SONiX Technology Co., Ltd.

SN8F5950 Series Datasheet

8051-based Microcontroller

SN8F5959FG
SN8F5958FG

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 32 MHz 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：16MIPs ~ 0.25MIPs

- 内存
  - 128 KB Flash 内存（IROM），支持在线编程功能
  - 4.5KB 数据存储器 (XROM)
  - 256 字节内部 RAM (IRAM)
  - 8K 字节外部 RAM (XRAM)

- I/O 引脚：
  - 双向 IO： P0, P1, P2, P3, P4, P5
  - 带上拉电阻 IO： P0, P1, P2, P3, P4, P5
  - P1,P2,P3, P4, P5 与 SEG 引脚共用
  - 唤醒 IO：P0, P1 电平变化触发
  - P0, P1 IDLE 模式, STOP 模式唤醒

- 中断：17 中断源，可控制其优先权，且对应唯一的中断向量
  - 13 内部中断： T0, TC0, TC1, TC2, ADC, UART-TX/RX(x2), I2C, SPI, LVD, BTR
  - 4 外部中断：INT0, INT1, INT2, INT3

- 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
  - 3 组 16 位定时器/计数器，带自动重装，PWM，Buzzer 输出 (TC0~TC2)
  - 1 个 Buzzer 引脚(P12)：1KHz/2KHz/4KHz/8KHz

- 复位引脚(P12)：外部复位

- UART/ I2C 接口，支持 SMBus

- 比较器低电压检测 2.2V~3.3V (每阶 0.1V)

- 电源：1.8~5.5V (AVDD/DVDD)

- 升压稳压器：
  - 输入电压：2.3V ~ 5.5V
  - AVDDR 稳压输出为模拟电路供电(ADC, PGIA)
  - AVDDR 电压选项：2.2V,2.4V, 2.8V ,3.2V
  - ACM 0.75V/1V 为 ADC 通用电压

- PGIA：可程序设计控制增益放大器，高输入阻抗
  - 增益：1x, 4x, 8x, 16x, 32x, 64x, 128x

- 24 位 Delta Sigma ADC：
  - ADC 增益：1x, 2x
  - ADC 输入通道： AI1 ~ AI9, ACM, VDD_DET(1/8*VBAT)
  - ADC 偏移选项： (-1/4, -2/4, -3/4) * Vref
  - 输入通道配置：单一或者差分
  - ADC 转换率：7.6Hz ~ 7.8KHz
  - ADC 时钟：250kHz, 333kHz

- 12 位 DAC：
  - DAC 内部参考：(x1, 1/2, 1/4) xAVDDR , xAVE
  - 外部参考： AVE

- BIA 电路：
  - BIA 测量：AC 频率： 1K~250Khz
  - 阻抗范围：1K ~ 30K Ω
  - 相位检测器：0~90°

- LCD 驱动：C-Type LCD 驱动高达 204 点
  - 4 com 或 6 com， 1/3 bias
  - 4x36 或 6x34 dots，LCD 引脚与 I/O 引脚共用
  - 可调节 VLCD 输出 2.6V ~ 4.5V

◆ OP A & Switch：
- OPA：Rail to Rail OPA1/OPA2
- Switch：集成电路 SC0~SC11

◆ 片上调试：
- 单线调试界面
- 5 个硬件断点
- 无限次软件断点
- ROM 数据安全保护
- 看门狗和可程序设计的外部复位

◆ 封装
- LQFP 80 pin, LQFP 64 pin ,Dice

◆ RS 高性能

◆ ESD 高性能

◆ 有效 CRC ： CRC-16 多项式，CRC-16-CCITT 多项式

1.2 应用领域

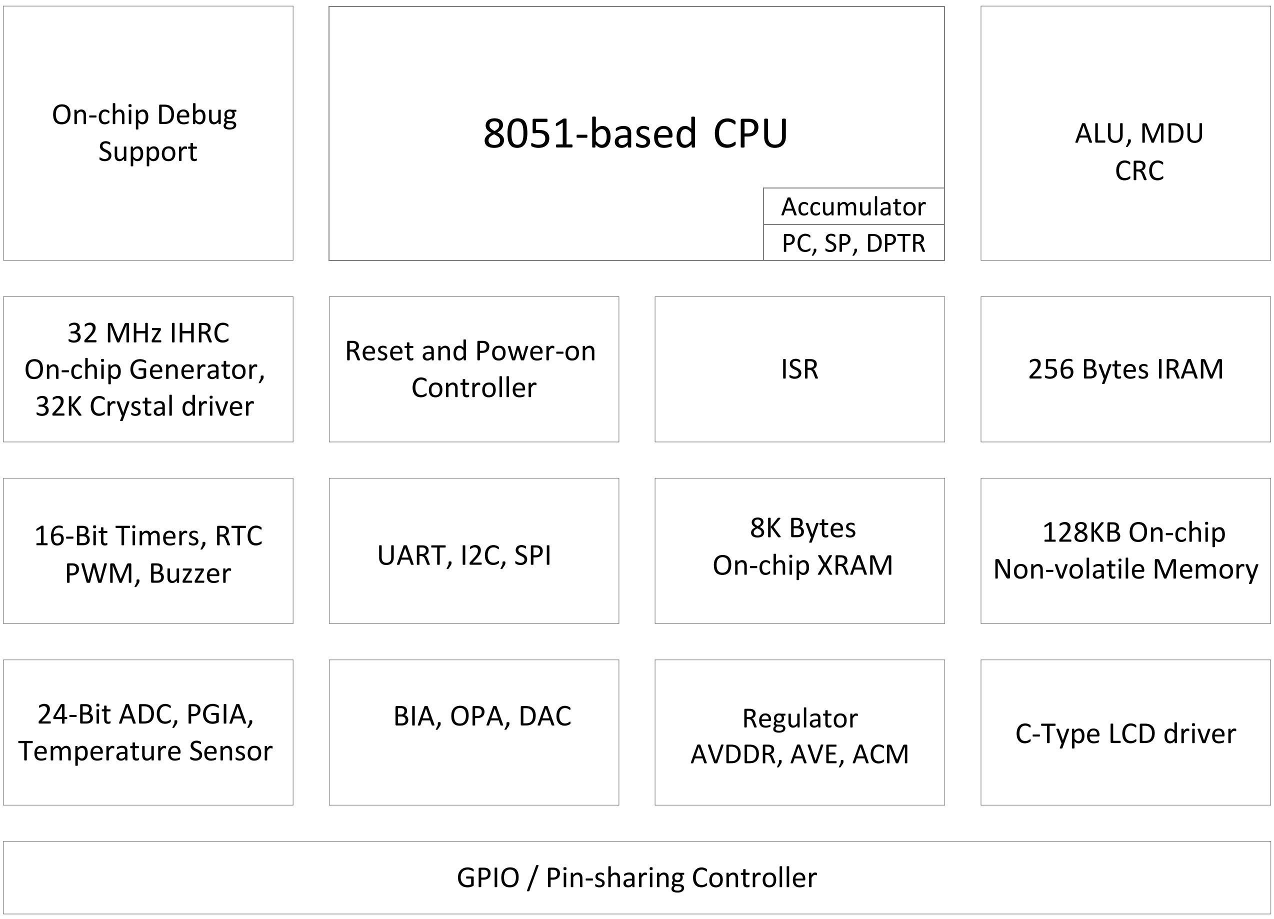
- 血糖仪
- 血压计
- 其它测量应用
- 温度计
- 体重秤

1.3 产品性能表

	I/O	LCD	RTC	16-Bit Timer / PWM	I2C	SPI	UART x2	ADC 24Bit	DAC 12Bit	OPA	CMP	BIA	Int. INT	Ext. INT	Package Types
SN8F5959	14+34*	4x36, 6x34	V	3	V	V	V	9-Ch	1	2	1	1	13	4	LQFP80
SN8F5958	14+22*	4x24, 6x22	V	3	V	V	V	9-Ch	1	1	1	1	13	4	LQFP64

* LCD 引脚与 I/O 引脚共用

1.4 框图



2 目录

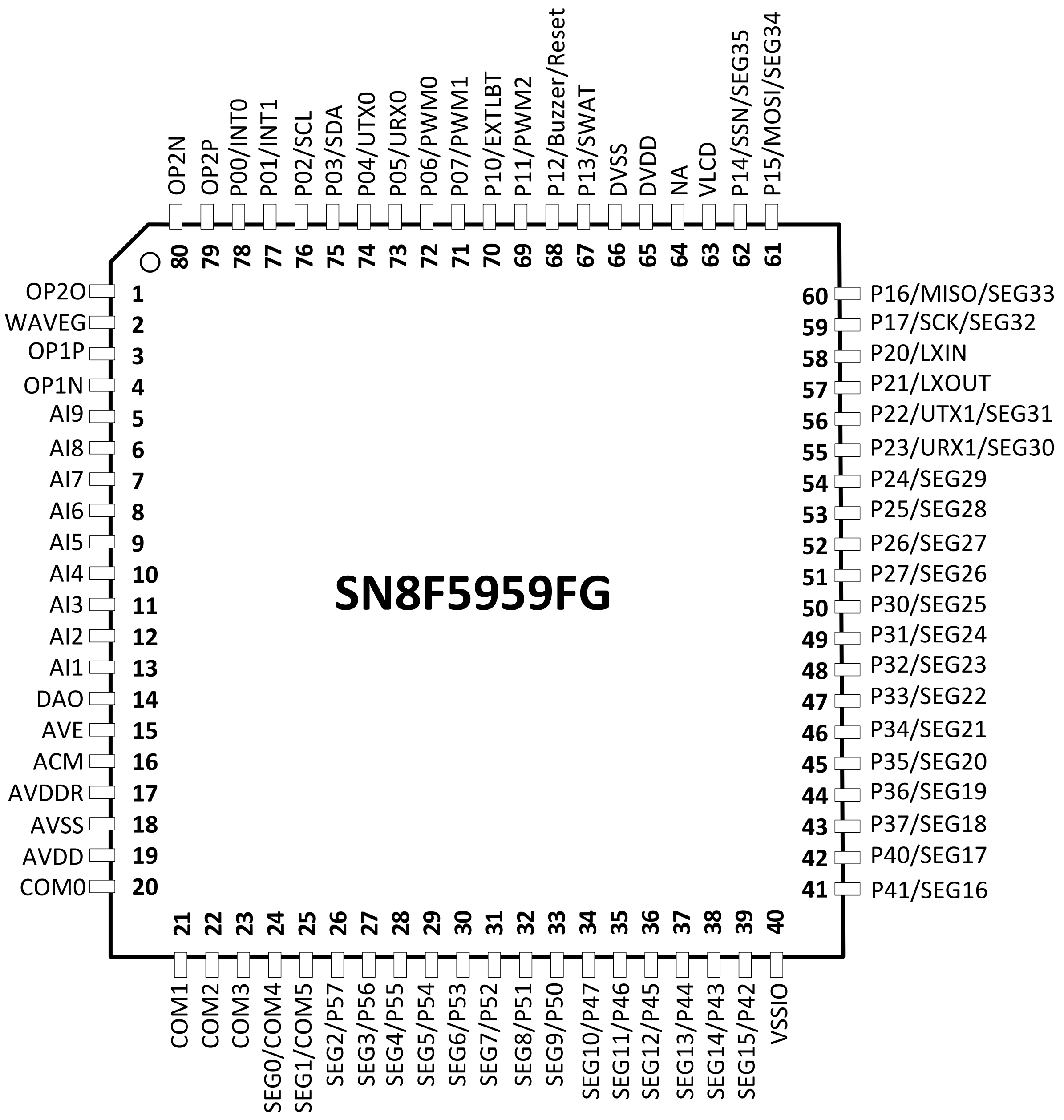
3 修订记录

版本	时间	修订说明
1.0	Jul.2022	1. 初版。
1.1	Jul. 2022	1. 更新 Starter-Kit 板电路； 2. 更新 OPA 控制寄存器；
1.1	Jul. 2022	1. In-System Program 章节。
1.2	Aug. 2022	1. 修改产品性能表; 2. 修改 XRAM：4.5KB->4KB。
1.3	Sep. 2022	1. 修改模拟设置图; 2. 增加 PAGESEL 寄存器。
1.4	Nov. 2022	1. 修改 P3/P4/P5 LCD 共用引脚描述; 2. 修改 P1W 寄存器表。
1.5	Jun. 2023	1. 增加 code option 表。
1.6	Apr.2023	1. 修改 DAM Bit[4..5] 描述; 2. 修改 LBT 框图。
1.7	June.2023	1. 增加数据内存注意事项。
1.8	Aug.2023	1. 修改 AVDDR 操作范围。
1.9	Apr.2024	1. 当 MCU 进入 Stop 模式，IO 状态必须设置输出高/低或输入上拉，避免掉电功耗异常; 2. 修改 S0CON2 寄存器。
2.0	Apr.2024	1. SN8F5928 修改为 SN8F5959.
2.1	Sept.2024	1. 增加 UART1 章节; 2. 修改 WDT 时钟 = FLRC/1024.
2.2		1. 修改 UART 表。

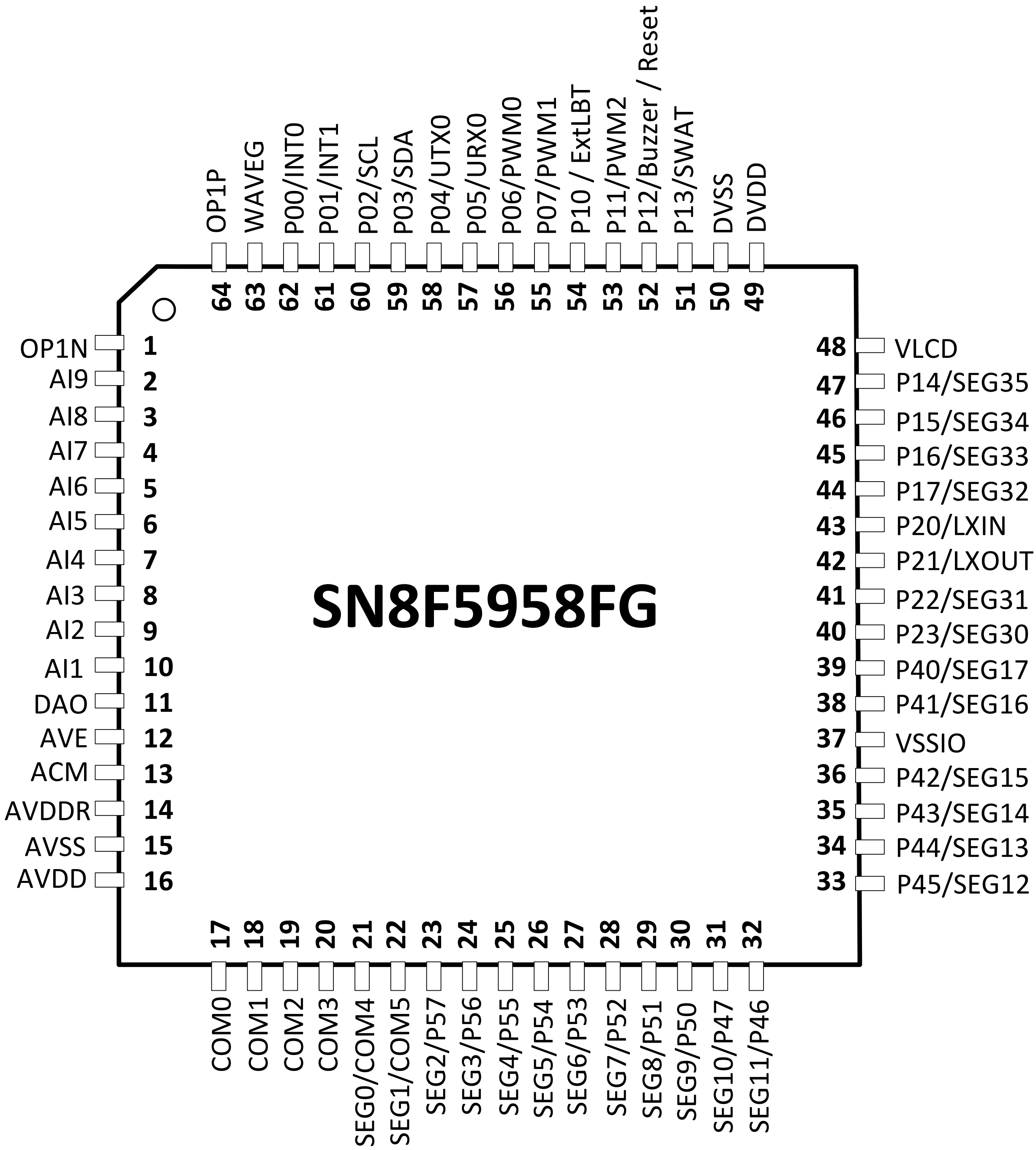
4 引脚配置

4.1 引脚图

SN8F5959F (LQFP80)



SN8F5958F (LQFP64)



4.2 引脚说明

电源引脚

引脚名称	类型	功能说明
DVDD	Power	数字电源(1.8V~5.5V)
DVSS	Power	数字地(0V)
AVDD	Power	模拟电源(1.8V~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	GPIO / INT0 中断输入
P0.1 / INT1	Digital I/O	GPIO / INT1 中断输入
P0.2 / SCL	Digital I/O	GPIO / I2C：时钟输出（主控）时钟输入（从动）
P0.3 / SDA	Digital I/O	GPIO / I2C：数据引脚
P0.4 / UTX0	Digital I/O	GPIO / UTX0：数据输出引脚
P0.5 / URX0	Digital I/O	GPIO / URX0：数据输入引脚
P0.6 / PWM0	Digital I/O	GPIO / PWM0 输出引脚
P0.7 / PWM1	Digital I/O	GPIO / PWM1 输出引脚

P1

引脚名称	类型	功能说明
P1.0 / LBT0	Digital / Analog	GPIO / 比较器外部输入引脚
P1.1 / LBT1	Digital / Analog	GPIO / 比较器外部输入引脚
P1.2 /RESET/Buzzer	Digital I/O	GPIO / 外部复位引脚 / Buzzer
P1.3 / MISO/SWAT	Digital I/O	GPIO /单线仿真接口 (*P1.3 口不能接任何器件@仿真模式)
P1.4/SSN/PWM2	Digital I/O	GPIO / SSN 从动选择引脚 / PWM2
P1.5/MOSI	Digital I/O	GPIO / MOSI：发送引脚（主控）接收引脚（从动）
P1.6/MISO	Digital I/O	GPIO / MISO：接收引脚（主控）发送引脚（从动）
P1.7/SCK	Digital I/O	GPIO / SCK：SIO 时钟引脚

P2

引脚名称	类型	功能说明
P2.0 / LXIN	Digital / Analog	GPIO / 系统时钟：外部时钟输入引脚
P2.1 / LXOUT	Digital / Analog	GPIO / 系统时钟：驱动外部晶振
P2.2 / SEG31 / UTX1	Digital I/O	GPIO / SEG31 / UTX1：数据输出引脚
P2.3 / SEG30 / URX1	Digital I/O	GPIO / SEG30 / URX1：数据输入引脚
P2.4 / SEG29	Digital / LCD	GPIO / SEG29
P2.5 / SEG28	Digital / LCD	GPIO / SEG28
P2.6 / SEG27	Digital / LCD	GPIO / SEG27
P2.7 / SEG26	Digital / LCD	GPIO / SEG26

P3

引脚名称	类型	功能说明
P3.0 / SEG25	Digital/ LCD	GPIO / SEG25
P3.1 / SEG24	Digital/ LCD	GPIO / SEG24
P3.2 / SEG23	Digital/ LCD	GPIO / SEG23
P3.3 / SEG22	Digital/ LCD	GPIO / SEG22
P3.4 / SEG21	Digital/ LCD	GPIO / SEG 21
P3.5 / SEG20	Digital/ LCD	GPIO / SEG 20
P3.6 / SEG19	Digital/ LCD	GPIO / SEG 19
P3.7 / SEG18	Digital/ LCD	GPIO / SEG 18

P4

引脚名称	类型	功能说明
P4.0 / SEG17	Digital/ LCD	GPIO / SEG17
P4.1 / SEG16	Digital/ LCD	GPIO / SEG16
P4.2 / SEG15	Digital/ LCD	GPIO / SEG15
P4.3 / SEG14	Digital/ LCD	GPIO / SEG14
P4.4 / SEG13	Digital/ LCD	GPIO / SEG13
P4.5 / SEG12	Digital/ LCD	GPIO / SEG12
P4.6 / SEG11	Digital/ LCD	GPIO / SEG11
P4.7 / SEG10	Digital/ LCD	GPIO / SEG10

P5

引脚名称	类型	功能说明
P5.0 / SEG9	Digital/ LCD	GPIO / SEG9
P5.1 / SEG8	Digital/ LCD	GPIO / SEG8
P5.2 / SEG7	Digital/ LCD	GPIO / SEG7
P5.3 / SEG6	Digital/ LCD	GPIO / SEG6
P5.4 / SEG5	Digital/ LCD	GPIO / SEG5
P5.5 / SEG4	Digital/ LCD	GPIO / SEG4
P5.6 / SEG3	Digital/ LCD	GPIO / SEG3
P5.7 / SEG2	Digital/ LCD	GPIO / SEG2

LCD

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

模拟引脚

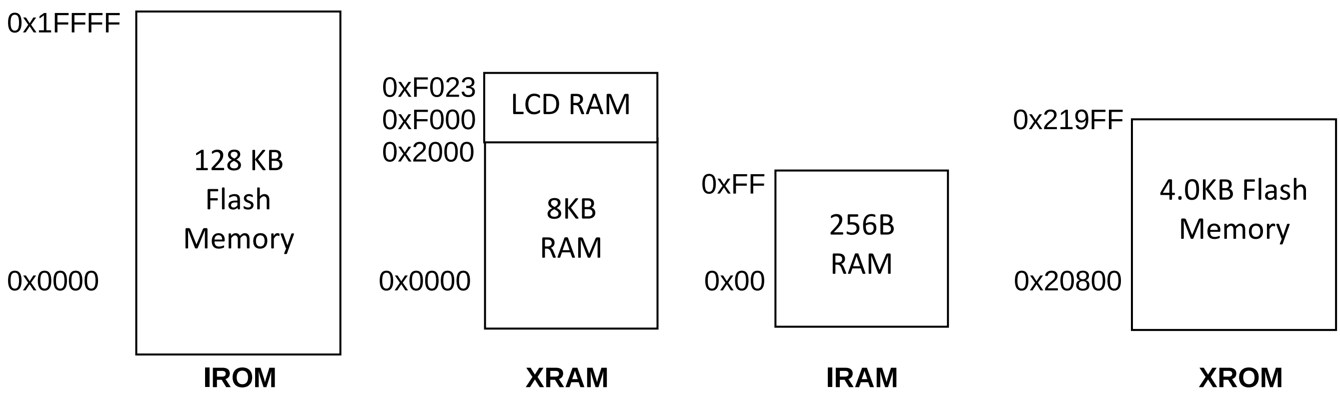
引脚名称	类型	功能说明
AVDD	Power	模拟电压输入端
AVSS	Power	模拟地
AVDDR	Analog Output	Regulator 输出模拟端，连接 1uF 电容到 AVSS 可选项： 2.4V,2.8V,3.2V
OP1N	Analog Input	OP1 负电压输入
OP1P	Analog Input	OP1 正电压输入
OP1O	Analog output	OP1 电压输出
OP2N	Analog Input	OP2 负电压输入
OP2P	Analog Input	OP2 正电压输入
OP2O	Analog output	OP2 电压输出
AVE	Analog Input	可选项：2.0V, 1.5V
ACM	Analog Output	ACM 电压输出 0.7V/1.0V 作为 ADC 通用电压（仅在灌电流时） 连接 0.1uF 电容到 AVDDR
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 I 3 Vref：外部参考电压正极输入引脚
AI4 R-	Analog Input Analog Input	ADC：输入通道 4 Vref：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

5 CPU

SN8F5000 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模拟（例如 Keil C51）。总的来说，在相同的频率下，SN8F5000 的 CPU 要比原始的 8051 快 9.4 到 12.1 倍。

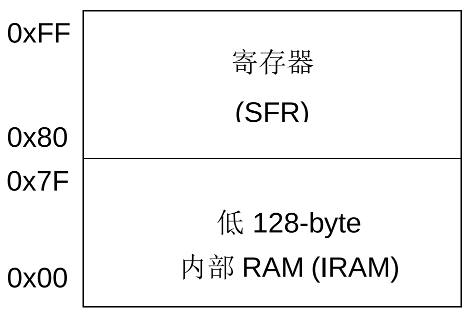
5.1 存储器结构

SN8F5950 内建了三个内存：内部 RAM(IRAM)，外部 RAM(XRAM)，以及程序内存(IROM)和数据内存（XROM）。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 8KB 大小，但需要更长的存取周期。程序内存是一个 128KB 的 FLASH 内存，最快的访问速度可达 16MHz。



5.2 直接寻址：IRAM 和 SFR

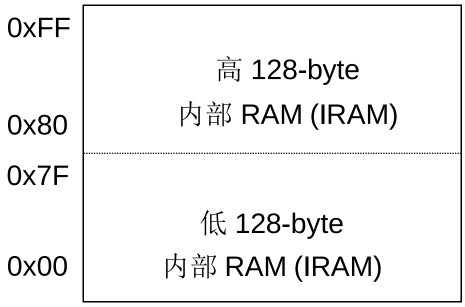
直接寻址指令（如 MOV A, direct）可以访问低 128 字节的内部 RAM（地址范围：00-7FH）和所有系统寄存器（SFR，地址范围：80-0FFH）。



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。

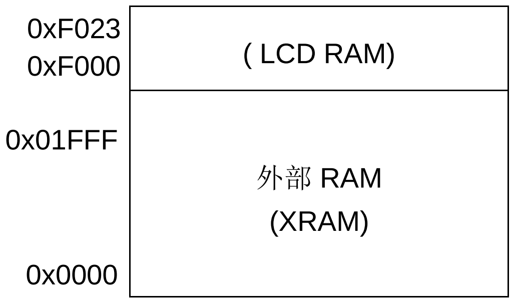
5.3 间接寻址：IRAM

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，但间接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)

程序内存是 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 章节）。

5.6 程序存储器安全

SN8F5950 内置 ROM 加密机制，防止 Flash ROM 数据被破解。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，数据指针可说明指定 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内存，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 的变量重迭。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。

预设情况下，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 Register (0x81)

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

DPL Register (0x82)

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

DPH Register (0x83)

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

DPL1 Register (0x84)

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

DPH1 Register (0x85)

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

DPS Register (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 Register (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表

BIN / HEX	000	001	010	011	100	101	110	111
F8	P5	P0M	P1M	P2M	P3M	P4M	P5M	PFLAG
F0	B	P0UR	P1UR	P2UR	P3UR	P4UR	P5UR	SRST
E8	P4	MD0	MD1	MD2	MD3	MD4	MD5	ARCON
E0	ACC	SPSTA	SPCON	SPDAT	P1OC	CLKSEL	CLKCMD	CHS1
D8	S0CON2	S0RBUF	I2CDAT	I2CADR	I2CCON	I2CSTA	SMBSEL	SMBDST
D0	PSW	CHS0	VREG	AMPM	ADCM1	ADCM2	BZRM	LBTM
C8	OPM1	TC1M	TC1RL	TC1RH	TC1CL	TC1CH	TC1DL	TC1DH
C0	IRCON	TC0M	TC0RL	TC0RH	TC0CL	TC0CH	TC0DL	TC0DH
B8	IEN1	IP1	S0RELH	T0M	T0C	MIN	SEC	IQCON1
B0	P3	LCDM1	LCDM2	DDSCON1	IQCON0	ADCDL	ADCDM	ADCDH
A8	IEN0	IP0	S0RELL	DAM	DABL	TC2DH	DDSCON2	PTW
A0	P2	TC2M	TC2RL	TC2RH	TC2CL	TC2CH	TC2DL	PERAMCNT
98	S0CON	S0TBUF	IEN2	PAGESEL	PEROMHH	P5CON	P3CON	P4CON
90	P1	P1W	DPS	DPC	PECMD	PEROML	PEROMH	PERAM
88	OPM2	P2CON	S1RBUF	S1RELH	S1RELL	S1CON	S1TBUF	PEDGE
80	P0	SP	DPL	DPH	DPL1	DPH1	WDTR	PCON

6.2 SRAM 存储表

BIN / HEX	000	001	010	011	100	101	110	111
FFF0	FRQH	FRQL	FRQCMD	-	-		OP2SW	-
FFE8	DDSDATA	FTW	P1CON	LCDM3	LVDCON	CRCM	CRCDATL	CRCDATH
FFE0		DDSCON0	PDRV	LFRQL	LFRQH	CLKCAL	-	DDSADR

