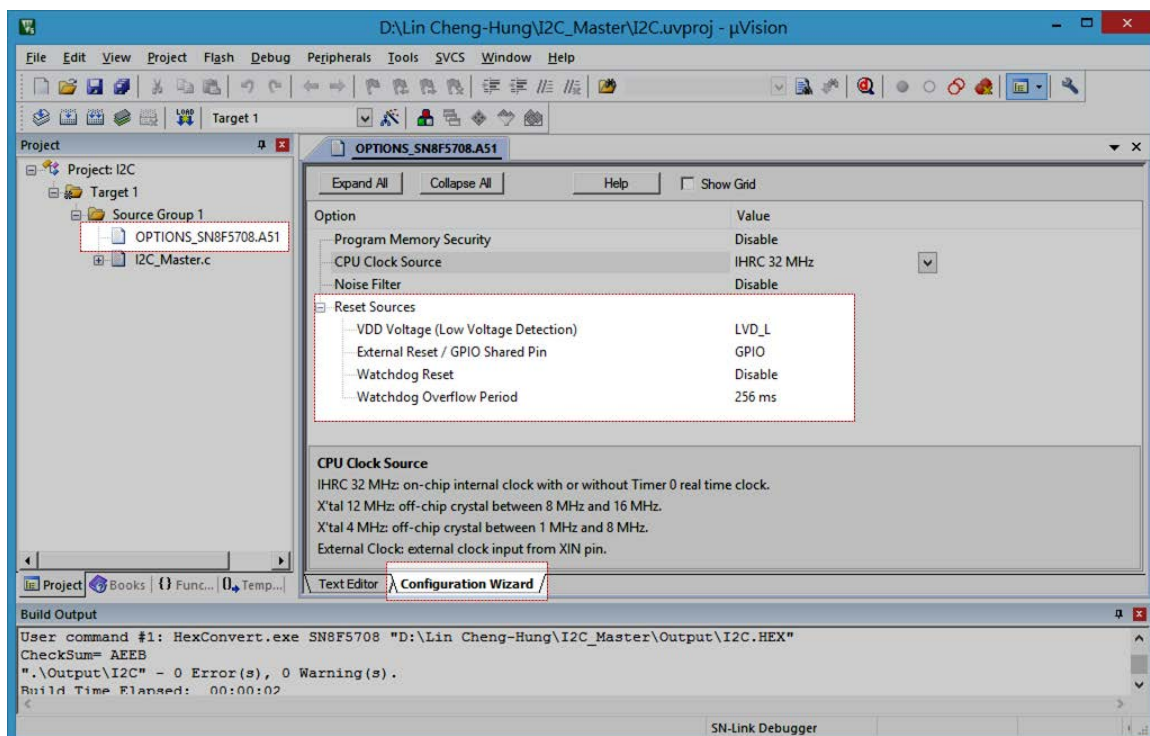
7 Reset and Power-on Controller

The reset and power-on controller has four 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 an *OPTIONS_SN8F5701.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.

- Program Memory Security
- CPU Clock Source
- Noise Filter
- Reset Source : VDD Voltage (Low Voltage Detection)
- Reset Source : External Reset / GPIO Shared Pin
- Reset Source : Watchdog Reset & Overflow Period



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 4.5 ms 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 only one level: 1.8 V. Depend on low voltage detection configuration, the comparison result can be seen as a system reset signal. The table below lists low voltage detection configuration, LVD_L, and the results of VDD pin's condition.

Condition	LVD_L
VDD ≤ 1.8 V	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;
```

Watchdog timer interval time = $256 * 1 / (\text{Internal Low-Speed oscillator frequency} / \text{WDT Pre-scaler})$
 = $256 / (F_{ILRC} / \text{WDT Pre-scaler}) \dots \text{sec}$

Internal low-speed oscillator	WDT pre-scaler	Watchdog interval time
F _{ILRC} =16 kHz	F _{ILRC} /4	256/(16000/4)=64ms
	F _{ILRC} /8	256/(16000/8)=128ms
	F _{ILRC} /16	256/(16000/16)=256ms
	F _{ILRC} /32	256/(16000/32)=512ms

The operation mode of watchdog is configurable in options file:

Always mode counts its internal timer in all CPU operation modes (normal, IDLE, SLEEP);

Enable mode counts its internal timer during CPU stays in normal mode, and it would not trigger watchdog reset in IDLE and STOP modes;

Disable mode suspends its internal timer at all CPU modes, and the watchdog would not trigger in this condition.

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

7.5 External Reset Pin

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

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

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	RST	-	-	-	-	-
SRST	-	-	-	-	-	-	-	SRSTREQ
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0

PFLAG Register

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

been reset by external reset pin.

4..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.

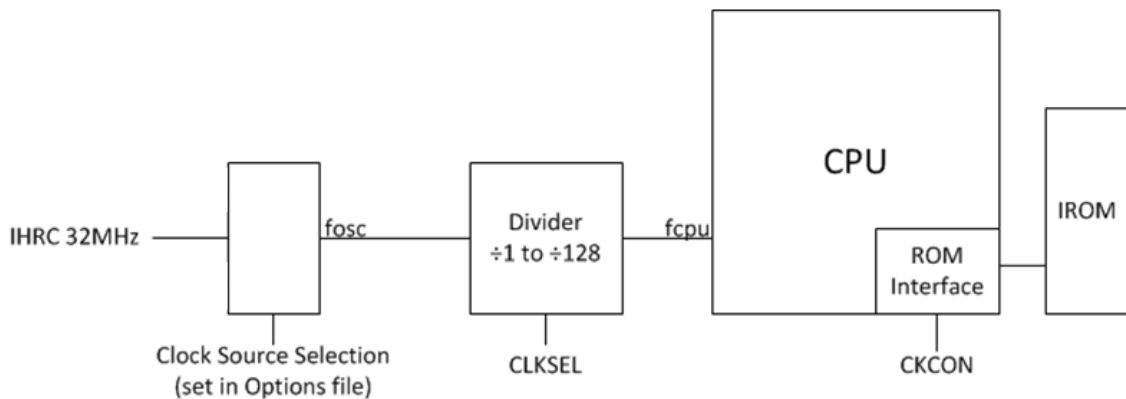
The normal mode means that CPU and peripheral functions are under normally execution. The system clock is based on the combination of source selection, clock divider, and program memory wait state. IDLE mode is the situation that temporarily suspends CPU clock and its execution, yet it remains peripherals' functionality (e.g. timers, PWM, and UART). By contrast, STOP mode disables all functions and clock generator until a wakeup signal to return normal mode.

8.1 System Clock

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

Subsequently, the selected clock source (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 8 MHz, three extended cycle must be placed by CKCON register.

$$\text{IROM fetching cycle} = \frac{f_{\text{cpu}}}{\text{PWSC}[2:0]+1} \leq 8\text{MHz}, \text{PWSC}[2:0] = 0\sim 7$$

8.2 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, and UART) and clock generator (IHRC 32 MHz) remain execution in this status. Any change from P0 input and interrupt events can make the microcontroller turns back to normal mode, and the IDLE bit would be cleared automatically.

- Any function can work in IDLE mode. Only CPU is suspended
- The IDLE mode wake-up sources are P0 level change trigger and any interrupt event.

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 can wake up the microcontroller and resume system's execution. STOP bit would be cleared automatically.

- CPU, peripheral functions, and clock generator are suspended.
- The STOP mode wake-up source is P0 level change trigger.

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 ( );
2 STOP ( );
```

8.3 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	-	-	-	-	GF0	STOP	IDLE

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 001: 1 cycle 010: 2 cycles 011: 3 cycles 100: 4 cycles 101: 5 cycles 110: 6 cycles 111: 7 cycles
Else	Reserved	R	0001	

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: $f_{cpu} = f_{osc} / 128$ 001: $f_{cpu} = f_{osc} / 64$ 010: $f_{cpu} = f_{osc} / 32$ 011: $f_{cpu} = f_{osc} / 16$ 100: $f_{cpu} = f_{osc} / 8$ 101: $f_{cpu} = f_{osc} / 4$ 110: $f_{cpu} = f_{osc} / 2$ 111: $f_{cpu} = f_{osc} / 1$

CLKCMD Register (0xE6)

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

PCON Register (0x87)

Bit	Field	Type	Initial	Description
7				Refer to other chapter(s)
6..3	Reserved	R	0x00	

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

9 Interrupt

The MCU provides 8 interrupt sources (3 external and 5 interrupt) 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. Some interrupt request flags must be cleared by software. However, most interrupt request flags can be cleared by hardware automatically. In the end, ISR is finished after complete the RETI instruction. The summary of interrupt source, interrupt vector, priority order and control bit are shown as the table below.

Interrupt	Enable Interrupt	Request (IRQ)	IRQ Clearance	Priority / Vector
System Reset	-	-	-	0 / 0x0000
INT0	EX0	IE0	Automatically	1 / 0x0003
INT2	EX2	IE2	Automatically	2 / 0x0083
Timer 0	ET0	TF0	Automatically	3 / 0x000B
ADC	EADC	ADCF	Automatically	4 / 0x008B
INT1	EX1	IE1	Automatically	5 / 0x0013
Timer 1	ET1	TF1	Automatically	6 / 0x001B
UART	ES0	TIO / RIO	By firmware	7 / 0x0023
Timer 3	ET3	TF3	Automatically	8 / 0x00EB

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 – 0x8B) 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 IPO/IP1 registers. Level 3 is the highest priority and Level 0 is the lowest. The interrupt sources inside the same group will share the same priority level. With the

same priority level, the priority rule follows default priority.

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

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

Group	Interrupt Source			
Group 0	INT0	INT2		
Group 1	T0	ADC		
Group 2	INT1			
Group 3	T1			
Group 4	UART			
Group 5	T3			

IP0, IP1 Registers

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

IP0 Register (0XA9)

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

IP1 Register (0XB9)

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

9.3 Request Flag Clearance

When the interrupt function is enabled, most of the interrupt request flags can be cleared by hardware automatically. Some still need to use the program to clear. However, when the interrupt function is turned off, the interrupt request flag only be cleared by program.

Most cases can be cleared by the ANL instruction. But some special cases, if the adjacent flag is issued asynchronously, it may be accidentally cleared.

```
1   TCON &= 0x7F;    // It is possible to cause adjacent flags to be cleared.
2   TF1 = 0;        // It is possible to cause adjacent flags to be cleared.
```

If you want to avoid the above, it is recommended to use the interrupt bit characteristics. Most of the interrupt request flag can't be written 1 by which you can avoid clearing asynchronous adjacent flags.

For user who is develop program in C language, the flag clear macros is strongly recommended to clear request flag, instead of clear request flag bits directly.

```
1   TCONCLR(bit);    // The marco can clear the flag of TCON. bit is 0~7.
2   IRCON2CLR(bit); // The marco can clear the flag of IRCON2. bit is 0~7.
```

9.4 Interrupt Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	-	ESO	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF
TCON	TF1	TR1	TF0	TRO	IE1	-	IE0	-
SOCON	SM0	SM1	SM20	RENO	TB80	RB80	TIO	RIO

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.
3	ET1	R/W	0	T1 timer interrupt control bit. 0: Disable T1 interrupt function. 1: Enable T1 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	

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
2	EX2	R/W	0	External P0.2 interrupt (INT2) control bit. 0: Disable INT2 interrupt function. 1: Enable INT2 interrupt function.
1	ET3	R/W	0	T3 timer interrupt control bit. 0: Disable T3 interrupt function. 1: Enable T3 interrupt function.
0	EADC	R/W	0	ADC interrupt control bit. 0: Disable ADC interrupt function. 1: Enable ADC interrupt function.
Else	Reserved	R	0	

IRCON2 Register (0xBF)

Bit	Field	Type	Initial	Description
2	IE2	R/W	0	External P0.2 interrupt (INT2) request flag 0: None INT2 interrupt request. 1: INT2 interrupt request.
1	TF3	R/W	0	T3 timer external reload interrupt request flag. 0: None T3 interrupt request 1: T3 interrupt request.
0	ADCF	R/W	0	ADC interrupt request flag. 0: None ADC interrupt request. 1: ADC interrupt request.
Else	Reserved	R	0	

TCON Register (0X88)

Bit	Field	Type	Initial	Description
7	TF1	R/W	0	T1 timer external reload interrupt request flag. 0: None T1 interrupt request 1: T1 interrupt request.
5	TF0	R/W	0	T0 timer external reload interrupt request flag. 0: None T0 interrupt request 1: T0 interrupt request.
3	IE1	R/W	0	External P0.1 interrupt (INT1) request flag 0: None INT1 interrupt request. 1: INTO interrupt request.
1	IE0	R/W	0	External P0.0 interrupt (INT0) request flag 0: None INTO interrupt request. 1: INTO interrupt request.
Else				Refer to other chapter(s)

SOCON Register (0X98)

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

10 GPIO

The microcontroller has up to 6 bidirectional general purpose I/O pin (GPIO). Unlike the original 8051 only has open-drain output, SN8F5701 builds in push-pull output structure to improve its driving performance.

