SONiX Technology Co., Ltd.

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 8 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: INT0/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 1 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-V low voltage detector
- Wide supply voltage (1.8 V – 5.5 V) and temperature (-40 °C to 85 °C) range

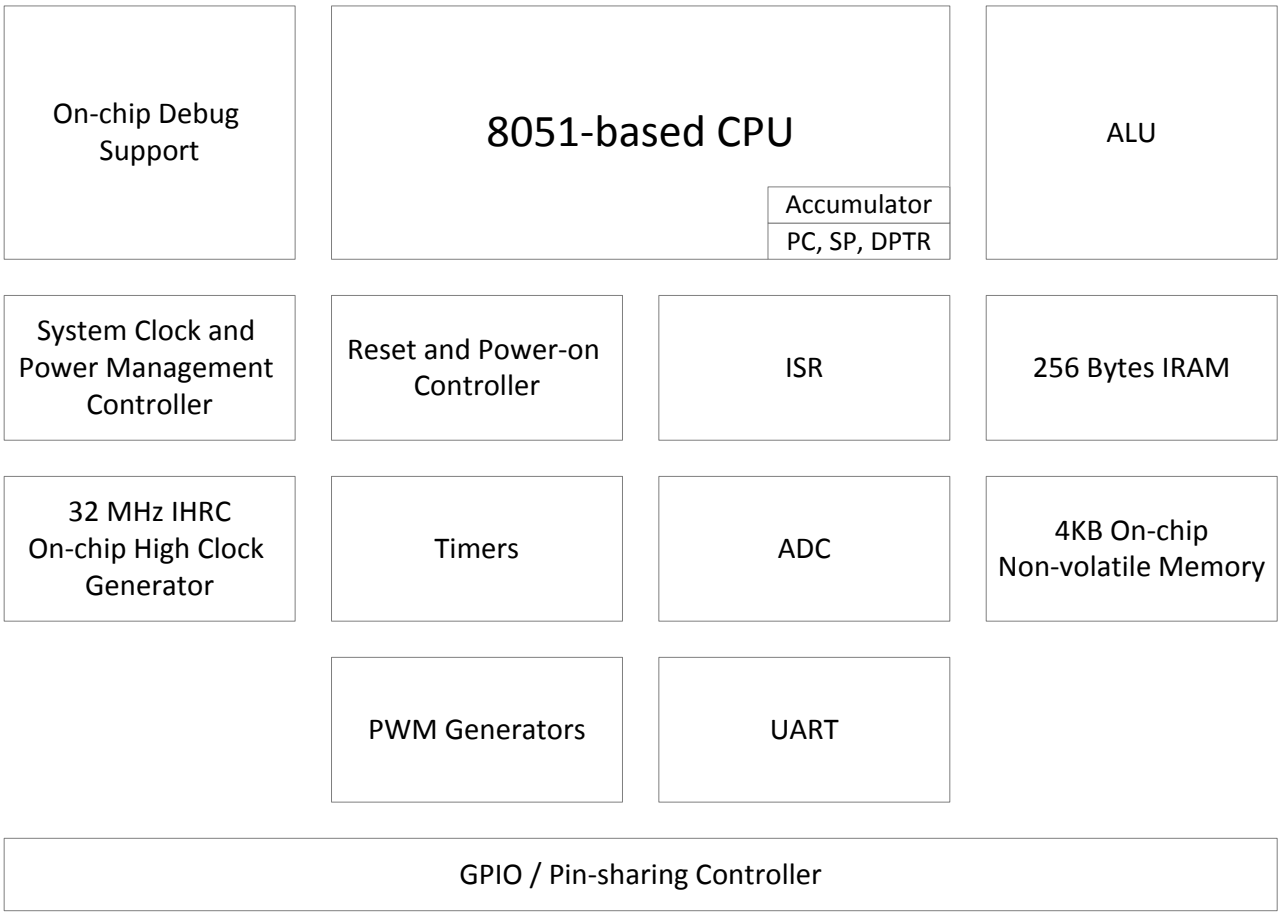
1.2 Applications

- 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



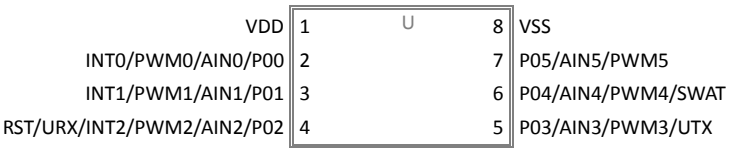
2 Table of Contents

3 Revision History

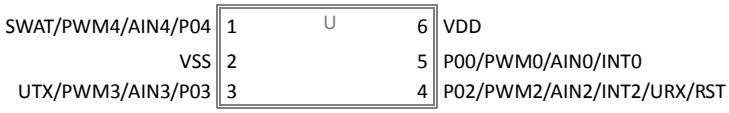
Revision	Date	Description
1.0	May 2017	First issue
1.1	Jul. 2017	Modify ADC VREFH description and Electrical Characteristic chapter.
1.2	Oct. 2017	1. Modify ordering information. 2. Add package information.
1.3	Jul. 2018	1. Remove 2.4/3.3V low voltage detectors description in features 2. Update Electrical Characteristics chapter
1.4	Mar. 2019	1. Repair an error, omission, etc. 2. MP5 Writer Programming Pin Mapping adds normal mode and high speed mode sections. 3. Modify Pin Circuit Diagrams section. 4. Modify ADC input offset range. 5. Modify power on sequence and system clock timing. 6. Modify Package Information section. 7. Modify Timer0/ Timer1 section.

4 Pin Assignments

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



4.2 SN8F57011D (SOT23-6L)



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: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
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: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
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: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
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

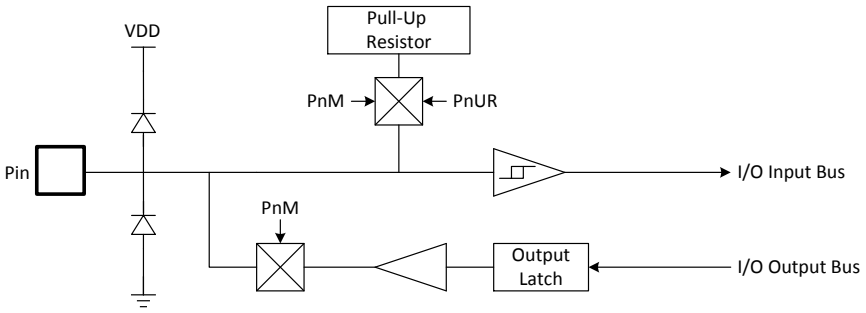
P0.3	Digital I/O	GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
AIN3	Analog Input	ADC: input channel 3.
PWM3	Digital Output	PWM: programmable PWM output.
UTX	Digital Output	UART: transmission pin.
P0.4	Digital I/O	GPIO: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
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: Bi-direction pin. Schmitt trigger structure as input mode. Built-in pull-up resisters. Level change wake-up.
AIN5	Analog Input	ADC: input channel 5.
PWM5	Digital Output	PWM: programmable PWM output.

4.4 Pin Characteristic

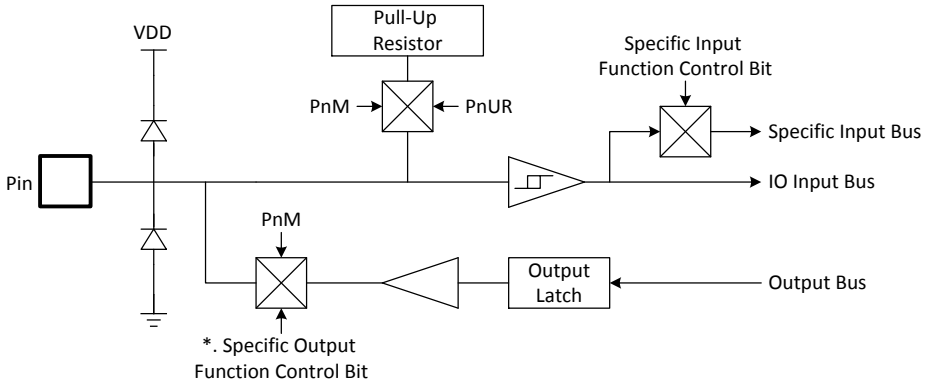
Port	Open-Drain	Sink Current 20mA VSS+0.5V	External Interrupt	Wakeup (Level change)	Shared Pin
P0.0	V	V	V	V	AIN0/PWM0/INT0
P0.1	V	V	V	V	AIN1/PWM1/INT1
P0.2	V	V	V	V	AIN2/PWM2/INT2/URX/RST
P0.3	V	V	-	V	AIN3/PWM3/UTX
P0.4	V	V	-	V	AIN4/PWM4/SWAT
P0.5	V	V	-	V	AIN5/PWM5

4.5 Pin Circuit Diagrams

Normal Bi-direction I/O Pin.

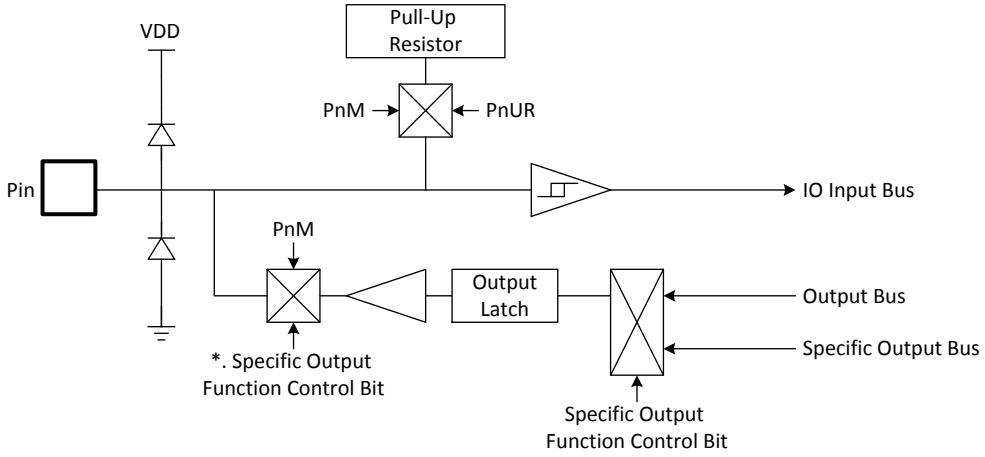


Bi-direction I/O Pin Shared with Specific Digital Input Function, e.g. INT0, UART.



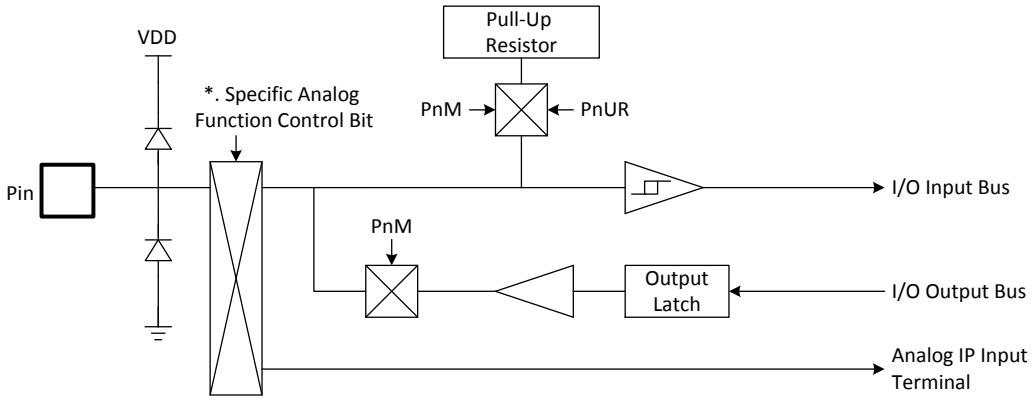
***. Some specific functions switch I/O direction directly, not through PnM register.**

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 Internal RAM (IRAM)

256 X 8-bit RAM (Internal Data Memory)

Address	RAM Location		
000h 01Fh	Work Register Area		00h-7Fh of RAM is direct and indirect access RAM
020h 02Fh	Bit Addressable Area		
030h … … 07Fh	General Purpose Area		
080h … … … 0FFh	General Purpose Area (Indirect Access)	Special Function Register (Direct Access)	080h-0FFh store special function registers. End of Bank 0

The 256-byte data RAM in internal data memory is a standard 8051 RAM access configuration. The upper 128-byte RAM is general purpose RAM and can configure by direct addressing access and indirect addressing access. The lower 128-byte can be indirect access RAM in general purpose or direct access RAM in special function register (SFR).

- 0x0000-0x007F: General purpose RAM contains work register area and bit addressable area. In this area, direct or indirect addressing can be used.
- 0x0000-0x001F: Work register area includes 4-bank. Each bank has 8 work registers (R0 - R7) which is selected by RS0/RS1 in PSW register.
- 0x0020-0x002F: Bit addressable area.

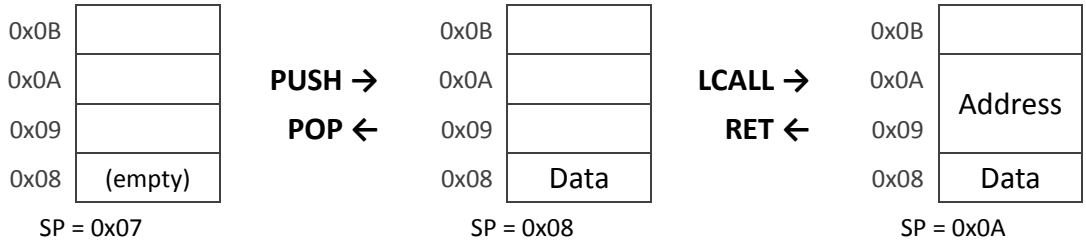
In the bit addressable area, user can read or write any single bit in this range by using the unique address for that bit. Supports 16bytes bit addressable RAM area giving 128 addressable bits. Each bit has individual address in the range from 00H to 7FH. Thus, the bit can be addressed directly. Bit0 of the byte 20H has bit address 00H and Bit 7 of the byte 20H has bit address 07H. Bit0 of the byte 2FH has bit address 78H and Bit 7 of the byte 2FH has bit address 7FH. When set “SETB 42H”, it means the bit2 of the byte 28H is set.

	Byte Address	Bite 0	Bite 1	Bite 2	Bite 3	Bite 4	Bite 5	Bite 6	Bite 7
Bit Addressable Area	**0x20**	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
	0x21	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
	0x22	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
	0x23	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
	0x24	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27
	0x25	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F
	0x26	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
	0x27	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F
	0x28	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47
	0x29	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F
	0x2A	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57
	0x2B	0x58	0x59	0x5A	0x5B	0x5C	0x5D	0x5E	0x5F
	0x2C	0x60	0x61	0x62	0x63	0x64	0x65	0x66	0x67
	0x2D	0x68	0x69	0x6A	0x6B	0x6C	0x6D	0x6E	0x6F
	0x2E	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77
	0x2F	0x78	0x79	0x7A	0x7B	0x7C	0x7D	0x7E	0x7F

- 0x0080~0x00FF: General purpose area in indirect addressing access or special function register in direct addressing access.

5.3 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 or 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.

✷ ***Note: Stack and IRAM share the same area, Keil C51 compiler will not display "error" or "warning" when overlap condition is occurred so user must pay attention.***

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-Link3, and a program can also self-update via in-system program process (refer to In-system Program).

Address	ROM	Comment
0000H	Reset vector	Reset vector
0001H 0002H	General purpose area	User program
0003H	**INT0 Interrupt vector**	**Interrupt vector**
000BH	**TIMER0 Interrupt vector**	
0013H	**INT1 Interrupt vector**	
001BH	**TIMER1 Interrupt vector**	
0023H	**UART Interrupt vector**	
0083H	**INT2 Interrupt vector**	
008BH	**ADC Interrupt vector**	
00EBH	**TIMER3 Interrupt vector**	
00ECH	General purpose area	User program End of user program
0FF6H 0FF7H . 0FFDH 0FFEH 0FFFH	Reserved	

The ROM includes reset vector, Interrupt vector, general purpose area and reserved area. The reset vector is program beginning address. The interrupt vector is the head of interrupt service routine when any interrupt occurring. The general purpose area is main program area including main loop, sub-routines and data table.

- 0x0000 Reset vector: Program counter points to 0x0000 after any reset events (power on reset, reset pin reset, watchdog reset, LVD reset…).
- 0x0001~0x0002: General purpose area to process system reset operation.
- 0x0003~0x00EB: Multi interrupt vector area. Each of interrupt events has a unique interrupt vector.
- 0x00EC~0x0FDF: General purpose area for user program and ISP (EEPROM function).
- 0x0FE0~0x0FF6: General purpose area for user program. Do not execute ISP.
- 0x0FF6~0x0FFF: Reserved area. Do not execute ISP.