6.3 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。
PCON	0x87	
OPM2	0x88	OPA 控制寄存器。
P2CON	0x89	P2 配置控制寄存器。
S1RBUF	0x8A	UART1 RX 数据缓存器。
S1RELH	0x8B	
S1RELL	0x8C	
S1CON	0x8D	UART1 控制寄存器。
S1TBUF	0x8E	UART1 TX 数据缓存器。
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	在线程序设计命令寄存器
PEROML	0x95	在线程序设计 ROM 地址低字节
PEROMH	0x96	在线程序设计 ROM 地址高字节
PERAM	0x97	在线程序设计 RAM 分配地址
S0CON	0x98	UART0 控制寄存器。
S0TBUF	0x99	UART0 TX 数据缓存器。
IEN2	0x9A	中断使能寄存器。
PAGESEL	0x9B	
PEROMHH	0x9C	在线程序设计 ROM 选择寄存器。
P5CON	0x9D	P5 配置控制寄存器。
P3CON	0x9E	P3 配置控制寄存器。
P4CON	0x9F	P4 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。
TC2M	0xA1	TC2 定时器模式控制寄存器。
TC2RL	0xA2	TC2 定时器计数器重装缓存器低字节。
TC2RH	0xA3	TC2 定时器计数器重装缓存器高字节。
TC2CL	0xA4	TC2 定时器低字节计数器。
TC2CH	0xA5	TC2 定时器高字节计数器。
TC2DL	0xA6	PWM2 占空比控制低字节缓存器。
TC2DH	0xA7	PWM2 占空比控制高字节缓存器。
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。
DAM	0xAB	DAC 模式寄存器
DABL	0xAC	DAC 数据低字节寄存器。
TC2DH	0xAD	TC2 占空比控制高字节缓存器。
TC2DL	0xAE	TC2 占空比控制低字节缓存器。
PERAMCNT	0xAF	
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。
DDSCON1	0xB3	-
IQCON0	0xB4	-
ADCDL	0xB5	ADC 输出数据低字节。
ADCDM	0xB6	ADC 输出数据中字节。
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	T0 定时器模式控制寄存器。
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。
IQCON1	0xBF	-

0xC0 - 0xDF R 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。
OPM1	0xC8	OPA 控制寄存器。
TC1M	0xC9	TC1 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。
CHS0	0xD1	ADC 输入通道选择寄存器。
VREG	0xD2	电压稳压器控制寄存器。
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。
S0RBUF	0xD9	UART RX 数据缓存器。
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。
SPSTA	0xE1	SPI 状态寄存器。
SPCON	0xE2	SPI 控制寄存器。
SPDAT	0xE3	SPI 数据缓存器。
P1OC	0xE4	开漏功能控制寄存器。
CLKSEL	0xE5	时钟开关切换选择寄存器。
CLKCMD	0xE6	时钟开关控制寄存器。
P4	0xE8	P4 数据缓存器。
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。
P4UR	0xF5	P4 上拉电阻控制寄存器。
P5UR	0xF6	P5 上拉电阻控制寄存器。
SRST	0xF7	软件复位控制寄存器。
P5	0xF8	P5 数据缓存器。
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。
P4M	0xFD	P4 输入/输出模式寄存器。
P5M	0xFE	P5 输入/输出模式寄存器。
PFLAG	0xFF	复位标志寄存器。

6.4 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 Register (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 Register (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 寄存器来保留临时资料。

PSW Register (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元后移出逻辑“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元后移出逻辑“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

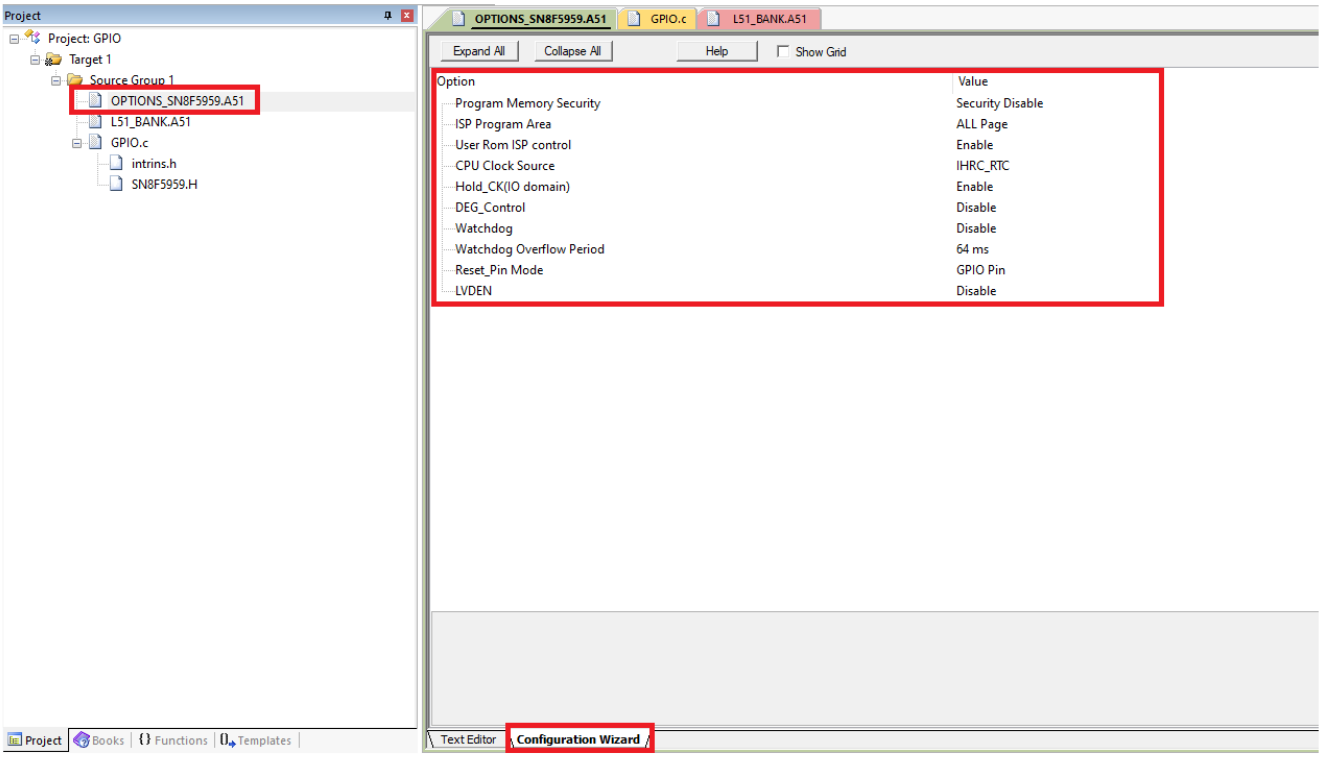
7 复位和上电控制

复位和上电控制有下列方式：低电压检测（LVDs），看门狗，可程序设计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 SN8F595x_OPTIONS.A51 文件（从松翰官网 www.sonix.com.tw 下载）。该档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。SN8F5000 的调试工具手册提供了更详细的内容。具体选项如下所示：

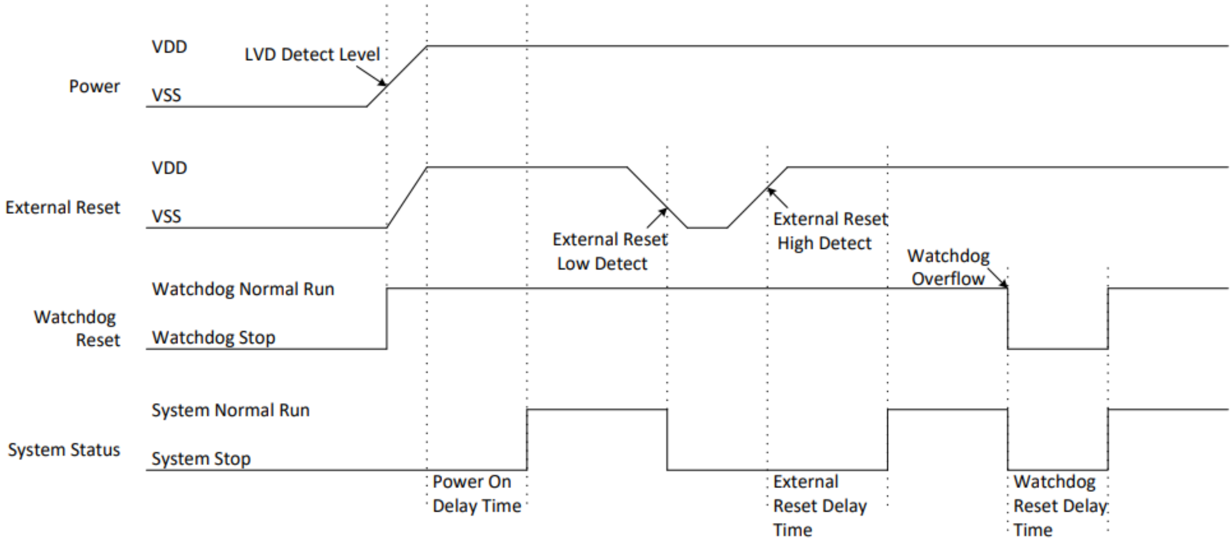
- 程序加密
- ISP 编程区域
- 用户 ROM ISP 控制
- CPU 时钟源
- Hold 时钟
- DGE 控制
- 看门狗
- 看门狗溢出周期
- 复位引脚
- LVDEN



编译选项	内容	功能说明
Program Memory Security	Disable (SC0/SC1)	禁用 ROM 代码加密功能，所有 ROM 区域内容都能访问。
	Enable (CS2)	使能“特殊 ROM 加密功能” 所有 ROM 区域内容访问全部都是 0x00，但地址 0x1E000 ~ 0x1FFBF 除外。
	Enable (CS3)	使能 ROM 代码加密功能，所有 ROM 区域内容访问全部都是 0x00。
ISP Program Area	ALL Page	内存 0x0000~0x1FFFF 被执行。
	Page 252~255	内存 0x1F800~0x1FFFF 被执行。
User Rom ISP control	Enable	所有页可执行 ISP。
	Disable	地址 0x20800~0x219FFF 可执行 ISP。
CPU Clock Source	IHRC	内部高速 32MHz RC。 XIN/XOUT 引脚为双向 GPIO 模式。
	IHRC_RTC	内部高速 32MHz RC。XIN/XOUT 引脚接外部低速时钟 32768Hz 晶振，低速震荡模式。
Hold_CK	Enable	Hold 时钟有效。
	Disable	Hold 时钟无效。
DEG_Control	Enable	Debug 模式控制有效。
	Disable	Debug 模式控制无效。
Watchdog Reset	Always	始终开启看门狗定时器，即使在 STOP 模式和 ILDE 模式下也处于开始状态。
	Enable	开启看门狗定时器，但在 STOP 和 ILDE 模式时关闭。
	Disable	关闭看门狗定时器。
Watchdog Overflow Period	64ms	看门狗定时器时钟源 F_{ILRC} /4
	128ms	看门狗定时器时钟源 F_{ILRC} /8
	256ms	看门狗定时器时钟源 F_{ILRC} /16
	512ms	看门狗定时器时钟源 F_{ILRC} /32
Reset Pin Mode	Reset with De-bounce	使能外部复位引脚，带 De-bounce。
	Reset without De-bounce	使能外部复位引脚，不带 De-bounce。
	GPIO with P12	使能 P12 GPIO 模式
LVDEN	Enable	LVD17 功能有效。
	Disable	LVD17 功能无效。

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 1.7V。VDD 的电压低于 1.7V 时，MCU 复位。

低电压检测监控 VDD 引脚的电压仅一个电压点：1.7V – 4.5V 之间。根据低电压检测配置，比较结果作为系统复位信号。下表列出了低电压检测配置“LVD”和 VDD 引脚状态结果

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LVDCON	-	-	INTLVL2	INTLVL1	INTLVL0	RSTLVL2	RSTLVL1	RSTLVL0
IEN2	ELVD	IE3	IE2	EADC	ETC2	ETC1	ETC0	EBTR
IRCON	ADCF	TCF2	TCF1	TCF0	LVDF	TF0	IE1	IE0

LVDCON Register (0xFFEC)

Bit	Field	Type	Initial	Description
7..6	Reserved	R	00	
5..3	INTLVL	R/W	0	LVD 电压选择标志位。 000：1.7V； 001：2.1V； 010：2.5V； 011：2.9V； 100：3.3V； 101：3.7V； 110：4.1V； 111：4.5V。
2..0	RSTLV	R/W	0	LVD 复位电压选择位。 000：1.7V； 001：2.1V； 010：2.5V； 011：2.9V； 100：3.3V； 101：3.7V； 110：4.1V； 111：4.5V。

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
7	ELVD	R/W	0	LVD 中断控制位。 0：禁止 LVD 中断功能； 1：使能 LVD 中断功能。

IRCON Register (0xC0)

Bit	Field	Type	Initial	Description
7	LVDF	R/W	0	LVD 中断请求标志位。 0：无 LVD 中断请求； 1：LVD 中断请求。

7.4 看门狗复位

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。

```
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 $= F_{ILRC} \div 1024$

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

SN8F5950 系列，内置一个外部复位引脚，当此引脚电平由高拉低，系统进入复位状态。

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。

```
SRST = 0x01;
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	-	NDT5	LVD14	LVD	TRIGFLAG	RST
SRST	-	-	-	-	-	-	-	SRSTREQ
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0

PFLAG Register

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 Register

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。

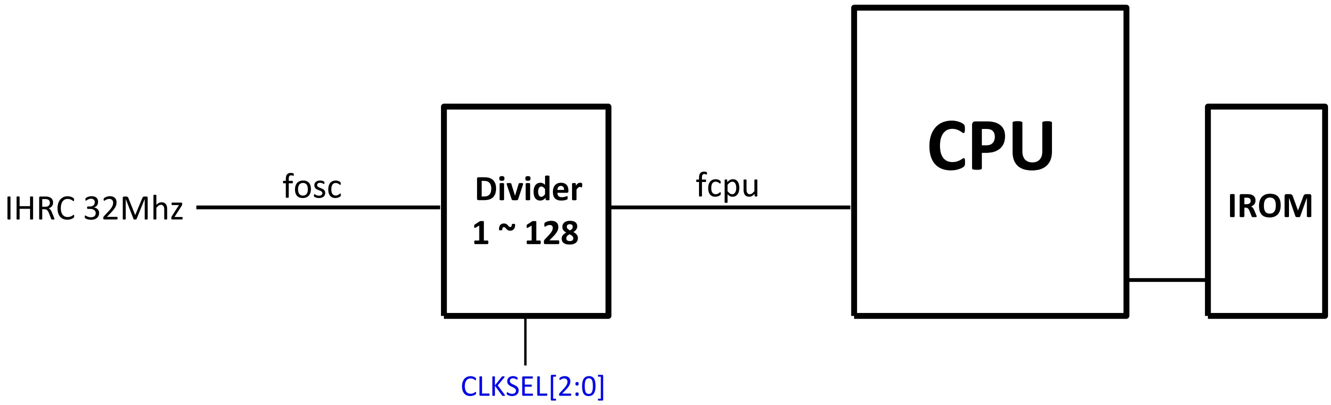
NORMAL 模式是指 CPU 和外设功能都正常工作，系统时钟是通过所选择的时钟源，程序设置的分频数，以及程序设置的 ROM 等待时间所确定的。IDLE 模式是指 CPU 时钟暂停的状态，但保留了外设功能（如定时器，UART 和 I2C）。与之相反的是，STOP 模式则禁止所有功能和时钟发生器，直至唤醒信号将系统唤醒进入 NORMAL 模式。此外，LCD 功能和 T0-RTC 功能也能在 STOP 模式下正常工作，为低待机电流应用。

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。

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



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 编译程序开发时，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。

```
IDLE();  Use C51 Macros into IDLE MODE
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
CLKSEL	-	-	-	-	-	CLKSEL2	CLKSEL1	CLKSEL0
CLKCMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PCON	SMOD	-	-	IDLE+	P2SEL	GF0	STOP	IDLE
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W

CLKSEL Register (0xE5)

Bit	Field	Type	Initial	Description
7..3	Reserved	R	0x00	
2..0	CLKSEL[2:0]	R/W	111	CLKSEL 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； ~~111：fcpu = fosc / 1。~~ *设置 CLKSEL[2:0]后，必须写入 0x69 以应用 CLKSEL 的设置。 *fosc =32Mhz.

CLKCMD Register (0xE6)

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

P1W Register (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。

PCON Register (0x87)

Bit	Field	Type	Initial	Description
7				参考其它章节。
6..5	Reserved	R	0x00	
4	IDLE+	R/W	0	1：单片机切换到 IDLE+ 模式。
3	P2SEL	R/W	1	高命令地址自己配置位。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。

9 中断

SN8F5950 包含 11 个中断源（2 个外部中断和 9 个内部中断），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
UART1 Rx	ERS1	RI1	软件清 0	2 / 0x0083
UART1 Tx	ETS1	TI1	软件清 0	3 / 0x0043
INT1	EX1	IE1	自动清 0	4 / 0x000B
INT2	EX2	IE2	自动清 0	5 / 0x003B
TC0	ETC0	TCF0	自动清 0	6 / 0x0013
T0	ET0	TF0	自动清 0	7 / 0x0093
TC1	ETC1	TCF1	自动清 0	8 / 0x001B
INT3	EX3	IE2	自动清 0	9 / 0x007B
TC2	ETC2	TCF2	自动清 0	10 / 0x009B
UART0 Rx	ETS0	RI0	软件清 0	11 / 0x0023
SPI	ESPI	SPIF	自动清 0	12 / 0x00A3
UART0 Tx	ERS0	TI0	软件清 0	13 / 0x0063
ADC	EADC	ADCF	软件清 0	14 / 0x002B
LVD	ELVD	LVDF	软件清 0	15 / 0x00BB
I2C	EI2C	SI	软件清 0	16 / 0x00AB
BTR	EBTR	TRIGFLAG	软件清 0	17/ 0x006B

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（0x03-0xAB），系统执行相对应的中断服务程序 ISR。

9.2 中断优先级

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。同组的中断源共享同样的优先级别，对于同样的优先级别，优先权的规则按照默认的优先权。

Priority Level	IP1.x	IP0.x
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	-	UART1-RX	-	UART1-TX
Group 1	INT1	INT2	-	-	-
Group 2	TC0	-	T0	-	-
Group 3	TC1	INT3	TC2	-	-
Group 4	RX	-	SPI	-	TX
Group 5	ADC	LVD	I2C	-	BTR

IP0, IP1 Registers

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

IP0 Register (0XA9)

Bit	Field	Type	Initial	Description
5..0	IP0[5:0]	R/W	0	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 Register (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.

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

***例：** 优先级 GP5>GP4>GP3>GP2=GP1=GP0.

```
IP0 = 0x28;
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	EX3	EX2	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	ETS1	ERS1	ESPI	EI2C
IEN2	ELVD	IE3	IE2	EADC	ETC2	ETC1	ETC0	EBTR
IRCON	ADCF	TCF2	TCF1	TCF0	-	TF0	IE1	IE0
S0CON	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
I2CCON	CR2	ENS1	STA	STO	SI	AA	CR1	CR0

IEN0 Register (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。
6	EX3	R/W	0	外部中断 P0.3（INT3）中断控制位。 0：禁止； 1：使能。
5	EX2	R/W	0	外部中断 P0.2（INT2）中断控制位。 0：禁止； 1：使能。
4	ETS0	R/W	0	UART0 TX 中断控制位。 0：禁止； 1：使能。
3	ERS0	R/W	0	UART0 RX 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 Register (0XB8)

Bit	Field	Type	Initial	Description
4	ETS1	R/W	0	UART1 TX 中断控制位。 0：禁止； 1：使能。
3	ERS1	R/W	0	UART1 RX 中断控制位。 0：禁止； 1：使能。
1	ESPI	R/W	0	SPI 中断控制位。 0：禁止； 1：使能。
0	EI2C	R/W	0	I2C 中断控制位。 0：禁止； 1：使能。
Else	Reserved	R	0	

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
7	ELVD	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
6	IE3	R/W	0	外部中断 P0.3（INT3）请求标志位。 0：无 INT3 断请求； 1：请求 INT3 中断。
5	IE2	R/W	0	外部中断 P0.2（INT2）请求标志位。 0：无 INT2 断请求； 1：请求 INT2 中断。
4	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
3	ETC2	R/W	0	TC2 溢出中断控制位。 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。
0	EBTR	R/W	0	血液检查触发中断控制位。 0：禁止； 1：使能。

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。
6	TCF2	R/W	0	TC2 中断请求标志位。 0：无 TC2 中断请求； 1：请求 TC2 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 Register (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 Register (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，移位元和规范化操作，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。移位元操作需要 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

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。

```
#include <SN8F5959.H>
#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 Registers (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 Register (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 规范

运算器不允许重入代码，不能在多线程，也不能同时在主程序和中断程序中使用。

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 <SN8F5959.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

SN8F5950 共有 40 个 GPIO 引脚，和 8051 只有开路输出不同，SN8F5950 还内置推挽输出结构，以增强其驱动能力。

SN8F5950 系列必须设置所有 40 个 IO 的状态，即使是低引脚数封装的那些未封出的 IO。如果设置为输入禁止上拉模式，这些引脚必须外接固定电压，否则 IO 位悬浮态造成额外的功耗。

11.1 输入输出控制

P0M-P5M 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	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
P3M	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
P1OC	-	P23OC	P22OC	P05OC	P04OC	P17OC	P16OC	P15OC

注： P40 不能设置为输出模式

P0M Register (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 Register (0xFA)

Bit	Field	Type	Initial	Description
7	P17M	R/W	0	P1.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P16M	R/W	0	P1.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P15M	R/W	0	P1.5 的模式选择控制位。 0：输入模式； 1：输出模式。
4	P14M	R/W	0	P1.4 的模式选择控制位。 0：输入模式； 1：输出模式。
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	P02M	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。

P2M Register (0xFB)

Bit	Field	Type	Initial	Description
7	P27M	R/W	0	P2.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P26M	R/W	0	P2.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P25M	R/W	0	P2.5 的模式选择控制位。 0：输入模式； 1：输出模式。

4	P24M	R/W	0	P2.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P23M	R/W	0	P2.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P22M	R/W	0	P2.2 的模式选择控制位。 0：输入模式； 1：输出模式。
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。

P3M Register (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。

P4M Register (0xFD)

Bit	Field	Type	Initial	Description
7	P47M	R/W	0	P4.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P46M	R/W	0	P4.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P45M	R/W	0	P4.5 的模式选择控制位。 0：输入模式； 1：输出模式。
4	P44M	R/W	0	P4.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P43M	R/W	0	P4.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P42M	R/W	0	P4.2 的模式选择控制位。 0：输入模式； 1：输出模式。
1	P41M	R/W	0	P4.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P40M	R/W	0	P4.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P5M Register (0xFE)

Bit	Field	Type	Initial	Description
7	P57M	R/W	0	P5.7 的模式选择控制位。 0：输入模式； 1：输出模式。
6	P56M	R/W	0	P5.6 的模式选择控制位。 0：输入模式； 1：输出模式。
5	P55M	R/W	0	P5.5 的模式选择控制位。 0：输入模式； 1：输出模式。
4	P54M	R/W	0	P5.4 的模式选择控制位。 0：输入模式； 1：输出模式。
3	P53M	R/W	0	P5.3 的模式选择控制位。 0：输入模式； 1：输出模式。
2	P52M	R/W	0	P5.2 的模式选择控制位。 0：输入模式； 1：输出模式。

1	P51M	R/W	0	P5.1 的模式选择控制位。 0：输入模式； 1：输出模式。
0	P50M	R/W	0	P5.0 的模式选择控制位。 0：输入模式； 1：输出模式。

P1OC Register (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。
2	P17OC	R/W	0	P1.7 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。
1	P16OC	R/W	0	P1.6 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。
0	P15OC	R/W	0	P1.5 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。

11.2 输入数据和输出数据

当从 P0~P5 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~P5 寄存器的值将被马上锁存，然而，只有在 P0M ~P5M 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~P5 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	P17	P16	P15	P14	P13	P12	P11	P10
P2	P27	P26	P25	P24	P23	P22	P21	P20
P3	P37	P36	P35	P34	P33	P32	P31	P30
P4	P47	P46	P45	P44	P43	P42	P41	P40
P5	P57	P56	P55	P54	P53	P52	P51	P50

P0 Register (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（P04M=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（P03M=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（P02M=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（P01M=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（P00M=1 时使能）

P1 Register (0x90)

Bit	Field	Type	Initial	Description
7	P17	R/W	1	读：P1.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P17M=1 时使能）
6	P16	R/W	1	读：P1.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P16M=1 时使能）
5	P15	R/W	1	读：P1.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P15M=1 时使能）
4	P14	R/W	1	读：P1.4 为逻辑低电平 写入 1/0：输出高电平/低电平（P14M=1 时使能）
3	P13	R/W	1	读：P1.3 为逻辑低电平 写入 1/0：输出高电平/低电平（P13M=1 时使能）
2	P12	R/W	1	读：P1.2 为逻辑低电平 写入 1/0：输出高电平/低电平（P12M=1 时使能）
1	P11	R/W	1	读：P1.1 为逻辑低电平 写入 1/0：输出高电平/低电平（P11M=1 时使能）
0	P10	R/W	1	读：P1.0 为逻辑低电平 写入 1/0：输出高电平/低电平（P10M=1 时使能）

P2 Register (0xA0)

Bit	Field	Type	Initial	Description
7	P27	R/W	1	读：P2.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P27M=1 时使能）
6	P26	R/W	1	读：P2.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P26M=1 时使能）
5	P25	R/W	1	读：P2.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P25M=1 时使能）
4	P24	R/W	1	读：P2.4 为逻辑低电平 写入 1/0：输出高电平/低电平（P24M=1 时使能）
3	P23	R/W	1	读：P2.3 为逻辑低电平 写入 1/0：输出高电平/低电平（P23M=1 时使能）
2	P22	R/W	1	读：P2.2 为逻辑低电平 写入 1/0：输出高电平/低电平（P22M=1 时使能）
1	P21	R/W	1	读：P2.1 为逻辑低电平 写入 1/0：输出高电平/低电平（P21M=1 时使能）
0	P20	R/W	1	读：P2.0 为逻辑低电平 写入 1/0：输出高电平/低电平（P20M=1 时使能）

P3 Register (0xFC)

Bit	Field	Type	Initial	Description
7	P37	R/W	1	读：P3.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P37M=1 时使能）
6	P36	R/W	1	读：P3.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P36M=1 时使能）
5	P35	R/W	1	读：P3.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P35M=1 时使能）
4	P34	R/W	1	读：P3.4 为逻辑低电平 写入 1/0：输出高电平/低电平（P34M=1 时使能）
3	P33	R/W	1	读：P3.3 为逻辑低电平 写入 1/0：输出高电平/低电平（P33M=1 时使能）
2	P32	R/W	1	读：P3.2 为逻辑低电平 写入 1/0：输出高电平/低电平（P32M=1 时使能）
1	P31	R/W	1	读：P3.1 为逻辑低电平 写入 1/0：输出高电平/低电平（P31M=1 时使能）
0	P30	R/W	1	读：P3.0 为逻辑低电平 写入 1/0：输出高电平/低电平（P30M=1 时使能）

P4 Register (0xE8)

Bit	Field	Type	Initial	Description
7	P47	R/W	1	读：P4.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P47M=1 时使能）
6	P46	R/W	1	读：P4.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P46M=1 时使能）
5	P45	R/W	1	读：P4.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P45M=1 时使能）
4	P44	R/W	1	读：P4.4 为逻辑低电平 写入 1/0：输出高电平/低电平（P44M=1 时使能）
3	P43	R/W	1	读：P4.3 为逻辑低电平 写入 1/0：输出高电平/低电平（P43M=1 时使能）
2	P42	R/W	1	读：P4.2 为逻辑低电平 写入 1/0：输出高电平/低电平（P42M=1 时使能）
1	P41	R/W	1	读：P4.1 为逻辑低电平 写入 1/0：输出高电平/低电平（P41M=1 时使能）
0	P40	R/W	1	读：P4.0 为逻辑低电平 写入 1/0：输出高电平/低电平（P40M=1 时使能）

P5 Register (0xF8)

Bit	Field	Type	Initial	Description
7	P57	R/W	1	读：P5.7 为逻辑低电平 写入 1/0：输出高电平/低电平（P57M=1 时使能）
6	P56	R/W	1	读：P5.6 为逻辑低电平 写入 1/0：输出高电平/低电平（P56M=1 时使能）
5	P55	R/W	1	读：P5.5 为逻辑低电平 写入 1/0：输出高电平/低电平（P55M=1 时使能）
4	P54	R/W	1	读：P5.4 为逻辑低电平 写入 1/0：输出高电平/低电平（P54M=1 时使能）
3	P53	R/W	1	读：P5.3 为逻辑低电平 写入 1/0：输出高电平/低电平（P53M=1 时使能）
2	P52	R/W	1	读：P5.2 为逻辑低电平 写入 1/0：输出高电平/低电平（P52M=1 时使能）
1	P51	R/W	1	读：P5.1 为逻辑低电平 写入 1/0：输出高电平/低电平（P51M=1 时使能）
0	P50	R/W	1	读：P5.0 为逻辑低电平 写入 1/0：输出高电平/低电平（P50M=1 时使能）

11.3 内置上拉寄存器

P0UR-P5UR 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	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR
P2UR	P27UR	P26UR	P25UR	P24UR	P23UR	P22UR	P21UR	P20UR
P3UR	P37UR	P36UR	P35UR	P34UR	P33UR	P32UR	P31UR	P30UR
P4UR	P47UR	P46UR	P45UR	P44UR	P43UR	P42UR	P41UR	P40UR
P5UR	P57UR	P56UR	P55UR	P54UR	P53UR	P52UR	P51UR	P50UR

P0UR Register (0xF1)

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	P04UR	R/W	0	P0.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P03UR	R/W	0	P0.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P02UR	R/W	0	P0.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P01UR	R/W	0	P0.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P00UR	R/W	0	P0.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

P1UR Register (0xF2)

Bit	Field	Type	Initial	Description
7	P17UR	R/W	0	P1.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P16UR	R/W	0	P1.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P15UR	R/W	0	P1.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P14UR	R/W	0	P1.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P13UR	R/W	0	P1.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P12UR	R/W	0	P1.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P11UR	R/W	0	P1.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P10UR	R/W	0	P1.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

P2UR Register (0xF3)

Bit	Field	Type	Initial	Description
7	P27UR	R/W	0	P2.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P26UR	R/W	0	P2.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P25UR	R/W	0	P2.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P24UR	R/W	0	P2.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P23UR	R/W	0	P2.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P22UR	R/W	0	P2.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P21UR	R/W	0	P2.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P20UR	R/W	0	P2.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

P3UR Register (0xF4)

Bit	Field	Type	Initial	Description
7	P37UR	R/W	0	P3.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P36UR	R/W	0	P3.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P35UR	R/W	0	P3.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P34UR	R/W	0	P3.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P33UR	R/W	0	P3.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P32UR	R/W	0	P3.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P31UR	R/W	0	P3.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P30UR	R/W	0	P3.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

P4UR Register (0xF5)

Bit	Field	Type	Initial	Description
7	P47UR	R/W	0	P4.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P46UR	R/W	0	P4.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P45UR	R/W	0	P4.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P44UR	R/W	0	P4.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P43UR	R/W	0	P4.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P42UR	R/W	0	P4.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P41UR	R/W	0	P4.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P40UR	R/W	0	P4.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 如果引脚为输出模式或者模拟功能，建议禁止上拉电阻。

P5UR Register (0xF6)

Bit	Field	Type	Initial	Description
7	P57UR	R/W	0	P5.7 的内置上拉电阻控制位。 0：禁止*； 1：使能。
6	P56UR	R/W	0	P5.6 的内置上拉电阻控制位。 0：禁止*； 1：使能。
5	P55UR	R/W	0	P5.5 的内置上拉电阻控制位。 0：禁止*； 1：使能。
4	P54UR	R/W	0	P5.4 的内置上拉电阻控制位。 0：禁止*； 1：使能。
3	P53UR	R/W	0	P5.3 的内置上拉电阻控制位。 0：禁止*； 1：使能。
2	P52UR	R/W	0	P5.2 的内置上拉电阻控制位。 0：禁止*； 1：使能。
1	P51UR	R/W	0	P5.1 的内置上拉电阻控制位。 0：禁止*； 1：使能。
0	P50UR	R/W	0	P5.0 的内置上拉电阻控制位。 0：禁止*； 1：使能。

* 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 功能共享的引脚

SN8F5950 内置 LCD 功能，LCD 驱动输出引脚与 GPIO 共享，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
P1CON	P1CON7	P1CON6	P1CON5	P1CON4	-	-	-	-
P2CON	P2CON7	P2CON6	P2CON5	P2CON4	P2CON3	P2CON2	-	-
P3CON	P3CON7	P3CON6	P3CON5	P3CON4	P3CON3	P3CON2	P3CON1	P3CON0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
P5CON	P5CON7	P5CON6	P5CON5	P5CON4	P5CON3	P5CON2	P5CON1	P5CON0

P3CON Register（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	EX3G1	EX3G0	EX2G1	EX2G0	EX1G1	EX1G0	EX0G1	EX0G0
IEN0	EAL	-	-	ETS0	ERS0	EX1	ET0	EX0
IRCON	ADCF	TCF2	TCF1	TCF0	-	TF0	IE1	IE0

PEDGE Register (0X8F)

Bit	Field	Type	Initial	Description
3..2	EX3G[1:0]	R/W	10	外部中断 INT3 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	EX2G[1:0]	R/W	10	外部中断 INT2 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。
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。

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 <SN8F5959.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_();
}
```

```

void INT1_ISR(void) interrupt ISRInt1   // Vector @  0x0B
{
    // cleared int1 flag by hardware
    // IE1 = 0;        // Clear INT1 flag

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

void Delay1ms(unsigned int n)
{
    unsigned int i, j;
    // int value
    i = 0;
    j = 0;

    for (i=0; i<n; i++) {
     for (j=0; j<220; j++) {
          _nop_();    _nop_();
          _nop_();    _nop_();
          _nop_();    _nop_();
          _nop_();    _nop_();
     }
  }
}



