SONiX Technology Co., Ltd.

SN8F5910 Series Datasheet

8051-based Microcontroller

SN8F5918FG

SN8F5919FG

1 简介

1.1 功能特性

◆ 增强型 8051，减少指令周期时间（最高达到 80C51 的 12 倍）
- 最高达 32MHz 可调的 CPU 频率
- 内部 32MHz 时钟发生器（IHRC）
- 实时时钟，带 32.768KHz 晶振
- Fcpu：8MIPs~0.25MIPs

◆ 存储器
- 32KB Flash 存储器（IROM），支持总线编程功能
- 256 字节内部 RAM（IRAM）
- 512 字节外部 RAM（XRAM）

◆ I/O 引脚：10 个纯 I/O 引脚，12 个与 LCD 功能共用的 I/O 引脚

◆ 中断：8 个中断源，可控制其优先权，且对应唯一的中断向量
- 6 个内部中断：T0，TC0，TC1，ADC，UART，I2C
- 2 个外部中断：INT0，INT1

◆ 硬件乘法/除法单元和 2 组 DPTR

◆ 定时器系统
- 1 组 8 位基本定时器 T0，带 RTC
- 2 组 16 位定时器/计数器，带自动装载，PWM，蜂鸣器输出（TC0~TC1）
- 1 个蜂鸣器引脚（P12）：1KHz/2KHz/4KHz/8KHz

◆ UART/I2C 接口，支持 SMBus

◆ 比较器低电压检测 2.2V~3.6V（每阶 0.2V）

◆ 电源：2.0~5.5V（AVDD/DVDD）

◆ 升压稳压器：
- 输入电压：2.0V~5.5V
- AVDDR 稳压输出为模拟电路供电（ADC/PGIA/OP...）
- AVDDR 电压选项：2.7V/3.0V/3.3V/3.6V
- ACM 1V 为 ADC 通用电压

◆ PGIA：可编程控制增益放大器，高输入阻抗
- 增益：1x/4x/8x/16x/32x/64x/128x

◆ 24 位 Delta Sigma ADC：
- 单一或者差分通道配置
- 10 个外部输入通道：AI1~AI10
- 6 个内部通道：1/8*VDD 检测，温度传感器，ACM，GND 和 BFOUT，VBIAS
- 内部或者外部参考电压
- 转换输出速率：3.8Hz~5.2KHz

◆ BIA 电路：
- 正弦发生器
- 电流源发生器

◆ LCD 驱动：C-Type LCD 驱动，高达 204 点
- 4com 或 6com，1/3 bias
- 4*36 或者 6*34 dots，LCD 引脚与 IO 引脚共用
- 可调节的 VLCD 输出 2.6V~4.5V

◆ 片上调试
- 单线调试接口
- 5 个硬件断点
- 无限次软件断点
- ROM 数据安全保护
- 看门狗和可编程的外部复位

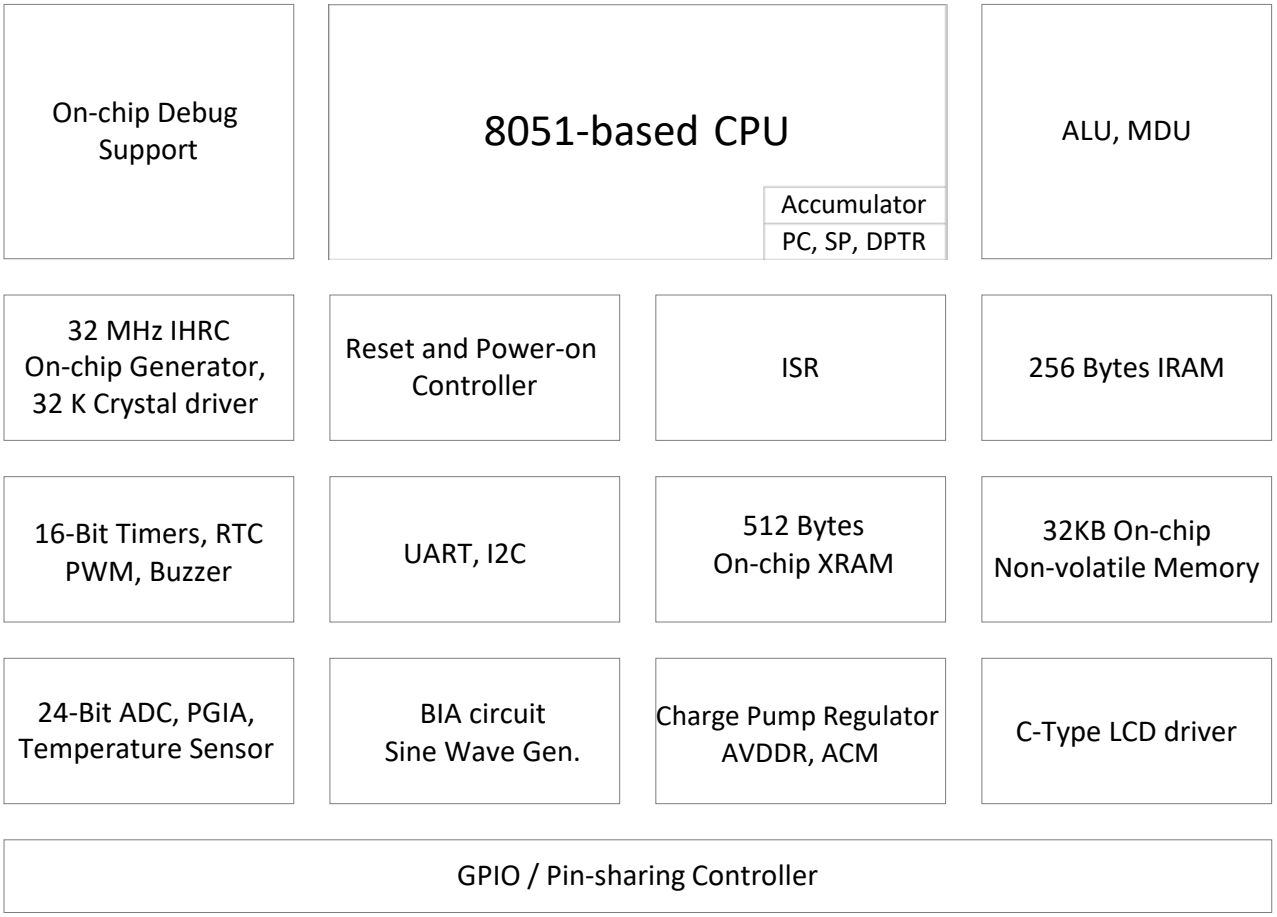
1.2 应用领域

- 体脂监测仪
- 血压计
- 其它测量应用
- 温度计
- 体重

1.3 产品性能表

	I/O	LCD	RTC	16-Bit Timer / PWM	I2C	SPI	UART	ADC	OPA	CMP	Int. INT	Ext. INT	Package Types
SN8F5918FG	22	4x28, 6x26	V	2	V	-	V	10-Ch	6	1	6	2	LQFP64
SN8F5919FG	22	4x36, 6x34	V	2	V	-	V	10-Ch	6	1	6	2	LQFP80

1.4 框图



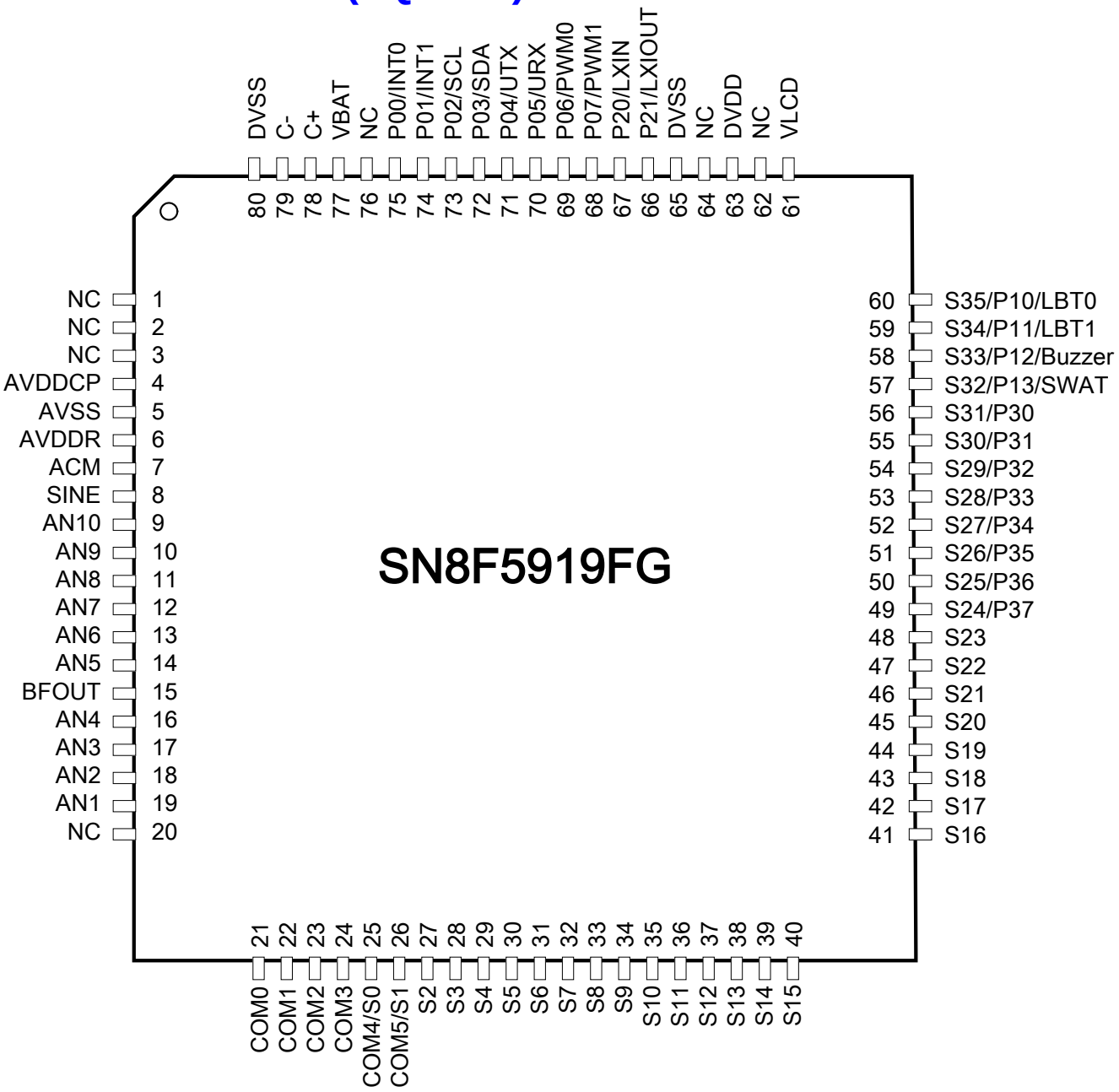
2 目录

3 修订记录

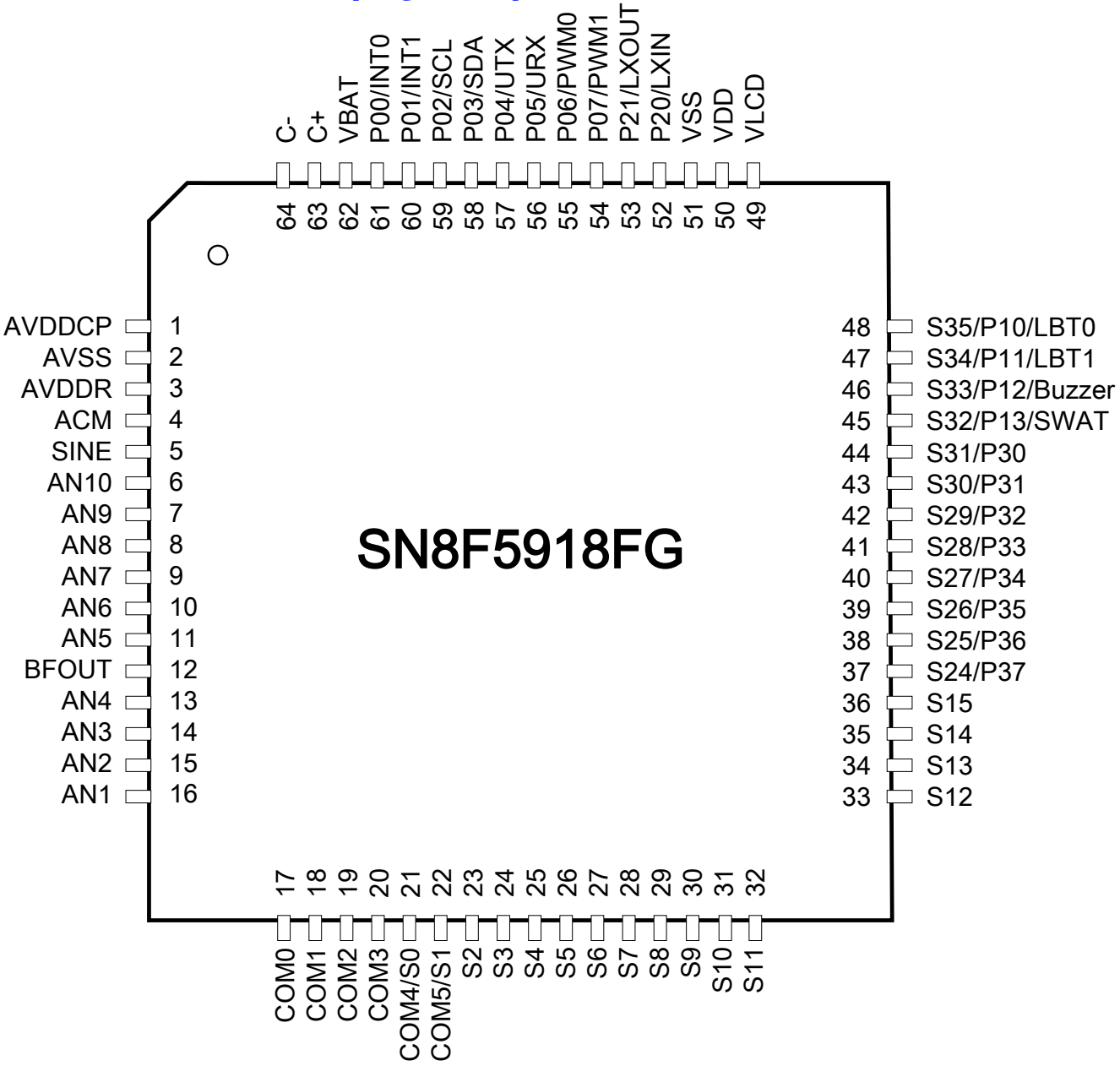
版本	时间	修订说明
1.0	Aug 2017	1. 修改 PGIA 寄存器描述。
1.1	Jan 2018	1. ILRC @5V 25°C 规格。
1.2	Jan 2018	1. 修改 BIA 章节。 2. 增加 ISP 指令 NOP 描述。 3. 增加 MDU 限制。 4. 修改 BIA 测量图表(P96) 。
1.3	Mar 2018	1. 修改 IROM 运行 8 MHz 描述。 2. 增加 CLKSEL 寄存器表格描述。 3. 增加 UART mode1 / mode3 波特率公式。
1.4	Mar 2018	1. 增加写 1-wire 程序模式。 2. 修改 ADCM2 bit4 ,bit5 ADCKS 内容。 3. 修改 SINE 频率 v.s Filter 设置表。
1.5	Apr 2018	1. 修改 CKCON, 010: 2 cycles (设置“010”,如果 Fcpu > 8MHz)。
1.6	Apr 2018	1. 修改 SN8F5919 封装 P20/P21 引脚信息。 2. 增加 CKCON V.S CLKSEL 建议表。
1.7	Jun.2021	1. 修改 IROM：0x7FFF。

4 引脚配置

4.1 SN8F5919FG (LQFP80)



4.2 SN8F5918FG (LQFP64)



4.3 引脚描述

电源引脚

引脚名称	类型	功能说明
DVDD	Power	数字电源(2.2V~5.5V)
DVSS	Power	数字地(0V)
AVDD	Power	模拟电源(2.2V~5.5V)
AVSS	Power	模拟地(0V)

P0

引脚名称	类型	功能说明
P0.0 INT0	Digital I/O Digital I/O	GPIO：双向输入输出引脚 INT0：中断输入引脚
P0.1 INT1	Digital I/O Digital I/O	GPIO：双向输入输出引脚 INT1：中断输入引脚
P0.2 SCL	Digital I/O Digital I/O	GPIO：双向输入输出引脚 I2C：时钟输出（主控）时钟输入（从动）
P0.3 SDA	Digital I/O Digital I/O	GPIO：双向输入输出引脚 I2C：数据引脚
P0.4 UTX	Digital I/O Digital Output	GPIO：双向输入输出引脚 UART：数据输出引脚
P0.5 URX	Digital I/O Digital Output	GPIO：双向输入输出引脚 UART：数据输入引脚
P0.6 PWM0	Digital I/O Digital Output	GPIO：双向输入输出引脚 PWM0：输出引脚
P0.7 PWM1	Digital I/O Digital Output	GPIO：双向输入输出引脚 PWM1：输出引脚

P1

引脚名称	类型	功能说明
P1.0 LBT0 SEG35	Digital I/O Analog Input LCD Output	GPIO：双向输入输出引脚 比较器外部输入引脚 LCD：LCD 驱动 Segment 引脚
P1.1 LBT1 SEG34	Digital I/O Analog Output LCD Output	GPIO：双向输入输出引脚 比较器外部输入引脚 LCD：LCD 驱动 Segment 引脚
P1.2 Buzzer SEG33	Digital I/O Digital Output LCD Output	GPIO：双向输入输出引脚 Buzzer 输出引脚 LCD：LCD 驱动 Segment 引脚
P1.3 SWAT SEG32	Digital I/O Digital I/O LCD Output	GPIO：双向输入输出引脚 单线仿真接口（*注：P1.7 口不能接任何器件@仿真模式） LCD：LCD 驱动 Segment 32 引脚

P2

引脚名称	类型	功能说明
P2.0 LXIN	Digital I/O Analog Input	GPIO：双向输入输出引脚 系统时钟：外部时钟输入引脚
P2.1 LXOUT	Digital I/O Analog Input	GPIO：双向输入输出引脚 系统时钟：驱动外部晶振

P3

引脚名称	类型	功能说明
P3.0 SEG31	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.1 SEG30	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.2 SEG29	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.3 SEG28	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.4 SEG27	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.5 SEG26	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.6 SEG25	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚
P3.7 SEG24	Digital I/O LCD Output	GPIO：双向输入输出引脚 LCD：LCD 驱动 Segment 引脚

LCD

引脚名称	类型	功能说明
VLCD	LCD Output	LCD：输出电压
COM0	LCD Output	LCD：LCD 驱动 common 引脚
COM1	LCD Output	LCD：LCD 驱动 common 引脚
COM2	LCD Output	LCD：LCD 驱动 common 引脚
COM3	LCD Output	LCD：LCD 驱动 common 引脚
COM4 SEG0	LCD Output LCD Output	LCD：LCD 驱动 common 引脚 LCD：LCD 驱动 Segment 引脚
COM5 SEG1	LCD Output LCD Output	LCD：LCD 驱动 common 引脚 LCD：LCD 驱动 Segment 引脚
SEG2~SEG23	LCD Output	LCD：LCD 驱动 Segment 引脚

模拟脚

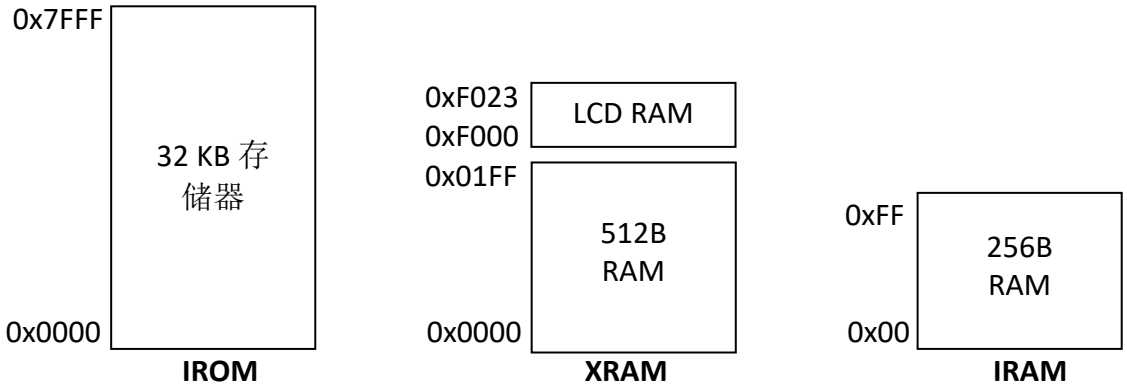
Pin Name	Type	Description
AVDD	Power	模拟电压输入端
AVSS	Power	模拟地
C+,C-	-	Charge pump 电容连接引脚 使能 Charge pump 时连接 1uF 电容
AVDDCP	Pump Output	Charge Pump 输出引脚 使能 Charge pump 时连接 2.2uF 电容到 DVSS
AVDDR	Analog Output	Regulator 输出模拟端，连接 1uF 电容到 AVSS。 可选项：2.7V/3.0V/3.3V/3.6V
ACM	Analog Output	ACM 电压输出 1.2V 作为 ADC 通用电压（仅在灌电流时） 连接 0.1uF 电容到 AVDDR
SINE	Analog Output	正弦波发生器输出引脚
BFOUT	Analog Output	体脂信号输出引脚
AI1	Analog Input	ADC：输入通道 1
AI2	Analog Input	ADC：输入通道 2
AI3/ R+	Analog Input	ADC：输入通道 3 ADC：外部参考电压正极输入引脚
AI4/ R-	Analog Input	ADC：输入通道 4 ADC：外部参考电压负极输入引脚
AI5	Analog Input	ADC：输入通道 5
AI6	Analog Input	ADC：输入通道 6
AI7	Analog Input	ADC：输入通道 7
AI8	Analog Input	ADC：输入通道 8
AI9	Analog Input	ADC：输入通道 9
AI10	Analog Input	ADC：输入通道 10

5 CPU

SN8F5910 系列是一颗增强型的 8051 微控制器，且完全兼容 MCS-51 指令集，因此可以使用目前流行的编译环境仿真（例如 Keil C51）。总的来说，在相同的频率下，SN8F5910 的 CPU 要比原始的 8051 快 9.4 到 12.1 倍。

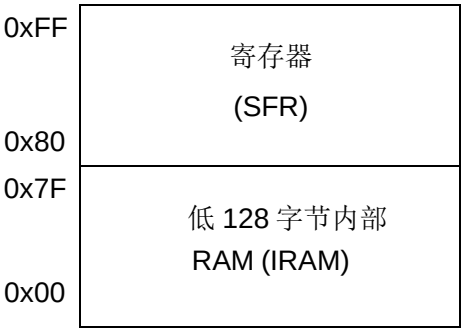
5.1 存储器结构

SN8F5910 内建了三个存储器：内部 RAM(IRAM)，外部 RAM(XRAM)，以及程序存储器(IROM)。内部 RAM 由 256 个字节组成，具有较高的存取性能（支持直接寻址和间接寻址）。相比之下，外部 RAM 有 512 个字节大小，但需要更长的存取周期。程序存储器是一个 32KB 的 Flash 存储器，最快的存取速度可达 8MHz。



5.2 直接寻址：IRAM和SFR

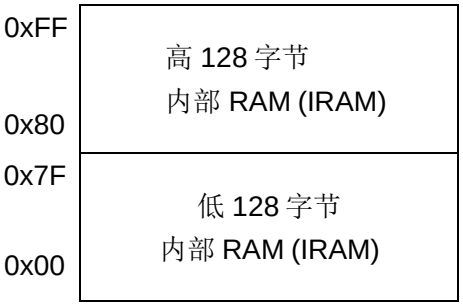
直接寻址指令（如 MOV A, direct）可以访问低 128 字节的内部 RAM（地址范围：00-7FH）和所有系统寄存器（SFR，地址范围：0x80-0xFF）。



除此之外，内部 RAM 的最低 32 字节（0x00-0x1F）可以看作是 4 组 R0-R7 工作寄存器，这 4 组工作寄存器可通过汇编指令（如 MOV A, R0）进行寻址。内部 RAM 的 0x20-0x2F 和每个 SFR 的结束段 0x0/0x8 都是位可寻址。

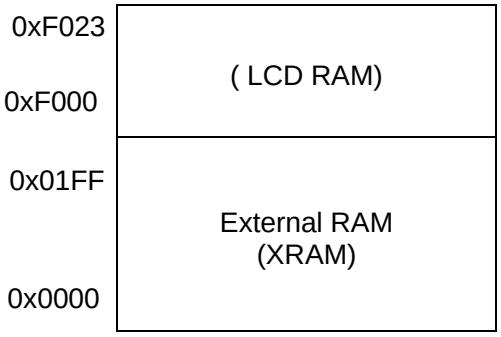
5.3 间接寻址 IRAM

虽然直接寻址指令可访问的内部 RAM 的周期比间接寻址的要少，第二种寻址类型可以访问内部 RAM 的所有区域，且是唯一可以访问内部 RAM 的高 128 字节（0x80-0xFF）的寻址方式。



5.4 外部RAM (XRAM)

外部 RAM 增加了变量的容量，但跟内部 RAM 相比，它的访问能力是最低的，这是因为频繁地使用变量和本地变量是为了存入内部 RAM，而外部 RAM 的绝大多数用法都是特定的。外部 RAM 可以作为优先级别较低的，或者在 ROM 中预装查表以加速访问周期的变量存储区域。LCD RAM 也位于外部 RAM 中。



5.5 程序存储器 (IROM)

程序存储器是 Flash 存储器，可以存储程序代码，ROM 查表数据以及其它的临时修改数据。可通过调试工具如 SN-Link Adapter 进行升级，也可以通过在线程序处理来自行更新（参考 In-System Program）。

5.6 程序存储器安全

SN8F5910 内置 ROM 加密机制，防止 Flash ROM 资料被破解。当使能加密功能时，就无法读出 ROM 里面的内容，所有的 ROM 地址都只能读到 0x00 的数据。

5.7 数据指针

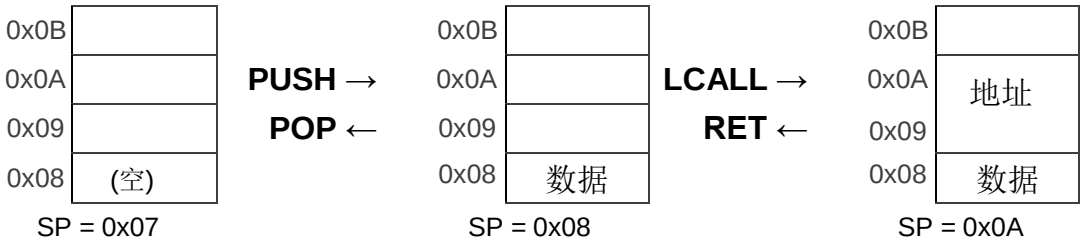
在执行 MOVX 和 MOVC 指令时，数据指针可帮助指定 XRAM 和 IROM 地址。该单片机有 2 组数据指针（DPH/DPL 和 DPH1/DPL1），可以通过 DPS 寄存器进行选择。DPC 寄存器控制 2 个功能：选择下一个数据指针和自动加减数据指针功能。

选择下一个数据指针的功能是在执行 MOVX @DPTR 指令后，可将指针指向所想要的地址。换句话说，DPS 可在 2 个数据指针之间自动切换。使能该功能的方法：首先写入 0 到 DPSEL，填 1 到 NDPS；然后写入 1 到 DPSEL，填 0 到 NDPS 寄存器。

自动加减数据指针的功能是在执行 MOVX @DPTR 指令后，可使得指针指向的地址自动加 1 或减 1。因此，它能够连续的访问外部存储器，而不需要重复的去指定数据指针指向的地址。这些功能由 DPC 寄存器控制，每个 DPTR 都有单独的 DPC 寄存器位，在数据传输中提供高灵活性。DPC 寄存器地址 0x93 指向窗口，在那里使用 DPS 寄存器选择实际的 DPC，与 DPTR 相同。

5.8 堆栈

可从内部 RAM（IRAM）中分出任意一部分作为堆栈使用，但要求手动分配以保证堆栈区域不会与其它 RAM 的变量重叠。堆栈溢出或者下溢会导致其它 RAM 的变量重写错误，故在分配堆栈的区域时必须考虑到这些问题以避免这种情形的发生。



默认情况下，堆栈指针（SP 寄存器）指向 0x07，就是指堆栈的区域从 IRAM 地址中的 0x08 开始。换句话说，如果计划想将堆栈区域设置为从 IRAM 的 0xC0 开始，就要在系统复位后将 SP 寄存器设置为 0xBF。

一条汇编 PUSH 指令占用堆栈中的一个字节，LCALL，ACALL 指令以及中断分别占用堆栈中的两个字节。POP 指令释放一个字节，RET/RETI 指令释放两个字节。

5.9 堆栈和数据指针寄存器

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

SP 寄存器 (0x81)

Bit	Field	Type	Initial	Description
7..0	SP	R/W	0x07	堆栈指针。

DPL 寄存器 (0x82)

Bit	Field	Type	Initial	Description
7..0	DPL[7:0]	R/W	0x00	DPTR0 的低字节。

DPH 寄存器 (0x83)

Bit	Field	Type	Initial	Description
7..0	DPH[7:0]	R/W	0x00	DPTR0 的高字节。

DPL1 寄存器 (0x84)

Bit	Field	Type	Initial	Description
7..0	DPL1[7:0]	R/W	0x00	DPTR1 的低字节。

DPH1 寄存器 (0x85)

Bit	Field	Type	Initial	Description
7..0	DPH1[7:0]	R/W	0x00	DPTR1 的高字节。

DPS 寄存器 (0x92)

Bit	Field	Type	Initial	Description
7..1	Reserved	R	0x00	
0	DPSEL	R/W	0	DPTR 选择控制位。 0：选择 DPH/DPL（DPTR0）； 1：选择 DPH1/DPL1（DPTR1）。

DPC 寄存器(0x93)

Bit	Field	Type	Initial	Description
7..4	Reserved	R	0x0	
3	NDPS	R/W	0	下一个 DPTR 选择位。 执行 MOVX @DPTR 指令后，自动加载该位到 DPSEL。
2..1	ATMS/ATMD	R/W	00	自动加减数据指针（使能 ATME 位时有效）。 00：执行 MOVX @DPTR 指令后+1； 01：执行 MOVX @DPTR 指令后-1； 10：执行 MOVX @DPTR 指令后+2； 11：执行 MOVX @DPTR 指令后-2。
0	ATME	R/W	0	自动加减数据指针功能。 0：关闭； 1：使能。

6 特殊功能寄存器

6.1 特殊功能寄存器存储器表

BIN / HEX	000	001	010	011	100	101	110	111
F8	FRQCMD	P0M	P1M	P2M	P3M	-	-	PFLAG
F0	B	P0UR	P1UR	P2UR	P3UR	FRQL	FRQH	SRST
E8	-	MD0	MD1	MD2	MD3	MD4	MD5	ARCON
E0	ACC	-	-	-	P1OC	CLKSEL	CLKCMD	-
D8	S0CON2	-	I2CDAT	I2CADR	I2CCON	I2CSTA	SMBSEL	SMBDST
D0	PSW	CHS	VREG	AMPM	ADCM1	ADCM2	BZRM	LBTM
C8	-	TC1M	TC1RL	TC1RH	TC1CL	TC1CH	TC1DL	TC1DH
C0	IRCON	TC0M	TC0RL	TC0RH	TC0CL	TC0CH	TC0DL	TC0DH
B8	IEN1	IP1	S0RELH	T0M	T0C	MIN	SEC	-
B0	P3	LCDM1	LCDM2	-	-	ADCDL	ADCDM	ADCDH
A8	IEN0	IP0	S0RELL	SWT	SINE	FILT_L	FILT_H	OPM
A0	P2	-	-	-	-	-	-	-
98	S0CON	S0BUF	IEN2	-	PEBYTE	-	P3CON	P1CON
90	P1	P1W	DPS	DPC	PECMD	PEROML	PEROMH	BUCK
88	-	-	-	-	-	-	CKCON	PEDGE
80	P0	SP	DPL	DPH	DPL1	DPH1	WDTR	PCON

6.2 特殊功能寄存器说明

0x80 - 0x9F 寄存器说明

Register	Address	Description
P0	0x80	P0 数据缓存器。
SP	0x81	堆栈指针寄存器。
DPL	0x82	数据指针 0 低字节寄存器。
DPH	0x83	数据指针 0 高字节寄存器。
DPL1	0x84	数据指针 1 低字节寄存器。
DPH1	0x85	数据指针 1 高字节寄存器。
WDTR	0x86	看门狗定时器清零寄存器。
-	0x87~0x8D	-
CKCON	0x8E	扩展周期寄存器。
PEDGE	0x8F	外部中断边沿控制寄存器。
P1	0x90	P1 数据缓存器。
P1W	0x91	P1 唤醒控制寄存器。
DPS	0x92	数据指针选择寄存器。
DPC	0x93	数据指针控制寄存器。
PECMD	0x94	In-System Program 命令寄存器。
PEROML	0x95	In-System Program ROM 地址低字节。
PEROMH	0x96	In-System Program ROM 地址高字节。
BUCK	0x97	In-System Program RAM 分配地址。
S0CON	0x98	UART 控制寄存器。
S0BUF	0x99	UART 数据缓存器。
IEN2	0x9A	中断使能寄存器。
-	0x9B	-
PEBYTE	0x9C	总线编程 ROM 字节写入地址。
-	0x9D	-
P3CON	0x9E	P3 配置控制寄存器。
P1CON	0x9F	P1 配置控制寄存器。