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 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	-	P0M	-	-	-	-	-	PFLAG
F0	B	P0UR	FRQL	FRQH	FRQCMD	-	-	SRST
E8	-	-	-	-	-	-	-	-
E0	ACC	-	-	-	P0OC	CLKSEL	CLKCMD	TCON0
D8	S0CON2	-	-	-	-	-	-	-
D0	PSW	-	ADM	ADB	ADR	VREFH	-	-
C8	-	-	-	-	-	-	-	-
C0	-	-	-	-	-	-	-	-
B8	-	IP1	S0RELH	-	PWNV	PWO	PWCH	IRCON2
B0	-	PW3DL	PW3DH	PW4DL	PW4DH	PW5DL	PW5DH	-
A8	IEN0	IP0	S0RELL	PW1DL	PW1DH	PW2DL	PW2DH	-
A0	-	T3M	T3CL	T3CH	T3YL	T3YH	PW0DL	PW0DH
98	S0CON	S0BUF	IEN2	-	-	-	P0CON	-
90	-	-	-	-	PECMD	PEROML	PEROMH	PERAM
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PEDGE
80	P0	SP	DPL	DPH	-	-	WDTR	PCON

✷ ***Note: All SFRs in the left-most column are bit-addressable. (Every 0x0/0x8-ending SFR addresses are bit-addressable).***

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
S0CON	0x98	UART control register.
S0BUF	0x99	UART data buffer.
IEN2	0x9A	Interrupts enable register
-	0x9B	-
-	0x9C	-
-	0x9D	-
P0CON	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
IP0	0xA9	Interrupts priority register.
S0RELL	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.
S0RELH	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	-
S0CON2	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.
P0UR	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	-
P0M	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	

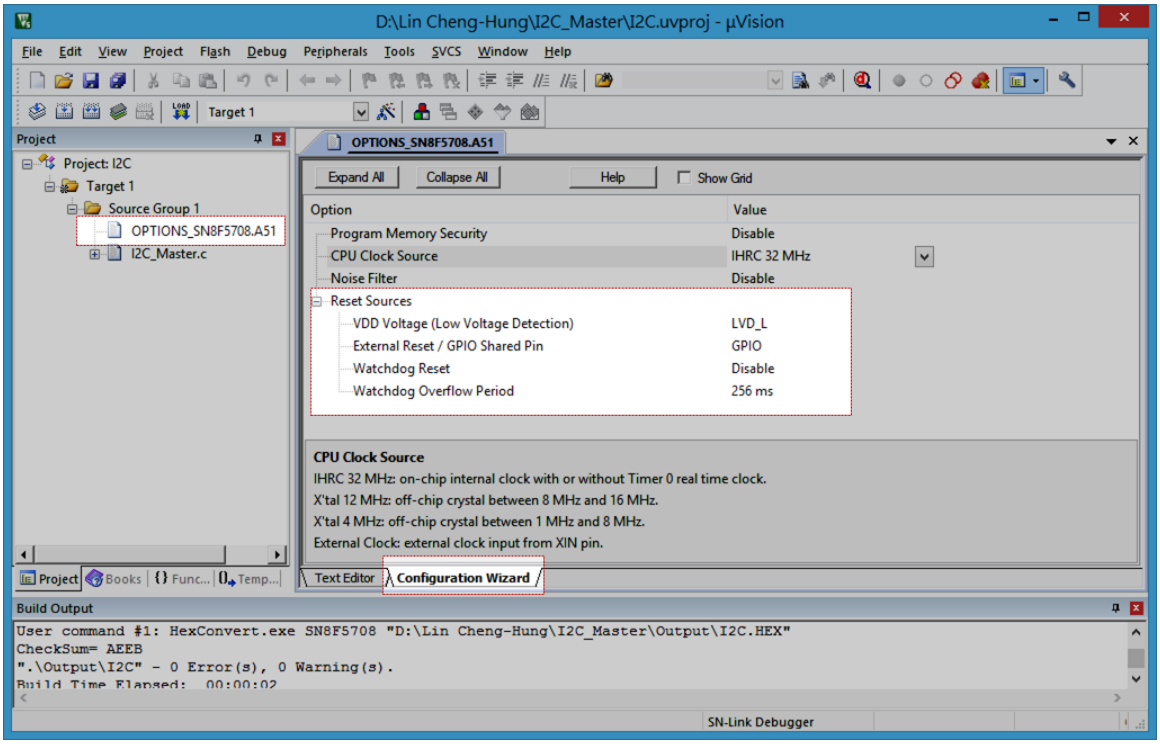
7 Reset and Power-on Controller

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. The option items are as following:

- Program Memory Security
- Noise Filter
- CK_Fine_Tuning
- Reset Source : External Reset / GPIO Shared Pin
- Reset Source : Watchdog Reset & Overflow Period

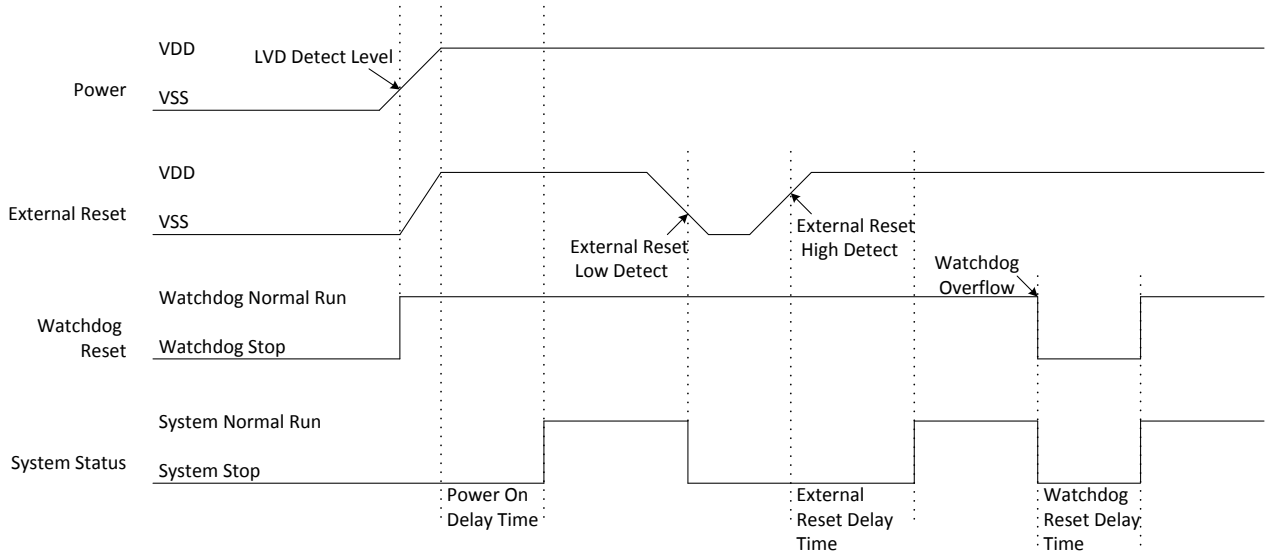


The code option is the system hardware configurations including noise filter option, watchdog timer operation, reset pin option and flash ROM security control. The code option items are as following table:

Code Option	Content	Function Description
Program Memory Security	Security Disable	Disable ROM code Security function
	Security Enable	Enable ROM code Security function
Noise Filter	Disable	Disable Noise Filter
	Enable	Enable Noise Filter
CK_Fine_Tuning	Disable	Disable CK_Fine_Tuning
	Enable	Enable CK_Fine_Tuning
External Reset	Reset with De-bounce	Enable External reset pin with De-bounce
	Reset without De-bounce	Enable External reset pin without De-bounce
	GPIO with P02	Enable P02
Watchdog Reset	Always	Watchdog timer is always on enable even in STOP mode and IDLE mode
	Enable	Enable watchdog timer. Watchdog timer stops in STOP mode and IDLE mode
	Disable	Disable Watchdog function
Watchdog Overflow Period	64ms	Watchdog timer clock source F_{ILRC} /4
	128ms	Watchdog timer clock source F_{ILRC} /8
	256ms	Watchdog timer clock source F_{ILRC} /16
	512ms	Watchdog timer clock source F_{ILRC} /32

7.2 Power-on Sequence

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

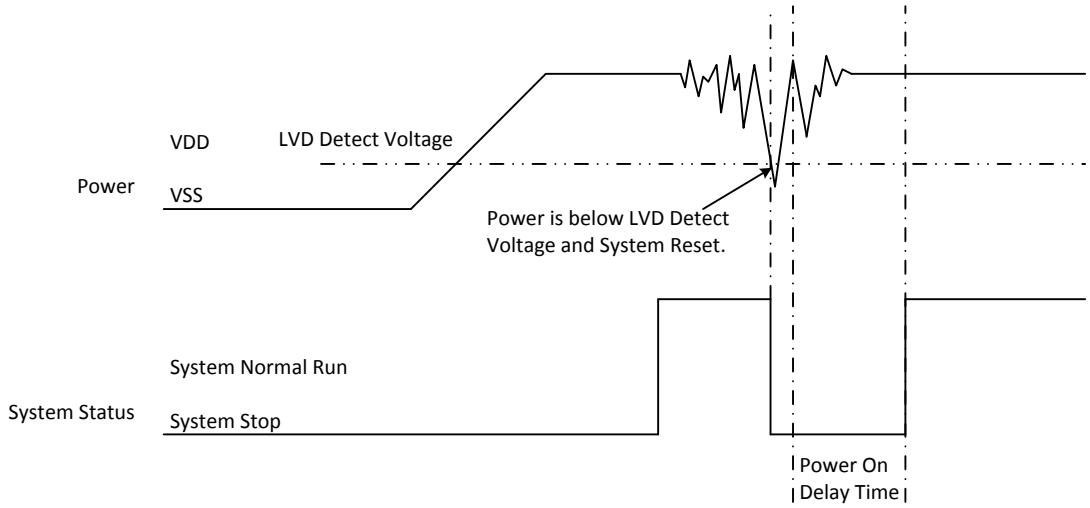


The power stabilization period spends 4.6 ms in typical condition. Afterward the microcontroller fetches CPU Clock Source selection automatically. The selected clock source would be driven, and the system counts 2048 times of the clock period and 5 times of the internal low-speed oscillator clocks to ensure its reliability.

✷ ***Note: In high power noise environment, user can put 10ohm resistor in the front of 0.1uF capacitor & VDD PAD to suppress power noise and avoid IC damage.***

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/ (Internal Low-Speed oscillator frequency/WDT Pre-scalar)
= 256 / (F_{ILRC}/WDT Pre-scaler) …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

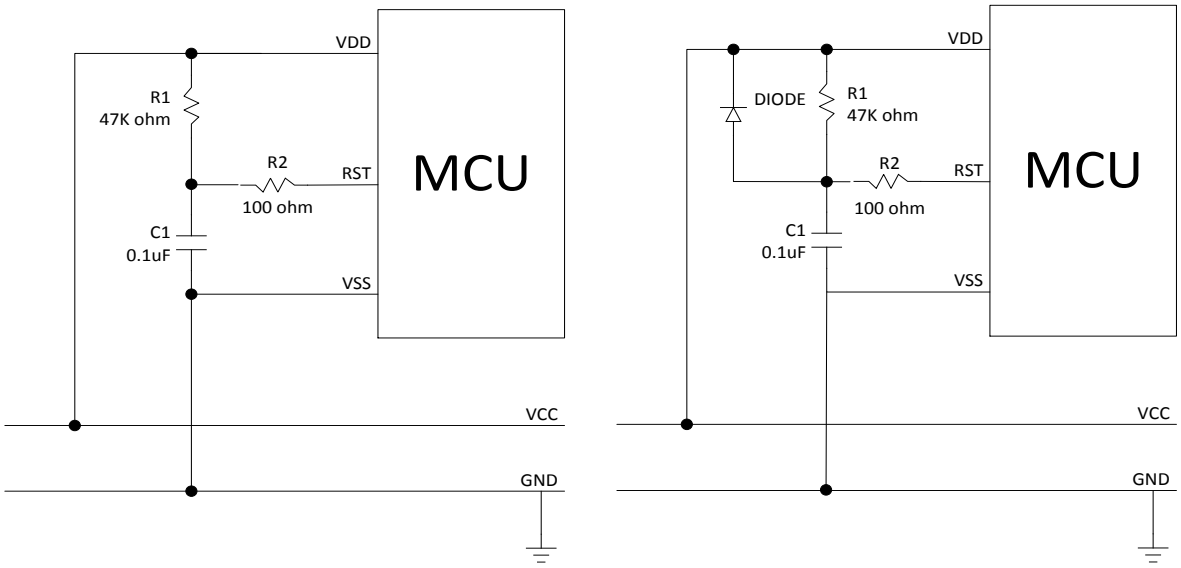
The operation mode of watchdog is configurable in options file:
Always mode counts its internal timer in all CPU operation modes (normal, IDLE, SLEEP);
Enable mode counts its internal timer during CPU stays in normal mode, and it would not trigger watchdog reset in IDLE and STOP modes;
Disable mode suspends 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 (lager than 70% of VDD).

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



✸ ***Note:***

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

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

7.6 Software Reset

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

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

7.7 Reset and Power-on Controller Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	POR	WDT	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.

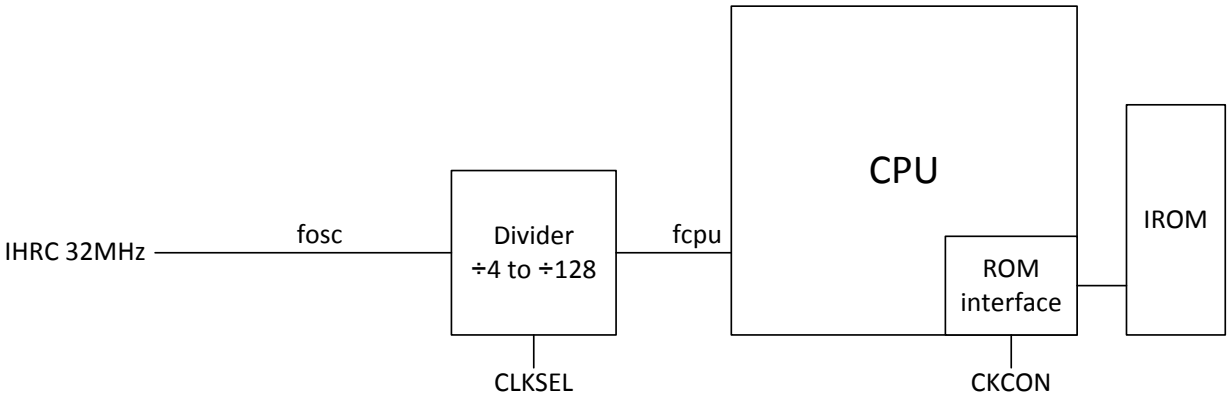
The normal mode means that CPU and peripheral functions are under normally execution. The system clock is based on the combination of source selection, clock divider, and program memory wait state. IDLE mode is the situation that temporarily suspends CPU clock and its execution, yet it remains peripherals' functionality (e.g. timers, PWM, and UART). 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 4 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.

```
CKCON = 0x70;     // For change safely the system clock
CLKSEL = 0x05;    // Set fcpu = fosc / 4
CLKCMD = 0x69;    // Apply CLKSEL's setting
CKCON = 0x00;    // IROM fetch = fcpu / 1
```



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.

IROM fetching cycle (Instruction cycle) ≦ 8MHz

✷ ***Note: For user develop program in C language or assembly, the first line of the program "must be set" CLKSEL= 0x05~0x00, CLKMD= 0x69 and then set CKCON= 0x00~0x70, this priority cannot be modified.***

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

Code Option CPU Clock Source	Fcpu = CLKSEL[2:0]	IROM Fetch = CKCON[6:4]
IHRC 32M	**Only Support** 000 = fosc / 128 001 = fosc / 64 010 = fosc / 32 011 = fosc / 16 100 = fosc / 8 101 = fosc / 4	**Only Support** **000 = fcpu / 1 => Recommend!** 001 = fcpu / 2 010 = fcpu / 3 011 = fcpu / 4 100 = fcpu / 5 101 = fcpu / 6 110 = fcpu / 7 111 = fcpu / 8

8.2 High Speed Clock

High-speed clock has only internal type. The internal high-speed oscillator is 32MHz RC type.

- IHRC 32M: The system high-speed clock source is internal high-speed 32MHz RC type oscillator.

8.3 Power Management

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

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

If IDLE bit is set by program, only CPU clock source would be gated. Consequently, peripheral functions (such as timers, PWM, 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/P1 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/P1 level change trigger.

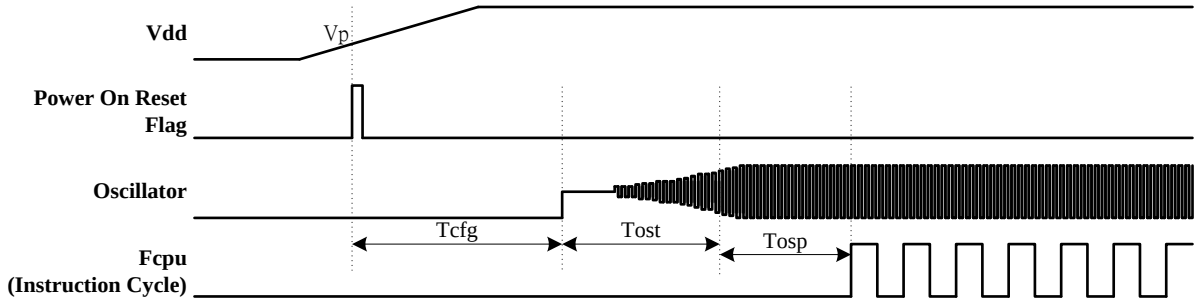
For user who is develop program in C language, IDLE and STOP macros is strongly recommended to control the microcontroller's system mode, instead of set IDLE and STOP bits directly.

```
1     IDLE();
2     STOP();
```

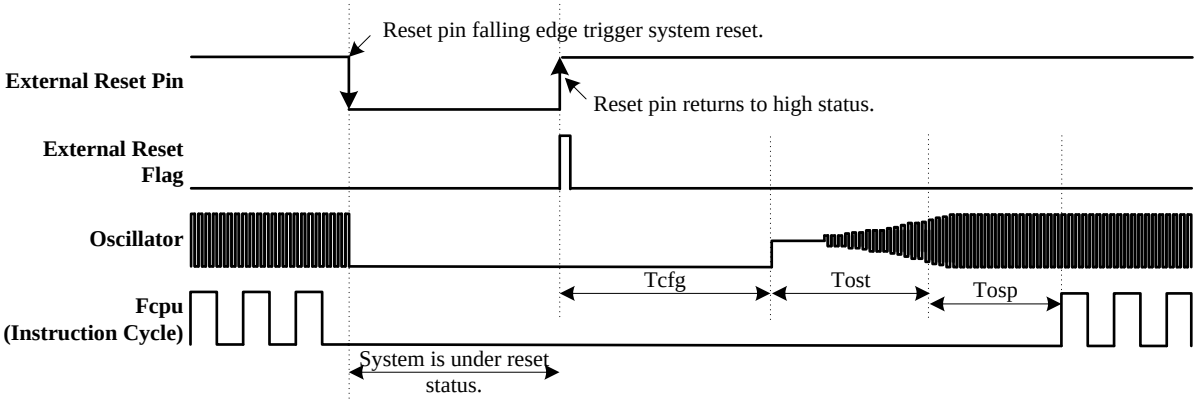
8.4 System Clock Timing

Parameter	Symbol	Description	Typical
Hardware configuration time	Tcfg	$8*F_{ILRC} + 2^{17}*F_{IHRC}$	4.6ms @ F_{ILRC} = 16KHz & F_{IHRC} = 32MHz
Oscillator start up time	Tost	The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator. The RC type oscillator's start-up time is faster than crystal type oscillator.	-
Oscillator warm-up time	Tosp	Oscillator warm-up time of reset condition. $2048*F_{hosc}+5*F_{ILRC}$ (Power on reset, LVD reset, watchdog reset, external reset pin active.)	377us @ F_{hosc} = 32MHz
		Oscillator warm-up time of power down mode wake-up condition. $64*F_{hosc}+5*F_{ILRC}$……RC type oscillator, e.g. internal high-speed RC type oscillator.	RC: 315us @ F_{hosc} = 32MHz