10.1 Input and Output Control

The input and output direction control is configurable through P0M 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	-	-	P05M	P04M	P03M	P02M	P01M	P00M
P0OC	-	-	P05OC	P04OC	P03OC	P02OC	P01OC	P00OC

P0M: 0xF9

Bit	Field	Type	Initial	Description
5	P05M	R/W	0	Mode selection of P0.5 0: Input mode 1: Output mode
4	P04M	R/W	0	Mode selection of P0.4 0: Input mode 1: Output mode
3	P03M	R/W	0	Mode selection of P0.3 0: Input mode 1: Output mode
2	P02M	R/W	0	Mode selection of P0.2 0: Input mode 1: Output mode
1	P01M	R/W	0	Mode selection of P0.1 0: Input mode 1: Output mode
0	P00M	R/W	0	Mode selection of P0.0 0: Input mode 1: Output mode
Else	Reserved	R	0	

P0OC Register (0xE4)

Bit	Field	Type	Initial	Description
5	P05OC	R/W	0	P0.5 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
4	P04OC	R/W	0	P0.4 open-drain output mode* 0: Disable 1: Enable, output high status becomes to input mode
3	P03OC	R/W	0	P0.3 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
2	P02OC	R/W	0	P0.2 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
1	P01OC	R/W	0	P0.1 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
0	P00OC	R/W	0	P0.0 open-drain output mode 0: Disable 1: Enable, output high status becomes to input mode
Else	Reserved	R	0	

* Recommended disable open-drain output if P04 executes debug interface function.

10.2 Input Data and Output Data

By a read operation from any register of P0, 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 which can monitor the bus condition in some case.

A write P0 register value would be latched immediately, yet the value would be outputted until the mapped POM is set to output mode. If the pin is currently in output mode, any value set to P0 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	-	-	P05	P04	P03	P02	P01	P00

P0: 0x80

Bit	Field	Type	Initial	Description
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	P04	R/W	1	Read: P0.4 pin's logic level Write 1/0: Output logic high or low (applied if P04M = 1)
3	P03	R/W	1	Read: P0.3 pin's logic level Write 1/0: Output logic high or low (applied if P03M = 1)
2	P02	R/W	1	Read: P0.2 pin's logic level Write 1/0: Output logic high or low (applied if P02M = 1)
1	P01	R/W	1	Read: P0.1 pin's logic level Write 1/0: Output logic high or low (applied if P01M = 1)
0	P00	R/W	1	Read: P0.0 pin's logic level Write 1/0: Output logic high or low (applied if P00M = 1)
Else	Reserved	R	0	

10.3 On-chip Pull-up Resistors

The P0UR register 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	-	-	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR

P0UR: 0xF1

Bit	Field	Type	Initial	Description
5	P05UR	R/W	0	On-chip pull-up resistor control of P0.5 0: Disable* 1: Enable
4	P04UR	R/W	0	On-chip pull-up resistor control of P0.4 0: Disable* 1: Enable
3	P03UR	R/W	0	On-chip pull-up resistor control of P0.3 0: Disable* 1: Enable
2	P02UR	R/W	0	On-chip pull-up resistor control of P0.2 0: Disable* 1: Enable

1	P01UR	R/W	0	On-chip pull-up resistor control of P0.1 0: Disable* 1: Enable
0	P00UR	R/W	0	On-chip pull-up resistor control of P0.0 0: Disable* 1: Enable
Else	Reserved	R	0	

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

10.4 Pin Shared with Analog Function

The microcontroller builds in analog functions, such as ADC. The Schmitt trigger of input channel is strongly recommended to switch off if the pin's shared analog function is enabled.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POCON	-	-	POCON5	POCON4	POCON3	POCON2	POCON1	POCON0

POCON: 0x9E

Bit	Field	Type	Initial	Description
5	POCON5	R/W	0	Schmitt trigger control of P0.5 0: Enable 1: Disable
4	POCON4	R/W	0	Schmitt trigger control of P0.4 0: Enable 1: Disable
3	POCON3	R/W	0	Schmitt trigger control of P0.3 0: Enable 1: Disable
2	POCON2	R/W	0	Schmitt trigger control of P0.2 0: Enable 1: Disable
1	POCON1	R/W	0	Schmitt trigger control of P0.1 0: Enable 1: Disable
0	POCON0	R/W	0	Schmitt trigger control of P0.0 0: Enable 1: Disable

Else	Reserved	R	0
------	----------	---	---

11 External Interrupt

INT0, INT1 and INT2 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/EX2) and global interrupt (EAL) are enabled, the external interrupt request flag (IE0/IE1/IE2) will be set to “1” as edge trigger event occurs. The program counter will jump to the interrupt vector (ORG 0x0003/ 0x0013/ 0x0083) and execute interrupt service routine. Interrupt request flag will be cleared by hardware before ISR is executed.

11.1 External Interrupt Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	EX2G1	EX2G0	EX1G1	EX1G0	EX0G1	EX0G0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
IRCON2	-	-	-	-	-	IE2	TF3	ADCF

PEDGE Register (0X8F)

Bit	Field	Type	Initial	Description
5..4	EX2G[1:0]	R/W	10	External interrupt 2 trigger edge control register. 00: Reserved. 01: Rising edge trigger. 10: Falling edge trigger (default) 11: Both rising and falling edge trigger
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	

11.2 Sample Code

The following sample code demonstrates how to perform INT0/INT1/INT2 with interrupt.

```

1  #define INT0Rsing      (1 << 0) //INT0 trigger edge is rising edge
2  #define INT0Falling   (2 << 0) //INT0 trigger edge is falling edge
3  #define INT0LeChge    (3 << 0) //INT0 trigger edge is level chagne
4  #define EINT0         (1 << 0) //INT0 interrupt enable
5
6  #define INT1Rsing      (1 << 2) //INT1 trigger edge is rising edge
7  #define INT1Falling   (2 << 2) //INT1 trigger edge is falling edge
8  #define INT1LeChge    (3 << 2) //INT1 trigger edge is level chagne
9  #define EINT1         (1 << 2) //INT1 interrupt enable
10
11 #define INT2Rsing      (1 << 4) //INT2 trigger edge is rising edge
12 #define INT2Falling   (2 << 4) //INT2 trigger edge is falling edge
13 #define INT2LeChge    (3 << 4) //INT2 trigger edge is level chagne
14 #define EINT2         (1 << 2) //INT2 interrupt enable
15
16 void EnableINT(void)
17 {
18     // INT0 rising edge, INT1 falling edge, INT2 level change
19     PEDGE = INT0Rising | INT1Falling | INT2LeChge;
20
21     // Enable INT0/INT1 interrupt
22     IEN0 |= EINT0 | EINT1;
23     // Enable INT2 interrupt
24     IEN2 |= EINT2;
25     // Enable total interrupt
26     IEN0 |= 0x80;
27
28     P0 = 0x00;
29     POM = 0x38;
30 }
31
32 void INT0Interrupt(void) interrupt ISRInt0 //0x03
33 { //IE0 clear by hardware
34     P03 = ~P03;
35 }
36
37 void INT1Interrupt(void) interrupt ISRInt1 //0x13
38 { //IE1 clear by hardware
39     P04 = ~P04;
40 }
41
42 void INT2Interrupt(void) interrupt ISRInt2 //0x83
43 { //IE2 clear by hardware
44     P05 = ~P05;
45 }

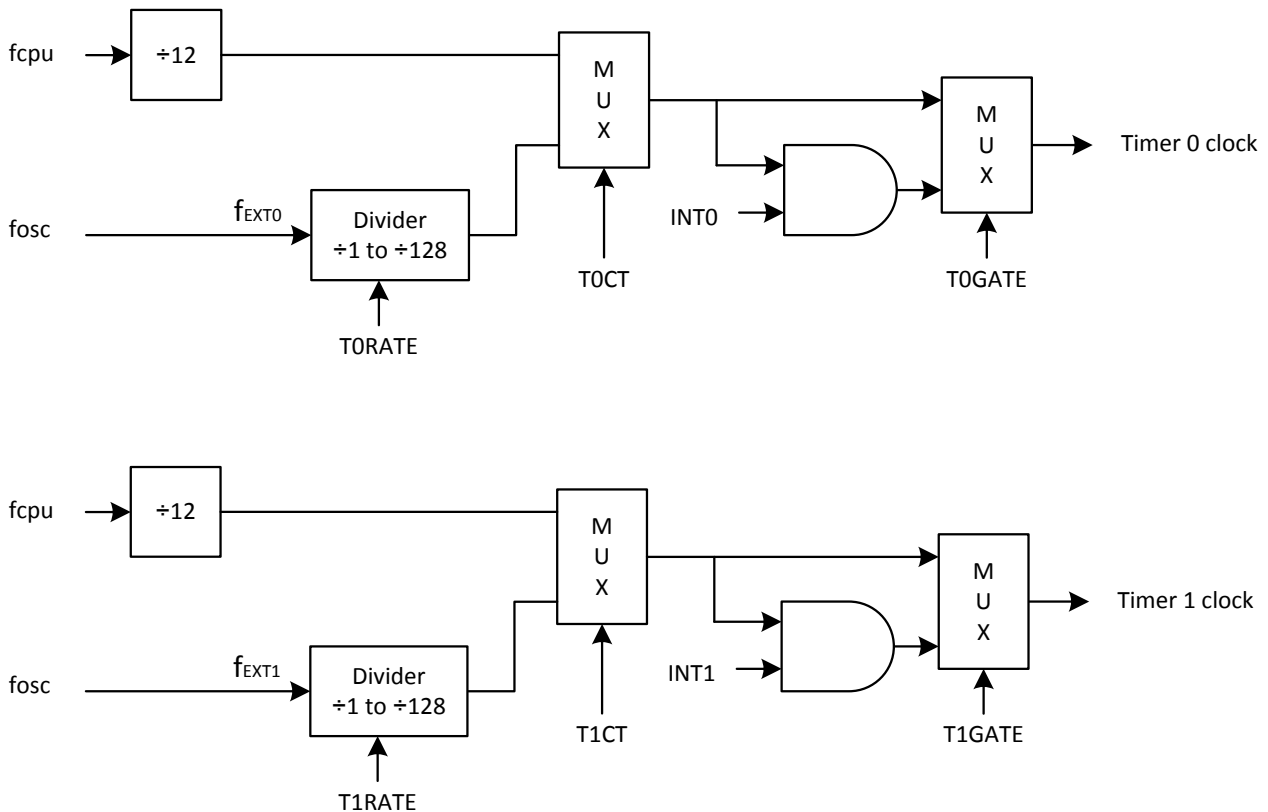
```

12 Timer 0 and Timer 1

Timer 0 and Timer 1 are two independent binary up timers. Timer 0 has four different operation modes: (1) 13-bit up counting timer, (2) 16-bit up counting timer, (3) 8-bit up counting timer with specified reload value support, and (4) separated two 8-bit up counting timer. By contrast, Timer 1 has only mode 0 to mode 2 which are same as Timer 0. Timer 0 and Timer 1 respectively support ET0 and ET1 interrupt function.

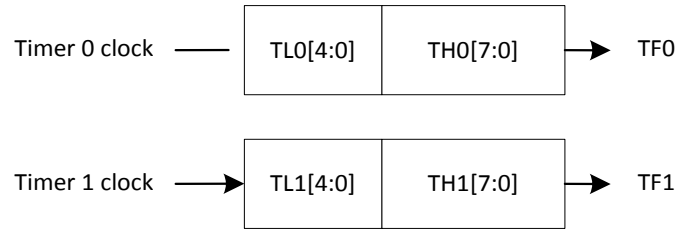
12.1 Timer 0 and Timer 1 Clock Selection

The figures below illustrate the clock selection circuit of Timer 0 and Timer 1. Timer 0 has two clock sources selection: fcpu and fosc. All clock sources can be gated (pause) by INTO pin if TOGATE is applied. Timer 1 clock sources selection: fcpu and fosc.