0xA0 - 0xBF 寄存器说明

Register	Address	Description
P2	0xA0	P2 数据缓存器。
-	0xA1~0xA7	-
IEN0	0xA8	中断使能寄存器。
IP0	0xA9	中断优先权寄存器。
S0RELL	0xAA	UART 重装低字节寄存器。
SWT	0xAB	转换 TEST 寄存器。
SINE	0xAC	正弦波发生器寄存器 。
FILT_L	0xAD	FILTER 低字节寄存器。
FILT_H	0xAE	FILTER 高字节寄存器。
OPM	0xAF	OPA 模式寄存器。
P3	0xB0	P3 数据缓存器。
LCDM1	0xB1	LCD 模式 1 控制寄存器。
LCDM2	0xB2	LCD 模式 2 控制寄存器。
-	0xB3	-
-	0xB4	-
ADCDL	0xB5	ADC 输出数据低字节。
ADCDM	0xB6	ADC 输出数据中间字节。
ADCDH	0xB7	ADC 输出数据高字节。
IEN1	0xB8	中断使能寄存器。
IP1	0xB9	中断优先权寄存器。
S0RELH	0xBA	UART 重装高字节寄存器。
T0M	0xBB	定时器 T0 模式控制寄存器。
T0C	0xBC	定时器 T0 计数寄存器。
MIN	0xBD	定时器 T0 分钟寄存器。
SEC	0xBE	定时器 T0 秒钟寄存器。
-	0xBF	-

0xC0 - 0xDF 寄存器说明

Register	Address	Description
IRCON	0xC0	中断请求寄存器。
TC0M	0xC1	TC0 定时器模式控制寄存器。
TC0RL	0xC2	TC0 定时器计数器重装缓存器低字节。
TC0RH	0xC3	TC0 定时器计数器重装缓存器高字节。
TC0CL	0xC4	TC0 定时器低字节计数器。
TC0CH	0xC5	TC0 定时器高字节计数器。
TC0DL	0xC6	PWM0 占空比控制低字节缓存器。
TC0DH	0xC7	PWM0 占空比控制高字节缓存器。
-	0xC8	-
TC1M	0xC9	TC1 定时器模式控制寄存器。
TC1RL	0xCA	TC1 定时器计数器重装缓存器低字节。
TC1RH	0xCB	TC1 定时器计数器重装缓存器高字节。
TC1CL	0xCC	TC1 定时器低字节计数器。
TC1CH	0xCD	TC1 定时器高字节计数器。
TC1DL	0xCE	PWM1 占空比控制低字节缓存器。
TC1DH	0xCF	PWM1 占空比控制高字节缓存器。
PSW	0xD0	系统标志寄存器。
CHS	0xD1	ADC 输入通道选择寄存器。
VREG	0xD2	电压 Regulator 控制寄存器。
AMPM	0xD3	PGIA 控制寄存器。
ADCM1	0xD4	ADC 控制寄存器 1。
ADCM2	0xD5	ADC 控制寄存器 2。
BZRM	0xD6	Buzzer 控制寄存器。
LBTM	0xD7	比较器和电池低电压检测控制寄存器。
S0CON2	0xD8	UART 波特率控制寄存器
-	0xD9	-
I2CDAT	0xDA	I2C 数据缓存器。
I2CADR	0xDB	I2C 从动地址。
I2CCON	0xDC	I2C 接口操作控制寄存器。
I2CSTA	0xDD	I2C 状态代码。
SMBSEL	0xDE	SMBUS 模式控制寄存器。
SMBDST	0xDF	SMBUS 内部超时寄存器。

0xE0 - 0xFF 寄存器说明

Register	Address	Description
ACC	0xE0	ACC 寄存器。
-	0xE1	-
-	0xE2	-
-	0xE3	-
P1OC	0xE4	开漏功能控制寄存器。
CLKSEL	0xE5	时钟切换选择寄存器。
CLKCMD	0xE6	时钟切换控制寄存器。
-	0xE7	-
-	0xE8	-
MD0	0xE9	MDU 控制寄存器 0。
MD1	0xEA	MDU 控制寄存器 1。
MD2	0xEB	MDU 控制寄存器 2。
MD3	0xEC	MDU 控制寄存器 3。
MD4	0xED	MDU 控制寄存器 4。
MD5	0xEE	MDU 控制寄存器 5。
ARCON	0xEF	MDU 算法控制寄存器。
B	0xF0	乘法/除法指令数据缓存器。
P0UR	0xF1	P0 上拉电阻控制寄存器。
P1UR	0xF2	P1 上拉电阻控制寄存器。
P2UR	0xF3	P2 上拉电阻控制寄存器。
P3UR	0xF4	P3 上拉电阻控制寄存器。
FRQL	0xF5	IHRC 频率微调功能控制低字节寄存器。
FRQH	0xF6	IHRC 频率微调功能控制高字节寄存器。
SRST	0xF7	软件复位控制寄存器。
FRQCMD	0xF8	IHRC 频率微调命令寄存器。
P0M	0xF9	P0 输入/输出模式寄存器。
P1M	0xFA	P1 输入/输出模式寄存器。
P2M	0xFB	P2 输入/输出模式寄存器。
P3M	0xFC	P3 输入/输出模式寄存器。
-	0xFD	-
-	0xFE	-
PFLAG	0xFF	复位标志寄存器。

6.3 系统寄存器

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 寄存器(0xE0)

Bit	Field	Type	Initial	Description
7..0	ACC[7:0]	R/W	0x00	8 位数据寄存器用于转移或操控 ALU 和数据存储器之间的数据，若操作结果溢出（OV）或者由借位（C 或 AC），以及相等情况（P）发生时，该标志位会在 PSW 寄存器中进行设置。

B 寄存器 (0xF0)

Bit	Field	Type	Initial	Description
7..0	B[7:0]	R/W	0x00	B 寄存器在使用乘法和除法指令时使用，而且还能作为 scratch-pad 寄存器来保留临时数据。

PSW 寄存器 (0xD0)

Bit	Field	Type	Initial	Description
7	CY	R/W	0	进位标志。 0：加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”或比较运算的结果<0； 1：加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”或比较运算的结果≥0。
6	AC	R/W	0	辅助进位标志。 0：BCD 操作时没有从 ACC 的第三位开始执行； 1：BCD 操作时从 ACC 的第三位开始执行。
5	F0	R/W	0	通用标志位，可任意设定。
4..3	RS[1:0]	R/W	00	寄存器 bank 选择控制位，用于选择工作寄存器 bank。 00：00H-07H（Bank0）；01：08H-0FH（Bank1）； 10：10H-17H（Bank2）；11：18H-1FH（Bank3）。
2	OV	R/W	0	溢出标志。 0：算术操作时，ACC 没有溢出； 1：算术操作时，ACC 溢出。
1	F1	R/W	0	通用标志位，可任意设定。
0	P	R	0	奇偶标志位。 0：A 中 1 的个数为偶数； 1：A 中 1 的个数为奇数

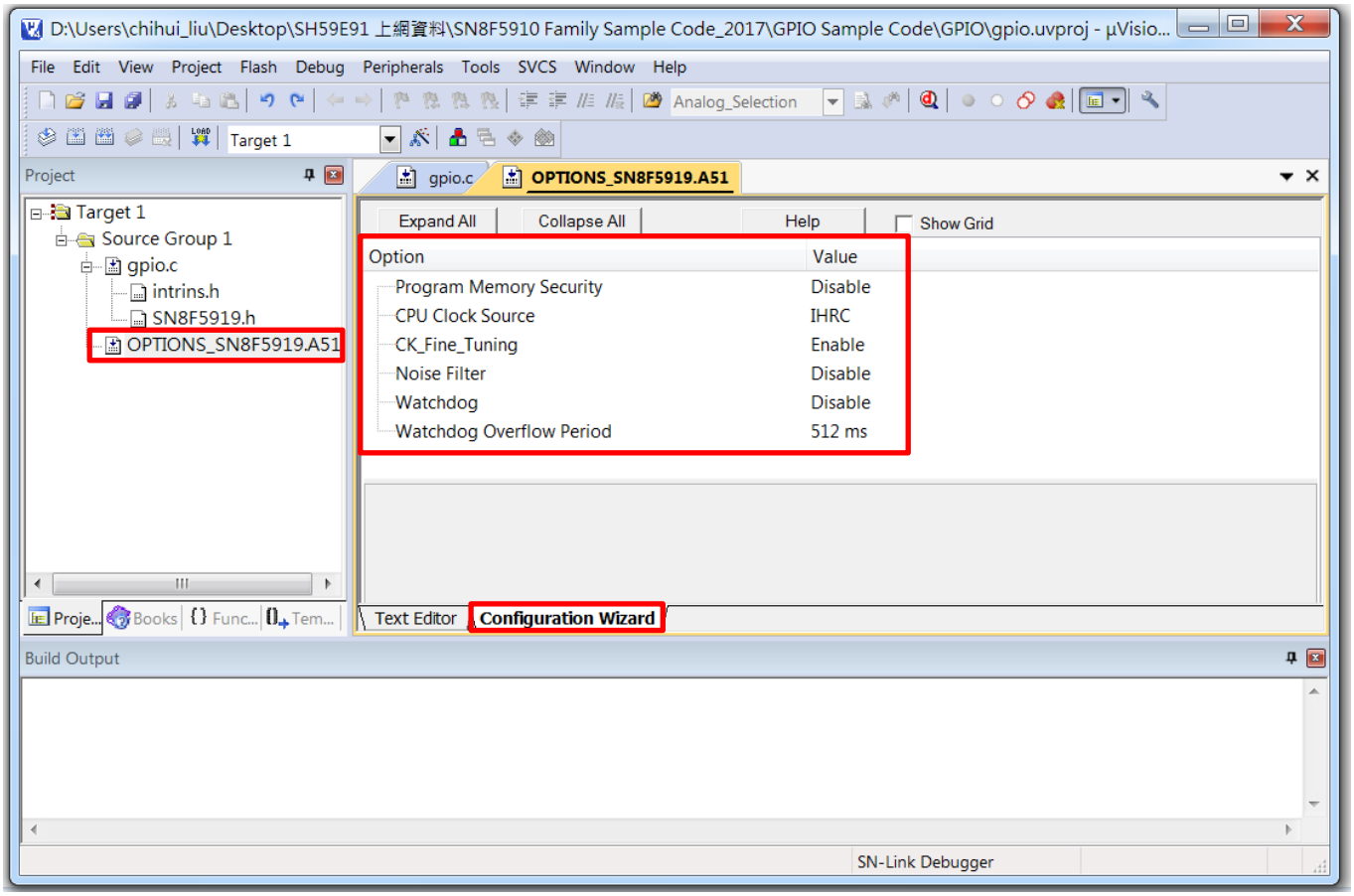
7 复位和上电控制

复位和上电控制有下列方式：低电压检测（LVD），看门狗，可编程的外部复位引脚和软件复位。前面三种方式可触发额外的上电流程，随后单片机初始化所有的寄存器，并是程序从复位向量（ROM 地址 0x0000）处重新开始执行。

7.1 复位配置和上电控制

SONiX 发布了一个 SN8F591x_OPTIONS.A51 文件（从松翰官网 www.sonix.com.tw 下载）。该文件包含复位源的合适的参数和 CPU 时钟源选择。强烈建议新建一个 Keil 工程。SN8F5910 的调试工具手册提供了更详细的内容。具体选项如下所示：

- 程序加密
- CPU 时钟源
- Noise Filter
- 看门狗



7.2 上电流程

LVD，看门狗和外部复位引脚可触发上电流程，在复位信号结束和执行程序之间的时候发生。总的来说，上电流程包括 2 个阶段：电源稳定期和时钟稳定期。

在典型条件下，电源稳定期花费 5ms，然后单片机自动选择 CPU 时钟源。驱动选择的时钟源后，系统计数时钟周期的 4096 次以确保时钟稳定。

7.3 LVD 复位

低电压检测监控 VDD 引脚的电压，其电压点为 1.8V。VDD 的电压低于 1.8V 时，MCU 复位。

7.4 Watchdog 复位

看门狗是周期性的复位信号发生器，用于监控程序的执行流程。其内部定时器可在程序流程的检测点被清零，因此，只有在发生软件问题后才会产生实际的复位信号。通过写入 0x5A 到 WDTR 就可以在程序中设置检测点。

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

WDT 时钟预分频器	Clock/1	Clock/2	Clock/4	Clock/8
看门狗间隔时间	64ms	128ms	256ms	512ms

看门狗的工作模式如下所示：

Always mode：其内部定时器在 CPU 的所有工作模式（NORMAL，IDLE，SLEEP）下都在计数。

Enable mode：其内部定时器只在 CPU 的 Normal 模式下计数，在 IDLE 和 SLEEP 模式下不会触发看门狗复位。

Disable mode：其内部定时器在 CPU 的所有工作模式下都不计数，在这种模式下不会触发看门狗复位。

当看门狗一直处于工作状态时，系统会消耗额外的功耗。

WDT clock = F_{ILRC} ÷ 512

SYM.	*DESCRIPTION*	*Condition*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
F_{ILRC}	内部低速时钟发生器	VDD = 5V @ 25°C	12	16	24	KHz

7.5 外部复位引脚

SN8F5910 系列没有外部复位引脚。

7.6 软件复位

连续设置 SRSTREQ 寄存器后会产生软件复位，因此，该程序使能固件的性能以复位单片机（如更新固件后复位）。下面的这段 C 程序代码就因为反复地设置 SRST 寄存器的最小位而导致软件复位。

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

7.7 复位和上电控制寄存器

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

PFLAG 寄存器

Bit	Field	Type	Initial	Description
7	POR	R	0	若单片机已经由 LVD 触发复位，则该位自动置 1。
6	WDT	R	0	若单片机已经由看门狗触发复位，则该位自动置 1。
5..0	Reserved	R	0	

SRST 寄存器

Bit	Field	Type	Initial	Description
7..1	Reserved	R	0	
0	SRSTREQ	R/W	0	连续设置该位 2 次触发软件复位。

WDTR Register (0x86)

Bit	Field	Type	Initial	Description
7..0	WDTR[7:0]	W	-	由 WDTR 寄存器清看门狗定时器，写入 0x5A 和到 WDTR 寄存器复位看门狗定时器。

8 系统时钟和电源管理

单片机内置 3 个不同的操作模式：NORMAL 模式，IDLE 模式和 STOP 模式以省电。

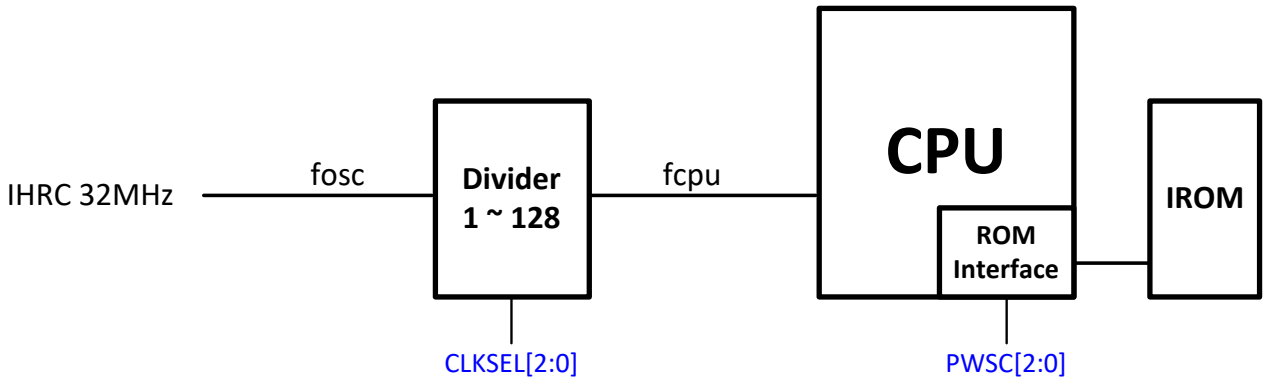
NORMAL 模式是指 CPU 和外设功能都正常工作，系统时钟是通过所选择的时钟源，程序设置的分频数，以及程序设置的 ROM 等待时间所确定的。IDLE 模式是指 CPU 时钟暂停的状态，但保留了外设功能（如定时器，SPI，UART 和 I2C）。与之相反的是，STOP 模式则禁止所有功能和时钟发生器，直至唤醒信号将系统唤醒进入 NORMAL 模式。此外，LCD 功能和 T0-RTC 功能也能在 STOP 模式下正常工作，为低待机电流应用。

8.1 系统时钟

该单片机内置时钟发生器（IHRC 32MHz）。在复位或上电过程中，系统自动加载 IHRC。IHRC 可以看作是 Fosc，Fosc 一旦选定就不能再改变。

之后，Fosc 可以分频为 Fosc/1~Fosc/128，由 CLKSEL 寄存器控制。CPU 使用分频之后的时钟作为它的时钟频率（称作 Fcpu）。写入 0x69 到 CLKCMD 寄存器时设置 CLKSEL。

```
1    CLKSEL  =  0x04;      // set fcpu = fosc / 8
2    CLKCMD  =  0x69;      // Apply CLKSEL's setting
```



ROM 接口位于 CPU 和 IROM（程序存储器）之间，可设置 ROM 读取周期以支持低速程序存储器。例如：CPU 计划运行在 32MHz，而 IROM 只能运行在 8MHz 以下，则在 CKCON 寄存器中必须设置 ROM 读取周期为增加 3 个以上的 Fcpu。

$$\text{IROM fetching cycle} = \frac{\text{fcpu}}{\text{PWSC[2:0]+1}} \leq 16\text{MHz, PWSC[2:0]} = 0\sim7$$

8.2 Noise Filter

Noise Filter 功能由 Noise Filter 选项控制，是一个低通滤波器，并支持晶振模式。这个选项的目的在于滤除外部高速震荡源上的高频噪声干扰。高干扰环境下，强烈建议使能 Noise Filter 以减少噪音干扰。

8.3 电源管理

复位信号和上电结束后，CPU 以 Fcpu 的速率开始执行程序。系统以 Normal 模式开始工作。

CPON 寄存器的最低 2 位（bit0-IDLE 和 bit1-STOP）控制单片机的电源管理部分。

若 IDLE 位设置为 1，只有 CPU 时钟源被关闭。因此，在这种状态下，外设功能（如定时器，PWM，UART 和 I2C）和时钟发生器（IHRC 32MHz）仍然正常工作。P0/P1 输入的任何改变和中断时间都会导致单片机返回到 Normal 模式，IDLE 位自动清零。

若 STOP 位设置为 1，CPU，外设和时钟发生器都处于停止状态，在这个模式下，寄存器中存储的数据和 RAM 都保持不变。P0/P1 输入的任何改变都可将单片机唤醒并使系统重新开始执行，STOP 位自动清零。

*注：用户用 C 编译器开发时，强烈建议使用 IDLE 和 STOP 宏来控制单片机的系统模式，来代替直接设置 IDLE 和 STOP 位。

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

8.4 系统时钟和电源管理寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CKCON	-	PWSC2	PWSC1	PWSC0	ESYN	EWSC2	EWSC1	EWSC0
CLKSEL	-	-	-	-	-	CLKSEL2	CLKSEL1	CLKSEL0
CLKCMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PCON	SMOD	-	-	-	P2SEL	GF0	STOP	IDLE
P1W	-	-	-	-	P13W	P12W	P11W	P10W

✷ *注：对于用户在 C 语言或汇编语言中开发的程序，程序的第一行"必须设置"CKCON=0x70,然后设置 CLKSEL= 0x07~0x00, CLKMD= 0x69,CKCON=0x00~0x70，这个优先级不能被修改。*

CKCON 寄存器 (0x8E)

Bit	Field	Type	Initial	Description
7	Reserved	R	0	
6..4	PWSC[2:0]	R/W	111	增加读取程序存储器的周期。 000：无；　（若 Fcpu≤8MHz，设置为 000） 001：1 个周期；（若 Fcpu>8MHz，设置为 001） 010：2 个周期；（保留） 011：3 个周期；（保留） 100：4 个周期；（保留） 101：5 个周期；（保留） 110：6 个周期；（保留） 111：7 个周期。（保留）
3	ESYN	R/W	0	增加额外的写数据到 XRAM 的周期。（用户设置为 0）
2..0	EWSC[2:0]	R/W	001	增加读取 XRAM 的周期。 000：无；（用户设置为 0） 001：1 个周期；- 010：2 个周期； 011：3 个周期；（保留） 100：4 个周期；（保留） 101：5 个周期；（保留） 110：6 个周期；（保留） 111：7 个周期。（保留）

CLKSEL 寄存器 (0xE5)

Bit	Field	Type	Initial	Description
7..3	Reserved	R	0x00	
2..0	CLKSEL[2:0]	R/W	111	CLKSEL 中的设置将在写 CLKCMD 之后生效。 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　(PWSC[2:0] 设置：01x, Cycle >2)； 111：fcpu = fosc / 1　(PWSC[2:0] 设置：1xx, Cycle >4)； *设置 CLKSEL[2:0]后，必须写入 0x69 以应用 CLKSEL 的设置； *fosc =32Mhz。

CLKCMD 寄存器 (0xE6)

Bit	Field	Type	Initial	Description
7..0	CMD[7:0]	W	0x00	写入 0x69 来应用 CLKSEL 的设置。

PCON 寄存器去(0x87)

Bit	Field	Type	Initial	Description
7				参考其它章节。
6..4	Reserved	R	0x00	
3	P2SEL	R/W	1	高命令地址自己配置为。执行 MOVX @Ri 操作时选择地址的高字节（XRAM[15:8]）。 0：XRAM[15:8]=P2REG，P2REG 为 P2 输出寄存器的内容； 1：XRAM[15:8]=00。
2	GF0	R/W	0	通用标志。
1	STOP	R/W	0	1：单片机切换到 STOP 模式。
0	IDLE	R/W	0	1：单片机切换到 IDLE 模式。

P1W 寄存器(0x91)

Bit	Field	Type	Initial	Description
7..0	P1nW	R/W	0	0：禁止 P1.n 的唤醒功能； 1：使能 P1.n 的唤醒功能。

9 中断

SN8F5910 包含 8 个中断源（2 个外部中断和 6 个外部中断），分为 4 个优先级别。每个中断源包含 1 个或多个中断请求标志。有中断发生时，相对应的中断请求位置为逻辑 1。若同时使能中断使能位和全局中断使能位（EAL=1）时，有中断请求时则执行该中断服务程序（ISR）。多数请求振荡器标志位必须由软件清零，而有些中断请求标志位由硬件自动清零。最后，执行 RETI 指令后，整个 ISR 结束。中断源的简称，中断向量，优先级别和控制位如下表所示：

中断源	使能中断标志位	请求标志位 (IRQ)	IRQ 清除	优先级 / 向量
系统复位	-	-	-	0 / 0x0000
INT0	EX0	IE0	自动清 0	1 / 0x0003
INT1	EX1	IE1	自动清 0	2 / 0x000B
TC0	ETC0	TCF0	自动清 0	3 / 0x0013
T0	ET0	TF0	自动清 0	4 / 0x0093
TC1	ETC1	TCF1	自动清 0	5 / 0x001B
UART	ES0	TI0 / RI0	软件清 0	6 / 0x0023
ADC	EADC	ADCF	软件清 0	7 / 0x002B
I2C	EI2C	SI	软件清 0	8 / 0x00AB

9.1 中断操作

中断操作由中断请求标志位和中断使能位控制。中断请求标志位显示中断源的状态，与中断功能的状态（使能或禁止）无关。同时使能中断使能位和全局中断使能位（EAL=1）且中断请求标志位有效时，程序计数器指向中断向量（0x03-0xEB），系统执行相对应的中断服务程序 ISR。

9.2 中断优先级别

每个中断源都有特定的优先级别。若同时发生 2 个中断，系统会先执行优先级别高的 ISR，然后再执行优先级别低的 ISR。下一次的 ISR 必须要等待前面的 ISR 执行完成之后才可以执行，不用理会中断的优先级别。

对应特定的优先权需求，需要用到 4 级的优先级别（级别 0-级别 3）。所有的中断源按优先级别分为 6 大类（Group0-Group5），每组设置为同样的特定优先级别，由寄存器 IP0/IP1 设置，级别 3 最高，级别 0 最低。同组的中断源共用同样的优先级别，对于同样的优先级别，优先权的规则按照默认的优先权。

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

首先执行优先级别较高的 ISR，甚至可以打断执行中的优先级别较低的 ISR，直到优先级别较高的 ISR 执行完成之后再执行优先级别较低的 ISR。

Group	Interrupt Source			
Group 0	INT0	-	-	-
Group 1	INT1	-	-	-
Group 2	TC0	T0	-	-
Group 3	TC1	-	-	-
Group 4	UART	-	-	-
Group 5	ADC	I2C	-	-

IP0, IP1 寄存器

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 寄存器(0XA9)

Bit	Field	Type	Initial	Description
5..0	IP0[5:0]	R/W	0	中断优先权。和 IP1 寄存器的相对应的位结合在一起可以指定相对应的中断的优先级别。
Else	Reserved	R	0	

IP1 寄存器 (0XB9)

Bit	Field	Type	Initial	Description
5..0	IP1[5:0]	R/W	0	中断优先权。和 IP0 寄存器的相对应的位结合在一起可以指定相对应的中断的优先级别。
Else	Reserved	R	0	

*例: 优先级 GP0>GP1>GP2>GP3=GP4=GP5.

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

*例: 优先级 GP5>GP4>GP3>GP2=GP1=GP0.

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

9.3 中断寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IEN0	EAL	-	-	ES0	-	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	-	EI2C
IEN2	-	-	-	EADC	-	ETC1	ETC0	-
IRCON	ADCF	-	TCF1	TCF0	-	TF0	IE1	IE0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

IEN0 寄存器 (0XA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	所有中断使能控制位。 0：禁止； 1：使能。
4	ES0	R/W	0	UART 中断控制位。 0：禁止； 1：使能。
2	EX1	R/W	0	外部中断 P0.1（INT1）中断控制位。 0：禁止； 1：使能。
1	ET0	R/W	0	T0 定时器中断控制位。 0：禁止； 1：使能。
0	EX0	R/W	0	外部中断 P0.0（INT0）中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN1 寄存器 (0XB8)

Bit	Field	Type	Initial	Description
0	EI2C	R/W	0	I2C 定时器中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN2 寄存器(0X9A)

Bit	Field	Type	Initial	Description
4	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
2	TC1	R/W	0	TC1 溢出中断控制位。 0：禁止； 1：使能。
1	TC0	R/W	0	TC0 溢出中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IRCON 寄存器 (0xC0)

Bit	Field	Type	Initial	Description
7	ADCF	R/W	0	ADC 中断请求标志位。 0：无 ADC 中断请求； 1：请求 ADC 中断。
5	TCF1	R/W	0	TC1 中断请求标志位。 0：无 TC1 中断请求； 1：请求 TC1 中断。
4	TCF0	R/W	0	TC0 中断请求标志位。 0：无 TC0 中断请求； 1：请求 TC0 中断。
2	TF0	R/W	0	T0 定时器中断请求标志位。 0：无 T0 断请求； 1：请求 T0 中断。
1	IE1	R/W	0	外部中断 P0.1（INT1）请求标志位。 0：无 INT1 断请求； 1：请求 INT1 中断。
0	IE0	R/W	0	外部中断 P0.0（INT0）请求标志位。 0：无 INT0 断请求； 1：请求 INT0 中断。
Else	Reserved	R	0	

S0CON 寄存器(0X98)

Bit	Field	Type	Initial	Description
1	TI0	R/W	0	UART 发送中断请求标志位，显示 UART 串行传输的完成状态。在模式 0 位 8 结束时，或者其它模式的停止位开始时由硬件设置为 1；必须由软件清零。 0：无 UART 发送中断请求； 1：UART 发送请求中断。
0	RI0	R/W	0	UART 接收中断请求标志位，显示 UART 串行接收的完成状态。在模式 0 位 8 结束时，或者其它模式的停止位中间时由硬件设置为 1；必须由软件清零。 0：无 UART 接收中断请求； 1：UART 接收请求中断。
Else				参考其它章节。

I2CCON 寄存器 (0XDC)

Bit	Field	Type	Initial	Description
7	SI	R/W	0	串行中断标志位。 当进入了 I2C 的 26 个状态当中的 25 个状态时，SI 标志位会被硬件置 1，只有当 I2C 状态寄存器为 0xF8 时，SI 标志位才没有被置 1，表示没有可用的相关状态信息。SI 标志位必须由软件清零，必须通过写 0 到 SI 标志才可以清 SI 标志位，写入 1 到该位并不能更改 SI 的值。
Else				参考其它章节。

10 MDU

乘除法单元是一个内置算法协处理器，可以使能单片机去执行额外扩展的算法操作。该单元提供 32 位无符号的除法，16 位无符号的乘法，移位和规范化操作，可以通过写入 MD0-MD5 寄存器不同的的顺序来识别这些操作。

10.1 乘法 (16 位 x 16 位)

乘法的基础原理包括 3 部分：被乘数、乘数和得数。进行一个乘法要求以下面的写入顺序：MD0（被乘数的低字节），MD4（乘数的低字节），MD1（被乘数的高字节），然后 MD5（乘数的高字节）。

写入 MD5 寄存器结束时，自动开始乘法操作，乘法操作共需要 11 个 CPU 周期，其有效的得数可通过特定的顺序读取：MD0（LSB），MD1，MD2，然后 MD3（MSB）寄存器。

10.2 除法 (32 位/16 位 和 16 位/16 位)

MDU 支持 2 种除法操作：32 位/16 位和 16 位/16 位。第一种需要 17 个 CPU 周期来进行计算；第二种只需要 9 个周期。

32 位除法以下面特定的写入顺序开始：MD0，MD1，MD2，MD3，MD4 和 MD5。32 位被除数存入 MD3（最高有效位）到 MD0 寄存器，16 位除数存入 MD5 到 MD4 寄存器（MSB 位于 MD5 寄存器）。

16 位除法只需要 4 个寄存器。16 位被除数存入 MD1 到 MD0 寄存器，16 位除数存入 MD5 到 MD4 寄存器（最高有效位位于 MD1 和 MD5 寄存器）。其合适顺序为：MD0，MD1，MD4 和 MD5。

MDU 从写入数据到 MD5 寄存器之后开始计算，需要 9 个或者 17 个 CPU 周期，取决于被除数的长度。计算得出的商存入 MD3 到 MD0 寄存器（32 位除法），或者 MD1 到 MD0 寄存器（16 位除法），LSB 位于 MD0 寄存器。不管是否执行了除法操作，余数都放在 MD5（MSB）和 MD4 寄存器中。但是必须最后读取 MD5 以指示是否完成全部除法操作。

10.3 移位和规范化

移位和规范化操作移动 32 位寄存器（MD3 到 MD0，MSB 位于 MD3）一段特定或者不确定的次数。

移位操作中，通过指定的位数左移或者右移 32 位无符号的整数，移动的方向和位数都是指定的，存在 ARCON 寄存器中。移位操作需要 3-18 个 CPU 周期，取决于移位的次数。.