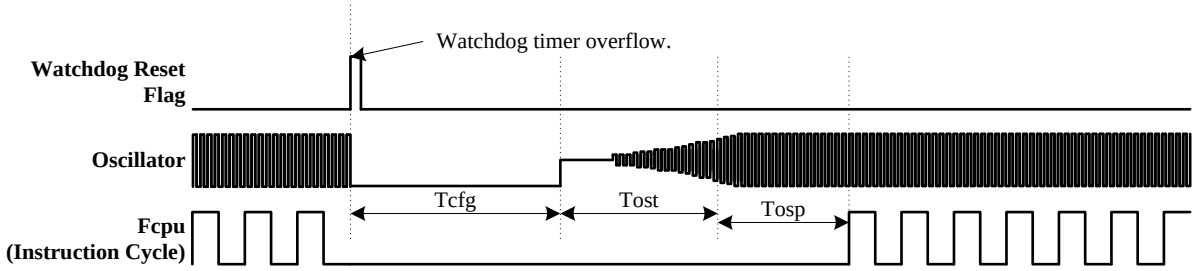
- Power On Reset Timing



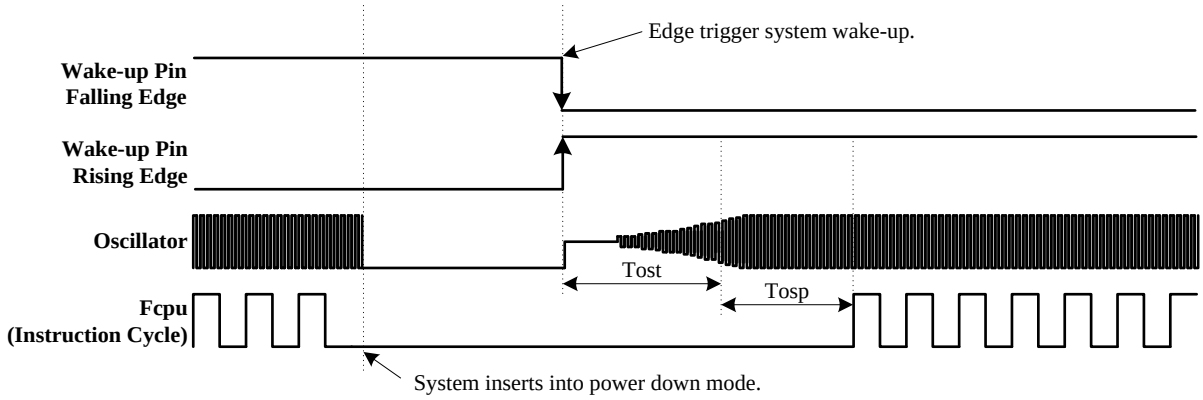
- External Reset Pin Reset Timing



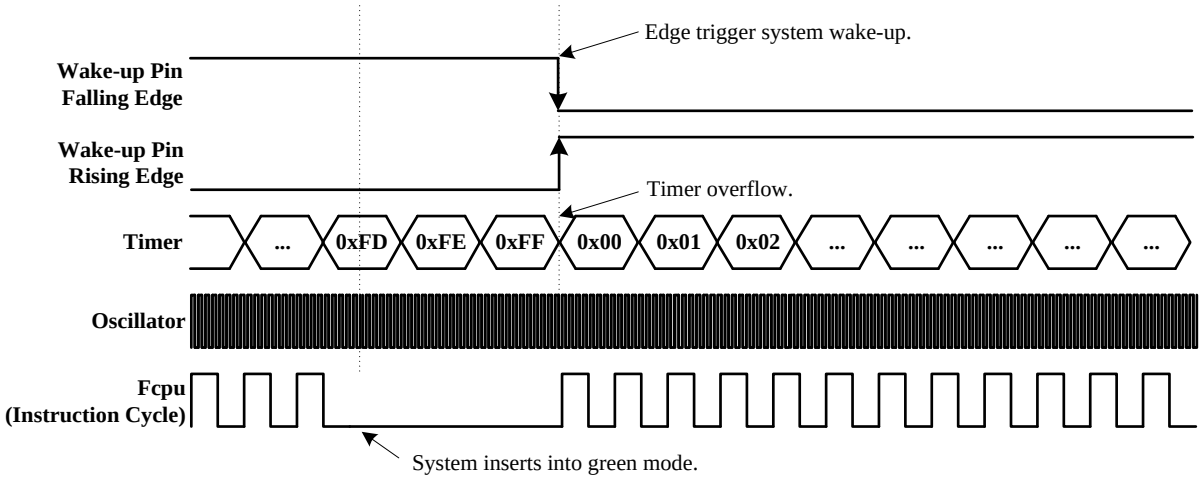
- Watchdog Reset Timing



- Power Down Mode Wake-up Timing

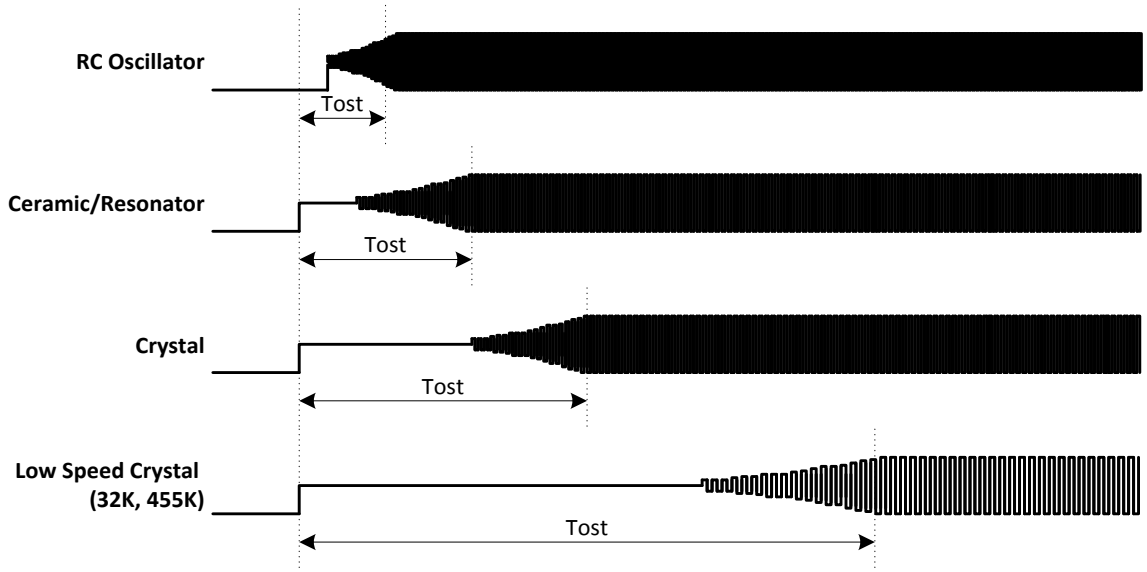


- Idle Mode Wake-up Timing



- Oscillator Start-up Time

The start-up time is depended on oscillator's material, factory and architecture. Normally, the low-speed oscillator's start-up time is lower than high-speed oscillator.



8.5 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
3..0	Reserved	R	0	

CLKSEL Register (0xE5)

Bit	Field	Type	Initial	Description
7..3	Reserved	R	0x00	
2..0	CLKSEL[2:0]	R/W	111	CLKSEL would be applied by writing CLKCMD. 000: fcpu = fosc / 128 001: fcpu = fosc / 64 010: fcpu = fosc / 32 011: fcpu = fosc / 16 100: fcpu = fosc / 8 101: fcpu = fosc / 4 110: fcpu = fosc / 2 111: fcpu = fosc / 1

CLKCMD Register (0xE6)

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

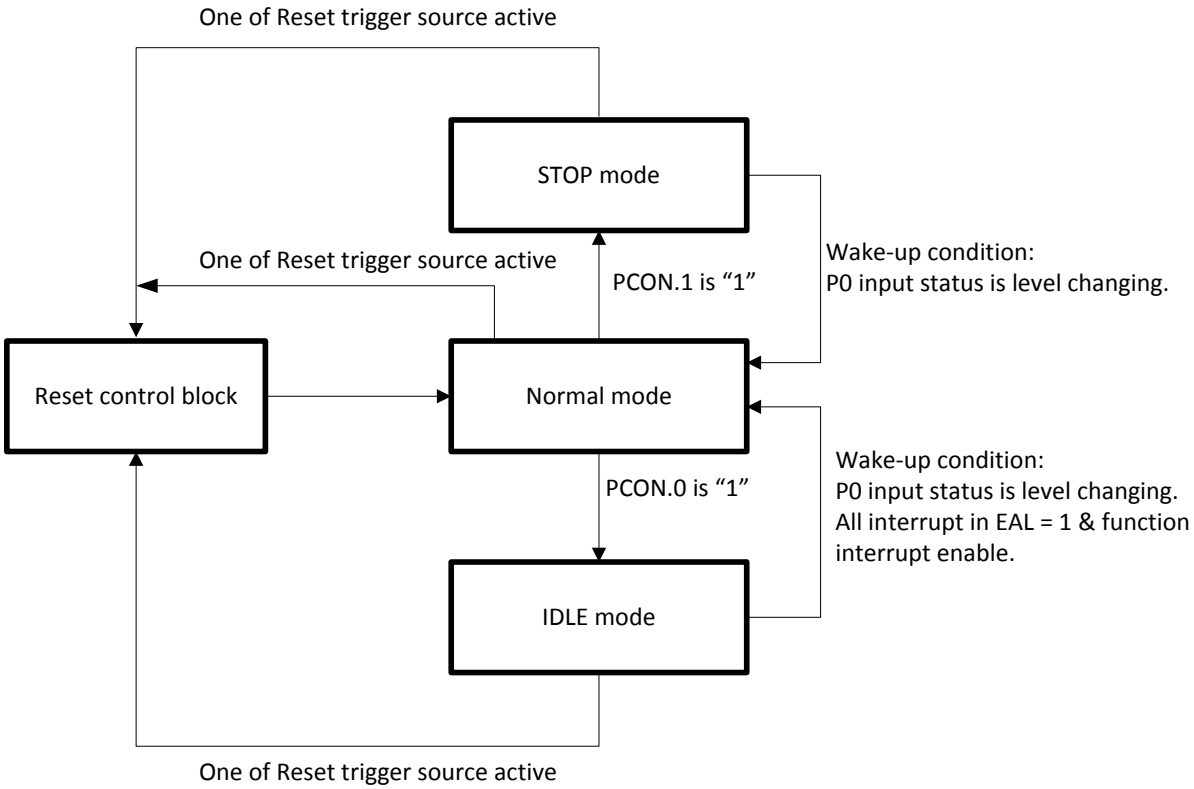
PCON Register (0x87)

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 System Operating Mode

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

- Normal mode: System high-speed operating mode
- IDLE mode: System idle mode (Green mode)
- STOP mode: System power saving mode (Sleep mode)



The operating mode clock control as following table:

Operating Mode	Normal Mode	IDLE Mode	STOP Mode
IHRC	IHRC: Running	IHRC: Running	Stop
ILRC	Running	Running	Watchdog always: Running Other : stop
CPU instruction	Executing	Stop	Stop
Timer 0 (Timer, Event counter)	Active by TR0	Active by TR0	Inactive
Timer 1 (Timer, Event counter)	Active by TR1	Active by TR1	Inactive
Timer 3 (Timer)	Active by T3EN	Active by T3EN	Inactive
UART	Active as enable	Active as enable	Inactive
ADC	Active as enable	Active as enable	Inactive
Watchdog timer	By Watchdog Code option	By Watchdog Code option	By Watchdog Code option
Internal interrupt	All active	All active	All inactive
External interrupt	All active	All active	All inactive
Wakeup source	-	P0, Reset, All interrupt in EAL = 1 & function interrupt enable	P0, Reset.

- IHRC: Internal high-speed oscillator RC type.
- ILRC: Internal low-speed oscillator RC type.

9.1 Normal Mode

The Normal Mode is system high clock operating mode. The system clock source is from high speed oscillator. The program is executed. After power on and any reset trigger released, the system inserts into normal mode to execute program. When the system is wake-up from STOP/IDLE mode, the system also inserts into normal mode. In normal mode, the high speed oscillator actives, and the power consumption is largest of all operating modes.

- The program is executed, and full functions are controllable.
- The system rate is high speed.
- The high speed oscillator and internal low speed RC type oscillator active.
- Normal mode can be switched to other operating modes through PCON register.
- STOP/IDLE mode is wake-up to normal mode.

9.2 STOP Mode

The STOP mode is the system ideal status. No program execution and oscillator operation. Only internal regulator actives to keep all control gates status, register status and SRAM contents. The STOP mode is waked up by P0 hardware level change trigger. P0 wake-up function is always enables. The STOP mode is wake-up to normal mode. Inserting STOP mode is controlled by stop bit of PCON register. When stop = 1, the system inserts into STOP Mode. After system wake-up from STOP mode, the stop bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- All oscillators including external high speed oscillator, internal high speed oscillator and internal low speed oscillator stop.
- Only internal regulator actives to keep all control gates status, register status and SRAM contents.
- The system inserts into normal mode after wake-up from STOP mode.

9.3 IDLE Mode

The IDLE mode is another system ideal status not like STOP mode. In STOP mode, all functions and hardware devices are disabled. But in IDLE mode, the system clock source keeps running, so the power consumption of IDLE mode is larger than STOP mode. In IDLE mode, the program isn't executed, but the timer with wake-up function actives as enabled, and the timer clock source is the non-stop system clock. The IDLE mode has 2 wake-up sources. One is the P0 level change trigger wake-up. The other one is any interrupt in EAL = 1 & function interrupt enable. That's mean users can setup any function with interrupt enable, and the system is waked up until the interrupt issue. Inserting IDLE mode is controlled by idle bit of PCON register. When idle = 1, the system inserts into IDLE mode. After system wake-up from IDLE mode, the idle bit is disabled (zero status) automatically.

- The program stops executing, and full functions are disabled.
- Only the timer with wake-up function actives.
- The oscillator to be the system clock source keeps running, and the other oscillators operation is depend on system operation mode configuration.
- If inserting IDLE mode from normal mode, the system insets to normal mode after wake-up.
- The IDLE mode wake-up sources are P0 level change trigger.
- If the function clock source is system clock, the functions are workable as enabled and under IDLE mode, e.g. Timer, PWM, event counter…
- All interrupt in EAL = 1 & function interrupt enable can wake-up in IDLE mode.

9.4 Wake up

Under STOP mode (sleep mode) or idle mode, program doesn't execute. The wakeup trigger can wake the system up to normal mode. The wakeup trigger sources are external trigger (P0 level change) and internal trigger (any interrupt in EAL = 1 & function interrupt enable). The wakeup function builds in interrupt operation issued request flag and trigger system executing interrupt service routine as system wakeup occurrence.

When the system is in STOP mode the high clock oscillator stops. When waked up from STOP mode, MCU waits for 2048 external high-speed oscillator clocks + 5 internal low-speed oscillator clocks and 64 internal high-speed oscillator clocks + 5 internal low-speed oscillator clocks as the wakeup time to stable the oscillator circuit. After the wakeup time, the system goes into the normal mode.

The value of the external high clock oscillator wakeup time is as the following.

The Wakeup time = 1/Fosc * 2048 (sec) + 1/Flosc * 5 + high clock start-up time

Example: In STOP mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

The wakeup time = 1/Fosc * 2048 + 1/Flosc * 5 = 0.825 ms (Fosc = 4MHz)

The total wakeup time = 0.825 ms + oscillator start-up time

The value of the internal high clock oscillator RC type wakeup time is as the following.

The Wakeup time = 1/Fosc * 64 (sec) + 1/Flosc * 5 + high clock start-up time

Example: In STOP mode (sleep mode), the system is waked up. After the wakeup time, the system goes into normal mode. The wakeup time is as the following.

The wakeup time = 1/Fosc * 64 +1/Flosc * 5= 315 us (Fhosc = 32MHz)

✷ ***Note: The high clock start-up time is depended on the VDD and oscillator type of high clock.***

Under STOP mode and green mode, the I/O ports with wakeup function are able to wake the system up to normal mode. The wake-up trigger edge is level changing in rising edge or falling edge. The Port 0 has wakeup function. Port 0 wakeup functions always enable.

10 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	TI0 / RI0	By firmware	7 / 0x0023
Timer 3	ET3	TF3	Automatically	8 /0x00EB

10.1 Interrupt Operation

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

10.2 Interrupt Priority

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

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

same priority level, the priority rule follows default priority.

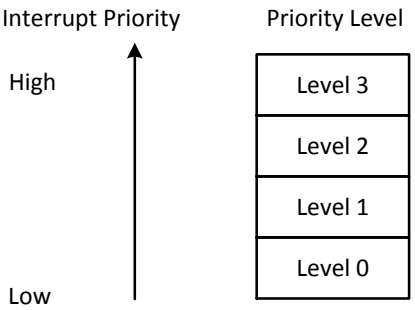
Priority Level	IP1.x	IP0.x
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

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

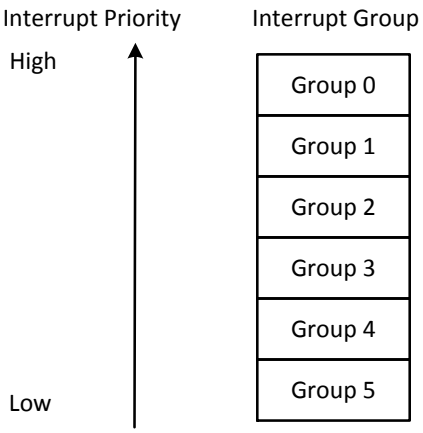
Group	Interrupt Source			
Group 0	INT0	INT2	-	-
Group 1	T0	ADC	-	-
Group 2	INT1	-	-	-
Group 3	T1	-	-	-
Group 4	UART	-	-	-
Group 5	T3	-	-	-

When more than one interrupt request occur, the highest priority request must be executed first. Choose the highest priority request according natural priority and priority level. The steps are as the following:

1. Choose the groups which have the highest priority level between all groups.
2. Choose the group which is the highest nature priority between the groups with the highest priority level.
3. Choose the ISR which has the highest nature priority inside the group with the highest priority.