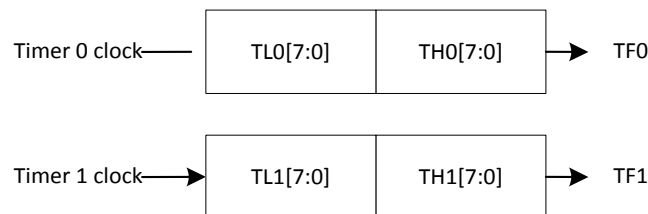
12.2 Mode 0: 13-bit Up Counting Timer

Mode 0 is a 13-bit up counting timer (the upper 3 bits of TL0 is suspended). Once the timer's counter is overflow (counts from 0xFF1F to 0x0000), TF0/TF1 flag would be issued immediately. This flag is readable and writable by firmware if ET0/ET1 does not apply, or can be handled by interrupt controller if ET0/ET1 is applied.



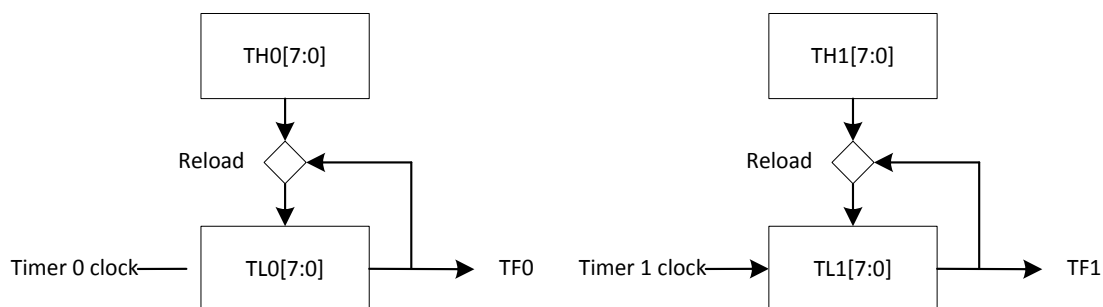
12.3 Mode 1: 16-bit Up Counting Timer

Mode 1 is a 16-bit up counting timer. Once the timer's counter overflow is occurred (from 0xFFFF to 0x0000), TF0/TF1 would be issued which is readable and writable by firmware or can be handled by interrupt controller (if ET0/ET1 applied).



12.4 Mode 2: 8-bit Up Counting Timer with Specified Reload Value Support

Mode 2 is an 8-bit up counting timer (TL0/TL1) with a specifiable reload value. An overflow event (TL0/TL1 counts from 0xFF to 0x00) issues its TF0/TF1 flag for firmware or interrupt controller; meanwhile, the timer duplicates TH0/TH1 value to TL0/TL1 register in the same time. As a result, the timer is actually counts from 0xFF to the value of TH0/TH1.

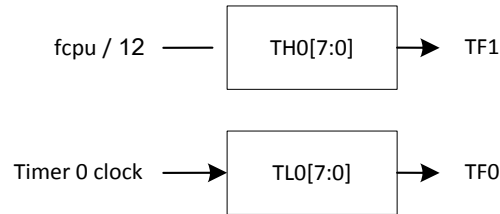


12.5 Mode 3 (Timer 0 only): Separated Two 8-bit Up Counting Timer

Mode 3 treats TH0 and TL0 as two separated 8-bit timers. TL0 is an 8-bit up counting timer with two clock sources selection (fcpu and fosc), whereas TH0 clock source is fixed at fcpu/12. Only TL0 clock source can be gated (pause) by INTO pin if TOGATE is applied.

In this mode TLO counter is enabled by TR0, and its overflow signal is reflected in TF0 flag. TH0 counter is controlled by TR1, and TF1 flag is also occupied by TH0 overflow signal.

Timer 1 cannot issue any overflow event in this situation, and it can be seen as a self-counting timer without flag support.



12.6 Timer 0 and Timer 1 Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
TCON0	-	TORATE2	TORATE1	TORATE0	-	T1RATE2	T1RATE1	T1RATE0
TMOD	T1GATE	T1CT	T1M1	T1M0	TOGATE	TOCT	TOM1	TOM0
TH0	TH07	TH06	TH05	TH04	TH03	TH02	TH01	TH00
TL0	TL07	TL06	TL05	TL04	TL03	TL02	TL01	TL00
TH1	TH17	TH16	TH15	TH14	TH13	TH12	TH11	TH10
TL1	TL17	TL16	TL15	TL14	TL13	TL12	TL11	TL10
IEN0	EAL	-	-	ESO	ET1	EX1	ETO	EXO

TCON Register (0x88)

Bit	Field	Type	Initial	Description
7	TF1	R/W	0	Timer 1 overflow event 0: Timer 1 does not have any overflow event 1: Timer 1 has overflowed This bit can be cleared automatically by interrupt handler, or manually by firmware
6	TR1	R/W	0	Timer 1 function 0: Disable 1: Enable
5	TF0	R/W	0	Timer 0 overflow event 0: Timer 0 does not have any overflow event 1: Timer 0 has overflowed This bit can be cleared automatically by interrupt handler, or manually by firmware
4	TRO	R/W	0	Timer 0 function 0: Disable 1: Enable
3	IE1	R/W	0	Refer to INT1
2	Reserved	R	0	
1	IE0	R/W	0	Refer to INTO
0	Reserved	R	0	

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
3	ET1	R/W	0	Timer 1 interrupt 0: Disable 1: Enable
1	ETO	R/W	0	Timer 0 interrupt 0: Disable 1: Enable

TCON0 Register (0xE7)

Bit	Field	Type	Initial	Description
7	Reserved	R	0	
6..4	TORATE[2:0]	R/W	000	Clock divider of Timer 0 external clock source 000: $f_{EXT0} / 128$ 001: $f_{EXT0} / 64$ 010: $f_{EXT0} / 32$ 011: $f_{EXT0} / 16$ 100: $f_{EXT0} / 8$ 101: $f_{EXT0} / 4$ 110: $f_{EXT0} / 2$ 111: $f_{EXT0} / 1$
3	Reserved	R	0	
2..0	T1RATE[2:0]	R/W	000	Clock divider of Timer 0 external clock source 000: $f_{EXT1} / 128$ 001: $f_{EXT1} / 64$ 010: $f_{EXT1} / 32$ 011: $f_{EXT1} / 16$ 100: $f_{EXT1} / 8$ 101: $f_{EXT1} / 4$ 110: $f_{EXT1} / 2$ 111: $f_{EXT1} / 1$

TH0 / TH1 Registers (TH0: 0x8C, TH1: 0x8D)

Bit	Field	Type	Initial	Description
7..0	TH0/TH1	R/W	0x00	High byte of Timer 0 and Timer 1 counter

TL0 / TL1 Register (TL0: 0x8A, TL1: 0x8B)

Bit	Field	Type	Initial	Description
7..0	TL0/TL1	R/W	0x00	Low byte of Timer 0 and Timer 1 counter

TMOD Register (0x89)

Bit	Field	Type	Initial	Description
7	T1GATE	R/W	0	Timer 1 gate control mode 0: Disable 1: Enable, Timer 1 clock source is gated by INT1
6	T1CT	R/W	0	Timer 1 clock source selection 0: $f_{\text{Timer1}} = f_{\text{cpu}} / 12$ 1: $f_{\text{Timer1}} = f_{\text{osc}} / \text{T1RATE}$ (refer to T1RATE) ^{*(1)}
5..4	T1M[1:0]	R/W	00	Timer 1 operation mode 00: 13-bit up counting timer 01: 16-bit up counting timer 10: 8-bit up counting timer with reload support 11: Reserved
3	TOGATE	R/W	0	Timer 0 gate control mode 0: Disable 1: Enable, Timer 0 clock source is gated by INTO
2	TOCT	R/W	0	Timer 0 clock source selection 0: $f_{\text{Timer0}} = f_{\text{cpu}} / 12$ 1: $f_{\text{Timer0}} = f_{\text{osc}} / \text{TORATE}$ (refer to TORATE) ^{*(2)}
1..0	T0M[1:0]	R/W	00	Timer 0 operation mode 00: 13-bit up counting timer 01: 16-bit up counting timer 10: 8-bit up counting timer with reload support 11: Separated two 8-bit up counting timer

*(1) $f_{\text{EXT1}} = f_{\text{osc}}$.

*(2) $f_{\text{EXT0}} = f_{\text{osc}}$.

12.7 Sample Code

The following sample code demonstrates how to perform T0/T1 with interrupt.

```

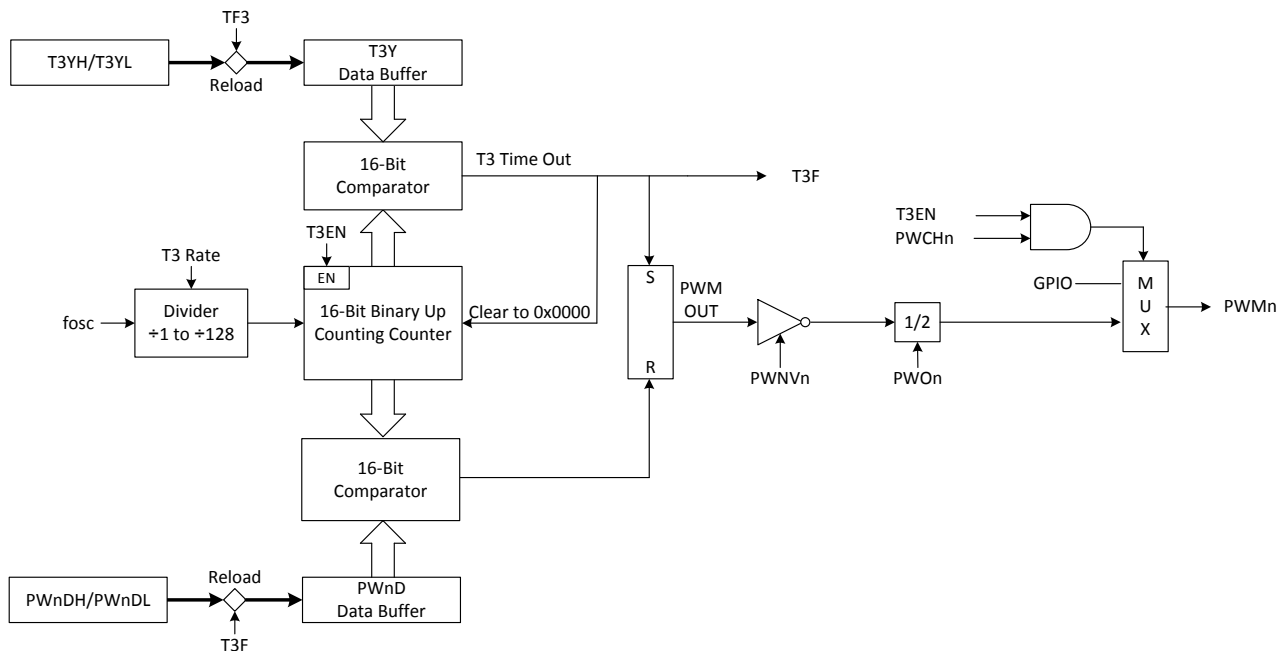
1  #define T0Mode0      (0 << 0) //T0 mode0, 13-bit counter
2  #define T0Mode1      (1 << 0) //T0 mode1, 16-bit counter
3  #define T0Mode2      (2 << 0) //T0 mode2, 8-bit auto-reload counter
4  #define T0Mode3      (3 << 0) //T0 mode3, T0 two 8-bit counter/T1 no flag
5  #define T0GATE       (8 << 0) //T0 gating clock by INT0
6  #define T0ClkFcpu    (0 << 0) //T0 clock source from Fcpu/12
7  #define T0ClkExt     (4 << 0) //T0 clock source from Fosc
8  #define T0ExtFosc    (0 << 4) //T0 clock source from Fosc
9
10 #define T1Mode0      (0 << 4) //T1 mode0, 13-bit counter
11 #define T1Mode1      (1 << 4) //T1 mode1, 16-bit counter
12 #define T1Mode2      (2 << 4) //T1 mode2, 8-bit auto-reload counter
13 #define T1Mode3      (3 << 4) //T1 mode3, T1 stop
14 #define T1GATE       (8 << 4) //T1 gating clock by INT1
15 #define T1ClkFcpu    (0 << 4) //T1 clock source from Fcpu/12
16 #define T1ClkExt     (4 << 4) //T1 clock source from Fosc
17
18 void InitT0T1(void)
19 {
20     // T0/T1_Initial
21     TH0 = 0x00;
22     TL0 = 0x00;
23     TH1 = 0x00;
24     TL1 = 0x00;
25     // T0 mode0 with gating clock by INT0, clock source from Fosc
26     TMOD |= T0Mode0 | T0GATE | T0ClkExt;
27     // T0 clock source = Fosc/1
28     TCON0 |= 0x70;
29     // T1 mode1, clock source from Fcpu/12
30     TMOD |= T1Mode1 | T1ClkFcpu;
31     // Timer 0/1 enable. Clear TF0/TF1
32     TCON |= 0x50;
33     // Enable T0/T1 interrupt
34     IEN0 |= 0x0A;
35     // Enable total interrupt
36     IEN0 |= 0x80;
37
38     P0 = 0x00;
39     POM = 0x03;
40 }
41
42 void T0Interrupt(void) interrupt ISRTimer0 //0x0B
43 { //TF0 clear by hardware
44     P00 = ~P00;
45 }
46 void T1Interrupt(void) interrupt ISRTimer1 //0x1B
47 { //TF1 clear by hardware
48     P01 = ~P01;
49 }

```

13 Timer3

Timer 3 is a 16-bit up counting timer and supports 6-channel general PWM function. By the counter reaches the up-boundary value (T3Y), it clears its counter and triggers an interrupt signal. PWM's duty cycle is controlled by PW0D~PW5D register. Each PWM channel has its own duty control.