规范化操作中，32 位无符号的整数会重复地向左移位，直到最高有效位（MD3 寄存器的第 7 位）为 1。规范化操作需要 4-19 个 CPU 周期，取决于实际移位的次数。

移位操作和规范化操作都以下面的写入顺序开始：MD0，MD1，MD2，MD3，最后为 ARCON 寄存器。最终结果存入 MD0-MD3 寄存器中，以下面顺序来读取：MD0，MD1，MD2，MD3。

10.4 与Keil C51 合作（Cooperate with Keil C51）

由于 Keil C51 支持硬件和软件的乘法/除法操作，在 C 中要求使用命令行'#pragma mdu_r515'来使能硬件 MDU 的功能以获得更高的性能。随后，Keil C51 编译数学运算支持 MDU。

```
1  #include < SN8F5919.H>
2  #pragma   mdu_r515                  //Keil C51 MDU command line
```

10.5 错误标志(MDEF)

错误标志 MDEF 指示未正确运行的操作（算法操作被重新开始或者被新操作打断），检测错误机制会在第一阶段写数据到 MD0 寄存器时自动使能，并在第三阶段的最后一条对 MD3（乘法或者移位/正常化）或者 MD5 寄存器（除法）进行读取指令时关闭。

出现下列情形时，错误标志置 1：

在 MDU 操作的第二阶段中对“MDx”寄存器（MD0~MD5 以及 ARCON 寄存器当中的任意一个）进行写操作（重新启动或计算中断）。

在错误标志位机制已经使能的情况下，在 MDU 操作的第二阶段中对“MDx”寄存器进行读取操作。此时错误标志位会被置 1，但是计算不会被打断。

错误标志位只有在读“ARCON”寄存器之后才会复位，且该标志位为只读。

10.6 溢出标志 (MDOV)

出现下列情形时，溢出标志 MDOV 置 1：

- 除数为 0；
- 乘法的结果大于 0x0000 或者 0xFFFF；
- MD3 的最高有效位置 1（MD3.7=1）时开始规范化操作。
- MDU 操作与上述情形不符合时，溢出标志清零。注意：溢出标志只受硬件控制，不能执行写操作。

10.7 MDU 寄存器

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

MD 寄存器 (MD0 – MD5: 0xE9 – 0xEE)

Bit	Field	Type	Initial	Description
7..0	MD[7:0]	R/W	0x00	乘法/除法寄存器。

ARCON 寄存器(0xEF)

Bit	Field	Type	Initial	Description
7	MDEF	R/W	0	MDU 错误标志 MDEF。 显示错误的操作（算法操作重新开始或者被新操作打断）。
6	MDOV	R/W	0	MDU 溢出标志。 MDU 操作时发生溢出。
5	SLR	R/W	0	移位方向。 0：左移； 1：右移。
4..0	SC[4:0]	R/W	0x00	移位计数器。 写入 0x00：执行规范化操作；完成后可读取实际的移位时间； 写入其它值：指定移位操作的次数。

10.8 MDU 限制

算术单元不允许重入代码，不能同时在多个线程或 MAIN 和中断例程中使用。

10.9 示例代码

下面的示例代码程序演示了如何执行 32 位/16 位的 MDU。

```
#include <SN8F5919.H>
#pragma   mdu_r515    //Keil C51 MDU command line

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

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

   while(1);
}

```

11 GPIO

SN8F5910 共有 22 个 GPIO 引脚，和 8051 只有开路输出不同，SN8F5910 还内置推挽输出结构，以增强其驱动能力。

11.1 输入输出控制

由 P0M-P3M 寄存器控制输入输出模式。

Registe	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P1M	-	-	-	-	P13M	P12M	P11M	P10M
P2M	-	-	-	-	-	-	P21M	P20M
P3M	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
P1OC	-	-	-	P05OC	P04OC	-	-	-

P0M 寄存器 (0xF9)

Bit	Field	Type	Initial	Description
7	P07M	R/W	0	P0.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P06M	R/W	0	P0.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P05M	R/W	0	P0.5 的模式选择控制位。 0：输入模式； 1：输出模式。
4	P04M	R/W	0	P0.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P03M	R/W	0	P0.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P02M	R/W	0	P0.2 的模式选择控制位。 0：输入模式； 1：输出模式。
1	P01M	R/W	0	P0.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P00M	R/W	0	P0.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P1M 寄存器 (0xFA)

Bit	Field	Type	Initial	Description
1	P21M	R/W	0	P2.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P20M	R/W	0	P2.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P2M 寄存器 (0xFB)

Bit	Field	Type	Initial	Description
3	P13M	R/W	0	P1.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P12M	R/W	0	P1.2 的模式选择控制位。 0：输入模式； 1：输出模式。
1	P11M	R/W	0	P1.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P10M	R/W	0	P1.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P3M 寄存器 (0xFC)

Bit	Field	Type	Initial	Description
7	P37M	R/W	0	P3.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P36M	R/W	0	P3.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P35M	R/W	0	P3.5 的模式选择控制位。 0：输入模式； 1：输出模式。
4	P34M	R/W	0	P3.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P33M	R/W	0	P3.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P32M	R/W	0	P3.2 的模式选择控制位。 0：输入模式； 1：输出模式。
1	P31M	R/W	0	P3.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P30M	R/W	0	P3.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P1OC 寄存器 (0xE4)

Bit	Field	Type	Initial	Description
4	P05OC	R/W	0	P0.5 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。
3	P04OC	R/W	0	P0.4 开漏输出模式控制位。 0：禁止； 1：使能，输出高电平时变为输入模式。

11.2 输入数据和输出数据

当从 P0~P3 寄存器进行读操作时，当前引脚的逻辑电平将取决于它的外部状态。在某些情况下，当 IO 口在和其他功能复用时，例如 UART 或 I2C，读操作依然可行。

写给 P0~P3 寄存器的值将被马上锁存，然而，只有在 P0M ~P3M 被设置为输出模式之后才会被输出。如果这个引脚已经是输出模式了，任何写到 P0~P3 寄存器的值将会马上输出到这个引脚上。

Registe	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
P1	-	-	-	-	P13	P12	P11	P10
P2	-	-	-	-	-	-	P21	P20
P3	P37	P36	P35	P34	P33	P32	P31	P30

P0 寄存器（0x80）

Bit	Field	Type	Initial	Description
7	P07	R/W	1	读：P0.7 为逻辑低电平。 写入 1/0：输出高电平/低电平（P07M=1 时使能）。
6	P06	R/W	1	读：P0.6 为逻辑低电平。 写入 1/0：输出高电平/低电平（P06M=1 时使能）。
5	P05	R/W	1	读：P0.5 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
4	P04	R/W	1	读：P0.4 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
3	P03	R/W	1	读：P0.3 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
2	P02	R/W	1	读：P0.2 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
1	P01	R/W	1	读：P0.1 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
0	P00	R/W	1	读：P0.0 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。

P1 寄存器（0x90）, P2: 0xA0, P3: 0xFC

Bit	Field	Type	Initial	Description
3	P03	R/W	1	读：P0.3 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
2	P02	R/W	1	读：P0.2 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
1	P01	R/W	1	读：P0.1 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。
0	P00	R/W	1	读：P0.0 为逻辑低电平。 写入 1/0：输出高电平/低电平（P05M=1 时使能）。

11.3 内置上拉寄存器

P0UR-P3UR 寄存器管理每个引脚的内部 100KΩ（典型值）的上拉电阻。

Registe	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07UR	P06UR	P05UR	P04UR	P03UR	P02UR	P01UR	P00UR
P1UR	-	-	-	-	P13UR	P12UR	P11UR	P10UR
P2UR	-	-	-	-	-	-	P21UR	P20UR
P3UR	P37UR	P36UR	P35UR	P34UR	P33UR	P32UR	P31UR	P30UR

P0UR: 0xF1, P1UR: 0xF2, P2UR: 0xF3, P3UR: 0xF4

Bit	Field	Type	Initial	Description
7	P07UR	R/W	0	P0.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P06UR	R/W	0	P0.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P05UR	R/W	0	P0.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4..0				其它

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

11.4 与LCD 功能共用的引脚

SN8F5910 内置 LCD 功能，LCD 驱动输出引脚与 GPIO 共用，通过 PxCON 寄存器进行设置。

Registe	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P3CON	P3CON7	P3CON6	P3CON5	P3CON4	P3CON3	P3CON2	P3CON1	P3CON0
P1CON	-	-	-	-	P1CON3	P1CON2	P1CON1	P1CON0

P3CON: 0x9E

Bit	Field	Type	Initial	Description
7	P3CON7	R/W	1	P37 功能控制位。 0：LCD 功能； 1：GPIO 功能。
6	P3CON6	R/W	1	P36 功能控制位。 0：LCD 功能； 1：GPIO 功能。
5	P3CON5	R/W	1	P35 功能控制位。 0：LCD 功能； 1：GPIO 功能。
4..0				其它

12 外部中断

外部中断源 INT0，INT1 内置边沿触发功能，由 PEDGE 寄存器控制。使能外部中断（EX0/EX1）和全局中断（EAL）后，发生边沿触发事件时，外部中断请求标志位（IE0/IE1）置 1，程序计数器跳转到中断向量（ORG 0x0003/0x000B）并执行中断服务程序。在执行 ISR 之前由硬件清中断请求标志。

12.1 外部中断寄存器

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

PEDGE 寄存器 (0X8F)

Bit	Field	Type	Initial	Description
3..2	EX1G[1:0]	R/W	10	外部中断 INT1 触发沿控制位。 00：保留； 01：上升沿； 10：下降沿（默认）； 11：上升/下降沿。
1..0	EX0G[1:0]	R/W	10	外部中断 INT0 触发沿控制位。 00：保留； 01：上升沿； 10：下降沿（默认）； 11：上升/下降沿。
Else	Reserved	R	0	

12.2 示例代码

下面的示例代码程序演示了如何执行 INT0/INT1 中断。

```
#include <intrins.h>
#include <SN8F5919.h>

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

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

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

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

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

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

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

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

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

13 TCx 16 位定时器/计数器

16 位二进制定时器 TC0/TC1 具有基本定时器、事件计数器和 PWM 功能。基本定时器功能支持标志显示（TCxIRQ 位）和中断操作（中断向量），间隔时间由 TCxM、TCxC 和 TCxR 寄存器控制。事件计数器更改 TC0 时钟源从系统时钟（Fcpu/Fosc/X'tal）为外部时钟类似信号（如 32768Hz 晶振）。TCx 为计数器记录外部时钟的号码以进行测量。TCx 内置占空比/周期可编程控制的 PWM，PWM 的周期和分辨率由 TCx 定时器时钟 rate、TCxR 和 TCxD 寄存器控制，故 PWM 具有良好的性能来处理 IR 载波信号、马达控制和亮度调节等。

TCx 定时器的主要功能如下：

- **16 位可编程计数定时器。**
  根据选择的时钟频率产生周期性的信号。
- **中断功能：**
  TCx 定时器功能支持中断功能，TCx 定时器溢出时，TCxIRQ 有效，系统执行中断服务程序。
- **事件计数器：**
  事件计数器对外部事件计数。
- **占空比/周期可编程控制的 PWM：**
  PWM 的占空比/周期由 TCxR 和 TCxD 寄存器控制。
- **Idle 模式功能：**
  Normal 和 Idle 模式下，所用 TCx 功能（定时器、PWM、事件计数、自动重装）正常工作，STOP 模式下则停止工作。

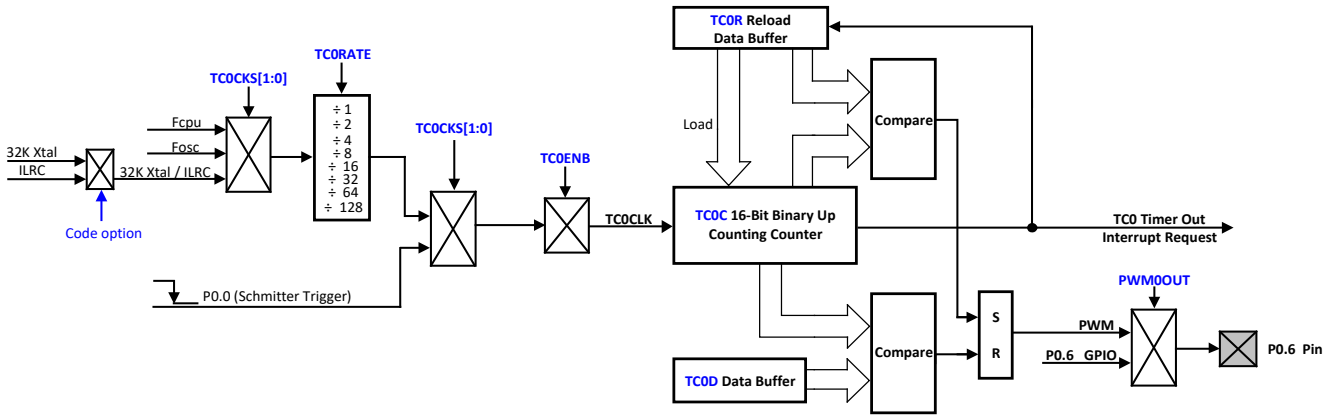
Note:

寄存器	功能描述
TCx = TC0, TC1	定时器计数器
TCxC = TC0C, TC1C	定时器计数器记录数据
TCxD = TCF0, TCF1	定时器 PWM 功能占空比数据
TCxR = TCF0, TCF1	定时器自动装载数据
TCxENB = TC0ENB, TC1ENB	定时器功能使能位
TCxM = TC0M, TC1M	定时器控制寄存器
ETCx = ETC0, ETC1	定时器溢出中断控制位
TCFx = TCF0, TCF1	定时器溢出中断请求位
TCxRATE = TC0RATE, TC1RATE	定时器时钟分频
x = 0, 1	

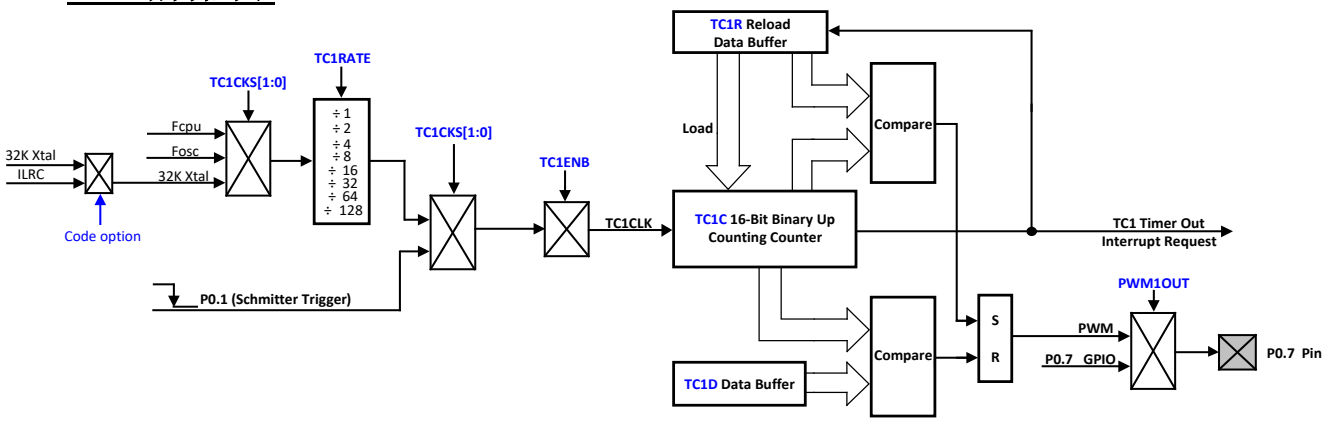
13.1 定时器计数器框图和时钟选择

下图是 TC0、TC1 的时钟选择电路，每个都有 3 个内部时钟源（Fcpu、Fosc 和 32K Xtal）和 1 个外部信号输入源可供选择。下面分别是 TC0、TC1 内部结构框图。

TC0 结构框图:

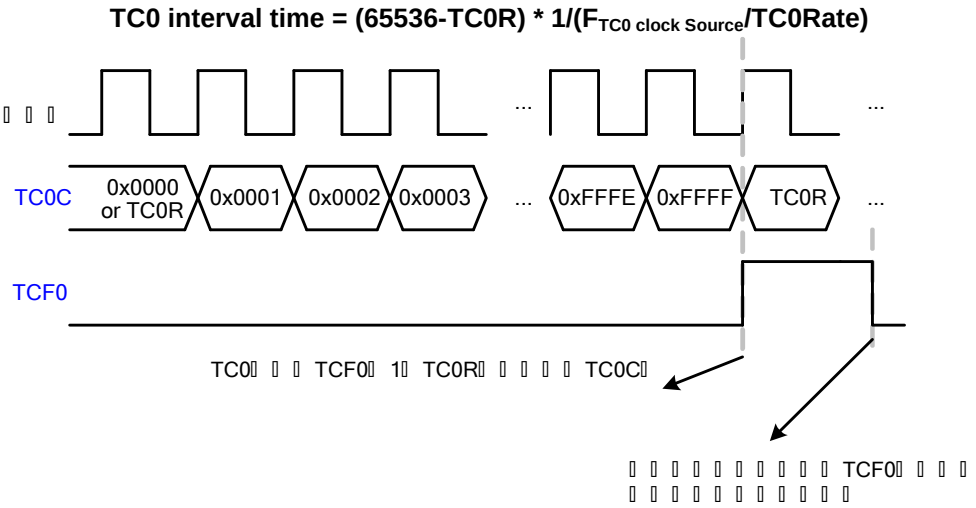


TC1 结构框图:



13.2 TC0 定时器操作 (包括 TC1)

TC0 定时器由 TC0ENB 控制。当 TC0ENB=1 时，TC0 开始计数。一个计数周期即为一个时钟源 rate。TC0ENB=1 时，TC0C 为 TC0 计数器并开始计数，TC0C 溢出（从 0xFFFF 到 0x0000）时，TCF0 置 1 以显示溢出状态。执行中断后，中断标志 TCF0 或由程序清零，或由硬件自动清零。TCx 内置自动重装功能并始终处于使能状态，TC0 溢出后，TC0R 的值由自动存入 TC0C。TC0 为双重缓存器，若 TC0R 的值由程序进行更改，则在下一次溢出时自动装载新值，否则 TC0 间隔时间将出错。如果使能 TC0 中断功能（ETC0=1），在 TC0 溢出时系统执行中断服务程序。



13.3 TC0 计数器功能和寄存器 (包括 TC1)

16 位计数器 TC0C 溢出时，TCF0 置 1，由硬件或软件清零。TC0C 决定 TC0 的间隔时间，由下面公式来计算出一个合适的值。首先必须写入一个合适的值到 TC0C 和 TC0R 寄存器，然后再使能 TC0 定时器以确保第一个周期无误。TC0 溢出后，自动加载 TC0R 寄存器的值到 TC0C 寄存器中，无需通过程序操作。

TC0CH 寄存器 (TC0CH : 0xC5)

Bit	Field	Type	Initial	Description
7..0	TC0CH	R/W	0x00	TC0 计数器的高字节

TC0CL 寄存器 (TC0CL : 0xC4)

Bit	Field	Type	Initial	Description
7..0	TC0CL	R/W	0x00	TC0 计数器的低字节

TC0C 初始值的计算公式如下：

TC0C 初始值 = 65536 - (TC0 间隔时间 * TC0 时钟 rate)

13.4 TC0 自动重装功能和寄存器 (包括TC1)

TC0 内置自动重装功能，TC0R 寄存器存储重装值。TC0C 溢出时，TC0R 的值自动装入 TC0C 中。TC0 定时器工作在计时模式时，要通过修改 TC0R 寄存器来修改 TC0 的间隔时间，而不是通过修改 TC0C 寄存器。在 TC0 定时器溢出后，新的 TC0C 值会被更新，TC0R 会将新的值装载到 TC0C 寄存器中。但在初次设置 TC0M 时，必须要在开启 TC0 定时器前把 TC0C 以及 TC0R 设置成相同的值。

TC0 为双重缓存器结构。若程序对 TC0R 进行了修改，那么修改后的 TC0R 值首先被暂存在 TC0R 的第一个缓存器中，TC0 溢出后，TC0R 的新值就会被存入 TC0R 缓存器中，从而避免 TC0 中断时间出错以及 PWM 误动作。

TC0RH 寄存器 (TC0RH：0xC3)

Bit	Field	Type	Initial	Description
7..0	TC0RH	R/W	0x00	TC0 重装缓存器的高字节

TC0RL 寄存器 (TC0RL：0xC2)

Bit	Field	Type	Initial	Description
7..0	TC0RL	R/W	0x00	TC0 重装缓存器的低字节

TC0R 初始值的计算公式如下：

TC0R 初始值 = 65536 - (TC0 间隔时间 * TC0 时钟 rate)

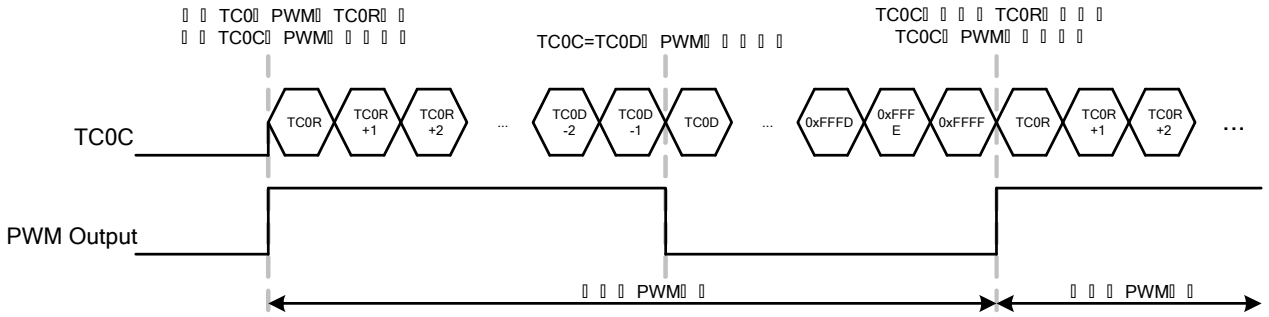
- **例：计算 TC0C 和 TC0R 的值，TC0 的间隔时间为 10ms。**

  TC0 时钟源 Fcpu = 32MHz/32 = 1MHz，TC0RATE=000 (Fcpu/128)
  TC0 间隔时间 = 10ms. TC0 时钟 rate = 32MHz/32/128

  TC0C/TC0R 初始值 = 65536 - (TC0 间隔时间 * 输入时钟)
  = 65536 - (10ms * 32MHz / 32 / 128)
  = 65536 - ($10^{-2} * 32 * 10^{6}$ / 32 / 128)
  = FFB2H

13.5 TC0 (TC1) PWM 功能

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1（使能 PWM 输出）时，由 PWM 输出引脚 P0.6 输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC0R 寄存器控制 PWM 的周期，TC0D 决定 PWM 的占空比（脉冲高电平的长度）。使能 TC0 定时器时，设置 TC0C 的初始值为 TC0R 重装值。当 TC0C=TC0D 时，PWM 高脉冲完成，转换为低电平。当 TC0 溢出（TC0C 计数由 0xFFFF 到 0x0000），一个完整的 PWM 周期完成。PWM 转变为高电平进入下一个周期。PWM 自动将 TC0R 的值装入 TC0C 中，以保持 PWM 的连续性。在 PWM 输出的过程由程序更改 PWM 的周期，则在下一个周期开始输出新的占空比的 PWM 信号。



TC0D 决定 PWM 的占空比。PWM 模式下，TC0R 控制 PWM 的周期，TC0D 控制 PWM 的占空比。其操作基于定时计数器的值。TC0C=TC0D 时，PWM 输出低电平，这样易于配置 TC0D 以选择正确的 PWM 占空比。

TC0D 初始值的计算公式如下:

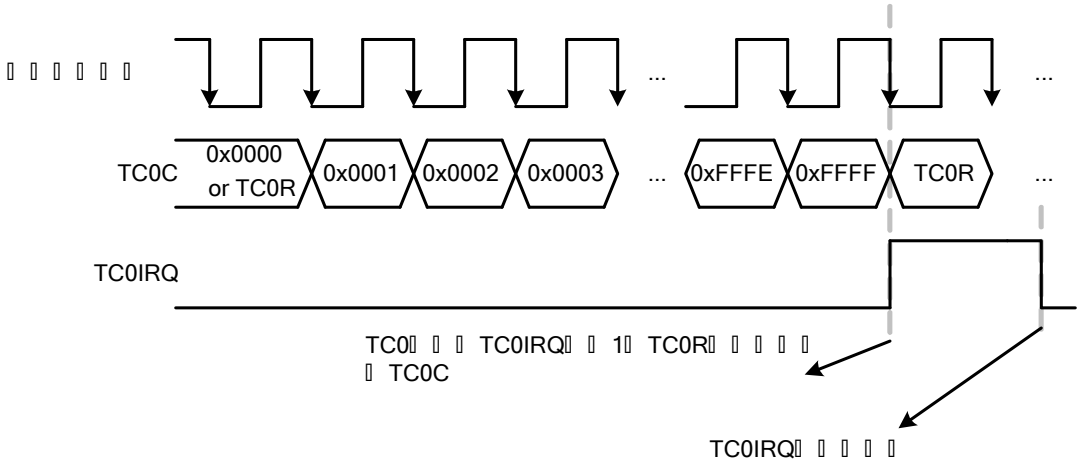
TC0D 初始值 = TC0R + (PWM 高脉冲宽度周期 / TC0 时钟 rate)

- **例：计算 TC0D 的值，PWM 信号为 100Hz，1/3 占空比。**
  TC0 时钟源为 Fcpu = 32MHz/32= 1MHz,C0RATE=000（Fcpu/128）
  TC0R = FFB2H（TC0RH：FFH，TC0RL：B2H），TC0 间隔时间 = 10ms，PWM 周期为 100Hz 占空比 1/3 条件下，高脉冲的宽度约为 3.33ms。

TC0D 间隔时间 = FFB2H + (PWM 高脉冲宽度周期 / TC0 时钟 rate)
= FFB2H + (3.33ms * 32MHz / 32 / 128)
= FFB2H + 1AH = FFCCH

13.6 TC0 事件计数器功能 (包括TC1)

TC0 事件计数器设置 TC0 时钟源为外部输入引脚 P0.0。TC0CKS1=1 时，TC0 时钟源切换为外部输入引脚 P0.0，TC0 事件计数器的触发沿为下降沿触发。下降沿触发时，TC0C 向上计一个数。TC0C 溢出时（从 0xFFFF 到 0x0000），TC0 触发溢出事件。使能 TC0 事件计数器功能时，禁止外部事件计数器输入引脚 GPIO 模式的唤醒功能以避免计数器触发系统唤醒功能而导致不能保持省电模式。使能 TC0 事件计数器功能时，也要禁止外部事件计数器输入引脚的外部中断功能，则 P00IRQ=0。事件计数器通常用于测量外部连续信号，如连续脉冲、R/C 振荡信号等。这些信号的相位与单片机的主时钟信号并不同步，针对不同的应用，使用 TC0 事件计数器进行测量时，需要在程序中计算出信号 rate。



注: TC0 外部输入信号 = P00
TC1 外部输入信号 = P01

13.7 TC0, TC1 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0RATE2	TC0RATE	TC0RATE	TC0CKS1	TC0CKS0	PWM0OU	-
TC0RL	TC0RL7	TC0RL6	TC0RL5	TC0RL4	TC0RL3	TC0RL2	TC0RL1	TC0RL0
TC0RH	TC0RH7	TC0RH6	TC0RH5	TC0RH4	TC0RH3	TC0RH2	TC0RH1	TC0RH0
TC0CL	TC0CL7	TC0CL6	TC0CL5	TC0CL4	TC0CL3	TC0CL2	TC0CL1	TC0CL0
TC0CH	TC0CH7	TC0CH6	TC0CH5	TC0CH4	TC0CH3	TC0CH2	TC0CH1	TC0CH0
TC0DL	TC0DL7	TC0DL6	TC0DL5	TC0DL4	TC0DL3	TC0DL2	TC0DL1	TC0DL0
TC0DH	TC0DH7	TC0DH6	TC0DH5	TC0DH4	TC0DH3	TC0DH2	TC0DH1	TC0DH0
TC1M	TC1ENB	TC1RATE2	TC1RATE	TC1RATE	TC1CKS1	TC1CKS0	PWM1OU	-
TC1RL	TC1RL7	TC1RL6	TC1RL5	TC1RL4	TC1RL3	TC1RL2	TC1RL1	TC1RL0
TC1RH	TC1RH7	TC1RH6	TC1RH5	TC1RH4	TC1RH3	TC1RH2	TC1RH1	TC1RH0
TC1CL	TC1CL7	TC1CL6	TC1CL5	TC1CL4	TC1CL3	TC1CL2	TC1CL1	TC1CL0
TC1CH	TC1CH7	TC1CH6	TC1CH5	TC1CH4	TC1CH3	TC1CH2	TC1CH1	TC1CH0
TC1DL	TC1DL7	TC1DL6	TC1DL5	TC1DL4	TC1DL3	TC1DL2	TC1DL1	TC1DL0
TC1DH	TC1DH7	TC1DH6	TC1DH5	TC1DH4	TC1DH3	TC1DH2	TC1DH1	TC1DH0

TC0M 寄存器 (0xC1)

Bit	Field	Type	Initial	Description
7	TC0ENB	R/W	0	TC0 功能控制位。 0：禁止； 1：使能。
6..4	TC0RATE[2:0]	R/W	0	TC0 定时器时钟分频位。 000：clock/128；　100：clock/8； 001：clock/64；　101：clock/4； 010：clock/32；　110：clock/2 ； 011：clock/16；　111：clock/1。
3..2	TC0CKS[1:0]	R/W	0	TC0 定时器时钟源选择位。 00：Fcpu； 01：Fosc； 10：外部 32K 晶振； 11：P0.0（事件计数器）
1	PWM0OUT	R/W	0	PWM0 输出控制位。 0：禁止，P0.6 为 GPIO 模式； 1：使能，P0.6 输出 PWM 信号。
0	Reserved	R	0	

TC1M 寄存器 (0xC9)

Bit	Field	Type	Initial	Description
7	TC1ENB	R/W	0	TC1 功能控制位。 0：禁止； 1：使能。
6..4	TC1RATE[2:0]	R/W	0	TC1 定时器时钟分频位。 000：clock/128；　100：clock/8； 001：clock/64；　101：clock/4； 010：clock/32；　110：clock/2 ； 011：clock/16；　111：clock/1。
3..2	TC1CKS[1:0]	R/W	0	TC1 定时器时钟源选择位。 00：Fcpu； 01：Fosc； 10：外部 32K 晶振； 11：P0.1（事件计数器）
1	PWM1OUT	R/W	0	PWM1 输出控制位。 0：禁止，P0.7 为 GPIO 模式； 1：使能，P0.7 输出 PWM 信号。
0	Reserved	R	0	

定时器时钟控制列表：

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

注: TCx = TC0, TC1

IEN2 寄存器 (0X9A)

Bit	Field	Type	Initial	Description
4	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
2	ETC1	R/W	0	TC1 溢出中断控制位。 0：禁止； 1：使能。
1	ETC0	R/W	0	TC0 溢出中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IRCON 寄存器(0xC0)

Bit	Field	Type	Initial	Description
7	ADCF	R/W	0	ADC 中断请求标志位。 0：无 ADC 中断请求； 1：请求 ADC 中断。
5	TCF1	R/W	0	TC1 中断请求标志位。 0：无 TC1 中断请求； 1：请求 TC1 中断。
4	TCF0	R/W	0	TC0 中断请求标志位。 0：无 TC0 中断请求； 1：请求 TC0 中断。
2	TF0	R/W	0	T0 中断请求标志位。 0：无 T0 中断请求； 1：请求 T0 中断。
1	IE1	R/W	0	外部 P0.1 中断（INT1）中断请求标志位。 0：无 INT1 中断请求； 1：请求 INT1 中断。
0	IE0	R/W	0	外部 P0.0 中断（INT0）中断请求标志位。 0：无 INT0 中断请求； 1：请求 INT0 中断。
Else	Reserved	R	0	

TC0CH / TC1CH 寄存器 (TC0CH : 0xC5, TC1CH : 0xCD)

Bit	Field	Type	Initial	Description
7..0	TCxCH	R/W	0x00	TCx 计数器的高字节。

TC0CL / TC1CL 寄存器 (TC0CL : 0xC4, TC1CL : 0xCC)

Bit	Field	Type	Initial	Description
7..0	TCxCL	R/W	0x00	TCx 计数器的低字节。

TC0RH / TC1RH 寄存器 (TC0RH : 0xC3, TC1RH : 0xCB)

Bit	Field	Type	Initial	Description
7..0	TCxRH	R/W	0x00	TCx 重装缓存器的高字节。

TC0RL / TC1RL 寄存器 (TC0RL : 0xC2, TC1RL : 0xCA)

Bit	Field	Type	Initial	Description
7..0	TCxRL	R/W	0x00	TCx 重装缓存器的低字节。

TC0DH / TC1DH 寄存器 (TC0DH : 0xC7, TC1DH : 0xCF)

Bit	Field	Type	Initial	Description
7..0	TCxDH	R/W	0x00	TCx 占空比控制的高字节

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

Bit	Field	Type	Initial	Description
7..0	TCxDL	R/W	0x00	TCx 占空比控制的低字节

13.8 示例代码

下面的示例程序展示了如何执行 TC0/TC1 中断。

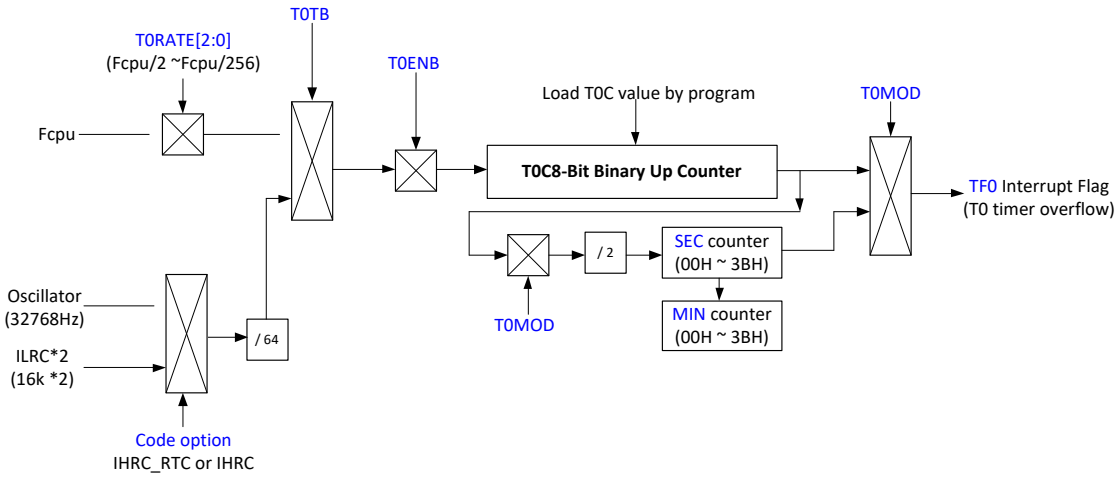
```
#include <intrins.h>    // for _nop_
#include < SN8F5919.h>

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

void Init_SysCLK(void)
{
    /* Clock Switch Select Register */
    CLKSEL = 0x02;        // Fcpu = Fhosc/32
    CLKCMD = 0x69;        // clock switch start