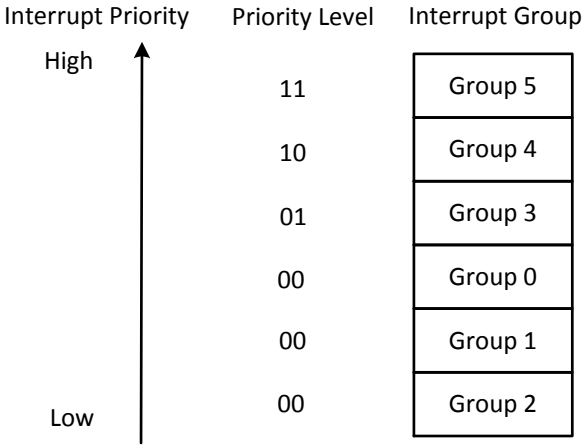
Higher priority level has higher priority



All groups within the same priority level
IP0.0 = IP0.1 = IP0.2 = IP0.3 = IP0.4 = IP0.5
IP1.0 = IP1.1 = IP1.2 = IP1.3 = IP1.4 = IP1.5

As the example, group5 has the highest priority level and group0~group2 have the lowest priority level. It means the interrupt vector in group5 has the highest interrupt priority, the 2nd interrupt priority in group4 and the 3rd interrupt priority in group3. Group0~ group2 have the same priority level thus the nature priority rule will be followed. Therefore, interrupt priority will be group5> group4> group3> group0> group1> group2.

```
MOV     IP0, #00101000B         ; Set group0 - group5 in different priority level.
MOV     IP1, #00110000B
```



Priority Level: GP5>GP4>GP3>GP2=GP1=GP0

Interrupt Priority: GP5>GP4>GP3>GP0>GP1>GP2

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	

10.3 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.
```

10.4 Interrupt Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	-	ES0	ET1	EX1	ET0	EX0
IEN2	-	-	-	-	-	EX2	ET3	EADC
IRCON2	-	-	-	-	-	IE2	TF3	ADCF
TCON	TF1	TR1	TF0	TR0	IE1	-	IE0	-
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0

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: INT0 interrupt request.
1	IE0	R/W	0	External P0.0 interrupt (INT0) request flag 0: None INT0 interrupt request. 1: INT0 interrupt request.
Else				Refer to other chapter(s)

S0CON Register (0X98)

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

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

```
                ORG     0               ; 0000H
                JMP     START           ; Jump to user program address.
                …
                ORG     0X0003          ; Jump to interrupt service routine address.
                JMP     ISR_INT0
                ORG     0X000B
                JMP     ISR_T0
                …
                ORG     0X0083
                JMP     ISR_INT2
                …
                ORG     0X00ECH
START:                                  ; 00ECH, The head of user program.
                …                       ; User program.
                …
                …
                JMP     START           ; End of user program.
                …
ISR_ INT0:                              ; The head of interrupt service routine.
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.
ISR_T0:                                 ;
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.
                …
ISR_INT2                                ;
                PUSH    ACC             ; Save ACC to stack buffer.
                PUSH    PSW             ; Save PSW to stack buffer.
                …
                POP     PSW             ; Load PSW from stack buffer.
                POP     ACC             ; Load ACC from stack buffer.
                RETI                    ; End of interrupt service routine.


                END                     ; End of program.
```

11 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.

11.1 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..0				et cetera

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..0	et cetera

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

11.2 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 P0M 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..0				et cetera

11.3 On-chip Pull-up Resisters

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

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR

P0UR: 0xF1

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

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

11.4 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
P0CON	-	-	P0CON5	P0CON4	P0CON3	P0CON2	P0CON1	P0CON0

P0CON: 0x9E

Bit	Field	Type	Initial	Description
5..0	P0CON[5:0]	R/W	0	P0 configuration control bit*. 0: P0 can be analog input pin or digital GPIO pin. 1: P0 is pure analog input pin and can't be a digital GPIO pin.

* P0CON [5:0] will configure related Port0 pin as pure analog input pin to avoid current leakage.

12 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.

12.1 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 2 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	

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.
2	EX1	R/W	0	External P0.1 interrupt (INT1) control bit. 0: Disable INT1 interrupt function. 1: Enable INT1 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				Refer to other chapter(s)

TCON Register (0X88)

Bit	Field	Type	Initial	Description
3	IE1	R/W	0	External P0.1 interrupt (INT1) request flag 0: None INT1 interrupt request. 1: INT1 interrupt request.
1	IE0	R/W	0	External P0.0 interrupt (INT0) request flag 0: None INT0 interrupt request. 1: INT0 interrupt request.
Else				Refer to other chapter(s)

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.
Else				Refer to other chapter(s)

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.
Else				Refer to other chapter(s)

12.2 Sample Code

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

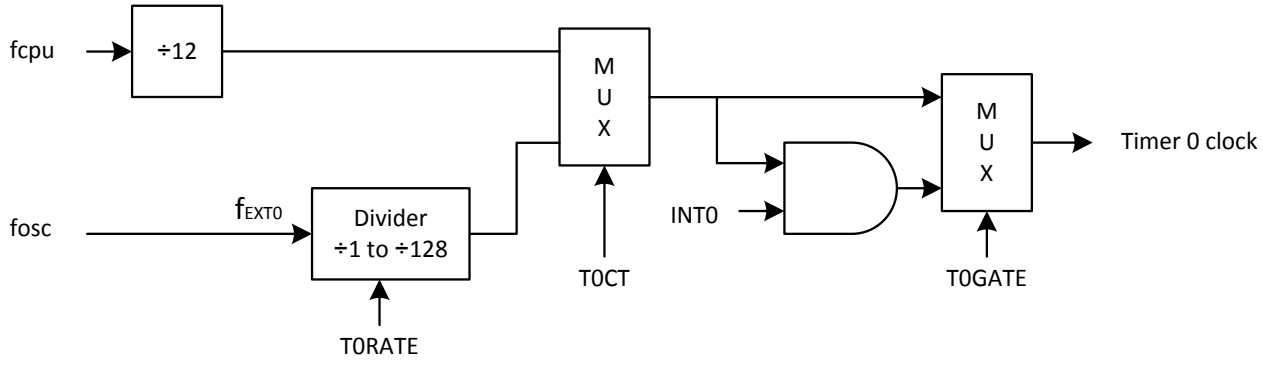
```
#define INT0Rsing     (1 << 0)  //INT0 trigger edge is rising edge
#define INT0Falling   (2 << 0)  //INT0 trigger edge is falling edge
#define INT0LeChge    (3 << 0)  //INT0 trigger edge is level chagne
#define EINT0         (1 << 0)  //INT0 interrupt enable

#define INT1Rsing     (1 << 2)  //INT1 trigger edge is rising edge
#define INT1Falling   (2 << 2)  //INT1 trigger edge is falling edge
#define INT1LeChge    (3 << 2)  //INT1 trigger edge is level chagne
#define EINT1         (1 << 2)  //INT1 interrupt enable

#define INT2Rsing     (1 << 4)  //INT2 trigger edge is rising edge
#define INT2Falling   (2 << 4)  //INT2 trigger edge is falling edge
#define INT2LeChge    (3 << 4)  //INT2 trigger edge is level chagne
#define EINT2         (1 << 2)  //INT2 interrupt enable

void EnableINT(void)
{
  // INT0 rising edge, INT1 falling edge, INT2 level change
  PEDGE = INT0Rising |INT1Falling | INT2LeChge;

  // Enable INT0/INT1 interrupt
  IEN0 |= EINT0 | EINT1;
  // Enable INT2 interrupt
  IEN2 |= EINT2;
  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M = 0x38;
}

void INT0Interrupt(void) interrupt ISRInt0  //0x03
{ //IE0 clear by hardware
  P03 = ~P03;
}

void INT1Interrupt(void) interrupt ISRInt1  //0x13
{ //IE1 clear by hardware
  P04 = ~P04;
}

void INT2Interrupt(void) interrupt ISRInt2  //0x83
{ //IE2 clear by hardware
  P05 = ~P05;
}
```

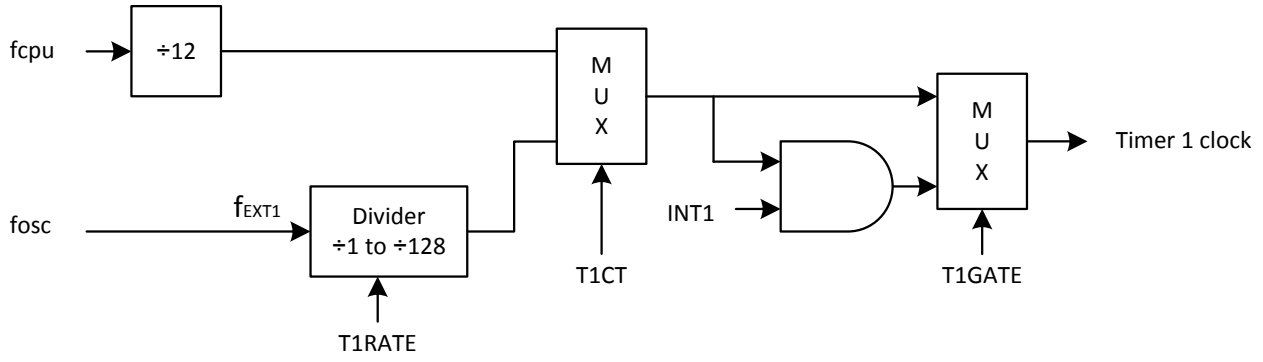
13 Timer 0 and Timer 1

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.

13.1 Timer 0 and Timer 1 Clock Selection

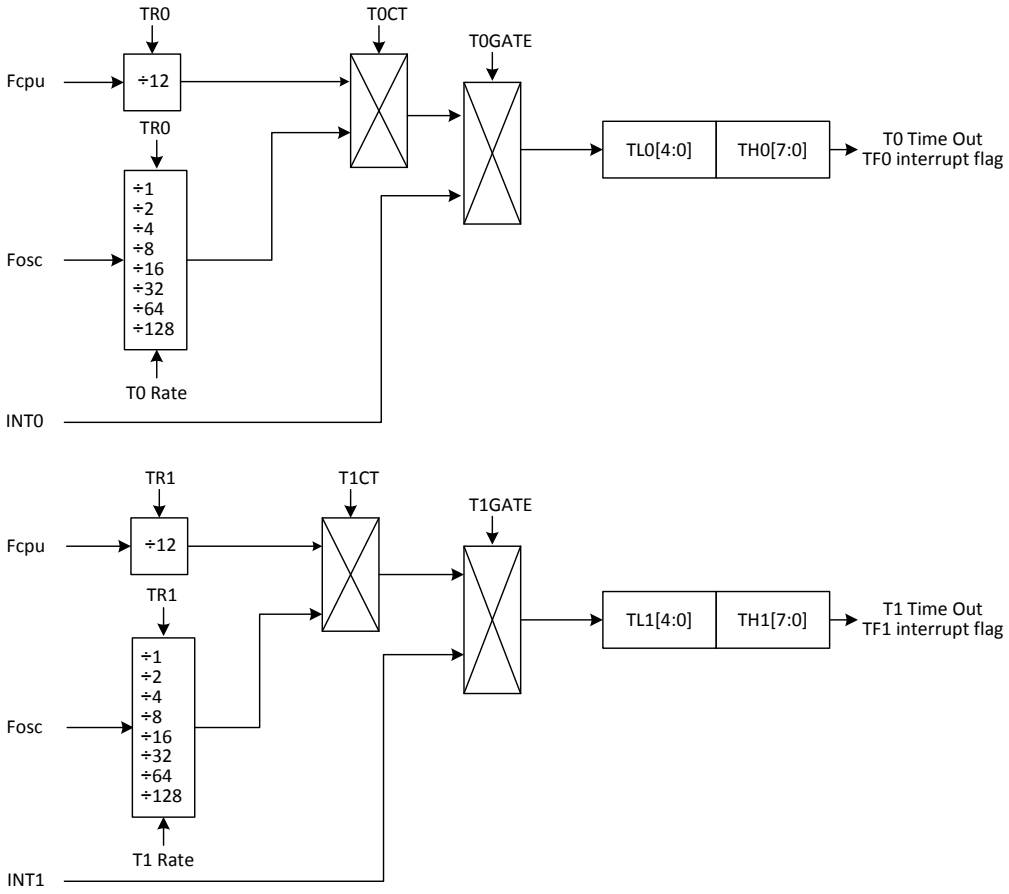
The figures below illustrate the clock selection circuit of Timer 0 and Timer 1. Timer 0 has two clock sources selection: fcpu and fosc. All clock sources can be gated (pause) by INT0 pin if T0GATE is applied. Timer 1 clock sources selection: fcpu and fosc. All clock sources can be gated (pause) by INT1 pin if T1GATE is applied.





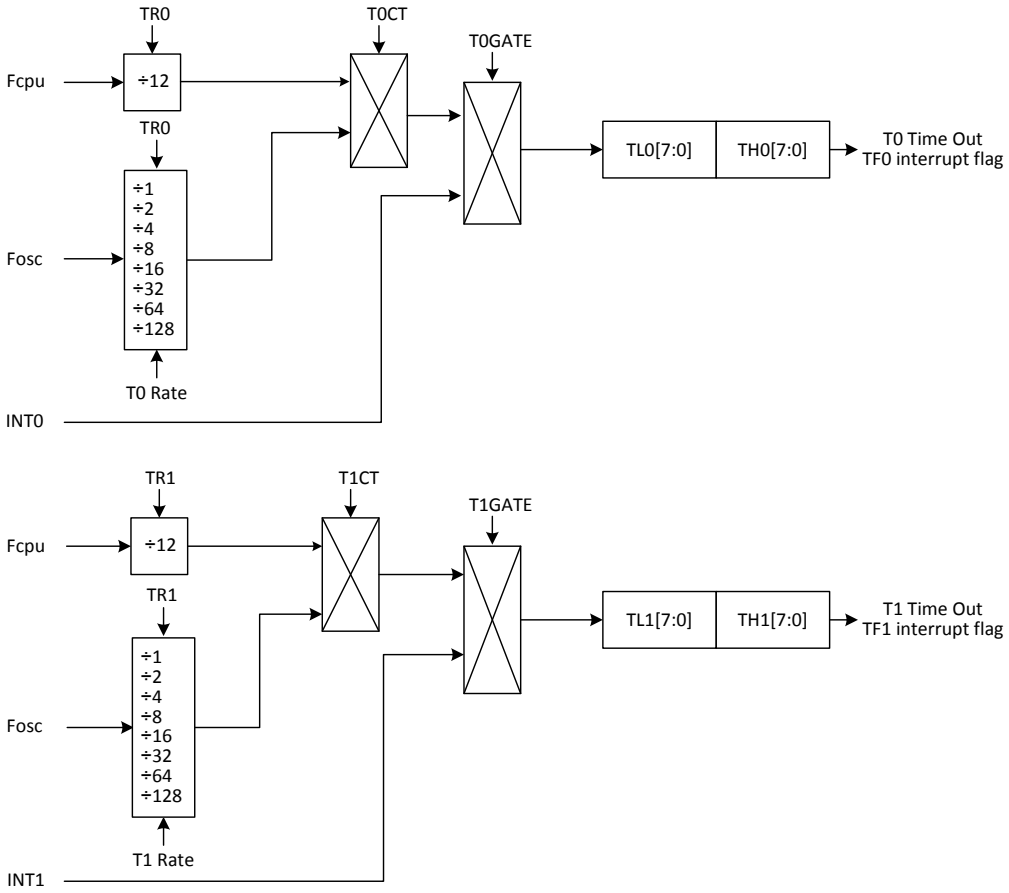
13.2 Mode 0: 13-bit Up Counting Timer

Timer 0 and Timer 1 in 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.



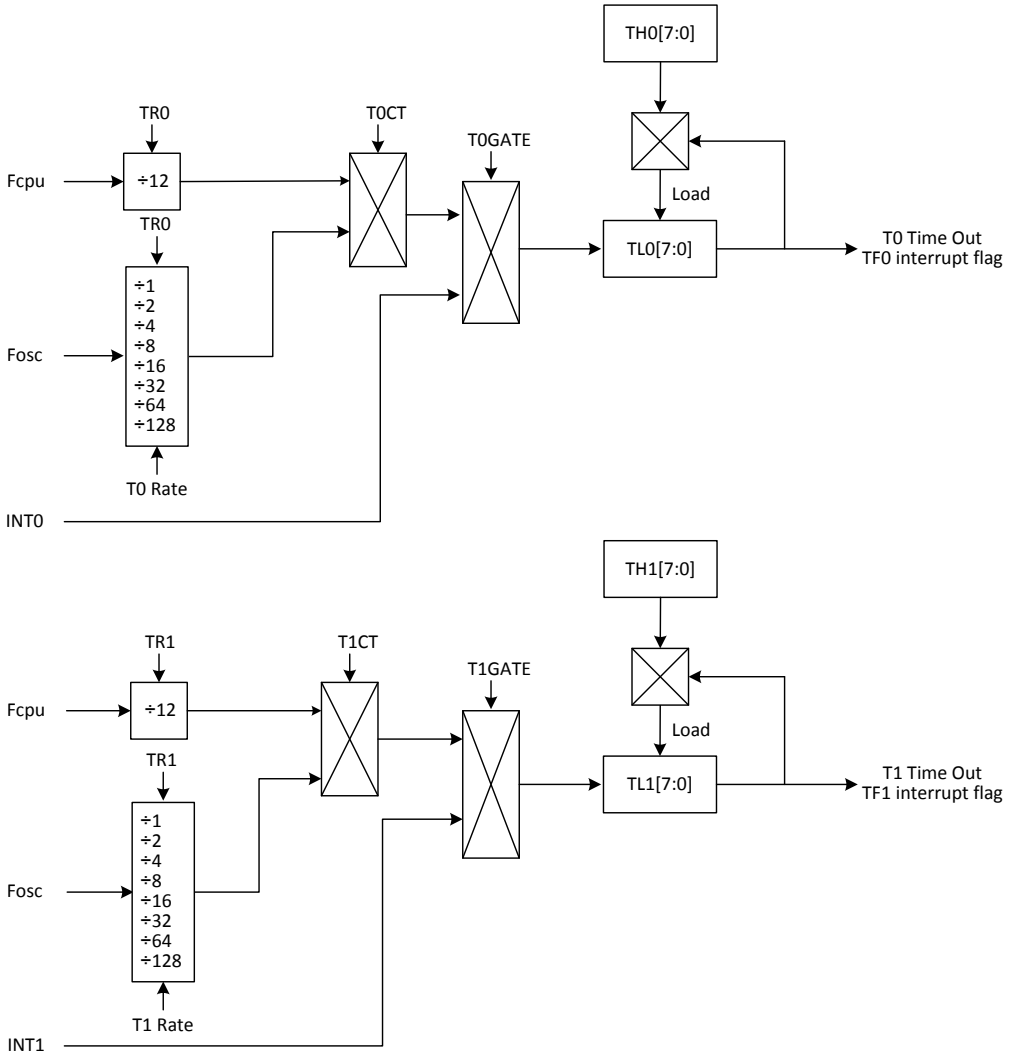
13.3 Mode 1: 16-bit Up Counting Timer

Timer 0 and Timer 1 in 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).



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

Timer 0 and Timer 1 in 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.

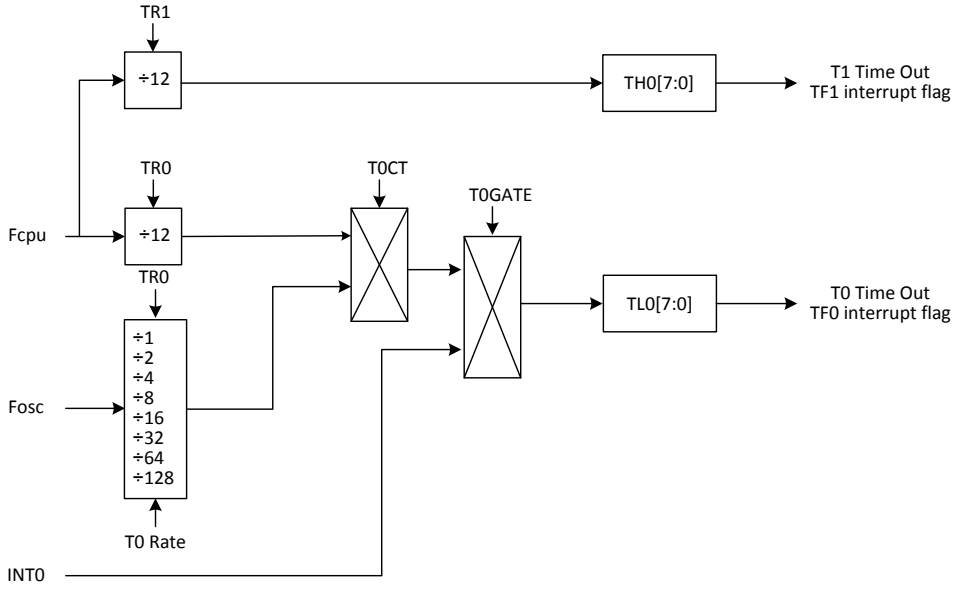


13.5 Mode 3 (Timer 0 only): Separated Two 8-bit Up Counting Timer

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 INT0 pin if T0GATE is applied.

In this mode TL0 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.



13.6 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	-	T0RATE2	T0RATE1	T0RATE0	-	T1RATE2	T1RATE1	T1RATE0
TMOD	T1GATE	T1CT	T1M1	T1M0	T0GATE	T0CT	T0M1	T0M0
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	-	-	ES0	ET1	EX1	ET0	EX0

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	TR0	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 INT0
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	ET0	R/W	0	Timer 0 interrupt 0: Disable 1: Enable
Else				Refer to other chapter(s)

TCON0 Register (0xE7)

Bit	Field	Type	Initial	Description
7	Reserved	R	0	
6..4	T0RATE[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_{Timer1} = fcpu / 12 1: $f_{Timer\ 1}$ = fosc / 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	T0GATE	R/W	0	Timer 0 gate control mode 0: Disable 1: Enable, Timer 0 clock source is gated by INT0
2	T0CT	R/W	0	Timer 0 clock source selection 0: f_{Timer0} = fcpu / 12 1: f_{Timer0} = fosc / T0RATE (refer to T0RATE) *(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_{EXT1} = fosc.

*(2) f_{EXT0} = fosc.

13.7 Sample Code

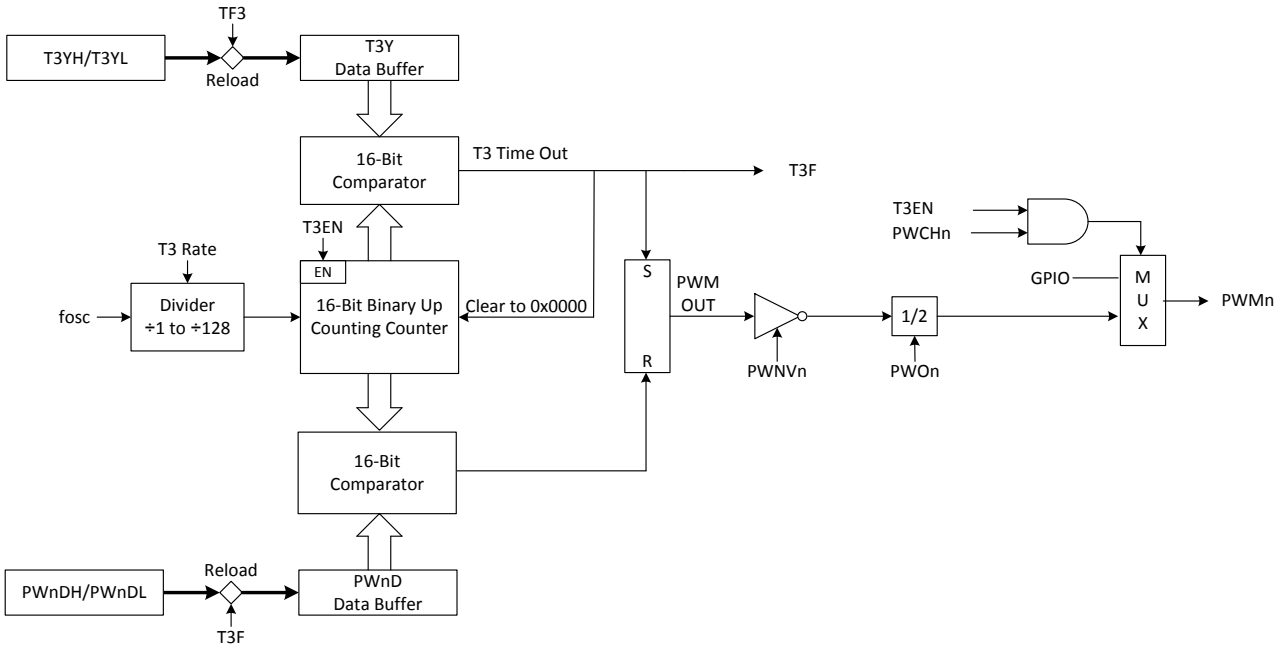
The following sample code demonstrates how to perform T0/T1 with interrupt.