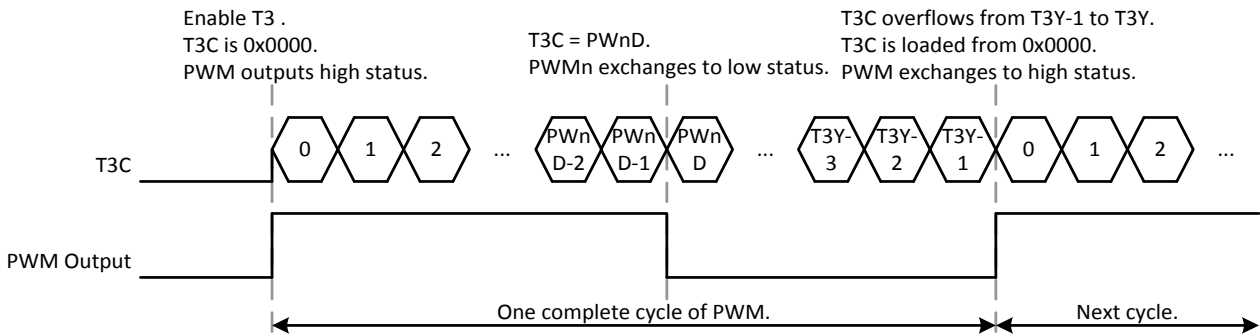
The PWM function has six programmable channels shared with GPIO pins and controlled by PWCH[5:0] bit. The output operation must be through enabled each bit/channel of PWCH[5:0] bits. The enabled PWM channel exchanges from GPIO to PWM output. When the PWCH[5:0] bits disables, the PWM channel returns to last status of GPIO mode. The Timer 3 build in IDLE Mode wake-up function if interrupt enable. When timer overflow occurs (counts from T3Y-1 to T3Y), T3F would be issued immediately which can read/write by firmware. T3 interrupt function is controlled by ET3.



13.1 General PWM

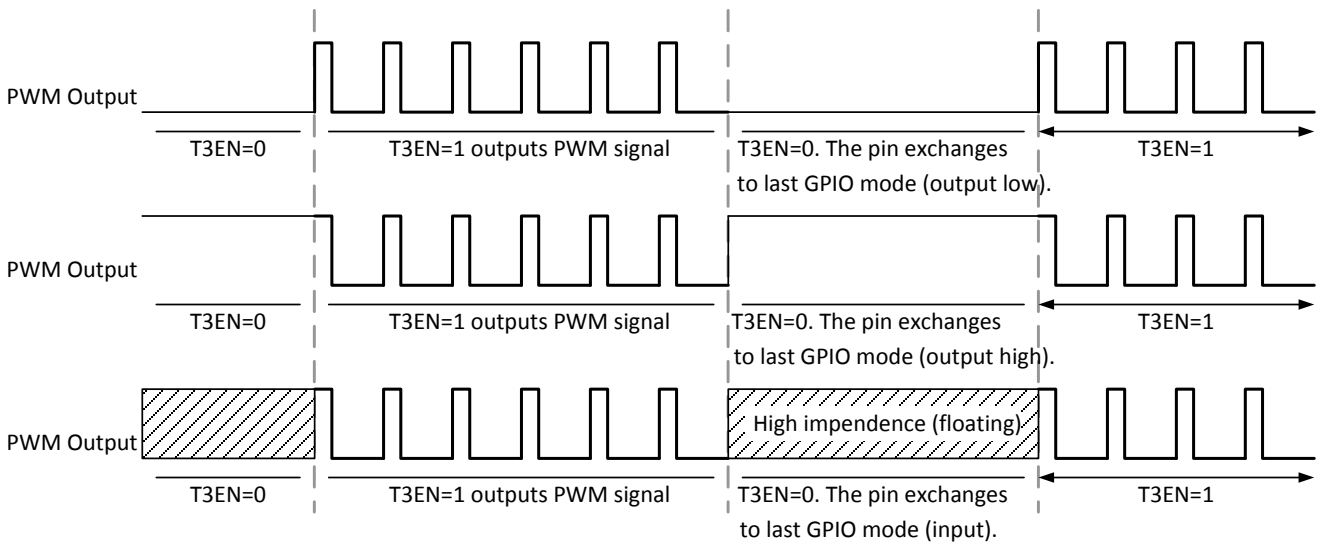
T3 timer builds in PWM function controlled by T3EN and PWCH[5:0] bits. PWM0 ~ PWM5 are output pins. Those output pins are shared with GPIO pin controlled by PWCH[5:0] bits. When output PWM function, we must be set T3EN =1. When PWM output signal synchronize finishes, the PWM channel exchanges from GPIO to PWM output. When T3EN = 0, the PWM channel returns to GPIO mode and last status. PWM signal is generated from the result of T3Y and PwND comparison combination. When T3C counts from 0x0000, the PWM outputs high status which is the PWM initial status. T3C is loaded new data from T3Y register to decide PWM cycle and

resolution. T3C keeps counting, and the system compares T3C and PWnD. When T3C=PWnD, the PWM output status exchanges to low and T3C keeps counting. When T3 timer overflow occurs (T3Y-1 to 0x0000), and one cycle of PWM signal finishes. T3C is reloaded from 0x0000 automatically, and PWM output status exchanges to high for next cycle. PWnD decides the high duty duration, and T3Y decides the resolution and cycle of PWM. PWnD can't be larger than T3Y, or the PWM signal is error. PWM clock source is fosc, T3RATE[2:0] bits: 000 = fosc/128, 001 = fosc/64, 010 = fosc/32, 011 = fosc/16, 100 = fosc/8, 101 = fosc/4, 110 = fosc/2, 111 = fosc/1.



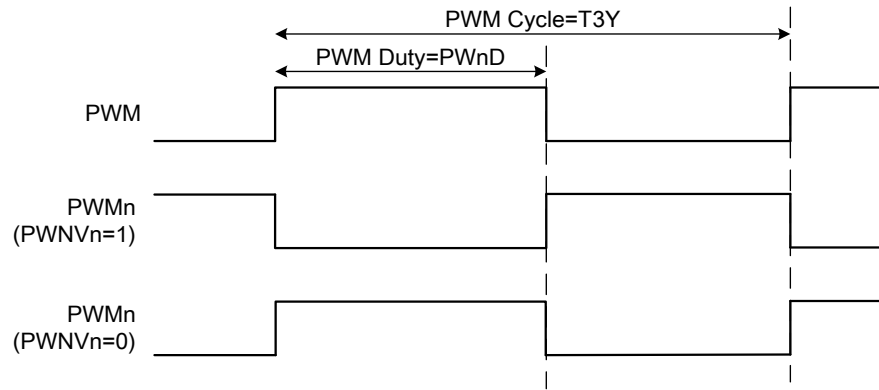
PWM Period = T3Y

PWM duty = (PWnD): (T3Y-PWnD)

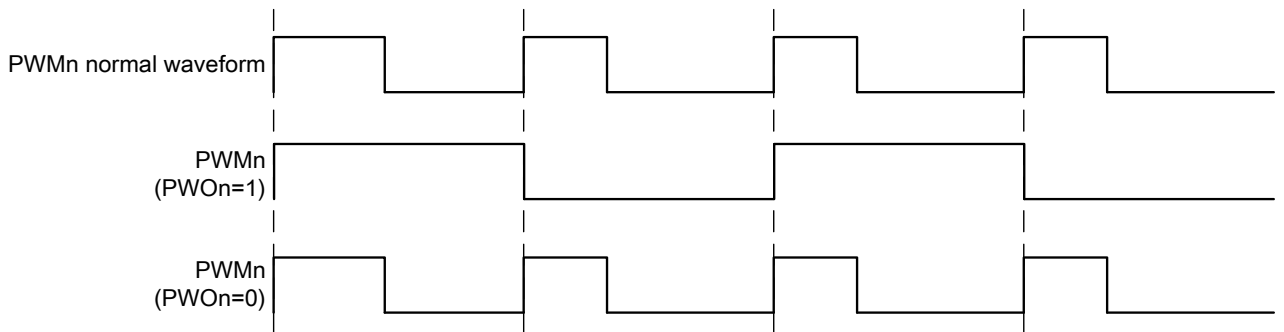


13.2 Inverse and Frequency Mode

The PWM builds in inverse output function. The PWM has one inverse PWM signal as PWNVn = 1. When PWNVn = 1, the PWMn outputs the inverse PWM signal of PWM. When PWNVn = 0, the PWMn outputs the non-inverse PWM signal of PWM. The inverse PWM output waveform is below diagram.



The PWM has frequency mode to change PWM output frequency. The frequency mode is controlled by $PWO[5:0]$. When $PWOn=1$, $PWMn$ pin outputs $1/2 * \text{frequency PWM}$ signal. When $PWOn=0$, $PWMn$ pin outputs $1 * \text{frequency PWM}$ signal.



13.3 PWM Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T3M	T3EN	T3rate2	T3rate1	T3rate0	-	-	-	-
PWNV	-	-	PWNV5	PWNV4	PWNV3	PWNV2	PWNV1	PWNV0
PWO	-	-	PWO5	PWO4	PWO3	PWO2	PWO1	PWO0
PWCH	-	-	PWCH5	PWCH4	PWCH3	PWCH2	PWCH1	PWCH0
T3CH	T3C15	T3C14	T3C13	T3C12	T3C11	T3C10	T3C9	T3C8
T3CL	T3C7	T3C6	T3C5	T3C4	T3C3	T3C2	T3C1	T3C0
T3YH	T3Y15	T3Y14	T3Y13	T3Y12	T3Y11	T3Y10	T3Y9	T3Y8
T3YL	T3Y7	T3Y6	T3Y5	T3Y4	T3Y3	T3Y2	T3Y1	T3Y0
PW0DH	PW0D15	PW0D14	PW0D13	PW0D12	PW0D11	PW0D10	PW0D9	PW0D8
PW0DL	PW0D7	PW0D6	PW0D5	PW0D4	PW0D3	PW0D2	PW0D1	PW0D0
PW1DH	PW1D15	PW1D14	PW1D13	PW1D12	PW1D11	PW1D10	PW1D9	PW1D8
PW1DL	PW1D7	PW1D6	PW1D5	PW1D4	PW1D3	PW1D2	PW1D1	PW1D0
PW2DH	PW2D15	PW2D14	PW2D13	PW2D12	PW2D11	PW2D10	PW2D9	PW2D8
PW2DL	PW2D7	PW2D6	PW2D5	PW2D4	PW2D3	PW2D2	PW2D1	PW2D0
PW3DH	PW3D15	PW3D14	PW3D13	PW3D12	PW3D11	PW3D10	PW3D9	PW3D8
PW3DL	PW3D7	PW3D6	PW3D5	PW3D4	PW3D3	PW3D2	PW3D1	PW3D0
PW4DH	PW4D15	PW4D14	PW4D13	PW4D12	PW4D11	PW4D10	PW4D9	PW4D8
PW4DL	PW4D7	PW4D6	PW4D5	PW4D4	PW4D3	PW4D2	PW4D1	PW4D0
PW5DH	PW5D15	PW5D14	PW5D13	PW5D12	PW5D11	PW5D10	PW5D9	PW5D8
PW5DL	PW5D7	PW5D6	PW5D5	PW5D4	PW5D3	PW5D2	PW5D1	PW5D0
IEN0	EAL	-	ET2	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF

T3 Registers (T3M: 0XA1)

Bit	Field	Type	Initial	Description
7	T3EN	R/W	0	T3 function 0: Disable 1: Enable*
6..4	T3RATE	R/W	000	T3 timer clock source 000: fosc / 128 001: fosc / 64 010: fosc / 32 011: fosc / 16 100: fosc / 8 101: fosc / 4 110: fosc / 2 111: fosc / 1
Else	Reserved	R	0	

* When the period is setting 0x0000, after T3 is set enable bit, the T3 will stop and the period can't update.