```

13 TCx 16 位定时器/计数器

16 位二进制定时器 TC0/TC1/TC2 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/周期可程序设计控制的 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：**
  事件计数器对外部时钟计数。
- **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。

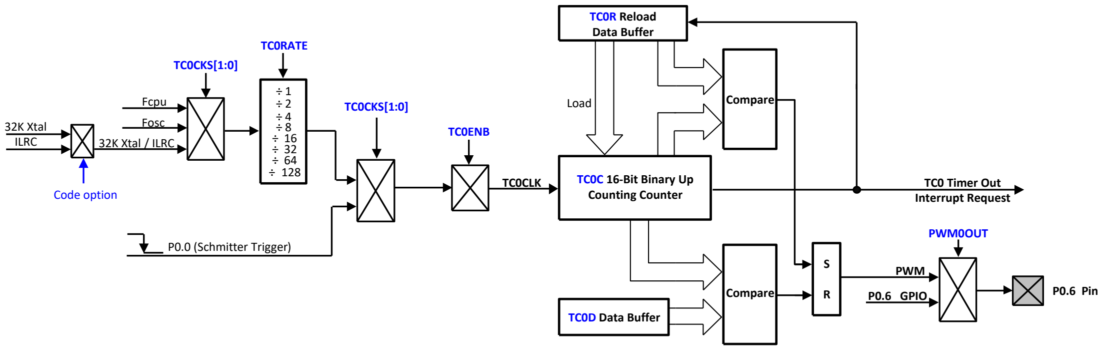
注：

寄存器	功能说明
TCx = TC0, TC1	定时器计数器
TCxC = TC0C, TC1C	定时器计数器记录数据
TCxD = TC0D, TC1D	定时器 PWM 功能占空比数据
TCxR = TC0R, TC1R	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	

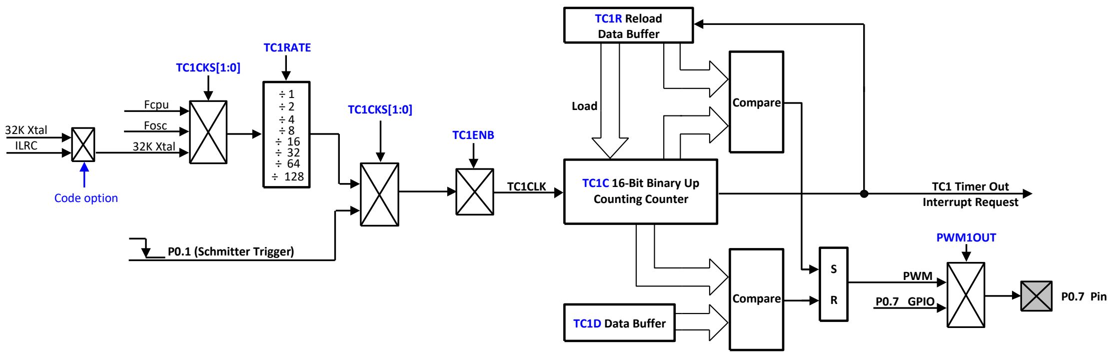
13.1 定时器计数器框图和时钟源选择

下图是 TC0、TC1 和 TC2 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 TCx 内部结构框图。

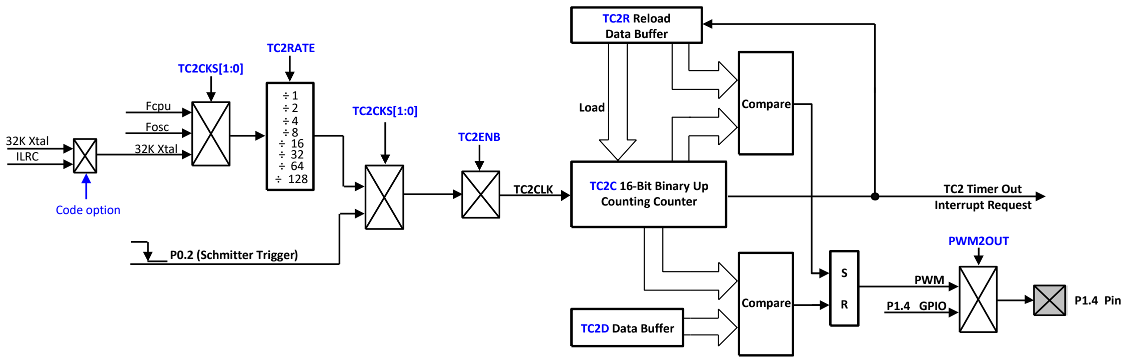
TC0 结构框图：



TC1 结构框图：



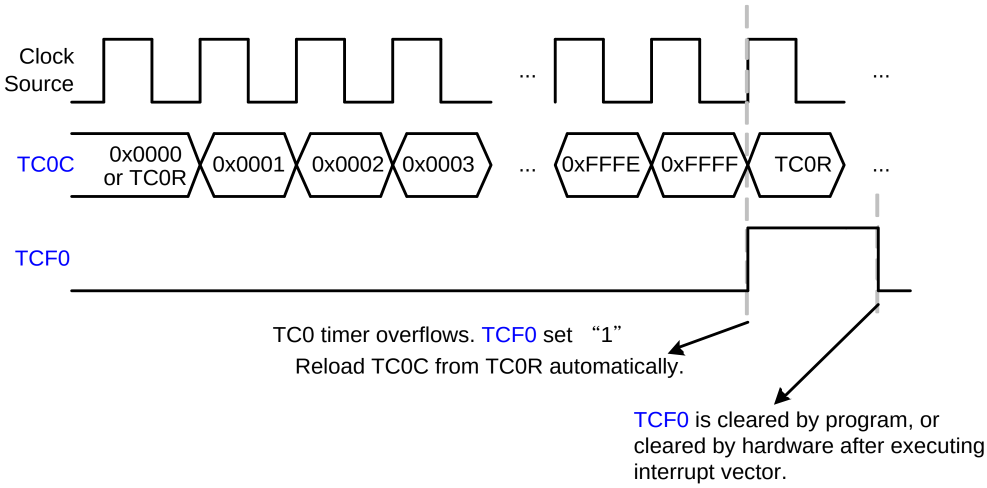
TC2 结构框图：



13.2 TC0 定时器操作 (包含 TC1/TC2)

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。

TC0 间隔时间 = (65536-TC0R) * 1/(F$_{TC0\ 时钟源}$/TC0Rate)



13.3 TC0 计数功能和寄存器(包括 TC1/TC2)

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 Register (TC0CH : 0xC5)

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

TC0CL Register (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 中断间隔时间 * TC0 时钟 rate)

13.4 TC0 自动重装功能和寄存器 (包括 TC1/TC2)

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 Register (TC0RH : 0xC3)

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

TC0RL Register (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 中断间隔时间 * TC0 时钟 rate)

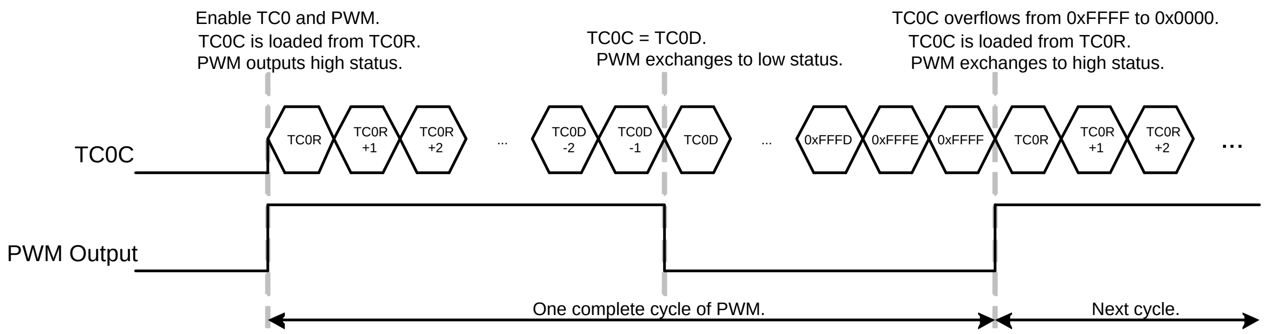
例：计算 TC0C 和 TC0R 的值，TC0 的间隔时间为 10ms。

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/TC2) PWM 功能

程序设计控制占空比/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)

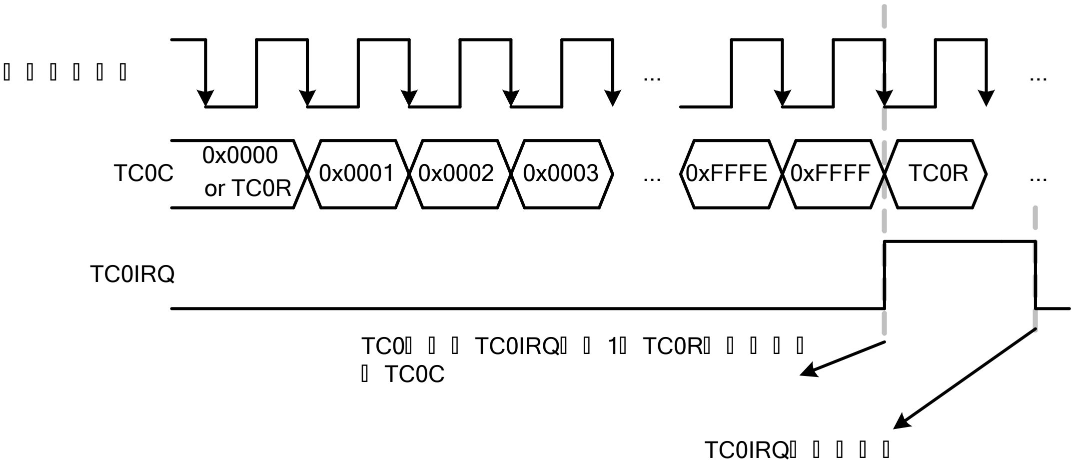
例：计算 TC0D 的值，PWM 信号为 100Hz，1/3 占空比。

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/TC2)

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
TC2 外部输入信号 = P02

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	TC0RATE1	TC0RATE0	TC0CKS1	TC0CKS0	PWM0OUT	-
TC0RL	TC0RL7	TC0RL6	TC0RL5	TC0RL4	TC0RL3	TC0RL2	TC0RL1	TC0RL0
TC0RH	TC0RH7	TC0RH6	TC0RH5	TC0RH4	TC0RH3	TC0RH2	TC0RH1	TC0RH0
TC0CL	TC0CL7	TC0CL6	TC0CL5	TC0CL4	TC0CL3	TC0CL2	TC0CL1	TC0CL0
TC0CH	TC0CH7	TC0CH6	TC0CH5	TC0CH4	TC0CH3	TC0CH2	TC0CH1	TC0CH0
TC0DL	TC0DL7	TC0DL6	TC0DL5	TC0DL4	TC0DL3	TC0DL2	TC0DL1	TC0DL0
TC0DH	TC0DH7	TC0DH6	TC0DH5	TC0DH4	TC0DH3	TC0DH2	TC0DH1	TC0DH0
TC1M	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS1	TC1CKS0	PWM1OUT	-
TC1RL	TC1RL7	TC1RL6	TC1RL5	TC1RL4	TC1RL3	TC1RL2	TC1RL1	TC1RL0
TC1RH	TC1RH7	TC1RH6	TC1RH5	TC1RH4	TC1RH3	TC1RH2	TC1RH1	TC1RH0
TC1CL	TC1CL7	TC1CL6	TC1CL5	TC1CL4	TC1CL3	TC1CL2	TC1CL1	TC1CL0
TC1CH	TC1CH7	TC1CH6	TC1CH5	TC1CH4	TC1CH3	TC1CH2	TC1CH1	TC1CH0
TC1DL	TC1DL7	TC1DL6	TC1DL5	TC1DL4	TC1DL3	TC1DL2	TC1DL1	TC1DL0
TC1DH	TC1DH7	TC1DH6	TC1DH5	TC1DH4	TC1DH3	TC1DH2	TC1DH1	TC1DH0
TC2M	TC2ENB	TC2RATE2	TC2RATE1	TC2RATE0	TC2CKS1	TC2CKS0	PWM2OUT	-
TC2RL	TC2RL7	TC2RL6	TC2RL5	TC2RL4	TC2RL3	TC2RL2	TC2RL1	TC2RL0
TC2RH	TC2RH7	TC2RH6	TC2RH5	TC2RH4	TC2RH3	TC2RH2	TC2RH1	TC2RH0
TC2CL	TC2CL7	TC2CL6	TC2CL5	TC2CL4	TC2CL3	TC2CL2	TC2CL1	TC2CL0
TC2CH	TC2CH7	TC2CH6	TC2CH5	TC2CH4	TC2CH3	TC2CH2	TC2CH1	TC2CH0
TC2DL	TC2DL7	TC2DL6	TC2DL5	TC2DL4	TC2DL3	TC2DL2	TC2DL1	TC2DL0
TC2DH	TC2DH7	TC2DH6	TC2DH5	TC2DH4	TC2DH3	TC2DH2	TC2DH1	TC2DH0

TC0M Register (0xC1)

Bit	Field	Type	Initial	Description
7	TC0ENB	R/W	0	TC0 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； 001：clock/64； 010：clock/32； 011：clock/16； 100：clock/8； 101：clock/4； 110：clock/2 ； 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	TC0GN	R/W	0	TC0 定时器 STOP 模式唤醒控制位。 0：使能； 1 ：禁止。

TC1M Register (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； 001：clock/64； 010：clock/32； 011：clock/16； 100：clock/8； 101：clock/4； 110：clock/2 ； 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 事件计数器功能
	001	32KHz / 64			
	010	32KHz / 32			
	011	32KHz / 16			
	100	32KHz / 8			
	101	32KHz / 4			
	110	32KHz / 2			
	111	32KHz / 1			

Note: TCx = TC0, TC1

IEN2 Register (0X9A)

Bit	Field	Type	Initial	Description
4	EADC	R/W	0	ADC 中断控制位。 0：禁止； 1：使能。
3	ETC2	R/W	0	TC2 溢出中断控制位。 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 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。
6	TCF2	R/W	0	TC2 中断请求标志位。 0：无 TC2 中断请求； 1：请求 TC2 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 / TC2CH Registers (TC0CH : 0xC5, TC1CH : 0xCD, TC2CH : 0xA5)

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

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

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

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

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

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

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

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

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

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

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

下面的示例代码程序演示了如何执行 TC0TC1 中断。

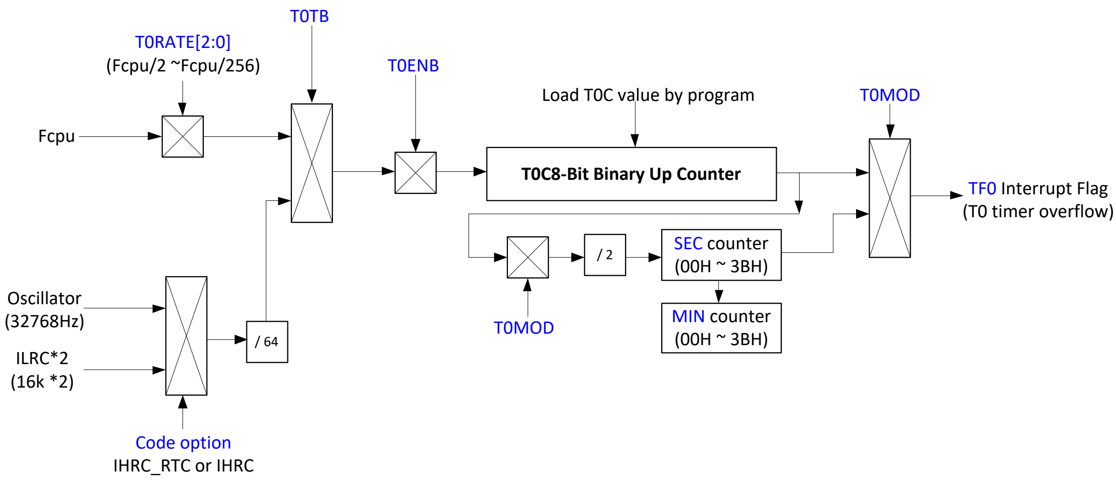
```
#include <intrins.h>    // for _nop_
#include <SN8F5959.h>

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

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

}

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

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

        // To Do...
  }
}

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

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

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

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

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

```
void TC0_ISR(void) interrupt ISRTC0 // Vector @  0x13
{
    // cleared TC0 interrupt flag by hardware
    // TCF0 = 0;          // Clear TCF0 flag

    // To Do...
    P0 ^= 0XFF;           // P0 Toggle
}

```

14 定时器 T0

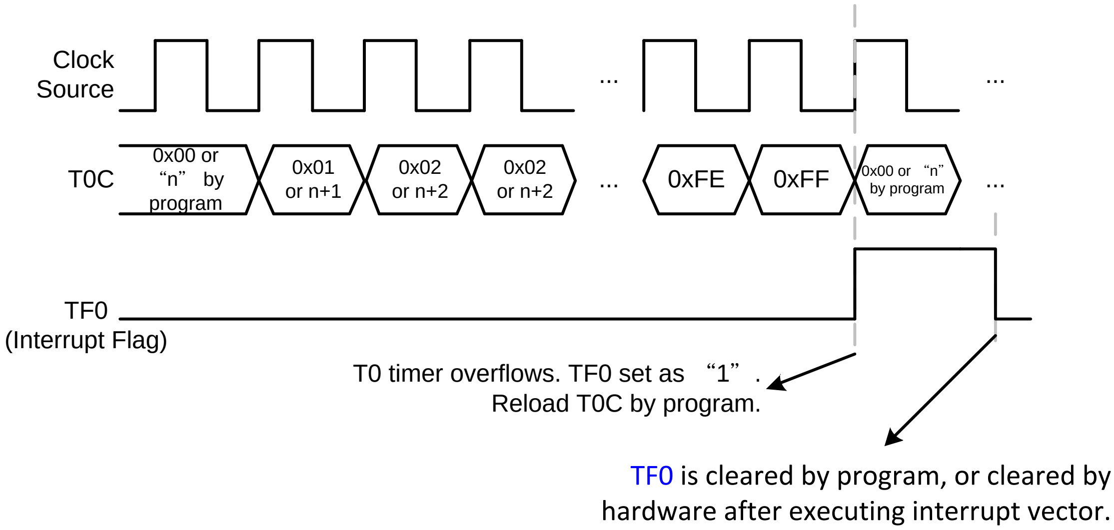
8 位二进制基本定时器 T0，支持标志指示（TF0）和中断操作（中断向量 0x93）。可以通过 T0M 和 T0C 寄存器控制间隔时间，支持 RTC 功能。T0 内置 STOP 模式唤醒功能，STOP 模式下，T0 溢出时，系统被唤醒进入 Normal 模式。

- **8 位可程序设计计数定时器**：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 @IHRC_RTC 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

TT0 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，中断过程是系统程序计数器指向中断向量（ORG 0093H）并在 T0 溢出事件发生后执行中断服务例程，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 Clock	High Speed Mode (Fcpu=1 MIP)	
		Max overflow Time	One Step = max/256
000	Fcpu / 256	65.563 ms	256 us
001	Fcpu / 128	32.768 ms	128 us
010	Fcpu / 64	16.384 ms	64 us
011	Fcpu / 32	8.192 ms	32 us
100	Fcpu / 16	4.096 ms	16 us
101	Fcpu / 8	2.048 ms	8 us
110	Fcpu / 4	1.024 ms	4 us
111	Fcpu / 2	0.512 ms	2 us

14.2 T0 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 Register (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； 001：Fcpu/128； 010：Fcpu/64； 011：Fcpu/32； 100：Fcpu/16； 101：Fcpu/8； 110：Fcpu/4； 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 计数控制器

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 Register (0xBC)

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

T0C 初始值计算公式如下：

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

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）
= 0xB2

14.4 其它相关寄存器

SEC Register (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。

MIN Register (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。

IEN0 Register (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 Register (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 <SN8F5959.h>

void T0_Init(void);

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

}

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

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

         // To Do...
    }
}

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

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

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

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

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 <SN8F5959.h>

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

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

}

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

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

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

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

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

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

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

    P0 ^= 0XFF;           // P0 Toggle
}

```

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[2:1]寄存器控制。

BZRM Register (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 功能示例代码

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

16 LCD 驱动

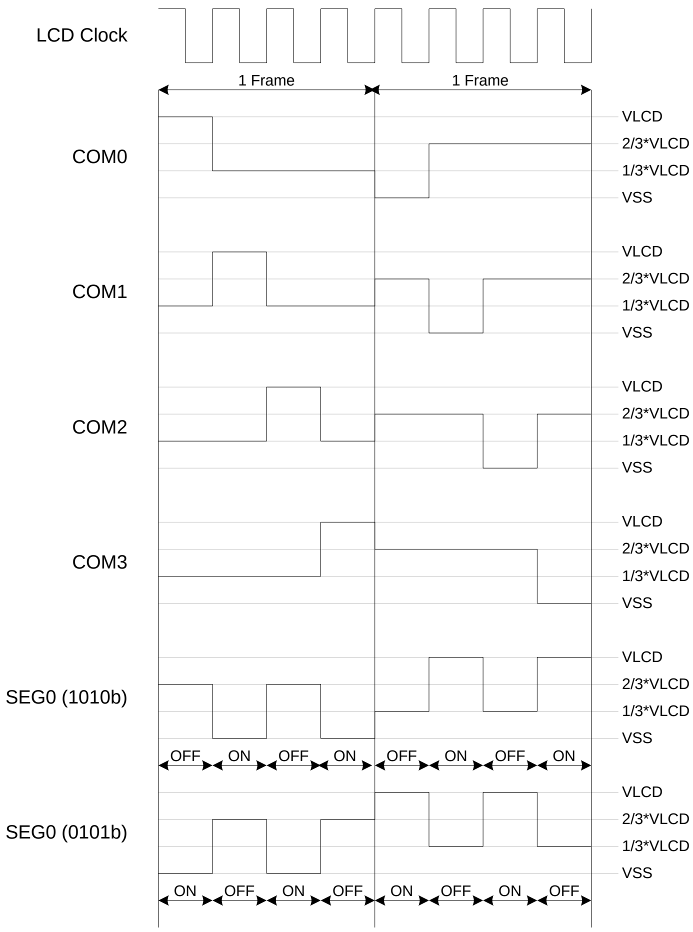
SN8F5950 LCD 驱动为 C 型结构，具体为 4 x 36 或 6 x 34，LCD 的扫墓时序为 1/4 或 1/6 占空比，1/3 偏压，具有 144 点或 204 点 LCD 驱动。C 型 LCD 驱动可输出可调节的 VLCD 输出电压和可调节的驱动电源以支持 LCD 面板的多样性。VLCD 引脚与 DVSS 直接必须连接一个 0.1uF 的电容用于 C 型 LCD 驱动。