```
#define T0Mode0     (0 << 0)  //T0 mode0, 13-bit counter
#define T0Mode1     (1 << 0)  //T0 mode1, 16-bit counter
#define T0Mode2     (2 << 0)  //T0 mode2, 8-bit auto-reload counter
#define T0Mode3     (3 << 0)  //T0 mode3, T0 two 8-bit counter/T1 no flag
#define T0GATE      (8 << 0)  //T0 gating clock by INT0
#define T0ClkFcpu    (0 << 0)  //T0 clock source from Fcpu/12
#define T0ClkExt     (4 << 0)  //T0 clock source from Fosc
#define T0ExtFosc    (0 << 4)  //T0 clock source from Fosc

#define T1Mode0     (0 << 4)  //T1 mode0, 13-bit counter
#define T1Mode1     (1 << 4)  //T1 mode1, 16-bit counter
#define T1Mode2     (2 << 4)  //T1 mode2, 8-bit auto-reload counter
#define T1Mode3     (3 << 4)  //T1 mode3, T1 stop
#define T1GATE      (8 << 4)  //T1 gating clock by INT1
#define T1ClkFcpu    (0 << 4)  //T1 clock source from Fcpu/12
#define T1ClkExt     (4 << 4)  //T1 clock source from Fosc

void InitT0T1(void)
{
  // T0/T1_Initial
  TH0 = 0x00;
  TL0 = 0x00;
  TH1 = 0x00;
  TL1 = 0x00;
  // T0 mode0 with gating clock by INT0, clock source from Fosc
  TMOD |= T0Mode0 | T0GATE | T0ClkExT;
  // T0 clock source = Fosc/1
  TCON0 |= 0x70;
  // T1 mode1, clock source from Fcpu/12
  TMOD |= T1Mode1 | T1ClkFcpu;
  // Timer 0/1 enable. Clear TF0/TF1
  TCON |= 0x50;
  // Enable T0/T1 interrupt
  IEN0 |= 0x0A;
  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M = 0x03;
}

void T0Interrupt(void) interrupt ISRTimer0  //0x0B
{ //TF0 clear by hardware
  P00 = ~P00;
}
void T1Interrupt(void) interrupt ISRTimer1  //0x1B
{ //TF1 clear by hardware
  P01 = ~P01;
}
```

14 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.

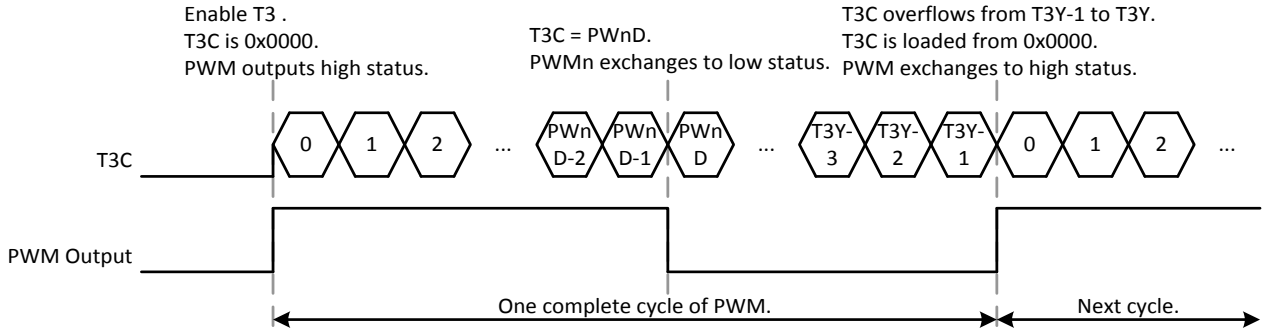
The PWM function has six programmable channels shared with GPIO pins and controlled by PWCH[5:0] bit. The output operation must be through enabled each bit/channel of PWCH[5:0] bits. The enabled PWM channel exchanges from GPIO to PWM output. When the PWCH[5:0] bits disables, the PWM channel returns to 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.



14.1 General PWM

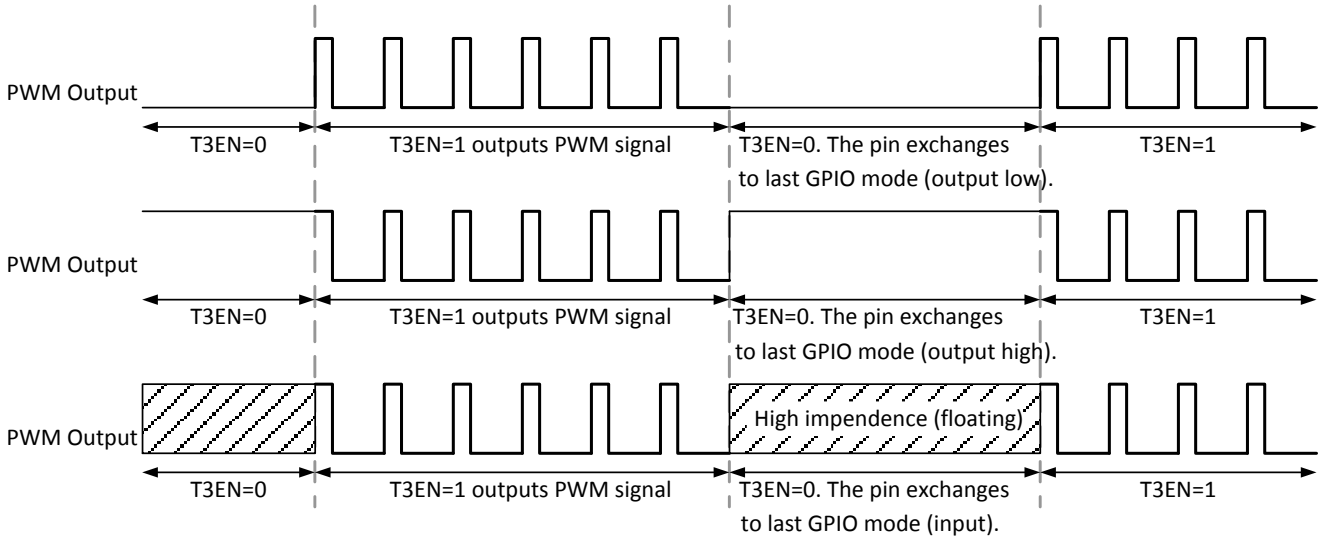
T3 timer builds in PWM function controlled by T3EN and PWCH[5:0] bits. PWM0 ~ PWM5 are output pins. Those output pins are shared with GPIO pin controlled by PWCH[5:0] bits. When output PWM function, we must be set T3EN =1. When PWM output signal synchronize finishes, the PWM channel exchanges from GPIO 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.



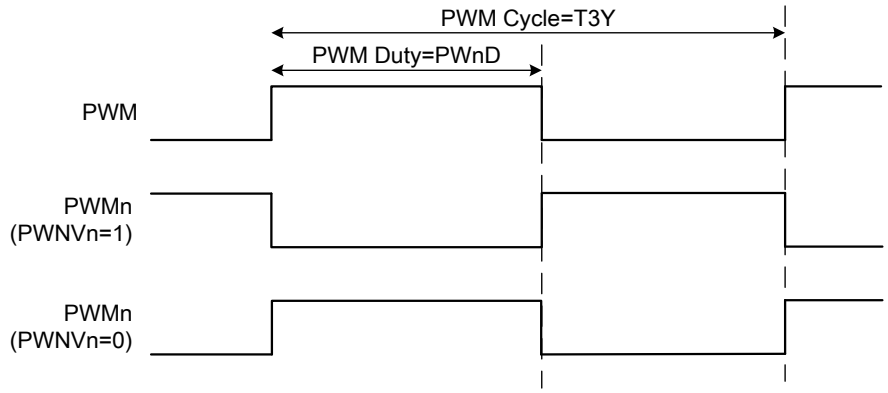
PWM Period = T3Y

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

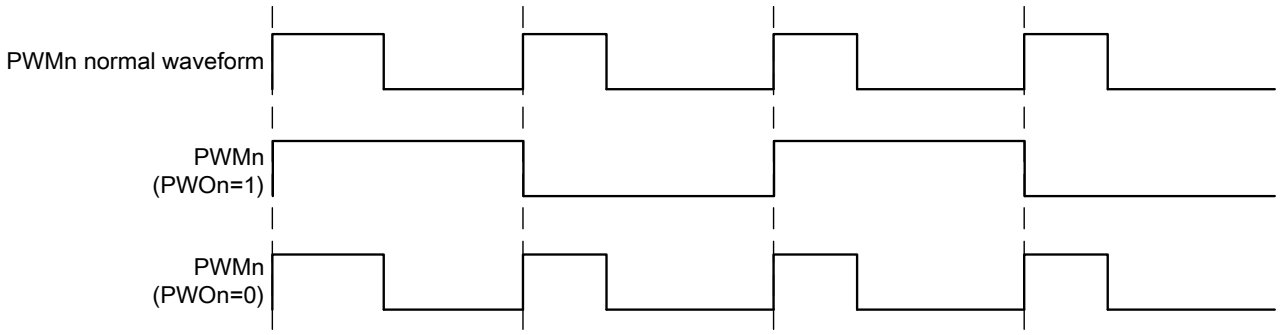


14.2 Inverse and Frequency Mode

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*frequency PWM signal. When PWOn=0, PWMn pin outputs 1*frequency PWM signal.



14.3 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)

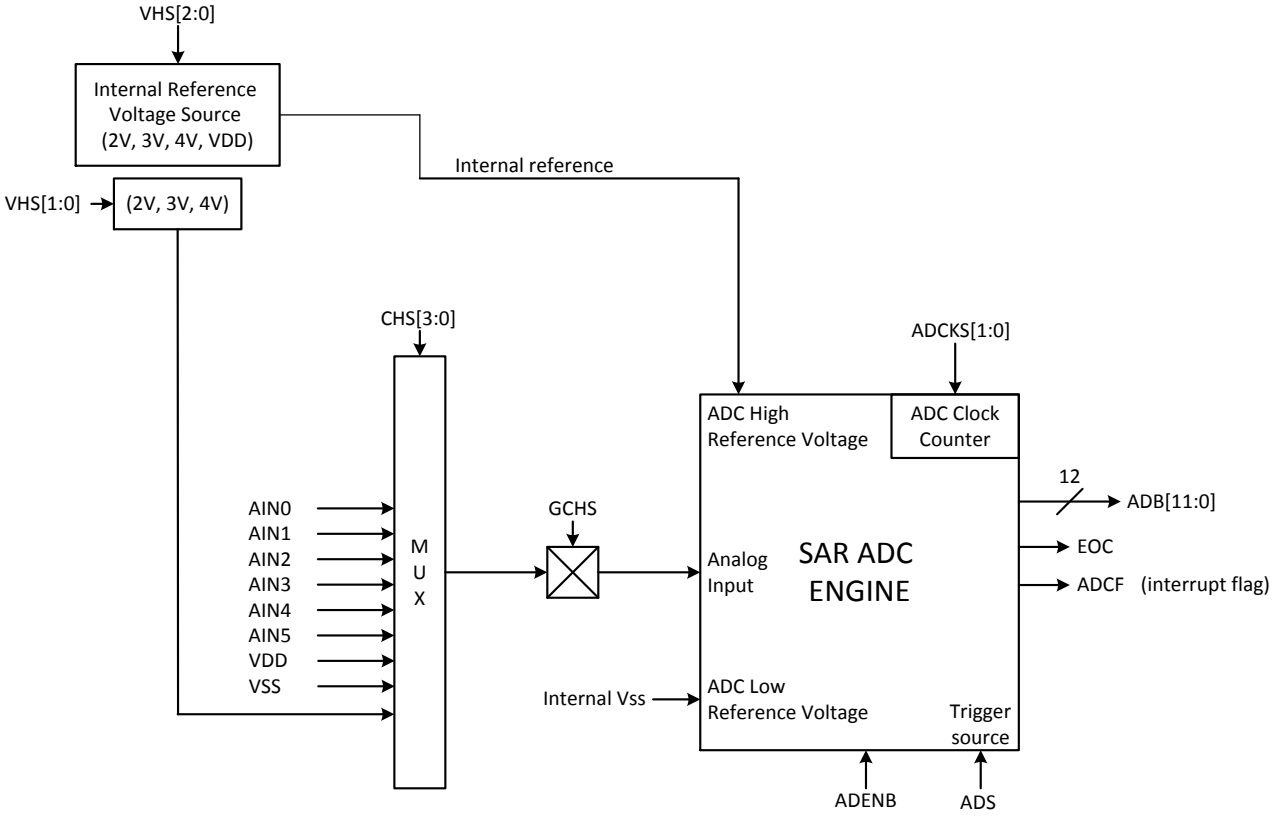
14.4 Sample Code

The following sample code demonstrates how to perform T3 with interrupt.