PWNV Register (0xBC)

Bit	Field	Type	Initial	Description
7..6	Reserved	R/W	0	
5..0	PWNV[5:0]	R/W	0	PWM5~PWM0 pin output control 0: Non-inverse. 1: Inverse

PWO Register (0xBD)

Bit	Field	Type	Initial	Description
7..6	Reserved	R/W	0	
5..0	PWO[5:0]	R/W	0	PWM5~PWM0 pin output control 0: 1*Frequency PWM signal. 1: 1/2 *Frequency PWM signal

PWCH Register (0xBE)

Bit	Field	Type	Initial	Description
7..6	Reserved	R/W	0	
5..0	PWCH[5:0]	R/W	0	PWM5~PWM0 shared-pin control 0: GPIO 1: PWM output (shared with P0.0~ P0.5)

T3CH/T3CL Registers (T3CH: 0xA3, T3CL: 0xA2)

Bit	Field	Type	Initial	Description
7..0	T3CH/L	R	0x00	16-bit Timer 3 counter.

T3YH/T3YL Registers (T3YH: 0xA5, T3YL: 0xA4)

Bit	Field	Type	Initial	Description
7..0	T3YH/L	R/W	0x00	16-bit Timer 3 period control*.

* The period configuration must be setup completely before starting Timer 3 function.

PWnDH/PWnDL Registers (PW0DH: 0xA7, PW0DL: 0xA6 / PW1DH: 0xAC, PW1DL: 0xAB / PW2DH: 0xAE, PW2DL: 0xAD / PW3DH: 0xB2, PW3DL: 0xB1 / PW4DH: 0xB4, PW4DL: 0xB3 / PW5DH: 0xB6, PW5DL: 0xB5)

Bit	Field	Type	Initial	Description
7..0	PWnDH/L	R/W	0x00	16-bit PWMn duty control

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)

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
1	ET3	R/W	0	T3 timer interrupt control bit. 0: Disable T3 interrupt function. 1: Enable T3 interrupt function.
Else				Refer to other chapter(s)

IRCON2 Register (0xBF)

Bit	Field	Type	Initial	Description
1	TF3F	R/W	0	T3 timer external reload interrupt request flag. 0: None T3 interrupt request 1: T3 interrupt request.
Else				Refer to other chapter(s)

13.4 Sample Code

The following sample code demonstrates how to perform PWM1 with interrupt.

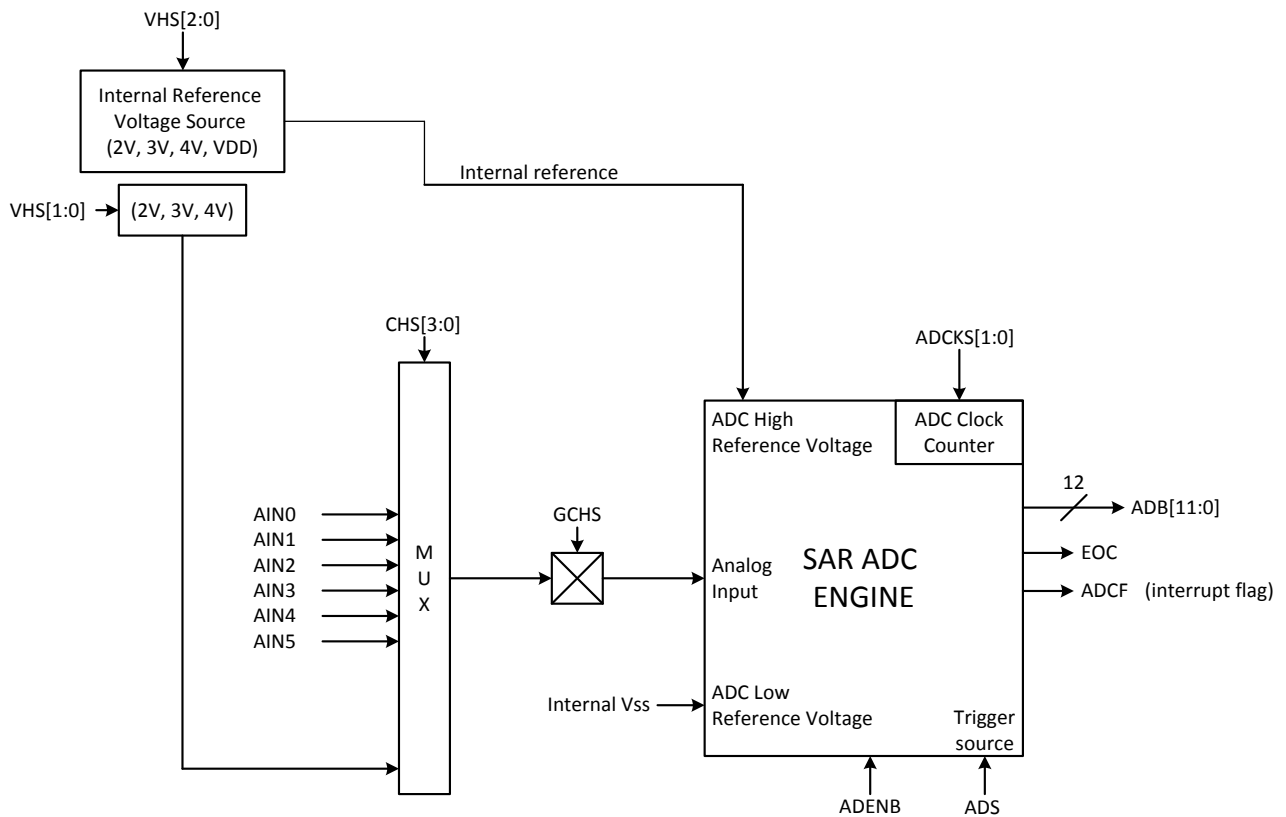
```

1  #define PWMInv0      (1 << 0) //PWM0 output inverse
2  #define PWMInv1      (1 << 1) //PWM1 output inverse
3  #define PWMInv2      (1 << 2) //PWM2 output inverse
4  #define PWMInv3      (1 << 3) //PWM3 output inverse
5  #define PWMInv4      (1 << 4) //PWM4 output inverse
6  #define PWMInv5      (1 << 5) //PWM5 output inverse
7  #define PWM2Frq0     (1 << 0) //PWM0 output 2*frequency PWM
8  #define PWM2Frq1     (1 << 1) //PWM1 output 2*frequency PWM
9  #define PWM2Frq2     (1 << 2) //PWM2 output 2*frequency PWM
10 #define PWM2Frq3     (1 << 3) //PWM3 output 2*frequency PWM
11 #define PWM2Frq4     (1 << 4) //PWM4 output 2*frequency PWM
12 #define PWM2Frq5     (1 << 5) //PWM5 output 2*frequency PWM
13 #define PWM0En       (1 << 0) //Enable PWM0 output function
14 #define PWM1En       (1 << 1) //Enable PWM1 output function
15 #define PWM2En       (1 << 2) //Enable PWM2 output function
16 #define PWM3En       (1 << 3) //Enable PWM3 output function
17 #define PWM4En       (1 << 4) //Enable PWM4 output function
18 #define PWM5En       (1 << 5) //Enable PWM5 output function
19 #define T3En         (1 << 7) //Enable T3 function
20
21 void InitT3(void)
22 {
23     // T3_Initial
24     T3YH = 0x80;
25     T3YL = 0x00;
26     PWLDH = 0x40;
27     PWLDL = 0x00;
28
29     // PWM1 channel enable
30     PWCH = PWM1En;
31
32     // T3 enable, PWM1 output inverse, clock = Fosc/32
33     T3M = T3En | PWMInv1 | 0x20;
34
35     // Enable T3 interrupt & clear T3F
36     IEN2 = 0x02;
37     IRCON2 = 0x00;
38
39     // Enable total interrupt
40     IEN0 |= 0x80;
41
42     P0 = 0x00;
43     POM |= 0x01;
44 }
45
46 void T3Interrupt(void) interrupt ISRT3 //0xEB
47 { //T3F clear by hardware
48     P00 = ~P00;
49 }

```

14 ADC

The analog to digital converter (ADC) is SAR structure with 6-input sources and up to 4096-step resolution to transfer analog signal into 12-bits digital buffers. The ADC builds in 6-channel input source to measure 10 different analog signal sources. The ADC resolution is 12-bit. The ADC has four clock rates to decide ADC converting rate. The ADC reference high voltage includes 4 sources. Four internal power source including VDD, 4V, 3V and 2V. The ADC builds in P0CON register to set pure analog input pin. After setup ADENB and ADS bits, the ADC starts to convert analog signal to digital data. ADC can work in idle mode. After ADC operating, the system would be waked up from green mode to normal mode if interrupt enable.



14.1 Configurations of Operation

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

1. Choose and enable the start of conversion ADC input channel. (By CHS[3:0] bits and GCHS bit)
2. The GPIO mode of ADC input channel must be set as input mode. (By PnM register)
3. The internal pull-up resistor of ADC input channel must be disabled. (By PnUR register)
4. The configuration control bit of ADC input channel must be set. (By PnCON register)
5. Choose ADC high reference voltage. (By VREFH register)
6. Choose ADC Clock Rate. (By ADCKS[1:0] bits)
7. After setup ADENB bits, the ADC ready to convert analog signal to digital data.

14.1.1 Start to Conversion

When ADC IP is enabled by ADENB bit, it is necessary to make a ADC start-up by program. Only ADS bit can start to convert analog signal. Conversions may be initiated by one of the following:

- Writing a 1 to the ADS bit of register ADM

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

14.2 ADC input channel

The ADC builds in 6-channel input source (AIN0 – AIN5) to measure 6 different analog signal sources controlled by CHS[3:0] and GCHS bits. The AIN8 is internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD, not internal 2V or 3V or 4V. AIN8 can be a good battery detector for battery system. To select appropriate internal AVREFH level and compare value, a high performance and cheaper low battery detector is built in the system.

ADC input channel

CHS[3:0]	Channel	Pin name	Remark
0000	AIN0	P0.0	
0001	AIN1	P0.1	
0010	AIN2	P0.2	
0011	AIN3	P0.3	
0100	AIN4	P0.4	
0101	AIN5	P0.5	
1000	AIN8	Internal 2V or 3V or 4V	Battery detector channel
1001 – 1111	-	-	Reserved

14.2.1 Pin Configuration

ADC input channels are shared with Port0. ADC channel selection is through CHS[3:0] bit. Only one pin of Port0 can be configured as ADC input in the same time. The pins of Port0 configured as ADC input channel must be set input mode, disable internal pull-up and enable P0CON first by program. After selecting ADC input channel through CHS[3:0], set GCHS bit as “1” to enable ADC channel function.

ADC input pins are shared with digital I/O pins. Connect an analog signal to COMS digital input pin, especially, the analog signal level is about 1/2 VDD will cause extra current leakage. In the power down mode, the above leakage current will be a big problem. Unfortunately, if users connect more than one analog input signal to Port0 will encounter above current leakage situation. Write “1” into PnCON register will configure related pin as pure analog input pin to avoid current leakage.

Note that When ADC pin is general I/O mode, the bit of P0CON must be set to “0”, or the digital I/O signal would be isolated.