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。

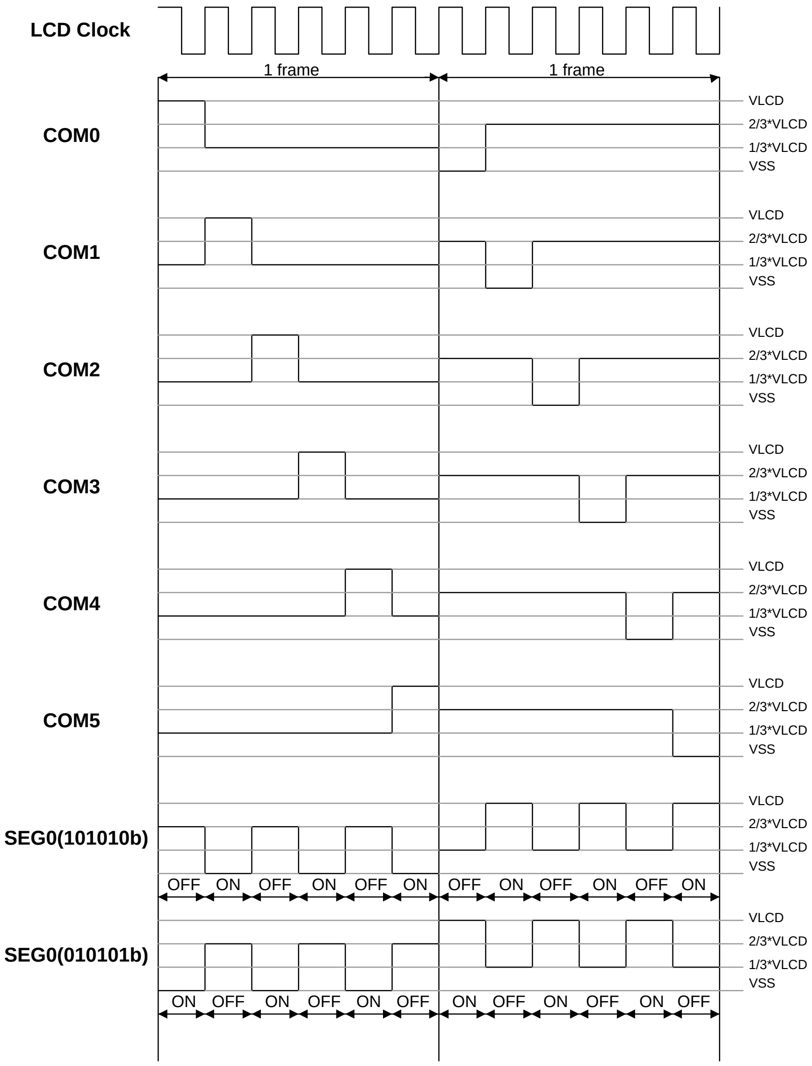
- **Stop 模式功能：**STOP 模式下 LCDEN=1&STOP=1 时，LCD 功能保持工作状态。
- **低功耗模式功能：**由 BGM 位控制 LCD 低功耗功能。

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

Code Option 决定 LCD 的时钟源，如下表：

CPU 时钟源	***LCD 时钟***
IHRC_RTC	外部 32K 振荡器
IHRC	内部 32K RC

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

16.1 LCD RAM 位置

LCD RAM 位于地址 0xF000 到 0xF023 之间。

LCD RAM 地址与 COM0 ~ COM3 vs. SEG0 ~ SEG23 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	F023H .1	F023H.2	F023H.3	N/A	N/A	N/A	N/A

LCD RAM 地址与 COM0 ~ COM5 vs. SEG2 ~ SEG23 LCD 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	F023H.0	F023H.1	F023H.2	F023H.3	F023H.4	F023H.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	RCMOD	BGM	DUTY1	DUTY0	VCP3	VCP2	VCP 1	VCP0
LCDM3	-	-	-	-	-	DUTYB1	DUTYB0	-

LCDM1 Register (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：1.必须先使能 LCD Pump（LCDPEN=1）并等待 1ms，然后使能 LCD 驱动（ LCDEN=1）

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

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

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

LCDM2 Register (0xB2)

Bit	Field	Type	Initial	Description

7	RCMOD	R/W	1	RCMOD。 0：C-Type LCD； 1：R-Type LCD。
6	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。
5..4	DUTY[1:0]	R/W	0	LCD pump 输出驱动能力控制位。 00：25%； 01：37.5%； 10：50%； 11：100%。 *注：BGM=0 时，DUTY [1:0]可程序设计控制用于 LCD 低功耗模式。 建议 100% Duty 。
3..0	VCP[3:0]	R/W	1000	VLCD 输出电压控制位。 0000：2.6V；　1000：3.4V； 0001：2.7V；　1001：3.5V； 0010：2.8V；　1010：3.6V； 0011：2.9V；　1011：3.7V； 0100：3.0V；　1100：3.9V； 0101：3.1V；　1101：4.1V； 0110：3.2V；　1110：4.3V； 0111：3.3V；　1111：4.5V； 其它： 保留。 *注：电池电压必须大于 1.9V。 @VLCD 输出电压 2.6~4.5V。

***Example:** 3. LCD 低功耗模式。

```
LCDM2 |= 0x30;   // Driving capacity = 100%.
LCDM2 &= 0xBF;   // Enable Low power Mode.
```

LCDM3 Register (0xFFEB)

Bit	Field	Type	Initial	Description
2..1	DUTYB[1:0]	R/W	0	LCD 低功耗 b 脉冲控制位。 00：Disable； 01：0.5kHz； 10：1kHz； 11：2kHz； *注：所有应用都甚至 DUTYB[1:0]= “11”。

P1CON Register (0xFFEA)

Bit	Field	Type	Initial	Description
7..4	P1CON[7:4]	R/W	0xFF	P1 功能控制位。 0：设置 LCD 功能（SEG32 ~ SEG35）； 1：设置 GPIO 功能（P14 ~ P17）。

P2CON Register (0x9B)

Bit	Field	Type	Initial	Description
7..2	P2CON[7:4]	R/W	0xFF	P2 功能控制位。 0：设置 LCD 功能（SEG26 ~SEG31）； 1：设置 GPIO 功能（P22 ~ P27）。

P3CON Register (0x9E)

Bit	Field	Type	Initial	Description
7..0	P3CON[7:0]	R/W	0xFF	P3 功能控制位。 0：设置 LCD 功能（SEG18 ~ SEG25）； 1：设置 GPIO 功能（P30 ~ P37）。

P4CON Register (0x9F)

Bit	Field	Type	Initial	Description
7..0	P4CON[7:0]	R/W	0xFF	P4 功能控制位。 0：设置 LCD 功能（SEG10 ~ SEG17）； 1：设置 GPIO 功能（P40 ~ P47）。

P5CON Register (0x9D)

Bit	Field	Type	Initial	Description
7..0	P5CON[7:0]	R/W	0xFF	P5 功能控制位。 0：设置 LCD 功能（SEG2 ~ SEG9）； 1：设置 GPIO 功能（P50 ~ P57）。

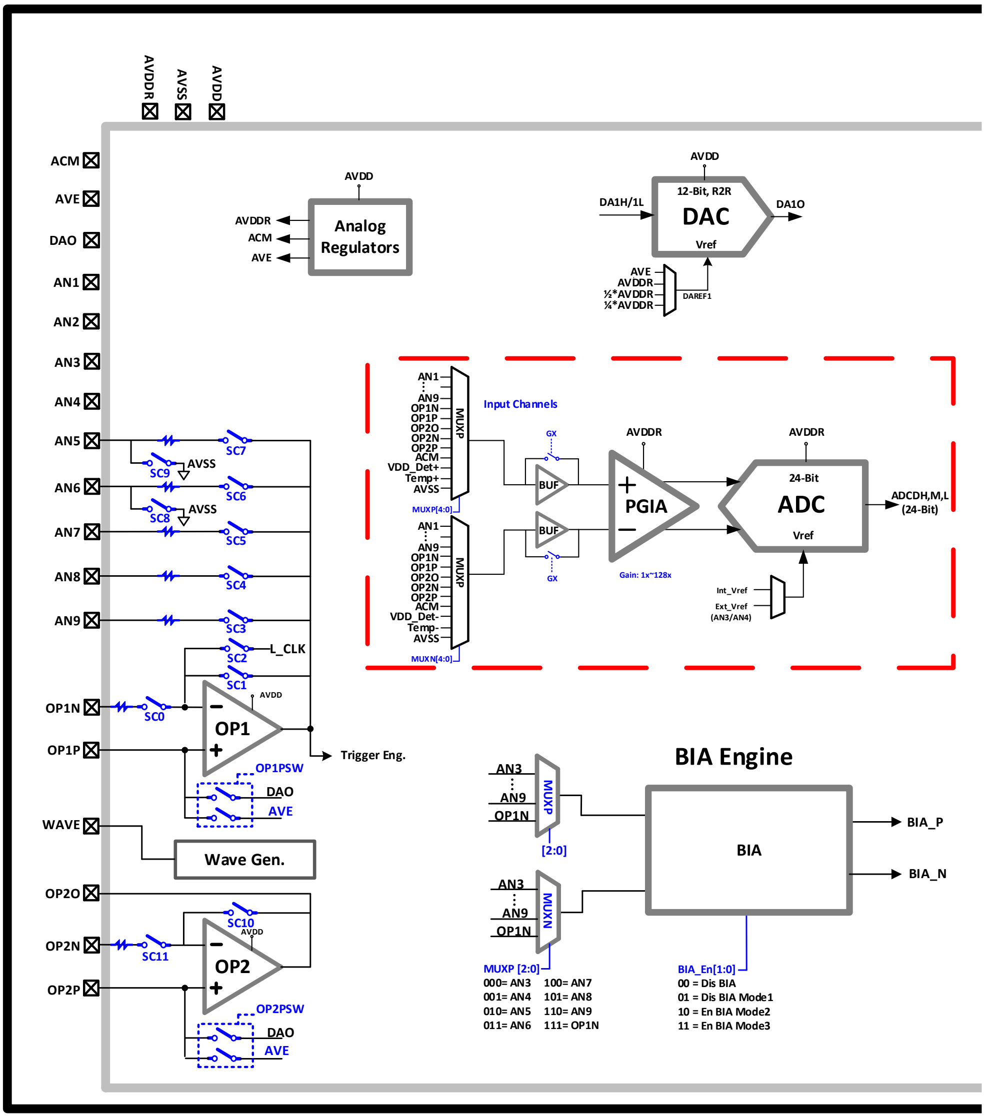
16.3 示例代码

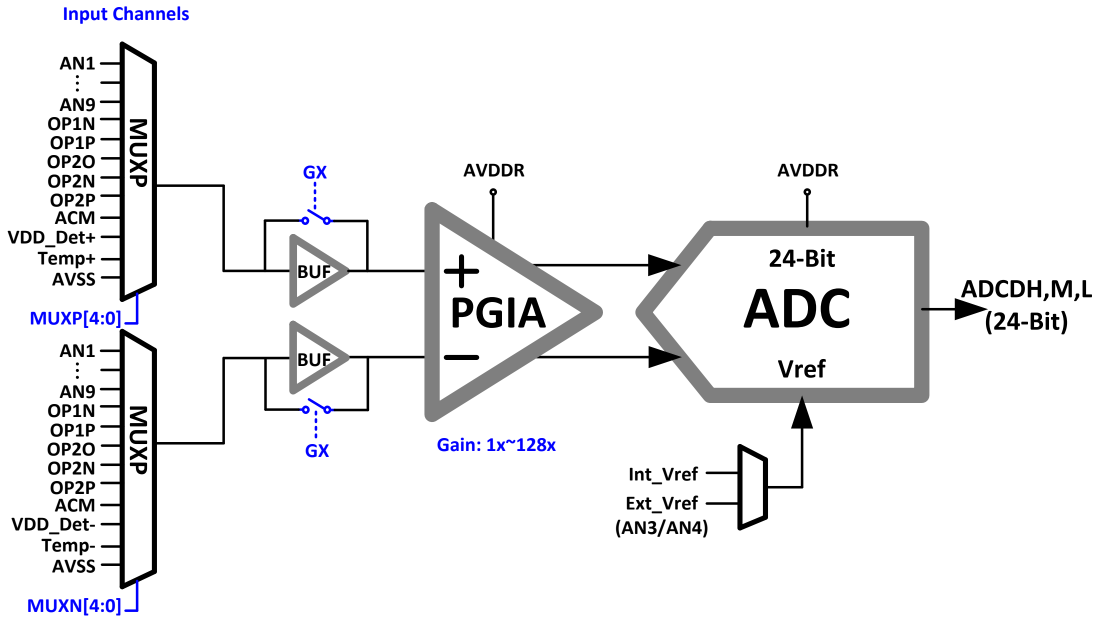
下面的示例程序展示了如何在 STOP 模式下执行 LCD 功能。

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

    LCDM2 &= 0x00;     // Clear LCDM2
    LCDM2 &= 0xBF;     // Enable Low power Mode @STOP MODE
    LCDM2 |= 0x08;     // VLCD=3.0V
    LCDM2 |= 0x30;     // Low power mode Driving capacity = 100%.

    LCDM1 |= 0x01;     // Enable LCD
}

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

17 ADC

SN8F5950 集成高分辨率 ΔΣ ADC，低噪音可程序设计增益放大器（PGIA），可输出 24 位分辨率 KHz，带 18 位精度。ADC 的转换 rate 范围为 7.6Hz 到 5.2KHz，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、高稳定性可程序设计增益放大器（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 9 个外部通道输入和 9 个内部通道输入，AI1~AI9 用于测量 9 个差分模拟信号源，由 CHS[4:0] 位控制。9 个内部通道包括 OP1P，OP1N，OP2O，OP2N，OP2P，ACM，GND，VDD_DET（VDD 检测）和温度信道。ADC 外部参考电压通过 AI3 和 AI4 输入。VDD_DET 信道为 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。

SN8F5950 包括低噪音、高稳定性可程序设计增益放大器（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。

模拟输入信号信道选择列表：

MUXP[4:0]	**PGIA 正极输入**	***MUXN[4:0]***	**PGIA 负极输入**
00000	AN1	00000	AN1
00001	AN2	00001	AN2
00010	AN3 / R+	00010	AN3 / R+
00011	AN4 / R-	00011	AN4 / R-
00100	AN5	00100	AN5
00101	AN6	00101	AN6
00110	AN7	00110	AN7
00111	AN8	00111	AN8
01000	AN9	01000	AN9
01001	OP1N	01001	OP1N
01010	OP1P	01010	OP1P
01011	OP2O	01011	OP2O
01100	OP2N	01100	OP2N
01101	OP2P	01101	OP2P
01110	ACM	01110	ACM
01111	VDD_DET+	01111	VDD_DET-
10000	Temperature+	10000	Temperature-
10001	AVss	10001	AVss

17.3 模拟输入缓存

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。

RVS[3:0]		Vref source(AVE)		RVS[3:0]		Vref source(AVDDR)			
		1.5v	2.0v			2.2V	2.4V	2.8V	3.2V
00 00	0.15*AVE	0.225V	0.3V	01 00	0.15xAVDDR	0.33V	0.36V	0.42V	0.48V
00 01	0.25*AVE	0.375V	0.5V	01 01	0.25xAVDDR	0.55V	0.6V	0.7V	0.8V
00 10	0.35*AVE	0.525V	0.7V	01 10	0.35xAVDDR	0.77V	0.84V	0.98V	1.12V
00 11	0.5*AVE	0.75V	1V	01 11	0.5xAVDDR	1.1V	1.2V	1.4V	1.6V
RVS[3:0]		**Vref source(VBG)**		**RVS[3:0]**		**Vref source**			
		1.2V				**External Input**			
10 xx		0.8V		11 xx		R+ / R-			

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* and *ADCDL*：

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

$$PGIA\_Gain : x1 \sim x128$$
$$ADC\_Gain : x1 \sim x2$$
$$Vref : 0.3V \sim 1.6V$$

16 位格式，ADC 输出码位于 *ADCDH* and *ADCDM*:

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

$$PGIA\_Gain : x1 \sim x128$$
$$ADC\_Gain : x1 \sim x2$$
$$Vref : 0.3V \sim 1.6V$$

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 7.6Hz~5.2KHz，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 表：

ADCKS	OSR [2:0]	ADC clock	Conversion Rate
0	000	250KHz	3.9 kHz
0	001		1.95 kHz
0	010		976 Hz
0	011		244 Hz
0	100		61 Hz
0	101		30.5 Hz
0	110		15.2 Hz
0	111		7.6 Hz
1	000	333KHz	5.2 kHz
1	001		2.6 kHz
1	010		1.3 kHz
1	011		325 Hz
1	100		81.3 Hz
1	101		40.7 Hz
1	110		20.3 Hz
1	111		10.2 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
CHS0	-	-	-	MUXN4	MUXN3	MUXN2	MUXN1	MUXN0
CHS1	-	-	-	MUXP4	MUXP3	MUXP2	MUXP1	MUXP0
VREG	BGREN	AVDDRS1	AVDDRS0	AVDDREN	ACMSEL	ACMEN	AVESEL	AVENB
AMPM	GX	AMPCKS1	AMPCKS0	GS2	GS1	GS0	PCHPEN	PGIAEN
ADCM1	GR	IRVS3	IRVS2	IRVS1	IRVS0	ADGN	ACHPEN	ADCEN
ADCM2	OFSEL1	OFSEL0	-	ADCKS	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	ELVD	-	-	EADC	ETC2	ETC1	ETC0	EBTR
IRCON	ADCF	TCF2	TCF1	TCF0	LVDF	TF0	IE1	IE0

CHS0 Register (0xD1)

Bit	Field	Type	Initial	Description
4..0	MUXN[4:0]	R/W	0	ADC 正极输入通道选择位。 00000：AI1； 01001：OP1N； 00001：AI2； 01010：OP1P； 00010：AI3/R+； 01011：OP2O； 00011：AI4/R-； 01100：OP2N； 00100：AI5； 01001：OP2P； 00101：AI6； 01010：ACM； 00110：AI7； 01010：VDD_DET-； 00111：AI8； 01011：Temperature-。

CHS1 Register (0xE7)

Bit	Field	Type	Initial	Description
4..0	MUXP[3:0]	R/W	0	ADC 负极输入通道选择位。 00000： AI1； 01001：OP1N； 00001：AI2； 01010：OP1P； 00010：AI3/R+； 01011：OP2O； 00011：AI4/R-； 01100：OP2N； 00100：AI5； 01001：OP2P； 00101：AI6； 01010：ACM； 00110：AI7； 01010：VDD_DET+； 00111：AI8； 01011：Temperature+； 01000：AI9； 01100：AVSS。

AMPM Register (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； 01：ADCKS / 32； 10：ADCKS / 16； 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； 001：4x； 010：8x； 011：16x； 100：32x； 101：64x； 110：128x； 111： 1x + Buff 模式 (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。

```
AMPM |= 0x80;     // Enable X+/X- Unit Gain Buffer @GAIN 1x1.
AMPM &= 0xE3;     // PGIA x 1, GX Buff always set 1.
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 Register (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..3	IRVS[3:0]	R/W	000	ADC 参考电压选择位。 11xx：外部 R+/R-； 10xx：内部 0.8V 用于温度传感器； 0111：内部 0.5 x AVDDR； 0110：内部 0.35 x AVDDR； 0101：内部 0.25 x AVDDR； 0100：内部 0.15 x AVDDR； 0011：内部 0.5 x AVE； 0010：内部 0.35 x AVE； 0001：内部 0.25 x AVE； 0000：内部 0.15 x AVE。
2	ADGN	R/W	0	ADC 增益选择位。 0：1x； 1：2x。
1	ACHPEN	R/W	00	ADC chopper 控制位。 0：禁止； 1：使能。（必须在所有应用中设置为 1）
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。**

```
ADCM1 |= 0x63;    // EXT Vref
ADCM1 |= 0x60;    // Enable GR Unit Gain Buffer.@EXT Vref
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 Register (0xD5)

Bit	Field	Type	Initial	Description
7..6	OFSEL[1:0]	R/W	11	ADC 偏移选择位。 00 ： -75%； 01：-50%； 10：-25%； 11：0%。
4	ADCKS[1:0]	R/W	0	ADC 时钟选择位。 0：250KHz； 1：333KHz。 （建议设置为 250KHz）
3..1	OSR[2:0]	R/W	000	ADC Over-Sampling-Rate 选择位。 000：64； 001：128； 010：256； 011：1024； 100：4096； 101：8192； 110：16384； 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 Register (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 EN0B (Noise Free Bit)

SN8F959 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	15.3	14.8	14.3	13.9	12.8	11.6	11.3	10.6
64x1	16	15.8	15	14.7	13.5	12.7	11.9	11.2
32x1	17	16.4	15.8	15.3	14.5	13.3	13	11.5
16x1	17.7	17	16.6	14.4	15.1	14	12.5	12
8x1	17.8	17.4	16.9	16.4	15.4	14.4	12.8	12.2
4x1	18	17.6	17.1	16.7	15.5	14.5	13	12.4
1x1	18.3	18	17.4	17	16.1	14.9	13.4	12.2

* 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)

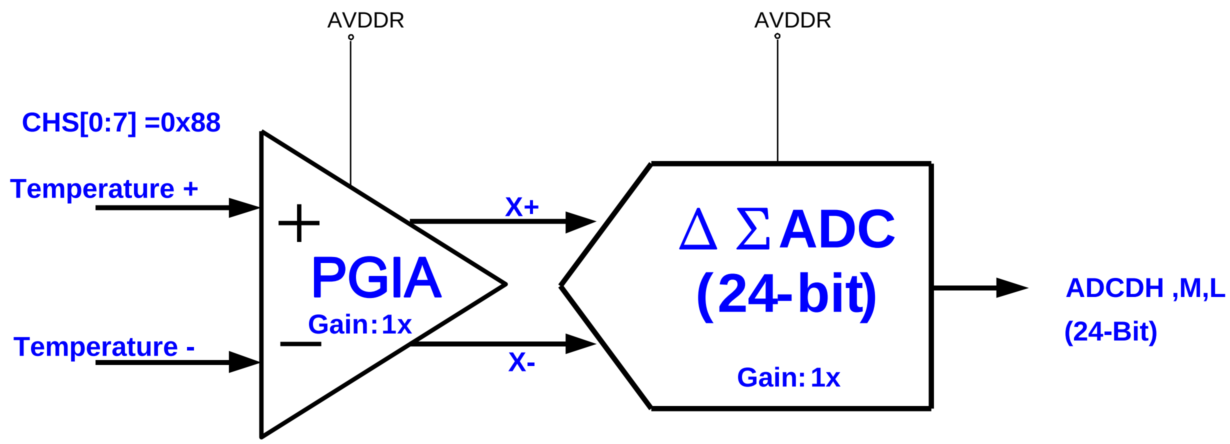
(3). RMS Noise = σ x LSB_Resolution where LSB_Resolution = Full Scale Range / 2^Bit, Bit=24

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，SN8F5959 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 信道达到测量环境温度的目的。



注：当选择温度传感器时，PGIA的增益要选择为1x，否则会出错。

注：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 10^0C，V（TS）就会下降 18Mv（VTS=VTS Offset + 18mV）；相反，若温度下降 10^0C，V（TS）则会上升 18Mv（ VTS=VTS Offset – 18mV）。

例：

Temperature	(Ts+) - (Ts-)	ADC Vref	ADC Output(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：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：温度传感器的典型温度参数是-1.7mV/C，每颗单片机不是完全相同的。（-69.5Cnt/℃）。

SYM	DESCRIPTION	PARAMETER	MIN	TYP	MAX	UNIT
TR	温度传感器范围	**AVDDR=2.4V ,VDD=3V**	**-**	**-**	**-**	℃
TS	温度传感器灵敏度	**AVDDR=2.4V ,VDD=3V**		**-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 <SN8F5959.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 &= 0xBF;      // AVDDR=2.4V

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


void Init_PGIA(void)
{
    CHS0 &= 0x00;      // Clear CHS
    CHS1 &= 0x00;      // Clear CHS
    CHS0  |= 0x00;     // MUXP channel [AI1]
    CHS1  |= 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
}
```

```

void Init_ADC(void)
{
    ADCM1 &= 0x00;     // Clear ADCM1
    ADCM1 |= 0x02;     // Enable ADC Chopper,
                              // ACHPEN always Enable
    ADCM1 &= 0xF7;     // ADC Gain=1x
    ADCM1 |= 0x30;     // ADC Vref = AVDDR*0.35

    ADCM2 &= 0x00;     // Clear ADCM2
    ADCM2 |= 0x00;     // ADC Clk=250kHz
    ADCM2 |= 0x0E;     // ADC ODR=32768

    ADCM2 &= 0xFE;     // Clear DRDY
    ADCM1 |= 0x01;     // Enable ADC
}


void ADC_Enable_ISR(void)
{
    ADCF = 0;          // Clear ADCF
    IEN2 &= 0x00;      // Clear IEN2
    IEN2 |= 0x10;      // ADC interrupt enable (EADC)
    EAL = 1;           // Interrupt enable
}


void ADC_ISR(void) interrupt ISRAdc
{
    // 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_();
    }
}

```

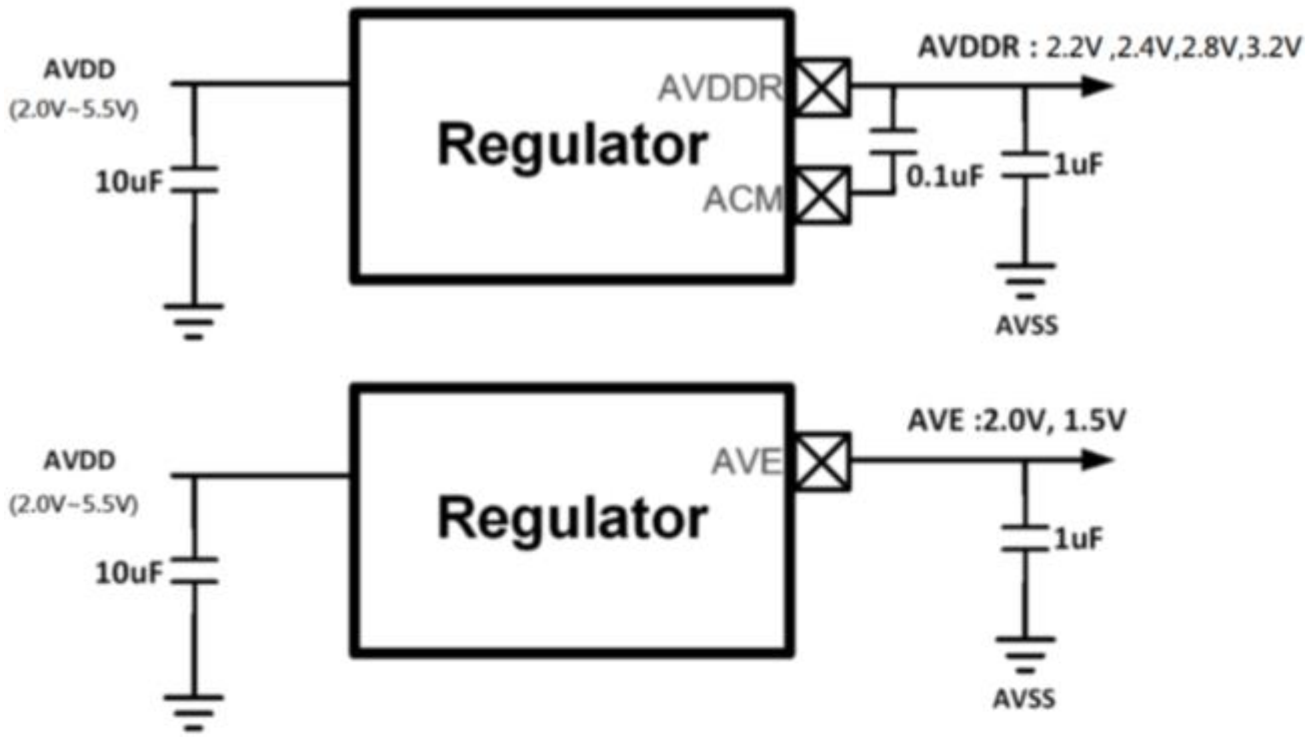
下面的示例程序展示了如何执行 ADC 温度。

```c
#include <intrins.h>    // for _nop_
#include <SN8F5959.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 &= 0xBF;      // AVDDR=2.4V

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

void Init_PGIA(void)
{
    CHS0 &= 0x00;       // Clear CHS
    CHS1 &= 0x00;       // Clear CHS
    CHS0 |= 0x10;       // MUXP channel [Temp S+]
    CHS1 |= 0x10;       // MUXP channel [Temp S-]

    ADCM1&= 0x87;       // Clear Vref
    ADCM1|= 0x40;       // Set VBG vref source
    AMPM &= 0x00;       // Clear AMPM
    AMPM |= 0x20;       // ADCLK=7.8k@ADCCLK=250K
    AMPM |= 0x02;       // Enable PGIA Chopper
    AMPM &= 0xE3;       // PGIA x 1
```

```
void Init_ADC(void)
{
    ADCM1 &= 0x00;     // Clear ADCM1
    ADCM1 |= 0x02;     // Enable ADC Chopper
    ADCM1 &= 0xFB;     // ADC Gain=1x
    ADCM1 |= 0x20;     // ADC Vref = AVDDR*0.36

    ADCM2 &= 0x00;     // Clear ADCM2
    ADCM2 |= 0x00;     // ADC Clk=250kHz
    ADCM2 |= 0x0E;     // ADC ODR=32768

