SH79F6481A 用户指南

-集成 24 路触摸按键输入和 PWM 的增强型 8051 微控制器

一、SH79F6481A I/O 用户指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 IO 特性为：

- 46 个双向 I/O 端口
- I/O 端口可与其它功能共享
- 部分第二功能可通过寄存器配置
- 2 个可选择的开漏极 I/O 口

SH79F6481A 提供 46 个位可编程双向 I/O 端口。端口数据在寄存器 Px 中。每个 I/O 口均有内部上拉电阻。端口控制寄存器（PxCRy）控制端口是作为输入或者输出。当端口作为输入时，每个 I/O 端口带有由 PxPCRy 控制的内部上拉电阻（x = 0-5，y = 0-7）。

2. 控制寄存器

IO 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	PxCRy（x=0-5，y=0-7）	IO 输入输出控制寄存器
	PxPCRy（x=0-5，y=0-7）	IO 上拉电阻控制寄存器
模块输出输入	Px.y（x=0-5，y=0-7）	IO 端口数据寄存器
数据指针选择	INSCON	特殊功能寄存器页选择

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注意：*在对 P5,P5CR,P5PCR 寄存器位于 BANK1，对其进行操作前，请先将 INSCON 中的 BSK0 位置 1。*

3. IO 设置

3.1. IO 输出设置

IO 作为输出模式，需要将其 PxCRy（x=0-5，y=0-7）设置为 1，此时若设置 Px.y（x=0-5，y=0-7）为 0，则 IO 输出低电平，低电平的值为 GND。若设置 Px.y（x=0-5，y=0-7）为 1，则 IO 输出高电平，高电平的值为 VDD。

3.2. IO 输入设置

IO 作为输入模式，需要将其 PxCRy（x=0-5，y=0-7）设置为 0，此时若设置 PxPCRy（x=0-5，y=0-7）为 0，则 IO 处于输入 Floating 状态，若设置 PxPCRy（x=0-5，y=0-7）为 1，则上拉电阻打

开，IO 处于输入高的状态。IO 输入的高电平为 0.8VDD，IO 输入低电平为 0.2VDD，具体参数请参考“SH79F6481A DATASHEET”。

3.3. 开漏极 I/O 设置

SH79F6481A 的 TWI 功能使能，相应口会自动被设置成 N 沟道开漏输出，通过设置 TWITOUT 中的 TWIPCR 位，打开内部上拉电阻。用户可根据实际应用情况选择外接或者打开内部上拉电阻。

4. 应用实例

4.1. 要求

将一个 0~5V 的三角波从 P3.0 输入，P3.0 的值通过 P5.0 输出

4.2. 例程

```
bit a;
void main(void)
{
    P3CR &= 0xFE;           //P3.0 输入
    P3PCR |= 0x01;          //P3.0 上拉电阻打开
    INSCON |= 0x40;         //切换至 BANK1
    P5CR |= 0x01;           // P5.0 输出
    INSCON &= 0xBF;         //切换至 BANK0
    while(1)
    {
        a = P3.0;
        INSCON |= 0x40;     //切换至 BANK1
        P5.0 = a;           // P5.0 输出值为 P3.0 输入值
        INSCON &= 0xBF;     //切换至 BANK0
    }
}
```

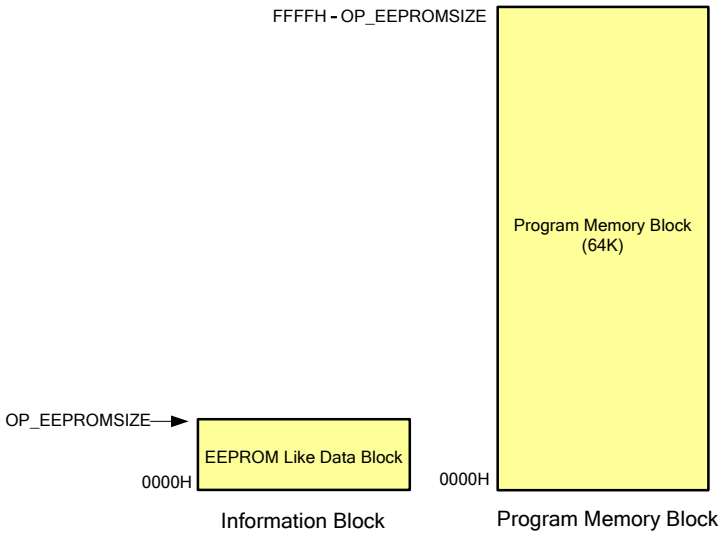
二、SH79F6481A Flash&EEPROM 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 Flash&EEPROM 特性为：

- Flash 存储器包括 128 X 512Byte 区块，总共 64KB
- EEPROM 存储器 0～4KB 代码选项可选
- 在工作电压范围内都能进行编程和擦除操作
- 在线编程（ICP）操作支持写入、读取和擦除操作
- 支持整体/扇区擦除和编程（SSP）
- 编程/擦除次数：程序区：至少 100,00 次
  类 EEPROM 区：至少 100,000 次
- 数据保存年限：至少 10 年
- 低功耗



SH79F6481A 为存储程序代码内置 64K 可编程 Flash（Program Memory Block），可以通过在线编程（ICP）模式和扇区自编程（SSP）模式对 Flash 存储器操作。每个扇区 512 字节。

SH79F6481A 还内置最大 4096 字节的类 EEPROM 存储区用于存放用户数据，每个扇区 512 字节，最多支持 8 个扇区。EEPROM 存储区位于 Flash 存储器，与程序存储区是共享的，举例说明：当 OP_EEPROMSIZE=0000 时，即 EEPROM 大小为 4KB，此时程序存储区的大小为 64KB-4KB=60KB；当 OP_EEPROMSIZE=0100 时，即 EEPROM 大小为 2KB，此时程序存储区的大小为 64KB-2KB=62KB。具体 EEPROM 大小选择详见在*代码选项* 章节。

2. 控制寄存器

Flash&EEPROM 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	XPAGE	编程用地址选择寄存器
	IB_OFFSET	编程用地址偏移寄存器
	IB_DATA	编程用数据寄存器
	IB_CON1	SSP 型选择寄存器
	IB_CON2	SSP 流程控制寄存器 1
	IB_CON3	SSP 流程控制寄存器 2
	IB_CON4	SSP 流程控制寄存器 3
	IB_CON5	SSP 流程控制寄存器 4
	FLASHCON	访问控制寄存器

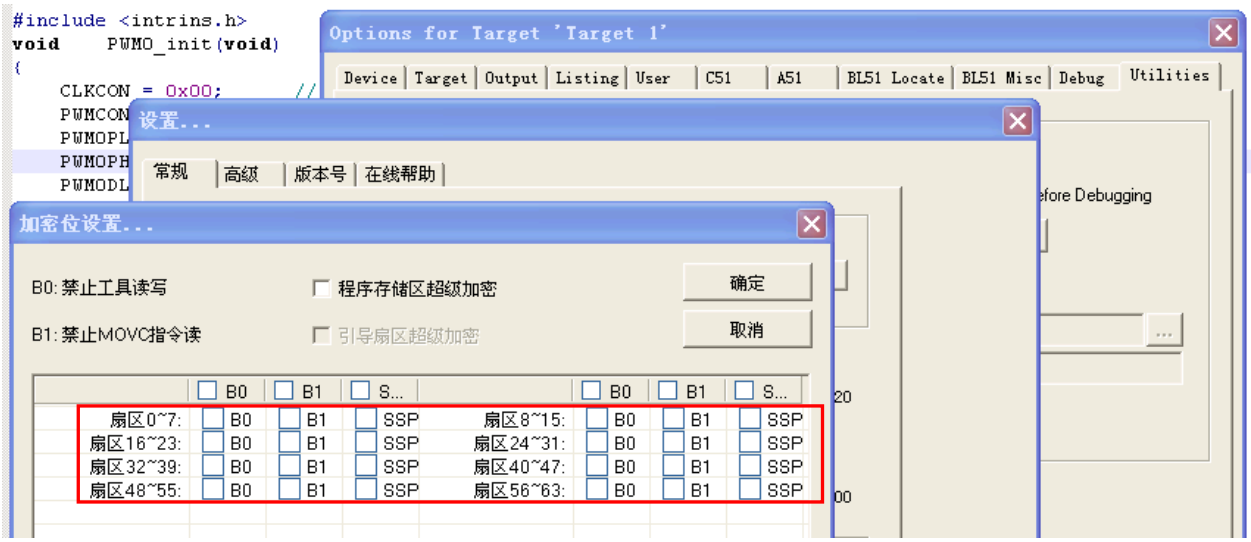
针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

3. SSP 编程设置

3.1. Flash SSP 编程设置

若使用 SSP 功能对 Flash 进行擦除或者编写，首先需要将 FLASHCON 的 FAC 位置 0，然后设置相应的 XPAGE，IB_OFFSET，IB_DATA（需要操作的地址和数据），再设置 IB_CON1，若 IB_CON1 为 0xE6，则相应的操作为扇区擦除，若 IB_CON1 为 0x6E，则相应的操作为存储单元编程。之后再设置 IB_CON2 为 0x05，IB_CON3 为 0x0A，IB_CON4 为 0x06，IB_CON5 为 0x09 完成 Flash 的整个操作。需要注意，SSP 扇区擦除操作不可选择 SSP 程序所在扇区作为操作对象。

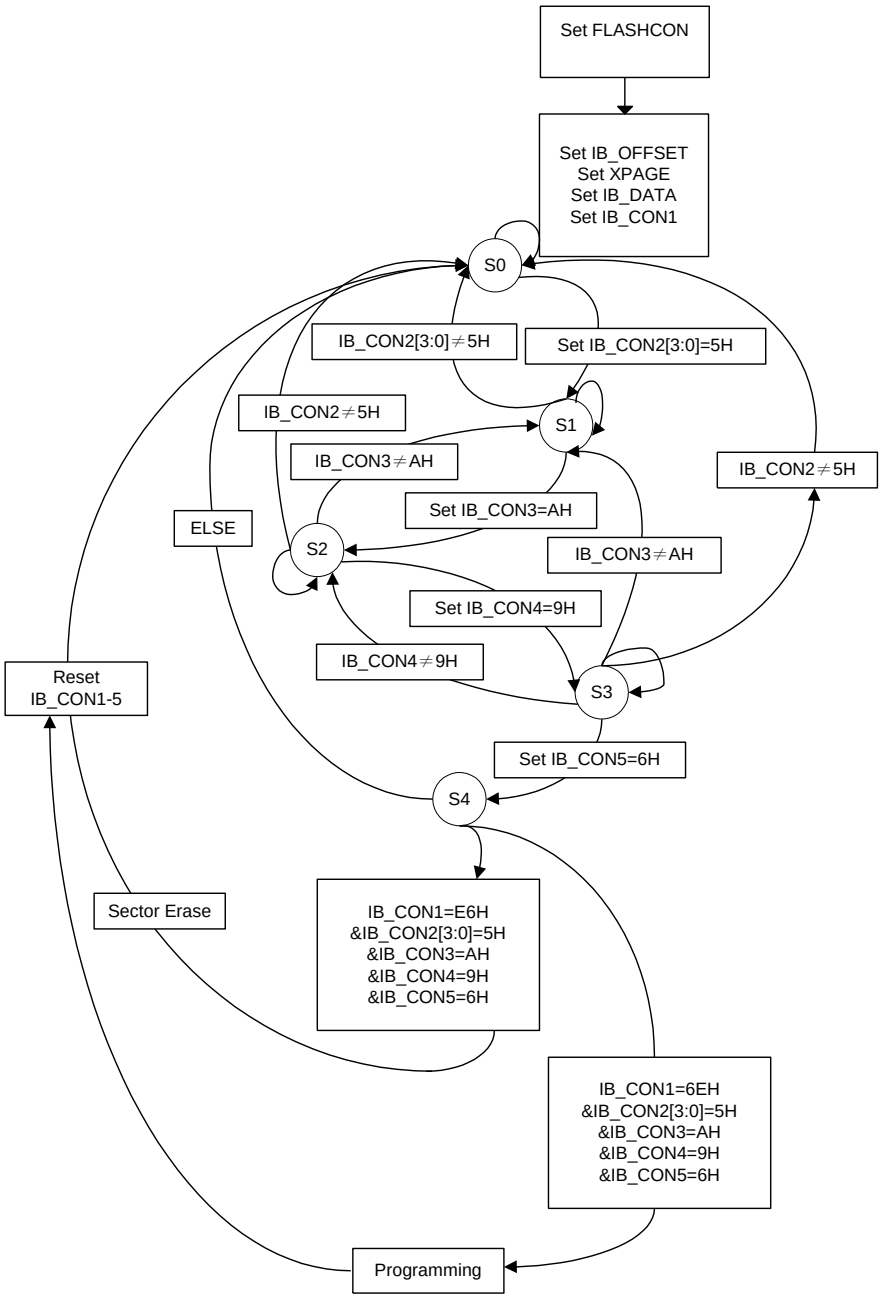
特别地，若需要对特定扇区进行保护，防止 SSP 误操作，可在 Keil 中通过“Options->Utilities->Settings->加密位…”路径设置 Sector 加密。其中，B0 加密对应 Flash 代码保护模式 0；B1 加密对应 Flash 代码保护模式 1。Flash 代码保护模式详细介绍，请参照“SH79F6481A DATASHEET”。



EEPROM SSP 编程设置

若使用 SSP 功能对 EEPROM 进行擦除或者编写，首先需要将 FLASHCON 的 FAC 位置 1，然后设置相应的 XPAGE，IB_OFFSET，IB_DATA（需要操作的地址和数据），再设置 IB_CON1，若 IB_CON1 为 0xE6，则相应的操作为扇区擦除，若 IB_CON1 为 0x6E，则相应的操作为存储单元编程。之后再设置 IB_CON2 为 0x05，IB_CON3 为 0x0A，IB_CON4 为 0x06，IB_CON5 为 0x09 完成 EEPROM 的整个操作。Flash 控制流程图如 3.3 所示。

3.2. Flash & EEPROM 控制流程图



3.3. 应用实例

3.3.1 要求

先将整个 Flash 区域整体擦除，再 sector0 写入 00，sector1 写入 01，依次类推至 sector15 写入 0F。

3.3.2 例程

```
#include <SH79F6481A.h>
#include <intrins.h>
#include <absacc.h>
#define uchar unsigned char
#define uint unsigned int
#define NOP _nop_();
uchar i,j,m;
uint n;
uint code *p;
uchar count = 0;
uchar Ssp_flag;
void main (void)
{
    uchar temp;
    CLKCON = 0;
    m = 0;
    n = 0;
    Ssp_flag = 0xA5;
/********************sector erase****************/
    EA = 0;
    for(i=0x00;i<0x40;i++)
    {
        FLASHCON = FLASHCON&0xFE;
        NOP
        EA = 0;
        IB_CON1 = 0xE6;
        IB_CON2 = 0x05;
        IB_CON3 = 0x0A;
        IB_CON4 = 0x09;
        if(!(Ssp_flag==0xA5)) goto Error;
```

```
            XPAGE = (i<<1)&0xFE;
            IB_CON5 = 0x06;
            NOP
            NOP
            NOP
            NOP
            NOP
        }
/********************sector write****************/
temp = 0x00;
for(m=0x00;m<0x10;m++)
        {
        for(n=0x00;n<512;n++)
            {
                temp = m;
                FLASHCON = FLASHCON&0xFE;
                NOP
                EA = 0;
                IB_OFFSET = n;
                XPAGE |= n>>8;
                XPAGE |= m<<1;
                IB_DATA = temp;
                IB_CON1 = 0x6E;
                IB_CON2 = 0x05;
                IB_CON3 = 0x0A;
                IB_CON4 = 0x09;
                if(!(Ssp_flag==0xA5)) goto Error;
                IB_CON5 = 0x06;
                NOP
                NOP
                NOP
                NOP
                NOP
                XPAGE = 0;
            }
        }
```

```
while(1);
Error:
    Ssp_flag = 0;
    IB_CON1 = 0x00;
    IB_CON2 = 0x00;
    IB_CON3 = 0x00;
    IB_CON4 = 0x00;
    IB_CON5 = 0x00;
    XPAGE = 0x00;
    FLASHCON = 0x00;
    EA = 0;
    while(1);
}
```