14.3 Reference Voltage

The ADC builds in four high reference voltage source controlled through VREFH register. There are four internal voltage source (VDD, 4V, 3V, 2V). ADC reference high voltage is from internal voltage source selected by VHS[2:0] bits. If VHS2 is “1”, ADC reference high voltage is VDD. If VHS[1:0] is “10”, ADC reference high voltage is 4V. If VHS[1:0] is “01”, ADC reference high voltage is 3V. If VHS[1:0] is “00”, ADC reference high voltage is 2V. The limitation of internal high reference voltage application is VDD can't below each of internal high voltage level, or the level is equal to VDD. If AIN8 channel is selected as internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference high voltage must be internal VDD, not internal 2V/3V/4V.

14.3.1 Signal Format

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is V_{ss} and not changeable. The ADC high reference voltage includes internal $VDD/4V/3V/2V$. ADC reference voltage range limitation is “(ADC high reference voltage - low reference voltage) $\geq 2V$ ”. ADC low reference voltage is $V_{ss} = 0V$. So ADC high reference voltage range is $2V$ to VDD . The range is ADC external high reference voltage range.

- ADC Internal Low Reference Voltage = $0V$.
- ADC Internal High Reference Voltage = $VDD/4V/3V/2V$.

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

- $ADC\ Low\ Reference\ Voltage \leq ADC\ Sampled\ Input\ Voltage \leq ADC\ High\ Reference\ Voltage$

14.4 Converting Time

The ADC converting time is from $ADS=1$ (Start to ADC convert) to $EOC=1$ (End of ADC convert). The converting time duration is depend on ADC clock rate. 12-bit ADC’s converting time is $1 / (ADC\ clock / 4) * 16$ sec. ADC clock source is f_{osc} and includes $f_{osc}/1$, $f_{osc}/2$, $f_{osc}/8$ and $f_{osc}/16$ controlled by $ADCKS[1:0]$ bits.

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

$$12\ bits\ ADC\ conversion\ time = \frac{16}{ADC\ clock\ rate/4}$$

ADC converting time

ADCKS[1:0]	ADC clock rate	fosc = 16MHz		fosc = 32MHz	
		Converting time	Converting rate	Converting time	Converting rate
00	fosc/16	$1/(16MHz/16/4)*16$ = 64us	15.625kHz	$1/(32MHz/16/4)*16$ = 32us	31.25kHz
01	fosc/8	$1/(16MHz/8/4)*16$ = 32us	31.25kHz	$1/(32MHz/8/4)*16$ = 16us	62.5kHz
10	fosc	$1/(16MHz/4)*16$ = 4us	250kHz	$1/(32MHz/4)*16$ = 2us	500kHz

11	fosc/2	$1/(16\text{MHz}/2/4)*16$ = 8us	125kHz	$1/(32\text{MHz}/2/4)*16$ = 4us	250kHz
----	--------	------------------------------------	--------	------------------------------------	--------

14.5 Data Buffer

ADC data buffer is 12-bit length to store ADC converter result. The high byte is ADB register, and the low-nibble is ADR[3:0] bits. The ADB register is only 8-bit register including bit 4 – bit 11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC data buffer is a read-only register and the initial status is unknown after system reset.

Table 14-1 The AIN input voltage vs. ADB output data

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.
.
.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

14.6 ADC Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	-	CHS3	CHS2	CHS1	CHS0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
ADR	-	GCHS	ADCKS1	ADCKS0	ADB3	ADB2	ADB1	ADB0
VREFH	-	-	-	-	-	VHS2	VHS1	VHS0
POCON	-	-	POCON5	POCON4	POCON3	POCON2	POCON1	POCON0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF

ADM Register (0xD2)

Bit	Field	Type	Initial	Description
7	ADENB	R/W	0	ADC control bit. In stop mode, disable ADC to reduce power consumption. 0: Disable 1: Enable
6	ADS	R/W	0	ADC conversion control Write 1: Start ADC conversion (automatically cleared by the end of conversion)
5	EOC	R/W	0	ADC status bit. 0: ADC progressing 1: End of conversion (automatically set by hardware; manually cleared by firmware)
3..0	CHS[3:0]	R/W	0x00	ADC input channel select bit. 0000: AIN0, 0001: AIN1, 0010: AIN2, 0011: AIN3, 0100: AIN4, 0101: AIN5, 1000: AIN8 ^{*(1)} , others: Reserved.

*(1) The AIN8 is internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference voltage must be internal VDD and External voltage, not internal 2V or 3V or 4V.

ADB Register (0xD3)

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

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

ADR Register (0xD4)

Bit	Field	Type	Initial	Description
6	GCHS	R/W	0	ADC global channel select bit. 0: Disable AIN channel. 1: Enable AIN channel.
5..4	ADCKS[1:0]	R/W	00	ADC's clock source select bit. 00 = fosc/16, 01 = fosc/8, 10 = fosc/1, 11 = fosc/2
3..0	ADB[3:0]	R	-	ADC Result Bit [3:0] [*] in 12-bit ADC resolution mode.

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

VREFH Register (0xD5)

Bit	Field	Type	Initial	Description
2..0	VHS[2:0]	R/W	00	ADC internal reference high voltage selects bits. ^{*(1)} 000: VREFH = 2.0V. 001: VREFH = 3.0V. 010: VREFH = 4.0V. 100: VREFH = VDD. 100: VREFH = VDD and AIN8 = 4.0V. 101: VREFH = VDD and AIN8 = 3.0V. 110: VREFH = VDD and AIN8 = 2.0V. Others: Reserved.

*(1) If AIN8 channel is selected as internal 2V or 3V or 4V input channel. There is no any input pin from outside. In this time ADC reference high voltage must be internal VDD, not internal 2V/3V/4V.

POCON Register (0x9E)

Bit	Field	Type	Initial	Description
5..0	POCON[5:0]	R/W	0x00	P0 configuration control bit *. 0: P0 can be analog input pin (ADC input pin) or digital GPIO pin. 1: P0 is pure analog input pin and can't be a digital GPIO pin.
Else	Reserved	R	0	

* POCAN [5:0] will configure related Port0 pin as pure analog input pin to avoid current leakage.

IEN2 Register (0x9A)

Bit	Field	Type	Initial	Description
0	EADC	R/W	0	ADC interrupt control bit. 0: Disable ADC interrupt function. 1: Enable ADC interrupt function.
Else				Refer to other chapter(s)

IRCON2 Register (0xBF)

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

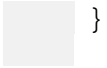
14.7 Sample Code

The following sample code demonstrates how to perform ADC to convert AIN5 with interrupt.

```

1  #define ADCAIN8-4V      (2 << 0) //AIN8 = 4.0V
2  #define ADCAIN8-3V      (1 << 0) //AIN8 = 3.0V
3  #define ADCAIN8-2V      (0 << 0) //AIN8 = 2.0V
4  #define ADCInRefVDD     (1 << 2) //internal reference from VDD
5  #define ADCSpeedDiv16   (0 << 4) //ADC clock = fosc/16
6  #define ADCSpeedDiv8    (1 << 4) //ADC clock = fosc/8
7  #define ADCSpeedDiv1    (2 << 4) //ADC clock = fosc/1
8  #define ADCSpeedDiv2    (3 << 4) //ADC clock = fosc/2
9  #define ADCChannelEn    (1 << 6) //enable ADC channel
10 #define SelAIN5         (5 << 0) //select ADC channel 5
11 #define ADCStart        (1 << 6) //start ADC conversion
12 #define ADCEn           (1 << 7) //enable ADC
13 #define EADC            (1 << 1) //enable ADC interrupt
14 #define ClearEOC        0xDF;
15
16 unsigned int  ADCBuffer;    // data buffer
17
18 void ADCInit(void)
19 {
20     P0 = 0x00;
21     P0M = 0x10;
22
23     // set AIN5 pin's mode at pure analog pin
24     P0CON |= 0x20;    //AIN5/P05
25     P0M  &= 0xDF;    //input mode
26     P0UR  &= 0xDF;    //disable pull-high
27
28     // configure ADC channel and enable ADC.
29     ADM = ADCEn | SelAIN5;
30
31     // enable channel and select conversion speed
32     ADR = ADCChannelEn | ADCSpeedDiv1;
33
34     // configure reference voltage
35     VREFH = ADCInRefVDD;
36
37     // enable gobal interrupt
38     IEN0 |= 0x80;    //enable global interrupt
39
40     // enable ADC interrupt
41     IEN2 |= EADC;
42
43     // start ADC conversion
44     ADM |= ADCStart;
45 }
46
47 void ADCInterrupt(void) interrupt ISRAdc
48 { //ADCF clear by hardware
49     P04 = ~P04;
50
51     ADCBuffer = (ADB << 4) + (ADR & 0x0F);
52     ADM &= ClearEOC;
53     ADM |= ADCStart;

```



15 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.

15.1 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.

15.2 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, or Timer 1 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.



15.3 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 Timer 1 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.



15.4 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 SORELH/SORELL registers (BD = 1) or Timer 1 overflow period (BD = 0). The SMOD bit doubles the frequency from the generator.

If the SORELH/SORELL is selected (BD = 1) in mode 1 and 3, the baud rate is generated as following equation.

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

Table 15-1 Recommended Setting for Common UART Baud Rates ($f_{cpu} = 32 \text{ MHz}$)

Baud Rate	SMOD	SORELH	SORELL	Accuracy
4800	0	0x03	0x98	-0.16 %
9600	0	0x03	0xCC	-0.16 %
19200	0	0x03	0xE6	-0.16 %
38400	0	0x03	0xF3	-0.16 %
56000	1	0x03	0xEE	0.79 %
57600	1	0x03	0xEF	-2.12 %
115200	1	0x03	0xF7	3.55 %
128000	1	0x03	0xF8	2.34 %
250000	1	0x03	0xFC	0 %
500000	1	0x03	0xFE	0 %
1000000	1	0x03	0xFF	0 %

If the Timer 1 overflow period is selected (BD = 0) in mode 1 and 3, the baud rate is generated as following equation. The Timer 1 must be in 8-bit auto-reload mode which can generate periodically

overflow signals.

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

Table 15-2 Recommended Setting T1 overflow period (T1 clock=32M) for Common UART Baud Rates (fcpu = 32 MHz)

Baud Rate	SMOD	Timer Period	TH1/TL1	Accuracy
4800	0	6.510 us	0x30	0.16 %
9600	1	6.510 us	0x30	0.16 %
19200	1	3.255 us	0x98	0.16 %
38400	1	1.628 us	0xCC	0.16 %
56000	1	1.116 us	0xDC	-0.80 %
57600	1	1.085 us	0xDD	-0.80 %
115200	1	0.543 us	0xEF	2.08 %
128000	1	0.488 us	0xF0	-2.40 %
250000	1	0.250 us	0xF8	0 %
500000	1	0.125 us	0xFC	0 %
1000000	1	0.063 us	0xFE	0 %

When clock generator source is T1 overflow rate, the system clock Fcpu minimum setting value follow $F_{cpu} \leq \text{Timer Period Value}$.

15.5 Power Saving

The UART module has clock gating function for saving power. When RENO bit is 0, the UART module internal clocks are halted to reduce power consumption. UART relevant register (SOCON, SOCON2, SOBUF, SORELL, SORELH and SMOD bit) are unable to access.

Conversely, when RENO bit is 1, UART internal clocks are run, and registers can access. The RENO bit must be set to 1, before the initial setting UART.

15.6 UART Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SOCON	SM0	SM1	SM20	RENO	TB80	RB80	TIO	RIO
SOCON2	BD	-	-	-	-	-	-	-
SOBUF	SOBUF7	SOBUF6	SOBUF5	SOBUF4	SOBUF3	SOBUF2	SOBUF1	SOBUF0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
SORELH	-	-	-	-	-	-	SOREL9	SOREL8
SORELL	SOREL7	SOREL6	SOREL5	SOREL4	SOREL3	SOREL2	SOREL1	RORELO
IENO	EAL	-	-	ESO	ET1	EX1	ETO	EXO

P0OC	-	-	P05OC	P04OC	P03OC	P02OC	P01OC	P00OC
P0M	-	-	P05M	P04M	P03M	P02M	P01M	P00M
P0	-	-	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	RENO	R/W	0	UART module (and reception function) 0: Disable for power saving* 1: Enable for UART operating
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

* When RENO bit is 0, UART relevant register are unable to access, and the module internal clocks are halted.