```
#define PWMInv0     (1 << 0)  //PWM0 output inverse
#define PWMInv1     (1 << 1)  //PWM1 output inverse
#define PWMInv2     (1 << 2)  //PWM2 output inverse
#define PWMInv3     (1 << 3)  //PWM3 output inverse
#define PWMInv4     (1 << 4)  //PWM4 output inverse
#define PWMInv5     (1 << 5)  //PWM5 output inverse
#define PWM2Frq0    (1 << 0)  //PWM0 output 2*frequency PWM
#define PWM2Frq1    (1 << 1)  //PWM1 output 2*frequency PWM
#define PWM2Frq2    (1 << 2)  //PWM2 output 2*frequency PWM
#define PWM2Frq3    (1 << 3)  //PWM3 output 2*frequency PWM
#define PWM2Frq4    (1 << 4)  //PWM4 output 2*frequency PWM
#define PWM2Frq5    (1 << 5)  //PWM5 output 2*frequency PWM
#define PWM0En      (1 << 0)  //Enable PWM0 output function
#define PWM1En      (1 << 1)  //Enable PWM1 output function
#define PWM2En      (1 << 2)  //Enable PWM2 output function
#define PWM3En      (1 << 3)  //Enable PWM3 output function
#define PWM4En      (1 << 4)  //Enable PWM4 output function
#define PWM5En      (1 << 5)  //Enable PWM5 output function
#define T3En        (1 << 7)  //Enable T3 function

void InitT3(void)
{
  // T3_Initial
  T3YH = 0x80;
  T3YL = 0x00;
  PW1DH = 0x40;
  PW1DL = 0x00;

  // PWM1 channel enable
  PWCH = PWM1En;

  // T3 enable, PWM1 output inverse, clock = Fosc/32
  T3M = T3En | PWMInv1 | 0x20;

  // Enable T3 interrupt & clear T3F
  IEN2 = 0x02;
  IRCON2 = 0x00;

  // Enable total interrupt
  IEN0 |= 0x80;

  P0 = 0x00;
  P0M |= 0x01;
}

void T3Interrupt(void) interrupt ISRT3  //0xEB
{ //T3F clear by hardware
    P00 = ~P00;
}
```

15 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 6 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.



15.1 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.

When ADC IP is enabled by ADENB bit, it is necessary to make an ADC start-up by program. 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. Clear ADCF by hardware automatically in interrupt procedure.

15.2 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.

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	-
0110	AIN6	VDD	Offset calibration
0111	AIN7	VSS	Offset calibration
1000	AIN8	Internal 2V or 3V or 4V	Battery detector channel
Others	-	-	Reserved

15.2.1 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.

15.3 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.

15.3.1 Signal Format

ADC sampling voltage range is limited by high/low reference voltage. The ADC low reference voltage is Vss 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) ≥ 2V". ADC low reference voltage is Vss = 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 ≤ ADC Sampled Input Voltage ≤ ADC High Reference Voltage

15.4 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 fosc and includes fosc/1, fosc/2, fosc/8 and fosc/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.

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

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/(16MHz/2/4)*16 = 8us	125kHz	1/(32MHz/2/4)*16 = 4us	250kHz

15.5 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 15-1 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

15.6 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
P0CON	-	-	P0CON5	P0CON4	P0CON3	P0CON2	P0CON1	P0CON0
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, 0110: VDD, 0111: VSS, 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 = 2.0V. 101: VREFH = VDD and AIN8 = 3.0V. 110: VREFH = VDD and AIN8 = 4.0V. 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.

P0CON Register (0x9E)

Bit	Field	Type	Initial	Description
5..0	P0CON[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	

* P0CON [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)

15.7 Sample Code

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

```
#define ADCAIN8_4V     (2 << 0)  //AIN8 = 4.0V
#define ADCAIN8_3V     (1 << 0)  //AIN8 = 3.0V
#define ADCAIN8_2V     (0 << 0)  //AIN8 = 2.0V
#define ADCInRefVDD    (1 << 2)  //internal reference from VDD
#define ADCSpeedDiv16  (0 << 4)  //ADC clock = fosc/16
#define ADCSpeedDiv8   (1 << 4)  //ADC clock = fosc/8
#define ADCSpeedDiv1   (2 << 4)  //ADC clock = fosc/1
#define ADCSpeedDiv2   (3 << 4)  //ADC clock = fosc/2
#define ADCChannelEn   (1 << 6)  //enable ADC channel
#define SelAIN5        (5 << 0)  //select ADC channel 5
#define ADCStart       (1 << 6)  //start ADC conversion
#define ADCEn          (1 << 7)  //enable ADC
#define EADC           (1 << 1)  //enable ADC interrupt
#define ClearEOC       0xDF;

unsigned int  ADCBuffer;    // data buffer

void ADCInit(void)
{
  P0  = 0x00;
  P0M = 0x10;
  // set AIN5 pin's mode at pure analog pin
  P0CON |= 0x20;     //AIN5/P05
  P0M   &= 0xDF;    //input mode
  P0UR  &= 0xDF;    //disable pull-high

  // configure ADC channel and enable ADC.
  ADM = ADCEn | SelAIN5;

  // enable channel and select conversion speed
  ADR = ADCChannelEn | ADCSpeedDiv1;

  // configure reference voltage
  VREFH = ADCInRefVDD;

  // enable gobal interrupt
  IEN0  |= 0x80;    //enable global interrupt

  // enable ADC interrupt
  IEN2  |= EADC;

  // start ADC conversion
  ADM   |= ADCStart;
}

void ADCInterrupt(void) interrupt ISRAdc
{//ADCF clear by hardware
    P04 = ~P04;

    ADCBuffer = (ADB << 4) + (ADR & 0x0F);
    ADM &= ClearEOC;
    ADM |= ADCStart;
}
```

16 UART

The UART provides a flexible full-duplex synchronous/asynchronous receiver/transmitter. The serial interface provides an up to 0.25MHz flexible full-duplex transmission. It can operate in four modes (one synchronous and three asynchronous). Mode0 is a shift register mode and operates as synchronous transmitter/receiver. In Mode1-Mode3 the UART operates as asynchronous transmitter/receiver with 8-bit or 9-bit data. The transfer format has start bit, 8-bit/ 9-bit data and stop bit. Transmission is started by writing to the S0BUF register. After reception, input data are available after completion of the reception in the S0BUF register. TB80/RB80 bit can be used as the 9th bit for transmission and reception in 9-bit UART mode. Programmable baud rate supports different speed peripheral devices.

The UART features include the following:

- Full-duplex, 2-wire synchronous/asynchronous data transfer.
- Programmable baud rate.
- 8-bit shift register: operates as synchronous transmitter/receiver
- 8-bit / 9-bit UART: operates as asynchronous transmitter/receiver with 8 or 9-bit data bits and programmable baud rate.

16.1 UART Operation

The UART UTX and URX pins are shared with GPIO. In synchronous mode, the UTX/URX shared pins must set output high by software. In asynchronous mode (8-bit/9-bit UART), the UTX shared pins must set output high and URX set input high by software. Thus, URX/UTX pins will transfers to UART purpose. When UART disables, the UART pins returns to GPIO last status.

The UTX/URX pins also support open-drain structure. The open-drain option is controlled by PnOC bit. When PnOC=0, disable UTX/URX open-drain structure. When PnOC=1, enable UTX/URX open-drain structure. If enable open-drain structure, UTX/URX pin must set high level (IO mode control will be ignored) and need external pull-up resistor.

The UART supports interrupt function. ES0 is UART0 transfer interrupt function control bit. UART transmitter and receiver interrupt function is controlled by ES0. When ES0 = 0, disable transmitter/receiver interrupt function. When ES0 = 1, enable UART transmitter/ receiver interrupt function. UART transmitter and receiver interrupt function are share interrupt vector 0x0023. When UART interrupt function enable, the program counter points to interrupt vector to do UART interrupt service routine after UART operating. TI0/RI0 is UART0 interrupt request flag, and also to be the UART operating status indicator when interrupt is disabled. TI0 and RI0 must clear by software.

UART provides four operating mode (one synchronous and three asynchronous) controlled by

S0CON register. These modes can be support in different baud rate and communication protocols.

SM0	SM1	Mode	Synchronization	Clock Rate	Start Bit	Data Bits	Stop Bit	UART pins' mode and data
0	0	0	Synchronous	Fcpu/12	X	8	X	UTX pin: P03M=1 and P03=1 URX pin: Transmitter: P02M=1 and P02=1 Receiver: P02M=0 and P02=1
0	1	1	Asynchronous	Baud rate generator or T1 overflow rate	1	8	1	UTX pin: P03M=1 and P03=1 URX pin: P02M=0
1	0	2	Asynchronous	Fcpu/64 or Fcpu/32	1	9	1	
1	1	3	Asynchronous	Baud rate generator or T1 overflow rate	1	9	1	

16.2 Mode 0: Synchronous 8-bit Receiver/Transmitter

Mode0 is a shift register mode. It operates as synchronous transmitter/receiver. The UTX pin output shift clock for both transmit and receive condition. The URX pin is used to transmit and receive data. 8-bit data will be transmit and receive with LSB first. The baud rate is fcpu/12. Data transmission is started by writing data to S0BUF register. In the end of the 8th bit transmission, the TI0 flag is set. Data reception is controlled by REN0 bit and clearing RI0 bits. When REN0=1 and RI0 is from 1 to 0, data transmission starts and the RI0 flag is set at the end of the 8th bit reception.

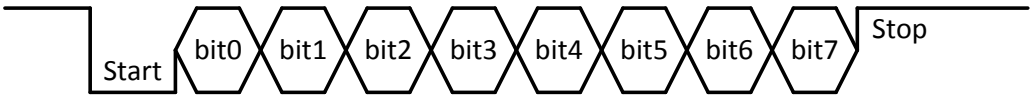
16.3 Mode 1: 8-bit Receiver/Transmitter with Variable Baud Rate

Mode1 supports an asynchronous 8-bit UART with variable baud rate. The transfer format includes 1 start bit, 8 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The baud rate clock source can be baud rate generator or T1 overflow controlled by BD bit. When BD=0, the baud rate clock source is from T1 overflow. When BD=1, the baud rate clock source is from baud rate generator controlled by S0RELH and S0RELL. Additionally, the baud rate can be doubled by SMOD bit.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted data into S0BUF, and then UART starts to transmit the pocket. The TI0 flag is set at the beginning of

the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, input data is stored in S0BUF register and the stop bit is stored in RB80.



16.4 Mode 2: 9-bit Receiver/Transmitter with Fixed Baud Rate

Mode2 supports an asynchronous 9-bit UART with fixed baud rate. The transfer format includes 1 start bit, 9 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The baud rate clock source is fixed to fcpu/64 or fcpu/32 and is controlled by SMOD bit. When SMOD=0, baud rate is fcpu/64. When SMOD=1, baud rate is fcpu/32.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted data into S0BUF, and then UART starts to transmit the pocket. The 9th data bit is taken from TB80. The TI0 flag is set at the beginning of the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, lower 8-bit input data is stored in S0BUF register and the 9th bit is stored in RB80.

16.5 Mode 3: 9-bit Receiver/Transmitter with Variable Baud Rate

Mode3 supports an asynchronous 9-bit UART with variable baud rate. The transfer format includes 1 start bit, 9 data bits (LSB first) and 1 stop bit. Data is transmitted by UTX pin and received by URX pin. The different between Mode2 and Mode3 is baud rate selection. In the Mode3, the baud rate clock source can be baud rate generator or T1 overflow controlled by BD bit. When BD=0, the baud rate clock source is from T1 overflow. When BD=0, the baud rate clock source is from baud rate generator controlled by S0RELH and S0RELL. Additionally, the baud rate can be doubled by SMOD bit.

Data transmission is controlled by REN0 bit. After transmission configuration, load transmitted

data into S0BUF, and then UART starts to transmit the pocket. The 9th data bit is taken from TB80. The TI0 flag is set at the beginning of the stop bit.

Data reception is controlled by REN0 bit. When REN0=1, data reception function is enabled. Data reception starts by receiving the start bit for master terminal, URX detects the falling edge of start bit, and then the RI0 flag is set in the middle of a stop bit. Until reception completion, lower 8-bit input data is stored in S0BUF register and the 9th bit is stored in RB80.

16.6 Multiprocessor Communication

UART supports multiprocessor communication between a master device and one or more slaver device in Mode2 and Mode3 (9-bit UART). The master identifies correct slavers by using the 9th data bit. When the communication starts, the master transmits a specific address byte with the 9th bit is set “1” to selected slavers, and then transmits a data byte with the 9th bit is set “0” in the following transmission.

Multiprocessor communication is controlled by SM20 bit. When SM20=0, disable multiprocessor communication. When SM20=1, enable multiprocessor communication. If SM20 is set, the UART reception interrupt is only generated when the 9th received bit is “1” (RB80). The slavers will compare received data with its own address data by software. If address byte is match, the slavers clear SM20 bit to enable interrupt function in the following data transmission. The slavers with unmatched address, their SM20 keep in “1” and will not generate interrupt in the following data transmission.

16.7 Baud Rate Control

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

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

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

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

Table 16-1 Recommended Setting for Common UART Baud Rates (fcpu = 8 MHz)

Baud Rate	SMOD	S0RELH	S0RELL	Accuracy
4800	0	0x03	0xE6	0.16 %
9600	0	0x03	0xF3	0.16 %
19200	0	0x03	0xF3	0.16 %
38400	0	0x03	0xF9	-6.99 %
56000	1	0x03	0xFB	-10.71 %
57600	1	0x03	0xFC	8.51 %
115200	1	0x03	0xFE	8.51 %
128000	1	0x03	0xFE	-2.34 %
250000	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.

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

Table 16-2 Recommended Setting T1 overflow period (T1 clock=32M) for Common UART Baud Rates (fcpu = 8 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 %

✷ ***Note:***

1. When baud rate generator source is T1 overflow rate, the max counter value is 0xFB. (Only supports 0x00~0xFB).

2. When baud rate generator source is T1 overflow rate, the system clock fcpu must be greater four times to T1 overflow rate.

16.8 Power Saving

The UART module has clock gating function for saving power. When REN0 bit is 0, the UART module internal clocks are halted to reduce power consumption. UART relevant register (S0CON, S0CON2, S0BUF, S0RELL, S0RELH and SMOD bit) are unable to access.

Conversely, when REN0 bit is 1, UART internal clocks are run, and registers can access. The REN0 bit must be set to 1, before the initial setting UART.

16.9 UART Registers

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
S0CON2	BD	-	-	-	-	-	-	-
S0BUF	S0BUF7	S0BUF6	S0BUF5	S0BUF4	S0BUF3	S0BUF2	S0BUF1	S0BUF0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
S0RELH	-	-	-	-	-	-	S0REL9	S0REL8
S0RELL	S0REL7	S0REL6	S0REL5	S0REL4	S0REL3	S0REL2	S0REL1	R0REL0
IEN0	EAL	-	-	ES0	ET1	EX1	ET0	EX0
P0OC	-	-	P05OC	P04OC	P03OC	P02OC	P01OC	P00OC
P0M	-	-	P05M	P04M	P03M	P02M	P01M	P00M
P0	-	-	P05	P04	P03	P02	P01	P00

S0CON Register (0x98)

Bit	Field	Type	Initial	Description
7..6	SM[0:1]	R/W	00	UART mode selection 00: Mode 0 01: Mode 1 10: Mode 2 11: Mode 3
5	SM20	R/W	0	Multiprocessor communication (mode 2, 3) 0: Disable 1: Enable
4	REN0	R/W	0	UART module (and reception function) 0: Disable for power saving* 1: Enable for UART operating
3	TB0	R/W	0	The 9th bit transmission data (mode 2, 3)
2	RB0	R/W	0	The 9th bit data from reception

1	TI0	R/W	0	UART interrupt flag of transmission
0	RI0	R/W	0	UART interrupt flag of reception

* When REN0 bit is 0, UART relevant register are unable to access, and the module internal clocks are halted.

✷ ***Note: TI0 and RI0 are clear by software when interrupt is enabled.***

S0CON2 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 S0RELH, S0RELL registers
6..0	Reserved	R	0x00	

S0BUF Register (0x99)

Bit	Field	Type	Initial	Description
7..0	S0BUF	R/W	0x00	Action of writing data triggers UART communication (LSB first). Reception data is available to read by the end of packages.

PCON Register (0x87)

Bit	Field	Type	Initial	Description
7	SMOD	R/W	0	UART baud rate control In UART mode 0: Unused. In UART mode 1, 3: The baud rate is generated as the equation in section 16.7 (Baud Rate Control). In UART mode 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.

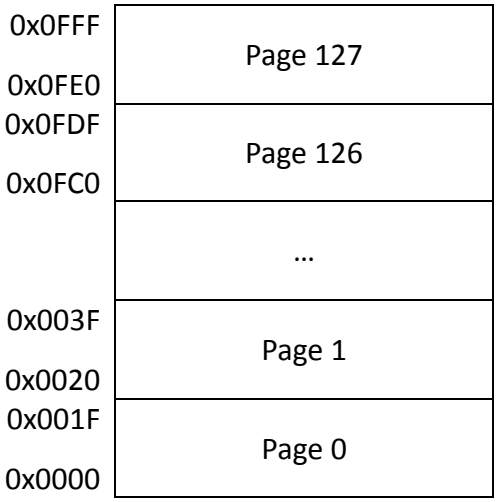
16.10 Sample Code

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

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

void SYSUartInit(void)
{
  // set UTX, URX pins' mode at here or at GPIO initialization
  P03 = 1;
  P0M = P0M | 0x08 & ~0x04;

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

  // configure UART baud rate
  PCON = SYSUartSMODE1;
  S0CON2 = SYSUartBD1;
  S0RELH = 0x03;
  S0RELL = 0xFE;

  // enable UART interrupt
  IEN0 |= SYSUartES0;

  // send first UTX data
  S0BUF = uartTxBuf;
}

void SYSUartInterrupt(void) interrupt ISRUart
{
  if (TI0 == 1) {
    S0BUF = uartTxBuf;
    TI0 = 0;
  } else if (RI0 == 1) {
    uartRxBuf = S0BUF;
    RI0 = 0;
  }
}
```

17 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.