4. 可读识别码

SH79F6481A 每颗芯片出厂后都固化有一个 40 位的可读识别码，它的值为 0 - 0xffffffffff 的随机值，它是无法擦除的（存放在地址信息存储区 0x127b - 127f），可以由程序或编程工具读出。

程序读出示例：

```
#include <SH79F6481A.h>
#include <intrins.h>
#include <absacc.h>
unsigned char Temp1,Temp2,Temp3,Temp4,Temp5;
void main()
{
    FLASHCON = 0x01;
    Temp1 = CBYTE[0x127b];
    Temp2 = CBYTE[0x127c];
    Temp3 = CBYTE[0x127d];
    Temp4 = CBYTE[0x127e];
    Temp5 = CBYTE[0x127f];
    FLASHCON = 0x00;
    while(1);
}
```

5. 编程注意事项

为确保顺利完成SSP编程，用户软件必须按以下步骤设置：

(1) 用于代码/数据编程：

1. 关闭中断；
2. 根据地址设置 XPAGE，IB_OFFSET；
3. 按编程需要，设置 IB_DATA；
4. 按照顺序设置 IB_CON1 - 5；
5. 添加 4 个 NOP 指令；
6. 开始编程，CPU 将进入 IDLE 模式；烧写完成后自动退出 IDLE 模式；
7. 如需继续写入数据，跳转至第 2 步；
8. XPAGE 寄存器清 0，恢复中断设置。

(2) 用于扇区擦除：

1. 关闭中断；
2. 按相应的扇区设置 XPAGE；
3. 按照顺序设置 IB_CON1 - 5；

4. 添加 4 个 NOP 指令；
5. 开始擦除，CPU 将进入 IDLE 模式；擦除完成后自动退出 IDLE 模式；
6. 如需要继续擦除数据，跳转至第 2 步；
7. XPAGE 寄存器清 0，恢复中断设置。

(3) 读取：

使用“MOVC A,@A+DPTR”或者“MOVC A,@A+PC”指令。

(4) 对于类 EEPROM 区域

对于类 EEPROM 的操作类似于 Flash 的操作，即类似上述(1)/(2)/(3)部分的描述。区别在于：

1. 在对类 EEPROM 进行擦除、写或读之前，应首先将 FLASHCON 寄存器的最低位 FAC 位置 1。
2. 类 EEPROM 的扇区大小可以通过 option 选择。

注意：

1. 系统时钟不得低于 200kHz 以确保 FLASH 的正常编程

2. 当不需对类 EEPROM 操作时，必须将 FAC 位清 0

6. FLASH/类 EEPROM 的烧写/擦除的超级抗干扰措施

1） 在程序下载时，选择代码选项中的“enable LVR function”。

2） 设置扇区时，对 XPAGE 写立即数。

3） 在调用烧写或擦除函数前，置一个标志，比如 0A5H；在烧写或擦除函数中判断此标志是否为 0A5H，否则清零此标志，退出函数。

下面类 EEPROM 示例仅供参考：

C 文件：

```
uchar Ssp_flag;

/********************SSP Erase****************/
    Ssp_flag = 0xA5;
    EA = 0;
    FLASHCON = 0x01;
    IB_CON1 = 0xE6;
    IB_CON2 = 0x05;
    IB_CON3 = 0x0A;
    IB_CON4 = 0x09;
    if(!(Ssp_flag==0xA5)) goto Error;
    XPAGE = 0x01;            //XPAGE 写立即数
    IB_CON5 = 0x06;
    NOP
    NOP
```

```
    NOP
    NOP

/********************sector write****************/
    Ssp_flag = 0x5A;
    EA = 0;
    FLASHCON = 0x01;
    IB_DATA = data;          //data 为烧写数据
    IB_OFFSET = addr;        //addr 为烧写地址低位
    IB_CON1 = 0x6E;
    IB_CON2 = 0x05;
    IB_CON3 = 0x0A;
    IB_CON4 = 0x09;
    if(!(Ssp_flag==0x5A)) goto Error;
    XPAGE = 0x01;            //烧写地址高位，写立即数
    IB_CON5 = 0x06;
    NOP
    NOP
    NOP
    NOP
    while(1);
Error:
    Ssp_flag = 0;
    IB_CON1 = 0x00;
    IB_CON2 = 0x00;
    IB_CON3 = 0x00;
    IB_CON4 = 0x00;
    IB_CON5 = 0x00;
    XPAGE = 0x00;
    FLASHCON = 0x00;
    EA = 0;
    while(1);
```

三、SH79F6481A Timer3 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。适用于小家电，白色家电，LED 显示等。

SH79F6481A 其 Timer3 特性为：

- 定时器3是16位自动重载定时器，且可以工作在掉电模式

定时器 3 是 16 位自动重载定时器，通过两个数据寄存器 TH3 和 TL3 访问，由 T3CON 寄存器控制。IEN0 寄存器的 ET3 位置 1 允许定时器 3 中断（详见中断章节）。

定时器 3 只有一个工作方式：16 位自动重载计数器/定时器，可以设置预分频比，并可以工作在 CPU 掉电模式。

定时器 3 有一个 16 位计数器/定时器寄存器（TH3，TL3）。当 TH3 和 TL3 被写时，用作定时器重载寄存器，当被读时，被用做计数寄存器。TR3 位置 1 使定时器 3 开始递增计数。定时器在 0xFFFF 到 0x0000 溢出并置 TF3 位为 1。溢出同时，定时器重载寄存器的 16 位数据被重新载入计数寄存器中，TH3 写操作也导致重载寄存器的数据重新载入计数寄存器。

TH3 和 TL3 读写操作遵循以下顺序：

写操作：先低位后高位

读操作：先高位后低位

定时器 3 可以工作在掉电模式。

当 T3CLKS [1:0] 选为 00 时，定时器 3 在掉电模式下不计数。

当 T3CLKS [1:0] 选为 01 时，定时器 3 可以工作在掉电模式。即使所有振荡器关闭，定时器 3 依然可以对 T3 计数。

当 T3CLKS [1:0] 选为 10 时，定时器 3 可以工作在掉电模式。但是如果在掉电模式下低频振荡器关闭则定时器 3 不计数。

详见下表：

T3CLKS [1:0]	振荡器状态	普通模式	掉电模式
00	不限	工作	不工作
01	不限	工作	工作
10	低频打开，且掉电模式低频关闭	工作	不工作
	低频打开，且掉电模式低频不关闭	工作	工作

2. 控制寄存器

Timer3 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	T3CON	控制寄存器
	TL3	低位计数器
	TH3	高位计数器
数据指针选择	INSCON	特殊功能寄存器页选择

***注意*:**

1. 在读或写 TH3 和 TL3 时，如果时钟源不是系统时钟，要确保 TR3=0。

2. 当定时器 3 用 T3 端口作为时钟源时，TR3 由 0 变为 1 之后的 1.5 个系统周期内，T3 的下降沿无效。

3. 定时器 3 相关的寄存器位于 Bank1，对 Timer3 的寄存器进行相关操作时请先将 INSCON 中的 BSK0 位置 1.

4. 进出中断时需对 INSCON 寄存器进行压栈出栈操作。

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”。

3. Timer3 设置

Timer3 只有一种工作方式：16 位自动重载计数器/定时器，且 Timer3 可以在掉电模式下计数，但是在掉电模式下计数对 Timer3 的工作时钟是有要求的，具体参见概述中的表格。

要使用 Timer3 需要要设置好 Timer3 的工作时钟，然后设置工作时钟的预分频比，再写入符合自己要求的 TL3，TH3。然后启动 Timer3 即可。

注：*Timer3 中断标志位由硬件清零*

4. 应用实例

4.1. 要求

Timer3 计数时钟选择 T3 口输入的时钟，T3 口输入的时钟频率为 2kHz，设置 Timer3 每一秒中唤醒 Power-Down 一次。

4.2. 例程

```
#include <SH79F6481A.H>
#include <intrins.H>
voidTimer3_init(void)
{
    CLKCON = 0x00;      // 系统时钟设置为 24MHz
    INSCON |= 0x40;     // 切换至 BANK1
    T3CON = 0x01;       // T3 口作为 Timer3 的时钟输入口，该口上拉自动打开
    TL3 = 0x30;         // 定时 1 s 产生中断
    TH3 = 0XF8;
```

```
    INSCON &= 0xBF;      // 切换至 BANK0
}

void main(void)
{
    Timer3_init();
    EA = 1;
    IEN0 |= 0x20;          // 允许 Timer3 中断
    INSCON |= 0x40;      // 切换至 BANK1
    TR3 = 1;               // Timer3 开始计数
    INSCON &= 0xBF;     // 切换至 BANK0
    while(1)
    {
        SUSLO = 0x55;
        PCON |= 0x02;          // 进入掉电模式
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void timer3_int(void)  interrupt 5
{
    _push_(INSCON);   // 进中断对 INSCON 进行压栈
    INSCON = 0x40;      // 切换至 BANK1
    TF3 = 0;
    INSCON = 0x00;    // 切换至 BANK0
    _pop_(INSCON);    // 出中断 INSCON 出栈
}
```

四、SH79F6481A Timer4 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 Timer4 特性为：

- 定时器4是16位自动重载定时器，且可以触发ADC转换

定时器 4 是 16 位自动重载定时器。两个数据寄存器 TH4 和 TL4 可作为一个 16 位寄存器来访问。由 T4CON 寄存器控制。IEN1 寄存器的 ET4 位置 1 允许定时器 4 中断（详见 DATASHEET 中断章节）。

当 TH4 和 TL4 被写时，用作定时器重载寄存器，当被读时，被用做计数寄存器。TR4 位置 1 使定时器 4 开始递增计数。定时器在 0xFFFF 到 0x0000 溢出并置 TF4 位为 1。溢出同时，定时器重载寄存器的 16 位数据重新载入计数寄存器中，对 TH4 的写操作也导致重载寄存器的数据重新载入计数寄存器。

TH4 和 TL4 读写操作遵循以下顺序：

写操作：先低位后高位

读操作：先高位后低位

2. 控制寄存器

Timer4 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	T4CON	控制寄存器
	TL4	低位计数器
	TH4	高位计数器

注：Timer4 相关的寄存器位于Bank1，对Timer4 相关的寄存器进行相关操作的时候请先将INSCON 寄存器中的BSK0 位置 1。进出中断时需对 INSCON 寄存器进行压栈出栈操作。

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

3. Timer4 设置

定时器 4 有两种工作方式：16 位自动重载定时器和有 T4 边沿触发的 16 位自动重载定时器。这些方式通过 T4CON 寄存器的 T4M[1:0]设置。Timer4 的溢出可以触发 ADC 转换。

3.1 方式 0 设置

定时器 4 在方式 0 为 16 位自动重载定时器。若使用其功能，需要将 **T4M[1:0]**设置为 00，再设置 **T4CLKS**，若为 0 则选择系统时钟，若为 1 则选择从 T4 口输入的外部输入时钟。然后设置 **T4PS[1:0]** 选择好分频比，选择好 TL4，TH4 的初值，最后使 TR4 置 1，这样 Timer4 方式 0 就启动了，当定时器从 0xFFFF 到 0x0000 溢出并置 TF4 位为 1，可供查询或者产生中断。

3.2 方式 2 设置

定时器 4 在方式 2 为 16 位自动重载定时器。T4CON.0 寄存器的 T4CLKS 位一直为 0，定时器 4 只能选择系统时钟为时钟源，其余设置与方式 0 一致。

方式 2 和方式 0 不同之处在于方式 2 的自动重载是带边沿触发的，这方式 2 下，T4 端口不能作为外部时钟输入，其功能为触发 Timer4 自动重载。若使用其功能，需要将 **T4M[1:0]**设置为 10（下降沿触发）或者 11（上升沿触发），再设置 **T4PS[1:0]**选择好分频比，然后设置 **TC4**，为 0 则 Timer4 不能再被触发，为 1 则 Timer4 可以再被触发，选择好 TL4，TH4 的初值，最后使 TR4 置 1，这样 TC4 的下降沿可以使 Timer4 启动，当定时器从 0xFFFF 到 0x0000 溢出并置 TF4 位为 1，同时会停止 Timer4 定时，等待下一个 T4 的下降沿或者下降沿可以启动 Timer4（TC4 = 1）。

***注**：当定时器 4 用作计数器时，T4 引脚的输入信号频率要小于系统时钟的一半。*

4. 应用实例

4.1. 方式 0 要求

Timer4 时钟为系统时钟，循环定时产生一个 100ms 的定时中断

4.2. 方式 0 例程

```
#include <SH79F6481A.H>
#include <intrins.H>

void Port_init(void)
{
    P1 = 0x00;
    P1CR = 0x01;          // P1.0 作为输出标志
}
void Timer4_init(void)
{
    CLKCON = 0x20;      // 系统时钟设置为 12MHz
    INSCON = 0x40;
    T4CON &= 0xF3;      // 工作方式选择 Mode0
    T4CON &= 0xFE;      // Timer4 时钟选择系统时钟,.T4 口作为普通 IO 口
    T4CON &= 0xEF;
    T4CON |= 0x20;      // 预分频比设置为 1/64
    TL4 = 0x3E;         // 定时 100ms 产生一次中断
    TH4 = 0X49;
    INSCON = 0x00;
}
```

```
void main(void)
{
    Timer4_init();
    Port_init();
    EA = 1;
    IEN1 |= 0x40;   // 使能 timer4 中断
    INSCON = 0x40;
    TR4 = 1;        // timer4 开始计数
    INSCON = 0x00;
    while(1);
}

void timer4_int(void)   interrupt 9
{
    _push_(INSCON);       // 进中断对 INSCON 进行压栈
    INSCON = 0x40;
    TF4 = 0;
    INSCON = 0x00;
    P1_0 = ~P1_0;         // P1.0 的翻转周期为 200ms，即频率为 5Hz
    _pop_(INSCON);        // 出中断 INSCON 出栈
}
```

4.3. 方式 2 要求

在 T4 的每个下降沿之后的 100ms 处产生一次 P1.0 翻转

4.4. 方式 2 例程

```
#include <SH79F6481A.H>
#include <intrins.H>

void Port_init(void)
{
    P1 = 0x00;
    P1CR = 0x01;        // P1.0 作为输出标志
}
```

```
void Timer4_init(void)
{
    CLKCON = 0x00;      //系统时钟  12MHz
    INSCON = 0x40;
    T4CON |= 0x0C;    //  选择 Mode2，上升沿触发
    T4CON &= 0xEF;
    T4CON |= 0x20;    //  预分频比选择 1/64
    T4CON |= 0x40;    // TC4 = 1，可以再次被触发
    TL4 = 0x3E;         //  定时 100ms  产生中断
    TH4 = 0X49;
    INSCON = 0x00;
}

void main(void)
{
    Timer4_init();
    Port_init();
    EA = 1;
    IEN1 |= 0x40;      //  使能  timer4  中断
    INSCON = 0x40;
    TR4 = 1;            // timer4  开始计数
    INSCON = 0x00;
    while(1);
}

void timer4_int(void)   interrupt 9
{
    _push_(INSCON);        //  进中断对 INSCON 进行压栈
    INSCON = 0x40;
    TF4 = 0;
    INSCON = 0x00;
    P1_0 = ~P1_0;           // P1.0 的翻转周期为 200ms，即频率为 5Hz
    _pop_(INSCON);         //  出中断 INSCON 出栈
}
```

五、SH79F6481A Timer5 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 Timer5 特性为：