    /* System Control Register */
    CKCON &= 0x00;
}

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

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

        // To Do...
 }
}

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

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

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

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

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

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

14 定时器 T0

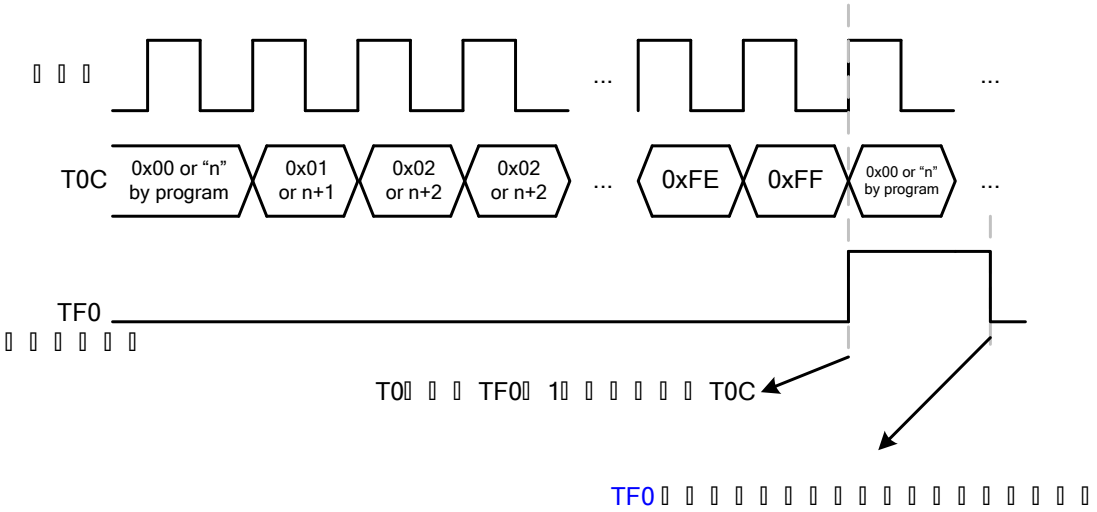
8 位二进制基本定时器 T0，支持标志指示（TF0）和中断操作（中断向量 0x93）。可以通过 T0M 和 T0C 寄存器控制间隔时间，支持 RTC 功能。T0 内置 STOP 模式唤醒功能，STOP 模式下，T0 溢出时，系统被唤醒进入 Normal 模式。

- **8 位可编程计数定时器**：根据选择的时钟频率周期性的产生中断请求。
- **中断功能**：T0 定时器支持中断功能，当 T0 溢出，TF0 有效，程序计数器跳到中断向量地址执行中断。
- **RTC 功能**：T0 RTC 功能由 T0TB 位控制，T0MOD=0 时，RTC 的周期为 0.5sec @32KHz，T0MOD=1 时，T0 中断周期为 60sec @32KHz。
- **STOP 模式功能**：STOP 模式下，T0 正常工作，并可在 **T0ENB=1&T0TB=1** 时将系统从 STOP 模式被唤醒，T0 溢出后，TF0 有效，系统被唤醒。



14.1 T0 定时器操作

T0 定时器由 T0ENB 控制。当 T0ENB=0 时，T0 停止工作；当 T0ENB=1 时，T0 开始计数。T0C 溢出（从 0xFF 到 0x00）时，TF0 置 1 显示溢出状态并由程序清零。T0 没有双重缓存器，故 T0 溢出时由程序加载新值给 T0C，以选定合适的间隔时间。如果使能 T0 中断（ET0=1），T0 溢出后系统执行中断服务程序，在中断下必须由程序清 TF0。T0 可以在 Normal 模式、Idle 模式和 Stop 模式下工作。Stop 模式下，T0 继续计数，设置 TF0I 为 1，T0 溢出时系统被唤醒。



T0 时钟源为 Fcpu（指令周期），由 T0RATE[2:0]控制其分频 Fcpu/2~Fcpu/256。T0 长度是 8 位（256 步），一个计数周期是每个输入时钟的周期。

T0RATE[2:0]	T0 时钟	高速模式(Fcpu=1 MIP)	
		最高溢出时间	单步时间= max/256
000	Fcpu / 256	65.563 ms	256 us
001	Fcpu / 128	32.768 ms	128 us
010	Fcpu / 64	16.384 ms	64 us
011	Fcpu / 32	8.192 ms	32 us
100	Fcpu / 16	4.096 ms	16 us
101	Fcpu / 8	2.048 ms	8 us
110	Fcpu / 4	1.024 ms	4 us
111	Fcpu / 2	0.512 ms	2 us

14.2 T0 模式控制寄存器

模式寄存器 T0M 设置 T0 的工作模式，包括 T0 前置分频器、时钟源等，这些设置必须在使能 T0 定时器之前完成。

T0M 寄存器 (0xBB)

Bit	Field	Type	Initial	Description
7	T0ENB	R/W	0	T0 使能控制位。 0：禁止； 1：使能。
6..4	T0RATE[2:0]	R/W	0	T0 时钟源选择位。 000：Fcpu/256.　100：Fcpu/16. 001：Fcpu/128.　101：Fcpu/8. 010：Fcpu/64.　110：Fcpu/4. 011：Fcpu/32.　111：Fcpu/2.
1	T0MOD	R/W	0	T0 中断模式控制位。 0：T0C 寄存器溢出时 T0 中断； 1：SEC 寄存器溢出时 T0 中断。
0	T0TB	R/W	0	T0 RTC 功能控制位。 0：禁止 RTC 功能，T0 时钟源为 Fcpu； 1：使能 RTC 功能，T0 时钟源为低速时钟。

14.3 T0C 计数寄存器

8 位计数器 T0C 溢出时，TF0 置 1 并由程序清零，用来控制 T0 的中断间隔时间。必须保证写入正确的值到 T0C 寄存器，然后使能 T0 定时器以保证第一个周期准确无误。T0 溢出后，由程序加载一个正确的值到 T0C 寄存器。

T0C 寄存器 (0xBC)

Bit	Field	Type	Initial	Description
7..0	T0C[7:0]	R/W	0	T0 计数寄存器

T0C 初始值的计算公式如下：

T0C 初始值 = 256 - (T0 中断间隔时间 * T0 时钟 rate)

- **例：计算 T0C，T0 的间隔时间为 10ms，**
  T0 时钟源 Fcpu=32MHz/32=1MHz，T0RATE = 001（Fcpu/128）。
  T0 间隔时间=10ms，T0 时钟 Rate=32MHz/32/128
  T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）
  = 256 -（10ms * 32MHz / 32 / 128）
  = 256 -（10-2 * 32 * 106 / 32 /128）
  = B2H

14.4 其它相关寄存器

SEC 寄存器 (0xBD)

Bit	Field	Type	Initial	Description
5..0	SEC[5:0]	R/W	0	T0 SEC 计数器 SEC 计数器范围为 0x00 ~ 0x3B (0 ~ 59) SEC 的值为 0x3B 且再加 1 时，计数器溢出并复位为 0x00，且 MIN 计数器加 1。and MIN counter will increase “1”。

MIN 寄存器 (0xBE)

Bit	Field	Type	Initial	Description
5..0	MIN[5:0]	R/W	0	T0 MIN 计数器。 MIN 计数器范围为 0x00H ~ 0x3B (0 ~ 59)。 MIN 的值为 0x3B 再加 1 时，计数器溢出并复位为 0x00，且 MIN 计数器加 1。and MIN counter will increase “1”。

IEN0 寄存器(0xBC)

Bit	Field	Type	Initial	Description
1	ET0	R/W	0	T0 中断使能控制位。 0：禁止； 1：使能。
Else				参考其它章节。

IRCON 寄存器(0xC0)

Bit	Field	Type	Initial	Description
2	TF0	R/W	0	T0 中断请求标志位。 0：无 T0 中断请求； 1：请求 T0 中断。
Else				参考其它章节。

14.5 示例代码

下面的示例程序展示了如何执行 T0 中断。

```
#include <intrins.h>     // for _nop_
#include <SN8F5919.h>

void T0_Init(void);

void Init_SysCLK(void)
{
    /* Clock Switch Select Register */
    CLKSEL = 0x02;      // Fcpu = Fhosc/32
    CLKCMD = 0x69;      // clock switch start

    /* System Control Register */
    CKCON &= 0x00;
}

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

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

          // To Do...
    }
}

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

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

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

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

下面的示例程序展示了在 STOP 模式下如何执行 T0（带 0.5s 或 60s）唤醒功能。

```
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

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

void Init_SysCLK(void)
{
    /* Clock Switch Select Register */
    CLKSEL = 0x02;      // Fcpu = Fhosc/32
    CLKCMD = 0x69;      // clock switch start

    /* System Control Register */
    CKCON &= 0x00;
}

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

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

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

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

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

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

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

15 Buzzer 功能

Buzzer 功能由 BZRENB 位控制，可设置为 GPIO 模式或 Buzzer 模式。Buzzer 通过 P1.2 输出方波信号，50%占空比，其输出频率通过设置 BZRCKS[1:0]寄存器控制。

BZRM 寄存器 (0xD6)

Bit	Field	Type	Initial	Description
2..1	BZRCKS[1:0]	R/W	0	Buzzer 输出频率。 00：0.98KHz； 01：1.96KHz； 10：3.9KHz； 11：7.8KHz。
0	BZRENB	R/W	0	Buzzer 输出控制位。 0：GPIO 模式； 1：Buzzer 模式，P12 输出 Buzzer 信号。

***例**: **Buzzer 功能示例程序。**

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

16 LCD 驱动

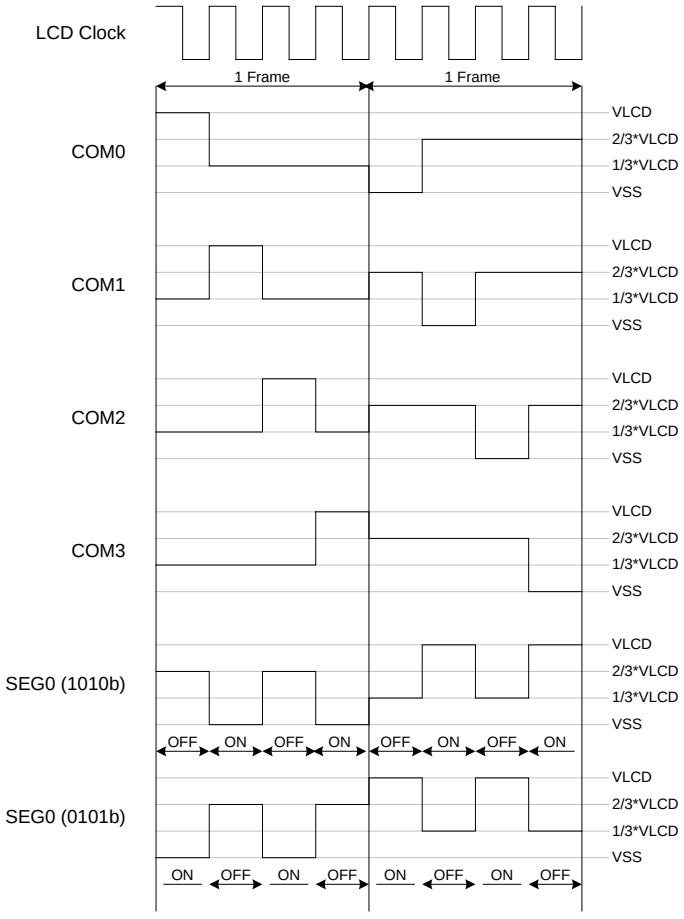
SN8F5910 LCD 驱动为 C 型结构，具体为 4 x 36 或 6 x 34，LCD 的扫墓时序为 1/4 或 1/6 占空比，1/3 偏压，具有 144 点或 204 点 LCD 驱动。C 型 LCD 驱动可输出可调节的 VLCD 输出电压和可调节的驱动电源以支持 LCD 面板的多样性。VLCD 引脚与 DVSS 直接必须连接一个 0.1uF 的电容用于 C 型 LCD 驱动。

LCD 督导输出 4 或 6 个 com 波形，由 LCDMODE 位控制，输出帧 rate 由 LCDRATE 位控制，详见下表所示。

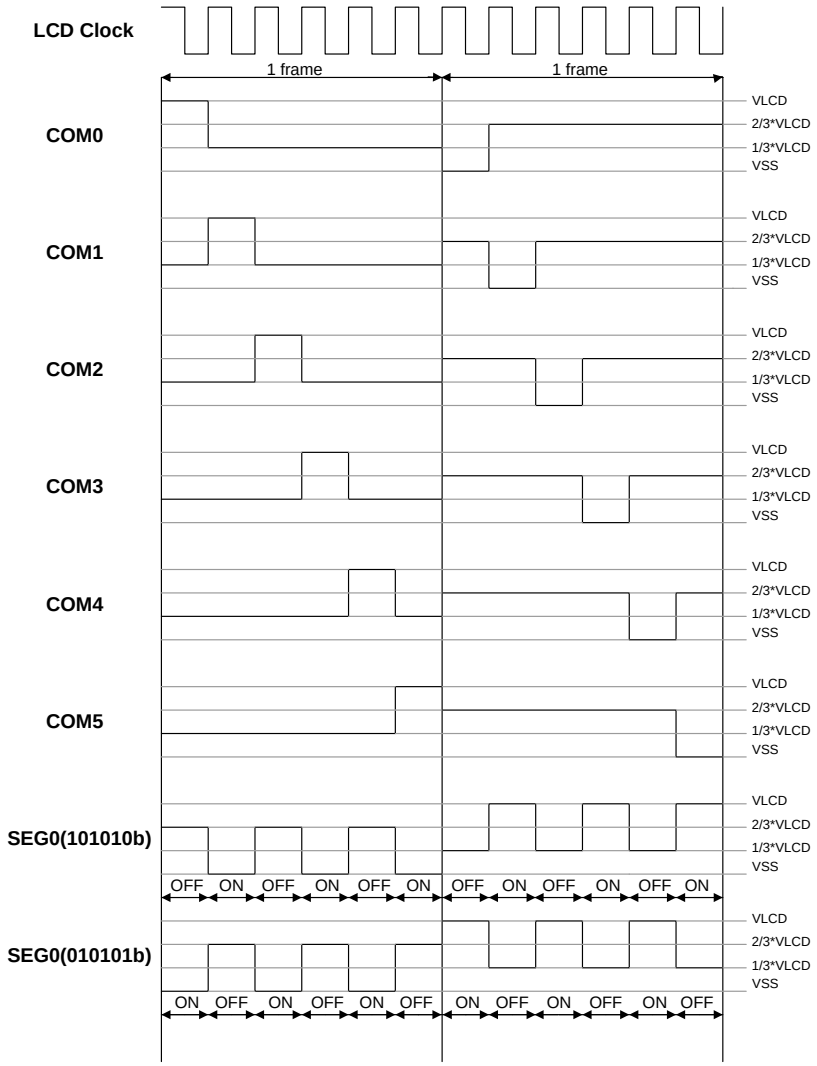
- **Stop 模式功能**：STOP 模式下 LCDEN=1&STOP=1 时，LCD 功能保持工作状态。
- **低功耗模式功能**：由 BGM 位控制 LCD 低功耗功能。

控制寄存器		LCD 时钟 (Hz)	Frame Rate (Hz)	Type
LCDRATE	LCDMODE			
0	0	32KHz / 128=256	64	4-COM (1/4 duty)
0	1		43	6-COM (1/6 duty)
1	0	32KHz / 64 =512	128	4-COM (1/4 duty)
1	1		85	6-COM (1/6 duty)

LCD 驱动波形, 1/4 duty, 1/3 bias :



LCD 驱动波形, 1/6 duty, 1/3 bias :



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

LCD 时钟源取决于 Code Option，如下表：

CPU Clock Source	LCD Clock
IHRC_RTC	外部 32K 振荡器
IHRC	内部 32K RC

SYM.	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
FILRC	内部低速时钟	VDD = 5V @ 25°C	24	32	48	KHz

16.1 LCD RAM 位置

LCD RAM 位于地址 0xF000 到 0xF02B 之间。

LCD RAM 地址与 COM0 ~ COM3 vs. SEG0 ~ SEG35 LCD 的关系如下表所示：

	Bit0 COM0	Bit1 COM1	Bit2 COM2	Bit3 COM3	Bit4 -	Bit5 -	Bit6 -	Bit7 -
SEG 0	F000H.0	F000H.1	F000H.2	F000H.3	N/A	N/A	N/A	N/A
SEG 1	F001H.0	F001H.1	F001H.2	F001H.3	N/A	N/A	N/A	N/A
SEG 2	F002H.0	F002H.1	F002H.2	F002H.3	N/A	N/A	N/A	N/A
SEG 3	F003H.0	F003H.1	F003H.2	F003H.3	N/A	N/A	N/A	N/A
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 35	F023H.0	F023 H.1	F023 H.2	F023 H.3	N/A	N/A	N/A	N/A

LCD RAM 地址与 COM0 ~ COM5 vs. SEG2 ~ SEG35LCD 的关系如下表所示：

	Bit0 COM0	Bit1 COM1	Bit2 COM2	Bit3 COM3	Bit4 COM4	Bit5 COM5	Bit6 -	Bit7 -
SEG 0 (COM4)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SEG 1 (COM5)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SEG 2	F002H.0	F002H.1	F002H.2	F002H.3	F002H.4	F002H.5	N/A	N/A
SEG 3	F003H.0	F003H.1	F003H.2	F003H.3	F003H.4	F003H.5	N/A	N/A
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 35	F02BH.0	F02B H.1	F02B H.2	F02B H.3	F02B H.4	F02B H.5	N/A	N/A

16.2 LCD控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	-	-	-	LCDBNK	LCDMODE	LCDRATE	LCDPEN	LCDEN
LCDM2	VAR1	VAR0	BGM	DUTY1	DUTY0	VCP2	VCP 1	VCP0

LCDM1 寄存器 (0xAD)

Bit	Field	Type	Initial	Description
4	LCDBNK	R/W	0	LCD blank 控制位。 0：正常显示； 1：关闭所有的 LCD 点。
3	LCDMODE	R/W	0	LCD 驱动 4-COM 或 6-COM 控制位。 0：4-COM； 1：6-COM。
2	LCDRATE	R/W	0	LCD 时钟 rate 控制位。 0：256Hz； 1：512Hz。
1	LCDPEN	R/W	0	C-Type LCD pump 使能位。 0：禁止； 1：使能。
0	LCDEN	R/W	0	LCD 驱动控制位。 0：禁止； 1：使能。

***例: 必须先使能 LCD Pump（LCDPEN=1）并等待 1ms，然后使能 LCD 驱动（ LCDEN=1）**

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

***例: STOP 模式下 LCDEN=1&STOP=1 时，LCD 功能保持工作状态。**

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

LCDM2 寄存器 (0xB2)

Bit	Field	Type	Initial	Description
7..6	VAR[1:0]	R/W	0	VLCD Pump 输出波纹控制位。 00：±30 波纹控（设置为 00）； 0thers：保留。
5	BGM	R/W	1	LCD 低功耗模式控制位。 0：使能低功耗模式; 1：Normal 模式。
4..3	DUTY[1:0]	R/W	0	LCD pump 输出驱动能力控制位。 00：25%；　　10：50%； 01：37.5%；　　11：100%。 *注：BGM=0 时，DUTY [1:0]可编程控制用于 LCD 低功耗模式。
2..0	VCP[2:0]	R/W	011	VLCD 输出电压控制位。 000：保留；　　011：3.2V； 001：2.8V；　　100：3.6V； 010：3.0V；　　101：4.5V； Others：保留。 *注：电池电压必须大于 1.9V。 @VLCD 输出电压 2.6~3.6V

***例: LCD 低功耗模式。**

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

P3CON 寄存器 (0x9E)

Bit	Field	Type	Initial	Description
7..0	P3CON[7:0]	R/W	0xFF	P3 功能控制位。 0：设置为 LCD 功能（SEG35 ~ SEG28）； 1：设置为 GPIO 功能（P30 ~ P37）。

P1CON Register (0x9F)

Bit	Field	Type	Initial	Description
3..0	P1CON[3:0]	R/W	0xFF	P1 功能控制位。 0：设置为 LCD 功能(SE32 ~SEG35) 1：设置为 GPIO 功能(P10 ~ P13)

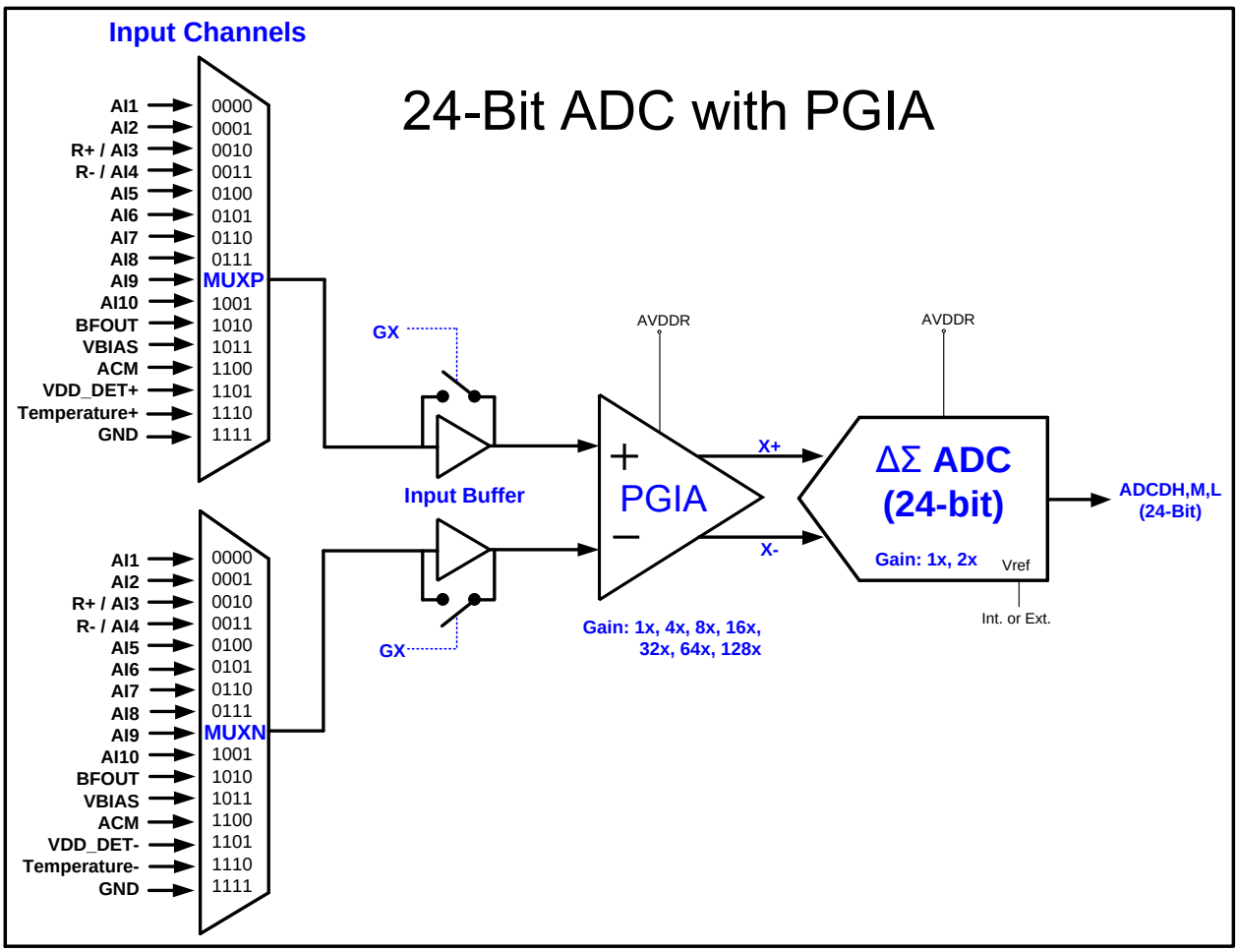
16.3 示例代码

下面的示例程序展示了如何在 STOP 模式下执行 LCD 功能。

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

17 ADC

SN8F5910 集成高分辨率 ΔΣ ADC，低噪音可编程增益放大器（PGIA），可输出 24 位分辨率 KHz，带 18 位精度。ADC 的转换 rate 范围为 2Hz 到 7.8KHz，ADC 的参考电压由内部或者外部（R+/R-）提供。ADC 可在 NORMAL 模式和 IDLE 模式下正常工作，若使能中断，ADC 结束后可执行中断或者唤醒功能。ADC 内部增益的选择选项为：1x 和 2x。PGIA 通过配置的 2 个多路选通器，MUXP 和 MUXN 输入，提供差分类型的输入和单结束类型的输入。优化 ADC 可在传感测量或医学测量中来测量低电平的单极或双极信号。低噪音、高稳定性可编程增益放大器（PGIA）的选择选项为：1x，4x，8x，16x，32x，64x 和 128x，以在 ADC 中适应这些应用。



17.1 操作配置

在开始 AD 转换之前，必须先完成下面配置的设置。具体设置如下：

1、必须使能模拟电源 AVDDR，ACM 和 Bandgap。（通过 AVDDREN，ACMEN 和 BGREN 位设置）

2、选择 ADC 输入通道。（通过 MUXP[3:0]和 MUXN[3:0]设置）

3、输入缓存器设置：若在应用中应用 PGIA 则关闭；若关闭 PGIA 则开启，传感器要求高输入阻抗。（由 GX 位设置）

4、若要求则使能 PGIA，并设置 PGIA 增益比率。（由 PGIAEN，GS[2:0]位控制）

5、开启 Chopper 功能和 PGIA 和 ADC 的时钟。（由 PCHPEN，ACHPEN[1:0]，AMPCKS[1:0]位设置）

6、设置 ADC 参考电压为内部电压或外部 R+/R-。（有 IRVS[2:0]位控制）

7、通过 ADC 时钟和 OSR Over-Sample Rate 设置 ADC 转换 rate。（由 ADCKS[1:0]和 OSR[2:0]位设置）

8、若需要则设置 ADC 中断功能并使能 ADC。（由 EADC 和 ADCEN 位控制）

9、使能 ADENB 位后，ADC 准备好进行转换。（24 位 ADCDH，ADCDM，ADCDL）

17.2 ADC输入通道和PGIA

ADC 内置 10 个外部通道输入和 6 个内部通道输入，AI1~AI10 用于测量 10 个差分模拟信号源，由 CHS[4:0]位控制。6 个内部通道包括　BFOUT，VBIAS,，ACM，GND，VDD_DET（VDD 检测）和温度通道。ADC 外部参考电压通过 AI3 和 AI4 输入。VDD_DET 通道为 1/8*VDD 电压输入 ADC，以通过 ADC 测量电池电量。温度传感器嵌入到 IC 内部以测量 IC 周边的温度或粗略测试室温。

SN8F5910 包括低噪音、高稳定性可编程增益放大器（PGIA），其选择选项为：1x，4x，8x，16x，32x，64x 和 128x，作为差分模式或者单端模式进行输入配置。PGIA 有高输入阻抗特性（约 5 G-Ohm），适合连接传感器进行高输入阻抗应用。

模拟输入信号通道选择列表：

MUXP[3:0]	PGIA 正极输入	MUXN[3:0]	PGIA 负极输入
0000	AN1	0000	AN1
0001	AN2	0001	AN2
0010	AN3 / R+	0010	AN3 / R+
0011	AN4 / -	0011	AN4 / R-
0100	AN5	0100	AN5
0101	AN6	0101	AN6
0110	AN7	0110	AN7
0111	AN8	0111	AN8
1000	AN9	1000	AN9
1001	AN10	1001	AN10
1010	BFOUT	1010	BFOUT
1011	VBIAS	1011	VBIAS
1100	ACM	1100	ACM
1101	VDD_DET+	1101	VDD_DET-
1010	Temperature+	1110	Temperature-
1111	AVss	1111	AVss

17.3 模拟输入缓存器

输入缓存器包括 ADC 信号输入缓存器和 ADC 外部参考输入缓存器 R+/R-，提供一个高阻抗的模拟输入信号，以减小 ADC 的输入电流，在进行灵敏测量时以避免负载的影响。PGIA 选择 1x 时，传感器的输出信号避开 PGIA，直接连接到 ADC 的输入端。在这种情况下，必须使能输入缓存器功能（GX=1）。若 ADC 选择外部 Vref，输入缓存器 R+/R-也必须使能（GR=1）。

17.4 ADC 参考电压

ADC 参考电压可选择外部 R+/R-输入或者内部电压源输入，通过设置 ADCM1 寄存器的 IRVS[2:0]进行选择，具体见下表所示。

IRVS[2:0]		ADC 参考电压			
		AVDDR 2.7V	AVDDR 3.0V	AVDDR 3.3V	AVDDR 3.6V
0xx	External	R+ / R-			
01x	Temperature Vref	0.8V			
100	0.2*AVDDR	0.54V	0.6V	0.66V	0.72V
101	0.3*AVDDR	0.81V	0.9V	0.99V	1.08V
110	0.4*AVDDR	1.08V	1.2V	1.32V	1.44V
111	0.5*AVDDR	1.35V	1.5V	1.65V	1.8V

17.5 ADC增益和ADC 输出码

ADC 内置内部增益选项：x1 和 x2，用于额外的信号的放大。通过 ADCM1 寄存器的 ADGN[2:0]位选择 ADC 的增益。

ADC 输出 24 位数据，由 ADCDH，ADCDM，ADCDL 寄存器组合而成，in 2's compliment 带符号位数字格式，位 ADCB23 为 ADC 数据的符号位。参考下面公式可计算 ADC 的转换数据，ADC 输出码计算如下：

24 位格式，ADC 输出码位于 ADCDH，ADCDM 和 ADCDL：

$$\frac{\Delta Input \times PGIA\_Gain \times ADC\_Gain}{Vref} \times 2^{(24-1)} = +(2^{(24-1)} - 1) \sim -2^{(24-1)}$$

$PGIA\_Gain : x1 \sim x128$

$ADC\_Gain : x1 \sim x2$

$Vref : 0.3V \sim 1.8V$

16 位格式，ADC 输出码位于 ADCDH 和 ADCDM：

$$\frac{\Delta Input \times PGIA\_Gain \times ADC\_Gain}{Vref} \times 2^{(16-1)} = +32767 \sim -32768$$

$PGIA\_Gain : x1 \sim x128$

$ADC\_Gain : x1 \sim x2$

$Vref : 0.3V \sim 1.8V$

17.6 ADC 中断控制

ADC 完成转换后，无论是否使能 EADC，ADCF=1。若 EADC 和触发时间 ADCF 都设置为 1，系统执行中断。若 EADC=0，ADCF=1，则系统不会执行中断。用户需注意多中断下的操作。

17.7 ADC 转换Rate

ADC 提供可变的输出转换 rate，范围为 2Hz~7.8KHz，由 ADCM2 寄存器的 ADCKS[1:0]和 OSR[2:0]位控制。调节 ADC 时钟（ADCKS）和 OSR，可获得合适的 ADC 输出 word rate。在高分辨率应用下，OSR 设置建议的最大值 32768Hz。低速输出 word rate 的 ADC 输出码要比快速的稳定。ADC 应用中，需要权衡 ADC 的输出速率和稳定性（ENOB）。下表显示 ADC 输出 word rate 的设置。

ADC 转换 Rate = ADC Clock / OSR

使能 ADC 或者切换输入通道后，ADC 输出在第 3 个数据开始稳定，第 1 个和第 2 个 ADC 输出数据是不稳定的，第 3、第 4 和第 5……数据都是稳定的。

ADC 转换 Rate 列表：

ADCKS [1:0]	OSR [2:0]	ADC 时钟	输出速率	ADCKS [1:0]	OSR [2:0]	ADC 时钟	输出速率
00	000	250KHz	3.9 kHz	01	000	125KHz	1.95 kHz
00	001		1.95 kHz	01	001		976 Hz
00	010		976 Hz	01	010		488 Hz
00	011		244 Hz	01	011		122 Hz
00	100		61 Hz	01	100		30.5 Hz
00	101		30.5 Hz	01	101		15.2 Hz
00	110		15.2 Hz	01	110		7.6 Hz
00	111		7.6 Hz	01	111		3.8 Hz
01	000	333KHz	5.2 kHz	11	000	166.5KHz	2.6 kHz
01	001		2.6 kHz	11	001		1.3 kHz
01	010		1.3 kHz	11	010		0.65 kHz
01	011		325 Hz	11	011		162.6 Hz
01	100		81.3 Hz	11	100		40.7 Hz
01	101		40.7 Hz	11	101		20.3 Hz
01	110		20.3 Hz	11	110		10.2 Hz
01	111		10.2 Hz	11	111		5.1 Hz

17.8 ADC 控制寄存器

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

CHS 寄存器 (0xD1)

Bit	Field	Type	Initial	Description
7..4	MUXP[3:0]	R/W	0	ADC 正极输入通道选择位。 0000 : AI1　0101 : AI6　1010 : BFOUT 0001 : AI2　0110 : AI7　1011 : VBIAS 0010 : AI3/R+　0111 : AI8　1100 : ACM 0011 : AI4/R-　1000 : AI9　1101 : VDD_DET+ 0100 : AI5　1001 : AI10　1110 : Temperature+ 1111 : AVSS
3..0	MUXN[3:0]	R/W	0	ADC 负极输入通道选择位。 0000 : AI1　0101 : AI6　1010 : BFOUT 0001 : AI2　0110 : AI7　1011 : VBIAS 0010 : AI3/R+　0111 : AI8　1100 : ACM 0011 : AI4/R-　1000 : AI9　1101 : VDD_DET- 0100 : AI5　1001 : AI10　1110 : Temperature- 1111 : AVSS

VREG 寄存器 (0xD2)　参考"Charge pump regulator"章节。

AMPM 寄存器 (0xD3)

Bit	Field	Type	Initial	Description
7	GX	R/W	0	ADC 输入缓存器使能位。 0：禁止； 1：使能。 （PGIA 增益设置为 x1）
6..5	AMPCKS[1:0]	R/W	00	PGIA chopper 频率控制位。 00：ADCKS / 128　　10：ADCKS / 16 01：ADCKS / 32　　11：ADCKS / 8 （在所有应用中请设置为 01）
4..2	GS[2:0]..	R/W	000	PGIA 增益选择位。 000 : 1x　　100 : 32x 001 : 4x　　101 : 64x 010 : 8x　　110 : 128x 011 : 16x　　111 : 1x + Buff mode (Always PGIAEN=1)
1	PCHPEN	R/W	0	PGIA chopper 使能位。 0：禁止； 1：使能。（使能 PGIA 时必须使能该位）
0	PGIAEN	R/W	0	PGIA 功能使能位。 0：禁止； 1：使能。