Program memory (IROM)

17.1 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.

ISP ROM MAP		ROM address bit0~bit4 (hex) =0
ROM address bit5~bit15 (hex)	0000	These pages include reset vector and interrupt sector. We strongly recommend to reserve the area not to do ISP erase.
	0020	
	0040	
	…	
	00C0	
	00E0	
	0100	One ISP Program Page
	0120	One ISP Program Page
	…	One ISP Program Page
	0300	One ISP Program Page
	0320	One ISP Program Page
	…	One ISP Program Page
	0700	One ISP Program Page
	0720	One ISP Program Page
	…	One ISP Program Page
	0FE0	This page includes ROM reserved area. We strongly recommend to reserve the area not to do ISP erase.

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.
5. Write ‘NOP’ instruction twice.

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.

17.2 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.
5. Write ‘NOP’ instruction twice.

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 EPRAM register. Subsequently, write ‘0x1E’ into PECMD [7:0] registers to duplicate the buffer’s data to the address 0x0FC5 of IROM.

✸ ***Note:***

1. Watch dog timer should be clear before the Flash write (program) operation, or watchdog timer would overflow and reset system during ISP operating.

2. Don't execute ISP flash ROM program operation for the first page and the last page, or affect program operation.

17.3 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.

17.4 Sample Code

```
unsigned cahr idata dataBuffer[32] _at_ 0xE0;  // IRAM 0xE0 to 0xFF

void SYSIspSetDataBuffer(unsigned char address, unsigned char data)
{
  dataBuffer[address & 0x1F] = data;
}

void SYSIspStart(unsigned int pageAddress)
{
  ISP(pageAddress, 0xE0);  //Page program
}

void SYSByteIspStart(unsigned int byteAddress)
{
  BISP(byteAddress, 0xE0);  //Byte program
}
```

18 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.

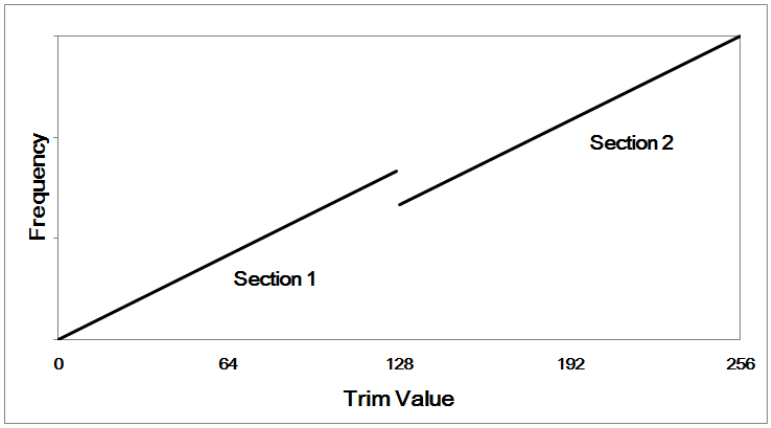
18.1 Clock Trim Section

Clock fine-tuning consists of 8 trim sections as Table 18-1. 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 18-1 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.



18.2 Clock Fine-Tuning Procedure

Theses 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 Table18-1.
5. Write the new clock trim value to FRQ[9:0].
6. Write '0x3C' into FRQCMD [7:0] to trigger clock fine-tuning function.

✸ ***Note: Please check IROM fetching cycle < 8MHz to avoid system error.***

18.3 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.

18.4 Sample Code

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

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

19 Electrical Characteristics

19.1 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

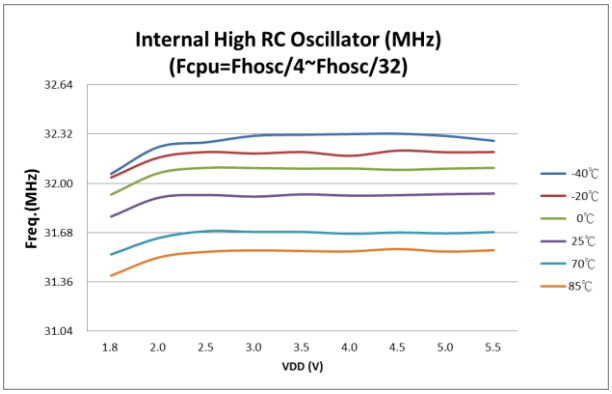
19.2 System Operation Characteristics

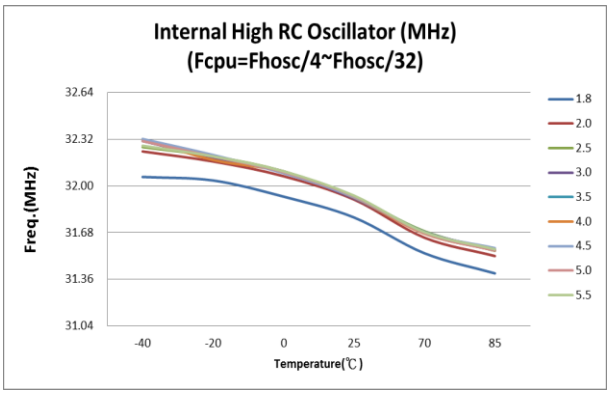
	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	Operating voltage	fcpu = 1MHz	1.8		5.5	V
V_{DR}	RAM data retention Voltage		1.5			V
V_{POR}	VDD rising rate*		0.05			V/ms
I_{DD1}	Normal mode supply current	VDD = 3V, fcpu = 1MHz		2.25		mA
		VDD = 5V, fcpu = 1MHz		2.30		mA
		VDD = 3V, fcpu = 4MHz		2.60		mA
		VDD = 5V, fcpu = 4MHz		2.65		mA
		VDD = 3V, fcpu = 8MHz		3.10		mA
		VDD = 5V, fcpu = 8MHz		3.15		mA
I_{DD2}	STOP mode supply current	VDD = 3V		3.0	8.5	µA
		VDD = 5V		3.5	9.0	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, -40°C to 85°C	31.36	32	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

  The IHRC Graphs are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range.





19.3 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	kΩ
		VDD = 5V	50	100	150	kΩ
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.

19.4 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			µs
DNL	Differential nonlinearity*	f_{ADSMP} = 62.5kHz		±1		LSB
		f_{ADSMP} = 250kHz		±1		LSB
		f_{ADSMP} = 500kHz		±8		LSB
INL	Integral Nonlinearity*	f_{ADSMP} = 62.5kHz		±2		LSB
		f_{ADSMP} = 250kHz		±2.5		LSB
		f_{ADSMP} = 500kHz		±8		LSB
NMC	No missing code*	f_{ADSMP} = 62.5kHz	10	11	12	Bit
		f_{ADSMP} = 250kHz		11		Bit
		f_{ADSMP} = 500kHz		8		Bit
V_{OFFSET}	Input offset voltage	Non-trimmed	-10	0	10	mV

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

19.5 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.

20 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

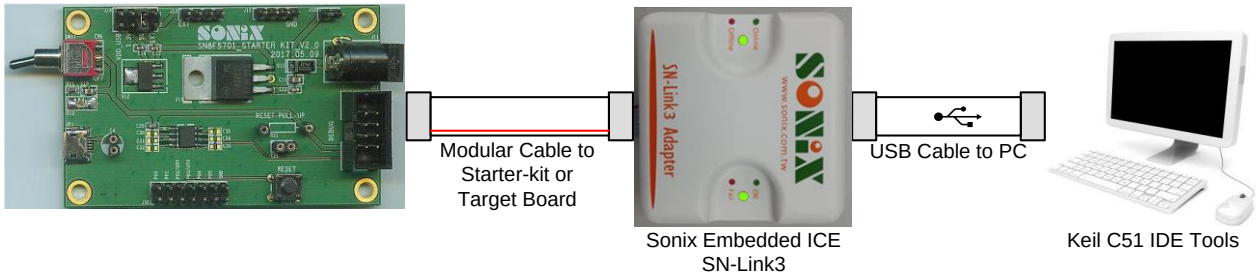
Mnemonic	Description
CLR C	Clear carry flag
CLR bit	Clear directly addressed bit
SETB C	Set carry flag
SETB bit	Set directly addressed bit
CPL C	Complement carry flag
CPL bit	Complement directly addressed bit
ANL C, bit	AND directly addressed bit to carry flag
ANL C, /bit	AND complement of directly addressed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

21 Development Environment

SONIX provides an Embedded ICE emulator system to offer SN8F5701 firmware development. The platform is an in-circuit debugger and controlled by Keil C51 IDE software on Microsoft Windows platform. The platform includes SN-Link3, SN8F5701 Starter-kit and Keil C51 IDE software to build a high-speed, low cost, powerful and multi-task development environment including emulator, debugger and programmer. To execute emulation is like run real chip because the emulator circuit integrated in SN8F5701 to offer a real development environment.



21.1 Minimum Requirement

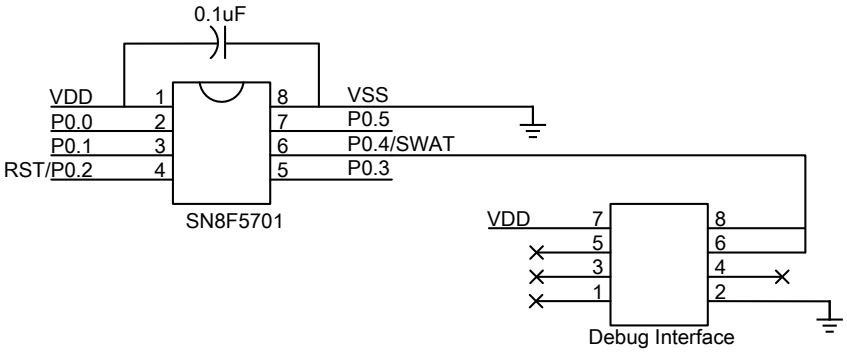
The following items are essential to build up an appropriate development environment. The compatibility is verified on listed versions, and is expected to execute perfectly on later version. SN-Link related information is available to download on SONiX website (www.sonix.com.tw);
Keil C51 is downloadable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

- **SN-Link3 Adapter** with updated firmware version 1.02
- **SN-Link Driver for Keil C51** version 1.00.317
- **Keil C51** version 9.50a and 9.54a or greater.

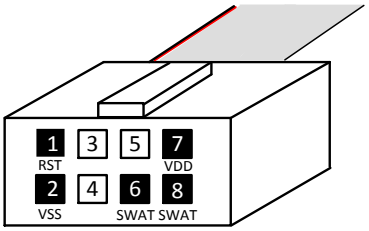
21.2 Debug Interface Hardware

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

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



Example circuit

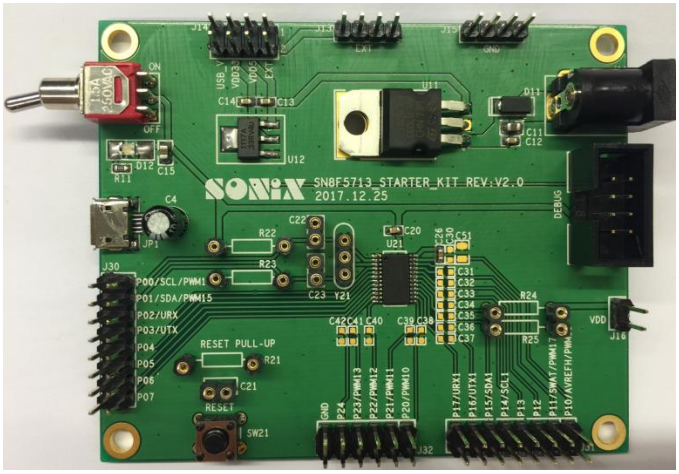


SN-Link header

21.3 Development Tool

SN-Link3 Adapter

Starter-Kit support SN8F5701, SN8F57011



MP5 Writer

22 SN8F5701 Starter-Kit

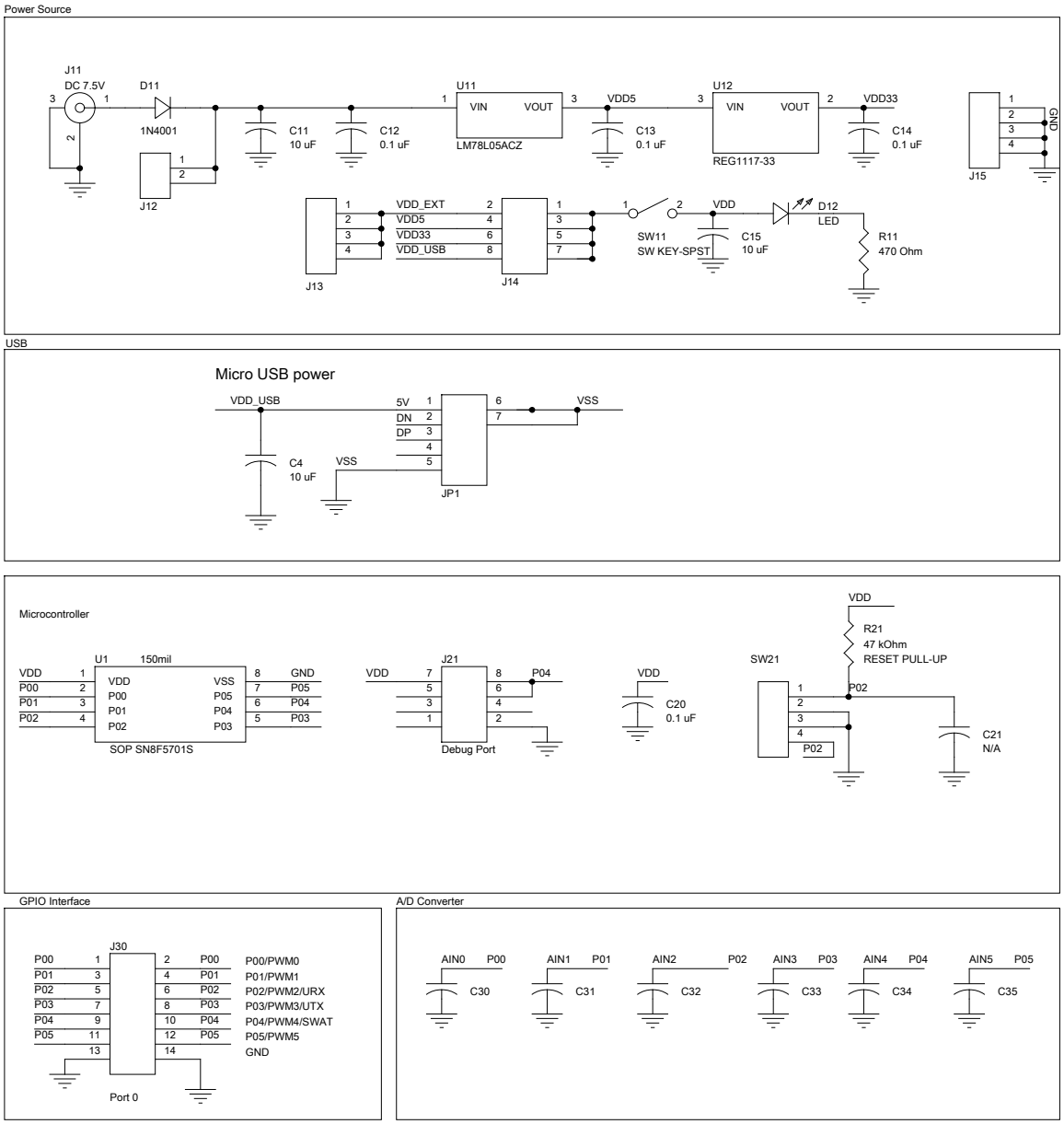
SN8F5000 Starter-Kit provides easy-development platform. It includes SN8F5000 family real chip and I/O connectors to input signal or drive device of user's application. It is a simple platform to develop application as target board not ready. The Starter-Kit can be replaced by target board, because SN8F5000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

22.1 Configurations of Circuit

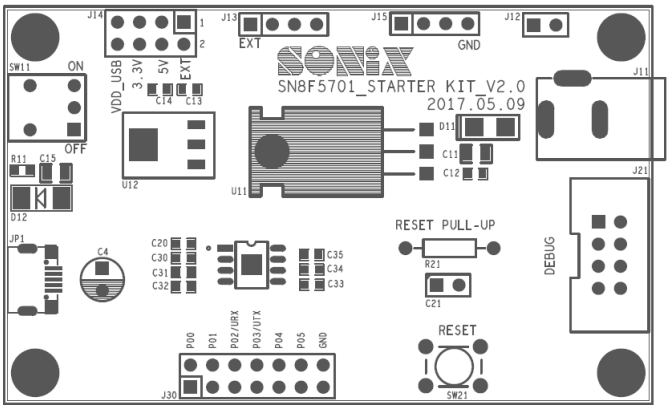
These configurations must be setup completely before starting Starter-Kit developing.