- 定时器5是16位自动重载定时器

定时器 5 是 16 位自动重载定时器。两个数据寄存器 TH5 和 TL5 可作为一个 16 位寄存器来访问。由 T5CON 寄存器控制。IEN0 寄存器的 ET5 位置 1 允许定时器 5 中断（详见 DATASHEET 中断章节）。

当 TH5 和 TL5 被写时，用作定时器重载寄存器，当被读时，被用做计数寄存器。TR5 位置 1 使定时器 5 开始递增计数。定时器在 0xFFFF 到 0x0000 溢出并置 TF5 位为 1。溢出同时，定时器重载寄存器的 16 位数据重新载入计数寄存器中，对 TH5 的写操作也导致重载寄存器的数据重新载入计数寄存器。

TH5 和 TL5 读写操作遵循以下顺序：

写操作：先低位后高位

读操作：先高位后低位

2. 控制寄存器

Timer5 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	T5CON	控制寄存器
	TL5	低位计数器
	TH5	高位计数器

***注**: Timer5 相关的寄存器位于Bank1，对Timer5 相关的寄存器进行相关操作的时候请先将INSCON 寄存器中的 BSK0 位置 1. 进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

3. Timer5 设置

定时器 5 有一种工作方式：16 位自动重载定时器。

Timer5 只能使用系统时钟作为模块的时钟，使用其时，先根据需要的定时时基结合系统时钟设置好预分频比选择位 **T5PS[1:0]**，然后 TR5 置 1，使能 Timer5，定时器在 0xFFFF 到 0x0000 溢出并置 TF5 位为 1，可供查询和中断。溢出同时，定时器重载寄存器的 16 位数据重新载入计数寄存器中。

4. 应用实例

4.1. 要求

Timer5 循环定时产生一个 5ms 的定时中断

4.2. 例程

```
#include <SH79F6481A.H>
#include <intrins.H>

void Port_init(void)
{
    P1 = 0x00;
    P1CR = 0x01;          // P1.0 作为输出标志
}
void Timer5_init(void)
{
    CLKCON = 0x20;      // 系统时钟 12MHz
    INSCON = 0x40;
    T5CON = 0x00;       // Timer5 时钟选择系统时钟 = 12MHz
    TL5 = 0XA0;         // 定时 5ms 产生一次中断
    TH5 = 0X15;
    INSCON = 0x00;
}

void main(void)
{
    Timer5_init();
    Port_init();
    EA = 1;
    ET5 |= 1;             // 使能 timer5 中断
    INSCON = 0x40;
    T5CON |= 0x02;        // timer5 开始计数
    INSCON = 0x00;
    while(1);
}

void timer5_int(void)   interrupt 3
{
    _push_(INSCON);       // 进中断对 INSCON 进行压栈
```

```
    INSCON = 0x40;
    T5CON &= 0x7F;
    INSCON = 0x00;
    _pop_(INSCON);          //  出中断对 INSCON 出栈
    P1_0 = ~P1_0;           // P1.0 的翻转周期为 10ms，即频率为 100Hz
}
```

六、SH79F6481A SPI 应用指南

1. 概述

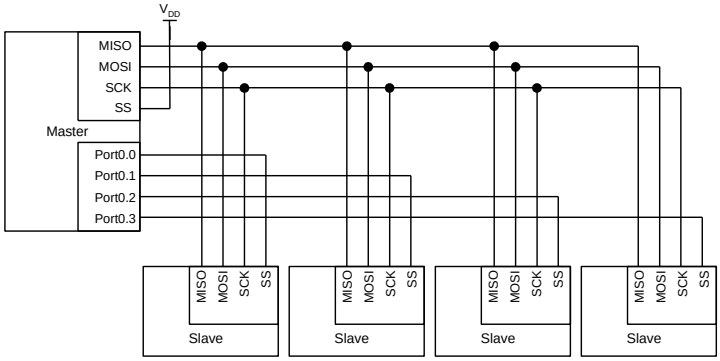
SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 SPI 模块特性为：

- 全双工，三线同步传输
- 主从机操作
- 7个可编程主时钟频率
- 极性相位可编程的串行时钟
- 带MCU中断的主模式故障出错标志
- 写入冲突标志保护
- 可选择LSB或MSB传输

串行外部设备接口（SPI）是一种高速串行通信接口，允许 MCU 与外围设备（包括其它 MCU）进行全双工，同步串行通讯。

下图所示即为典型的由一个主设备和若干从属外部设备组成的 SPI 总线网络，主设备通过 3 条线连接所有从设备，主设备控制连接从属设备$\overline{SS}$引脚的 4 个并行端口来选中其中一个从属设备进行通讯。



2. 控制寄存器

SPI 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	SPCON	控制寄存器
	SPSTA	状态寄存器
	SPDAT	数据寄存器

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：*进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. SPI 设置

3.1. 波特率设置

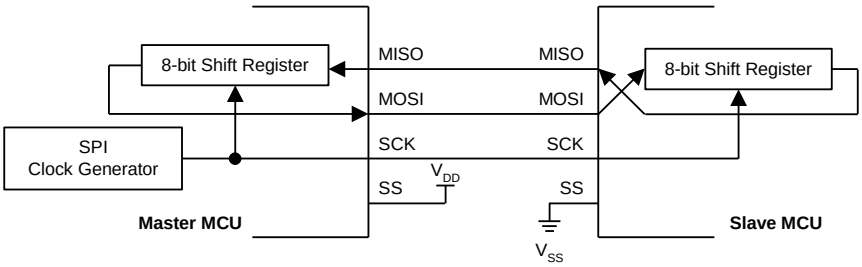
在主模式下，SPI 的波特率有 8 种可选择的频率，分别是内部时钟的 4，8，16，32，64，128，256 或 512 分频，可以通过设定 SPCON 寄存器的 SPR[2:0]位进行选择。

3.2. 工作模式设置

SPI 可配置为主模式或从属模式中的一种。SPI 模块的配置和初始化通过设置 SPCON 寄存器（串行外围设备控制寄存器）和 SPSTA（串行外围设备状态寄存器）来完成。配置完成后，通过设置 SPCON，SPSTA，SPDAT（串行外围设备数据寄存器）来完成数据传送。

在 SPI 通讯期间，数据同步地被串行的移进移出。串行时钟线（SCK）使两条串行数据线（MOSI 和 MISO）上数据的移动和采样保持同步。从设备选择线（$\overline{SS}$）可以独立地选择 SPI 从属设备；如果从设备没有被选中，则不能参与 SPI 总线上的活动。

当 SPI 主设备通过 MOSI 线传送数据到从设备时，从设备通过 MISO 线发送数据到主设备作为响应，这就实现了在同一时钟下数据发送和接收的同步全双工传输。发送移位寄存器和接收移位寄存器使用相同的特殊功能器地址，对 SPI 数据寄存器 SPDAT 进行写操作将写入发送移位寄存器，对 SPDAT 寄存器进行读操作将获得接收移位寄存器的数据。



全双工主从互联图

主模式

(1) 模式启动

SPI 主设备控制 SPI 总线上所有数据传送的启动。当 SPCON 寄存器中的 MSTR 位置 1 时，SPI 在主模式下运行，只有一个主设备可以启动传送。

(2) 发送

在 SPI 主模式下，写一个字节数据到 SPI 数据寄存器 SPDAT，数据将会写入发送移位缓冲器。如果发送移位寄存器已经存在一个数据，那么主 SPI 产生一个 WCOL 信号以表明写入太快。但是在发送移位寄存器中的数据不会受到影响，发送也不会中断。另外如果发送移位寄存器为空，那么主设备立即按照 SCK 上的 SPI 时钟频率串行地移出发送移位寄存器中的数据到 MOSI 线上。当传送完毕，SPSTA 寄存器中的 SPIF 位被置 1。如果 SPI 中断被允许，当 SPIF 位置 1 时，也会产生一个中断。

(3) 接收

当主设备通过 MOSI 线传送数据给从设备时，相对应的从设备同时也通过 MISO 线将其发送移位寄存器的内容传送给主设备的接收移位寄存器，实现全双工操作。因此，SPIF 标志位置 1 即表示传

送完成也表示接收数据完毕。从设备接收的数据按照 MSB 或 LSB 优先的传送方向存入主设备的接收移位寄存器。当一个字节的数据完全被移入接收寄存器时，处理器可以通过读 SPDAT 寄存器获得该数据。如果发生超限（SPIF 标志未被清 0，就试图开始下一次传送），RXOV 位置 1，表示发生数据超限，此时接收移位寄存器保持原有数据并且 SPIF 位置 1，这样直到 SPIF 位被清 0，SPI 主设备将不会接收任何数据。

从模式

(1) 模式启动

当 SPCON 寄存器中的 MSTR 位清 0，SPI 在从模式下运行。在数据传送之前，从设备的$\overline{SS}$引脚必须被置低，而且必须保持低电平直到一个字节数据传送完毕。

(2) 发送与接收

从属模式下，按照主设备控制的 SCK 信号，数据通过 MOSI 引脚移入，MISO 引脚移出。一个位计数器记录 SCK 的边沿数，当接收移位寄存器移入 8 位数据（一个字节）同时发送移位寄存器移出 8 位数据（一个字节），SPIF 标志位被置 1。数据可以通过读取 SPDAT 寄存器获得。如果 SPI 中断被允许，当 SPIF 置 1 时，也会产生一个中断。

为防止超限，SPI 从设备在向接收移位寄存器移入数据之前也必须软件清零 SPIF 标志位，否则 RXOV 位置 1，表示发生数据超限。此时接收移位寄存器保持原有数据并且 SPIF 位置 1，这样 SPI 从设备将不会接收任何数据直到 SPIF 清 0。

SPI 从设备不能启动数据传送，所以 SPI 从设备必须在主设备开始一次新的数据传送之前将要传送的数据写入发送移位寄存器。如果从设备在第一次开始发送之前未写入数据，从设备将传送“0x00”字节给主设备。如果写 SPDAT 操作发生在传送过程中，那么 SPI 从设备的 WCOL 标志位置 1，即如果传送移位寄存器已经含有数据，SPI 从设备的 WCOL 位置 1，表示写 SPDAT 冲突。但是移位寄存器的数据不受影响，传送也不会被中断。

3.3. 出错检测

SPSTA 寄存器中的标志位表示在 SPI 通讯中的出错情况：

(1) 模式故障（MODF）

SPI 主模式下的模式故障出错表明$\overline{SS}$引脚上的电平状态与实际的设备模式不一致。SPSTA 寄存器中 MODF 位置 1 后，表明系统控制存在多主设备冲突的问题。这种情况下，SPI 系统受到如下影响：

- 产生 SPI 接收/错误 CPU 中断请求；
- SPSTA 寄存器的 SPEN 位清 0，SPI 被禁止；
- SPCON 寄存器的 MSTR 位清 0。

当 SPCON 寄存器的$\overline{SS}$引脚禁止位（SSDIS）清 0，$\overline{SS}$引脚信号为低时，MODF 标志位置 1。然而，对于只有一个主设备的系统来说，主设备的$\overline{SS}$引脚被拉低，那决不是另外一个主设备试图驱动网络。这种情况下，为防止 MODF 置 1，可使 SPCON 寄存器中的 SSDIS 位置 1，$\overline{SS}$引脚作为普通 I/O 口或是其它功能引脚。

重新启动串行通信时，用户必须将 MODF 位软件清 0，将 SPCON 寄存器中的 MSTR 位和 SPSTA 寄存器的 SPEN 位置 1，重新启动主模式。

(2) 写冲突（WCOL）

在发送数据序列期间写入 SPDAT 寄存器会引起的写冲突，SPSTA 寄存器中的 WCOL 位置 1。WCOL 位置 1 不会引起中断，发送也不会中止。WCOL 位需由软件清 0。

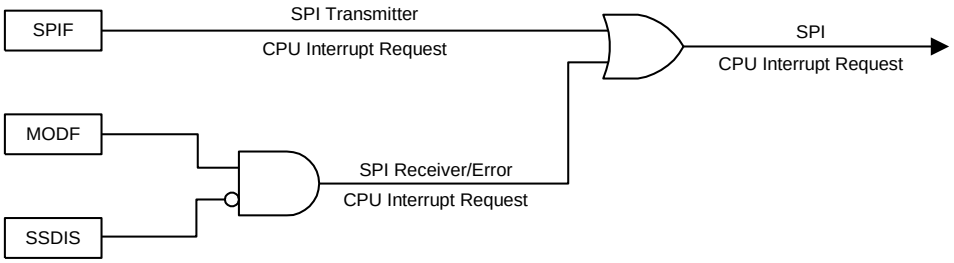
(3) 超限情况（RXOV）

主设备或从设备尚未清除 SPIF 位，主或从设备又试图发送几个数据字节时，超限情况发生。在这种情况下，接收移位寄存器保持原有数据，SPIF 置 1，同样 SPI 设备直到 SPIF 被清除后才会再接收数据。在 SPIF 位被清除之前继续调用中断，发送也不会中止。RXOV 位置 1 不会引起中断，RXOV 位需由软件清 0。

两种 SPI 状态标志 SPIF & MODF 能产生一个 CPU 中断请求。

串行外围设备数据发送标志，SPIF：完成一个字节发送后由硬件置 1。

模式故障标志，MODF：该位被置 1 表示$\overline{SS}$引脚上的电平与 SPI 模式不一致的。SSDIS 位为 0 并且 MODF 置 1 将产生 SPI 接收器/出错 CPU 中断请求。当 SSDIS 置 1 时，无 MODF 中断请求产生。



SPI 中断请求的产生

4. 应用实例

4.1. 要求

主模式，SCK = SYS/64，无效 SS 脚，不停循环发送 0x00~0xFF。

4.2. 例程

```
#include <SH79F6481A.H>
#include <intrins.h>

unsigned char out_flag = 0;
unsigned char r_data = 0;

void spi_init(void)
{
    CLKCON = 0x00;      // 系统时钟  = 24Mz
    SPCON |= 0x40;      // 主模式
```

```
    SPCON |= 0x08;       // 关闭 SS 引脚
    SPCON |= 0x05;       // 串行外部设备时钟速率 = 系统时钟频率 / 64
    SPSTA = 0x00;
}

void main(void)
{
    spi_init();
    EA = 1;
    IEN1 |= 0x01;        // 使能 spi 中断
    SPSTA |= 0x80;       // 使能 spi 模块
    SPDAT = 0x00;
    while(1)
    {
        if(out_flag == 1)
        {
            SPDAT ++;
            out_flag = 0;
        }
    }
}

void spi_interrupt(void) interrupt  19
{
    _push_(INSCON);          // 进中断对 INSCON 进行压栈
    out_flag = 1;
    SPSTA &= 0xBF;
    _pop_(INSCON);           // 出中断对 INSCON 出栈
}
```

七、SH79F6481A EUART 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 EUART 模块特性为：

- SH79F6481A带有3个EUART，兼容传统8051
- EUART0/EUART1/EUART2自带波特率发生器，波特率可选择系统时钟分频或自带波特率发生器溢出率的1/16
- 增强功能包括帧出错检测及自动地址识别
- 支持TTL电平转换
- EUART有四种工作方式

2. 控制寄存器

EUART 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	SCON	EUART 控制及状态寄存器
	SBUF	EUART 数据寄存器
	PCON	电源控制寄存器
	SADDR	EUART 从属地址寄存器
	SADEN	EUART 地址掩码寄存器
	SFINE	波特率微调寄存器
	SBRT	波特率发生器寄存器
	UTOS	TTL 电平使能寄存器

注意：1.EUART1/EUART2 相关寄存器在 BANK1 中，对其进行操作前，请先将 INSCON 中的 BSK0 位置 1；进出中断时需对 INSCON 寄存器进行压栈出栈操作。

2.建议设置 Tx 对应 IO 为输出高。

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

3. EUART 设置

EUART 有 4 种工作方式。在通信之前用户必须先初始化 SCON，选择方式和波特率。

在所有四种方式中，任何将 SBUF 作为目标寄存器的写操作都会启动发送。在方式 0 中由条件 RI = 0 和 REN = 1 初始化接收。这会在 TxD 引脚上产生一个时钟信号，然后在 RxD 引脚上移 8 位数据。在其它方式中由输入的起始位初始化接收（如果 REN = 1）。通过发送起始位，外部发送器开始通信。