SOCON2 Register (0xD8)

Bit	Field	Type	Initial	Description
7	BD	R/W	0	Baud rate generators selection (mode 1, 3) 0: Timer 1 overflow period 1: Controlled by SORELH, SORELL registers
6..0	Reserved	R	0x00	

SOBUF Register (0x99)

Bit	Field	Type	Initial	Description
7..0	SOBUF	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 0, 2) 0: fcpu/64 1: fcpu/32
6..0				Refer to other chapter(s)

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	Interrupts enable. Refer to Chapter Interrupt
4	ES0	R/W	0	Enable UART interrupt
Else				Refer to other chapter(s)

P0OC Register (0xE4)

Bit	Field	Type	Initial	Description
3	P03OC	R/W	0	0: Switch P0.3 (UTX) to push-pull mode 1: Switch P0.3 (UTX) to open-drain mode
2	P02OC	R/W	0	0: Switch P0.2 (URX) to input mode (required) 1: Switch P0.2 (URX) to open-drain mode*
Else				Refer to other chapter(s)

* Setting P02OC as high causes URX cannot receive data.

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
3	P03M	R/W	0	0: Set P0.3 (UTX) as input mode* 1: Set P0.3 (UTX) as output mode (required)
2	P02M	R/W	0	0: Set P0.2 (URX) as input mode (required) 1: Set P0.2 (URX) as output mode*
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
3	P03	R/W	0	0: Set P0.3 (UTX) always low* 1: Make P0.3 (UTX) can output UART data (required)
2	P02	R/W	0	This bit is available to read at any time for monitoring the bus statue.
Else				Refer to other chapter(s)

* Setting P03 initially high because UART block drive the shared pin low signal only.

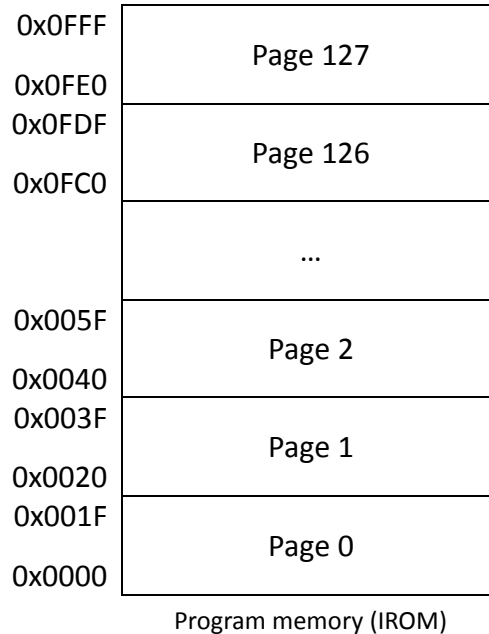
15.7 Sample Code

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     P03 = 1;
13     P0M = P0M | 0x08 & ~0x04;
14
15     // configure UART mode between SM0 and SM3, enable URX
16     S0CON = SYSUartSM1 | SYSUartREN;
17
18     // configure UART baud rate
19     PCON = SYSUartSMODE1;
20     S0CON2 = SYSUartBD1;
21     SORELH = 0x03;
22     SORELL = 0xFE;
23
24     // enable UART interrupt
25     IEN0 |= SYSUartES0;
26
27     // send first UTX data
28     SOBUF = uartTxBuf;
29 }
30
31 void SYSUartInterrupt(void) interrupt ISRUart
32 {
33     if (TI0 == 1) {
34         SOBUF = uartTxBuf;
35         TI0 = 0;
36     } else if (RI0 == 1) {
37         uartRxBuf = SOBUF;
38         RI0 = 0;
39     }
40 }
```

16 In-System Program

SN8F5701 builds in an on-chip 4 KB program memory, aka IROM, which is equally divided to 128 pages (32 bytes per page). The in-system program is a procedure that enables a firmware to freely modify every page's data; in other word, it is the channel to store value(s) into the non-volatile memory and/or live update firmware.



16.1 Page Program

Because each page of the program memory has 32 bytes in length, a page program procedure requires 32 bytes IRAM as its data buffer.

These configurations must be setup completely before starting Page Program. ISP is configured using the following steps:

1. Save program data into IRAM. The data continues for 32 bytes.
2. Set the start address of the content location to PERAM.
3. Set the start address of the anticipated update area to PEROM [15:5]. (By PEROMH/PRROML registers)
4. Write '0x5A' into PECMD [7:0] to trigger ISP function.

As an example, assume the 126th page of program memory (IROM, 0x0FC0 – 0x0FDF) is the anticipated update area; the content is already stored in IRAM address 0x60 – 0x7F. To perform the in-system program, simply write starting IROM address 0x0FC0 to EPROMH/EPROML registers, and then specify buffer starting address 0x60 to EPRAM register. Subsequently, write '0x5A' into PECMD [7:0] registers to duplicate the buffer's data to 126th 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 127) 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.

16.2 Byte Program

Byte program supports one byte memory program, one byte program procedure requires 1 byte IRAM as its data buffer.

These configurations must be setup completely before starting Byte Program. ISP is configured using the following steps:

1. Save program data into IRAM. The data only for 1 byte.
2. Set the start address of the content location to PERAM.
3. Set the start address of the anticipated update area to PEROM [15:0]. (By PEROMH/PRROML registers)
4. Write '0x1E' into PECMD [7:0] to trigger ISP function.

As an example, assume the address 0x0FC5 of IRPM is the anticipated update area; the content is already stored in IRAM address 0x60. To perform the in-system byte program, simply write starting IROM address 0x0FC5 to EPROMH/EPROML registers, and then specify buffer starting address 0x60 to EPROM register. Subsequently, write '0x1E' into PECMD [7:0] registers to duplicate the buffer's data to the address 0x0FC5 of IROM.

16.3 In-system Program Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAM	PERAM7	PERAM6	PERAM5	PERAM4	PERAM3	PERAM2	PERAM1	PERAM0
PEROMH	PEROM15	PEROM14	PEROM13	PEROM12	PEROM11	PEROM10	PEROM9	PEROM8
PEROML	PEROM7	PEROM6	PEROM5	PEROM4	PEROM3	PEROM2	PEROM1	PEROM0
PECMD	PECMD7	PECMD6	PECMD5	PECMD4	PECMD3	PECMD2	PECMD1	PECMD0

PERAM Register (0x97)

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

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 (IROM)

PEROML Register (0x95)

Bit	Field	Type	Initial	Description
7..0	PEROM[7:0]	R/W	000	The first address (7 th – 0 th) of program page (IROM)

PECMD Register (0x94)

Bit	Field	Type	Initial	Description
7..0	PECMD[7:0]	W	0x0	0x5A: Start page program procedure 0x1E: Start byte program procedure Else values: Reserved ^{*(1)}

*(1) Not permitted to write any other to PECMD register.

16.4 Sample Code

```
1 unsigned char idata dataBuffer[32] _at_ 0xE0; // IRAM 0xE0 to 0xFF
2
3 void SYSIspSetDataBuffer(unsigned char address, unsigned char data)
4 {
5     dataBuffer[address & 0x1F] = data;
6 }
7
8 void SYSIspStart(unsigned int pageAddress)
9 {
10    ISP(pageAddress, 0xE0); //Page program
11 }
12
13
14 void SYSByteIspStart(unsigned int byteAddress)
15 {
16    BISP(byteAddress, 0xE0); //Byte program
17 }
```

17 Clock Fine-Tuning

SN8F5701 builds in clock fine-tuning function that is a procedure to fine-tune system clock frequency by firmware. The function is enabled by code option (CK_Fine_Tuning). When CK_Fine_Tuning = 0, the clock fine-tuning function is disabled. When CK_Fine_Tuning = 1, the clock fine-tuning function is enabled. After system power-on, the 10-bit initial clock trim value will be loaded to FRQ[9:0] buffer by hardware. The trim value corresponds to IHRC 32MHz. Change the trim value of FRQ[9:0] to modify internal clock frequency.

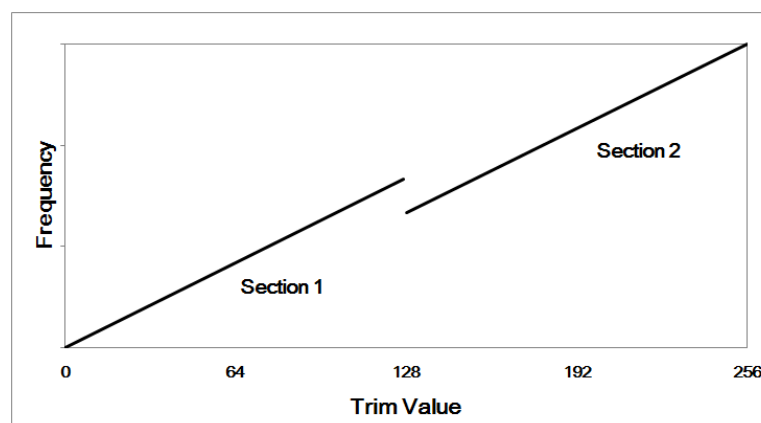
17.1 Clock Trim Section

Clock fine-tuning consists of 8 trim sections as Table 17-1. Each section includes 128 trim steps and each step is about (32MHz *0.1%) in the same section. The larger the Trim value, the faster the frequency.

Table 177-1 Clock Trim Section

Section	Trim Value	Frequency
1	000H ~ 07FH	Low
2	080H ~ 0FFH	
3	100H ~ 17FH	
4	180H ~ 1FFH	
5	200H ~ 27FH	
6	280H ~ 2FFH	
7	300H ~ 37FH	
8	380H ~ 3FFH	High

Each adjacent section has a frequency gap as below. Thus the frequency in trim value =127 will faster than trim value =128.



17.2 Clock Fine-Tuning Procedure

These configurations must be setup completely before starting clock fine-tuning. The steps as the following:

1. Select code option CK_Fine_Tuning = 1 to enable clock fine-tuning function.
2. As the Max. IROM fetching cycle is 8MHz, it is recommended to set PWSC[2:0]=7 at first to avoid system error.
3. Read 10-bit 32MHz trim value from FRQ[9:0]
4. Check the fine-tuning range from the Table17-1.
5. Write the new clock trim value to FRQ[9:0].
6. Write '0x3C' into FRQCMD [7:0] to trigger clock fine-tuning function.

17.3 Clock Fine-Tuning Register

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FRQH	-	-	-	-	-	-	FRQ9	FRQ8
FRQL	FRQ7	FRQ6	FRQ5	FRQ4	FRQ3	FRQ2	FRQ1	FRQ0
FRQCMD	FRQCMD7	FRQCMD6	FRQCMD5	FRQCMD4	FRQCMD3	FRQCMD2	FRQCMD1	FRQCMD0

FRQ Register (FRQH:0xF3, FRQL:0xF2)

Bit	Field	Type	Initial	Description
9..0	FRQ[9:0]	R/W	0x00	The system clock calibration value

FRQCMD Register (0xF4)

Bit	Field	Type	Initial	Description
7..0	FRQCMD[7:0]	W	0x00	0x3C: Start clock fine-tuning procedure Else values: Reserved ^{*(1)}

*(1) Not permitted to write any other to FRQCMD register.

17.4 Sample Code

The following sample code demonstrates how to perform clock fine-tuning.