1. Confirm to the circuit board whether elements are complete.
2. The power source of Starter-Kit circuit is chosen from 5.0V, 3.3V, external power or Micro USB via jumper.
3. The power source comes from 5.0V or 3.3V which must be connect to DC 7.5V power adapter.
4. If the power source is chosen from external power, then external power source connects to EXT pin.
5. The "RST" pin needs to connect pull high resister to VDD when external reset is chosen to use.
6. The Debug Port can connect SN-LINK Adapter for emulation or download code.
7. The MCU LED will light up and SN8F5000 family chip will be connected to power when power (VDD) is switched on.

22.2 Schematic



22.3 Floor Plan of PCB layout



22.4 Component Description

Number	Description
C30 – C35	6-ch ADC capacitors.
D12	MCU LED
J11	DC 7.5V power adapter
J13/J15	External power source.
SW21	External reset trigger source
J14	VDD power source is 5.0V, 3.3V or external power.
J21	Debug Port
J30	I/O connector.
R21, C21	External reset pull-high resister and capacitor.
SW11	Target power (VDD) switch
U1	SN8F5701S real chip (Sonix standard option).
JP1	Micro USB port

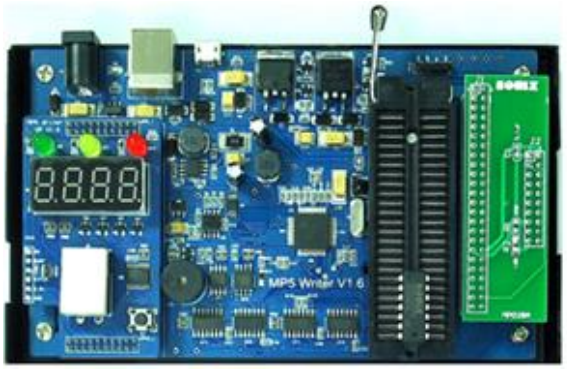
23 ROM Programming Pin

SN8F5710 Series Flash ROM erase/program/verify support SN-Link and MP5 Writer

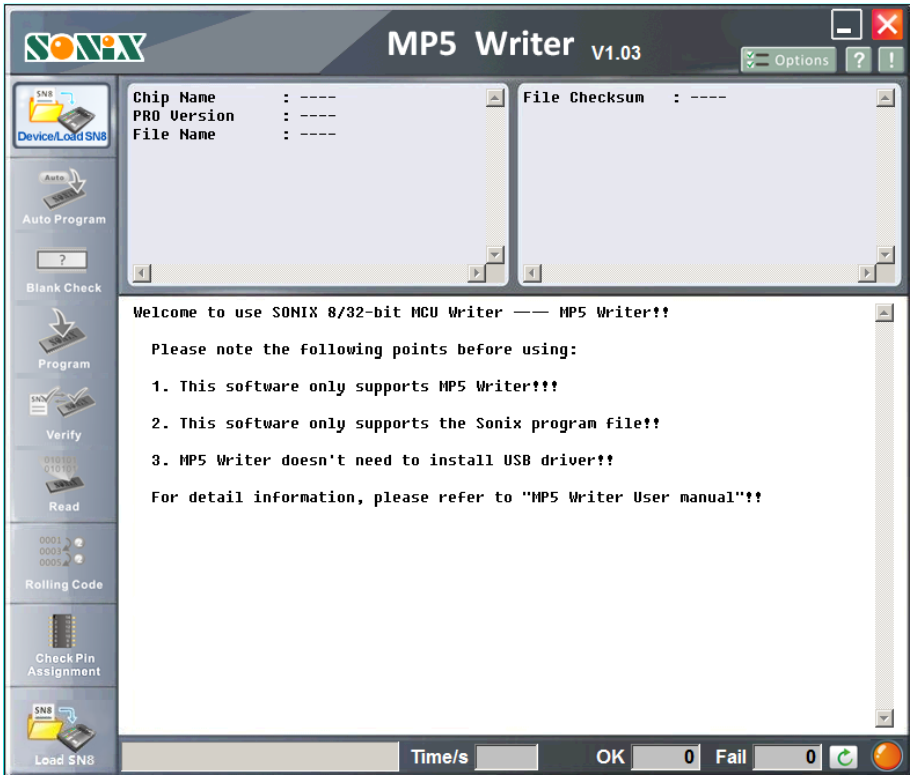
- SN-Link: Debug interface and on board programming.
- MP5 Writer: For SN8F5710 series version mass programming.

23.1 MP5 Hardware Connecting

Different package type with MCU programming connecting is as following, DIP and SOP Illustration.

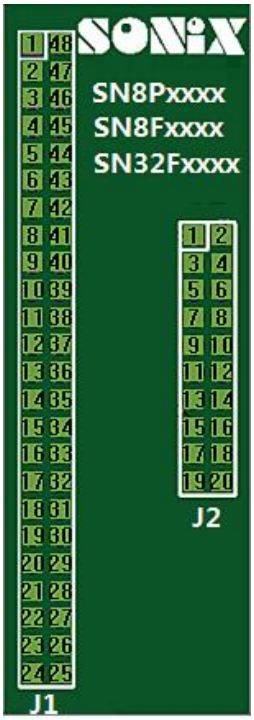


MP5 Software operation interface is as following.



23.2 MP5 Writer Transition Board Socket Pin Assignment

MP5 Writer Transition Board:



23.3 MP5 Writer Programming Pin Mapping

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

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

23.3.1 Normal Mode

Writer Connector		MCU Pin Number	SN8F5701P/S/T		SN8F57011D					
J2 Pin Number	J2 Pin Name		MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number
1	VDD	VDD	1	21	6	27				
2	GND	VSS	8	28	2	23				
7	SWAT	P0.4	6	26	1	22				
9	SWAT	P0.4	6	26	1	22				
20	PDB	P0.3	5	25	3	24				

23.3.2 High Speed Mode

Writer Connector		MCU Pin Number	SN8F5701P/S/T							
J2 Pin Number	J2 Pin Name		MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number	MCU Pin Number	J1 Pin Number
1	VDD	VDD	1	21						
2	GND	VSS	8	28						
3	DFTCLK	P0.0	2	22						
5	SEL	P0.1	3	23						
7	SWAT	P0.4	6	26						
9	SWAT	P0.4	6	26						
11	DAH	P0.2	4	24						
13	DAL	P0.5	7	27						
20	PDB	P0.3	5	25						

23.4 SN-Link ISP Programming

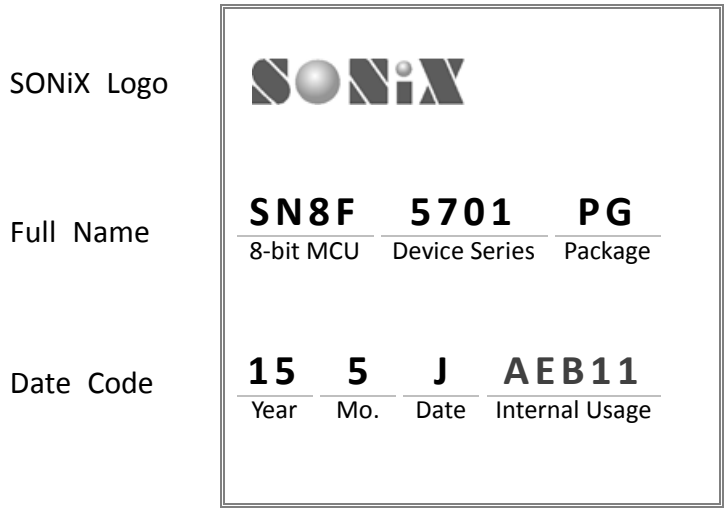
SN-Link ISP programming hardware and software are as following.

23.5 SN-Link ISP Programming Pin Mapping

SN-Link Connector		MCU Pin Number	SN8F5701P/S/T	SN8F57011D		
Pin Number	Pin Name		Pin Number	Pin Number	Pin Number	Pin Number
7	VDD	VDD	1	6		
2	GND	VSS	8	2		
6	SWAT	P0.4	6	1		
8	SWAT	P0.4	6	1		

24 Ordering Information

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



24.1 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
SN8F57011DGR	SOT23-6L, 6 pins, TR Green package

✷ ***Note: TR package (Tape & Reel Packing) will add "R" character after the package name.***

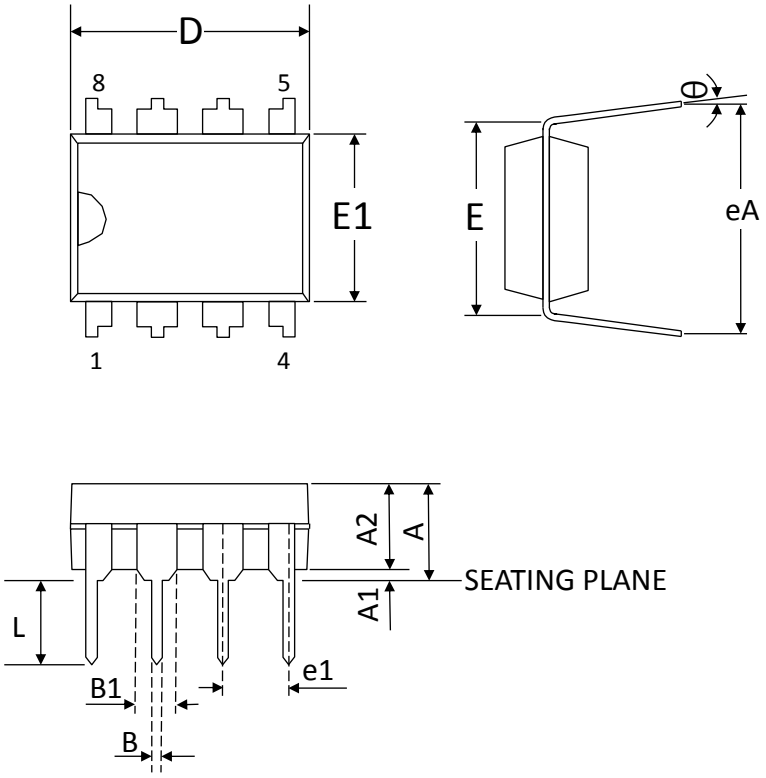
24.2 Date Code

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

25 Package Information

25.1 P-DIP8

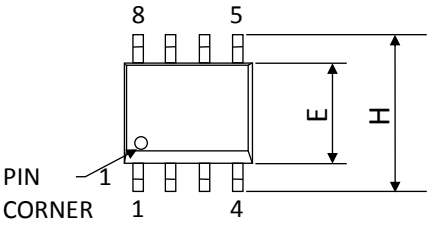


SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	5.33	--	--	0.210
A1	0.38	--	--	0.015	--	--
A2	3.18	3.30	3.43	0.125	0.130	0.135
B	0.46 typ.			0.018 typ.		
B1	1.52 typ.			0.060 typ.		
D	9.02	9.27	10.16	0.355	0.365	0.400
E	7.62 BSC.			0.300 BSC		
E1	6.22	6.35	6.47	0.245	0.250	0.255
e1	2.54 typ.			0.100 typ.		
L	2.92	3.30	3.81	0.115	0.130	0.150
eA	7.62	9.02	9.53	0.300	0.355	0.375
θ	0°	7°	15°	0°	7°	15°

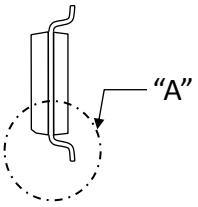
Notes：

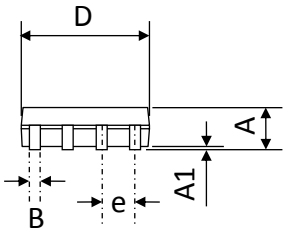
1. JEDEC OUTLINE：MS-001 BA
2. CONTROLLING DIMENSION：inch

25.2 SOP8

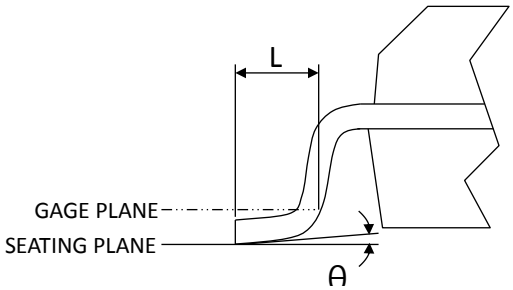


TOP VIEW





SIDE VIEW



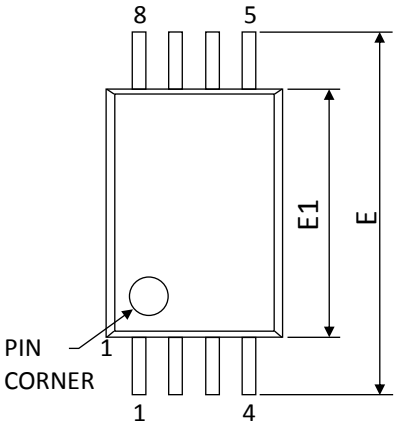
DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.75	--	--	0.069
A1	0.10	--	0.25	0.004	--	0.010
B	0.31	--	0.51	0.012	--	0.020
D	4.90 BSC			0.193 BSC		
E	3.90 BSC			0.153 BSC		
e	1.27 BSC			0.050 BSC		
H	6.00 BSC			0.236 BSC		
L	0.40	--	1.27	0.016	--	0.050
θ	0°	--	8°	0°	--	8°

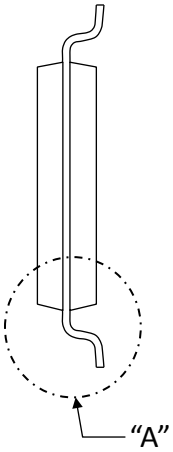
Notes：

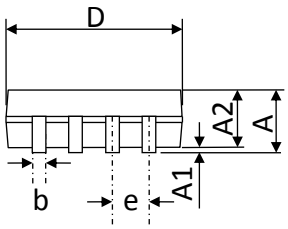
1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE : MS-012 AA

25.3 TSSOP8

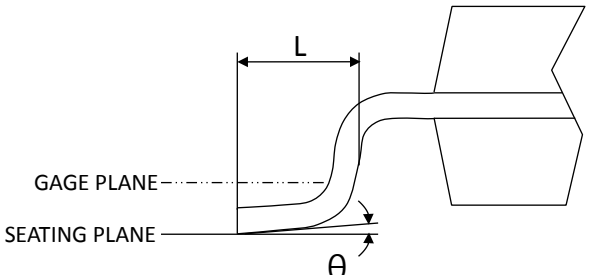


TOP VIEW





SIDE VIEW



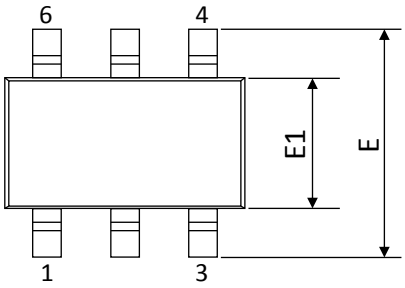
DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.20	--	--	0.047
A1	0.05	--	0.15	0.002	--	0.006
A2	0.80	1.00	1.05	0.031	0.039	0.041
b	0.22 typ.			0.009 typ.		
D	2.90	3.00	3.10	0.114	0.118	0.122
E	6.40 BSC.			0.252 BSC.		
E1	4.30	4.40	4.50	0.169	0.173	0.177
e	0.65 typ.			0.026 BSC.		
L	0.45	--	0.75	0.018	--	0.030
θ	0°	--	8°	0°	--	8°

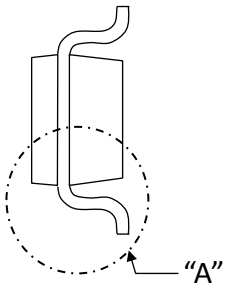
Notes：

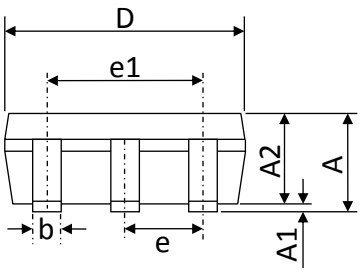
1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE : MO-153
3. DIMENSION 'D' DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BERRES.
4. DIMENSION 'E1' DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
5. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION.

25.4 SOT23-6L

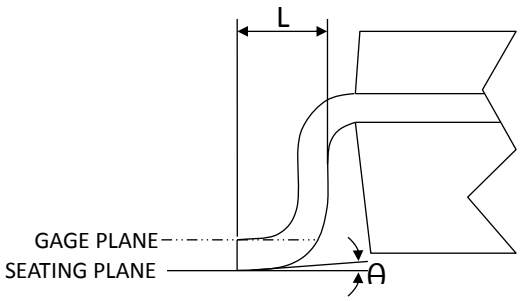


TOP VIEW





SIDE VIEW



DETAIL "A"

SYMBOLS	Dimension in mm			Dimension in inch		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	--	--	1.45	--	--	0.057
A1	0.00	--	0.15	0.000	--	0.006
A2	0.90	1.15	1.30	0.035	0.045	0.051
b	0.30	--	0.50	0.012	--	0.020
D	2.90 BSC.			0.114 BSC.		
E	2.80 BSC.			0.110 BSC.		
E1	1.60 BSC.			0.063 BSC.		
e	0.95 BSC.			0.037 BSC.		
e1	1.90 BSC.			0.075 BSC.		
L	0.30	--	0.60	0.012	--	0.024
θ	0°	--	8°	0°	--	8°

Notes：

1. CONTROLLING DIMENSION：mm
2. JEDEC OUTLINE : MS-178 AB

26 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