***例**：在应用中设置 PGIA 增益为 1x 时，AI+/AI-信号绕过 PGIA，直接输入 ADC。禁止 PGIA（PGIAEN=0）以省电，必须使能 ADC 输入缓存器（GX=1）用于 ADC 的输入高阻抗特性。

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

□ □ **注：当输入缓存器使能(GX=1 or GR=1)，信号的输入绝对电压范围必须为 0.4V~AVDDR-1.0V。**

ADCM1 寄存器 (0xD4)

Bit	Field	Type	Initial	Description
7	GR	R/W	0	ADC 参考缓存器使能位。 0：禁止； 1：使能。 （ADC Vref 设置为外部 R+/R-）
6..4	IRVS[2:0]	R/W	000	ADC 参考电压选择位。 0xx：外部 R+/R-； 100：内部 0.2 x AVDDR； 101：内部 0.3 x AVDDR； 110：内部 0.4 x AVDDR； 111：内部 0.5 x AVDDR。
3	ADGN	R/W	0	ADC 增益选择位。 0：1x； 1：2x。
1	ACHPEN	R/W	00	ADC chopper 控制位。 11：使能。（必须在所有应用中设置为 11）
0	ADCEN	R/W	0	ADC 功能使能位。 0：禁止； 1：使能。

***例**: ADC Vref 设置为外部 R+/R-时，必须使能 ADC 的 R+/R-输入缓存器（GR=1）用于 ADC Vref 输入高阻抗特性。

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

□ □ **注：当输入缓存器使能(GX=1 or GR=1)，信号的输入绝对电压范围必须为 0.4V~1.4V。**

ADCM2 寄存器 (0xD5)

Bit	Field	Type	Initial	Description
5..4	ADCKS[1:0]	R/W	0	ADC 时钟选择位。 00：500KHz；　　10：125KHz； 01：250KHz；　　11：62.5KHz。 （建议设置为 250KHz）
3..1	OSR[2:0]	R/W	000	ADC Over-Sampling-Rate 选择位。 000：64；　　100：4096； 001：128；　　101：8192； 010：256；　　110：16384； 011：1024；　　111：32768。
0	DRDY	R/W	0	ADC 转换完成位。 0：ADCDH、ADCDL 和 ADCDLL 转换数据没有准备好； 1：ADC 输出（更新）新的转换数据到 ADCDH、ADCDL 和 ADCDLL。

ADCDH 寄存器(0xB7)，ADC 输出高字节寄存器

ADCDM 寄存器(0xB6)，ADC 输出中间字节寄存器

ADCDL 寄存器(0xB5)，ADC 输出低字节寄存器

IEN2 寄存器 (0X9A)

Bit	Field	Type	Initial	Description
4	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
	others	R/W	0	参考中断章节。

IRCON Register (0xC0)

Bit	Field	Type	Initial	Description
7	ADCF	R/W	0	ADC 中断请求标志位。 0：无 ADC 中断请求； 1：请求 ADC 中断。
	others	R/W	0	参考中断章节。

17.9 ADC 电气特性

SYM.	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
I_{ADC}	运行功耗	Run 模式 @ 2.4V	-	200	-	uA
I_{PDN}	Power down 功耗	Stop 模式 @ 2.4V	-	0.1	-	μA
F_{SMP}	转换率(WR)	ADC 时钟 =62.5KHz, OSR=32768	-	1.9	-	Hz
		ADC 时钟 =250KHz, OSR=64	-	3.9	-	kHz
T_{ADCSTL}	ADC 设置时间	3*(1/WR), if WR=61Hz, T_{ADCSTL} = 3*16.4ms = 49.2ms		3		WR
V_{REF}	输入电压参考电压	外部 VREF 输入范围 (R+ - R-)	0.3		1.8	V
		内部 VREF 输入范围	0.54		1.8	V
V_R	ADC 参考信号绝对电压	GR=1, R+和 R- 绝对输入电压	0.4		AVDDR-1V	V
		GR=0, R+ 和 R-绝对输入电压	0.4		AVDDR-1V	V
V_{AI}	ADC 输入信号绝对电压	GX=1, AI 绝对输入电压	0.4		AVDDR-1V	V
		GX=0, AI 绝对输入电压	0		AVDDR-1V	V
V_x	PGIA 输出信号绝对电压	X+ and X-绝对输出电压(GX=0)	0.4		AVDDR-1	V
DNL	微分非线性	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 2		LSB
INL	积分非线性	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 4		LSB
NMC	No missing code	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 24-bits)		18		bit
NFB	Noise free bits	Gain:1, Vref:0.8V, OSR:32768, 输入短路		18.5		bit
		Gain=128, Vref=0.8V, OSR:32768, 输入短路		15.5		bit
ENOB	有效位数	Gain:1, Vref:0.8V, OSR:32768, 输入短路		21		bit
		Gain=128, Vref=0.8V, OSR:32768, 输入短路		18		bit
V_{AIN}	ADC 输入微分范围	ADC 输入信号, signal after PGIA application	0.3		1.44	V
T_{Drift}	ADC 温度漂移	AVDDR = 2.7V, PGIA x 16, T= 0 ~ 50 ℃		30		PPM/℃

SN8F5919 ADC Performance ENOB (Noise Free Bit) vs. Output Rate and Gain								
OSR	32768	16384	8192	4096	1024	256	128	64
WR	7.6Hz	15Hz	30.5Hz	61Hz	244Hz	976Hz	1.9KHz	3.9K
128x1	15.4	15	14.5	14	13.1	12.0	11.5	10.8
64x1	16.4	15.9	15.3	14.8	13.9	12.9	12.3	11.1
32x1	17.1	16.4	15.9	15.5	14.1	13.6	13.0	11.2
16x1	17.6	17.0	16.5	15.9	15.0	14.0	13.2	11.3
8x1	17.7	17.3	16.7	16.2	15.2	14.2	13.3	11.5
4x1	17.7	17.3	16.8	16.3	15.2	14.3	13.3	11.5
1x1	18.3	17.8	17.3	16.7	15.7	14.8	13.3	11.5

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

所有测试条件: ADC 250kHz, Input-Short, Vref=0.81V, Gain = PGIA x ADC, Collect 1024 ADC date.

(1). Noise Free 分辨率= Log2 (Full Scale Range / Peak-Peak Noise)
Where 满量程 = 2 x Vref / Gain (ex. Vref=0.81V, Gain=1x~128x)

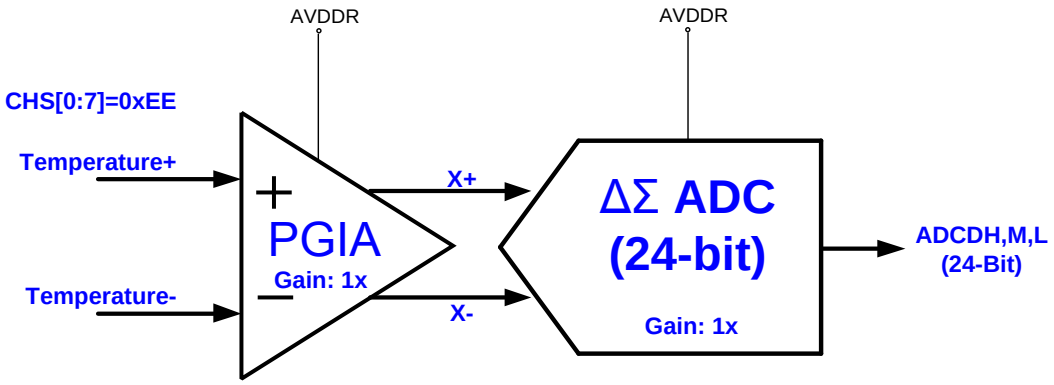
(2). 有效分辨率 = Log2 (Full Scale Range / RMS_Noise)

(3). RMS Noise = σ x LSB_Resolution where LSB_Resolution = Full Scale Range / 2^Bit, Bit=24
σ = standard deviation of 1024 ADC output data.

(4). Peak-Peak Noise = 6.6 x RMS Noise, or code variation range x LSB_Resolution where Code variation range = ADC counts max-min of 1024 data.

17.10 温度传感器 (TS)

在应用中，不同的环境温度会使传感器的特性也有所不同，为了得到不同的温度信息，SN8F5919 内置了一个温度传感器（TS）来测量工作环境温度。通过相对应的 PGIA 通道达到测量环境温度的目的。



□ □ **注1：当选择温度传感器时，PGIA的增益要选择为1x，否则会出错。**
□ □ **注2：这里的温度传感器只是一个参考数据而不是真实的室温，在精确的应用中，请选用外部的温度传感器。**

在 250C 的环境下，V（TS）典型值大约是 VTS Offset。如果温度上升 100C，V（TS）就会下降 18Mv（VTS=VTS Offset + 18mV）；相反，若温度下降 100C，V（TS）则会上升 18Mv（ VTS=VTS Offset – 18mV）。

例:

Temperature	(Ts+) - (Ts-)	ADC Vref	ADC 输出(16bit)
15℃	**V_{TS} Offset + 0.018V**	**0.8V(2/3VBG)**	**ADC offset + 695**
25℃	**V_{TS} Offset**	**0.8V(2/3VBG)**	**ADC offset**
35℃	**V_{TS} Offset - 0.018V**	**0.8V(2/3VBG)**	**ADC offset - 695**

通过 V（TS）ADC 输出，可以得到温度信息和系统补偿。

□ □ **注1：每颗单片机的V（TS）和温度曲线都是有差异的，建议在应用温度传感器时进行室内温度校准。**
□ □ **注2：温度传感器的典型温度参数是-1.8mV/C，每颗单片机不是完全相同的。（-69.5Cnt/℃）。**
□ □ **注3：使用内部温度传感器RVS [2:0]需要设置“RVS [01x] ”。**

SYM	DESCRIPTION	PARAMETER	MIN	TYP	MAX	UNIT
T_R	温度传感器范围	AVDDR=3V ,VDD=5V	-	-	-	℃
T_S	温度传感器灵敏度	AVDDR=3V ,VDD=5V		-1.7		mV/℃
ETS	温度传感器精度	1 个温度点 25℃ 校准	-10	-	+10	%
		2 个温度点校准	-1	-	+1	%

17.11 示例代码

下面的示例程序展示了如何在中断中执行 ADC。

```
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

unsigned short ADCValue;

void Init_VREG(void);
void Init_PGIA(void);
void Init_ADC(void);
void ADC_Enable_ISR(void);
void ADC_ISR(void);

void main(void)
{
    Init_VREG();        // Initial VREG
    Init_PGIA();       // Initial PGIA
    Init_ADC();        // Initial ADC
    ADC_Enable_ISR();   // ADC ISR Enable

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


void Init_VREG(void)
{
    VREG &= 0x00;       // Clear VREG
    VREG |= 0x80;       // Enable Band gap
    VREG &= 0x9F;       // AVDDR=2.7V

    VREG |= 0x08;       // Enable ACM
    VREG |= 0x10;       // Enable AVDDR
}


void Init_PGIA(void)
{
    CHS &= 0x00;        // Clear CHS
    CHS &= 0x0F;        // MUXP channel [AI1]
    CHS |= 0x01;       // MUXP channel [AI2]

    AMPM &= 0x00;       // Clear AMPM
    AMPM |= 0x20;       // ADCLK=7.8k@ADCCLK=250K
    AMPM |= 0x02;       // Enable PGIA Chopper
    AMPM |= 0x18;       // PGIA x 128

    AMPM |= 0x01;       // Enable PGIA
}
```

```
55
56 void Init_ADC(void)
57 {
58     ADCM1 &= 0x00;      // Clear ADCM1
59     ADCM1 |= 0x02;      // Enable ADC Chopper,
60                                 // ACHPEN always Enable
61     ADCM1 &= 0xF7;      // ADC Gain=1x
62     ADCM1 |= 0x50;      // ADC Vref = AVDDR*0.3
63
64     ADCM2 &= 0x00;      // Clear ADCM2
65     ADCM2 |= 0x00;      // ADC Clk=250kHz
66     ADCM2 |= 0x0E;      // ADC ODR=32768
67
68     ADCM2 &= 0xFE;      // Clear DRDY
69     ADCM1 |= 0x01;      // Enable ADC
70 }
71
72
73 void ADC_Enable_ISR(void)
74 {
75     ADCF = 0;           // Clear ADCF
76     IEN2 &= 0x00;       // Clear IEN2
77     IEN2 |= 0x10;       // ADC interrupt enable (EADC)
78     EAL = 1;            // Interrupt enable
79 }
80
81
82 void ADC_ISR(void) interrupt ISRAdc
83 {
84     // Get ADC value
85     if (ADCF) {
86      ADCF = 0;        // Clear ADC interrupt edge flag (ADCF)
87
88           /* Get16Bit ADC Data ( ADCDH,ADCDM ) */
89           ADCValue = ADCDH;
90           ADCValue = ADCValue<<8;
91           ADCValue = ADCValue+ADCDM;
92           _nop_();
93     }
94 }
95
96
```

下面的示例程序展示了如何执行 ADC 温度。

```
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

unsigned short ADCValue;

void Init_VREG(void);
void Init_PGIA(void);
void Init_ADC(void);

void main(void)
{
    Init_VREG();        // Initial VREG
    Init_PGIA();        // Initial PGIA
    Init_ADC();         // Initial ADC

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

    // Get ADC value
    if (ADCF) {
        ADCF = 0;   // Clear ADC interrupt edge flag (ADCF)

                /* Get16Bit ADC Data( ADCDH,ADCDM ) */
            ADCValue = ADCDH;
            ADCValue = ADCValue<<8;
            ADCValue = ADCValue+ADCDM;
            _nop_();
        }
  }
}

void Init_VREG(void)
{
    VREG &= 0x00;      // Clear VREG
    VREG |= 0x80;      // Enable Band gap
    VREG &= 0x9F;      // AVDDR=2.7V

    VREG |= 0x08;      // Enable ACM
    VREG |= 0x10;      // Enable AVDDR
}

void Init_PGIA(void)
{
    CHS &= 0x00;       // Clear CHS
    CHS |= 0xE0;       // MUXP channel [Temp S+]
    CHS |= 0x0E;       // MUXP channel [Temp S-]

    AMPM &= 0x00;      // Clear AMPM
    AMPM |= 0x20;      // ADCLK=7.8k@ADCCLK=250K
    AMPM |= 0x02;      // Enable PGIA Chopper
    AMPM &= 0xE3;      // PGIA x 1

    AMPM |= 0x01;      // Enable PGIA
}
```

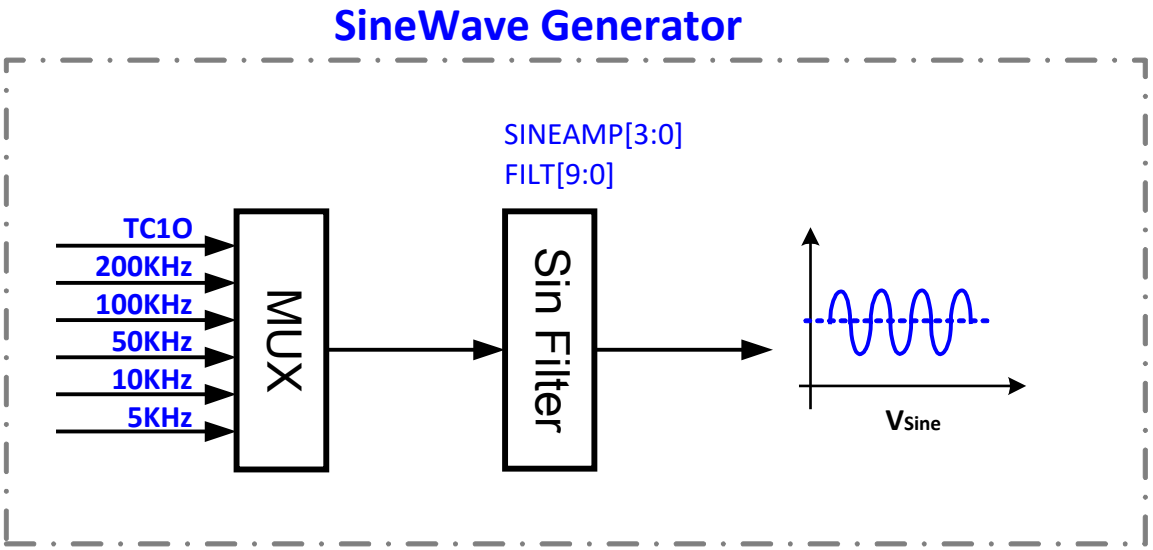
```
55 void Init_ADC(void)
56 {
57     ADCM1 &= 0x00;     // Clear ADCM1
58     ADCM1 |= 0x02;     // Enable ADC Chopper
59     ADCM1 &= 0xF7;     // ADC Gain=1x
60     ADCM1 |= 0x50;     // ADC Vref = AVDDR*0.3
61
62     ADCM2 &= 0x00;     // Clear ADCM2
63     ADCM2 |= 0x00;     // ADC Clk=250kHz
64     ADCM2 |= 0x0E;     // ADC ODR=32768
65
66     ADCM2 &= 0xFE;     // Clear DRDY
67     ADCM1 |= 0x01;     // Enable ADC
68 }
69
```

18 BIA

BIA 是生物电阻抗分析电路。该系统包括了正弦波波发生器和电流源发生器。正弦波发生器可以将时钟波传送到正弦波，由 SINEAMP[3:0]调整的正弦波幅值。生物电阻抗通过电流源发生器电路的测量。

18.1 正弦波发生器

- 来自数字部件的时钟源，频率范围 5KHz ~ 200KHz
- 由 SINEAMP[3:0] 寄存器调整正弦波幅值
- Sin过滤器需要为正弦波形设置FILT_H / FILT_L寄存器
  (参考: BIA 时钟 v.s 参数选择表)



18.2 BIA控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SINE	SINAMP3	SINAMP2	SINAMP1	SINAMP0	SW10EN	SW9EN	BIASEN	SINEN
FILT_L	FILTL 7	FILTL 6	FILTL 5	FILTL 4	FILTL 3	FILTL 2	FILTL 1	FILTL 0
FILT_H	-	-	-	-	-	-	FILTH 1	FILTH 0
OPM	BIA1-	BIA0-	BIA1+	BIA0+	BIAP2	BIAP1	BIA2EN	BIA1EN

SINE 寄存器 (0xAC)

Bit	Field	Type	Initial	Description
7..4	SINEAMP[3:0]	R/W	1000	正弦波振幅控制位
1	BIASEN	R/W	1	VBIAS 使能位 0： 禁止 1： 使能
0	SINEN	R/W	0	正弦波使能位 0： 禁止 1： 使能

Bit[7:4] SINEAMP[3:0]: 正弦波输出振幅控制位

SINEAMP[3:0]	Amplitude	SINEAMP[3:0]	Amplitude
0000	Level 0	1000	Level 8
0001	Level 1	1001	Level 9
0010	Level 2	1010	Level 10
0011	Level 3	1011	Level 11
0100	Level 4	1100	Level 12
0101	Level 5	1101	Level 13
0110	Level 6	1110	Level 14
0111	Level 7	1111	Level 15

FILT_H 寄存器 (0xAE)

Bit	Field	Type	Initial	Description 正弦波频率控制位
7..5	BCLK [2:0]	R/W	0	**000:** TCO **001:** 200K **010:** 100K **011:** 50K **100:** 10K **101:** 5K
1	FILT 9	R/W	0	0：禁止 1：使能
0	FILT 8	R/W	0	0：禁止 1：使能

正弦波频率 选择表

BCLK[2:0]	MUX output
000	TCO
001	200K
010	100K
011	50K
100	10K
101	5K

FILT_L (0xAD) 寄存器

Bit	Field	Type	Initial	Description
7	FILT 7	R/W	0	0 : 禁止 1 : 使能
6	FILT 6	R/W	0	0 : 禁止 1 : 使能
0 : 0	FILT 0 : FILT 0			

正弦频率 **v.s** 滤波器设置表格

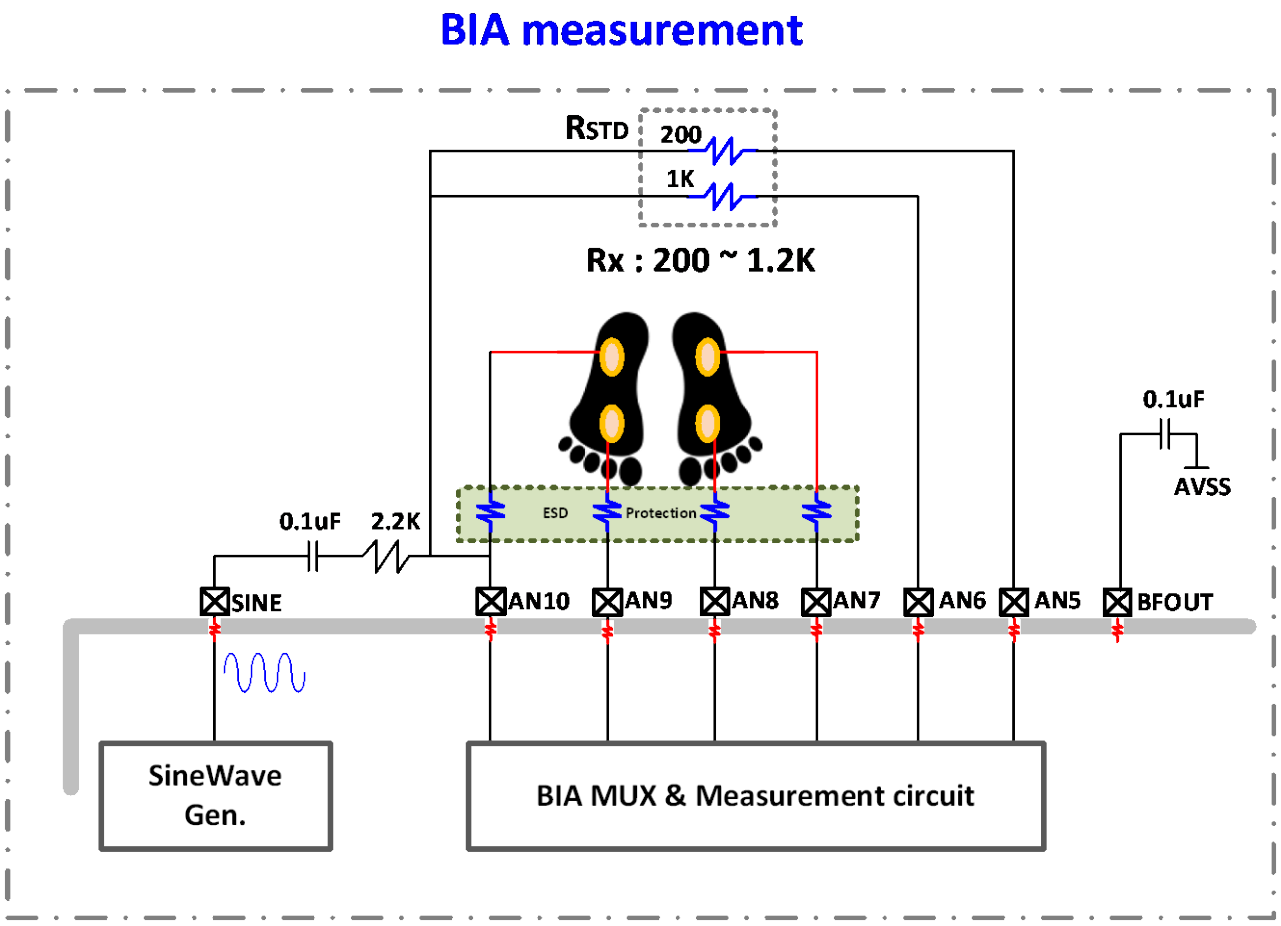
SINE Frequency	FILT_H[7:0]	FILT_L[7:0]
200K	0x20	0x0D
100K	0x40	0x1B
50K	0x60	0x37
10K	0x81	0x13
5K	0xA2	0x26

```
55
56 // ========== Sine-wave generator ==========
57 #include <intrins.h>    // for _nop_
58 #include < SN8F5919.h>
59
60 VREG   = 0x9C;     //  ACM ,AVDDR = 2.7v
61 SINE   = 0xE3;     //  Enable Sine-wave
62                    //  Enable VBIAS
63                                //  Amplitude : LeveL15
64 FILT_H = 0x60;     //  Sine wave = 50Khz.
65 FILT_L = 0x37;     //  Parameter = 220(50Khz Sine)
66 OPM    = 0x03;     //  Enable BIA1EN & BIA2EN.
```

18.3 电流源发生器

BIA电路

- 恒流通过Std. R200Ω/R1000Ω 和 人体双足
- BF_OUT 输出微分电压电路



18.4 OPM寄存器

Bit	Field	Type	Initial	Description
7..6	BIA-	R/W	00	BIA-[1:0] 通道选择 01：AN8 10：AN9 11：AN10
5..4	BIA+	R/W	00	BIA+[1:0] 通道选择 01：AN5 10： AN6 11：AN8
3..2	BIAP	R/W	00	BIAP [1:0], BIAP 当前路径选择 00：Current 禁止 01：200Ohm 环路 10：1KOhm 环路 11： 测量环路
1	BIA2EN	R/W	0	BIA2EN 使能位 0：禁止 1：使能
0	BIA1EN	R/W	0	BIA1EN 使能位 0：禁止 1：使能

```
// ========== Switch current path ==========
#include <intrins.h>    // for _nop_
#include < SN8F5919.h>

//======== Switch 200 Ohm Loop ==========
//OPM  = 0x97;     // AN5 + AN9
OPM    = 0xD7;  // AN5 + AN10
//=======================================

//======== Switch Measure X Loop =========
OPM = 0xBF;          // AN8 + AN9
//=======================================

//======== Switch 1000 Ohm Loop ===========
//OPM = 0xAB;  // AN6 + AN9
OPM   = 0xEB;  // AN6 + AN10
//=======================================
```

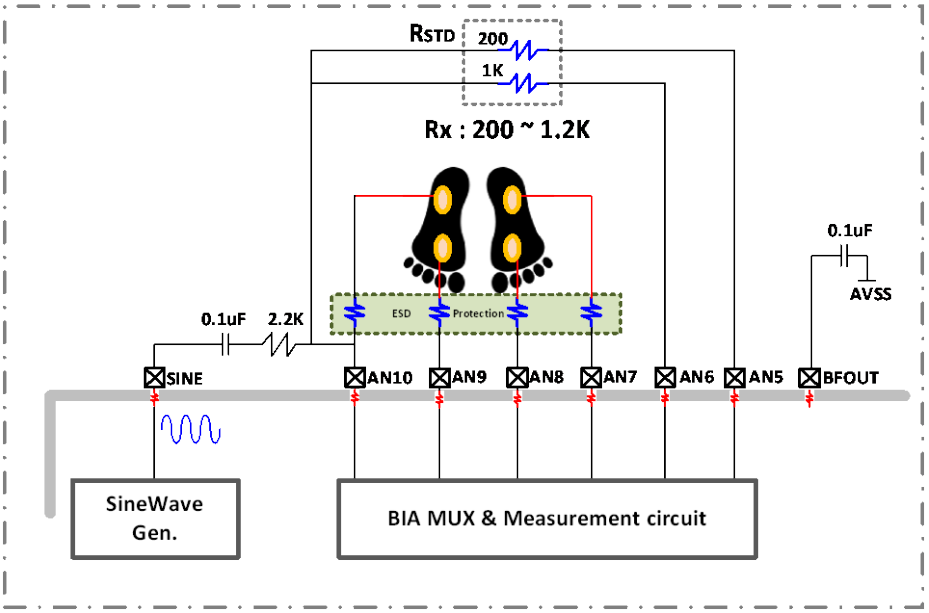
BIA AFE (Only Two Legs Mode)						
PARAMETER	***SYM.***	***Condition***	***MIN.***	***TYP.***	***MAX.***	***UNIT***
正弦波发生器	F_{req}	正弦波输出频率	5	50	200	KHz
	V_{PP}	正弦波电压 (Vpp) 与 16 阶调整	0.2		AVDDR-0.2	V
	I_{RMS}	正弦波电流源与 16 阶调整	100		600	uA
BIA 阻值范围	R_{BIA}		200		1200	Ω

18.5 BIA应用

SN8F5919 通过 BIA 功能生成正弦波形。正弦波振幅和频率由 SINE 寄存器和 FILT_H 寄存器控制。正弦波振幅调整范围从 1 级到 16 级。正弦波频率输出范围从 10 KHz 到 200 KHz。SN8F5919 SINE 是正弦波输出 pad。用户可以通过它来观察正弦输出波形。

SN8F5919 包含了三个阻抗测量路径。 第 1st 路径是 200Ω 电流环路，第 2nd 路径是 1KΩ 电流环路，第 3rd 路径是体阻抗测量回路。 200Ω 电流环路由外部 200 Ω 电阻建立。1KΩ 电流环路由外部 1KΩ 电阻建立。体阻抗测量回路是由脚在电流回路中建立的。

BIA measurement



BIA 功能设置步骤如下：

- 使能 BIA 电压。 （SINE 寄存器位[1:0]， OPM 寄存器位[1:0] ）
- 设置正弦波振幅 。（SINE 寄存器位[7:4]，默认 9 级 ）
- 设置 正弦波频率 。（FILT_H [7:5]，默认 50 KHz）
- 设置 BIA 测量路径。 （OPM [7:2]）

18.6 示例代码

使能 BIA 电压。

```
// ========== Enable BIA voltage ==========
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

//======== Enable BIA voltage ==========
SINE    |= 0x01;    // Enable Sin wave
SINE    |= 0x02;   // Enable Sin wave BIAS
OPM     |= 0x03;    // Enable BIA1EN & BIA2EN.
```

设置正弦波振幅。

```
// ========== Sine wave amplitude ==========
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

//======== Sine wave amplitude ==========
SINE   = 0xE3;        //  Amplitude : LeveL15
```

设置 正弦波频率。

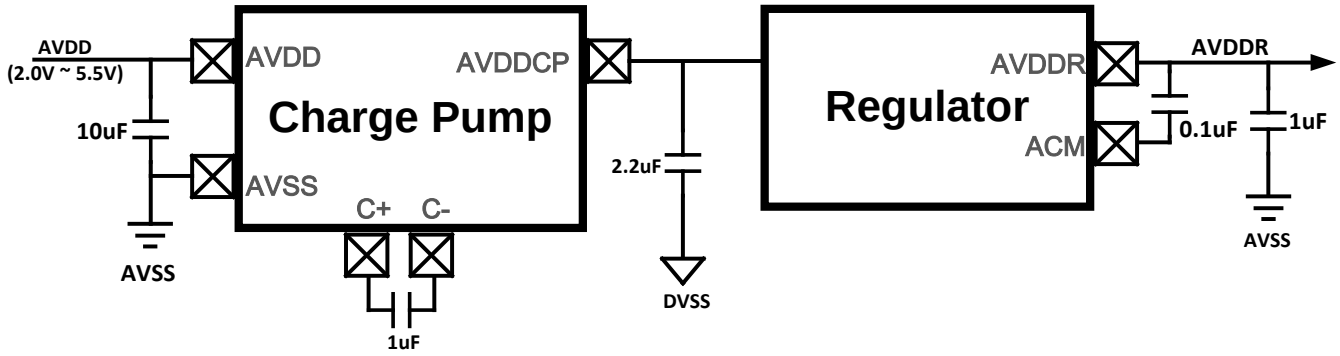
```
// ========== Sine wave frequency ==========
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

//======== Sine wave frequency ==========
FILT_H = 0x60;     // BCLK = 50Khz.
FILT_L = 0x37;     // 50Khz Parameter 0x37.
```

设置 BIA 测量路径。

```
// ========== Impedance measure path ==========
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

//======== Impedance measure path ==========
//======== Switch 200 Ohm Loop ==========
OPM    = 0xD7;  // AN5 + AN10
//=======================================

//======== Switch Measure X Loop =========
OPM    = 0xBF;  // AN8 + AN9
//=======================================

//======== Switch 1000 Ohm Loop ==========
OPM    = 0xEB;  // AN6 + AN10
//=======================================
```

19 升压稳压器（CPR）

SN8F5910 内置电压升压稳压器（CPR）为 ADC、PAIA、OP 和外部传感器提供稳定的电压 AVDDR。升压稳压器设计为 efficient type of boost 和 buck 模式，在 VDD 的输入范围为 2.0V~5.5V 时应用。可调节 AVDDR 输出电压 2.7V、3.0V、3.3V 和 3.6V，最大驱动电流为 7.5mA。另一种 regulator ACM 1.2V 用于 ADC common 电压，仅在 sink-type 下工作。



19.1 Charge Pump 寄存器

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

VREG Register (0xD2)

Bit	Field	Type	Initial	Description
7	BGREN	R/W	0	ADC Band Gap 使能位。 0：禁止；（系统在 STOP 模式下是必须禁止） 1：使能。（使能 AVDDR、ADC、PGIA、OPA 和 LBT 功能时必须先使能）
6..5	AVDDRS[1:0]	R/W	00	AVDDR regulator 输出电压控制位。 00：2.7V；　　10：3.3V； 01：3.0V；　　11：3.6V。
4	AVDDREN	R/W	0	AVDDR regulator 功能使能位。 0：禁止； 1：使能。 注：AVDDR 为模拟部分的电压源。
3	ACMEN	R/W	0	ACM regulator 功能使能位。 0：禁止； 1：使能。（使能 ADC 时必须使能）
2	CPCKS	R/W	0	Charge pump 时钟控制位。 0：500KHz；（请设置为 500KHz） 1：1MHz。
1..0	CPMOD[1:0]	R/W	00	Charge pump 模式控制位。 00：禁止； 01：Buck-Boost 模式； 10：保留； 11：2x Pump 模式。