EUART 方式列表

SM0	SM1	方式	类型	波特率	帧长度	起始位	停止位	第 9 位
0	0	0	同步	f_{SYS}/（4 或 12）	8 位	无	无	无
0	1	1	异步	自带波特率发生器的溢出率/16	10 位	1	1	无
1	0	2	异步	f_{SYS}/（32 或 64）	11 位	1	1	0，1
1	1	3	异步	自带波特率发生器的溢出率/16	11 位	1	1	0，1

3.1. 方式 0 设置

使用方式 0，需要先通过置 SM2 位（SCON.5）为 0 或 1，波特率固定为系统时钟的 1/12 或 1/4。当 SM2 位为 0 时，串行端口以系统时钟的 1/12 运行。当置 1 时，串行端口以系统时钟的 1/4 运行。

然后任何将 SBUF 作为目标寄存器的写操作都会启动发送。下一个系统时钟 Tx 控制块开始发送。数据转换发生在移位时钟的下降沿，移位寄存器的内容逐次从左往右移位，空位置 0。当移位寄存器中的所有 8 位都发送后，Tx 控制模块停止发送操作，然后在下一个系统时钟的上升沿将 TI 置 1（SCON.1）。

若接收到数据，则 REN（SCON.4）置 1 和 RI（SCON.0）清 0 初始化接收。下一个系统时钟启动接收，在移位时钟的上升沿锁存数据，接收转换寄存器的内容逐次向左移位。当所有 8 位都接收到接收移位寄存器中后，Rx 控制块停止接收，然后在下一个系统时钟的上升沿上 RI 置 1，直到被软件清 0 才允许接收。

3.2. 方式 1 设置

方式 1 提供 10 位全双工异步通信，10 位由一个起始位（逻辑 0），8 个数据位（低位为第一位）和一个停止位（逻辑 1）组成。在接收时，这 8 个数据位存储在 SBUF 中而停止位储存在 RB8（SCON.2）中。方式 1 中的波特率固定为自带波特率发生器溢出率的 1/16。

任何将 SBUF 作为目标寄存器的写操作都会启动发送，实际上发送是从 16 分频计数器中的下一次跳变之后的系统时钟开始的，因此位时间与 16 分频计数器是同步的，与对 SBUF 的写操作不同步。起始位首先在 TXD 引脚上移出，然后是 8 位数据位。在发送移位寄存器中的所有 8 位数据都发送完后，停止位在 TXD 引脚上移出，在停止位发出的同时 TI 标志置位。

只有 REN 位置 1 时才允许接收。当 RxD 引脚检测到下降沿时串行口开始接收串行数据。为此，CPU 对 RxD 不断采样，采样速率为波特率的 16 倍。当检测下降沿时，16 分频计数器立即复位，这有助于 16 分频计数器与 RxD 引脚上的串行数据位同步。16 分频计数器把每一位的时间分为 16 个状态，在第 7、8、9 状态时，位检测器对 RXD 端的电平进行采样。为抑制噪声，在这 3 个状态采样中至少有 2 次采样值一致数据才被接收。如果所接收的第一位不是 0，说明这位不是一帧数据的起始位，该位被忽略，接收电路被复位，等待 RxD 引脚上另一个下降沿的到来。若起始位有效，则移入移位寄存器，并接着移入其它位到移位寄存器。8 个数据位和 1 个停止位移入之后，移位寄存器的内容被分别装入 SBUF 和 RB8 中，RI 置 1，但必须满足下列条件：

1. RI = 0
2. SM2 = 0 或者接收的停止位 = 1

如果这些条件被满足，那么停止位装入 RB8，8 个数据位装入 SBUF，RI 被置 1。否则接收的帧会丢失。这时，接收器将重新去探测 RXD 端是否另一个下降沿。用户必须用软件清除 RI，然后才能再次接收。

3.3. 方式 2 设置

方式 2 是使用异步全双工通信中的 11 位。一帧由一个起始位（逻辑 0），8 个数据位（低位为第一位），一个可编程的第 9 数据位和一个停止位（逻辑 1）组成。方式 2 支持多机通信和硬件地址识别（详见多机通信章节）。在数据传送时，第 9 数据位（SCON 中的 TB8）可以写 0 或 1，例如，可写入 PSW 中的奇偶位 P，或用作多机通信中的数据/地址标志位。当接收到数据时，第 9 数据位进入 RB8 而停止位不保存。PCON 中的 SMOD 位选择波特率为系统工作频率的 1/32 或 1/64。

任何将 SBUF 作为目标寄存器的写操作都会启动发送，同时也将 TB8 载入到发送移位寄存器的第 9 位中。实际上发送是从 16 分频计数器中的下一次跳变之后的系统时钟开始的，因此位时间与 16 分频计数器是同步的，与对 SBUF 的写操作不同步。起始位首先在 TXD 引脚上移出，然后是 9 位数据。在发送转换寄存器中的所有 9 位数据都发送完后，停止位在 TXD 引脚上移出，在停止位开始发送时 TI 标志置位。

只有 REN 位置 1 时才允许接收。当 RxD 引脚检测到下降沿时串行口开始接收串行数据。为此，

CPU 对 RxD 不断采样，采样速率为波特率的 16 倍。当检测下降沿时，16 分频计数器立即复位。这有助于 16 分频计数器与 RxD 引脚上的串行数据位同步。16 分频计数器把每一位的时间分为 16 个状态，在第 7、8、9 状态时，位检测器对 RXD 端的电平进行采样。为抑制噪声，在这 3 个状态采样中至少有 2 次采样值一致数据才被接收。如果所接收的第一位不是 0，说明这位不是一帧数据的起始位，该位被忽略，接收电路被复位，等待 RxD 引脚上另一个下降沿的到来。若起始位有效，则移入移位寄存器，并接着移入其它位到移位寄存器。9 个数据位和 1 个停止位移入之后，移位寄存器的内容被分别装入 SBUF 和 RB8 中，RI 置 1，但必须满足下列条件：

1. RI = 0
2. SM2 = 0 或者接收的第 9 位 = 1，且接收的字节符合实际从机地址

如果这些条件被满足，那么第 9 位移入 RB8，8 位数据移入 SBUF，RI 被置 1。否则接收的数据帧会丢失。

在停止位的当中，接收器回到寻找 RXD 引脚上的另一个下降沿。用户必须用软件清除 RI，然后才能再次接收。

3.4. 方式 3 设置

方式 3 使用方式 2 的传输协议以及方式 1 的波特率产生方式。

3.5. EUART 波特率设置

在**方式 0** 中，波特率可编程为系统时钟的 1/12 或 1/4，由 SM2 位决定。当 SM2 为 0 时，串行端口在系统时钟的 1/12 下运行。当 SM2 为 1 时，串行端口在系统时钟的 1/4 下运行。

在**方式 1** 和**方式 3** 中，使用自带波特率发生器，波特率可微调，精度为一个系统时钟，公式如下：

$$BaudRate = \frac{Fsys}{16 \times (32768 - SBRT) + SFINE}$$

例如：Fsys = 8MHz，需要得到 115200Hz 的波特率，SBRT 和 SFINE 值计算方法如下：

8000000/16/115200 = 4.34

SBRT = 32768 - 4 = 32764

115200 = 8000000/(16 X 4 + SFINE)

SFINE = 5.4 ≈ 5

此微调方式计算出的实际波特率为 115942，误差为 0.64%；

在**方式 2** 中，波特率固定为系统时钟的 1/32 或 1/64，由 SMOD 位（PCON.7）决定。当 SMOD 位为 0 时，EUART 以系统时钟的 1/64 运行。当 SMOD 位为 1 时，EUART 以系统时钟的 1/32 运行。

$$BaudRate = 2^{SMOD} \times (\frac{f_{SYS}}{64})$$

3.6. EUART TTL 电平设置

针对 5V 供电源应用中，UART 与 3V 设备通讯电平不匹配问题，SH79F6481A 的 EUART 增加了 RXD 脚的 TTL 电平转换功能，由 UTOS 寄存器控制。

当 UTOS 的 bit 为 0 时，RXD 引脚的输入高电平阈值为 0.8VDD，输入低电平阈值为 0.2VDD。此时，若 SH79F6481A 的 VDD=5V，则 RXD 引脚的输入高电平阈值为 4V，输入低电平阈值为 1V，无法与 3V 设备的 UART 正常通讯；UTOS 的 bit1=1 时，若 VDD = 4.5V~5.5V，RXD 引脚的输入高电平阈值为 2.0V，输入低电平阈值为 0.8V。若 VDD = 2.7V～4.5V，则 RXD 引脚的输入高电平阈值为 0.25VDD + 0.8，输入低电平阈值为 0.15VDD，能够正确接收 3V 设备所发送的数据。

4. 应用实例

4.1. 要求

使用 EUART 进行串口通讯，通讯的波特率为 9600，1 位起始位，8 位数据位，1 位停止位，通讯内容为：收到一个数据后发送收到的数据出去。

4.2. 例程

```
#include <SH79F6481A.H>
unsigned char       r_data;
unsigned char       out_flag;
voiduart0_init(void)
{
    CLKCON = 0x40;              // 系统时钟选择 6MHz
```

```
        SBRTH=0Xff;
        SBRTL=0xd9;
        SFINE=0x01;
        SCON = 0x50;                    //  工作方式选择 Mode0
}
void main(void)
{
        uart0_init();
        /* TX 对应 IO 设置为输出高*/
        P0CR |= 0x02;
        P0 |= 0x02;         //TXD P0.1 OUTPUT HIGH
        r_data = 0x00;
        out_flag = 0;
        TI = 0;
        RI = 0;
        IEN0 = 0x90;
        REN = 1;
        TR2 = 1;
        while(1)
        {
                if(out_flag == 1)
                {
                        SBUF = r_data;                  //  发送接收到的数据
                        out_flag = 0;
                }
        }
}
void uart0_int(void) interrupt   4
{
        _push_(INSCON);         //  进中断对 INSCON 进行压栈
        if(TI == 1)
        {
                TI = 0;
        }
```

```
    else if(RI == 1)
    {
        r_data = SBUF;
        RI = 0;
        out_flag = 1;
    }
    _pop_(INSCON);          //  出中断对 INSCON 出栈
}
```

5. UART1/2

UART1/2 的用法同 UART0，需要注意的是 UART1/2 的相关寄存器在 BANK1。

八、SH79F6481A ADC 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 ADC 模块特性为：

- 12 位分辨率
- 最高转换速率可达 1Msps
- 参考电压 VDD
- 9 路外接模拟输入，1 路内部模拟通道
- 4 路触发源可选自动触发 AD 转换
- 支持序列转换，而且每个信道都可以配置为多路模拟输入中的任意一路
- 序列转换时相邻通道转换之间的时间间隔可以软件设定
- 其中7路（AN0-AN2/AN4-7）ADC转换速率为100KSPS
- 其中2路（AN3/AN8）ADC转换速率最高可达1MSPS

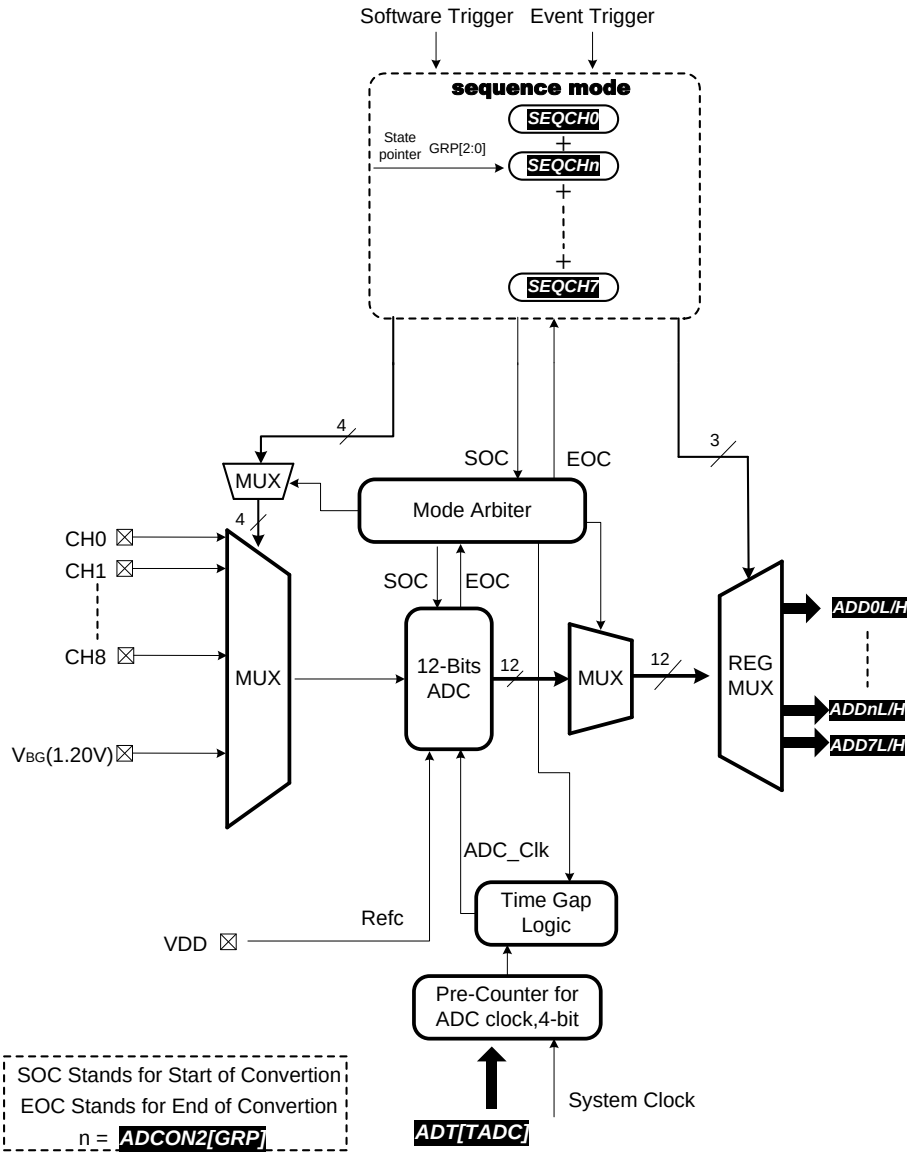
SH79F6481A 包括一个单端型、12 位逐次逼近型模/数转换器（ADC，Analog-to-Digit Converter），模块如图所示。ADC 负责模拟信号到相应 12 位的数字信号的转换，当输入为 GND 时，输出为 0；当输入大于等于 VDD – 1LSB 时，输出最大值。ADC 的基准电压为 VDD。

该模块中有 9 路模拟输入（CH0 – CH8， V_{BG}），通过配置信道寄存器，都可以编入序列中自动进行转换。结果储存在对应的结果寄存器 ADDxH，ADDxL（x = 0 - 8）中，每转换一次序列，结果寄存器的值更新一次。结果寄存器和模拟输入之间的映射关系可以随意设置来组成一个转换序列，而且可以将某一模拟输入通道在序列中重复设置，从而在结果寄存器中获得此模拟输入通道连续多次转换的结果。

对于其中两个信道，转换速率最高可达 1MSPS，可由寄存器设置 ADC 时钟速率以及采样时间。序列中相邻通道之间的时间间隔亦可通过寄存器设置（TGAP[2:0]）。

ADC 模块能在 Idle 模式下工作，并且 ADC 中断能够唤醒 Idle 模式。在 Power-Down 模式下，ADC 模块不工作。

ADC 模块图如下：


Software Trigger
Event Trigger
sequence mode
SEQCH0
State pointer
GRP[2:0]
SEQCHn
SEQCH7
4
3
SOC
EOC
MUX
Mode Arbiter
4
CH0
CH1
CH8
VBG(1.20V)
MUX
SOC
EOC
12-Bits ADC
12
MUX
12
REG MUX
ADD0L/H
ADDnL/H
ADD7L/H
ADC_Clk
Time Gap Logic
Refc
VDD
Pre-Counter for ADC clock,4-bit
System Clock
ADT[TADC]
SOC Stands for Start of Convertion
EOC Stands for End of Convertion
n = ADCON2[GRP]