    ADCM2 &= 0xFE;     // Clear DRDY
    ADCM1 |= 0x01;     // Enable ADC
}
```

18 Regulator

SN8F5950 内置 Regulator 为 ADC、PGIA 和外部传感器提供稳定的电压 AVDDR 和 AVE。可调节 AVDDR 输出电压 2.2V、2.4V、2.8V 和 3.2V，可调节 AVE 输出电压 1.5V 和 2.0V，最大驱动电流为 7.5mA。另一种 Regulator ACM 用于 ADC common 电压，仅在 sink-type 下工作。



18.1 Regulator 寄存器

VREG Register (0xD2)

Bit	Field	Type	Initial	Description
7	BGREN	R/W	0	Band Gap 使能位。 0：禁止； 1：使能。
6..5	AVDDRS[1:0]	R/W	00	AVDDR regulator 输出电压控制位。 00：2.2V； 01：2.4V； 10：2.8V； 11：3.2V。
4	AVDDREN	R/W	0	AVDDR Regulator 控制位。 0：禁止 Regulator 和 AVDDR 输出电压； 1：使能 Regulator 和 AVDDR 输出电压。

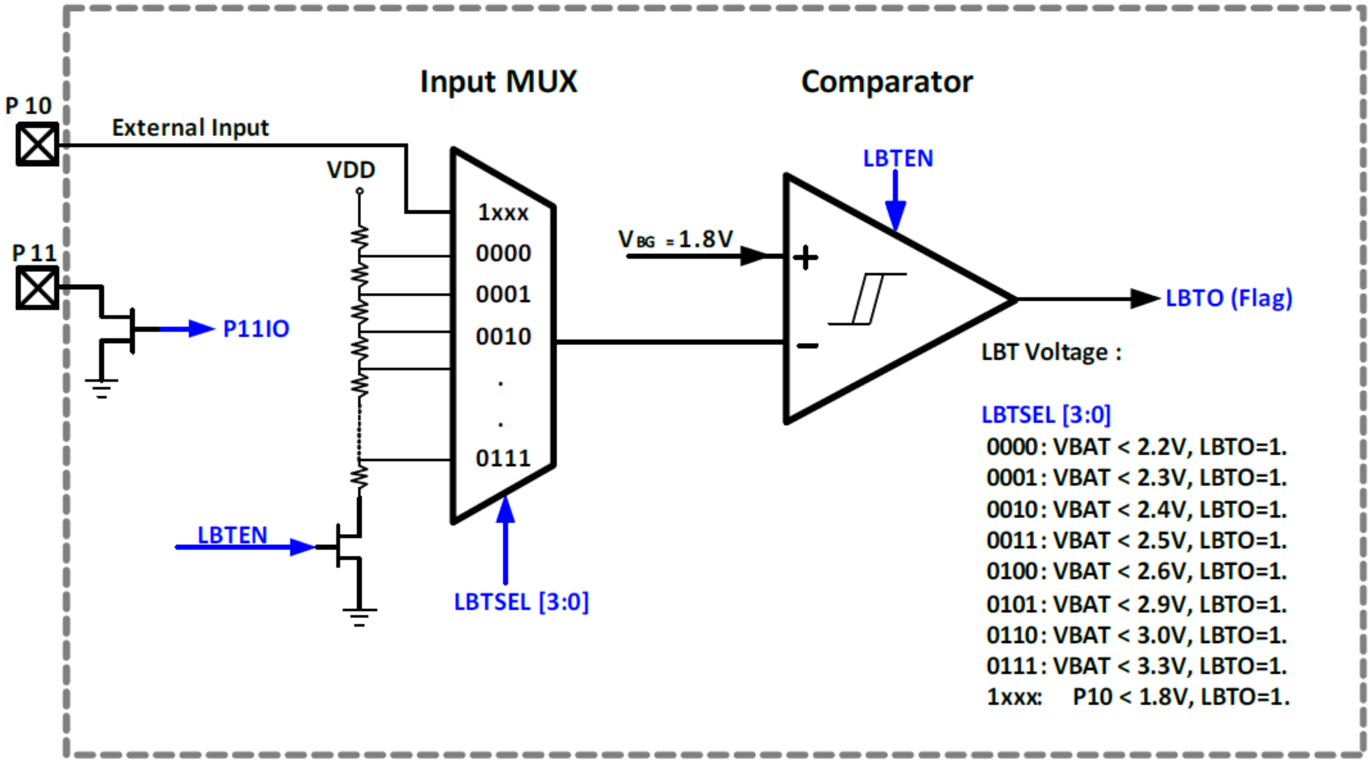
3	ACMSEL	R/W	0	ACM 输出电压选择位。 0：0.75V； 1：1.0V。
2	ACMEN	R/W	0	ACM 电压控制位。 0：禁止； 1：使能。
1	AVESEL	R/W	0	AVE 输出电压选择位。 0：1.5V； 1：2.0V。
0	AVE	R/W	0	AVE 电压输出控制位。 0：禁止 AVE 输出电压； 1：使能 AVE 输出电压。

18.2 Regulator 电气特性

ACM						
Analog common 电压	V_{ACM}	输出电压	0.9	1.0	1.1	V
		输出电压漂移 vs. 温度 △10°C	-	±0.1	-	%
		输出电压漂移 vs. AVDD (2V~5.5V)	-	±0.1	-	%
VACM driving capacity	I_{SRC}	仅 ADC 使用	-	-	10	uA
VACM sinking capacity	I_{SNK}	仅 ADC 使用	-	-	1	mA
AVDDR						
Regulator 输出电压 AVDDR	V_{AVDDR}	AVDDR 输出范围= 2.2V, 2.4V, 2.8V, 3.2V	2.2	-	3.2	V
		AVDDR 设置 2.4V	2.35	2.4	2.45	V
		AVDDR 设置 2.8V	2.75	2.8	2.85	V
		AVDDR 设置 3.2V	3.15	3.2	3.25	V
		AVDDR Load regulation △5mA	V_{AVDDR} ± 0.05V			V
		输出电压漂移 vs. 温度 △10°C	-	±0.1	-	%
		输出电压漂移 vs. AVDD (2V~5.5V)	-	±0.1	-	%
AVDDR 拉/灌电流	I_{AVDDR}		-	-	7.5	mA
Quiescent 功耗	I_{QI}	AVDDR + ACM		200		uA
		AVE				
Regulator 输出电压 AVDDR	V_{AVE}	AVE 输出范围=2.0V	1.95	2.0	2.05	V
		AVE 输出范围=1.5V	1.45	1.5	1.55	V
		AVE Load regulation △5mA			20	mV
		输出电压漂移 vs. 温度 △10°C		±0.1	-	%
		输出电压漂移 vs. AVDD (2V~5.5V)	-	±0.1	-	%
AVE 拉/灌电流	I_{AVE}				7.5	mA

19 比较器

SN8F5950 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 1.8V，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[3:0]寄存器控制，其范围为 2.2V~3.3V，每次 0.1V 一个电压点。



19.1 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 Register (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。
5.2	LBTSEL[3:0]	R/W	010	内部 VDD 电压分频控制位。 0000：VDD < 2.2V, LBTO=1； 0001：VDD < 2.3V, LBTO=1； 0010：VDD < 2.4V, LBTO=1； 0011：VDD < 2.5V, LBTO=1； 0100：VDD < 2.6V, LBTO=1； 0101：VDD < 2.7V, LBTO=1； 0110：VDD < 2.8V, LBTO=1； 0111：VDD < 2.9V, LBTO=1； 1xxx：P10 < 1.8V ,LBTO =1。
1	LBTO	R/W	0	比较器输出标志位。 0：比较器负极输入电压大于 1.8V； 1：比较器负极输入电压小于 1.8V。
0	LBTEN	R/W	00	比较器功能使能位。 0：禁止； 1：使能。

19.2 比较器电气特性

SYM.	*DESCRIPTION*	*PARAMETER*	*MIN.*	*TYP.*	*MAX.*	*UNIT*
VILBT	内部低电压检测	条件：LBTSEL [3:0] = 0000, VLBT=2.2V	2.15	2.2	2.25	V
		条件：LBTSEL [3:0] = 0100, VLBT=2.6V	2.55	2.6	3.65	
		条件： LBTSEL [3:0] = 0101, VLBT=2.9V	2.85	2.9	2.95	
VELBT	外部低电压检测	条件： LBTSEL [3:0] = 1xxx, VLBT=P10 输入	1.75	1.8	1.85	
VHY	比较器迟滞窗口			50	-	mV
ICOMP	功耗	AVDD = 3V		50	-	uA

19.3 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 <SN8F5959.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 &= 0xBF;       // AVDDR=2.4V

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

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

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

20 UART 0

UART0（通用同步异步收发器）提供 1MHz 灵活的全双工传输模式。UART0 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 UTX0 信号，P0.5 接收 URX0 信号。由 P2.2 输出 UTX1 信号，P2.3 接收 URX1 信号。UART0 有 4 种模式：Mode0 ~ Mode3，UART1 只有 2 种模式：Mode1 和 Mode3。

20.1 UART0 模式 0：同步 8 位收发器

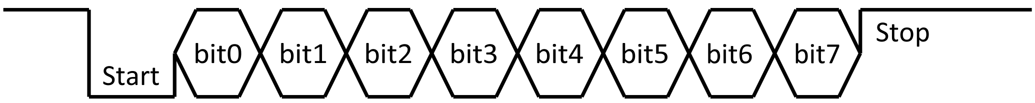
模式 0 下，UART0 发送/接收 8 位长度的数据，没有起始/停止位。总线的首位为最低有效位 LSB，波特率固定为 Fcpu/12。通过设置 REN0 位为 1 和将 RI0 位清零开始接收；而通过写入数据到 S0TBUF 开始发送。

20.2 UART0 模式 1：异步 8 位收发器

模式 1 下，UART0 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。

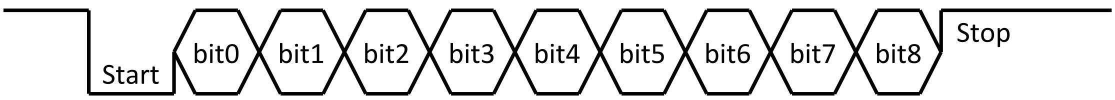


20.3 UART0 模式 2/3：异步 9 位收发器

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 MSB 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。



20.4 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 SMOD 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：

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

表 19-1 UART 波特率的设置建议如下表设置(fcpu = 16 MHz)

Baud Rate	SMOD	CD	S0RELH	S0RELL	Accuracy
4800 bps	1	0	3	98	-0.08 %
9600 bps	1	0	3	CC	0.16 %
19200 bps	1	0	3	E6	0.16 %
38400 bps	1	0	3	F3	0.16 %
56000 bps	1	0	3	F7	-0.80 %
57600 bps	1	1	3	F6	0.79 %
115200 bps	1	1	3	FB	0.79 %
250k bps	1	0	3	FE	0 %
500k bps	1	0	3	FF	0 %

注：波特率为 57600，115200 和 12800bps 时 CD 必须设置为 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。TC2 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。

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

表 19-2 TC1 溢出周期 UART 波特率的设置建议如下表设置(fcpu = 16 MHz)

Baud Rate	SMOD	CD	Timer Priod	TC1CH/	TC1RH/	Accuracy
4800 bps	1	0	13.00 us	0xFE30	0xFF30	-0.16 %
9600 bps	1	0	6.500 us	0xFF98	0xFF98	-0.16 %
19200 bps	1	0	3.250 us	0xFFCC	0xFFCC	-0.16 %
38400 bps	1	0	1.628 us	0XFFE6	0XFFE6	-0.16 %
56000 bps	1	0	1.125 us	0xFFEE	0xFFEE	0.79 %
57600 bps	1	1	1.250 us	0xFFEC	0xFFEC	0.79 %
115200 bps	1	1	0.625 us	0XFFF6	0XFFF6	0.79 %
128k bps	1	0	0.500 us	0xFFF8	0xFFF8	2.34 %
250k bps	1	0	0.250 us	0xFFFC	0xFFFC	0 %
500k bps	1	0	0.125 us	0xFFFE	0xFFFE	0 %

*例：模式 1 和模式 3 中，选择 TC2 溢出周期（BD=0），波特率 115200 如下面格式设置。（Fcpu=16MHz）

```
TC2CH = 0x0FF;    // baud rate 9600
TC2CL = 0x098;    // 6.51us Overflow
TC2RH = 0x0FF;    // auto-reload value
TC2RL = 0x098;    //
TC2M |= 0x0F0;    // TC1 ENABLE, TC1 timer clock= Fpcu/1

S0CON2 = 0x00;    // baud rate = TC1 Timer overflow period
PCON |= 0x80;     // SMOD=1,Fcpu*2 in mode1/mode3;
S0CON |=0x50;     // UART Enable,UART Mode 1

```

20.5 UART0 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	CD1	CD	TEN0	TEN1	ENDBS	PULSEL	URCHS
S0RBUF	S0RBUF 7	S0RBUF 6	S0RBUF 5	S0RBUF 4	S0RBUF 3	S0RBUF 2	S0RBUF 1	S0RBUF 0
S0TBUF	S0TBUF 7	S0TBUF 6	S0TBUF 5	S0TBUF 4	S0TBUF 3	S0TBUF 2	S0TBUF 1	S0TBUF 0
PCON	SMOD	-	-	IDLE+	P2SEL	GF0	STOP	IDLE
S0RELH	S0RELH 7	S0RELH 6	S0RELH 5	S0RELH 4	S0RELH 3	S0RELH 2	S0RELH 1	S0RELH 0
S0RELL	S0REL7	S0REL6	S0REL5	S0REL4	S0REL3	S0REL2	S0REL1	R0REL0
S1CON	SM	-	SM21	REN1	TB81	RB81	TI1	RI1
S1TBUF	S1TBUF 7	S1TBUF 6	S1TBUF 5	S1TBUF 4	S1TBUF 3	S1TBUF 2	S1TBUF 1	S1TBUF 0
S1RBUF	S1RBUF 7	S1RBUF 6	S1RBUF 5	S1RBUF 4	S1RBUF 3	S1RBUF 2	S1RBUF 1	S1RBUF 0
S1RELH	S1RELH 7	S1RELH 6	S1RELH 5	S1RELH 4	S1RELH 3	S1RELH 2	S1RELH 1	S1RELH 0
S1RELL	S1REL7	S1REL6	S1REL5	S1REL4	S1REL3	S1REL2	S1REL1	R1REL0
IEN0	EAL	-	-	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	ETS1	ERS1	ESPI	EI2C
P1OC	-	P23OC	P22OC	P05OC	P04OC	P17OC	P16OC	P15OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
P0	P07	P06	P05	P04	P03	P02	P01	P00

S0CON Register (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 Register (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	CD1	R/W	0	串口 1 时钟分频。为 1 时精确度功能使能，波特率 4800~128000 bps < 1% 精度调整位。
5	CD0	R/W	0	串口 0 时钟分频。为 1 时精确度功能使能，波特率 4800~128000 bps < 1% 精度调整位。
4	TEN0	R/W	0	UART0 发送功能。 0：禁止； 1：使能。
3	TEN1	R/W	0	UART1 发送功能。 0：禁止； 1：使能。
2..0	Reserved	R	0	为 0。

S0TBUF Register (0x99)

Bit	Field	Type	Initial	Description
7..0	S0TBUF	R/W	0x00	写入数据的操作触发 UART 通讯（首先 LSB）

S0RBUF Register (0xD9)

Bit	Field	Type	Initial	Description
7..0	S0RBUF	R/W	0x00	在数据包未读取接收数据。

PCON Register (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 Registers (S0RELH: 0xBA, S0RELL: 0xAA)

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

S1CON Register (0x8D)

Bit	Field	Type	Initial	Description
7..6	SM	R/W	00	UART 模式选择。 0：模式 1； 1：模式 3。
5	SM21	R/W	0	多重处理器通讯控制位（模式 3） 0：禁止； 1：使能。
4	REN1	R/W	0	UART 接收功能。 0：禁止； 1：使能。
3	TB1	R/W	0	发送数据的第 9 位（模式 3）。
2	RB1	R/W	0	接收数据的第 9 位。
1	TI1	R/W	0	发送 UART 的中断标志。
0	RI1	R/W	0	接收 UART 的中断标志。

S1TBUF Register (0x8E)

Bit	Field	Type	Initial	Description
7..0	S1TBUF	R/W	0x00	写入数据的操作触发 UART1 通讯（首先 LSB）

S1RBUF Register (0x8A)

Bit	Field	Type	Initial	Description
7..0	S1RBUF	R/W	0x00	在数据包未读取接收数据。

S1RELH/S1RELL Registers (S0RELH: 0xBC, S0RELL: 0x8B)

Bit	Field	Type	Initial	Description
15..10	Reserved	R	0x00	
9..0	S0REL[9:0]	R/W	0x00	S1RELH[1:0] & S1RELL[7:0]，UART 重装寄存器，用户产生 UART 波特率。

IEN0 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	使能中断，请参考中断章节。
4	ETS0	R/W	0	使能 UART0 TX 中断。
3	ERS0	R/W	0	使能 UART0 RX 中断。
Else				请参考其它章节。

IEN1 Register (0xB8)

Bit	Field	Type	Initial	Description
3	ETS1	R/W	0	使能 UART1 TX 中断。
2	ERS1	R/W	0	使能 UART1 RX 中断。
Else				请参考其它章节。

P1OC Register (0xE4)

Bit	Field	Type	Initial	Description
6	P23OC	R/W	0	0：切换 P2.3（URX1）为输入模式（要求）； ~~1：切换 P2.3（URX1）为开漏模式*。~~
5	P22OC	R/W	0	0：切换 P2.2（UTX1）为上拉模式； 1：切换 P2.2（UTX1）为开漏模式。
4	P05OC	R/W	0	0：切换 P0.5（URX0）为输入模式（要求）； ~~1：切换 P0.5（URX0）为开漏模式*。~~
3	P04OC	R/W	0	0：切换 P0.4（UTX0）为上拉模式； 1：切换 P0.4（UTX0）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 Register (0xF9)

Bit	Field	Type	Initial	Description
5	P05M	R/W	0	0：设置 P0.5 (URX0) 为输入模式（要求）； ~~1：设置 P0.5 (URX0) 为输出模式。~~ *
4	P04M	R/W	0	~~0：设置 P0.4 (UTX0) 为输入模式；~~ * 1：设置 P0.4 (UTX0) 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。

P2M Register (0xFB)

Bit	Field	Type	Initial	Description
3	P23M	R/W	0	0：设置 P2.3 (URX0) 为输入模式（要求）； ~~1：设置 P2.3 (URX0) 为输出模式。~~ *
2	P22M	R/W	0	~~0：设置 P2.2 (UTX0) 为输入模式；~~ * 1：设置 P2.2 (UTX0) 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 Register (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 UART 数据 (required)
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。

P2 Register (0xA0)

Bit	Field	Type	Initial	Description
3	P23	R/W	0	随时读取该位以监控总线状态。
2	P22	R/W	0	~~0：设置 P2.2 (UTX) 始终为低电平；~~ * 1： P2.2 (UTX) 输出 UART 数据 (required)
Else				请参考其它章节。

20.6 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 = 0xCC;

   // 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;
   }
}

```

21 UART 1

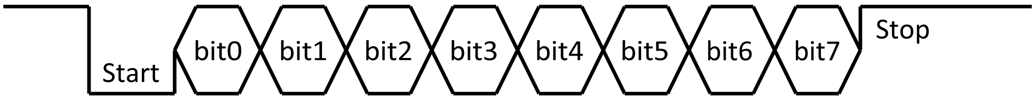
UART1（通用同步异步收发器）提供 1MHz 灵活的全双工传输模式。UART1 共有 2 种异步操作模式，支持 8 位和 9 位数据的发送，包含起始/停止位。由 P2.2 输出 UTX1 信号，P2.3 接收 URX1 信号。

21.1 UART1 模式 B：异步 8 位收发器

模式 B 下，UART1 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 S1RELH/S1RELL 寄存器.

通过写入数据到 S1BUF 寄存器开始发送，发送的首位是起始位（始终为 0），然后是 S1BUF 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 TI1 位设置为 1。

若使能接收功能（REN0=1），UART1 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，S1BUR 会进行更新。

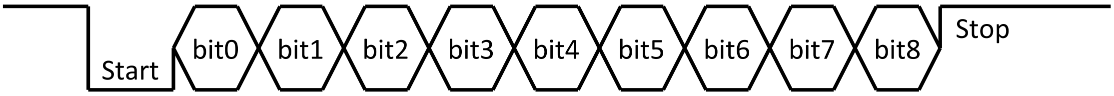


21.2 UART1 模式 A：异步 9 位收发器

模式 A 包含起始位、9 位数据位和停止位。模式 A 的波特率可通过 S1RELH/S1RELL 寄存器控制。

第 9 位数据位（在 S1BUF 寄存器的 MSB 最高位传输完毕之后传输）可通过写入 TB81 寄存器来传送，接收后在 RB81 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 SM21 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 SM21 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 SM21 清 0，地址不匹配的从机，保持 SM21 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。



21.3 波特率控制

UART1 模式 A 和模式 B 的波特率由 S1RELH/S1RELL 产生。

在模式 A 和模式 B 波特率由下面公式产生：

$$Baud\ rate = \frac{\mathrm{fcpu}}{(32 - 4 \times \mathrm{CD1}) \times (1024 - \mathrm{S1REL})}$$

表 22-1 UART1 波特率的设置建议如下表设置(fcpu = 16 MHz)

Baud Rate	CD 1	S1RELH	S1RELL	Accuracy
4800 bps	0	3	98	0.16 %
9600 bps	0	3	CC	0.16 %
19200 bps	0	3	E6	0.16 %
38400 bps	0	3	F3	0.16 %
56000 bps	0	3	F7	-0.79 %
57600 bps	1	3	F6	-0.79 %
115200 bps	1	3	FB	-0.79 %
250k bps	0	3	FE	0 %
500k bps	0	3	FF	0 %

注：波特率为 57600，115200 和 12800bps 时 CD1 必须设置为 1。

21.4 UART1 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S0CON2	BD	CD1	CD	TEN0	TEN1	-	-	-
S1CON	SM	-	SM21	REN1	TB81	RB81	TI1	RI1
S1TBUF	S1TBUF 7	S1TBUF 6	S1TBUF 5	S1TBUF 4	S1TBUF 3	S1TBUF 2	S1TBUF 1	S1TBUF 0
S1RBUF	S1RBUF 7	S1RBUF 6	S1RBUF 5	S1RBUF 4	S1RBUF 3	S1RBUF 2	S1RBUF 1	S1RBUF 0
S1RELH	S1RELH 7	S1RELH 6	S1RELH 5	S1RELH 4	S1RELH 3	S1RELH 2	S1RELH 1	S1RELH 0
S1RELL	S1REL7	S1REL6	S1REL5	S1REL4	S1REL3	S1REL2	S1REL1	R1REL0
IEN0	EAL	-	-	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	ETS1	ERS1	ESPI	EI2C
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
P2					P23	P22	P21	P20

S1CON Register (0x8B)

Bit	Field	Type	Initial	Description
7	SM	R/W	00	UART1 模式选择位。 0：模式 A 9 bit UART； 1：模式 B 8 bit UART。
5	SM21	R/W	0	多重处理器通讯控制位。 0：禁止； 1：使能。 当 S1CON 寄存器的 SM21 置起时，只有在第 9 位（S1CON 的 RB81 位）为 1 时，才会产生接受中断，否则接收不会产生中断。
4	REN1	R/W	0	UART1 接收功能控制位。 0：禁止； 1：使能。
3	TB81	R/W	0	发送数据的第 9 位（模式 A）。 模式 A 中通过串行端口 1 传输数据时使用该位，该位的状态与发送数据的第 9 位状态相对应（例如，奇偶检验或多重处理通讯位），有软件控制。
2	RB81	R/W	0	接收数据的第 9 位。 模式 A 中通过串行端口 1 传输数据时使用该位，它反应接收数据的第 9 位。在模式 B 中，如果多重处理器通讯控制使能（SM21=0），则该位为接收到的停止位。
1	TI1	R/W	0	发送 UART1 的中断标志。
0	RI1	R/W	0	接收 UART1 的中断标志。

S0CON2 Register (0xD8)

Bit	Field	Type	Initial	Description
6	CD1	R/W	0	串口 1 时钟分频。为 1 时精确度功能使能，波特率 4800~128000 bps < 1% 精度调整位。 1：使能 UART1 精度功能； 0：禁止 UART1 精度功能；
7..0	Reserved	R	0	参考其它章节。

S1TBUF Register (0x8C)

Bit	Field	Type	Initial	Description
7..0	S1TBUF	R/W	0x00	写入数据的操作触发 UART1 通讯（首先 LSB）

S1RBUF Register (0x88)

Bit	Field	Type	Initial	Description
7..0	S1RBUF	R/W	0x00	在数据包未读取接收数据。

S1RELH/S1RELL Registers (S1RELH: 0x89, S1RELL: 0x8A)

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 Register (0xA8)

Bit	Field	Type	Initial	Description
7	EAL	R/W	0	使能中断，请参考中断章节。
6..0				请参考其它章节。

IEN1 Register (0xB8)

Bit	Field	Type	Initial	Description
3	ETS1	R/W	0	使能 UART1 TX 中断。
2	ERS1	R/W	0	使能 UART1 RX 中断。
Else				请参考其它章节。

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）。

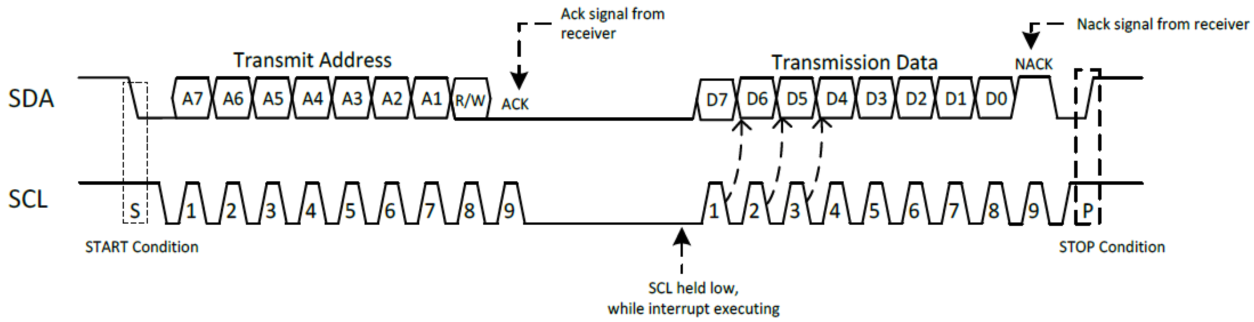
主机发送数据给从机时，叫做“WRITE”操作；从机发送数据给主机时，叫做“READ”操作。I2C 也支持多主机通讯，通过一种仲裁方式来决定哪个主机拥有控制总线以及传输数据的权限，从而保证数据传送的正确性。

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

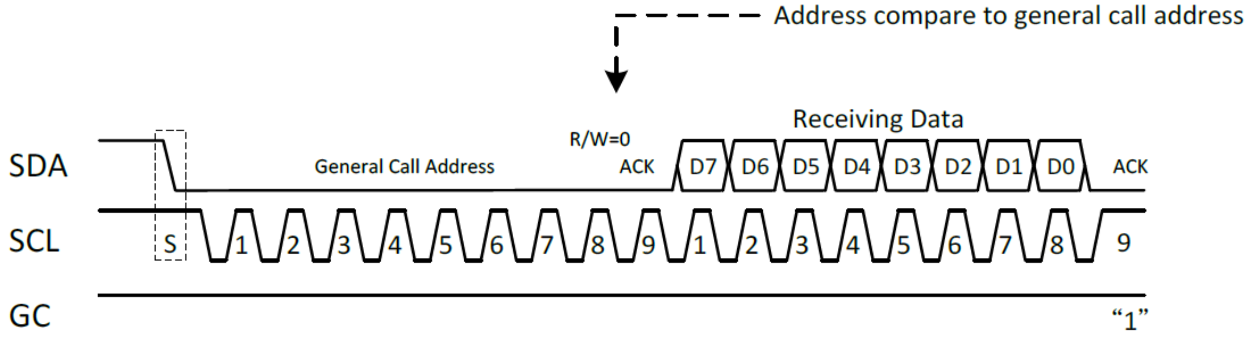
从机发送模式为从机发送数据给主机，串行时钟由 SCL 输入时，串行数据由 SDA 输出。接收到来自主机的 START 信号后开始数据发送。接收到 START 信号后，开始接收从机设备指定的地址字节。地址字节包含 7 位地址位，第 8 位为数据方向位（R/W），该位设为 1 时使能从机发送。若接收到的地址字节与 I2CADR 寄存器中的地址相匹配，从机将发送 ACK 信号；另外，若广播呼叫地址标志位设为 1（GC=1），接收到广播呼叫地址（00H）后，从机设备也会发送一个 ACK 信号。接下来，从机发送一个或多个数据字节给主机，每发送一个字节之后，从机要等待主机返回的 ACK 信号。最后，主机产生 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 Prescaler	Bit Frequency (kHz) 4MHz	8MHz	16MHz
0	0	0	256	15.6	31.3	62.5
0	0	1	224	17.9	35.7	71.4
0	1	0	192	20.8	41.7	83.3
0	1	1	160	25	50	100
1	0	0	960	4.2	83.3	16.7
1	0	1	120	33.3	66.7	133.3
1	1	0	60	66.7	133.3	266.7
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：

```
TC2CH = 0x0FF;    // clock rate 400k Hz = 1/0.3125us/8
TC2CL = 0x0F6;    // 0.3125us Overflow
TC2RH = 0x0FF;    // auto-reload value
TC2RL = 0x0F6;    //
TC2M |= 0x0F0;    // TC2 ENABLE, TC2 timer clock= Fpcu/1

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

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 在线的异常，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（SM Bus）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 位的缓存器来控制。计算公式如下：

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

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]	SMBDS	Descripti
000	Tmext_	选择 Tmext 寄存器的低字节
001	Tmext_	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	-	-	-	-	SMBSTP2	SMBSTP1	SMBSTP0
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 Register (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。当它不在字节移位元的过程中时，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 Register (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 Register (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：（TC2 定时器溢出率）/8。
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 Register (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	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。
	20H	已发送 SLA+W；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。
	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；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
	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	加载数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			加载数据字节	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。	加载数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			加载数据字节	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。	加载数据字节	X	0	0	0	发送最后一个字节的数据，并将接收到 ACK 信号
			加载数据字节	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	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。

		信号。	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 Register (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 Register (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 Register (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 Register (0xB8)

Bit	Field	Type	Initial	Description
0	EI2C	R/W	0	I2C 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;
```

```
        case 0x58:            // read last byte & stop
            I2C_RXDatan = I2CDAT;
            I2CCON |= 0x10;     // STOP (STO)
            break;

        default:
            I2CCON |= 0x10;     // STOP (STO)
    }
    I2CCON &= 0xF7;           // Clear I2C flag (SI)
}
```