下表显示 pump 的工作模式：

CPMOD[1:0]	Charge Pump 模式	Pump Behavior
00	禁止模式	VDD 输入范围 2.0V ~ 5.5V 禁止 Pump AVDDCP = VDD
01	Buck-Boost 模式	VDD 输入范围 2.0V ~ 5.5V - 使能 Pump - AVDDCP = AVDDR + 0.5V - AVDDCP =3.2V (禁止 AVDDR)
10	Reserved	-
11	2x Pump 模式	VDD 入范围 3.6V ~ 5.5V - 禁止 Pump (VDD >3.6V 时必须使能) - AVDDCP = VDD VDD 输入范围 2.0V ~ 3.4V - 使能 Pump - AVDDCP = 2x VDD (使能或禁止 AVDDR)
VDD : Pump 输入电压 AVDDCP: Pump 输出电压 **Pump 模拟功能的使能流程：** (1). 使能 BandGap (2). 使能 Pump (3). 延时 1ms 等待 pump 输出稳定 (4). 使能 AVDDR/ ACM (5). 延时 0.5ms (6). 使能 ADC, PGIA, OPA.。		

□ □ **注：当电池电压低于AVDDR+0.3 V时，PUMP模式可以设置buck-boost模式。**

19.2 Charge Pump Regulator 设置

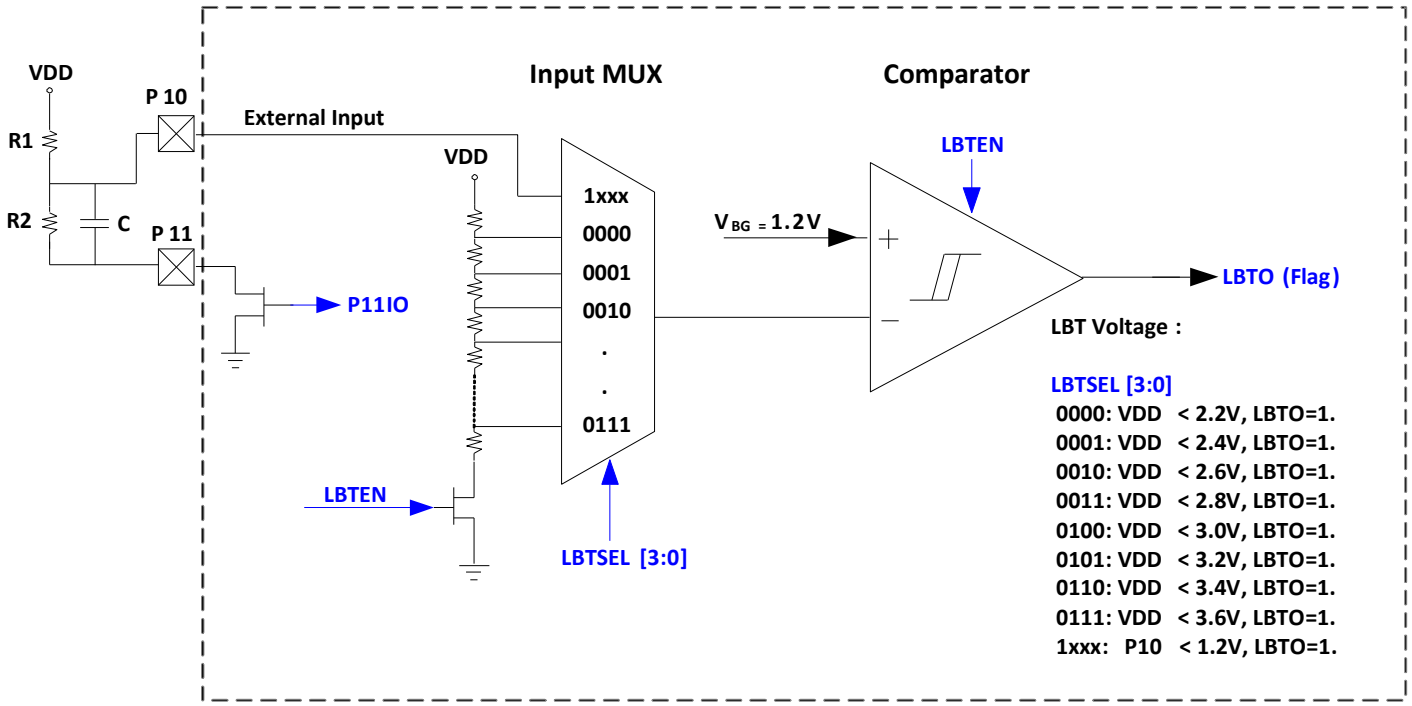
VDD 输入范围	Charge Pump	AVDDCP	AVDDR	Note
5.5V ~ 4.0V	禁止	* 无电容 CAVDDCP & CPump * AVDDCP 与 AVDD 短路	* 接 1uF 到 AVSS * 2.7V, 3V, 3.3V, 3.6V 有效	无双倍功耗
5.5V ~ 3.3V	禁止		* C 接 1uF 到 AVSS * 2.7V, 3V 有效	
5.5V ~ 3.0V	禁止		* 接 1uF 到 AVSS * 2.7V 有效	
5.5V ~ 2.0V	使能	* 接 CAVDDCP 2.2uF 到 DVSS (接 CPump 1uF 到 C+/C-)	* 接 1uF 阿斗 AVSS * 2.7V, 3V, 3.3V, 3.6V 有效	AVDDR 消耗双倍功耗
3.6V ~ 2.0V				

19.3 Charge Pump Regulator 电气特性

Charge Pump						
PARAMETER	***SYM.***	***Condition***	***MIN.***	***TYP.***	***MAX.***	***UNIT***
运行电压 (Charge Pump 输入)	V_{oper}	使能 PUMP，输入电压范围 (AVDD)	2.0	-	5.5	V
		禁止 PUMP，输入电压范围 (AVDD)	2.0	-	5.5	V
Charge pump 输出范围	V_{AVDDCP}	Charge Pump 关闭，AVDD 从 2V 到 5.5V AVDDCP = AVDD	2.0	-	5.5	V
		Busk-boost 模式，AVDD 从 2V 到 5.5V AVDDCP = 3.2V ~ 4.1V (AVDDR + 0.5V)	AVDDR + 0.5V			V
		2x Pump 模式，AVDD from 2V ~ 3.3V AVDDCP = 4V ~ 6.6V	2x AVDD			V
		2x Pump 模式，AVDD 从 3.3V ~ 5.5V AVDDCP = AVDD	AVDD			V
Charge pump 输出功能	I_{AVDDCP}	Charge Pump On 或 OFF	-	5	7.5	mA
Charge pump 内在功耗	I_{PUMP}	Busk-boost 模式， AVDD=3V		160		uA
ACM						
Analog common 电压	VACM	输出电压	1.1	1.2	1.3	V
		输出电压漂移 vs. 温度 Δ10°C	-	±0.1	-	%
		输出电压漂移 vs. AVDD (2V~5.5V)	-	±0.1	-	%
VACM driving capacity	ISRC	仅 ADC 使用	-	-	10	uA
VACM sinking capacity	ISNK	仅 ADC 使用	-	-	1	mA
AVDDR						
Regulator 输出电压 AVDDR	VAVDDR	AVDDR 输出范围 = 2.7V, 3.0V, 3.3V, 3.6V	2.7	-	3.6	V
		AVDDR 设置 2.7V	2.65	2.7	2.75	V
		AVDDR 设置 3.0V	2.95	3.0	3.05	V
		AVDDR 设置 3.3V	3.25	3.3	3.35	V
		AVDDR 设置 3.6V	3.54	3.6	3.66	V
		AVDDR Load regulation Δ5mA	V_{AVDDR} ± 0.05V			V
		输出电压漂移 vs. 温度 Δ10°C	-	±0.1	-	%
		输出电压漂移 vs. AVDD (2V~5.5V)	-	±0.1	-	%
AVDDR 拉/灌电流	IAVDDR		-	-	7.5	mA
Quiescent current	I_{QI}	Pump + AVDDR + ACM		200		uA

20 比较器

SN8F5910 内置比较器功能：正极输入电压大于负极输入电压时，比较器输出高电平（LBTO=1）；正极输入电压小于负极输入电压时，比较器输出低电平（LBTO=0）。比较器的正极电压始终为内部 1.2V，负极电压由外部 P10 和 VDD 电压分频控制。针对不同的应用，比较器提供标志显示（LBTO），类似于低电压检测。内部 VDD 分频后作为比较器的负极输入电压，共有 8 个电压点，由 LBTSEL[2:0]寄存器控制，其范围为 2.1V~3.5V，每次 0.2V 一个电压点。



20.1 比较器控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	P11IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTEN

LBTM 寄存器 (0xD7)

Bit	Field	Type	Initial	Description
6	P11IO	R/W	0	P1.1 配置位。 0：P1.1 为 GPIO 模式； 1：设置 P1.1 LBT 功能，P1.1 内部连接到地。
5	LBTSEL3	R/W	0	比较器负极输入原选择位。 0：P1.0 为 GPIO 模式； 1：外部 VDD 电压分频。
4..2	LBTSEL[2:0]	R/W	010	内部 VDD 电压分频控制位。 000：VDD < 2.2V，LBTO=1； 001：VDD < 2.4V，LBTO=1； 010：VDD < 2.6V，LBTO=1； 011：VDD < 2.8V，LBTO=1； 100：VDD < 3.0V，LBTO=1； 101：VDD < 3.2V，LBTO=1； 110：VDD < 3.4V，LBTO=1； 111：VDD < 3.6V，LBTO=1。
1	LBTO	R/W	0	比较器输出标志位。 0：比较器负极输入电压大于 1.2V； 1：比较器负极输入电压小于 1.2V。
0	LBTEN	R/W	00	比较器功能使能位。 0：禁止； 1：使能。

20.2 比较器电气特性

SYM.	*DESCRIPTION*	*PARAMETER*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
V_{ILBT}	内部低电压检测	条件：LBTSEL [3:0] = 0000，VLBT=2.2V	2.04	2.15	2.26	V
		条件：LBTSEL [3:0] = 0100, VLBT=3V	2.80	2.95	3.10	
		条件：LBTSEL [3:0] = 0111, VLBT=3.6V	3.40	3.55	3.70	
V_{ELBT}	外部低电压检测	条件：LBTSEL [3:0] = 1xxx, VLBT=P10 输入	1.1	1.2	1.3	
V_{HY}	比较器迟滞窗口			50	-	mV
I_{COMP}	功耗	AVDD = 3V		50	-	uA

20.3 示例代码

下面的示例程序展示了如何执行比较器功能。

```
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

unsigned char LBTOValue;

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

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

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

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


void Init_VREG(void)
{
    VREG &= 0x00;       // Clear VREG
    VREG |= 0x80;       // Enable Band gap
    VREG &= 0x9F;       // AVDDR=2.7V

    VREG |= 0x08;       // Enable ACM
    VREG |= 0x10;       // Enable AVDDR
}

void Init_LBTM(void)
{
    LBTM &= 0x00;       // Clear LBTM
    /* Low Battery Detect Voltage as 2.6v */
    LBTM |= 0x08;

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

20.4 BIA 内部 OP-Amp电气特性

BIA 内部运算放大器						
PARAMETER	**SYM.**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
工作电压	V_{oper}	电压来自 AVDDR	2.7	-	3.6	V
输入偏移电压	V_{OS}			±3		mV
工作电流	I_{oper}	Per OP-Amp	-	200	-	uA
模拟输入电压范围	V_{IN}	OP+ / OP-	0.05		AVDDR-0.1	V
模拟输出电压范围	V_{out}	OPOUT	0.05		AVDDR-0.1	V
输出短路电流	I_{OH}/I_{OL}	Unit Gain Buffer. Vo = 0V~3.2V, AVDDR=3.3V	-	±5	-	mA
缓冲模式 Ron	R_{BUF}	AVDDR 3.3V, VOUT=1.65V.	-	50	--	Ω
增益带宽	GB	RL=300KΩ, CL=50pF		1.4	-	MHz
转换速率	SR	10% to 90%	-	0.8	-	V/uS
启动时间	T_{ON}		-	20	-	uS

21 UART

UART（通用同步异步收发器）提供 1MHz 灵活的全双工传输模式。UART 共有 4 种操作模式：一种同步，3 种异步的。同步模式发送 8 位数据，没有起始/停止位；其它模式则分别支持 8 位和 9 位数据的发送，包含起始/停止位。由 P0.4 输出 UTX 信号，P0.5 接收 URX 信号。

21.1 UART 模式 0: 同步 8 位收发器

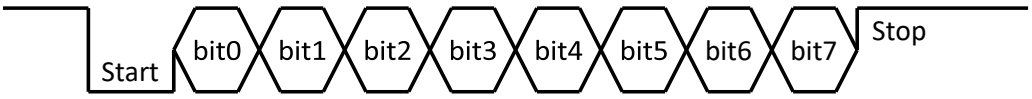
模式 0 下，UART 发送/接收 8 位长度的数据，没有起始/停止位。总线的首位为最低有效位 LSB，波特率固定为 Fcpu/12。通过设置 REN0 位为 1 和将 RI0 位清零开始接收；而通过写入数据到 S0TBUF 开始发送。

21.2 UART 模式 1: 异步 8 位收发器

模式 1 下，UART 操作为典型的格式协议：包括起始位，8 位数据位和停止位。波特率由 S0RELH/S0RELL 寄存器，或取决于 BD 寄存器的 TC1 的溢出周期控制。

通过写入数据到 S0BUF 寄存器开始发送，发送的首位是起始位（始终为 0），然后是 S0BUF 的数据（首先 LSB），发送完停止位（始终为 1）后，有通知/中断时自动将 TI0 位设置为 1。

若使能接收功能（REN0=1），UART 从主机同步起始位。把数据位的第一位当初最低有效位（LSB），接收完成后，S0BUR 会进行更新。



21.3 UART 模式 2/3: 异步 9 位收发器

模式 2 和模式 3 都包含起始位、9 位数据位和停止位。模式 2 的波特率只有 2 个选项：Fcpu/32 和 Fcpu/64；而模式 3 的波特率可通过 S0RELH/S0RELL 寄存器或 TC1 的溢出周期来控制。

第 9 位数据位（在 S0BUF 寄存器的最高位传输完毕之后传输）可通过写入 TB80 寄存器来传送，接收后在 RB80 中进行读取。也可以一直设置第 TB80 数据为 1，表示每次发送 2 个 STOP 信号，也可以将第 9bit 数据作为奇偶校验位。

第 9bit 数据也经常被使用来自定义为一对多的通信协议，当 SM20 寄存器设置为 1 时，仅仅收到的第 9bit 数据为 1 时，才会产生接收中断。利用这个功能就可以实现一主多从的通信协议。所有的从机都设置 SM20 为 1，主机端先发送需要通信的从机地址，同时设置第 9bit 为 1，这样所有的从机都会产生接收中断。从机判断接收到的地址是否为发给自己的。如果地址匹配，则对应的从机就将 SM20 清 0，地址不匹配的从机，保持 SM20 为 1。同时主机再发送后续的数据时，同时将第 9bit 数据设置为 0 再发出，那么其他从机就不会再产生接收中断。

21.4 波特率控制

UART 模式 0 的波特率是固定的，为 Fcpu/12；模式 2 有 2 个波特率选项，由 SMOCD 寄存器选择控制，其选项分别为 FCPU/32（SMOD=0）和 Fcpu/64（SMOD=1）。

UART 模式 1 和模式 3 的波特率由 S0RELH/S0RELL 寄存器（BD=1）或者 TC1 溢出周期（BD=0）产生。通过设置 SMOD 位可以将波特率增加 1 倍。

在模式 1 和模式 3 中若选择 S0RELH/S0RELL（BD=1），则波特率由下面公式产生：

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

普通 UART 波特率的设置建议如下表设置（Fcpu=32MHz）：

Baud Rate	SMOD	CD	S0RELH	S0RELL	Accuracy
4800 bps	1	0	0x03	0x30	-0.16 %
9600 bps	1	0	0x03	0x98	-0.16 %
19200 bps	1	0	0x03	0xCC	-0.16 %
38400 bps	1	0	0x03	0xE6	-0.16 %
56000 bps	1	0	0x03	0xEE	0.79 %
57600 bps	1	1	0x03	0xEC	0.79 %
115200 bps	1	1	0x03	0xF6	0.79 %
128k bps	1	1	0x03	0xF7	0.79 %
250k bps	1	0	0x03	0xFC	0 %
500k bps	1	0	0x03	0xFE	0 %
1M bps	1	0	0x03	0xFF	0 %

注: 波特率为 57600, 115200 和 128K bps 时 CD 位设置 为 1 。

在模式 1 和模式 3 中若选择 TC1 溢出周期（BD=0），则波特率由下面公式产生。TC1 必须为 16 位自动重装模式，这样才能产生周期性的溢出信号。

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

TC1 溢出周期普通 UART 波特率的设置建议如下表设置（Fcpu=32MHz）：

Baud Rate	SMOD	Timer Priod	TC1CH/ L	TC1RH/ L	Accuracy
4800 bps	1	13.021 us	0xFE5F	0xFE5F	-0.08 %
9600 bps	1	6.510 us	0xFF30	0xFF30	0.16 %
19200 bps	1	3.255 us	0xFF98	0xFF98	0.16 %
38400 bps	1	1.628 us	0xFFCC	0xFFCC	0.16 %
56000 bps	1	1.116 us	0xFFDC	0xFFDC	-0.80 %
57600 bps	1	1.085 us	0xFFDD	0xFFDD	-0.80 %
115200 bps	1	0.543 us	0xFFEF	0xFFEF	2.08 %
128k bps	1	0.488 us	0xFFF0	0xFFF0	-2.40 %
250k bps	1	0.250 us	0xFFF8	0xFFF8	0 %
500k bps	1	0.125 us	0xFFFC	0xFFFC	0 %

*** 注：系统时钟Fcpu最小设置值Fcpu< Timer Priod值，@PWSC[2：0]=001**
另 ：如果TC1定时器溢出期是6.510us,系统时钟Fcpu可以设置32m/64或32m/128

***例**: 模式 1 和模式 3 中，选择 TC1 溢出周期（BD=0），波特率 115200 如下面格式设置。（Fcpu=32MHz）

```
1     TC1CH = 0x0FF;     // baud rate 115200
2     TC1CL = 0x0EF;     // 0.543us Overflow
3     TC1RH = 0x0FF;     // auto-reload value
4     TC1RL = 0x0EF;     //
5     TC1M |= 0x0F0;     // TC1 ENABLE, TC1 timer clock= Fpcu/1
6
7     S0CON2 = 0x00;     // baud rate = TC1 Timer overflow period
8     PCON |= 0x80;      // SMOD=1,Fcpu*2 in mode1/mode3;
9     S0CON |=0x50;      // UART Enable,UART Mode 1
10
```

21.5 UART 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
S0CON2	BD	CD	-	-	-	-	-	-
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	-	ET2	ES0	ET1	EX1	ET0	EX0
P1OC	-	-	-	P05OC	P04OC	-	-	-
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P0	P07	P06	P05	P04	P03	P02	P01	P00

S0CON 寄存器(0x98)

Bit	Field	Type	Initial	Description
7..6	SM[0:1]	R/W	00	UART 模式选择位 00: 模式 0 01: 模式 1 10: 模式 2 11: 模式 3
5	SM20	R/W	0	多重处理器通讯控制位（模式 2、3）。 0：禁止； 1：使能。
4	REN0	R/W	0	UART 接收功能控制位。 0：禁止； 1：使能。
3	TB0	R/W	0	发送数据的第 9 位（模式 2、3）。
2	RB0	R/W	0	接收数据的第 9 位。
1	TI0	R/W	0	发送 UART 的中断标志。
0	RI0	R/W	0	接收 UART 的中断标志。

S0CON2 寄存器 (0xD8)

Bit	Field	Type	Initial	Description
7	BD	R/W	0	波特率产生器选择位（模式 1、3）。 0：TC1 溢出周期； 1：由寄存器 S0RELH/S0RELL 控制。
6	CD	R/W	0	UART 波特率 4800~128000 bps < 1%精度调整位 1：精确度功能使能。 0：精确度功能禁止。
5..0	Reserved	R	0x00	

S0BUF 寄存器(0x99)

Bit	Field	Type	Initial	Description
7..0	S0BUF	R/W	0x00	写入数据的操作触发 UART 通讯（首先 LSB），可在 package 结束时读取接收到的数据。

PCON 寄存器 (0x87)

Bit	Field	Type	Initial	Description
7	SMOD	R/W	0	UART 波特率控制位。 **（UART 模式 2）** 0：Fcpu/64 1：Fcpu/32 **（UART 模式 1、3）** 0：Fcpu*1 1：Fcpu*2
6..0				请参考其它章节。

S0RELH/S0RELL 寄存器 (S0RELH: 0xBA, S0RELL: 0xAA)

Bit	Field	Type	Initial	Description
15..10	Reserved	R	0x00	
9..0	S0REL[9:0]	R/W	0x00	S0RELH[1:0] & S0RELL[7:0]，UART 重装寄存器，用户产生 UART 波特率。

IEN0 寄存器 (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	使能中断，请参考中断章节。
4	ES0	R/W	0	使能 UART 中断
Else				请参考中断章节。

P1OC 寄存器(0xE4)

Bit	Field	Type	Initial	Description
4	P05OC	R/W	0	0：切换 P0.5（URX）为输入模式（要求）； ~~1：切换 P0.5（URX）为开漏模式*。~~
3	P04OC	R/W	0	0：切换 P0.4（UTX）为上拉模式； 1：切换 P0.4（UTX）为开漏模式。
Else				请参考其它章节。

*设置 P05OC 为高电平会导致 URX 不能接收数据。

P0M 寄存器(0xF9)

Bit	Field	Type	Initial	Description
5	P05M	R/W	0	0：设置 P0.5（URX）为输入模式（要求）； ~~1：设置 P0.5（URX）为输出模式*。~~
4	P04M	R/W	0	~~0：设置 P0.4（UTX）为输入模式*；~~ 1：设置 P0.4（UTX）为输出模式（要求）。
Else				请参考其它章节。

*URX 和 UTX 分别要求为输入模式和输出模式，以对应的接收和发送数据。

P0 寄存器 (0x80)

Bit	Field	Type	Initial	Description
5	P05	R/W	0	随时读取该位以监控总线状态。
4	P04	R/W	0	~~0：设置 P0.4（UTX）始终为低电平*；~~ 1：P0.4（UTX）输出 UART0 数据（要求）。
Else				请参考其它章节。

* 初始化 P04 时一定需要设置为高电平，因为 UART 模块仅驱动共有引脚的低电平。

21.6 示例代码

下面的示例程序代码显示了在中断中如何执行 UART 模式 1。

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

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

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

   // configure UART baud rate PCON
   = SYSUartSMODE1; S0CON2 =
   0x80;
   S0RELH = 0x03;
   S0RELL = 0xFE;

   // enable UART interrupt IEN0 |=
   SYSUartES0;

   // global interrupt enable
   EAL = 1;

   // 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;
   }
}
```

22 I2C

I2C 是串行通讯接口，在单片机与单片机，或单片机与其它外设之间进行数据传输。I2C 可作为主机或从机进行双向 IO 传输数据，SDA（串行数据输出）和 SCL（串行时钟输出）。

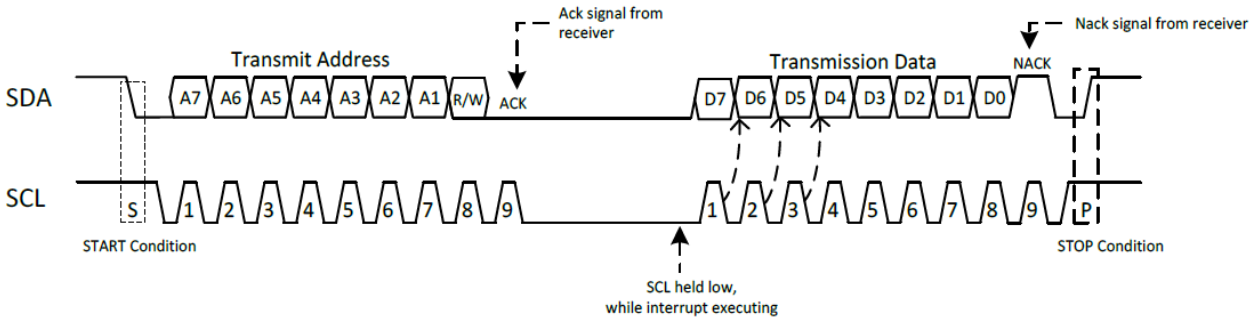
主机发送数据给从机时，叫做“WRITE”操作；从机发送数据给主机时，叫做“READ”操作。I2C 也支持多主机通讯，通过一种仲裁方式来决定哪个主机拥有控制总线以及传输数据的权限，从而保证数据传送的正确性。

22.1 I2C 协议

I2C 发送结构包括 START(S)开始 0 信号，8 位地址字节，一个或多个数据字节，以及 STOP (P)结束信号。开始信号由主机产生，以便对传输进行初始化。

发送的数据为最高有效位（MSB）优先。在地址字节中，高 7 位为地址位，低位为数据方向位（R/W）。R/W=0 时，表示该传输为“WRITE”操作；R/W=1 时，表示该传输为“READ”操作。

接收到每个字节后，接收器（主机或从机）必须发送一个 ACK 信号。若发送器没有接收到 ACK 信号，则会识别为 NACK 信号。在 WRITE 操作中，主机发送数据给从机，然后等待从机返回 ACK 信号。在 READ 操作中，从机发送数据给主机，然后等待主机返回 ACK 信号。最后，主机产生一个 STOP 信号来结束数据传输。



22.2 I2C 传输模式

I2C 可作为主机/从机，执行 8 位串行数据的发送/接收操作，因此，该模块共有 4 种操作模式：主机发送，主机接收，从机发送和从机接收。

22.3 主机发送模式

主机发送模式为主机发送数据给从机，串行时钟由 SCL 输出时，串行数据由 SDA 输出。主机通过发送 START 信号来开始数据发送。发送 START 信号后，开始发送从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 0 时使能主机发送。接下来，主机发送一个或多个数据字节给从机，每发送一个字节之后，主机要等待从机返回的 ACK 信号。最后，主机产生 STOP 信号来结束此次数据传输。

22.4 主机接收模式

主机接收模式为主机接收来自从机的数据，串行时钟由 SCL 输出时，串行数据由 SDA 输入。主机通过发送 START 信号来开始数据接收。发送 START 信号后，开始发送从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 1 时使能主机接收。接下来，主机接收来自从机的一个或多个数据字节，每接收一个字节之后，通过设置 I2CCON 寄存器的 AA 标志位来将 ACK 或 NACK 信号发送给从机。最后，主机产生 STOP 信号来结束此次数据传输。

22.5 从机发送模式

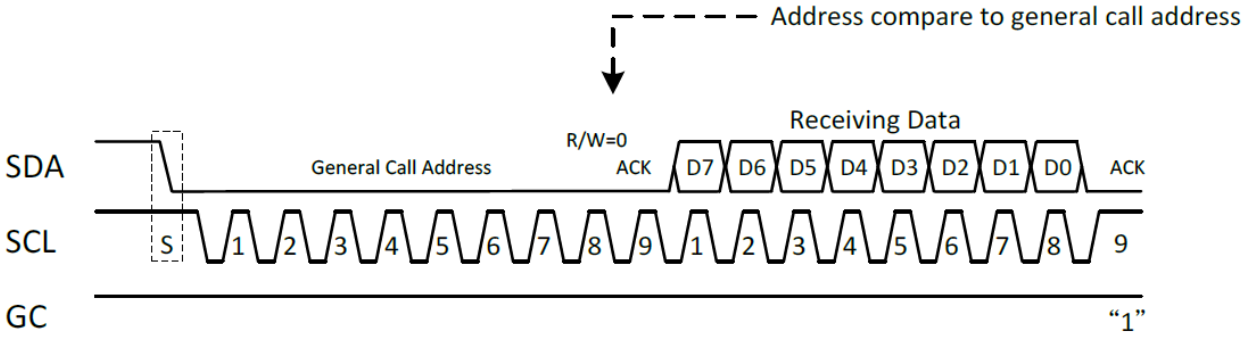
从机发送模式为从机发送数据给主机，串行时钟由 SCL 输入时，串行数据由 SDA 输出。接收到来自主机的 START 信号后开始数据发送。接收到 START 信号后，开始接收从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 1 时使能从机发送。若接收到的地址字节与 I2CADR 寄存器中的地址相匹配，从机将发送 ACK 信号；另外，若广播呼叫地址标志位设为 1（GC=1），接收到广播呼叫地址（00H）后，从机设备也会发送一个 ACK 信号。接下来，从机发送一个或多个数据字节给主机，每发送一个字节之后，从机要等待主机返回的 ACK 信号。最后，主机产生 STOP 信号来结束此次数据传输。

22.6 从机接收模式

从机接收模式为从机接收来自主机的数据，串行时钟和串行数据都由 SCL 和 SDA 输入。接收到来自主机的 START 信号后开始数据接收。接收到 START 信号后，开始接收从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 0 时使能从机接收。若接收到的地址字节与 I2CADR 寄存器中地址相匹配，从机将产生 ACK 信号；另外，若广播呼叫地址标志位设为 1（GC=1），接收到广播呼叫地址（0x00）后，从机设备也会产生一个 ACK 信号。接下来，从机接收一个或多个来自主机的数据字节，每接收一个字节之后，从机产生 ACK 或 NACK 信号，通过设置 I2CCON 寄存器的 AA 标志位来将 ACK 或 NACK 信号发送给主机。最后，主机产生 STOP 信号来结束此次数据传输。

22.7 广播呼叫地址

在 I2C 总线中，开始信号之后的第一个字节中的头 7 位为从机地址，只有该地址与从机地址相匹配时，从机才会响应 ACK 信号。只有广播呼叫地址是个例外，广播呼叫地址可以寻址所有的从机设备。当总线上出现广播呼叫地址时，所有设备应当响应一个 ACK 信号。广播呼叫地址是一个由全 0 组成的 7 位特殊地址。广播呼叫地址功能通过 GC 标志位来控制，设置 GC 标志位为 1 将使能广播呼叫地址，清 0 将关闭广播呼叫地址。当 GC=1 时，将会识别是否是广播呼叫地址，否则当 GC=0 时，将会忽略广播呼叫地址。



22.8 串行时钟发生器

在主机模式下，SCL 时钟速率发生器由 I2CCON 寄存器中的 CR[2:0]标志位来控制。

当 CR[2:0]=000~110 时，SCL 时钟速率来自内部时钟源。

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

当 CR[2:0]=111 时，SCL 时钟速率来自 TC1 定时器的溢出频率。

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

下表列出了不同设置下的时钟速率。

CR 2	CR 1	CR 0	I2C Prescale	Bit 频率 (kHz) 4MHz	8MHz	16MHz	32MHz
0	0	0	256	15.6	31.3	62.5	125
0	0	1	224	17.9	35.7	71.4	142.9
0	1	0	192	20.8	41.7	83.3	166.7
0	1	1	160	25	50	100	200
1	0	0	960	4.2	83.3	16.7	33.3
1	0	1	120	33.3	66.7	133.3	266.7
1	1	0	60	66.7	133.3	266.7	533.3
1	1	1	(TC1 定时器溢出 rate)/8				

*例：CR[2:0]=111 时，SCL 时钟 rate 来自 TC1 溢出 rate，时钟 rate 为 400KHz，如下所示：（Fcpu = 32M Hz）

```
TC1CH = 0x0FF;    // clock rate 400k Hz = 1/0.3125us/8
TC1CL = 0x0F6;    // 0.3125us Overflow
TC1RH = 0x0FF;    // auto-reload value
TC1RL = 0x0F6;    //
TC1M |= 0x0F0;    // TC1 ENABLE, TC1 timer clock= Fpcu/1

I2CCON |= 0x83;   // SPR[2:0]:111,SCL source from TC1 Timer
I2CCON |= 0x40;   // I2C enable (ENS1)
```

22.9 同步与仲裁

在多主机条件下，同一时间只有一个主机可在总线上传输数据。需要决定由哪个主机可控制总线以及实现传输。时钟同步与仲裁应用于配置多主机传输。时钟同步会在和其他设备同步 SCL 信号时运行。

当同一时间有两个主机要传输时，时钟同步将会在 SCL 由高电平转变成低电平的时候开始。如果主机 1 先将 SCL 切换为低电平，那么主机 1 会将 SCL 锁定在低电平状态直到将其切换成高电平状态。不过，如果有其他主机的 SCL 时序依然保持在低电平状态的话，那么主机 1 将 SCL 时序由低电平切换成高电平状态的过程将不会改变 SCL 的状态，SCL 时序将仍保持在低电平状态。也就是说，SCL 线被有最长低电平周期的主机保持在低电平状态。当所有设备的时钟周期都转换到高电平时，SCL 线将由低转成到高电平状态。在这其间，主机 1 将保持在由低电平到高电平的等待状态，之后再继续它的传输。经过时钟同步之后，所有设备的时钟和 SCL 时钟是一样的了。仲裁是用来决定哪个主机能够通过 SDA 信号来完成它的传输。两个主机可能会在同一时间发出开始信号以及传输数据，导致两者会互相影响。仲裁会强制使得其中一个主机失去总线的控制权。数据传输依然会继续，直到两个主机输出了不同的数据信号。如果其中一个主机传输了高电平状态，而另外一个主机传输了低电平状态，SDA 线会被拉低。输出高电平状态的主机将会检测到 SDA 线上的异常，并失去总线的控制权。输出低电平状态的主机成功获得总线的控制权，并继续它的传输，仲裁的过程中不会丢失数据。