2. 控制寄存器

ADC 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	ADT	ADC 时钟设置寄存器
	ADCON1	ADC 控制寄存器 1
	ADCON2	ADC 控制寄存器 2
	SEQCON	映像控制寄存器
	ADCH1	设置 AD 信道配置寄存器 1
	ADCH2	设置 AD 信道配置 2
	SEQCHx	通道寄存器，x=0-7
	ADDxL	ADC 结果低位寄存器，x=0-7
	ADDxH	ADC 结果高位寄存器，x=0-7

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：进出中断时需对 INSCON 寄存器进行压栈出栈操作。

3. ADC 设置

软件启动 ADC 转换步骤：

1. 使能ADC模块
2. 设置转换序列，包括信道数设置以及信道中模拟输入的选择
3. 清0ADCIF
4. GO/$\overline{\text{DONE}}$置1开始ADC转换
5. GO/$\overline{\text{DONE}}$为0或者ADCIF为1，如果ADC中断使能，则进入中断处理程序，软件清0 ADCIF
6. 通过设置映射寄存器，依次读取出转换序列中各通道结果寄存器ADDxH/ADDxL的转换数据
7. 重复2~6开始另一次转换

硬件启动 ADC 转换步骤：

1. 使能ADC模块
2. 设置转换序列，包括信道数设置以及信道中模拟输入的选择
3. 设置触发源
4. 清0ADCIF
5. 若ADCIF为1，如果ADC中断使能，则进入中断处理程序，软件清0ADCIF
6. 通过设置映射寄存器，依次读取出转换序列中各通道结果寄存器ADDxH/ADDxL的转换数据

4. 应用实例

4.1. 要求

连续顺序采样 AN7，AN6, AN5, AN4, AN3, AN2, AN1，AN0 端口的输入电压，Vref = Vdd，转换结果放在 ADC_res 数组中。

4.2. 例程

```
#include "SH79F6481A.h"
#include "intrins.h"
#include "cpu.h"
UINT16 ADC_res[8];
void init_adc()
{
    UCHAR i;
    ADCON1 = 0x80;    //ADC 参考电压选择 VDD，无触发
    ADT=0x21;         //系统时钟 24M Sysclk, 500k sps.
    ADCON2 = 0x72;    // 8 路采样通道采样间隔时间为:4Tad
    ADCH1 = 0xff;     //P3.7~P3.4, P4.3~P4.0  作为 ADC 采样通道 AN7~AN0
//  配置通道采样序列为：AN7,AN6, AN5, AN4, AN3, AN2, AN1, AN0
    SEQCON = 0x0;    //配置 SEQCH0
    SEQCHX = 0x7;
    SEQCON = 0x01;   //配置 SEQCH1
    SEQCHX = 0x06;
    SEQCON = 0x02;   //配置 SEQCH2
    SEQCHX = 0x05;
    SEQCON = 0x03;   //配置 SEQCH3
    SEQCHX = 0x04;
    SEQCON = 0x04;   //配置 SEQCH4
    SEQCHX = 0x03;
    SEQCON = 0x05;   //配置 SEQCH5
    SEQCHX = 0x02;
    SEQCON = 0x06;   //配置 SEQCH6
    SEQCHX = 0x01;
    SEQCON = 0x07;   //配置 SEQCH7
    SEQCHX = 0x0;
    SEQCON &= 0x7f;   //转换结果左对齐
    ADCON1 |= 0x01;     //开始转换
```

```
while(ADCON1 & 0x01);     //检测是否转换完成
for(i = 0; i < 8; i++)    //获取转换结果
{
      SEQCON = i;
      ADC_res[i] = ((ADDXH << 4) + (ADDXL >> 4));
}    }
```

九、SH79F6481A PCA 应用指南

1. 概述

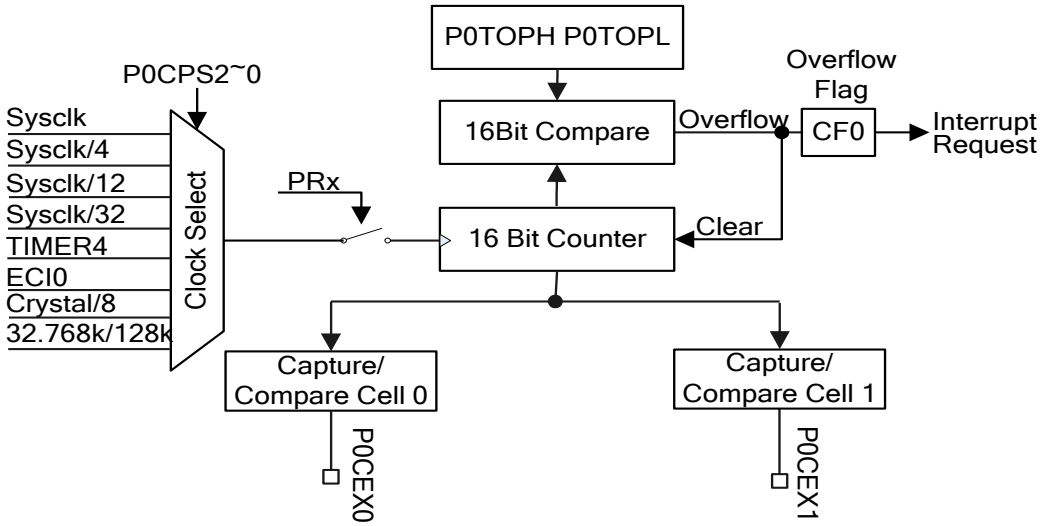
SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 PCA 特性为：

- SH79F6481A有1个16位定时器PCA0，具有两路独立比较捕捉模块；
- 可以实现相位修正，相频修正

说明：①小下标 n 为比较捕捉模块序号，例如 P0CEXn（n=0、1），下文将统一使用 P0CEXn，不再说明 n 的值。

可编程计数器阵列 PCA0 提供增强的定时器功能,与标准 8051 的计数器/定时器相比,它需要较少的 CPU 干预。PCA0 由一个专用的 16 位计数器/定时器和 2 个 16 位捕捉/比较模块组成，PCA0 的原理框图如下所示，每个捕捉/比较模块有其自己的 I/O 线（P0CEXn（n=0，1）。



16 位的 PCA0 计数器/定时器内部集成 16 位的计数单元， 16 位的计数溢出值寄存器由 P0TOPH 和 P0TOPL 组成，用户可以自由地配置 P0TOP 寄存器定义计数器的溢出值，P0TOP 寄存器上电初始值为 0xFFFF。

16 位定时器/计数器是 PCA0 最基本，也是各个模块正常运转必不可少的单元。通过 PCACON 寄存器的 PR0 位可以开启/禁止定时器/计数器工作，当 PR0 设置为逻辑‘0’时，定时器/计数器 16 位 counter 也被强制清‘0’。当计时器从 0x0000 向 P0TOP 计数溢出（P0TOP 到 0x0000 的同一个系统时钟里，即单斜坡模式）或计数器从 P0TOP 递减计数为 0x0000（PCA0 Counter 工作在双斜坡模式）时，P0CF 寄存器中的溢出标志（CF0）被置为逻辑’1’并产生一个中断请求（P0CMD 中 ECF0 位设置为逻辑‘1’即可允许 CF0 标志产生中断请求）。当 CPU 转向中断服务程序时，CF0 位不能被硬件自动清除，必须用软件清除。

注意：16 位寄存器 P0TOPH 和 P0TOPL、P0CPHn 和 P0CPLn 读写操作遵循以下顺序：

写操作：先高位后低位

读操作：读 P0TOPH 和 P0TOPL、P0CPHn 和 P0CPLn 对 PCA0 计数无影响。

2. 控制寄存器

PCA 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	P0CF	PCA 中断标志寄存器
	PCACON	PCA 计数使能寄存器
	P0CMD	PCA 方式寄存器
	P0CPMn	PCA 比较捕捉寄存器
	P0FORCE	比较捕捉模块强制输出控制寄存器
	P0TOPL	PCA0 计数最大值低字节
	P0TOPH	PCA0 计数最大值高字节
	P0CPLn	PCA 比较捕捉模块低字节
	P0CPHn	PCA 比较捕捉模块高字节

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注意：*n 表示比较捕捉模块的序号；进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. PCA 设置

3.1. PCA 时钟设置

PCA0 的计数器/定时器有一个可编程选择时钟源：系统时钟、系统时钟/4、系统时钟/12、系统时钟/32、定时器 4 溢出或 ECI0 输入引脚上的外部时钟信号、内建 128kHzRC,通过 P0CMD 寄存器中的 P0CPS2-P0CPS0 位选择定时器/计数器的时钟源（详见 DATASHEET），如下表所示。

PxCPS2	PxCPS1	PxCPS0	时钟源
0	0	0	系统时钟
0	0	1	系统时钟的 4 分频
0	1	0	系统时钟的 12 分频
0	1	1	系统时钟的 32 分频
1	0	0	定时器 4 溢出
1	0	1	系统时钟
1	1	0	ECI0 下降沿（最大速率=系统时钟频率/4）
1	1	1	内建 128kHzRC

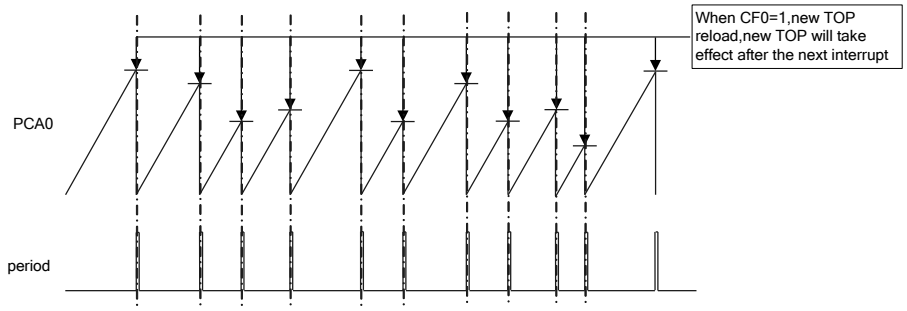
注意：

(1) 系统时钟周期不得低于计数时钟周期的4倍频（计数时钟为系统时钟时除外），否则PCA0 Counter将不能正确计数。

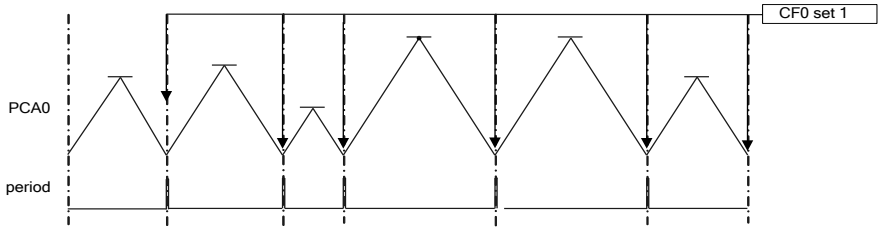
(2) 当时钟源选择内建128kHzRC时，系统时钟必须选择双时钟OP_OSC = 0011。

3.2. PCA 计数方式设置

PCA0 的计数器/定时器有一个计数方式可选择的控制位，通过 P0CMD 寄存器中的 P0SDEN 为 0 表示单斜坡，P0SDEN 为 1 表示双斜坡。单斜坡和双斜坡的时序波形图如下所示：



单斜坡时序图



双斜坡示意图

3.3. PCA 模式设置

PCA0 计数器挂载 2 路捕捉/比较模块均可以实现增强功能。每个捕捉/比较模块 n 都可被配置为独立工作，每个模块在系统控制器中都有属于自己的特殊功能寄存器（SFR），这些寄存器用于配置模块的工作方式和与模块交换数据。可以通过配置各自模块 P0CPMn 寄存器中 P0SMPn 和 P0SMNn 两位使能该模块工作在以下 4 种工作模式之一：边沿触发捕捉、软件定时器、频率输出、PWM 输出模式。模式选择表如下所示：

模式	PxSDEN	P0SMPn	P0SMNn	P0FSPn	P0FSNn	功能说明
Mode0	0	0	0	0	X	正沿触发捕捉（单斜坡）
				1	0	负沿触发捕捉（单斜坡）
				1	1	任意沿触发捕捉（单斜坡）
Mode1	0	0	1	0	X	连续软件定时（单斜坡）
				1	X	单次软件定时（单斜坡）
Mode2	0	1	0	X	X	频率输出（单斜坡）
Mode3	0	1	1	0	0	8 位 PWM（单斜坡）
				0	1	16 位 PWM（单斜坡）
	1			1	0	16 位相位修正 PWM（双斜坡）
	1			1	1	16 位相频修正 PWM（双斜坡）
其它						PCA0 Counter 正确计数，但比较/捕捉模块不工作。

注意： *1 X 表示任意;*

2 当 PCA0 被设定为两种斜坡中的一种时，比较捕捉模块的另一种斜坡模式即使被配置也是无效的。

3 改变 P0TOP 值必须保证新的 P0TOP 值不小于所有比较寄存器的数值。

4 比较/捕捉模块作为 PWM 输出功能时，若 P0CPHn 等于 0x00，输出一直保持为低电平；若 P0CPHn 等于 P0TOP，输出则保持为高电平。引脚取反则输出正好相反；

5 PCA0 的所有比较/捕捉模块只能工作于同一斜坡模式（例：PCA0 的比较/捕捉模块 0 和比较/捕捉模块 1 只能工作在同一斜坡模式）。

3.4. PCA 模式 0 设置

模式 0 称为边沿触发的捕捉模式，在该方式，P0CEXn 引脚上出现的电平跳变将捕捉 PCA0 计数器/定时器的当前计数值并将其装入到对应模块的 16 位捕捉/比较寄存器（P0CPLn 和 P0CPHn ）中。

模式 0：需配置 P0CPMn 的 bit7:bit6=00. （P0SMPn：P0SMNn）

正沿触发捕获：需配置 P0CPMn 的 bit5:bit4=00 或 01 (P0FSPn:P0FSNn)

负沿触发捕获：需配置 P0CPMn 的 bit5:bit4=10

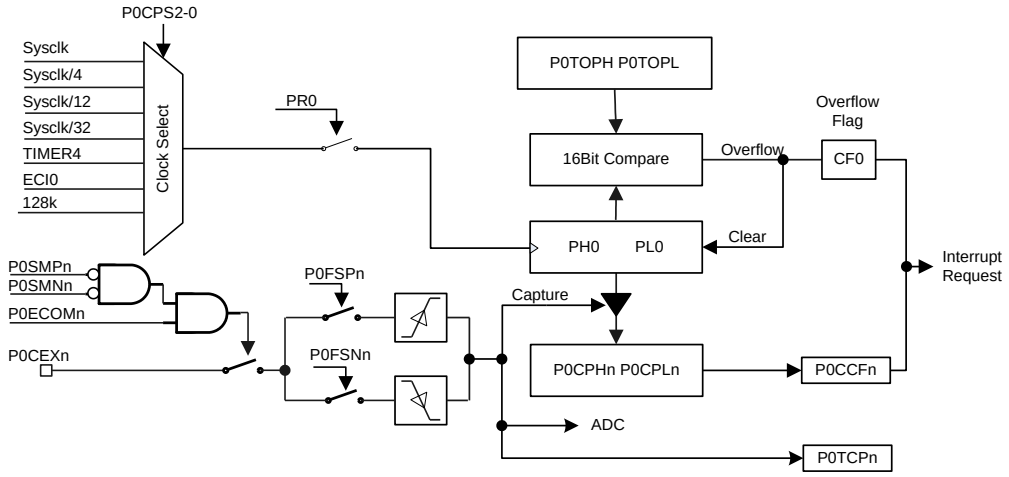
双沿触发捕获：需配置 P0CPMn 的 bit5:bit4=11

使能比较捕捉模块：需配置 P0CPMn 的 bit3=1 (P0ECOMn)