23 SPI

串行外设接口 SPI 有 2 种操作模式：主控模式和从动模式。主机通过 SCK 引脚发送总线时钟信号；反之从机设备基于 SCK 引脚的时钟输入处理通讯。从动模式下由寄存器使能片选引脚（SSN）。

23.1 SPI 主控模式

SPI 控制模式共有 7 种类型的时钟发生器，从 Fcpu/2~Fcpu/128，产生的时钟由 SCK 引脚（与 P1.7 共享）输出，其 IDLE 状态由 CPOL 控制。

输入输出数据的相位由 CPHZ 寄存器自动指定。主控模式下，MOSI 引脚（与 P1.5 共享）负责输出数据，MISO 引脚（与 P1.6 共享）负责获取来自从机设备的资料。通过写入数据到 SPDAT 寄存器开始 SPI 通讯；数据发送完成后，从 MISO 引脚读取接收到的数据。

主控模式下，有 2 种包含中断功能的状态标志：

- SPIF 寄存器显示一字节数据通讯的结束，若此时使能 ESPI 位则释放中断。
- 发送时由 SSN（与 P1.4 共享）低电平状态释放 MODF，通过设置 SSDIS 位屏蔽中断源。

23.2 SPI 从动模式

SPI 从动模式监控 SCK 引脚控制 MISO 和 MOSI 通讯，但其最大时钟 rate 限制在 Fcpu/8。在连接到 SPI 总线的配置相同时，从动设备可用于指定 CPOL 和 CPHA 设置。

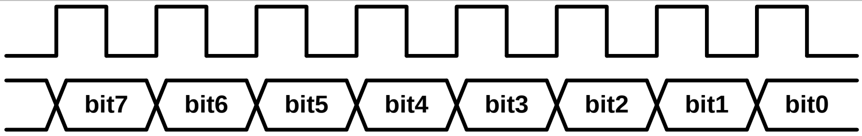
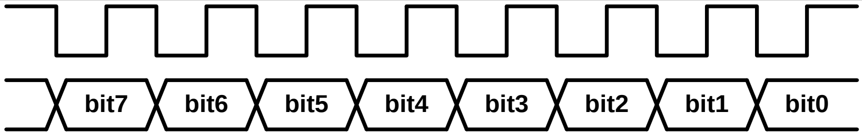
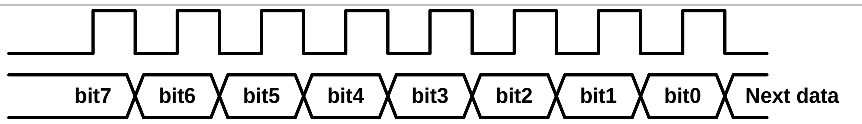
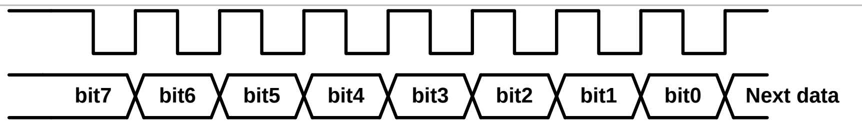
输入数据时可将从动模式看作是 MOSI 引脚，发送数据时可看作是 MISO 引脚。由于疏忽，SSDIS 寄存器为低电平状态，即从动选择引脚（SSN）是有功能的。若 SSN 引脚为低电平状态，则处理 SPI 通讯；而 SSN 为高电平状态时，从动设备处于悬浮状态。

从动模式下，有 2 种包含中断功能的状态标志：

- SPIF 寄存器显示一字节数据通讯的结束，发送 SPDAT 的初始值后，再通过 SPDAT 读取从 MOSI 接收的到的数据。
- MODF 显示的是：在完成一字节数据通讯之前改变从动选择引脚 SSN 为高电平，换句话说就是打断最后一次 SPI 通讯。

23.3 SPI 操作

下表说明了 CPOL 和 CPHA 的 4 种不同的设置，和每种组合下的总线操作。

CPOL	CPHA	Diagrams	Description
0	1		SCK 低电平空闲 下降沿锁存资料
1	1		SCK 高电平空闲 上升沿锁存数据
0	0		SCK 低电平空闲 上升沿锁存数据
1	0		SCK 高电平空闲 下降沿锁存资料

23.4 SPI 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPCON	SPR2	SPEN	SSDIS	MATR	CPOL	CPHA	SPR1	SPR0
SPSTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
SPDAT	SPDAT7	SPDAT6	SPDAT5	SPDAT4	SPDAT3	SPDAT2	SPDAT1	SPDAT0
IEN0	EAL	-	-	ETS0	ERS0	EX1	ET0	EX0
IEN1	-	-	-	-	-	-	ESPI	EI2C
P1OC	-	-	-	P05OC	P04OC	P17OC	P16OC	P15OC
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
P1	P17	P16	P15	P14	P13	P12	P11	P10

SPCON 寄存器 (0xE2)

Bit	Field	Type	Initial	Description
7,1,0	SM[2:0]	R/W	000	SPI 波特率发生器（仅在主控模式下有效）。 000：Fcpu/2； 001：Fcpu/4； 010：Fcpu/8； 011：Fcpu/16； 100：Fcpu/32； 101：Fcpu/64； 110：Fcpu/128； ~~111：保留。~~
6	SPEN	R/W	0	SPI 通讯功能。 0：禁止； 1：使能。
5	SSDIS	R/W	0	从动选择引脚功能（仅在 MSTR = 0，CPHA = 0 时有效）。 0：使能从动选择引脚 SSN 功能； 1：禁止从动选择引脚 SSN 功能。
4	MSTR	R/W	1	SPI 模式选择位。 0：从动模式； 1：主控模式。
3	CPOL	R/W	0	SCK 引脚空闲状态位。 0：SCK 低电平空闲； 1：SCK 高电平空闲
2	CPHA	R/W	1	数据的时钟相位锁存控制位。 0：由第一个时钟沿锁存资料； 1：由第二个时钟沿锁存资料。

SPSTA 寄存器 (0xE1)

Bit	Field	Type	Initial	Description
7	SPIF	R	0	SPI 通讯完成标志位。 在通讯结束时自动设置为 1； 通过读取 SPSTA，SPDAT 寄存器自动清零。
6	WCOL	R	0	写操作冲突标志位。 在通讯时对 SPDAT 执行写操作自动设置为 1； 通过读取 SPSTA，SPDAT 寄存器自动清零。
5	SSERR	R	0	同步从动选择引脚错误位。 若 SSN 错误控制时自动设置为 1； 通过清 SPEN 自动清零。
4	MODF	R	0	模式错误标志位。
3..0	Reserved	R	0x00	

SPDAT 寄存器 (0xE3)

Bit	Field	Type	Initial	Description
7..0	SPDAT	R/W	0x00	主控模式：写操作触发 SPI 通讯；接收的数据可在一字节通讯结束时读取（SPIF 自动设置为 1）。 从动模式：通过 SCK 输入发送写入的数据；接收的数据可在一字节通讯结束时读取（SPIF 自动设置为 1）。

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。
Else				请参考其它章节。

IEN1 寄存器(0xB8)

Bit	Field	Type	Initial	Description
1	ESPI	R/W	0	使能 SPI 中断。
Else				请参考其它章节。

P1OC 寄存器 (0xE4)

Bit	Field	Type	Initial	Description
2	P17OC	R/W	0	0：切换 P1.7（SCK）为输入或输出模式； 1：切换 P1.7（SCK）为开漏模式。
1	P16OC	R/W	0	0：切换 P1.6（MISO）为输入或输出模式； 1：切换 P1.6（MISO）为开漏模式。
0	P15OC	R/W	0	0：切换 P1.5（MOSI）为输入或输出模式； 1：切换 P1.5（MOSI）为开漏模式。
Else				请参考其它章节。

P1M Register (0xFA)

Bit	Field	Type	Initial	Description
7	P17M	R/W	0	0：设置 P1.7（SCK）为输入模式[从动模式] 1：设置 P1.7（SCK）为输出模式[主控模式]
6	P16M	R/W	0	0：设置 P1.6（MISO）为输入模式[主控模式] 1：设置 P1.6（MISO）为输出模式[从动模式]
5	P15M	R/W	0	0：设置 P1.5（MOSI）为输入模式[从动模式] 1：设置 P1.5（MOSI）为输出模式[主控模式]
4	P14M	R/W	0	0：设置 P1.4（SSN）为输入模式*；。 1：设置 P1.4（SSN）为输出模式*。
Else				请参考其它章节。

[1] 在从动模式下本就要设置 SCK 为输入模式；在主控模式下建议设置为输出模式。
[2] 在主控模式下本就要设置 MISO 为输入模式；在从动模式下建议设置为输出模式。
[3] 在从动模式下本就要设置 MOSI 为输入模式；在主控模式下建议设置为输出模式。
[*] 若从动模式下使能 SSN 功能：本就是要设置 SSN 为输入模式。

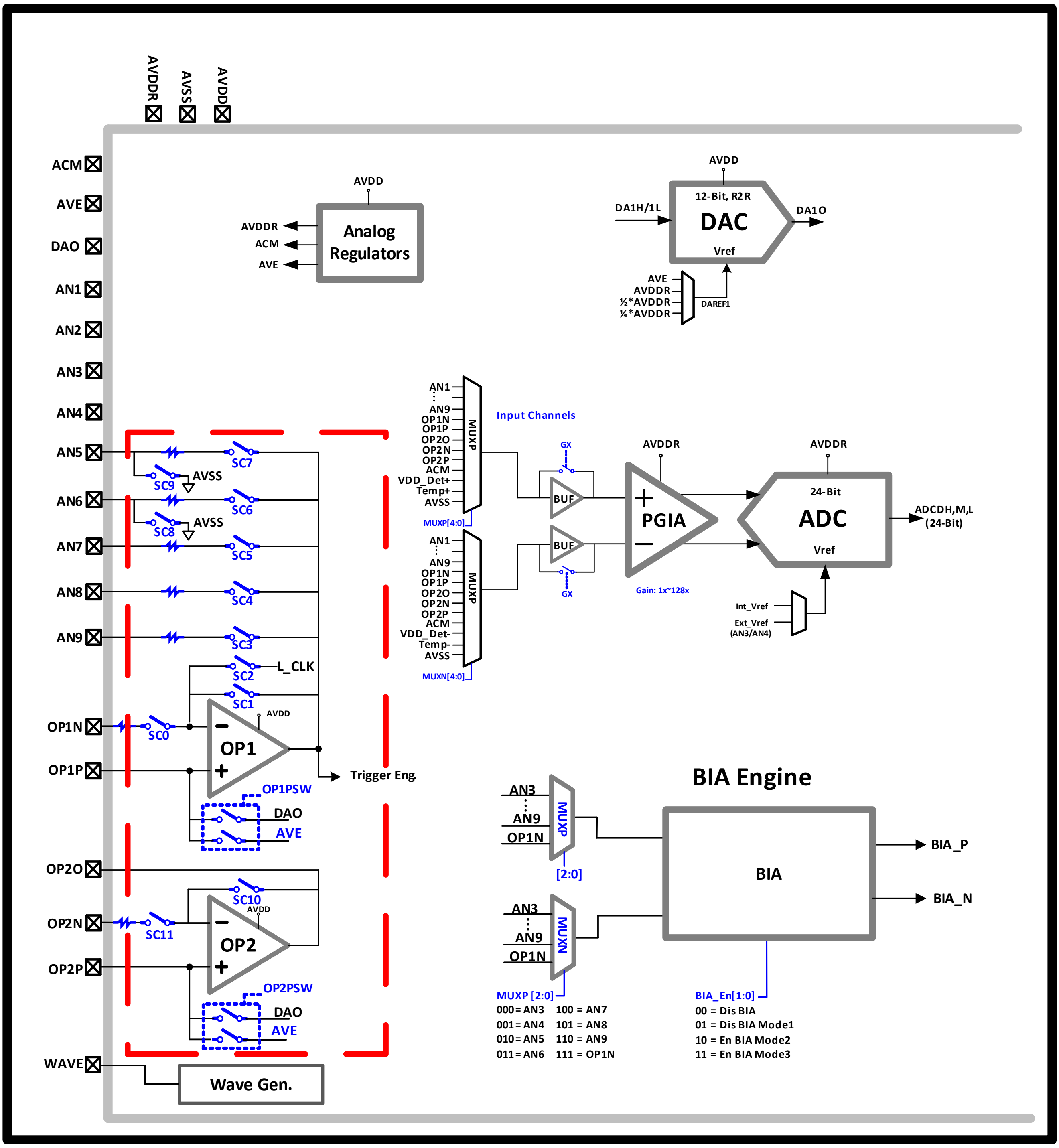
23.5 示例代码

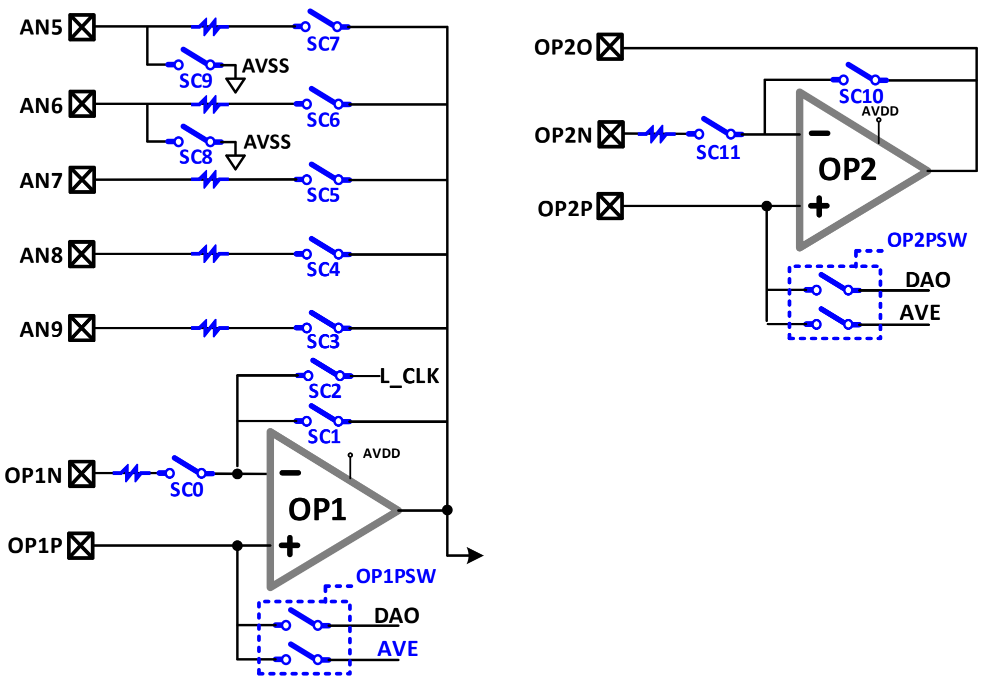
下面的示例代码程序演示了如何执行 SPI。

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

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

void SpiMaster(void)
{
   unsigned char u8RcvData = 0;

   //SCK & MOSI = output, MISO = input P1M |=
   0x14;
   //Enable Spi, Master mode, SSN pin disable, Fclk/128
   //SCK idle low, data latch at falling edge
   SPCON = SpiEn | SpiMaster | SpiMode1 | SpiSSNDis | 0x82;
   //Enable Global/SPI interrupt IEN1    |=
   0x02;
   IEN0   |=  0x80;       //enable global interrupt

   while (1) { SPDAT =
      0x55;
      while(!u8TxCompleted);              // wait end of transmition
      u8TxCompleted  =  0;                // clear sw flag
      u8RcvData  =  u8SpiData;             // receive 0x66

      SPDAT = 0x99;
      while(!u8TxCompleted);              // wait end of transmition
      u8TxCompleted  =  0;                // clear sw flag
      u8RcvData  =  u8SpiData;             // receive 0xAA
   }
}