22.10 系统管理总线（SMBus）扩展

可选的系统管理总线(SMBus) 协议的硬件支持三种类型的超时检测： (1) Tmext 超时检测：一个字节的累积持续时钟周期；(2)Tsext 超时检测：开始信号束信号之间的累积持续时钟周期；(3)超时检测：低电平时钟周期的测量。

通过 SMBSEL 和 SMBDST 寄存器来控制超时检测。SMBSEL 寄存器中的 SMBEXE 标志位是 SMBus 拓展功能的使能位。当 SMBEXE=1 时，使能 SMBus 拓展功能。否则，关闭 SMBus 拓展功能。超时类型和周期设置由 SMBTOP[2:0]和 SMBDST 寄存器来控制。SMBus 超时的周期由 Tmex，Tsext 以及 Tout 这个 16 位的暂存器来控制。计算公式如下：

$$\text{Tmext/Tsext/Tout} = \frac{\text{Timer Period(sec)} * F_{CPU}\text{(Hz)}}{1024}$$

Tmext 由 Tmext_L 和 Tmext_H 这两个 8 位寄存器组成，Tmext_L 为低字节，Tmext_H 为高字节。Tsext 由 Tsext_L 和 Tsext_H 这两个 8 位寄存器组成，Tsext_L 为低字节，Tsext_H 为高字节。Tout 由 Tout_L 和 Tout_H 这两个 8 位寄存器组成，Tout_L 为低字节，Tout_H 为高字节。

Type	超时周期	Fcpu=32MHz	
		十进制	十六进制
Tmext	5ms	157	9D
Tsext	25ms	782	30E
Tout	35ms	1094	446

通过设置 SMBTOP[2:0]来选择要设置的寄存器类型（如下表所示），将要配置的数据写到 SMBDST 寄存器中即可。

SMBTOP[2:0]	SMBDST	Descriptio
000	Tmext_L	选择 Tmext 寄存器的低字节
001	Tmext_H	选择 Tmext 寄存器的高字节
010	Tsext_L	选择 Tsext 寄存器的低字节
011	Tsext_H	选择 Tsext 寄存器的高字节
100	Tout_L	选择 Tout 寄存器的低字节
101	Tout_H	选择 Tout 寄存器的高字节

当 SMBus 拓展功能使能之后，I2CSTA 寄存器的低三位指示超时的相关信息，如下表所示：

I2CSTA	Description
XXXX X000	没有超时错误
XXXX XXX1	Tout 超时错误
XXXX XX1X	Tsext 超时错误
XXXX X1XX	Tmext 超时错误

22.11 I2C 寄存器

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

I2CDAT 寄存器 (0xDA)

Bit	Field	Type	Initial	Description
7:0	I2CDAT[7:0]	R/W	0x00	I2CDAT 寄存器存储的是要通过 I2C 总线发送出去的一字节数据，或者是刚从 I2C 总线上接收到的一字节数据。当它不在字节移位的过程中时，控制器可以读写这个 8 位的直接寻址特殊功能寄存器。由于 I2CDAT 寄存器没有缓存和双缓冲，因此当发生 I2C 中断时，用户只能读 I2CDAT 寄存器。

I2CADR 寄存器(0xDB)

Bit	Field	Type	Initial	Description
7:1	I2CADR[6:0]	R/W	0x00	I2C 从机地址。
0	GC	R/W	0	广播呼叫地址（0x00）。 0：忽略； 1：识别。

I2CCON 寄存器 (0xDC)

Bit	Field	Type	Initial	Description
7,1,0	CR[2:0]	R/W	0	I2C 时钟 rate。 000：Fcpu/256； 001：Fcpu/224； 010：Fcpu/192； 011：Fcpu/160； 100：Fcpu/960； 101：Fcpu/120； 110：Fcpu/60； 111：保留。
6	ENS1	R/W	0	I2C 使能位。 0：关闭； 1：使能。
5	STA	R/W	0	START 标志位。 0：没有发送 START 开始信号； 1：若总线空闲则发送 START 开始信号。
4	STO	R/W	0	STOP 标志位。 0：没有发送 STOP 结束信号； 1：若 I2C 总线为主机模式，则发送 STOP 结束信号。
3	SI	R/W	0	串行中断标志位。 当进入了 I2C 的 26 个状态当中的 25 个状态时，SI 标志位会被硬件置 1，只有当 I2C 状态寄存器为 F8h 时，SI 标志位才没有被置 1，表示没有可用的相关状态信息。SI 标志位必须由软件清零，必须通过写 0 到 SI 标志才可以清 SI 标志位，写入 1 到该位并不能更改 SI 的值。
2	AA	R/W	0	有效 ACK 标志位。 0：接收到 1 个字节后返回 NACK； 1：接收到 1 个字节后返回 ACK。

I2CSTA 寄存器(0xDD)

Bit	Field	Type	Initial	Description
7:3	I2CSTA[7:3]	R	11111	I2C 状态代码。
2..0	I2CSTA[2:0]	R	000	SMBus 状态代码。

I2C 状态代码和状态

模式	状态代码	I2C 的状态	应用软件响应					I2C 硬件引起的下步操作
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
主机发送器/接收器	08H	已发送 START 开始信号	载入 SLA+R	X	0	0	X	发送 SLA+R；接收 ACK。
	10H	已发送重复 START 开始信号	载入 SLA+R	X	0	0	X	发送 SLA+R；接收 ACK。
			载入 SLA+W					发送 SLA+W；I2C 切换到主机发送模式。
主机发送器	18H	已发送 SLA+W；并已接收到 ACK	载入数据字节	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	20H	已发送 SLA+W；并已接收 NACK	载入数据字节*	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	28H	已发送 I2CDAT 中的数据字节；并已接收 ACK	加载数据字节	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	30H	已发送 I2CDAT 中的数据字节；并已接收 NACK	加载数据字节*	0	0	0	X	发送数据字节；接收 ACK。
			无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
主机接收器	40H	已发送 SLA+R；并已接收 ACK	无动作	0	0	0	0	接收数据字节；返回 NACK。
			无动作	0	0	0	1	接收数据字节；返回 ACK。
	48H	已发送 SLA+R；并已接收 NACK	无动作	1	0	0	X	发送重复 START 开始信号。
			无动作	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			无动作	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。
	50H	已接收数据字节；并已返回 ACK	读数据字节	0	0	0	0	接收数据字节；返回 NACK。
			读数据字节	0	0	0	1	接收数据字节；返回 ACK。
	58H	接收数据字节；并已返回 NACK	读数据字节	1	0	0	X	发送重复 START 开始信号。
			读数据字节	0	1	0	X	发送 STOP 结束信号；复位 STO 标志位。
			读数据字节	1	1	0	X	发送 STOP 结束信号后接着再发送 START 开始信号；复位 STO 标志位。

模式	状态代码	I2C 的状态	应用软件响应					I2C 硬件引起的下步操作
			To/from I2CDAT	TO I2CCON				
				STA	STO	SI	AA	
从机接收器	60H	已接收到对应的 SLA+W ；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	68H	主机在发送 SLA+R/W 时仲裁丢失；并接收到了对应的 SLA+W，已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	70H	接收到了广播呼叫地址（00H）；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	78H	主机在发送 SLA+R/W 时仲裁丢失；并接收到广播呼叫地址（00H）；已返回 ACK	无动作	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	80H	从机地址已寻址成功；	读数据字节	X	0	0	0/1	接收数据字节；返回 NACK/ACK

		已接收 DATA；并返回 ACK						
	88H	从机地址已寻址成功；已接收 DATA；并返回 NACK	读数据字节	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			读数据字节	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			读数据字节	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			读数据字节	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	90H	广播呼叫地址已寻址成功；已接收 DATA；并返回 ACK	读数据字节	X	0	0	0/1	接收数据字节；返回 NACK/ACK
	98H	广播呼叫地址已寻址成功；已接收 DATA；并返回 NACK	读数据字节	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			读数据字节	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			读数据字节	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			读数据字节	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	A0H	当从机接收器或者从机发生器仍然被寻址时，接收到了 STOP 结束信号或者重复 START 开始信号	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
从机发送器	A8H	已接收对应的 SLA+R；已返回 ACK	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	B0H	主机在发送 SLA+R/W 时仲裁丢失了；已接收到了对应的 SLA+R，且已返回 ACK 信号。	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	B8H	数据字节已经发送，将接收到 ACK 信号。	载入数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			载入数据字节	X	0	0	1	发送一个字节的数据，并将接收到 ACK 信号。
	C0H	数据字节已经发送，并已接收到 NACK 信号	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。
			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
	C8H	最后一个字节的数据已经发送，并已接收到 ACK 信号。	无动作	0	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址。
			无动作	0	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到。

			无动作	1	0	0	0	切换至未寻址的从机模式；没识别到从机地址或广播呼叫地址；总线空闲之后将发送开始信号。
			无动作	1	0	0	1	切换至未寻址的从机模式；从机地址或广播呼叫地址将会被识别到；总线空闲之后将发送开始信号。
其他	F8H	没有可用的相关状态信息；SI=0	无动作	No action				等待或者进行当前的传输
	38H	仲裁丢失	无动作	0	0	0	X	I2C 总线将被释放；将发送开始信号。
			无动作	1	0	0	X	当总线空闲时（进入了主机模式）
	00H	在主机模式下或已选址的从机模式时，总线出错。	无动作	0	1	0	1	在主机模式或已寻址的从机模式时，只有内部硬件才会被影响。在所有情况下，总线已被释放，I2C 接口已切换至未寻址的从机模式，STO 标志位已复位。

“SLA”表示从机地址，“R”表示 R/W=1，“W”表示 R/W=0。
*表示接收到 NACK 之后不意味着通信的结束。

SMBSEL 寄存器 (0xDE)

Bit	Field	Type	Initial	Description
7	SMBEXE	R/W	0	SMBus 扩展功能。 0：禁止； 1：使能。
2..0	SMBSTP[2:0]	R/W	000	SMBus 超时寄存器。

SMBDST 寄存器 (0xDF)

Bit	Field	Type	Initial	Description
7..0	SMBD[7:0]	R/W	0x00	SMBDST 寄存器是用来提供一个读写 SMBus 超时寄存器的窗口。从 SMBDST 寄存器读写的数据实际上是对由 SMBSEL 指定的超时寄存器进行读写的数据。

IEN0 寄存器 (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	中断使能位。请参考中断章节。
Else				请参考其它章节。

IEN1 寄存器 (0xB8)

Bit	Field	Type	Initial	Description
0	EI2C	R/W	0	中断使能位。请参考中断章节。
Else				请参考其它章节。

P0M Register (0xF9)

Bit	Field	Type	Initial	Description
3	P03M	R/W	0	0：时钟 P0.3（SDA）为输入模式（要求）。 ~~1：时钟 P0.3（SDA）为输出入模式。*~~
2	P02M	R/W	0	0：时钟 P0.2（SCL）为输入模式（要求）。 ~~1：时钟 P0.2（SCL）为输出入模式。*~~
Else				请参考其它章节。

* 要求 P02M 和 P03M 分别设为输入模式。

22.12 示例代码

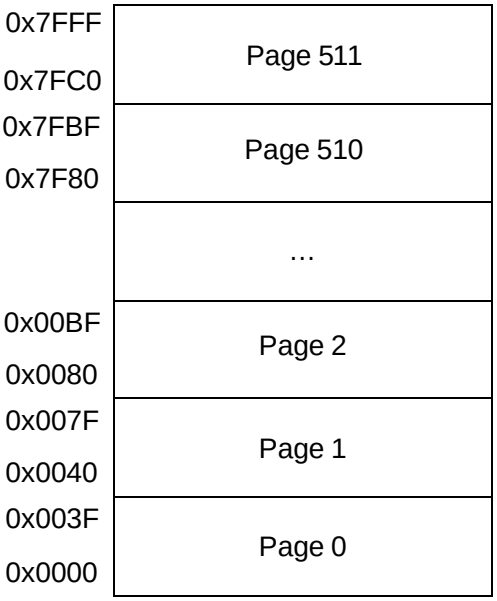
下面的示例代码程序演示了如何执行 I2C。

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


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

   I2CCON |= 0x40;       // I2C enable (ENS1)
   I2CCON |= 0x82;       // Clock rate bit : 110b ; 533.3KHz at 32MHz

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


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

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

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

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

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

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


```

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

23 In-System Program

SN8F5910 内置 32KB 程序存储（IROM），均分为 512 页（每页 64 字节）。In-System Program 就是使用软件随意的更改每页的数据，换句话说，就是存储数据到 Flash 存储器或者现场升级软件的一个通道。



程序存储器 (IROM)

23.1 页编程

由于程序存储器的每页都是 64 字节的长度，页编程就要求 64 字节 IRAM 作为其数据缓存器。

举例来说，假设程序存储器的第 510 页（IROM，0x7F80~0x7FBF）为计划升级区域，已经存入 IRAM 地址 0x60~0x9F 的内容，执行 In-System Program，只要写入开始 IROM 地址 0x7F80 到 EPROMH/EPROML 寄存器，然后指定缓存器开始地址 0x60 到 EPRAM 寄存器，随后写入 0xA5A 到 PECMD[11:0]寄存器，复制缓存器的数据到 IROM 的第 510 页。

通常而言，每页的内容都可通过 In-System Program 进行修改，但是第 1 页和最后一页（Page0 和 Page511）分别存储复位向量和上电控制管理的信息，不正确地执行页编程（如编程时切断电源）会导致上电错误或复位错误。

23.2 字节编程

SN8F5910 支持在所有的程序存储器区域执行字节编程功能，字节编程要求 1 个字节的 IRAM 在数据缓存器中。

举例说明：若程序存储器（IROM 0x7F82）第 510 页的低 3 个字节需要更新，其内容已经存入 IRAM 地址 0x60，执行 In-System Program，只写开始 IROM 页的直到 0x7F80 到 EPROMH/EPROML 寄存器中，写字节地址 0x02 到 PEBYTE 寄存器中，然后指定缓存器开始地址 0x60 到 EPRAM 寄存器，再然后写 0xA1E 到 PECMD[11:0]寄存器来复制缓存器的主机到 IROM 的第 510 页的第 3 个字节。

23.3 In-System Program寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BUCK	RAM7	RAM6	RAM5	RAM4	RAM3	RAM2	RAM1	RAM0
PEROMH	-	ROM14	ROM13	ROM12	ROM11	ROM10	ROM9	ROM8
PEROML	ROM7	ROM6	-	-	CMD11	CMD10	CMD9	CMD8
PECMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PEBYTE			PEBYTE5	PEBYTE4	PEBYTE	PEBYTE2	PEBYTE1	PEBYTE0

PECMD 寄存器 (0x94)

Bit	Field	Type	Initial	Description
7..0	PECMD[7:0]	W	-	0x5A：开始整页编程。 0x1E：开始字节编程。 *设置 PECMD[7：0]后，必须写 2 次 NOP 指令，请参考 23.4 范例 code。

* PECMD 禁止写其他值。

PEROML 寄存器 (0x95)

Bit	Field	Type	Initial	Description
7..6	PEROM[7:6]	R/W	00	编程页（IROM）的第一个地址（7^{th}– 6^{th}）。
5..4	Reserved	R	0	
3..0	PECMD[11:8]	W	-	0xA：使能 In-System Program 其它值：禁止 In-System Program *

*禁止 In-System Program 可避免误触发 ISP 功能。

PEROMH 寄存器 (0x96)

Bit	Field	Type	Initial	Description
7..0	PEROM[14:8]	R/W	0x00	编程页（IROM）的第一个地址（14^{th}– 8^{th} bit）。

BUCK 寄存器 (0x97)

Bit	Field	Type	Initial	Description
7..0	BUCK[7:0]	R/W	0x00	数据缓存器（IRAM）的第一个地址。

PEBYTE 寄存器 (0x9C)

Bit	Field	Type	Initial	Description
5..0	PEBYTE[5:0]	R/W	0x00	一页的字节地址

23.4 示例代码

下面的示例程序展示了如何执行 ISP。

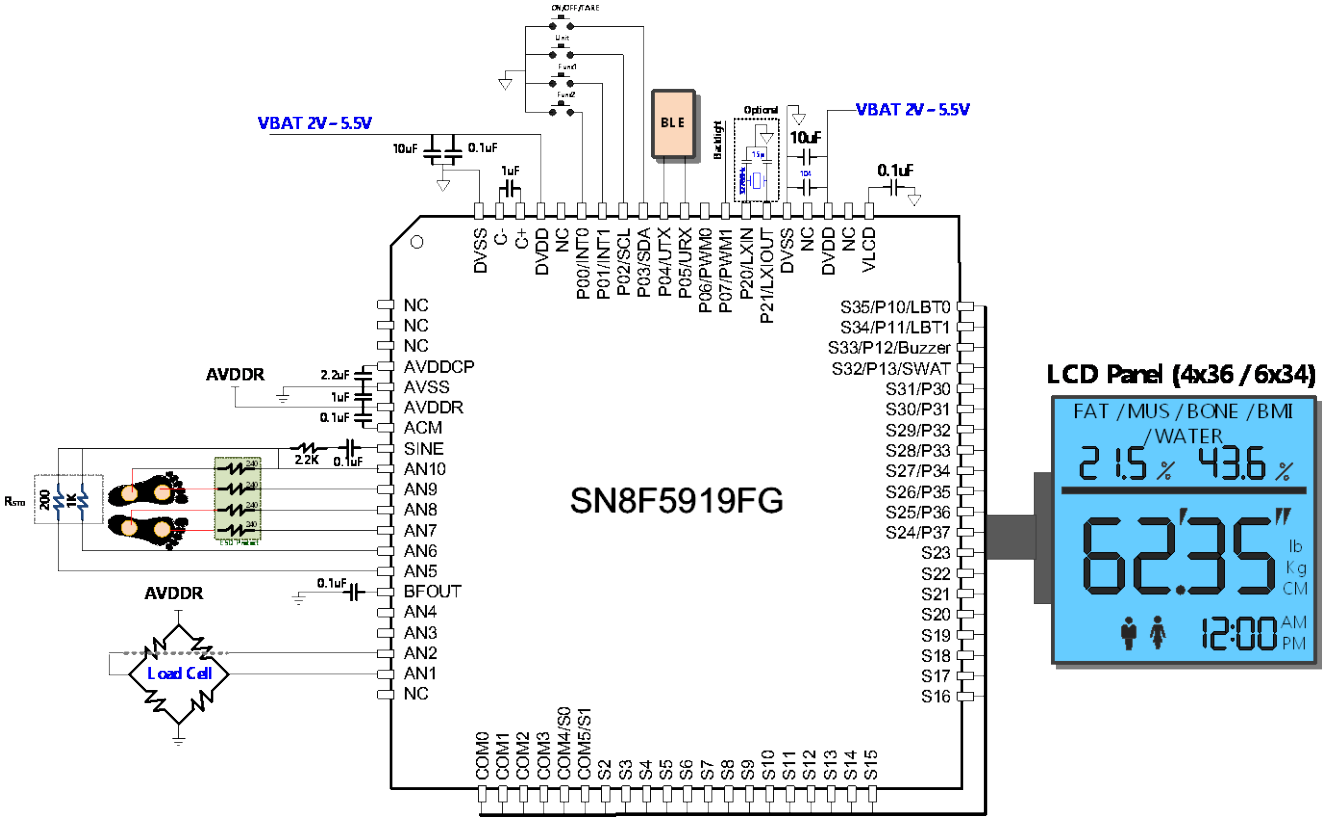
```
#include <intrins.h>    // for _nop_
#include <SN8F5919.h>

#include "GenericTypeDefs.h"

void ISPSetROMAddr(UINT16 u16addr);
void ISPSetRAMAddr(UINT8 u8addr);
void ISPWritePage(void);
void ISPWriteByte(void);
void ISPExecute_PAGE(void);
void ISPExecute_Byte(void);

void main(void)
{
   WDTR = 0x5A;          // clear watchdog if watchdog enable

   ISPExecute_PAGE();    // Program one page by ISP

   ISPExecute_Byte();    // Program one Byte by ISP

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

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

   UINT8 idata i =0;

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

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

   ISPSetRAMAddr(i);
   // step 3 : Set ROM start addr (Range is 0x0000~0x7FFF)
   ISPSetROMAddr(0x7880);