使能比较捕捉模块中断使能：P0CPMn 的 bit1=1

详细的解释参见 SH79F6481A DATASHEET 寄存器 P0CPMn 的描述。

捕获的原理框图如下：



PCA0 捕捉方式原理框图

3.5. PCA 模式 1 设置

模式 1 称之为软件定时方式，软件定时器方式也称为比较输出方式（通过配置 P0SMPn：P0SMNn=01 使能该方式）。在该方式，P0FSPn ：P0FSNn =0x，可以实现连续软件定时，PCA0 将计数器/定时器的计数值与模块的 16 位捕捉/比较寄存器（P0CPHn 和 P0CPLn）进行比较。当发生匹配时，P0CF 中的捕捉/比较标志（P0CCFn）被置为逻辑‘1’（当 P0MATn = 1 时）并产生一个中断请求（如果 P0CCFn 中断被允许），并且模块的 P0CEXn 引脚上的逻辑电平将发生变化（设置 P0TCPn 位使能该功能）。当 CPU 转向中断服务程序时，P0CCFn 位不能被硬件自动清除，必须用软件清 0。

模式 1：需配置 P0CPMn 的 bit7:bit6=01　　（P0SMPn：P0SMNn）

连续软件定时：需配置 P0CPMn 的 bit5:bit4=00 或 01　　(P0FSPn:P0FSNn)

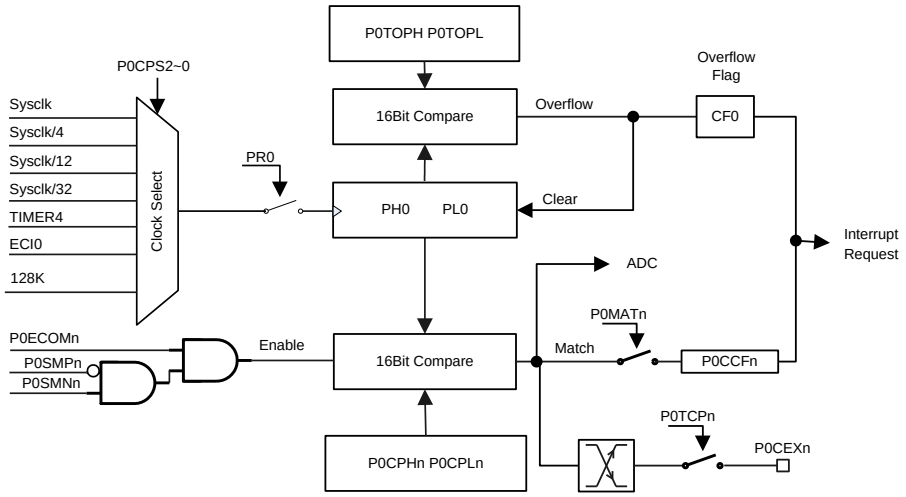
单次软件定时：需配置 P0CPMn 的 bit5:bit4=10 或 11　　(P0FSPn:P0FSNn)

使能比较捕捉模块：需配置 P0CPMn 的 bit3=1　　(P0ECOMn)

P0CEXn 引脚输出电平：需配置 P0CPMn 的 bit2=1　　(P0TCPn)

详细的解释参见 SH79F6481A DATASHEET 寄存器 P0CPMn 的描述。

软件定时原理框图如下：



软件定时原理框图

3.6. PCA 模式 2 设置

模式 2 称为频率输出方式，频率输出方式可在模块的 P0CEXn 引脚产生可编程频率的方波(配置 P0SMPn:P0SMNn=10 使能该模式，在此模式下，P0CPn 寄存器的更新不使用双缓冲机制)。捕捉/比较模块的高字节 P0CPH0 保持输出电平改变前要计的 PCA 时钟数。所产生的方波的频率 FP0CEXn= FPCA0 /(2 × P0CPHn)

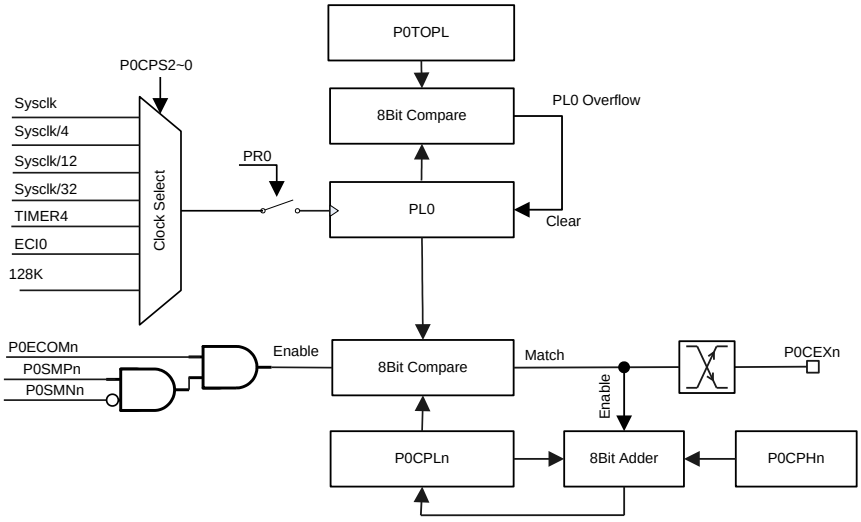
注：对于该方程，P0CPHn 中的值为 0x00 时，相当于 256。

捕捉/比较模块的低字节 P0CPLn 与 PCA0 计数器的低字节 PLx 进行比较；若两者匹配时，P0CEXn 引脚的电平发生改变，同时高字节 P0CPHn 中的偏移值被加到 P0CPLn，PLx 继续计数直到再次匹配，P0CEXn 引脚的电平改变，周而复始，P0CEXn 引脚输出频率由 P0CPH0 控制。如果 PCA0 的某个比较/捕捉模块使能该模式，P0TOPL 的值固定为 0XFF，用户可以配置 P0TOPH 值来改变计数最大值。

模式 2：需配置 P0CPMn 的 bit7:bit6=10　　（P0SMPn：P0SMNn）

使能比较捕捉模块：需配置 P0CPMn 的 bit3=1　　(P0ECOMn)

频率输出原理框图如下：



频率输出原理图

3.7. PCA 模式 3 设置

模式三称为 PWM 模式，PWM 模式分为四种，如下表所示：

P0FSPn	P0FSNn	功能说明
0	0	8 位 PWM（单斜坡）
0	1	16 位 PWM（单斜坡）
1	0	16 位相位修正 PWM（双斜坡）
1	1	16 位相频修正 PWM（双斜坡）

模式 3：需配置 P0CPMn 的 bit7:bit6=11

PWM 模式需配置 bit5:bit4 (P0FSPn:P0FSNn),如上表所示。

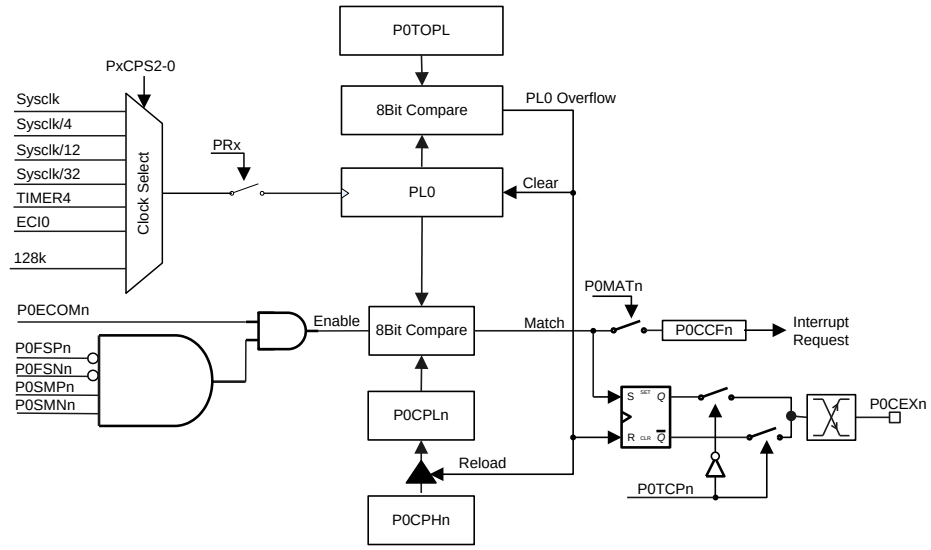
3.7.1 8 位 PWM 模式（单缓冲机制，单斜坡）

当比较/捕捉模块工作在 8 位 PWM 功能时，PCA0 Counter 低 8 位 PLx 从 0x00 向 P0TOPL 递增计数（单斜坡模式），当 PLx 溢出时（从 0xFF 到 0x00），保存在 P0CPHn 中的值被自动装入到 P0CPLn，这个过程不需软件干预，当 P0TCPn =0 时，PCA 计数器/定时器的低字节（PLx）与 P0CPLn 中的值相等时， P0CEXn 引脚上的输出被清‘0’；当 PLx 中的计数值溢出时，P0CEXn 输出被置 ‘1’；当 P0TCPn =1 时，P0CEXn 引脚输出极性相反的波形。

8 位 PWM 方式的占空比 Duty=（256-（P0CPHn+1））/256。

如果 PCA0 的某个比较/捕捉模块使能该模式，P0TOPL 的值固定为 0XFF，用户可以配置 P0TOPH 值来改变计数最大值，但不影响 8 位 PWM 输出周期。

原理框图如下所示。详情参见 SH79F6481A DATASHEET。



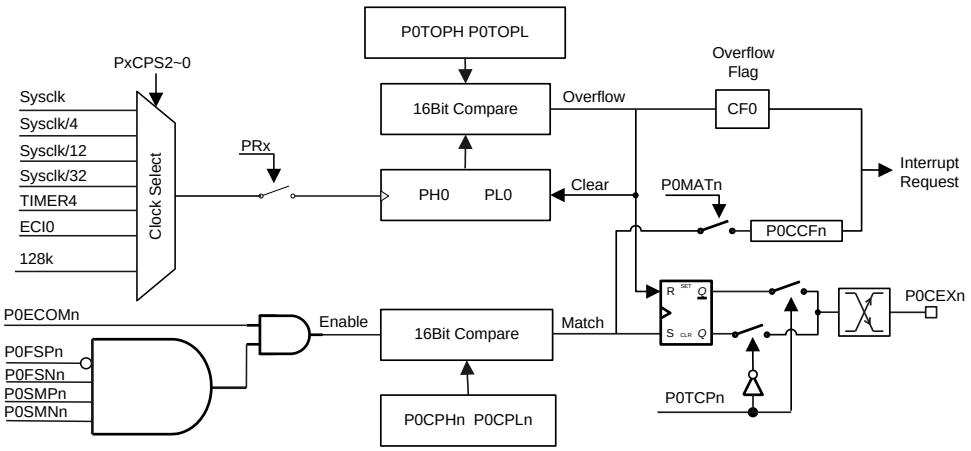
8 位脉宽调制器（PWM）方式原理框图

3.7.2 16 位 PWM 模式（单缓冲机制，单斜坡）

16 位脉宽调制 PWM 同 8 位 PWM 模式类似，都是基于 PCA0 Counter 的单沿计数模式。在该方式下，16 位捕捉/比较模块 PxCPn 用来定义 PWM 信号低电平时间的 PCA0 时钟数。当 P0TCPn =0 时，PCA0 计数器与模块的匹配寄存器 PxCPn 值匹配时，P0CEXn 的输出被置为清‘0’；当计数器溢出时，P0CEXn 输出被置‘1’，当 P0TCPn =1 时，P0CEXn 引脚输出极性相反的波形。

16 位 PWM 方式的占空比 Duty=（65536-（PxCPn+1））/65536

原理框图如下所示。详情参见 SH79F6481A DATASHEET。



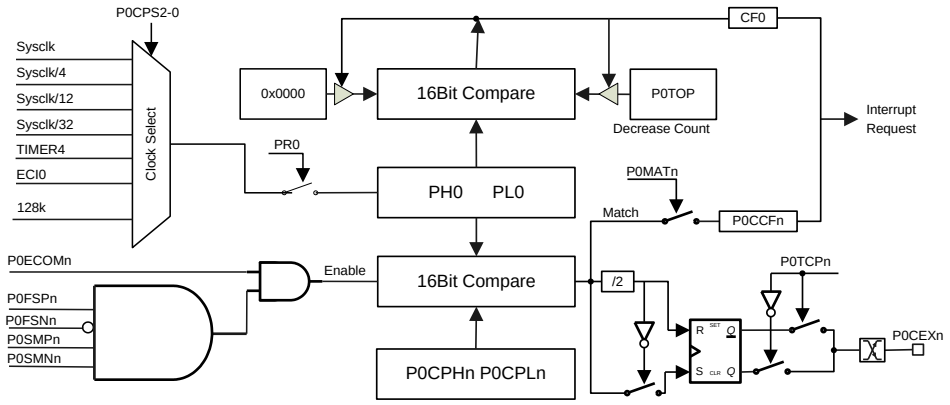
16 位 PWM 模式框图

3.7.3 16 位相位修正 PWM 模式（双缓冲机制，双斜坡，TOP 端更新）

相位修正 PWM(XPWM)功能为用户提供了一个获得高精度的、相位准确的 PWM 波形的方法。此模式是基于双斜坡操作。即计时器重复地从 0x0000 计到 P0TOP，然后又从 P0TOP 倒退回到 0x0000。当 P0TCPn=0 时，当计时器往 P0TOP 计数时若 PCA0 Counter 与 PxCPn 匹配，P0CEXn 将清零为低电

平；而在计时器往 0x0000 计数时若 PCA0 Counter 与 P0CPn 匹配，P0CEXn 将置位为高电平。当 P0TCPn =1 时，P0CEXn 引脚输出极性相反的波形。

工作于相位修正 PWM 模式时，计数器的数值一直累加到 P0TOP 值，然后改变计数方向。在一个定时器时钟周期里 PCA0 值等于 P0TOP 值，然后在下一次计数到来时 P0TOP 和 P0CPn 将得到更新。原理框图如下所示。详情参见 SH79F6481A DATASHEET。



16 位相位修正模式框图

3.7.4 16 位相频修正 PWM 模式（双缓冲机制，双斜坡，BOTTOM 端更新）

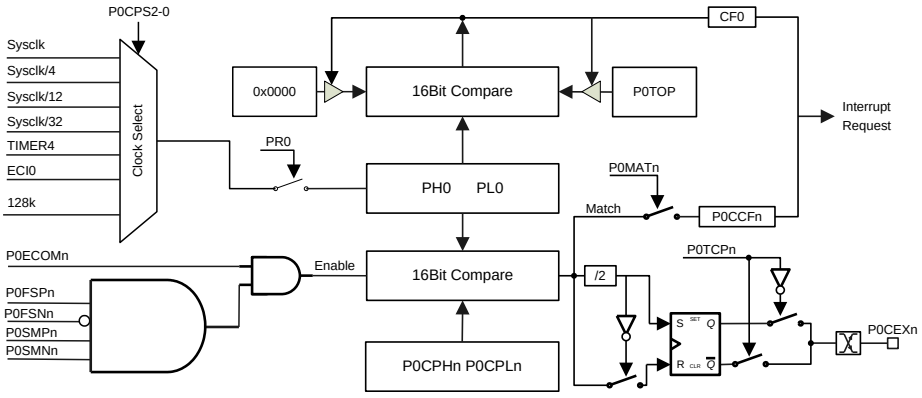
相位与频率修正 PWM(XPPWM)模式以下简称相频修正 PWM 模式。与相位修正模式类似，该功能也是基于双斜坡