```
1  FRQH = 0x01;
2  FRQL = 0x59;    // Set FRQ[9:0]= 0x159
3  FRQCMD = 0x3C; // Apply CLKSEL's setting
```

18 Electrical Characteristics

18.1 Absolute Maximum Ratings

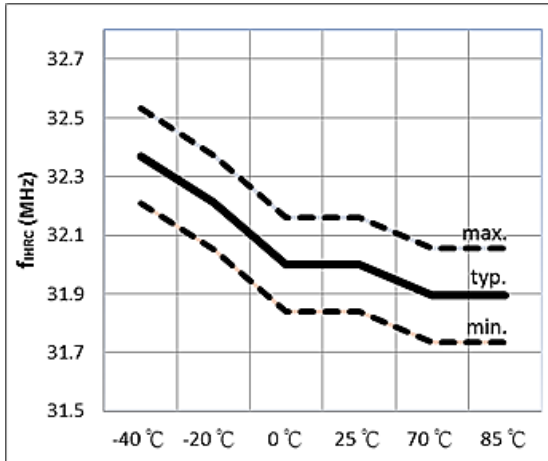
Voltage applied at VDD to VSS	- 0.3V to 6.0V
Voltage applied at any pin to VSS.....	- 0.3V to VDD+0.3V
Operating ambient temperature.....	-40°C to 85°C
Storage ambient temperature	-40°C to 125°C
Junction Temperature	-40°C to 125°C

18.2 System Operation Characteristics

	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	Operating voltage	fcpu = 1MHz	1.8		5.5	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, fcpu = 1MHz		2.25		mA
		VDD = 5V, fcpu = 1MHz		2.30		mA
		VDD = 3V, fcpu = 4MHz		2.60		mA
		VDD = 5V, fcpu = 4MHz		2.65		mA
		VDD = 3V, fcpu = 8MHz		3.10		mA
		VDD = 5V, fcpu = 8MHz		3.15		mA
		VDD = 3V, fcpu = 32MHz		4.10		mA
		VDD = 5V, fcpu = 32MHz		4.15		mA
I _{DD2}	STOP mode supply current	VDD = 3V		3.0		μA
		VDD = 5V		3.5		μA
I _{DD3}	IDLE mode supply current (fcpu = 1MHz)	VDD = 3V, 32MHz IHRC		0.55		mA
		VDD = 5V, 32MHz IHRC		0.60		mA
F _{IHRC}	Internal high clock generator	VDD = 1.8V to 5.5V, 25°C	31.84	32	32.16	MHz
		VDD = 1.8V to 5.5V, 25°C to 85°C	31.68		31.99	MHz
		VDD = 1.8V to 5.5V, -40°C to 25°C	32.31		32.64	MHz
F _{ILRC}	Internal low clock generator	VDD = 5V, 25°C	12	16	24	kHz
V _{LVD18}	LVD18 detect voltage	25°C	1.7	1.8	1.9	V
		-40°C to 85°C	1.6	1.8	2.0	V

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

- IHRC Frequency - Temperature Graph



18.3 GPIO Characteristics

Parameter	Test Condition	MIN	TYP	MAX	UNIT	
V_{IL1}	Low-level input voltage (P00)	VSS	0.1VDD		V	
V_{IH1}	High-level input voltage (P00)	0.5VDD		VDD	V	
V_{IL2}	Low-level input voltage (P01- P05)	VSS	0.3VDD		V	
V_{IH2}	High-level input voltage (P01- P05)	0.7VDD		VDD	V	
I_{LEKG}	I/O port input leakage current	$V_{IN} = VDD$		2	μA	
R_{UP}	Pull-up resister	$VDD = 3V$	100	200	300	$k\Omega$
		$VDD = 5V$	50	100	150	$k\Omega$
I_{OH}	I/O output source current	$VDD = 5V, V_O = VDD-0.5V$		12	16	mA
I_{OL}	I/O sink current	$VDD = 5V, V_O = VSS+0.5V$		15	20	mA

* Ambient temperature is 25°C.

18.4 ADC Characteristics

	Parameter	Test Condition	MIN	TYP	MAX	UNIT
V_{ADC}	Operating voltage		2.0		5.5	V
V_{AIN}	AIN channels input voltage	VDD = 5V	0		V_{REFH}	V
V_{REFH}	AVREFH pin input voltage	VDD = 5V	2		VDD	V
V_{IREF}	Internal VDD reference voltage	VDD = 5V		VDD		V
	Internal 4V reference voltage	VDD = 5V	3.92	4	4.08	V
	Internal 3V reference voltage	VDD = 5V	2.94	3	3.06	V
	Internal 2V reference voltage	VDD = 5V	1.96	2	2.04	V
I_{AD}	ADC current consumption	VDD = 3V		0.45		mA
		VDD = 5V		0.50		mA
f_{ADCLK}	ADC clock	VDD = 5V			32	MHz
f_{ADSMP}	ADC sampling rate	VDD = 5V			500	kHz
t_{ADEN}	ADC function enable period	VDD = 5V	100			μ s
DNL	Differential nonlinearity*	$f_{ADSMP} = 62.5\text{kHz}$		± 1		LSB
		$f_{ADSMP} = 250\text{kHz}$		± 1		LSB
		$f_{ADSMP} = 500\text{kHz}$		± 8		LSB
INL	Integral Nonlinearity*	$f_{ADSMP} = 62.5\text{kHz}$		± 2		LSB
		$f_{ADSMP} = 250\text{kHz}$		± 2.5		LSB
		$f_{ADSMP} = 500\text{kHz}$		± 8		LSB
NMC	No missing code*	$f_{ADSMP} = 62.5\text{kHz}$	10	11	12	Bit
		$f_{ADSMP} = 250\text{kHz}$		11		Bit
		$f_{ADSMP} = 500\text{kHz}$		8		Bit
V_{OFFSET}	Input offset voltage	Non-trimmed	-10	0	10	mV
		Trimmed	-2	0	2	mV

* Parameters with star mark: VDD = 3V, $V_{REFH} = 3V$, 25°C.

18.5 Flash Memory Characteristics

	Parameter	Test Condition	MIN	TYP	MAX	UNIT
V_{dd}	Supply voltage		1.8		5.5	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=32 bytes, 25°C		6	8	ms

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

19 Instruction Set

This chapter categorizes the SN8F5701 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.

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

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

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

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

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

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

20 Debug Interface

The debug interface, aka SWAT, shares one pin with GPIO P0.4 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.

20.1 Minimum Requirement

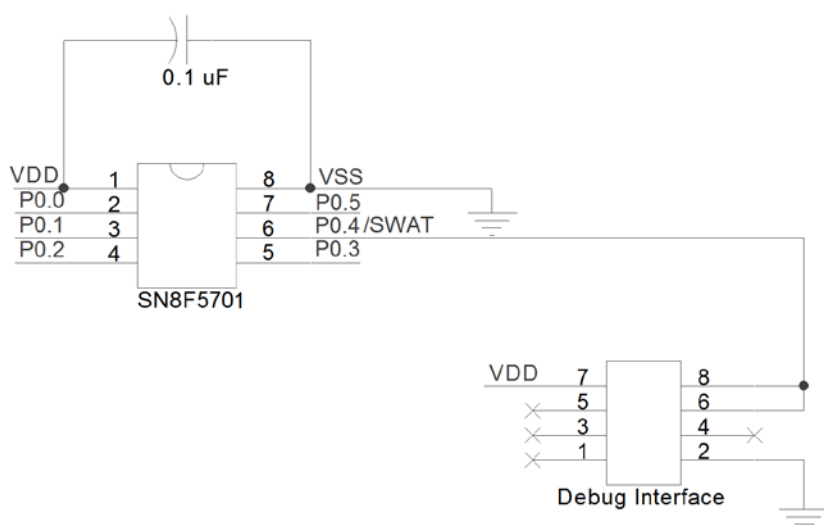
The following items are essential to build up an appropriate development environment. The compatibility is verified on listed versions, and is expected to execute perfectly on later version. SN-Link related information is available to download on SONiX website (www.sonix.com.tw); Keil C51 is downloadable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

- **SN-Link Adapter II** with updated firmware version 3.2.5
- **SN-Link Driver for Keil C51** version 1.00.35
- **Keil C51** version 9.50a and 9.54a

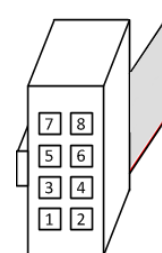
20.2 Debug Interface Hardware

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 5th and 7th pins of SN-Link, and respectively link VDD and VSS to 8th pin and 1st 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



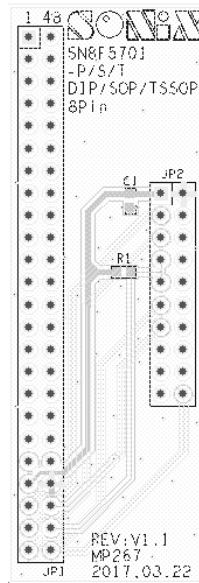
SN-Link header

21 ROM Programming Pin

SN8F5701 Series Flash ROM erase/program/verify support SN-LINK and MP5 Writer

- SN-LINK: Debug interface.
- MP5 Writer: For SN8F5701 series version.

21.1 MP5 Writer Transition Board Socket Pin Assignment



JP7 (Mapping to 48-pin text tool):

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41
DIP9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

Writer JP5/JP6:

VDD	1	2	GND
DFTCLK	3	4	-
SEL	5	6	-
SWAT	7	8	-
SWAT	9	10	-
DAH	11	12	-
DAL	13	14	-
-	15	16	-
-	17	18	-
-	19	20	PDB

JP5 for Writer transition board

JP6 for MCU <48 pin package

21.2 MP5 Writer Programming Pin Mapping

SN8F5701P/S/T (DIP8/SOP8/TSSOP8):

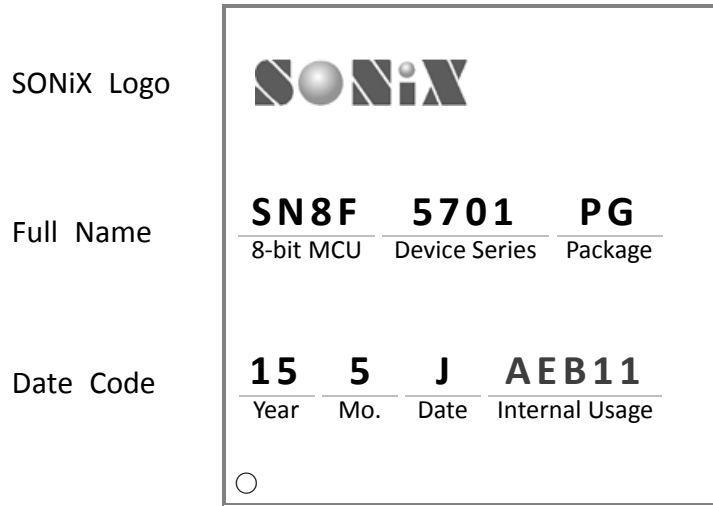
Writer Connector		MCU and JP7 48-pin text tool pin assignment		
JP5/JP6 Pin Number	JP5/JP6 Pin Name	MCU Pin Number	MCU Pin Name	JP7 Pin Number
1	VDD	1	VDD	21
2	GND	8	VSS	28
3	DFTCLK	2	P0.0	22
5	SEL	3	P0.1	23
7	SWAT	6	P0.4	26
9	SWAT	6	P0.4	26
11	DAH	4	P0.2	24
13	DAL	7	P0.5	27
20	PDB	5	P0.3	25

SN8F57011D (SOT23-6L):

Writer Connector		MCU and JP7 48-pin text tool pin assignment		
JP5/JP6 Pin Number	JP5/JP6 Pin Name	MCU Pin Number	MCU Pin Name	JP7 Pin Number
1	VDD	6	VDD	27
2	GND	2	VSS	23
3	DFTCLK	2	P0.0	22
5	SEL	3	P0.1	23
7	SWAT	1	P0.4	22
9	SWAT	1	P0.4	22
11	DAH	4	P0.2	24
13	DAL	7	P0.5	27
20	PDB	3	P0.3	24

22 Ordering Information

A typical surface of SONiX microcontroller is printed with three columns: logo, device's full name, and date code.



22.1 Device Nomenclature

Full Name	Packing Type
S8F5701W	Wafer
SN8F5701H	Dice
SN8F5701PG	PDIP, 8 pins, Green package
SN8F5701SG	SOP, 8 pins, Green package
SN8F5701TG	TSSOP, 8 pins, Green package
SN8F57011DG	SOT23-6L, 6 pins, Green package

22.2 Date Code

The date code includes two parts: date of manufacture and production serial code. The first part is public information which is encoded by following principles.

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

23 Appendix: Reference Document

Sonix provides reference document for users to help them quickly familiar SN8F5000 family (downloadable on cooperative website: www.sonix.com.tw).

Document Name	Description
SN8F5000 Starter-Kit User Manual	This documentation introduces SN8F5000 family all Starter-Kit, providing the user selects an appropriate starter-kit for development.
SN8F5000 Family Instruction Set	The document details the 8051 instruction set, and a simple example illustrates operation.
SN8F5000 Family Instruction Mapping Table	This document supplies the information about mapping assembly instructions from 8-Bit Flash/ OTP Type to 8051 Flash Type.
SN8F5000 Packaging Information	This documentation introduces SN8F5000 family microcontrollers' mechanical data, such as height, width and pitch information.
SN8F5000 Debug Tool Manual	This document teaches the user to install software Keil C51, and helped create a new project to be developed.

SN8F5701 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