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

   //erase all USER ROM
   //ISPEraseAllROM();
}
```

```
55
56 void ISPExecute_Byte(void)
57 {
58         UINT8 idata u8data_byte[1] = {0};
59         UINT8 idata j =0;
60
61     // step 1 : Get data
62         u8data_byte[0] = 0XAA;     // write data for test
63     // step 2 : Set RAM addr of data
64         j = u8data_byte;                // get start addr
65     ISPSetRAMAddr(j);
66     // step 3 : Set ROM start addr (Range is 0x0000~0x7FFF)
67     ISPSetROMAddr(0x7880);
68
69     // step 4 : Set ROM Byte addr(Range is 0x00~0x3F)
70         PEBYTE =  0X20;
71
72     // step 5 : Progarm one byte (1 byte)
73     ISPWriteByte();
74
75     //erase all USER ROM
76     //ISPEraseAllROM();
77 }
78
79 void ISPSetROMAddr(UINT16 u16addr)
80 {
81     // set ROM addr
82     PEROML &= 0x0F;
83         //ISP Target Start ROM address Low byte
84     PEROML |= ((u16addr & 0x00F0));
85         //ISP Target Start ROM address High byte
86         PEROMH |= ((u16addr & 0xFF00)>>8);
87         _nop_();
88
89 }
90
91 void ISPSetRAMAddr(UINT8 u8addr)
92 {
93     // set RAM addr
94     PERAM = u8addr;
95 }
96
97 /*void ISPEraseAllROM(void)
98 {
99     // erase whole USER ROM
100    PECMD = 0x96;
101    PEROML |= 0x0A;
102 }*/
103
104
105
106
107
108
109
```

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

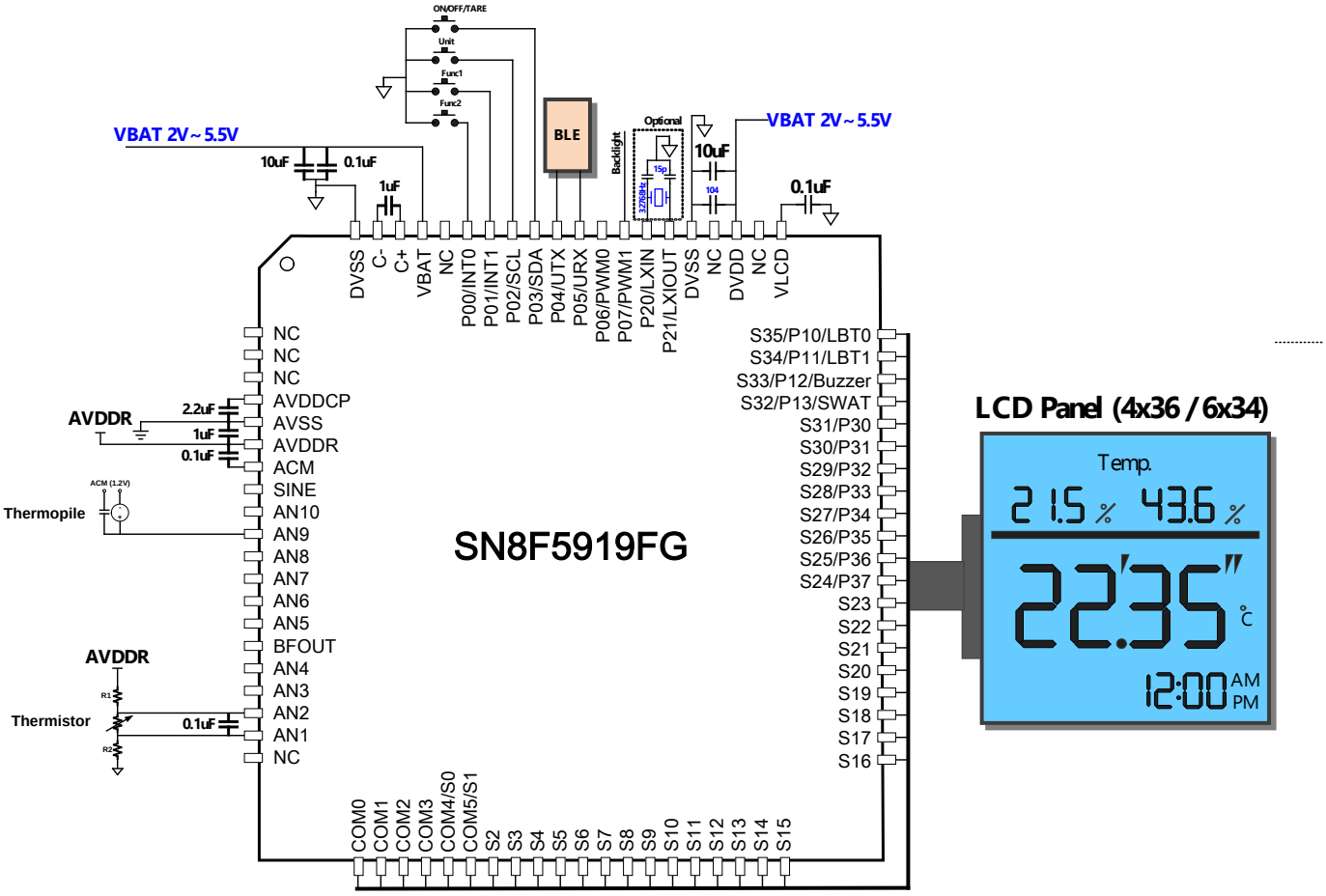
24 应用电路

24.1 BIA 应用电路



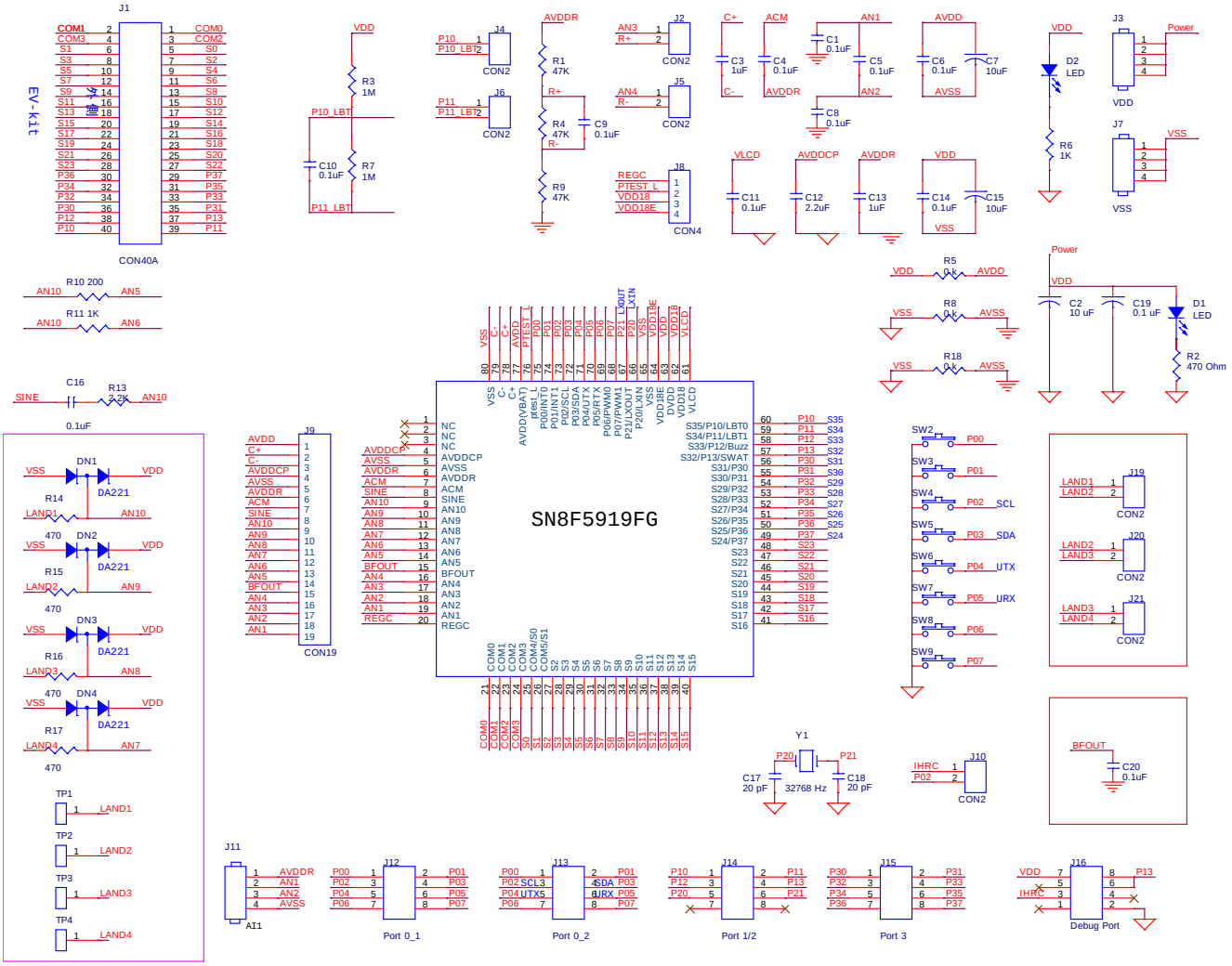
□ *□ 注: **DVDD/AVDD电容应该尽可能的接近IC的引脚。**

24.2 温度计应用电路



□ *□ 注: **DVDD/AVDD** 电容应该尽可能的接近 **IC** 的引脚。

24.3 Starter-Kit 电路



25 电气特性

25.1 极限参数

VDD 到 VSS 的电压..... - 0.3V to 6.0V
任意引脚到 VSS 电压.....- 0.3V to VDD+0.3V
运行环境温度.....-40°C to 85°C
存储环境温度..... -40°C to 125°C

25.2 系统操作特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	运行电压	fcpu = 1MHz	2.0		5.5	V
V_{DR}	RAM 数据保留电压		1.5			V
V_{POR}	VDD 上升率*		0.05			V/ms
I_{DD1}	Normal 模式功耗	VDD = 3V ~ 5V, fcpu = 0.25MHz		2.8		mA
		VDD = 3V ~ 5V, fcpu = 1MHz		3.0		mA
		VDD = 3V ~ 5V, fcpu = 2MHz		3.3		mA
		VDD = 3V ~ 5V, fcpu = 4MHz		3.6		mA
		VDD = 3V ~ 5V, fcpu = 8MHz		4.3		mA
		VDD = 3V ~ 5V, fcpu = 16MHz		5.7		mA
		VDD = 3V ~ 5V, fcpu = 32MHz		8.4		mA
I_{DD2}	STOP 模式功耗	VDD = 3V		2.0		μA
		VDD = 5V		2.5	5.0	μA
		VDD = 3V, RTC 使能 (32768Hz Crystal)		3.0		μA
I_{DD3}	IDLE 模式功耗 (fcpu = 0.25MHz)	VDD = 3V		0.56		mA
		VDD = 5V		0.57		mA
f_{IHRC}	内部高速时钟发生器	VDD = 2.0V ~ 5.5V, 温度 25°C	-1%	32	+1%	MHz
		VDD = 2.0V to 5.5V, 温度-40°C to 85°C	-2%	32	+2%	MHz
f_{ILRC}	内部低速时钟发生器	VDD = 5.0V @ 25°C	12	16	24	KHz
V_{LVD16}	LVD16 检测电压	25°C	1.58	1.65	1.72	V
		-40°C to 85°C	1.60	1.65	1.80	V
V_{LVD12}	LVD12 检测电压	25°C	1.20	1.25	1.32	V
		-40°C to 85°C		1.25		V
ESD	人体模式 (HBM)			4	4.5	KV
	机械模式(MM)			250	300	V
LU	Latch Up	VDD < 5.5 V	-400		+400	mA
EFT	电快速瞬变脉冲群 (Fcpu = 1mips)	V_{L+}			4500	V
		V_{L-}			4500	
		V_{N+}			4500	
		V_{N-}			4500	
		V_{LN+}			4500	
		V_{LN-}			4500	

* 未经验证的设计参考。环境温度是 25°C。

25.3 GPIO 特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
V_{IL}	低电平输入电压		VSS		0.3VS	V
V_{IH}	高电平输入电压		0.7VDD		VDD	V
I_{LEKG}	I/O 口输入漏电流	VIN = VDD			2	μA
R_{UP}	上拉电阻	VDD = 3V	50	100	150	kΩ
		VDD = 5V	25	50	75	kΩ
I_{OH}	I/O 输出灌电流	VDD = 3V, VO = VDD-0.5V	8	10		mA
I_{OL}	I/O 拉电流	VDD = 3V, VO = VSS+0.5V	9	10		mA

25.4 ADC 特性

SYM.	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
I_{ADC}	操作电流	Run 模式 @ 2.4V	-	200	-	uA
I_{PDN}	睡眠电流	Stop 模式 @ 2.4V	-	0.1	-	μA
F_{SMP}	转换率(WR)	ADC 时钟=62.5KHz, OSR=32768	-	1.9	-	Hz
		ADC 时钟=500KHz, OSR=64	-	7.8	-	kHz
T_{ADCSTL}	ADC 设置时间	3*(1/WR), if WR=61Hz, T_{ADCSTL} = 3*16.4ms = 49.2ms	3			WR
V_{REF}	参考电压输入电压	外部 VREF 输入范围 (R+ - R-)	0.3		1.8	V
		Internal VREF Input Range.	0.3		1.8	V
V_R	ADC 参考信号绝对电压	GR=1, R+ 和 R- 绝对输入电压	0.4		AVDDR-1V	V
		GR=0, R+ 和 R- 绝对输入电压	0.4		AVDDR-1V	V
V_{AI}	ADC 输入信号绝对电压	GX=1, AI 绝对输入电压	0.3		AVDDR-1V	V
		GX=0, AI 绝对输入电压	0		AVDDR-1V	V
V_x	PGIA 输出信号绝对电压	X+ and X-绝对输入电压	0.3		AVDDR-1V	
	PGIA 增益比*	VDD=5V , PGIA=x128	-3		+3	%
DNL	微分非线性	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 2		LSB
INL	积分非线性	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 4		LSB
NMC	无丢码	ADC 范围 ± 131072 x 0.9. (0.9 x Vref , 18-bits)		18		bit
NFB	Noise free bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		18.5		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		15.5		bit
ENOB	有效位数	Gain:1, Vref:0.8V, OSR:32768, Input-short		21		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		18		bit
V_{AIN}	ADC 输入微分范围	ADC 输入信号， signal after PGIA application	0.3		1.44	V
T_{Drift}	ADC 温度漂移	AVDDR = 2.7V		30		PPM /°C

25.5 Charge Pump Regulator 特性

Charge Pump						
PARAMETER	**SYM.**	**Condition**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
操作电压 (Charge Pump 输入)	V_{oper}	使能 Pump，输入电压范围 (AVDD)	2.0	-	5.5	V
		禁止 Pump，输入电压范围 (AVDD)	2.0	-	5.5	V
Charge pump 输出范围	V_{AVDDCP}	Charge Pump 关, AVDD 从 2V 到 5.5V AVDDCP = AVDD	2.0	-	5.5	V
		Busk-boost 模式, AVDD 从 2V 到 5.5V AVDDCP = 3.2V ~ 4.1V (AVDDR + 0.5V)	AVDDR + 0.5V			V
		2x Pump Mode, AVDD from 2V ~ 3.3V AVDDCP = 4V ~ 6.6V	2x AVDD			V
		2x Pump Mode, AVDD from 3.3V ~ 5.5V AVDDCP = AVDD.	AVDD			V
Charge pump 输出功耗	I_{AVDDCP}	Charge Pump 打开或关闭	-	5	7.5	mA
Charge pump 固有功耗	I_{PUMP}	Busk-boost 模式, AVDD=3V		160		uA
ACM						
模拟通用电压	V_{ACM}	输出电压	1.1	1.2	1.3	V
		输出电压偏移. 温度 Δ10°C	-	±0.1	-	%
		输出电压偏移. AVDD (2V~5.5V)	-	±0.1	-	%
VACM 驱动能力	I_{SRC}	仅为 ADC 使用	-	-	10	uA
VACM 反向能力	I_{SNK}	仅为 ADC 使用	-	-	1	mA
AVDDR						
Regulator 输出电压 AVDDR	V_{AVDDR}	AVDDR 输出范围 = 2.7V, 3.0V, 3.3V, 3.6V	2.7	-	3.6	V
		AVDDR 设置 2.7V	2.65	2.7	2.75	V
		AVDDR 设置 3.0V	2.95	3.0	3.05	V
		AVDDR 设置 3.3V	3.25	3.3	3.35	V
		AVDDR 设置 3.6V	3.54	3.6	3.66	V
		AVDDR Load regulation Δ5mA	V_{AVDDR} ± 0.05V			V
		输出电压偏移. 温度 Δ10°C	-	±0.1	-	%
		输出电压偏移. AVDD (2V~5.5V)	-	±0.1	-	%
AVDDR 驱动/反向电流	I_{AVDDR}		-	-	7.5	mA
静态电流	I_{QI}	Pump + AVDDR + ACM		200		uA

25.6 BIA 内部 OP-Amp 电气特性

Operation Amplifier						
SYM.	**PARAMETER**	**DESCRIPTION**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
V_{oper}	操作电压	电源来自 AVDDR	2.7	-	3.6	V
V_{OS}	输入补偿电压			±3		mV
I_{oper}	工作电流	Per OP-Amp	-	200	-	uA
V_{IN}	模拟输入电压范围	OP+ / OP-	0.05		AVDDR-0.1	V
V_{out}	模拟输出电压范围	OPOUT	0.05		AVDDR-0.1	V
I_{OH}/I_{OL}	输出短路电流	Unit Gain Buffer. Vo = 0V~3.2V, AVDDR=3.3V	-	±5	-	mA
R_{BUF}	缓冲模式开关	AVDDR 3.3V, VOUT=1.65V.	-	50	--	Ω
GB	增益带宽	RL=300KΩ, CL=50pF		1.4	-	MHz
SR	转换率	10% to 90%	-	0.8	-	V/uS
T_{ON}	启动时间		-	20	-	uS

25.7 比较器特性

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
V_{ILBT}	内部低电压检测	条件: LBTSEL [3:0] = 0000, VLBT=2.2V	2.04	2.15	2.26	V
		条件: LBTSEL [3:0] = 0100, VLBT=3V	2.80	2.95	3.10	
		条件: LBTSEL [3:0] = 0111, VLBT=3.6V	3.40	3.55	3.70	
V_{ELBT}	外部低电压检测	条件:LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.1	1.2	1.3	
V_{HY}	比较器迟滞窗口			50	-	mV
I_{COMP}	电流消耗	AVDD = 3V		50	-	uA

25.8 LCD 特性

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
I_{CLCD}	C-Type LCD 工作电流	VDD:3~5V, 无面板低功耗模式 (BGM=0,DUTY[1:0]=00)		9		uA
		VDD:3~5V, 1/3 bias, 无面板低功耗模式 (BGM=0,DUTY[1:0]=01)		10		
		VDD:3~5V, 1/3 bias, 无面板低功耗模式 (BGM=0,DUTY[1:0]=10)		12		
		VDD:3~5V, 1/3 bias, 无面板低功耗模式 (BGM=0,DUTY[1:0]=11)		17		
		VDD:3~5V, 1/3 bias, 无面板低功耗模式 (BGM=1,DUTY[1:0]= xx)		127		
V_{LCD}	C-Type VLCD 工作电流	VDD: 2.0 ~ 5.5V, VLCD set 3V (VCP [2:0]=010)	2.8	3.0	3.2	V
		VDD: 3.0 ~ 5.5V, VLCD set 4.5V (VCP [2:0]=101)	3.4	4.5	4.7	

25.9 Flash存储器特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
Vdd	供电电压		2.0		5.5	V
Ten	持续时间	25°C		*100K		cycle
Iwrt	写功耗	25°C		3	4	mA
Twrt	写时间	Write 1 page=64 bytes, 25°C		5	8	ms

* 未验证的设计参考。

25.10 特殊条件或环境中使用建议

条件 / 环境	建议
条件 (1) 当 MCU I/O 连接到其他电路, 它有负瞬时电压出现在 I/O 上, 峰值电压 -3V, MCU 可能发生延迟现象。	连接 10-Ohm 电阻在电源输入MCU的DVDD pin之前, 这将解决在I/O中出现的异常瞬态电压出现的延迟。
条件 (2) 当 MCU 电源上有高干扰且 MCU 功能必须好好工作时. (高 EFT 干扰 MCU)	MCU 必须设置 “Noise filter Enable” 在高EFT环境中增强抗干扰能力。

26 指令集

本章对 SN8F5910 的综合汇编指令进行分类，总共包括 5 类：算术运算、逻辑运算、数据传送运算、布尔操作和程序分支指令，这些指令全部都与标准 8051 兼容。

26.1 符号说明

符号	功能描述
Rn	工作寄存器 R0 - R7
direct	直接地址：128 个内部 RAM 地址之一或特殊功能寄存器地址
@Ri	通过寄存器 R0 或者 R1 来间接寻址内部或外部 RAM 地址
#data	8 位常量（立即操作数）
#data16	16 位常量（立即操作数）
bit	位于内部 RAM 中的 128 个位变量之一，或者是可以位操作的特殊功能寄存器当中的任何标志位
addr16	LCALL 和 LJMP 的目标地址，可以是 64KB 程序存储空间当中的任意地址
addr11	ACALL 和 AJMP 的目标地址，与下一条指令的第一个字节在同一个 2KB 区内（即 16 位地址中的高 5 位地址相同
rel	SJMP 和所有条件跳转指令中的一个 8 位偏移字节，相对于下一条指令的第一个字节，可偏移的范围为+127/-128 字节
A	累加器

26.2 运算指令

指令	功能描述
ADD A, Rn	将累加器与寄存器的内容相加，结果存回累加器
ADD A, direct	将累加器与直接地址的内容相加，结果存回累加器
ADD A, @Ri	将累加器与间接地址的内容相加，结果存回累加器
ADD A, #data	将累加器与常量相加，结果存回累加器
ADDC A, Rn	将累加器与寄存器的内容及进位 C 相加，结果存回累加器
ADDC A, direct	将累加器与直接地址的内容及进位 C 相加，结果存回累加器
ADDC A, @Ri	将累加器与间接地址的内容及进位 C 相加，结果存回累加器
ADDC A, #data	将累加器与常量及进位 C 相加，结果存回累加器
SUBB A, Rn	将累加器的值减去寄存器的值减借位 C，结果存回累加器
SUBB A, direct	将累加器的值减去直接地址的值减借位 C，结果存回累加器
SUBB A, @Ri	将累加器的值减去间接地址的值减借位 C，结果存回累加器
SUBB A, #data	将累加器的值减常量减借位 C，结果存回累加器
INC A	将累加器的值加 1
INC Rn	将寄存器的值加 1
INC direct	将直接地址的内容加 1
INC @Ri	将间接地址的内容加 1
INC DPTR	将数据指针寄存器的值加 1
DEC A	将累加器的值减 1
DEC Rn	将寄存器的值减 1
DEC direct	将直接地址的内容减 1
DEC @Ri	将间接地址的内容减 1
MUL AB	将累加器的值与 B 寄存器的值相乘，乘积的低字节存回累加器，高字节存回 B 寄存器
DIV	将累加器的值除以 B 寄存器的值，结果的商存回累加器，余数存回 B 寄存器
DA A	将累加器的值作十进制调整

26.3 逻辑指令

指令	功能描述
ANL A, Rn	将累加器的值与寄存器的值作逻辑与运算，结果存回累加器
ANL A, direct	将累加器的值与直接地址的内容作逻辑与运算，结果存回累加器
ANL A, @Ri	将累加器的值与间接地址的内容作逻辑与运算，结果存回累加器
ANL A, #data	将累加器的值与常量作逻辑与运算，结果存回累加器
ANL direct, A	将直接地址的内容与累加器的值作逻辑与运算，结果存回直接地址
ANL direct, #data	将直接地址的内容与常量作逻辑与运算，结果存回直接地址
ORL A, Rn	将累加器的值与寄存器的值作逻辑或运算，结果存回累加器
ORL A, direct	将累加器的值与直接地址的内容作逻辑或运算，结果存回累加器
ORL A, @Ri	将累加器的值与间接地址的内容作逻辑或运算，结果存回累加器
ORL A, #data	将累加器的值与常量作逻辑或运算，结果存回累加器
ORL direct, A	将直接地址的内容与累加器的值作逻辑或运算，结果存回直接地址
ORL direct, #data	将直接地址的内容与常量作逻辑或运算，结果存回直接地址
XRL A, Rn	将累加器的值与寄存器的值作逻辑异或运算，结果存回累加器
XRL A, direct	将累加器的值与直接地址的内容作逻辑异或运算，结果存回累加器
XRL A, @Ri	将累加器的值与间接地址的内容作逻辑异或运算，结果存回累加器
XRL A, #data	将累加器的值与常量作逻辑异或运算，结果存回累加器
XRL direct, A	将直接地址的内容与累加器的值作逻辑异或运算，结果存回直接地址
XRL direct, #data	将直接地址的内容与常量作逻辑异或运算，结果存回直接地址
CLR A	将累加器的值清零
CPL A	将累加器的值取反
RL A	将累加器的值左移一位
RLC A	将累加器的值带进位 C 左移一位
RR A	将累加器的值右移一位
RRC A	将累加器的值带进位 C 右移一位
SWAP A	将累加器的高 4 位与低 4 位的内容交换

26.4 数据传输指令

指令	功能描述
MOV A, Rn	将寄存器的内容赋值给累加器
MOV A, direct	将直接地址的内容赋值给累加器
MOV A, @Ri	将间接地址的内容赋值给累加器
MOV A, #data	将常量赋值给累加器
MOV Rn, A	将累加器的内容赋值给寄存器
MOV Rn, direct	将直接地址的内容赋值给寄存器
MOV Rn, #data	将常量赋值给寄存器
MOV direct, A	将累加器的内容赋值给直接地址
MOV direct, Rn	将寄存器的内容赋值给直接地址
MOV direct1, direct2	将直接地址 2 的内容赋值给直接地址 1
MOV direct, @Ri	将间接地址的内容赋值给直接地址
MOV direct, #data	将常量赋值给直接地址
MOV @Ri, A	将累加器的内容赋值给间接地址
MOV @Ri, direct	将直接地址的内容赋值给间接地址
MOV @Ri, #data	将常量赋值给间接地址
MOV DPTR, #data16	16 位常量赋值给数据指针寄存器
MOVC A, @A+DPTR	累加器的值加数据指针寄存器的值作为其所指定地址，将该地址的内容赋值给累加器
MOVC A, @A+PC	累加器的值加程序计数器的值作为其所指定地址，将该地址的内容赋值给累加器
MOVX A, @Ri	将间接地址所指定外部存储器的内容赋值给累加器（8 位地址）
MOVX A, @DPTR	将数据指针所指定外部存储器的内容赋值给累加器（16 位地址）
MOVX @Ri, A	将累加器的内容赋值给间接地址所指定的外部存储器（8 位地址）
MOVX @DPTR, A	将累加器的内容赋值给数据指针所指定的外部存储器（16 位地址）
PUSH direct	将直接地址的内容压入堆栈区
POP direct	将栈顶的内容弹出至直接地址
XCH A, Rn	将累加器与寄存器的内容互换
XCH A, direct	将累加器与直接地址的内容互换
XCH A, @Ri	将累加器与间接地址的内容互换
XCHD A, @Ri	将累加器的低 4 位与间接地址的低 4 位互换

26.5 布尔运算指令

指令	功能描述
CLR C	将进位 C 清零
CLR bit	将直接寻址位清零
SETB C	设置进位 C 为 1
SETB bit	设置直接寻址位为 1
CPL C	将进位 C 的值取反
CPL bit	将直接寻址位的值取反
ANL C, bit	将进位 C 的值与直接寻址位的值作逻辑与运算，结果存回进位 C
ANL C, /bit	将进位 C 的值与直接寻址位的反码作逻辑与运算，结果存回进位 C
ORL C, bit	将进位 C 的值与直接寻址位的值作逻辑或运算，结果存回进位 C
ORL C, /bit	将进位 C 的值与直接寻址位的反码作逻辑或运算，结果存回进位 C
MOV C, bit	将直接寻址位的值赋值给进位 C
MOV bit, C	将进位 C 的值赋值给直接寻址位

26.6 程序跳转指令

指令	功能描述
ACALL addr11	绝对调用
LCALL addr16	长调用
RET	从子程序返回
RETI	从中断返回
AJMP addr11	绝对跳转
LJMP addr16	长跳转
SJMP rel	短跳转（相对地址）
JMP @A+DPTR	跳至累加器的值加数据指针寄存器的值所指定的地址
JZ rel	若累加器的值为 0，则跳转至 rel 所指定的地址
JNZ rel	若累加器的值不为 0，则跳转至 rel 所指定的地址
JC rel	若进位 C 的值为 1，则跳转至 rel 所指定的地址
JNC rel	若进位 C 的值为 0，则跳转至 rel 所指定的地址
JB bit, rel	若直接寻址位的值为 1，则跳转至 rel 所指定的地址
JNB bit, rel	若直接寻址位的值为 0，则跳转至 rel 所指定的地址
JBC bit, rel	若直接寻址位的值为 1，则跳转至 rel 所指定的地址，并将该位值清零
CJNE A, direct, rel	将累加器的内容与直接地址的内容比较，若不相等则跳转至 rel 所指定的地
CJNE A, #data, rel	将累加器的内容与常量比较，若不相等则跳转至 rel 所指定的地址
CJNE Rn, #data, rel	将寄存器的内容与常量比较，若不相等则跳转至 rel 所指定的地址
CJNE @Ri, #data, rel	将间接地址的内容与常量比较，若不相等则跳转至 rel 所指定的地址
DJNZ Rn, rel	将寄存器的内容减 1，若不等于 0 则跳转至 rel 所指定的地址
DJNZ direct, rel	将直接地址的内容减 1，若不等于 0 则跳转至 rel 所指定的地址
NOP	空指令

27 调试界面

调试界面（SWAT），与 GPIO P1.3 共用一个引脚，可更新内置程序存储器 IROM，并可与开发环境配合工作。若单片机在上电前连接到 SN-Link，P1.3 共用引脚会自动设置为调试界面功能；相反，若单片机在上电过程中没有检测到任何握手信号，该引脚会设置为其它功能。

*注：引脚 P1.3 在仿真模式不能连接任何元器件。

27.1 最低要求

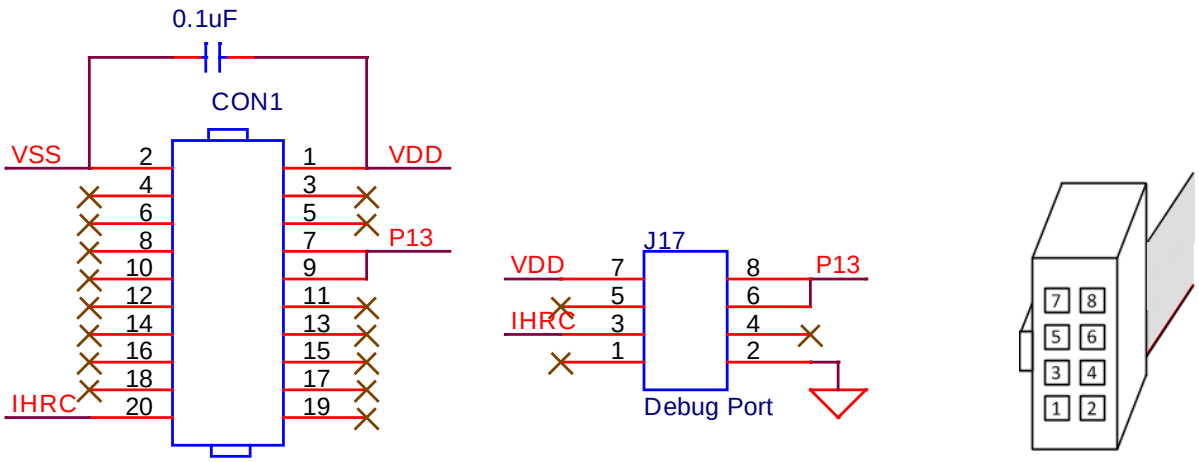
下面几项内容是建立合适的开发环境的基本要求，其兼容性已经经过验证，在后面的版本中可以很好的执行。SN-Link 的相关信息可以 SONiX 网址 www.sonix.com.tw 下载，Keil C51 可从 www.keil.com/c51 下载。

- **SN-Link Adapter II** 更新固有版本。
- **SN-Link Driver for Keil C51** 版本。
- **Keil C51** 版本。

27.2 硬件调试界面

下面的电路图显示了如何正确地连接单片机到 SWAT 引脚和 SN-Link Adapter II。

开始调试之前，必须切断单片机的电源（VDD）。把 SWAT 引脚连接到 SN-Link 的第 6 脚和第 8 脚，SN-Link 的第 2 脚和第 7 脚分别连接 VSS 和 VDD。打开单片机，就会自动开始握手操作，SN-Link 的红色指示灯（Run）显示连接成功。（详细内容请参考 SN8F5910 调试工具使用手册。）



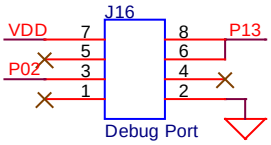
27.3 IHRC 校准

MCU 内部 32 MHz 时钟生成器（IHRC），无论封装或 Dice 的形式，在 IC 批量生产后，总是在 1%的精度下进行校准。但用户使用 DICE 开发产品时，可能由于 bonding 线和封胶的原因会造成 IHRC 的频率有所偏差。我们强烈建议 P02 和 P13 引脚必须存在，并保留为作者重新编程和 IHRC 重新校准，如有必要，在处理后，用 IHRC 的频移来解决少数 IC 的问题。

故 SN8F5910 系列提供 IHRC 校准功能。SN8F5910 系列提供的调试界面的 Writer 第 20Pin 用于 IHRC 校准。

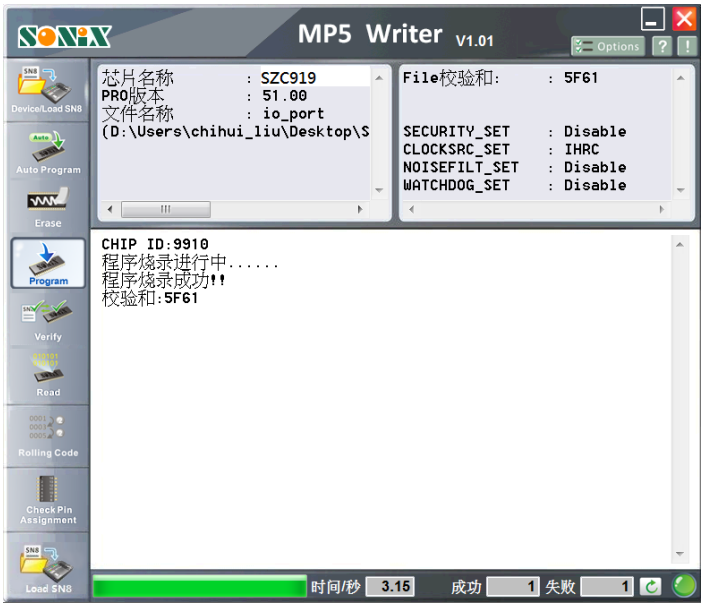
IHRC 校准的步骤如下：

- Step1: 从 SONIX 官网下载。
  (http://www.sonix.com.tw/article-tw-433-21531)
- Step2: 连接 **P02** 到 Writer 的 20 脚。



Writer 接口

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



*** 注：IHRC 校准功能包括在自动烧录功能中。**

28 ROM 烧录引脚

SN-LINK 和 MP5 Writer 支持 SN8F5910 系列 Flash ROM 的擦除/烧录/校正。

- SN-LINK：调试界面主要用于板上烧录；
- MP5 Writer：SN8F5910 系列版本。

28.1 MP5 烧录器烧录引脚图

SN8F5919FG:

Writer 连接器			MCU tool pin assignment		
JP5/IP6 引脚编号	JP5/IP6 引脚编号	MCU 引脚编号	MCU(4-wire) 引脚编号	MCU(1-wire) 引脚编号	80pin Board Converter
1	VDD	77/63	VDD	VDD	1
2	GND	65/80	VSS	VSS	2
3	DFTCLK	71	P0.4	-	3
5	SEL	70	P0.5	-	5
7	SWAT	57	P1.3	P1.3	7
9	SWAT	57	P1.3	P1.3	9
11	DAH	69	P0.6	-	11
13	DAL	68	P0.7	-	13
20	PDB	73	P0.2	P0.2	20

SN8F5918FG:

Writer 连接器			MCU tool pin assignment		
JP5/IP6 引脚编号	JP5/IP6 引脚编号	MCU 引脚编号	MCU(4-wire) 引脚编号	MCU(1-wire) 引脚编号	64pin Board Converter
1	VDD	62/50	VDD	VDD	1
2	GND	2/51	VSS	VSS	2
3	DFTCLK	57	P0.4	-	3
5	SEL	56	P0.5	-	5
7	SWAT	45	P1.3	P1.3	7
9	SWAT	45	P1.3	P1.3	9
11	DAH	55	P0.6	-	11
13	DAL	54	P0.7	-	13
20	PDB	59	P0.2	P0.2	20

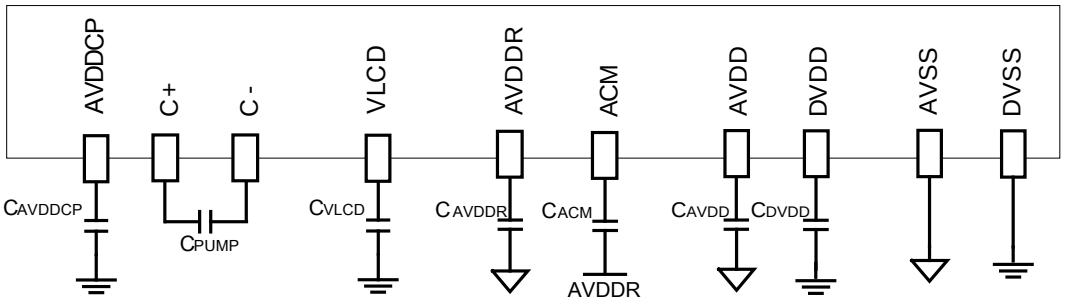
29 模拟设置与应用

SN8F5910 应用于许多 DC 测量应用程序，比如体重秤，压力测量，血压测量仪和温度计。下表显示了不同的应用程序设置，MCU 电源来自 CR2032 电池、AA/AAA 干电池或外部调节器。

Charge pump 使能电容表：

Power	AVDDR	ACM	AVDDCP	C+/-	VLCD	AVDD	DVDD
	C_{AVDDR}	C_{ACM}	C_{AVDDCP}	C_{PUMP}	C_{VLCD}	C_{AVDD}	C_{DVDD}
CR2032 (2.4 ~ 3.6V)	0.47uf	0.1uf	2.2uf	1uf	0.1uf	10uf/104	10uf/104
2*AA/AAA Bat (2.4 ~ 3.6V)	1uf	0.1uf	2.2uf	1uf	0.1uf	10uf/104	10uf/104
4*AA/AAA Bat (4 ~ 5.5V)							

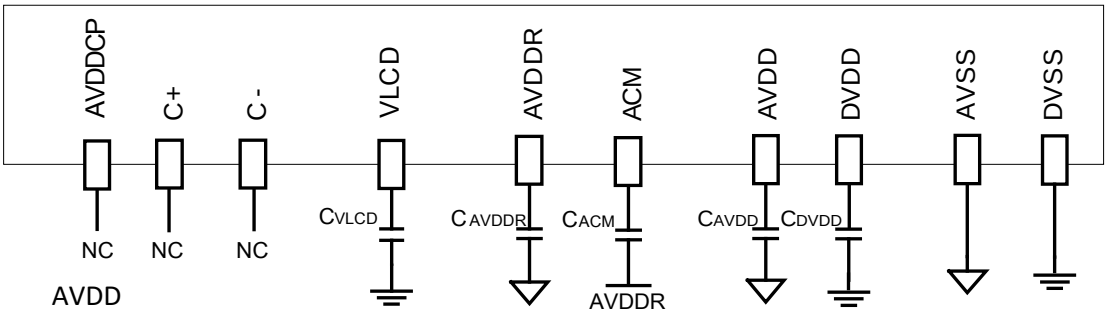
模拟电容连接 (Charge pump 使能):



Charge pump 禁止电容表 (Charge pump 禁止):

Power type	AVDDR	ACM	AVDDCP	C+/-	VLCD	AVDD	DVDD
	C_{AVDDR}	C_{ACM}	C_{AVDDCP}	C_{PUMP}	C_{VLCD}	C_{AVDD}	C_{DVDD}
4*AA/AAA Bat (4V ~5.5V)	1uf	0.1uf	NC *	NC	0.1uf	10uf/104	10uf/104

模拟电容连接 (Charge pump 禁止):



* AVDDCP与 AVDD 短路

30 订购信息

SONiX 的单片机的表面印有三栏信息：商标，单片机全称和日期码。

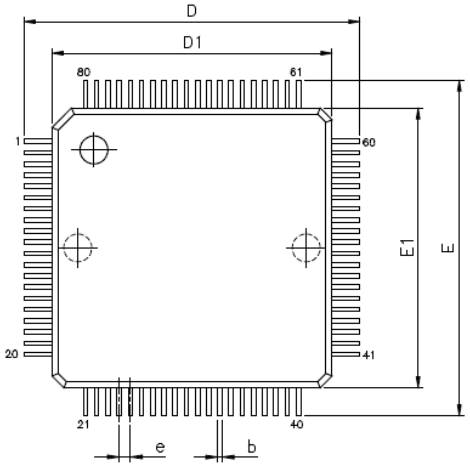
日期码：

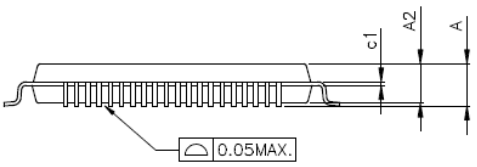
Year	15: 2015 16: 2016 17: 2017 et cetera
Month	1: January 2: February 3: March A: October B: November C: December et cetera
Date	1: 01 2: 02 3: 03 A: 10 B: 11 et cetera

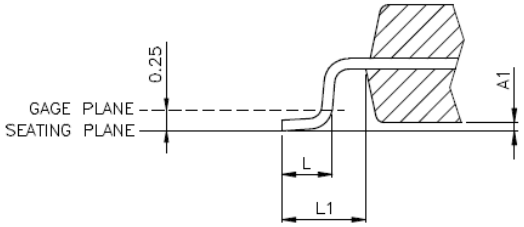
30.1 命名规则

单片机全称	封装类型
SN8F5919W	Wafer
SN8F5919H	Dice
SN8F5919FG	LQFP, 80 脚，绿色封装
SN8F5918FG	LQFP, 64 脚，绿色封装

30.2 LQFP 80 PIN

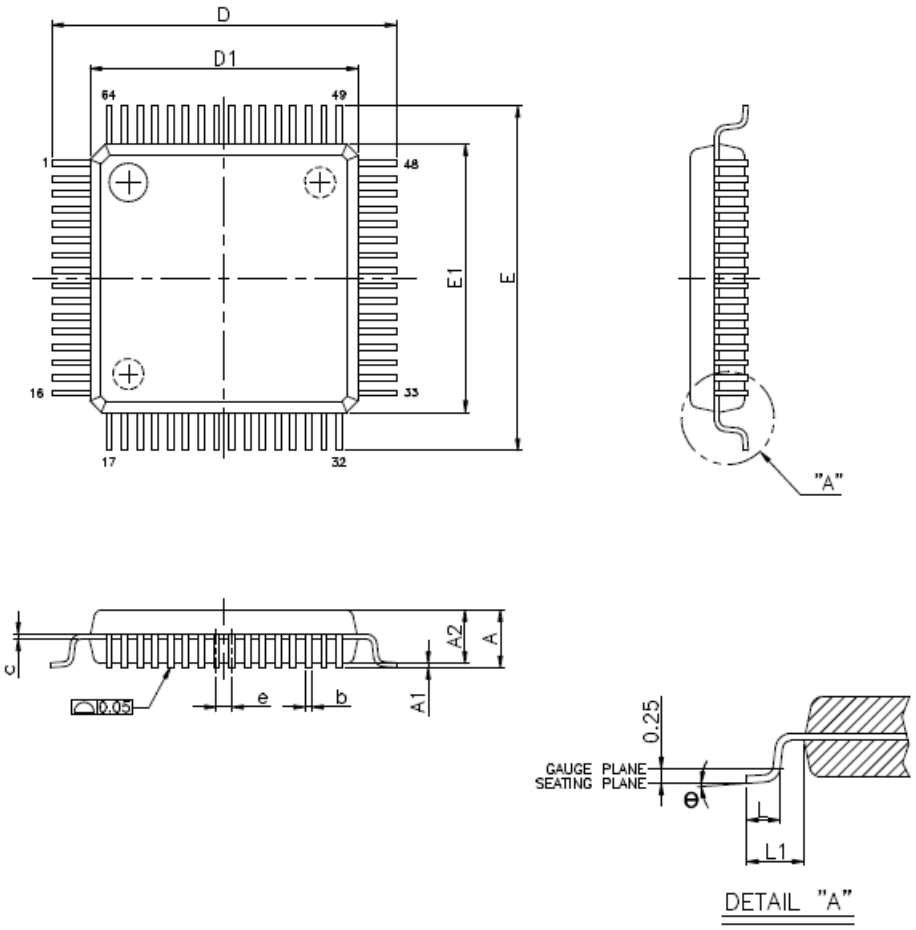






	Min	Typical (inch)	Max	Min	Typical (mm)	Max
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
c1	0.004		0.006	0.09		0.16
D		0.473 BSC			12 BSC	
D1		0.393 BSC			10 BSC	
E		0.473 BSC			12 BSC	
E1		0.393 BSC			10 BSC	
e		0.016 BSC			0.4 BSC	
B	0.005		0.009	0.13		0.23
L	0.018		0.030	0.45		0.75
L1		0.039 REF			1 .00 REF	

30.3 LQFP 64 PIN



	Min	Typical (inch)	Max	Min	Typical (mm)	Max
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
c	0.004		0.008	0.09		0.20
D		0.354 BSC			9 BSC	
D1		0.276 BSC			7 BSC	
E		0.354 BSC			9 BSC	
E1		0.276 BSC			7 BSC	
e		0.016 BSC			0.4 BSC	
b	0.005		0.009	0.13		0.23
L	0.018		0.030	0.45		0.75
L1		0.039 REF			1 .00 REF	

SN8F5910 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