操作。XPPWM 可以产生高精度的、相位与频率都准确的 PWM 波形。其实现原理框图如下图 XXX 所示。计时器重复地从 0x0000 计到 P0TOP，然后又从 P0TOP 倒计数到 0X0000。P0TCPn=0 时，当计时器往 P0TOP 计数时若 PCA0 Counter 与 PxCPn

匹配，P0CEXn 将清零为低电平；而在计时器往 0x0000 计数时若 PCA0 Counter 与 PxCPn 匹配，P0CEXn 将置位为高电平。当 P0TCPn=1 时，P0CEXn 引脚输出极性相反的波形。

相频修正 PWM 输出的 PWM 在所有的周期中均为对称的信号，这是由于相位修正 PWM 是在 P0TOP 点更新 PxCPn 和 P0TOP 寄存器值，而相频修正 PWM 则是在 0x0000 点更新 PxCPn 和 P0TOP 寄存器的值。

原理框图如下所示。详情参见 SH79F6481A DATASHEET。



16 位相频修正模式框图

4. 应用实例

4.1. 方式 0 例程

```
#include "SH79F6481A.h"
uchar Val_l;
uchar Val_h;
void main()
{
    IEN0 |=0x80;          //开总中断 EA
    IEN2 |= 0x20;         //允许 PCA0 中断
    INSCON=0x40;          //切换至 BANK1
    P0CMD=0x00;           //选择系统时钟，单斜坡方式，禁止溢出中断
    P0CPM0=0x39;          //方式 0，双沿触发捕获，允许捕获中断
    P0TOPH=0x08;
    P0TOPL=0x00;
    PCACON=0x1;           //启动计数
    While(1);
}
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);       // 进中断对 INSCON 压栈
    INSCON=0x40;
    Val_l=P0CPL0;         //从 P0CEX0 灌上升沿或下降沿后将捕获的值赋给变量
    Val_h=P0CPH0;
    P0CF =0x0;
    _pop_(INSCON);        // 出中断对 INSCON 出栈
}
```

4.2. 方式 1 例程

要求：P0CEX0 输出方波

```
#include "SH79F6481A.h"
void main()
{
    IEN0 |=0x80;          //开总中断 EA
    IEN2 |= 0x20;         //允许 PCA0 中断
```

```
    INSCON=0x40;
    P0CMD=0x00;  //选择系统时钟，单斜坡方式，禁止溢出中断
    P0CPM0=0x47;  //方式 1，允许 P0CEX0 输出波形，允许比较捕捉模块中断
    P0TOPH=0x40;
    P0TOPL=0x20;
    P0CPH0=0x00;
    P0CPL0=0x20;
    P0CPM0|=0x08;  //使能比较捕捉模块
    PCACON=0x1;    //启动计数
    While(1);
}
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);    // 进中断对 INSCON 压栈
    INSCON=0x40;
    P0CF =0x0;
    _pop_(INSCON);     // 出中断对 INSCON 出栈
}
```

4.3. 方式 2 例程

要求：P0CEX0 输出一定频率的方波

```
#include "SH79F6481A.h"
void main()
{
    IEN0 |= 0x80;       //开总中断 EA
    IEN2 |= 0x20;       //允许 PCA0 中断
    INSCON=0x40;
    P0CMD=0x00;       //选择系统时钟，单斜坡方式，禁止溢出中断
    P0CPM0=0x80;      //方式 2
    P0CPH0=0x10;
    P0CPL0=0x20;
    PCACON=0x1;
    P0CPM0|=0x08;    //使能比较捕捉模块
    PCACON=0x1;      //启动计数
    While(1);
}
```

```
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);     // 进中断对 INSCON 压栈
    INSCON=0x40;
    P0CF =0x0;
    _pop_(INSCON);      // 出中断对 INSCON 出栈
}
```

4.4. 方式 3 例程

4.4.1 8 位 PWM 方式

要求：P0CEX0 和 P0CEX1 输出一对无死区的方波。

```
#include "SH79F6481A.h"
void main()
{
    IEN0 |= 0x80;         //EA
    IEN2 |= 0x20;        //允许 PCA0 中断
    INSCON=0x40;
    P0CMD=0x80;         //选择系统时钟，单斜坡方式，允许溢出中断
    P0CPM0=0xC3;       //比较捕捉模块 0 方式 3（8 位 PWM）,允许比较捕捉模块中断
    P0CPH0=0x30;
    P0CPL0=0x30;
    P0CPM0|=0x08;      //使能比较捕捉模块 0
    P0CPM1=0xC3;       // 比较捕捉模块 1 方式 3（8 位 PWM）,允许比较捕捉模块中断
    P0CPH1=0x30;
    P0CPL1=0x30;
    P0CPM1|=0x0c;      //使能比较捕捉模块 1
    PCACON=0x1;        //启动计数
    While(1);

}
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);     // 进中断对 INSCON 压栈
    INSCON=0x40;
    P0CF =0x0;
```

```
    _pop_(INSCON);      // 出中断对 INSCON 出栈
}
```

4.4.2 16 位 PWM 方式

```
#include "SH79F6481A.h"
要求：P0CEX0 和 P0CEX1 输出一对无死区的方波。
void main()
{
    IEN0 |= 0x80;    // EA
    IEN2 |= 0x20;    //允许 PCA0 中断
    INSCON=0x40;
    P0CMD=0x80;   //选择系统时钟，单斜坡方式，允许溢出中断
    P0CPM0=0xD3; //比较捕捉模块 0 方式 3（16 位 PWM）,允许比较捕捉模块中断
    P0CPH0=0x30;
    P0CPL0=0x30;
    P0CPM0|=0x08; //使能比较捕捉模块 0
    P0CPM1=0xD3; //  比较捕捉模块 1 方式 3（16 位 PWM）,允许比较捕捉模块中断
    P0CPH1=0x30;
    P0CPL1=0x30;
    P0CPM1|=0x0c; //使能比较捕捉模块 1
    PCACON=0x1;   //启动计数
    While(1);
}
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);    // 进中断对 INSCON 压栈
    INSCON=0x40;
    P0CF =0x0;
    _pop_(INSCON);      // 出中断对 INSCON 出栈
}
```

4.4.3 16 位相位调整 PWM 方式

```
要求：P0CEX0 和 P0CEX1 输出一对无死区的方波。
void main()
{
IEN0 |= 0x80;       //EA
```

```
IEN2 |= 0x20;        //允许 PCA0 中断
INSCON=0x40;
P0CMD=0xC0;      //选择系统时钟，双斜坡方式，允许溢出中断
P0TOPH=0x01;
P0TOPL=0x00;
P0CPM0=0xE3; //比较捕捉模块 0 方式 3（16 位相位修正 PWM）,允许比较捕捉模块中断
P0CPH0=0x0;
P0CPL0=0x90;
P0CPM0|=0x08; //使能比较捕捉模块 0
P0CPM1=0xE3; //比较捕捉模块 1 方式 3（16 位相位修正 PWM）,允许比较捕捉模块中断
P0CPH1=0x0;
P0CPL1=0x90;
P0CPM1|=0x0c; //使能比较捕捉模块 1
PCACON=0x1;   //启动计数
While(1);
}
void INT_PCA0(void) interrupt 20
{
    _push_(INSCON);     // 进中断对 INSCON 压栈
    INSCON=0x40;
    P0CF =0x0;
    _pop_(INSCON);      // 出中断对 INSCON 出栈
}
```

4.4.4 16 位相频调整 PWM 方式

要求：P0CEX0 和 P0CEX1 输出一对无死区的方波。

```
void main()
{
IEN0 |= 0x80;        //EA
IEN2 |= 0x20;        //允许 PCA0 中断
INSCON=0x40;
P0CMD=0xC0;      //选择系统时钟，双斜坡方式，允许溢出中断
P0TOPH=0x01;
P0TOPL=0x00;
P0CPM0=0xF3;     //比较捕捉模块 0 方式 3（16 位相频修正 PWM）,允许比较捕捉模块中断
```

```
P0CPH0=0x0;
P0CPL0=0x90;
P0CPM0|=0x08;    //使能比较捕捉模块 0
P0CPM1=0xF3;     //比较捕捉模块 0 方式 3（16 位相频修正 PWM）,允许比较捕捉模块中断
P0CPH1=0x0;
P0CPL1=0x90;
P0CPM1|=0x0c;    //使能比较捕捉模块 1
PCACON=0x1;      //启动计数
While(1);
}
void INT_PCA0(void) interrupt 20
{
      _push_(INSCON);     // 进中断对 INSCON 压栈
       INSCON=0x40;
       P0CF =0x0;
       _pop_(INSCON);      // 出中断对 INSCON 出栈
}
```

十、SH79F6481A LED 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

LED 驱动器包含一个控制器，8 个 COM 输出引脚和 16 个 SEG 输出引脚。支持 1/1~1/8 占空比电压驱动方式。

LED 驱动器有两种运行模式。

模式 1：亮灭 LED 模式

当 LED 驱动器工作在亮灭 LED 模式时，每一个 LEDRAM 位控制一个 LED 灯，当 LEDRAM 位为 0 时，LED 熄灭，当 LEDRAM 位为 1 时，LED 点亮；在 LED 一帧或者一个 COM 扫描结束后，LED 驱动器对应的中断标志位 LEDIF 或者 COMIF 标志位置 1。

模式 2：调光 LED 模式

当 LED 驱动器工作在调光 LED 模式时，每一个 LEDRAM byte 控制正在扫描的 COM 周期内 SEG 的占空比；该占空比总共可以 256 档可选；当 LEDRAM byte 为 0xff 时，SEG 输出最大占空比，当 LEDRAM byte 为 0x00 时，SEG 输出最小占空比；当 LEDRAM byte 为 0x00~0xff 中间值时，SEG 输出 相对应的占空比；针对 LEDRAM byte 的修改，会在下一个 COM 扫描周期生效；

在 LED 每一个 COM 周期扫描结束后，立即进行下一个 COM 周期的扫描，LED 驱动器对应的中断标志位 COMIF 会置 1，由于 LED 驱动器工作在模式 2，所以在 LED 驱动使能后，COMIF 中断标志位也会置 1，方便用户修改 LEDRAM 的值；在扫描完成一帧后，LED 驱动器对应的中断标志位 LEDIF 会置 1；因此在使用 LED 中断方式检测时，LED 模块使能之前，将所需的 LED 中断源及总中断源打开。

无论 LED 驱动器工作在模式 1 或者模式 2，COM 和 SEG 的非选电平都为浮动电平， COM 的输出有效电平为低电平，SEG 的输出有效电平为输出高电平。LEDCOM 寄存器控制 LED 驱动器的 COM 个数；SEG01/SEG02 寄存器控制 LED 驱动器 SEG 的个数； DISPCOM 寄存器可以设置每一个 COM 周期的宽度；为了防止两个 COM 显示切换之间的不确定状态，LED 驱动器会在 2 个 COM 之间插入一段死区时间，在死区时间内，COM 输出浮动电平，死区时间的长度为 N 个系统时钟宽度，可以通过 LEDDZ 寄存器来设置；

在上电复位、引脚复位、低电压复位或看门狗复位期间，LED 被关闭。当 LED 被关闭时，被选中 COM 和 SEG 输出为浮动电平。

IDLE 模式下，LED 驱动器保持工作，Power-Down 模式下，LED 驱动器关闭。在双时钟模式下，LED 工作在高频时钟，需要设置高频时钟下的 LED 时钟寄存器 DISCOM，切换到低频时钟时，需要将 LED 时钟寄存器 DISCOM 做相应修改。

2. 控制寄存器

LED 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	LEDCON	控制寄存器
	DISCOM	COM 扫描宽度控制寄存器
	LEDDZ	死区宽度控制寄存器
	SEG01/SEG02	SEG 模式选择寄存器
	LEDCOM	COM 模式选择寄存器

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：*进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. LED 设置

模式 1 举例：

TB 为 LED 单 COM 扫描宽度，TSYS 为系统时钟宽度, TLED 为 LED 扫描时间宽度

TB = TSYS*256*DISCOM

TLED = TB *S

S 为 LED 扫描的 COM 数量：扫描 4COM 即 S=4，扫描 5COM 即 S=5，以此类推。

以需要显示的 LED 帧频 200HZ(5ms)为例,当 LED 为 5COM 且系统时钟为 RC 24MHz 时：

TB= TSYS *256* DISCOM

=41.66ns*256*94(5EH)

=1002506ns=1002.506u s=1.002ms

需要设置 TLED =5ms

TLED= TB*S =1.002ms*5=5.010ms

设置需要开启的中断功能(ELED=1)，可选择帧中断(LEDFY=1) (必须)或者 COM 中断(LEDCY=1)(可选)。开启 LED 模块并启动 LED 扫描，当一帧/一 COM 波形扫描结束后，相应选择的帧中断(LEDIF=1)/COM 中断 (COMIF=1)中断标志会置起。(SEG 和 COM 的非选电平都为浮动电平，COM 有效电平为低电平，SEG 有效电平为高电平。)

模式 2 举例：

TB 为 LED 单 COM 扫描宽度，TSYS 为系统时钟宽度, TLED 为 LED 扫描时间宽度

TB = TSYS*256*DISCOM

TLED = TB *S

S 为 LED 扫描的 COM 数量：扫描 4COM 即 S=4，扫描 5COM 即 S=5，以此类推。

以需要显示的 LED 帧频 200HZ(5ms)为例,当 LED 为 5COM 且系统时钟为 RC 24MHz 时：

TB= TSYS *256* DISCOM

=41.66ns*256*94(5EH)

=1002506ns=1002.506u s=1.002ms

需要设置 TLED =5ms

TLED= TB*S =1.002ms*5=5.010ms

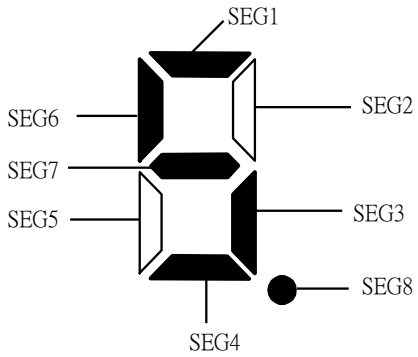
1. 第一个 COM 的 SEG 的波形 SEGXduty(X=0~16)(如果输出满幅设置 0XFF，即第一个 COM 周期显示 100%占空比高电平，如果不显示，设置 0X00，即显示浮动电平，如果设置 0X7F,SEG 输出 50%占空比高电平)，设置需要开启的中断功能（ELED=1），需要选择帧 COM 中断(LEDCY=1)(必须)，或加上帧中断(LEDFY=1)（可选），开启总中断（EA）（必须）后开启 LED 模块。

2. 开启 LED 功能前，设置第一个 COM 的波形，启动 LED 功能后，在 LED 中断(COMIF=1)来临时，改写第二个 COM 周期的 SEG 的波形 SEGXduty(X=0~16)。如果不修改，则显示上周期波形。以此类推，到一帧(LEDIF=1)结束。一帧最后一个 COM 中断来临时，填写下一帧的第一个 COM 的 SEG 的波形。

4. 应用实例

4.1. 亮灭模式（Mode1）例程

要求：选用 COM1-4，SEG1-8，显示四个共阴八段码 LED，显示内容为 1，2，3，4，八段码 LED 的 SEG 定义如下图所示：




```
#include <SH79F6481A.H>
#include <intrins.h>
#include <absacc.h>
xdata unsigned char COM1 _at_ 0x530;
xdata unsigned char COM2 _at_ 0x531;
xdata unsigned char COM3 _at_ 0x532;
xdata unsigned char COM4 _at_ 0x533;

voidLED_init(void)
{
```

```
    CLKCON = 0x00;          //系统时钟  24MHz
    LEDCON = 0x0;           //模式 1
    DISCOM = 0x3f;          //设置单 COM 扫描时间
    LEDDZ = 0x10;           //设置死区时间
    SEG01 = 0xFF;           //LED_S1~LED_S8 复用打开
    LEDCOM = 0x0f;          //LED_C1~LED_C4 复用打开
}

void main(void)
{
    LED_init();
    COM1 = 0x06;        //show 1
    COM2 = 0x5B;        //show 2
    COM3 = 0x4F;        //show 3
    COM4 = 0x666;       //show 4
    LEDCON |= 0x80;  //使能  LED  驱动模块
    while(1);
}
```