void SpiInterrupt(void) interrupt ISRSpi //0x4B
{
   switch (  SPSTA  )                     // Clear SPI flag (SPIF) by reading
   {
      case  0x80: u8SpiData =
         SPDAT;
         u8TxCompleted = 1;
         break;
      case 0x10:
         // Mode Fault
         break;
   }
}
```

24 OPA 和 Switch

SN8F5930 内置 2 个 rail to rail 的放大器 OPA 和模拟集成开关 SC0~SC11。OP1 正极输入连接 DAO 和 AVE 开关，OP1 输出连接 SC10。SC3~SC7，连接 AN9~AN5。OP2 正极输入连接 DAO 和 AVE 开关，OP1 输出连接开关 SC1。OPA 连接的分立元件可配置逆变放大或 I-V 逆变放大。





24.1 OPA 控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM1	SC1	SC0	OP2PSW1	OP2PSW0	OP1PSW1	OP1PSW0	OP2EN	OP1EN
OPM2	SC9	SC8	SC7	SC6	SC5	SC4	SC3	SC2
OP2SW	-	-	-	-	-	-	SC11	SC10

OPM1 Register (0xE2)

Bit	Field	Type	Initial	Description
7	SC1	R/W	0	SC1 开关。 0：禁止； 1：使能。
6	SC0	R/W	0	SC0 开关。 0：禁止； 1：使能。
5..4	OP2PSW	R/W	00	00：断开所有开关； 01：CMOS 开关连接 DAO 和 OP2P； 10：CMOS 开关连接 AVE 和 OP2P。
3..2	OP1PSW	R/W	0	00：断开所有开关； 01：CMOS 开关连接 DAO 和 OP1P； 10：CMOS 开关连接 AVE 和 OP1P。
1	OP2EN	R/W	1	OP2 控制位。 0：禁止； 1：使能。
0	OP1EN	R/W	0	OP1 控制位。 0：禁止； 1：使能。

OPM2 Register (0x88)

Bit	Field	Type	Initial	Description
7	SC9	R/W	0	SC9 开关控制位。 0：禁止； 1：使能。
6	SC8	R/W	0	SC8 开关控制位。 0：禁止； 1：使能。
5	SC7	R/W	0	SC7 开关控制位。 0：禁止； 1：使能。
4	SC6	R/W	0	SC6 开关控制位。 0：禁止； 1：使能。
3	SC5	R/W	0	SC5 开关控制位。 0：禁止； 1：使能。

2	SC4	R/W	0	SC4 开关控制位。 0：禁止； 1：使能。
1	SC3	R/W	0	SC3 开关控制位。 0：禁止； 1：使能。
0	SC2	R/W	0	SC2 开关控制位。 0：禁止； 1：使能。

OP2SW Register (0xFFF5)

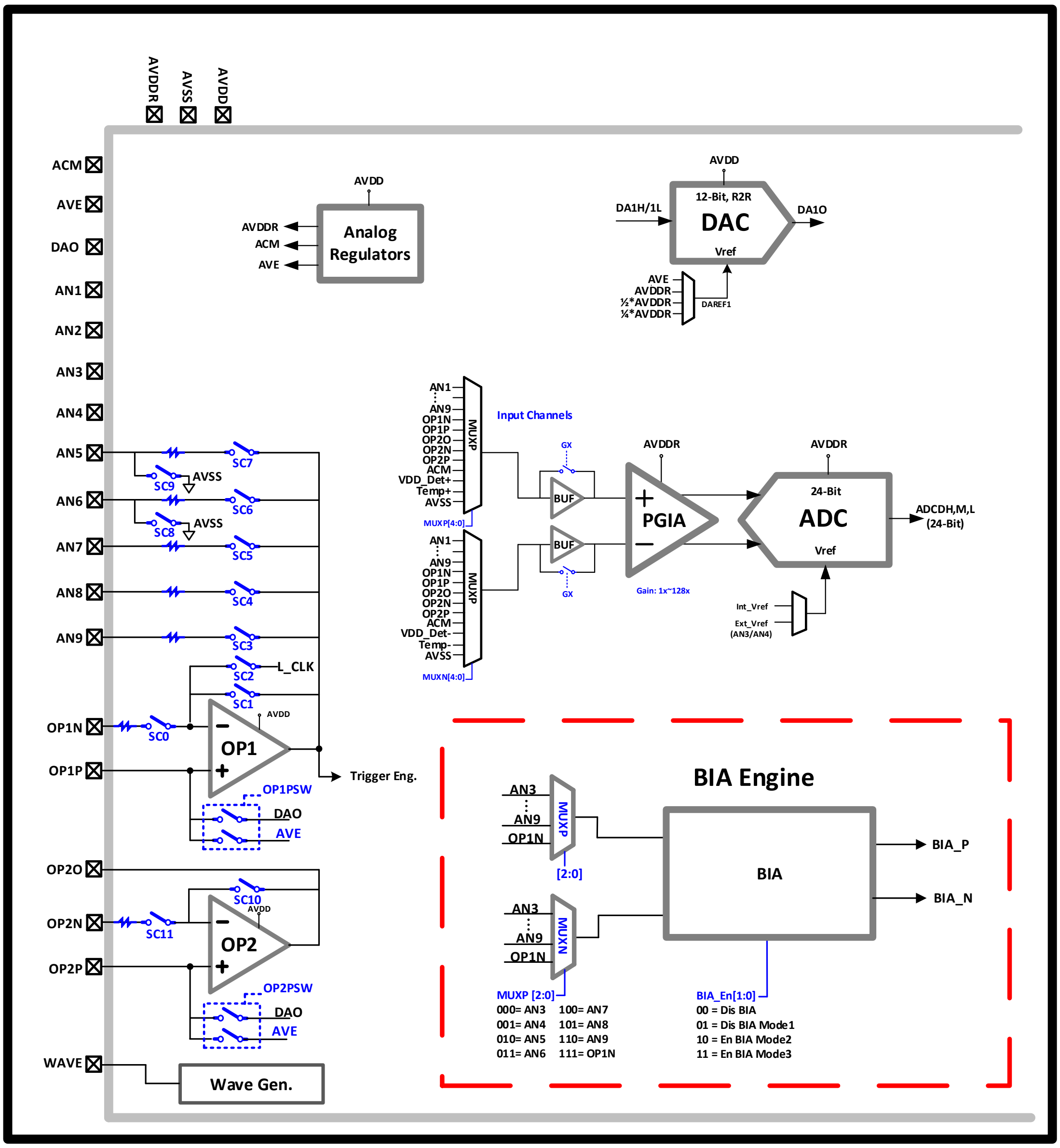
Bit	Field	Type	Initial	Description
1	SC11	R/W	0	SC11 控制位。 0：禁止； 1：使能。
0	SC10	R/W	0	SC10 chopper 控制位。 0：禁止； 1：使能。

25 BIA

SN8F5930 内置 BIA Engine，用于测量生物阻抗数据。通过 BIA Engine 分析被测器件的成分。

25.1 BIA 控制寄存器

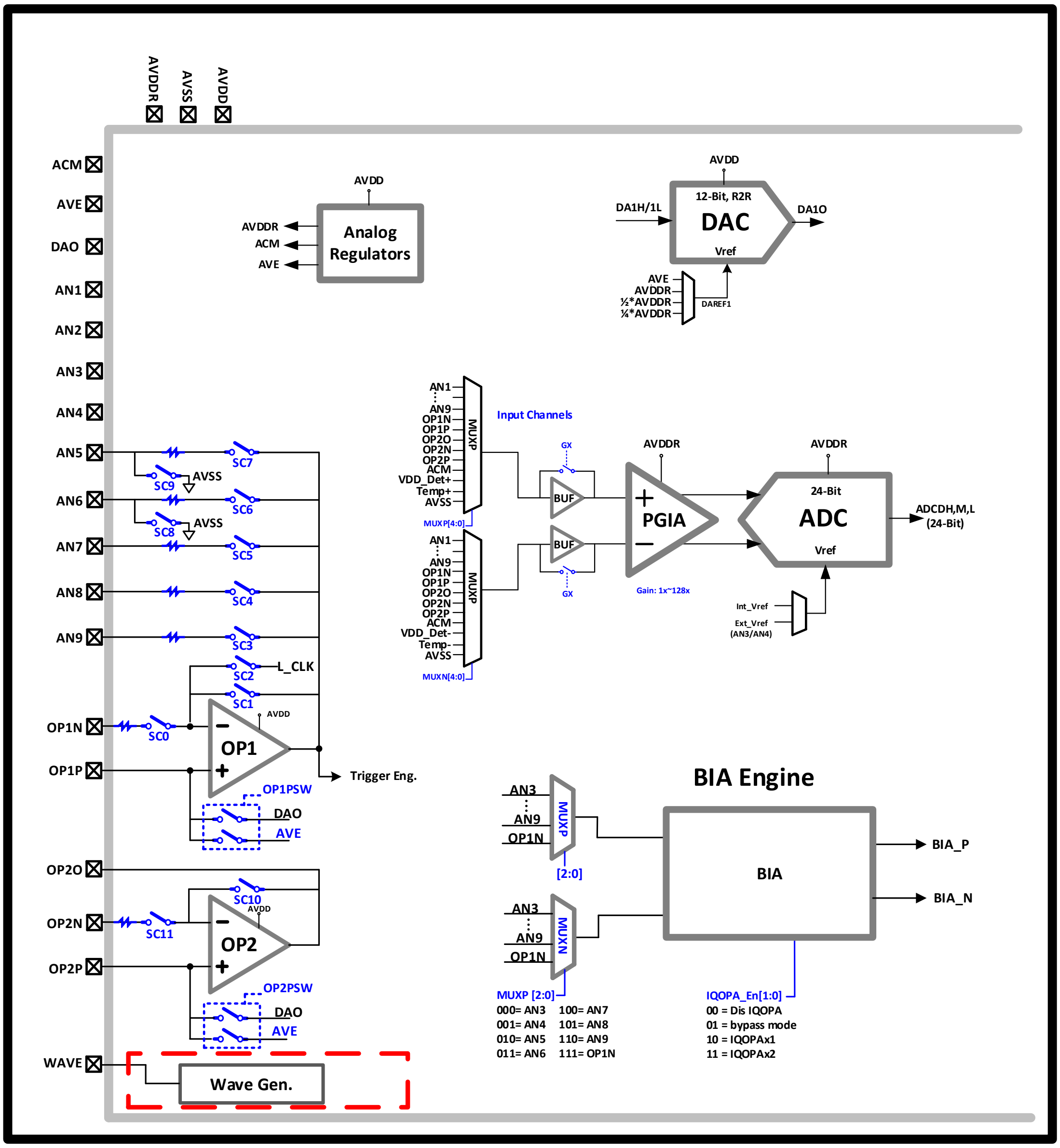
Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IQCON0	-	-	-	-	-	-	-	-
IQCON1	-	-	-	-	-	-	-	-



注：关于 HCT 参考样本代码和工程方案，请联系 SONIX 测量系统设计部。

26 Wave 发生器

SN8F5930 内置 BIA 信号输入发生器。周期信号由高速 DAC 产生。



26.1 DDS 控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IQCON0	-	-	-	-	-	-	-	-
IQCON1	-	-	-	-	-	-	-	-
DDSCON1	-	-	-	-	-	-	-	-
DDSCON2	-	-	-	-	-	-	-	-
FTW	FTW7	FTW6	FTW5	FTW4	FTW3	FTW2	FTW1	FTW0
PTW	PTW7	PTW6	PTW5	PTW4	PTW3	PTW2	PTW1	PTW0
DDSADR	DDSADR7	DDSADR6	DDSADR5	DDSADR4	DDSADR3	DDSADR2	DDSADR1	DDSADR0
DDSDATA	DDSDATA7	DDSDATA6	DDSDATA5	DDSDATA4	DDSDATA3	DDSDATA2	DDSDATA1	DDSDATA0

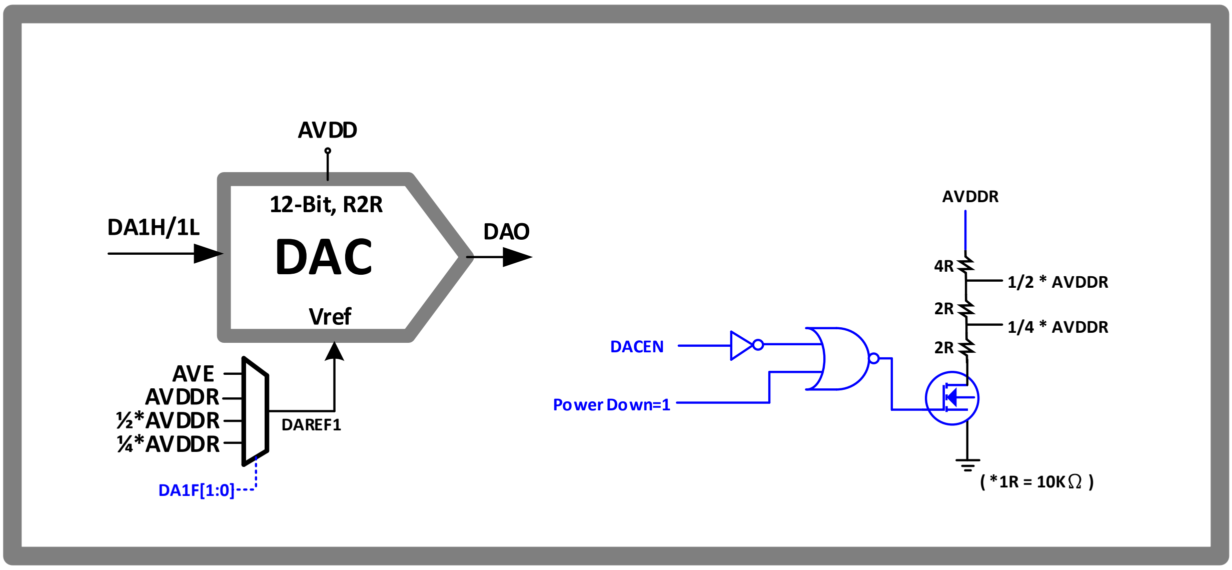
27 Blood 触发器

Blood 触发电路是血糖仪省电电路。

注：关于 Blood 触发参考样本代码和工程方案，请联系 SONIX 测量系统设计部。

28 DAC

DAC 结构的分辨率为 12 位。DAC 输出电路在 OP1&OP2 操作放大器中建立，以提高模拟输出驱动电流。当 DAEN=0 时，禁用 DAC 函数。当 DAEN=1 时，启用 DAC 功能。DAC 输出信号连接到 IO pad“DAO”，和内部 OP1P/OP2P 引脚（由 OP1PSW/OP2PSW 开关控制）。



28.1 DAC 控制寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAM	DAEN	-	DAF1	DAF0	DAB11	DAB10	DAB9	DAB8
DABL	DAB7	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB8

28.2 DAM 寄存器 (0xAB)

Bit	Field	Type	Initial	Description
7	DAEN	R/W	0	DAC 功能控制位。 0：禁止； 1：使能。
5..4	DAREF	R/W	0	DAC 参考源。 00：AVE； 01：AVDDR； 10：1/2 AVDDR； 00：1/4 AVDDR。
3..0	DAB[11:8]	R/W	0	12 位 DAC 设置数据的高字节[11:8] 。

28.3 DABL 寄存器 (0xAC)

Bit	Field	Type	Initial	Description
7..0	DAB[7:0]	R/W	0	12 位 DAC 设置数据的低字节[7:0]。

29 In-System Program

SN8F5950 内置 128KB 程序存储（IROM），均分为 255 页（每页 512 字节）。在线程序设计就是通过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。

SN8F5950 内置可编程（ISP）FLASH 存储器，方便升级存储代码。通过 SONIX 8051 MCU 可编程接口 OR 应用代码进行 FLASH 存储器编程。SN8F5950 提供给用户一个安全选项，防止未经授权访问存储器信息。ISP FLASH ROM 提供给用户一个简单的存储数据方式。ISP 是一种内存映射概念，即将 RAM 数据映射到 FLASH ROM 中，选择 ROM/RAM 地址，执行 ROM 编程命令（PECMD）。对 PERAMCNT 控制的字进行编程后，PERAML/PERAMCNT 数据将被编程成到地址 PEROML/PEROMH/PEROMHH 中。

RAM Address	RAM (byte) bit7 ~ bit0		ROM Address	Flash ROM (byte) bit7 ~ bit0
X	DATA0		Y	DATA0
X+1	DATA1		Y+1	DATA1
X+2	DATA2	=========>	Y+2	
X+3	DATA3		Y+3	
...			...	
...			...	
X+N	DATAN		Y+M	DATAN

在 FLASH 编程 OR 擦除操作时。暂停 MCU，外围设备（定时器，WDT，I/O，PWM 等）保持运行状态。当 PECMD 置起，执行 ISP 程序和擦除操作，程序计数器停止，操作代码不能从 FLASH ROM 转储，指令停止，程序执行停止。此时硬件取决于 ISP 操作配置，自动进行 FLASH ROM 擦除和编程。ISP 操作完成后，硬件释放系统时钟，程序计数器运行，系统返回上一个操作模式。建议 ISP 操作后增加两条“NOP”指令。

- ISP FLASH ROM 编程时间 = 35us …… 1- byte；

……

- ISP FLASH ROM 编程时间 = 530us …… 64- byte；
- ISP FLASH ROM 擦除时间 = 20us …… 1-扇区，512 - byte；

29.1 页擦除

ISP FLASH ROM 擦除操作是将 FLASH ROM 内容清除位空片状态“1”。擦除 ROM 长度为 512-byte，有 ROM 页限制。ISP FLASH ROM 擦除 ROM 映射如下：

<table>
<tr><th colspan="2">ISP ROM MAP</th><th>ROM address bit 0~ bit7(hex)=0x00</th></tr>
<tr><td rowspan="8">ROM address bit8~bit16(hex)</td><td>0000</td><td>这些页包括复位向量和中断向量。强烈建议保留该区，不做ISP擦除动作。</td></tr>
<tr><td>0200</td><td>一个 ISP 程序页</td></tr>
<tr><td>0400</td><td>一个 ISP 程序页</td></tr>
<tr><td>…</td><td>一个 ISP 程序页</td></tr>
<tr><td>10000</td><td>一个 ISP 程序页</td></tr>
<tr><td>10200</td><td>一个 ISP 程序页</td></tr>
<tr><td>…</td><td>一个 ISP 程序页</td></tr>
<tr><td>1FE00</td><td>该页包括ROM保留区。强烈建议保留该区，不做ISP擦除动作。</td></tr>
<tr><td rowspan="6">Data memory bit8~bit17(hex)</td><td>20800</td><td>一个 ISP 程序页</td></tr>
<tr><td>…</td><td>一个 ISP 程序页</td></tr>
<tr><td>20A00</td><td>一个 ISP 程序页</td></tr>
<tr><td>20C00</td><td>一个 ISP 程序页</td></tr>
<tr><td>…</td><td>一个 ISP 程序页</td></tr>
<tr><td>21800</td><td>一个 ISP 程序页</td></tr>
</table>

ISP FLASH ROM 擦除限制页擦除长度 512-byte。但是第一个 512-byte FLASH ROM（0x0000 – 0x01FF）存储复位向量和和中断向量，和最后一页 128-byte FLASH ROM（0x1FE00 – 0x1FFFF）包括 ROM 保留区，这两个区我们强烈建议不能执行 ISP FLASH ROM 擦除操作。其余 FLASH ROM 区域（0x0200 – 0x1FDFF）包括 254 页作为 ISP FLASH ROM 擦除操作区域。

ISP FLASH ROM 擦除第一步为 ROM 页位置的地址，此地址必须是页面区域的首地址，例如 0x0200，0x0400…0x4600，0x4800，0x4A00。PEROML[7:0]，PEROMH[7:0]，PEROMHH[0]定义 FLASH ROM 的目标起始地[16:0]，将起始地址写到 PEROMHH/PEROMH/PEROML 寄存器，随后写入 0xC3 到 PECMD 寄存器，系统开始执行 ISP FLASH ROM 擦除操作。

除去 Main code ISP (0x0200~0x1FDFF) 254-page，SN8F5950 内置 Data memory (20800~21600) 8-page。 Data memory 与 main code 的擦除功能一样。

29.2 页编程

ISP FLASH ROM 编程操作是通过程序写入数据到 FLASH ROM 中。程序 ROM 不限制写入的 ROM 地址和长度，但是限制一个页面 64-byte 长度。在同一时间 ISP FLASH ROM 编程操作可以是 1-byte ~ 64-byte，但是这些 words 必须在同一页。ISP FLASH ROM 编程 ROM 映射如下：

ISP ROM MAP		ROM address bit 0~ bit5(hex) = 0x00
ROM address bit6 ~ bit16(hex)	0000	这些页包括复位向量和中断向量。强烈建议保留该区，不做ISP擦除动作。
	0040	
	…	
	01E0	
	0200	一个 ISP 程序页
	0240	一个 ISP 程序页
	…	一个 ISP 程序页
	10000	一个 ISP 程序页
	10020	一个 ISP 程序页
	…	一个 ISP 程序页
	1FE00	该页包括ROM保留区。强烈建议保留该区，不做ISP擦除动作。
	…	
	1FFE0	
Data memory bit6~bit17(hex)	20800	一个 ISP 程序页
	20840	一个 ISP 程序页
	..	一个 ISP 程序页
	20A00	一个 ISP 程序页
	20A40	一个 ISP 程序页
	…	一个 ISP 程序页
	217E0	一个 ISP 程序页

ISP FLASH ROM 编程限制页编程长度 64-byte。但是第一个 256-byte FLASH ROM（0x0000 – 0x00FF）存储复位向量和和中断向量，和最后一页 128-byte FLASH ROM（0x1FE00 – 0x1FFFF）包括 ROM 保留区，（0x1FE00 – 0x1FFFF）我们强烈建议不能执行 ISP FLASH ROM 编程操作。其余 FLASH ROM 区域（0x0200 – 0x1FDFF）作为 ISP FLASH ROM 编程操作区域。

ISP FLASH ROM 编程操作是一种简单的内存映射操作。第一步规划一个 RAM 区域存储编程数据，并为 ISP RAM 寻址保存 RAM 地址；第二部是规划一个 ROM 区域，将 RAM 区域中的数据通过 ISP FLASH ROM 操作编程写到 ROM 区域。RAM 寻址通过 PERAML[7:0]配置 RAM 的起始地址。

ISP 编程长度可以是 1-byte ~ 64-byte，长度由 6 位格式的 PERAMCNT[5:0]寄存器控制。在执行 ISP ROM 编程之前，通过程序设置编程长度。PEROML[7:0]，PEROMH[7:0]和 PEROMHH[0]定义 FLASH ROM 的目标起始地[16:0]，将起始地址写到 PEROMHH/PEROMH/PEROML 寄存器，随后写入 0xC3 到 PECMD[7:0]寄存器，系统开始执行 ISP FLASH ROM 擦除操作；写入 0x5A 到 PECMD[7:0]寄存器，系统开始执行 ISP FLASH ROM 编程操作。如果编程长度超过 ISP FLASH ROM 的页编程长度，硬件完成对 ROM 页最后一个 word 编程后立即停止对 FLASH ROM 的编程。因此，正确的规划 ROM 地址和编程长度非常重要。

在开始页擦除和编程之前必须将下列配置设置完成，按以下步骤配置 ISP：

1. 保存编程数据到 IRAM，这些数据连续 64 字节。
2. 在 PERAM 中设置数据位置的起始地址。
3. 在 PEROM[15:6]中设置计划升级区域的起始地址（由 PEROMHH/PEROMH/PEROML 寄存器设置）。
4. 在 PERAMCNT 中设置 ISP 编程长度。
5. 写入 0xC3 到 PECMD[7:0]，触发 ISP 页擦除功能。
6. 写入 0x5A 到 PECMD[7:0]，触发 ISP 页编程功能。
7. 写 NOP 指令两次。
8. **在数据内存编程后，必须将 PEROMHH 的 BIT2 设置为 0。**

举例来说，假设程序存储器（IROM，0x4600 – 0x463F）为计划升级区域，内容已经存入 IRAM 地址 0x40 – 0x7F。为了执行 In-System Program，只要写入开始 IROM 地址 0x4700 到 EPROMH/EPROML 寄存器，然后指定缓存器开始地址 0x40 到 EPRAM 寄存器，在 PERAMCNT 中设置 ISP 编程长度，随后写入 0xC3 到 PECMD[7:0]寄存器，之后写入 0x5A 到 PECMD[7:0]寄存器，复制缓存器的数据到（IROM，0x4600 – 0x463F）。

29.3 In-system Program 寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERAM	RAM7	RAM6	RAM5	RAM4	RAM3	RAM2	RAM1	RAM0
PEROMHH	-	-	-	-	-	DFLASH	-	ROM16
PEROMH	ROM15	ROM14	ROM13	ROM12	ROM11	ROM10	ROM9	ROM8
PEROML	ROM7	ROM6	PEBYTE 5	PEBYTE 4	PEBYTE 3	PEBYTE 2	PEBYTE 1	PEBYTE 0
PECMD	CMD7	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
PAGESEL	-	-	-	-	-	-	PAGESEL1	PAGESEL0

PECMD Register (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28.4范例code。

* 禁止 PECMD 中写入任何其它数据。

PEROMHH Register (0x9C)

Bit	Field	Type	Initial	Description
0	PEROM[16]	R/W	0x00	编程页（IROM）的第一个地址（16th）。
2	DFLASH	R/W	0x00	数据 Flash 控制位。 1：数据 flash 内存(0x20800~0x219FF)； 0：Code memory

PEROMH Register (0x96)

Bit	Field	Type	Initial	Description
7..0	PEROM[15:8]	R/W	0x00	编程页（IROM）的第一个地址（15th– 8th bit）。

PEROML Register (0x95)

Bit	Field	Type	Initial	Description
7..6	PEROM[7:6]	R/W	00	编程页（IROM）的第一个地址（6th– 7th bit）。
5..0	PEBYTE[5:0]	R/W	0x00	页面字节地址。

*禁止在线程序设计可避免误触发 ISP 功能。

PERAM Register (0x97)

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

PAGESEL Register (0x9B)

Bit	Field	Type	Initial	Description
1..0	PAGESEL[1:0]	R/W	0x01	在 Banking 规划中提供一个附加的程序内存地址 MEMADDR[16:15]。注意默认值为 1，用于复位后 PAGESEL 寄存器不写入时，提供正确的地址生成（8000H 的逻辑地址等于物理地址）。这个值不能使用 0，它会导致 Bank 区域（逻辑地址 8000H~FFFFH）在共用 Bank（0000H~7FFFH）在物理上重叠。

29.4 示例代码

The 下面的示例程序展示了如何执行 ISP。

```c
#include <intrins.h>    // for _nop_
#include <SN8F5959.h>

#include "GenericTypeDefs.h"

void ISPSetROMAddr(UINT32 u32addr);
void ISPSetRAMAddr(UINT8 u8addr);
void ISPWritePage(void);
void ISPWriteByte(void);
void ISPPageErase(void);
void ISPExecute_Main_Page_Write(UINT32 Start_addr);
void ISPExecute_Main_Byte_Write(UINT32 Start_addr,UINT8 data_byte);
void ISPExecute_Main_Page_Erase(UINT32 Start_addr);
void ISPExecute_Inf_Page_Write(UINT32 Start_addr);
void ISPExecute_Inf_Byte_Write(UINT32 Start_addr,UINT8 data_byte);
void ISPExecute_Inf_Page_Erase(UINT32 Start_addr);


void ISPSetROMAddr(UINT32 u32addr)
{
    // set ROM addr
    PEROML = ((u32addr & 0x000000FF));
        // ISP Target Start ROM address Low byte
    PEROMH = ((u32addr & 0x0000FF00)>>8);
        // ISP Target Start ROM address High byte
    PEROMHH = ((u32addr & 0x00030000)>>16);
        // ISP Target Start ROM address High High byte
    _nop_();
}


void ISPSetRAMAddr(UINT8 u8addr)
{
    // set RAM addr
    PERAM = u8addr;
}


void ISPWritePage(void)
{
    PERAMCNT = 0X3F;     //WR 64 Byte Count

    // execute Byte Write * 64
    if(EAL == 0) {
        PECMD = 0x5A;    //page-program
        _nop_(); _nop_();
    }
    else {
        EAL = 0;
        PECMD = 0x5A;    //page-program
        _nop_(); _nop_();
        EAL = 1;
    }
}
```

```
void ISPWriteByte(void)
{
    PERAMCNT = 0;     //WR 1 Byte Count
    // execute Byte Write
    if(EAL == 0) {
        PECMD = 0x5A;    //page-program
        _nop_(); _nop_();
    }
    else {
        EAL = 0;
        PECMD = 0x5A;    //page-program
        _nop_(); _nop_();
        EAL = 1;
    }
}

void ISPPageErase(void)
{
    PERAMCNT = 0X00;
    // execute Page Erase
    if(EAL == 0) {
        PECMD = 0xC3;    //Page-Erase
        _nop_(); _nop_();
    }
    else {
        EAL = 0;
        PECMD = 0xC3;    //Page-Erase
        _nop_(); _nop_();
        EAL = 1;
    }
}



// Other Sample please download form SONiX SN8F5959 // Website
```

30 CRC

循环冗余校验(CRC)计算用于从 8 位数据字和一个生成多项式中得到一个 CRC 码。在其他应用程序中，基于 CRC 的技术用于验证数据传输或存储的完整性。CRC 计算电路在运行时帮助计算软件的信号，与链接时生成的参考信号进行比较，并存储在给定的内存位置。

- 支持

  CRC-16 多项式：X16+X15+X2+1

  CRC-16-CCITT 多项式：X16+X12+X5+1
- 处理 8 位数据大小
- 单输入/输出 8 位数据寄存器
- 输入缓存器，避免在计算中总线停止
- 32KB ROM 计算时间为 32ms

30.1 CRC 计算进程

CRC 支持 EEPROM 自动计算模式和数据计算模式这两种计算模式，用于获取 CRC 代码。

EEPROM 自动计算模式进程：

- 支持通过 CRCROM 位选择 CRC 执行范围。如果 CRCROM = 0，CRC 执行范围为 0x0000~0xFFFF。反之 CRCROM = 1，CRC 执行范围为 0x0000~0x7FFF。
- 通过 CRCPOL 位选择 CRC 多项式。当 CRCPOL=0，选择 CRC-16-CCITT (X16+X12+X5+1)，反之选择 CRC-16(X16+X15+X2+1) 。
- 设置 CRCRST 位复位初始 seed 值/ CRC 数据缓存和 BUSY 位为 0。
- 设置 URCRCEN 位启动 CRC 计算和检测 BUSY 位。
- BUSY 位从 1 到 0 表示 CRC 计算完成。从 CRCDATH/L 中读取 CRC code。

数据计算模式进程：

- 通过 CRCPOL 位选择 CRC 多项式。当 CRCPOL=0，选择 CRC-16-CCITT (X16+X12+X5+1)，反之选择 CRC-16(X16+X15+X2+1)。
- 设置 CRCRST 位复位初始 seed 值/ CRC 数据缓存和 BUSY 位为 0。
- 将数据输入到 CRCDATL。
- 设置 URCRCEN 位启动 CRC 计算和检测 BUSY 位。
- BUSY 位从 1 到 0 表示 CRC 计算完成。从 CRCDATH/L 中读取 CRC code。

30.2 CRC 寄存器

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

CRCM Register (0xFFED)

Bit	Field	Type	Initial	Description
7	URCRCSTA	R/W	0	使能 16KB/32KB 用户 ROM 的 CRC 计算。 0：禁止； 1：使能。
6	BUSY	R/W	0	停止/完成操作。 0：CRC 计算完成； 1：CRC 计算进程中。
5	CRCRST	R/W	0	Reset 位。 0：无效果； 1：复位 CRC 计算电路。
4	CRCPOL	R/W	0	CRC 多项式。 0：CRC-16-CCITT； 1：CRC-16。
3..2	CRCROM [1:0]	R/W	0x00	CRC EEPROM 计算范围。 00：0x0000~0xFFFF (1/2 ROM)； 01：0x0000~0x7FFF (1/4 ROM)； 10：0x0000~0x1FDFD。
0	CRCEN	R/W	0	使能 CRC IP。 0：禁止； 1：使能。

CRCDATL Register (0xFFEE)

Bit	Field	Type	Initial	Description
7..0	CRCDAT[7:0]	R/W	0x00	16 位 CRC 数据低字节。

CRCDATH Register (0xFFEF)

Bit	Field	Type	Initial	Description
7..0	CRCDAT[15:8]	R/W	0x00	16 位 CRC 数据高字节。

31 ILRC 自动校正

ILRC 自动校准功能是硬件校正，支持 16KHz ILRC。校正功能是由 CALEN 位启动。设置 CALEN 位后，系统开始校正 ILRC 频率至 16KHz。校正时间约为 20ms。当校准完成时，CALEN 位被重置为逻辑 0。校准完成后，如果 ILRC 校准成功，电路将设置 CALDONE 位。

31.1 Clock 微调寄存器

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LFRQH	-	-	-	-	-	-	-	FRQ8
LFRQL	FRQ7	FRQ6	FRQ5	FRQ4	FRQ3	FRQ2	FRQ1	FRQ0
FRQCMD	FRQCMD7	FRQCMD6	FRQCMD5	FRQCMD4	FRQCMD3	FRQCMD2	FRQCMD1	FRQCMD0
CLKCAL	CALEN	CALDONE	-	-	-	-	-	-

LFRQ Register (FRQH:0xFFE4, FRQL:0xFFE3)

Bit	Field	Type	Initial	Description
8..0	LFRQ[8:0]	R/W	00	系统 ILRC 校正值。

FRQCMD Register (0xFFF2)

Bit	Field	Type	Initial	Description
7..0	FRQCMD[7:0]	R/W	0x4B	0x4B：启动 ILRC 校准进程； 其它值： 保留。

CLKAL Register (0xFFE5)

Bit	Field	Type	Initial	Description
7	CALEN	R/W	0	ILRC 校正使能位。 0：禁止； 1：使能 (校正结束后自动清除)。
6	CALDONE	R/W	0	ILRC 校正结果。 0：校正失败； 1：校正成功。ILRC 频率为 16KHz。

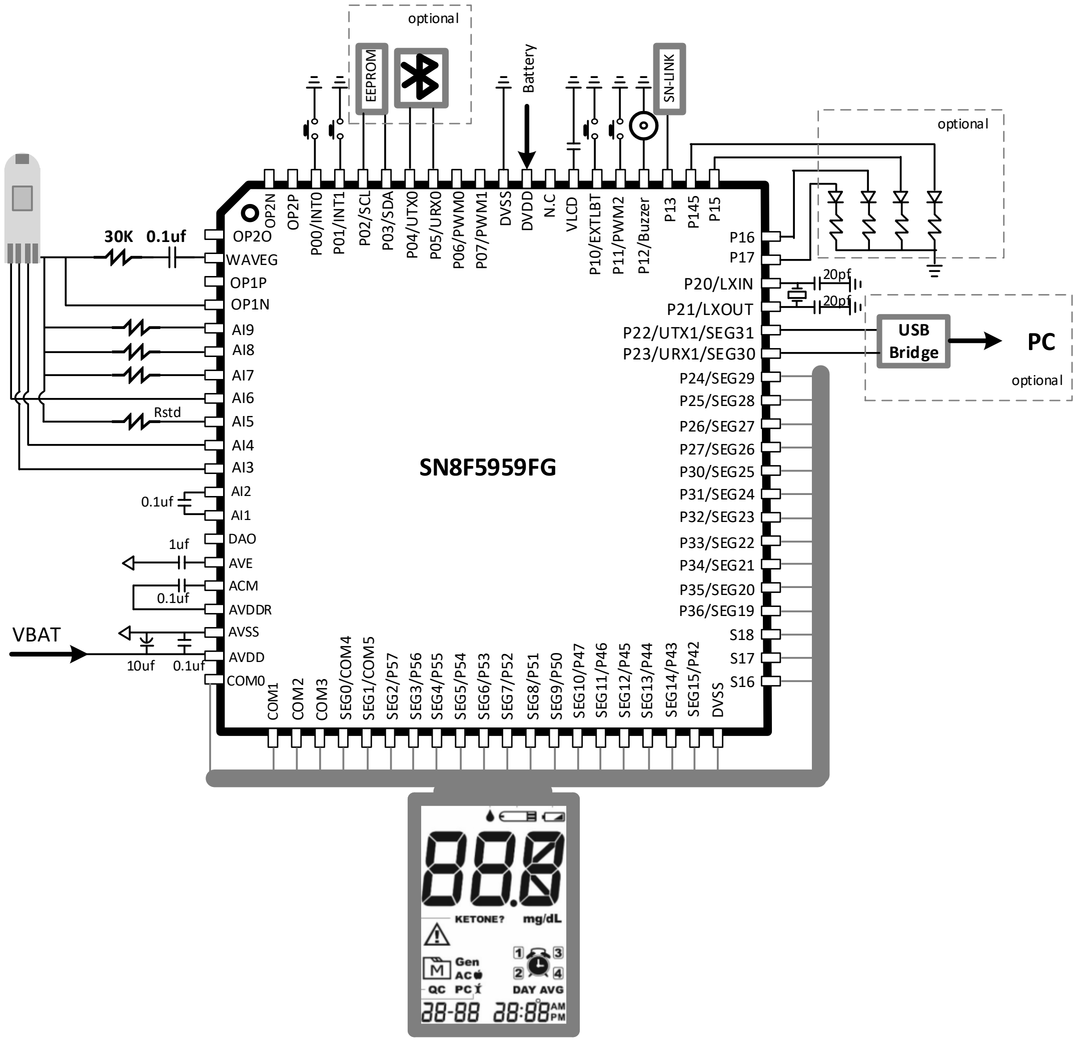
31.2 示例代码

下面的示例代码程序演示了如何执行 ILRC 校正。

```
#define  __XRAM_SFR_H__      //for SRAM SFR
#include <intrins.h>         //for _nop_
#include <SN8F5959.h>

void ILRCCalStart (void)
{
   // Enable ILRC calibration function
   CLKCAL |= 0x80;

   // Start ILRC calibration function
   FRQCMD = 0x4B;
}

void Main (void)
{
  ILRCCalStart();
  while((CLKCAL & 0x40)!= 0x40){ //ILRC 16K Calibration pass
  }
      while (1) {
    WDTR = 0x5A;              // clear watchdog if watchdog enable
      // To Do ...
  }
}
```

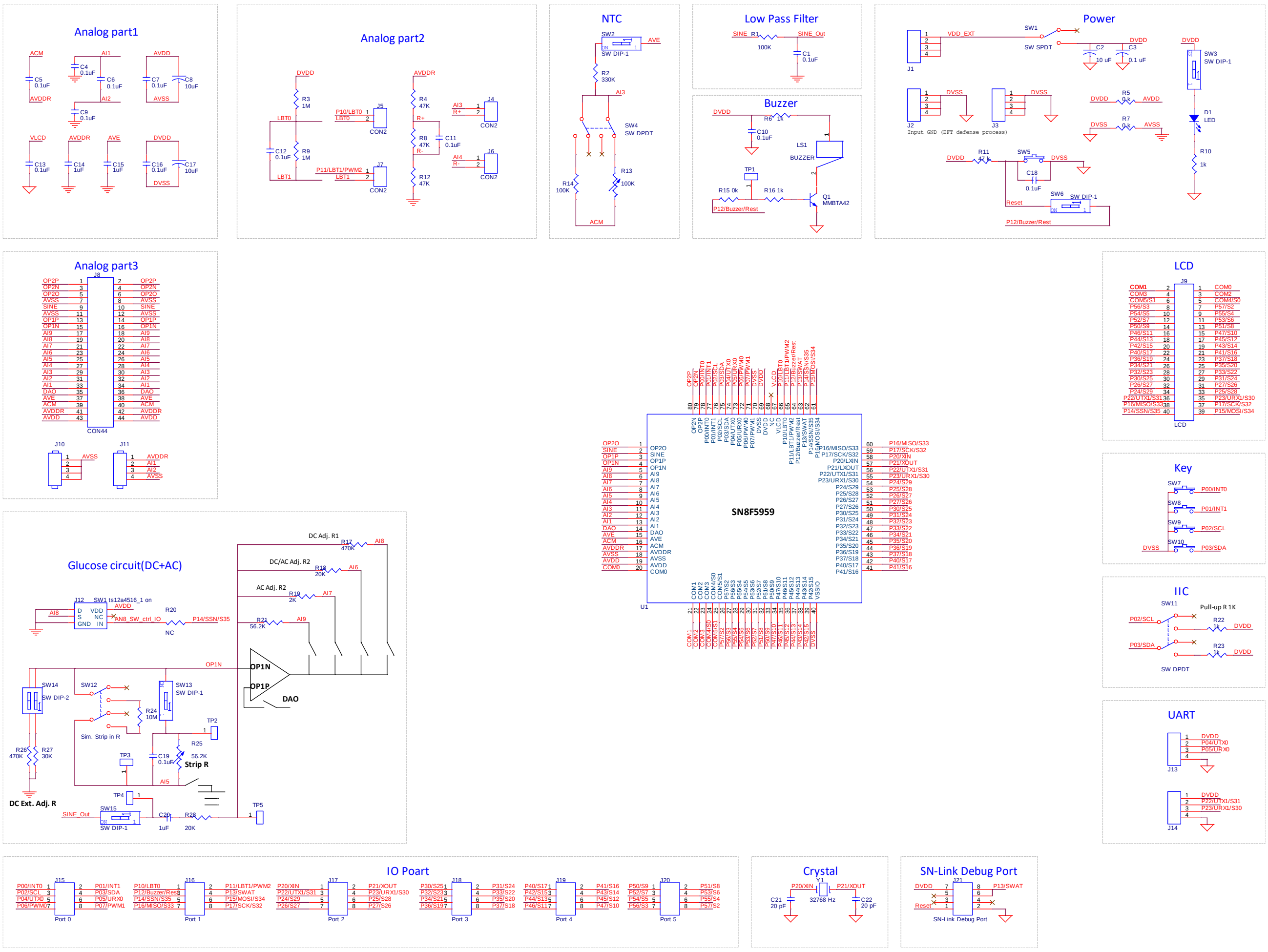
32 应用电路

32.1 血糖仪应用电路



* 注: DVDD/AVDD电容应该尽可能的接近IC的引脚。

32.2 Starter-Kit 电路



33 电气特性

33.1 极限参数

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

33.2 系统操作特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
VDD	运行电压	fcpu = 1MHz	1.8		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 = 2V ~ 5.5V, fcpu = 0.25MHz		0.5		mA
		VDD = 2V ~ 5.5V, fcpu = 1MHz		0.7		mA
		VDD = 2V ~ 5.5V, fcpu = 4MHz		1		mA
		VDD = 2V ~ 5.5V, fcpu = 8MHz		1.3		mA
		VDD = 2V ~ 5.5V, fcpu = 16MHz		1.8		mA
I_{DD2}		VDD = 5V		2.0		μA
		VDD = 5V, RTC enable (32768Hz Crystal)		3.0		μA
I_{DD3}	STOP 模式功耗	VDD = 5V		0.4		mA
f_{IHRC}		VDD = 2.0V ~ 5.5V, Temp. 25°C	-1%	32	+1%	MHz
		VDD = 2.0V to 5.5V, Temp. -40°C to 105°C	-2%*	32	+2%*	MHz
f_{ILRC}	IDLE 模式功耗 (fcpu = 0.25MHz)	VDD = 3.0V @ 20°C	12	16	24	KHz
V_{LVD} 14	LVD14 检测电压	25°C	1.35	1.38	1.41	V
		-40°C to 105°C		1.38		V
V_{LVD} 10	LVD10 检测电压	25°C	0.95			V
		-40°C to 105°C	0.95			V
ESD	人体模式 (HBM)			4	8	KV
	机械模式(MM)			200		V
LU	Latch Up		-125		+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。

33.3 GPIO 特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
V_{IL}	低电平输入电压		VSS		0.3VSS	V
V_{IH}	高电平输入电压		0.7VDD		VDD	V
I_{LEK}	I/O 口输入漏电流	V_{IN} = VDD			2	μA
R_{UP}	上拉电阻	VDD = 3V	62	100	125	kΩ
		VDD = 5V	25	50	75	kΩ
I_{OH}	I/O 输出拉电流	VDD = 3V, V_O = VDD-0.5V		10		mA
I_{OL}	I/O 灌电流	VDD = 3V, V_O = VSS+0.5V		17		mA

33.4 ADC 特性

SYM	DESCRIPTION	Condition	MIN.	TYP.	MAX.	UNIT
IADC	操作电流	Run mode @ 2.4V	-	200	-	uA
IPDN	睡眠电流	Stop mode @ 2.4V	-	0.1	-	μA
FSMP	转换率(WR)	ADC Clock=250KHz, OSR=32768	-	7.6	-	Hz
		ADC Clock=333KHz, OSR=64	-	5.2	-	kHz
TADCSTL	ADC 设置时间	3*(1/WR), if WR=61Hz, TADCSTL = 3*16.4ms = 49.2ms		3		WR
VREF	参考电压输入电压	External VREF Input Range (R+ - R-)	0.3		1.35	V
		Internal VREF Input Range.	0.3		1.35	V
VR	ADC 参考信号绝对电压	GR=1, R+ and R- absolutely input Voltage	0.3		AVDD R-1V	V
		GR=0, R+ and R- absolutely input Voltage	0.3		AVDD R-1V	V
VAI	ADC 输入信号绝对电压	GX=1, AI absolutely input Voltage	0.3		AVDD R-1V	V
		GX=0, AI absolutely input Voltage	0		AVDD R-1V	V
Vx	PGIA 输出信号绝对电压	X+ and X- absolutely output Voltage	0.3		AVDD R-1V	
	PGIA 增益比*	VDD=2.4V , PGIA=x128	-3		+3	%
DNL	微分非线性	ADC range ± 131072 x 0.9. (0.9 x Vref，18-bits)		± 2		LSB

INL	积分非线性	ADC range ± 131072 x 0.9. (0.9 x Vref , 18-bits)		± 4		LSB
NMC	无丢码	ADC range ± 131072 x 0.9. (0.9 x Vref , 18-bits)		18		bit
NFB	Noise free bits	Gain:1, Vref:0.8V, OSR:32768, Input-short		18.5		bit
		Gain=128, Vref=0.8V, OSR:32768, Input-short		15.5		bit
ENOB	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
VAIN	ADC 输入微分范围	ADC input signal, signal after PGIA application	0.3		1.44	V
TDrift	ADC 温度漂移	AVDDR = 2.4V		30		PPM/°C

33.5 Regulator 特性

ACM						
模拟通用电压	V_{ACM}	Output voltage	1.1	1.2	1.3	V
		Output Voltage drift vs. Temperature △10°C	-	±0.1	-	%
		Output Voltage drift vs. AVDD (2V~5.5V)	-	±0.1	-	%
VACM 驱动能力	I_{SRC}	Only for ADC use	-	-	10	uA
VACM 反向能力	I_{SNK}	Only for ADC use	-	-	1	mA
AVDDR						
Regulator 输出电压 AVDDR	V_{AVDDR}	AVDDR Output Range = 2.2V, 2.4V, 2.8V, 3.2V	2.2	-	3.2	V
		AVDDR set 2.4V	2.35	2.4	2.45	V
		AVDDR set 2.8V	2.75	2.8	2.85	V
		AVDDR set 3.2V	3.15	3.2	3.25	V
		AVDDR Load regulation △5mA	V_{AVDDR} ± 0.05V			V
		Output Voltage drift vs. Temperature △10°C	-	±0.1	-	%
		Output Voltage drift vs. 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
		AVE				
Regulator 输出电压 AVDDR	V_{AVE}	AVE Output Range = 1.5V, 2.0V	1.5		2.0	V
		AVE Load regulation △5mA			20	mV
		Output Voltage drift vs. Temperature △10°C		±0.1	-	%
		Output Voltage drift vs. AVDD (2V~5.5V)	-	±0.1	-	%
AVE 驱动/反向电流	I_{AVE}				7.5	mA

33.6 OPA 特性

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
运行电压	Voper	Power source from AVDD	2.0		5.5	V
输入偏移电压	VOS				±3	mV
输入偏移 w/temp.				10		uV/°C
运行电流	Ioper	OP+ = 0V, Buffer Mode	-	300		uA
逻辑输入电压范围	Vin	OP+ / OP-	0.1		AVDD-0.1	V
逻辑输出电压范围	Vout	OPOUT	0.1		AVDD-0.1	V
输出短路电流	IOH/IOL	Unit Gain Buffer. Vo = AVDD-0.1, AVDD=3V		±5		mV
电源抑制比	PSRR	Voper=2.0~5.5V (AVDD)		80		dB
共模抑制比	CMRR	Vcm=0~(AVDD-0.1)		70		dB
开环电压增益		Voper=2.0 ~ 5.5V (AVDD)	50	60		dB
相位幅度	∮m	CL=50Pf	50	60		degree
OP1 Gain 带宽	GB			1.4		MHz
OP2 Gain 带宽	GB			1.4		MHz
斜率	SR	10% to 90%		1.2		V/uS
输出高电压	VOH		AVDD-0.15		AVDD	V
输出低电压	VOL		AVSS			V
开启时间	TON			10	20	uS
关闭时间	TOFF				1	uS

33.7 DAC 特性

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
运行电压	Voper	Power source from AVDD	2.0	-	5.5	V
DAREF 内部输入电压 (异常电压)	VDRIN	DAC VREF from source internal 1/2*AVDDR	-2%	1%	+2%	%
		DAC VREF from source internal 1/4*AVDDR	-2%	1%	+2%	%
偏移电压					10	mV
正极电流	IDAC	Supply current @ no loading		300		uA
相对精度	INL			±2	±4	LSB
微分非线性	DNL	Monotonic		±2	±4	LSB
输出电压范围	VDAO		0.03		AVDD-0.03	V
DAC 负载电流		Buffer stage output (Drive / Sink)	-1		1	mA
DAC 最大负载电容					100	pF
Full-scale 设置时间				40	80	uS
DAC 启动时间				15	30	uS

33.8 Comparator 特性

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
VILBT	内部低电压检测	条件： LBTSEL [3:0] = 0000, VLBT=2.2V	2.15	2.2	2.25	V
		条件：LBTSEL [3:0] = 0100, VLBT=3V	2.55	2.6	2.65	
		条件：LBTSEL [3:0] = 0111, VLBT=2.9V	2.85	2.9	2.95	
VELBT	外部低电压检测	条件： LBTSEL [3:0] = 1xxx, VLBT=P10 input.	1.75	1.8	1.85	
VHY	比较器迟滞窗口			50	-	mV
ICOMP	电流消耗	AVDD = 3V		50	-	uA

33.9 LCD 特性

SYM.	DESCRIPTION	PARAMETER	MIN.	TYP.	MAX.	UNIT
ICLCD	C-Type LCD 工作电流	VDD：2.0~4.5V，无面板低功耗模式(BGM=0,DUTY[1:0]=00)		9		uA
		VDD：2.0~4.5V, 1/3 bias，无面板低功耗模式(BGM=0,DUTY[1:0]=01)		10		
		VDD：2.0~4.5V, 1/3 bias，无面板低功耗模式(BGM=0,DUTY[1:0]=10)		12		
		VDD：2.0~4.5V, 1/3 bias，无面板低功耗模式(BGM=0,DUTY[1:0]=11)		17		
		VDD：2.0~4.5V, 1/3 bias，无面板低功耗模式 (BGM=1,DUTY[1:0]= xx)		127		
VLCD	C-Type VLCD 输出电压	VDD：2.0~4.5V, VLCD set 2.6V (VCP [3:0]=0000)	2.4	2.6	2.8	V
		VDD：2.0~4.5V, VLCD set 3.2V (VCP [3:0]=0110)	3.0	3.2	3.4	

33.10 Flash Memory 特性

SYM.	Parameter	Test Condition	Min	TYP	MAX	UNIT
Vdd	供电电压		1.8		5.5	V
Ten	持续时间	25°C		*100K		cycle
Iwrt	写功耗	25°C		3.5		mA
Twrt	写时间	Write 1 page=512 bytes, 25°C		0.6		ms

* 未验证的设计参考。

33.11 特殊条件或环境使用建议

条件 / 环境	建议
条件 (1) 当单片机 I/O 连接到其他电路时，出现负暂态电压，峰值电压超过-3V，单片机可能会发生锁存现象	在电源输入到单片机 DVDD 引脚之前连接 10 R 电阻，解决了 I/O 中出现异常暂态电压引起的锁存问题。
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。

34 指令集

本章对 SN8F5950 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。

34.1 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

34.2 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

34.3 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

34.4 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标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内存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内存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内存（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内存（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

34.5 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

34.6 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 所指定的地址
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

35 开发环境

调试接口（SWAT）与 GPIO 引脚 P1.3 共用，用于更新程序存储器（IROM）并配合开发环境。如果微控制器上电前连接了 SN-LI NK，该共用引脚保留为调试接口，但在上电顺序中没有检测到任何握手信号的话，该引脚将保留其它功能。

*注： P13 不能接任何负载 @Debug Mode。

35.1 最低要求

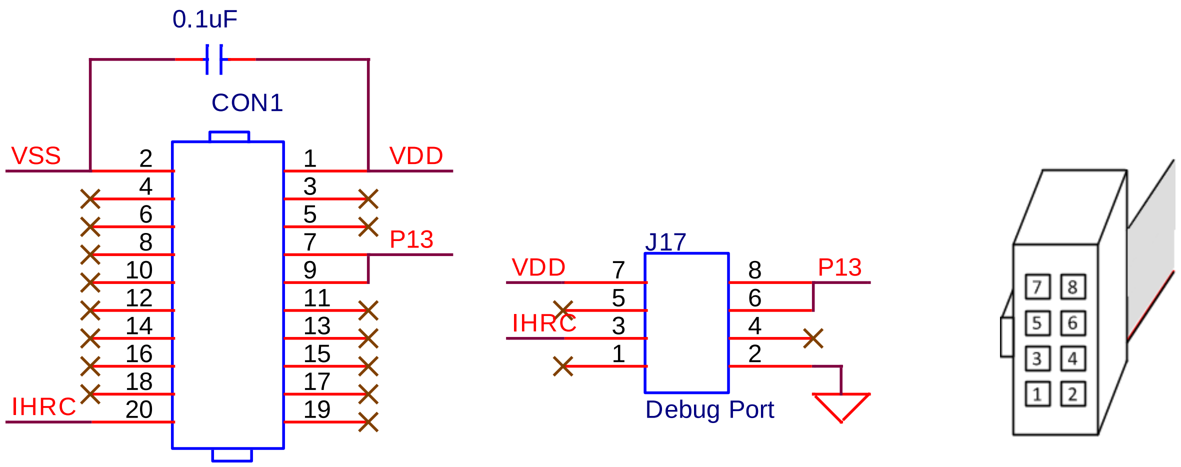
下面几项内容是建立合适的开发环境的基本要求，其兼容性已经经过验证，在后面的版本中可以很好的执行。SN-Link 的相关信息可以 SONiX 网址 www.sonix.com.tw 下载，Keil C51 可从 www.keil.com/c51 下载。

- SN-Link3 适配器，FW 版本升级为 1.xx；
- Keil C51 的 SN-Link 驱动器，版本 2.00.xx；
- Keil C51 版本 9.50a 和 9.54a，或更高。

35.2 硬件调试接口

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 SN8F5000 调试工具使用手册。）



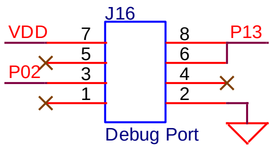
35.3 IHRC 校准

IC 量产无论是封装片还是 Dice，MCU 内部 32 MHz 时钟发生器 (IHRC)始终在 1%精度校准。在客户 COB 过程中中，由于 Dice 的应力变化，IHRC 频率偏移可能超过 1%。我们强烈建议保留 P02 和 P13 口，并预留给 Writer 再次编程和 IHRC 校准，以解决少量 IC 在 COB 进程后的 IHRC 偏移问题。

SN8F5950 系列提供 IHRC 频率偏移校准功能，而且还提供 Writer 的第 20 引脚为调试接口，用于 IHRC 校准。

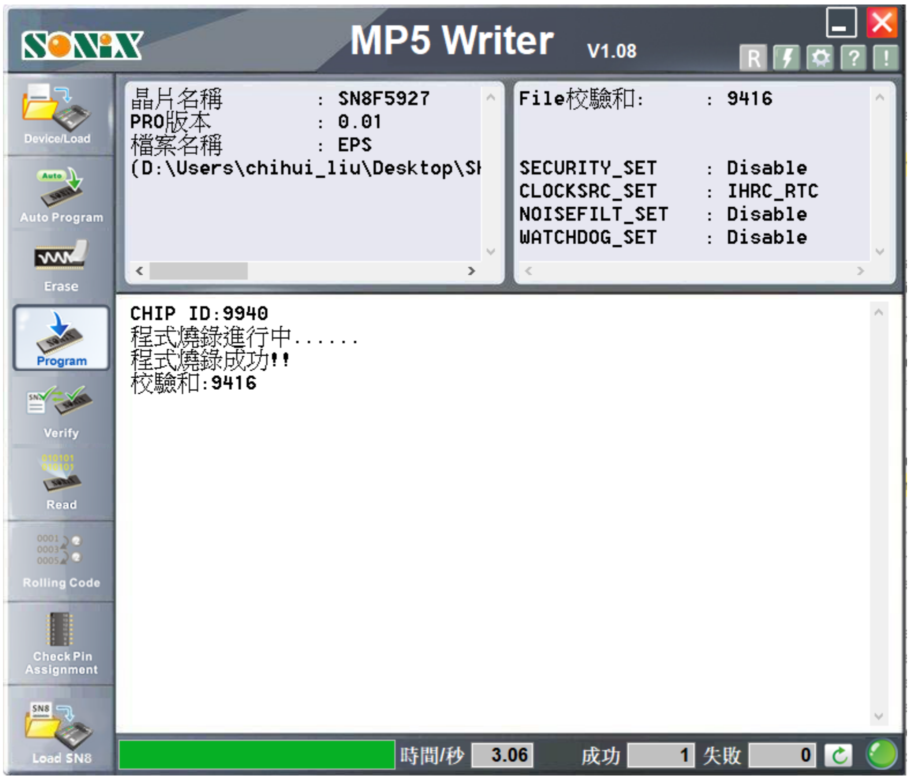
以下为 IHRC 校准步骤：

- 第一步：从 SONIX 网站下载 MP5-Writer (http://www.sonix.com.tw/article-tw-433-21531)
- 第二步：将 P02 与 Writer 的 20pin 相连



Writer Interface

- 第三步：打开 MP5-Writer.exe → Device/Load SN8 → Select hex file format → Auto Program



**注：IHRC校准功能包含在Auto Program 功能里。*

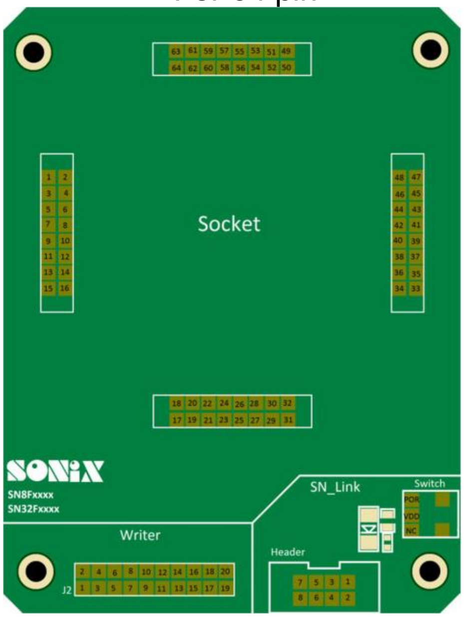
36 ROM 烧录引脚

SN-LINK 和 MP5 Writer 支持 SN8F5959 系列 Flash ROM 的擦除/烧录/校正。

- **SN-LINK：调试界面。**
- **MP5 Writer：SN8F5959 系列。**

36.1 MP5 Writer 转接板

For 64 pin



36.2 MP5 Writer 烧录引脚图

Writer 接口		MCU 引脚名称	SN8F5959F	SN8F5958F
引脚编号	引脚名称		MCU 引脚编号	MCU 引脚编号
1	VDD	VDD	19 / 56	16 / 49
2	GND	VSS	18 / 40 / 66	15 / 37
7,9	SWAT	P1.3	67	51
20	PDB	P0.2	76	60

36.3 SN-Link ISP 烧录

SN-Link ISP 烧录硬件和软件如下：

36.4 SN-LINK ISP 烧录引脚图

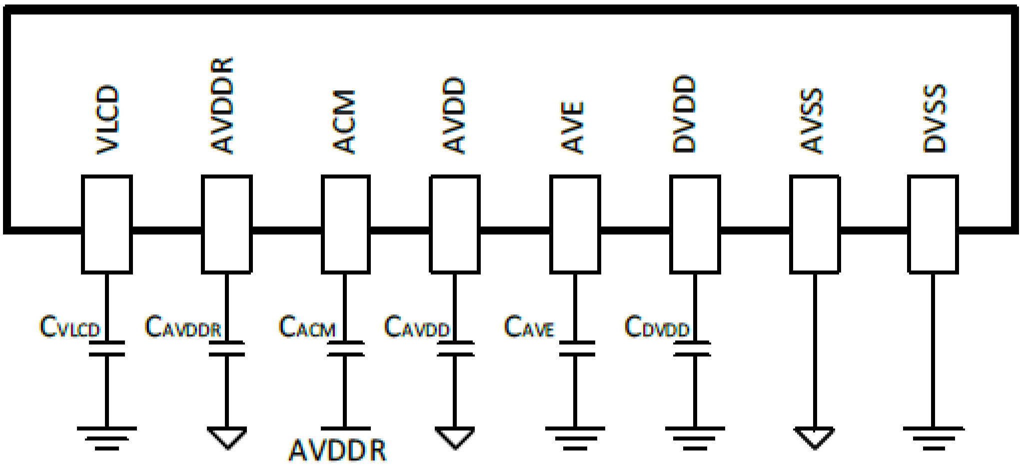
SN-LINK 接口		MCU 引脚名称	SN8F5959F	SN8F5958F
引脚编号	引脚名称		MCU 引脚编号	MCU 引脚编号
7	VDD	VDD	19 / 65	16 / 49
2	GND	VSS	18 / 40 / 66	15 / 37
6,8	SWAT	P1.3	67	51

37 模拟设置与应用

SN8F5950 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。

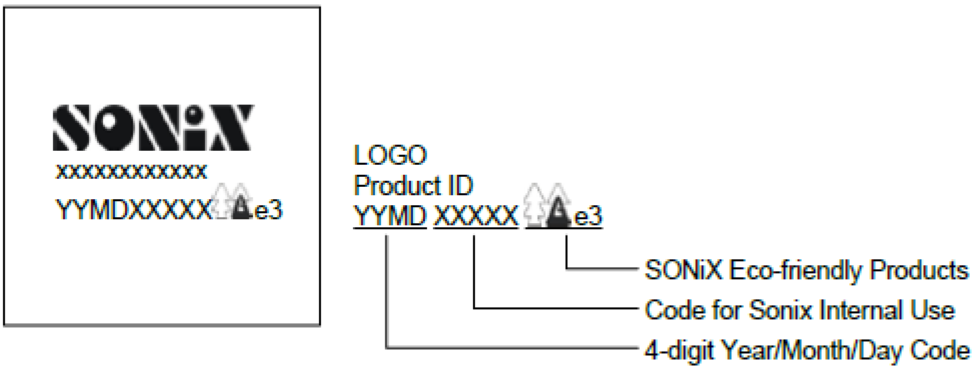
Charge pump 使能电容表：

Power	AVDDR	ACM	VLCD	AVDD	DVDD
	C_{AVDDR}	C_{ACM}	C_{VLCD}	C_{AVDD}	C_{DVDD}
CR2032 (2.4 ~ 5.5V)	0.47uf	0.1uf	0.1uf	10uf/104	10uf/104
2*AA/AAA Bat (2.4 ~ 5.5V)	1uf	0.1uf	0.1uf	10uf/104	10uf/104



38 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。



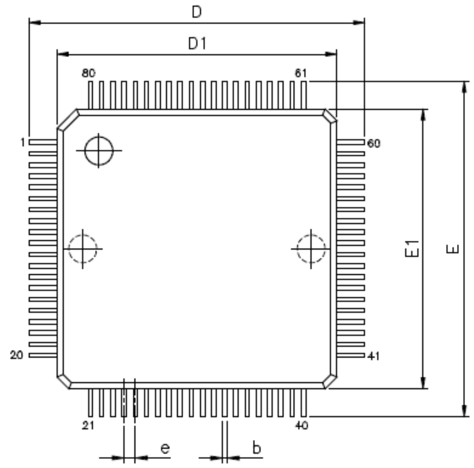
38.1 命名规格

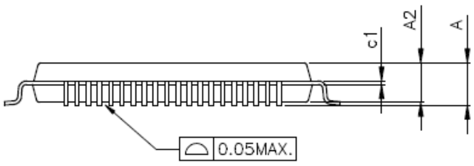
单片机全称	封装类型
S8F5959W	Wafer
SN8F5959H	Dice
SN8F5959FG	LQFP，80 脚，绿色封装
SN8F5958FG	LQFP，64 脚，绿色封装

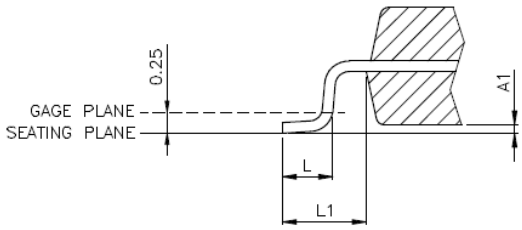
38.2 日期码

Year	18：2018 19：2019 20：2020 21：2021 et cetera
Month	1：January 2：February 3：March 4：April 5：May 6：June 7：July 8：August 9：September A：October C：November D：December
Date	1：01 2：02 3：03 4：04 5：05 6：06 7：07 8：08 9：09 A：10 C：11 D：12 E：13 F：14 G：15 H：16 J：17 K：18 L：19 M：20 N：21 P：22 Q：23 R：24 S：25 T：26 U：27 V：28 W：29 X：30 Y：31

38.3 LQFP 80 PIN

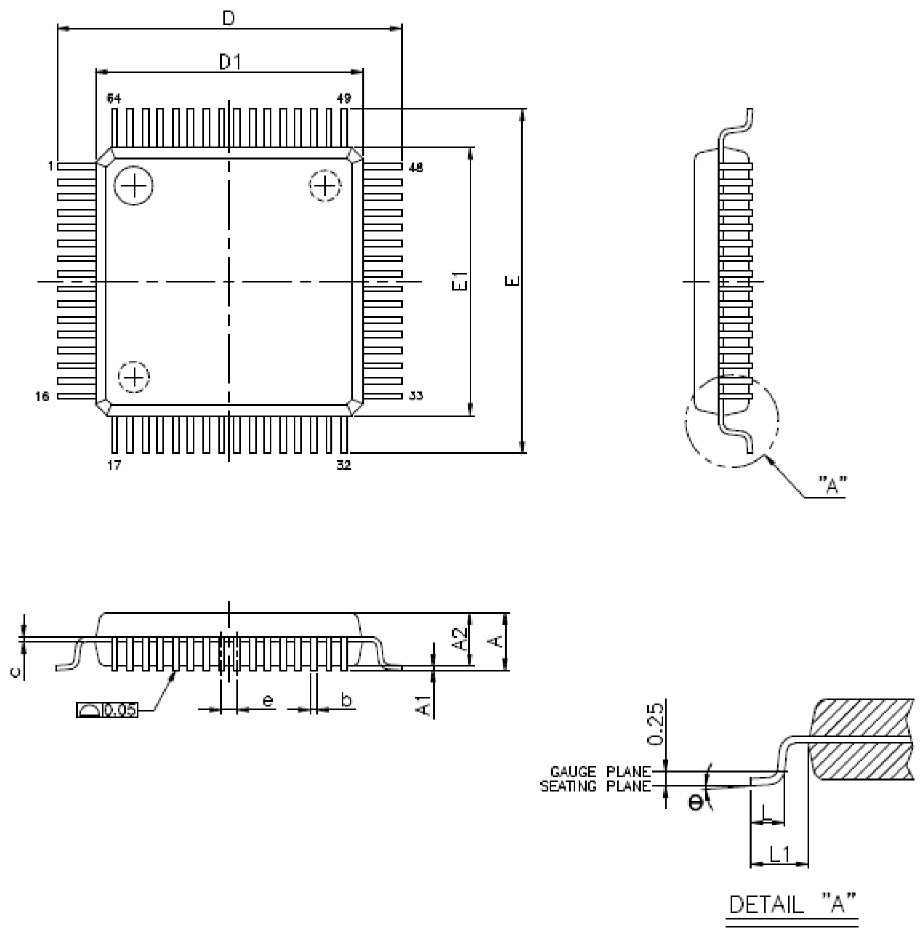






	Min	Typical	Max	Min	Typical	Max
		(inch)			(mm)	
A			0.063			1.6
A1	0.002		0.006	0.05		0.15
A2	0.053		0.057	1.35		1.45
c1	0.004		0.006	0.09		0.16
D		0.473 BSC			12 BSC	
D1		0.393 BSC			10 BSC	
E		0.473 BSC			12 BSC	
E1		0.393 BSC			10 BSC	
e		0.016 BSC			0.4 BSC	
B	0.005		0.009	0.13		0.23
L	0.018		0.030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

38.4 LQFP 64 PIN



	Min	Typical	Max	Min	Typical	Max
		(inch)			(mm)	
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	

SN8F5950 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