4.2. 调光模式（Mode2）例程

要求：选用 COM1-4，SEG1-4，实现在第 1 个周期中每个 SEG 口在 COM1 时输出占空比为 1，COM2 时输出占空比为 1/2；在第 2 个周期中每个 SEG 口在 COM1 时输出占空比为 1/2，COM2 时输出占空比为 1；不断重复以上过程；

```
#include <SH79F6481A.H>
#include <intrins.h>
#include <absacc.h>
xdata unsigned char LED_RAM[4] _at_ 0x530;
unsigned char Display_Data;

void LED_init(void)
{
    CLKCON = 0x00;          //系统时钟  24MHz
    LEDCON = 0x20;          //模式 2
    DISCOM = 0x3f;          //设置单 COM 扫描时间
    LEDDZ = 0x10;           //设置死区时间
```

```
    SEG01 = 0x0F;           //LED_S1~LED_S4 复用打开
    LEDCOM = 0x03;          //LED_C1~LED_C2 复用打开
    IEN1 = 0X02;        //允许 LED 中断
    ELEDCON = 0X01;         //允许 LED_COM 中断
}

void LED_Dispaly(unsigned char uc_temp)
{
    unsigned char i;
    for(i=0x00;i<4;i++)
{
        LED_RAM[i] = uc_temp;
}
}
void main(void)
{
    LED_init();
    Display_Data = 0xff;        //第一个 COM 显示的 SEG 占空比
    LED_Dispaly(Display_Data);
    EA = 1;                 //使能总中断
    LEDCON |= 0x80;         //使能 LED 驱动模块
    while(1);
}

void INT_LED(void) interrupt 8
{
    _push_(INSCON);
    INSCON = 0X00;
    LEDCON &= 0XF7;         //清除 COM 结束标志位
    Display_Data = Display_Data - 0x40;
    LED_Dispaly(Display_Data);
    _pop_(INSCON);
}
```

十一、SH79F6481A TWI 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A TWI 特性为：

- 两线模式，简单快捷
- 支持主机模式(Master)和从机模式(Slave)
- 允许发送数据(Transmitter)和接收数据(Receiver)
- 支持多主机通讯的仲裁功能
- 具有低电平总线超时判断(Timeout)
- 空闲模式下可唤醒系统
- 地址可编程

2. 控制寄存器

TWI 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	TWICON	控制寄存器
总线超时控制	TWITOUT	总线超时计数寄存器
状态	TWISTA	状态寄存器
高电平控制	TWIFREE	高电平超时计数寄存器
比特率	TWIBR	比特率寄存器
地址	TWIADR	地址寄存器
数据	TWIDAT	数据寄存器
地址屏蔽	TWIAMR	地址屏蔽寄存器

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：*进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. TWI 设置

TWI 功能打开时，相应的 SCL 和 SDA 被配置成开漏输出，开漏输出结构不具有输出高电平的能力。因此，需要上拉电阻（可以外接或者 TWIPCR = 1 打开内部上拉）来配合使用。此时，应保证 IO 口上的电平不超过 VDD+0.3V，否则有可能损坏芯片。

可以通过 LCM 功能，配置 TWI 到不同的 IO 口。

对于 TWI 的相关协议，请参照“SH79F6481A DATASHEET”。

注意：避免 TWI 因为干扰造成 TWI 模块异常，可在外面增加软件 TWI WDT 保护机制。

4. 应用实例

4.1. 要求

SH79F6481A 作为主机，向从机传送数据，分别为“0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70, 0x80”。

4.2. 例程

```
#include <SH79F6481A.H>
#include "intrins.h"
#define SLAVE_Address    0x66                    //  从机器件地址
void    TwiInit(void);
void main()
{
    TwiInit();
    while(1);
}
UCHAR TWI_flag;                                  //防止 TWI 通讯会进入异常，造成 TWI 通讯异常
void TwiInit()
{
//=====System Init==========
    CLKCON = 0x08;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    CLKCON |= 0x04;                              // RC24M
//======TWI LCM==========
    INSCON |=0x40;                               //切换到 Bank1
     TWICR = 0X23;                               // SCK = P2.2    SDA = P2.3
    INSCON &=0xBF;        //切换到 Bank0
//初始化 TIME3，用于避免 TWI 启动异常
    IEN0 |= 0xA0;        //打开定时器 3 中断  打开总中断
    _push_(INSCON);
    INSCON |=0x40;          //切换到 Bank1
    T3CON = 0x00;        //TIM3 时钟为系统时钟
```

```
//配置定时器的初值为 0xff00,溢出时间为 256 个 24M 时钟
    TL3 = 0x00;
    TH3 = 0xff;
    T3CON |=0x04;                          //启动定时器 3
   _pop_(INSCON);
//=======TWI Init==========
    TWITOUT = 0x02;                        // 开启 TWI 内部上拉电阻
    TWIBR = 28;                  // 设置波特率，对应 SCK 时钟：24000/(16+2*28*4)=100K
    TWISTA = 0x04;                         // SCK 时钟设置 4 分频
//=======Configure Slave Address and Response Signal=============
    TWIADR = (SLAVE_Address<<1)&0xFE;  // 配置从机器件地址,禁止响应通用地址
    TWIAMR = 0X00;                         // 不屏蔽地址各位
    Send_ACK;                              // ACK 信号
    TWTFREE   = 0xFF;                      // SCK 高电平超时 Count，
    TWICON    |= 0x01;                     // Enable SCL 高电平超时标志中断允许位
    TWITOUT |= 0xC0;                       // 总线超时计数：200000
    TWISTA    |= 0x01;                     // 使能总线超时中断
    TWICON |= 0x40;                        //使能 TWI 模块
    TWI_flag = 1; //发送 STA 信号前，先置起 TWI 异常标志
    TWICON |= 0x20;
//=========TWI Interrupt=====
    IEN1 |= 0x01;                          // 使能 TWI 中断
    EA = 1;
}
void TWI_Interrupt (void) interrupt 7
{
    unsigned char state;
   unsigned char Send_Len = 0;                    // 发送数据长度
   unsigned char Send_buf[8]={0x10,0x20,0x30,0x40,0x50,0x60,0x70,0x80};//接收数据
    _push_(INSCON);
    TWICON &= 0xDF;
    TWI_flag = 0;       //TWI 工作正常
    state = TWISTA & 0xf8;
    switch(state)
```

```
{
        case 0x08:
            TWIDAT = (SLAVE_Address<<1)&0xFE;
            break;
        case 0x18:
        case 0x20:
            TWIDAT = Send_buf[0];
            break;
        case 0x28:
        case 0x30:
            if(Send_Len<7)
            {
                TWIDAT = Send_buf[Send_Len+1] ;
                Send_Len ++;
            }
            else
            {
                TWICON |= 0x10;
            }
            break;
        /****其它模式****/
        case 0xF8:
            STO = 1;
            Send_ACK;
            break;
        case 0x00:
            STO = 1;
            Send_ACK;
            break;
        default:
            STO = 1;
            Send_ACK;
            break;
}
```

```
    TWICON &= 0x75;                         // Clearn TWINT/TFREE/TOUT
    _pop_(INSCON);
}

unsigned char   TWI_Counter;                //用于 TWI 异常检测
void INT_TIMER3(void) interrupt 5
{
    _push_(INSCON);
    INSCON &=0xBF;          //切换到 Bank0
    if(1 == TWI_flag)
    {
        TWI_Counter++;
        if(200<TWI_Counter)
        {
            TWICON &= 0xBF;   //关闭 TWI
            TWICON |= 0x40;    //开启 TWI
            TWICON |= 0x20;
            TWI_Counter = 0;
        }
    }
    else
    {
       TWI_Counter = 0;
    }
    _pop_(INSCON);

}
```

十二、SH79F6481A LPD 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。SH79F6481A 其 LPD 模块特性为：

- 低电压检测并产生中断
- 可选的LPD检测电压
- LPD含双边去抖动功能，当检测电压高于/低于检测点持续约10ms时，LPD发生并产生中断

低电压检测（LPD）功能用来监测电源电压，如果电压低于指定值时产生内部标志。LPD 功能用来通知 CPU 电源是否被切断或电池是否用尽，因此在电压低于最小工作电压之前，软件可以采取一些保护措施。

在进入 Idle 以及 PD 之前，必须软件关闭 LPD 模块。

2. 控制寄存器

LPD 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	LPDCON	控制寄存器
档位选择	LPDSEL	电压检测档位控制寄存器

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：*进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. LPD 设置

低电压检测主要应用于当系统电压低于某个值的时候，系统认为存在断电的风险，需要做一些保护措施，来防止短时间的掉电，如果在一定时间内，系统重新上电，可以继续掉电前的状态往下运行。使用 LPD 时，需要先设置检测模式和一个合适的电压值，SH79F6481A 提供 2 种检测模式和 16 档检测电压，可以通过 LPDCON 和 LPDSEL 来选择，设置完毕，使能 LPD 模块，可以通过查询 LPDF 或者使能 LPD 中断来快速检测掉电。

4. 应用实例

4.1. 要求

系统工作电压为 5V，当电压小于 3.6V 时系统认为存在掉电风险，若 Vdd 电压小于 3.6V，则将关键参数存于 C0H~CFH 中，然后系统进入 Power-Down 模式。

4.2. 例程

```
#include <SH79F6481A.H>
#include <intrins.h>
```

```
unsigned char outflag = 0;
idata unsigned char temp0 _at_ 0xC0;
idata unsigned char temp1 _at_ 0xC1;
idata unsigned char temp2 _at_ 0xC2;
idata unsigned char temp3 _at_ 0xC3;
idata unsigned char temp4 _at_ 0xC4;
idata unsigned char temp5 _at_ 0xC5;
idata unsigned char temp6 _at_ 0xC6;
idata unsigned char temp7 _at_ 0xC7;
idata unsigned char temp8 _at_ 0xC8;
idata unsigned char temp9 _at_ 0xC9;
idata unsigned char temp10 _at_ 0xCA;
idata unsigned char temp11 _at_ 0xCB;
idata unsigned char temp12 _at_ 0xCC;
idata unsigned char temp13 _at_ 0xCD;
idata unsigned char temp14 _at_ 0xCE;
idata unsigned char temp15 _at_ 0xCF;
unsigned char keydata0;
unsigned char keydata1;
unsigned char keydata2;
unsigned char keydata3;
unsigned char keydata4;
unsigned char keydata5;
unsigned char keydata6;
unsigned char keydata7;
unsigned char keydata8;
unsigned char keydata9;
unsigned char keydata10;
unsigned char keydata11;
unsigned char keydata12;
unsigned char keydata13;
unsigned char keydata14;
unsigned char keydata15;
```

```
void main(void)
{
    CLKCON = 0x00;                  //系统时钟 = 24MHz
    LPDCON = 0x80;                  //使能 LPD 模块
    LPDSEL = 0x08;                  //设置 Vlpd =3.6V

    Delay();                //延时 20us
    LPDCON &= 0xef;         //清标志位
    EA = 1;                 //开启总中断
    IEN1 |= 0x40;           //允许 LPD 中断
    while(1)
    {
        if(outflag == 1)        //LPD 发生
        {
            temp0 = keydata0;
            temp1 = keydata1;
            temp2 = keydata2;
            temp3 = keydata3;
            temp4 = keydata4;
            temp5 = keydata5;
            temp6 = keydata6;
            temp7 = keydata7;
            temp8 = keydata8;
            temp9 = keydata9;
            temp10 = keydata10;
            temp11 = keydata11;
            temp12 = keydata12;
            temp13 = keydata13;
            temp14 = keydata14;
            temp15 = keydata15;
            LPDCON &= 0x7F;                     //关闭 LPD 模块
            SUSLO = 0x55;
            PCON |= 0x02;
            _nop_();
```

```
                _nop_();
                _nop_();
            }
        }
    }
    void LPD_int(void)    interrupt    13
    {
        _push_(INSCON); //  进中断对 INSCON 压栈
        LPDCON &= 0xEF;
        outflag = 1;
        _pop_(INSCON);    //  出中断对 INSCON 出栈
    }
```

十三、SH79F6481A PWM 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

SH79F6481A 其 PWM 驱动器特性为：

- 2路12bit精度PWM模块
- 提供每个PWM周期溢出中断
- 输出极性可选择

SH79F6481A 集成了两路 12 位 PWM 模块。PWM 模块可以产生周期和占空比分别可调整的脉宽调制波形。PWMxCON（x = 0 - 1）寄存器用于控制 PWMx 模块的时钟、波形输出以及周期中断，PWMxPH/L 寄存器用于控制 PWMx 输出波形的周期，PWMxDH/L（x = 0-1）寄存器用于控制 PWMx 模块输出波形的占空比。

在 PWM 输出允许期间，可以修改这三个寄存器，但在下一个 PWM 周期修改才会起作用。

2. 控制寄存器

PWM0 模块使用的所有控制寄存器如下表所示：

类别	缩写符号	功能说明
模块控制	PWM0CON	控制寄存器
	PWM0PH	周期控制寄存器高 4 位
	PWM0PL	周期控制寄存器低 8 位
	PWM0DL	占空比控制寄存器低 8 位
	PWM0DH	占空比控制寄存器高 4 位

针对每个寄存器的详细介绍，请参照“SH79F6481A DATASHEET”

注：*进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

3. PWM0 设置

PWM 编程分为以下几步：

(1) 选择 PWM 模块时钟源。

(2) 通过写适当的值到 PWM 周期控制寄存器（PWMxPH/L）或 PWM 占空比寄存器（PWMxDH/L）设置 PWM 周期/占空比，先设置低位，再设置高位。注意,即使高位数值不变,也要重写一次,否则,低位的修改无效。

(3) 通过设置 PWM 控制寄存器（PWMxCON）的 PWMxS 位选择 PWMx 输出模式(高电平有效或低电平有效)。

(4) 如果 PWM 周期或者占空比需要改变，操作流程如同步骤 2 或者步骤 3 说明。修改后的重载计数器的值在下一个周期开始有效。

4. 应用实例

4.1. 要求

使用 PWM0，产生一个 2KHz，占空比为 50%的 PWM 波驱动蜂鸣器。

4.2. 例程

```
#include <SH79F6481A.H>
#include <intrins.h>
void PWM0_init(void)
{
    CLKCON = 0x00;          //系统时钟 = 24MHz
    PWM0CON = 0x29;             //时钟为系统时钟/32,占空比期间输出高，输出允许
    PWM0PL = 0x77;
    PWM0PH = 0x01;          //PWM0 频率为 2KHz
    PWM0DL = 0xBB;
    PWM0DH = 0x00;          //PWM0 占空比为 50%
    }
void main(void)
{
    PWM0_init();
    PWM0CON |= 0x80;            //使能 PWM0
    while(1);
}
```

PWM1 用法同 PWM0.

十四、SH79F6481A LCM 应用指南

1. 概述

SH79F6481A 是一种高速高效率 8051 可兼容单片机。在同样振荡频率下，较之传统的 8051 芯片它有着运行更快速的优越特性。

- 12 种数字逻辑功能口可以通过数字逻辑可配置模块重新映像到 I/O，且每种功能可在 8 个 IO 口中选择其一。

2. 控制寄存器

LCM 模块所用的寄存器如下表所示：

类别	缩写符号	功能说明
引脚配置寄存器	UART0CR	TXD0 和 RXD0 配置寄存器
	UART1CR	TXD1 和 RXD1 配置寄存器
	TWICR	SCK 和 SDA 配置寄存器
	PWMCR	PWM0 和 PWM1 配置寄存器
	CEXCR	P0CEX1 和 P0CEX0 配置寄存器
	ECICR	ECI0 和 ECI1 配置寄存器

3. 应用实例

3.1. 要求

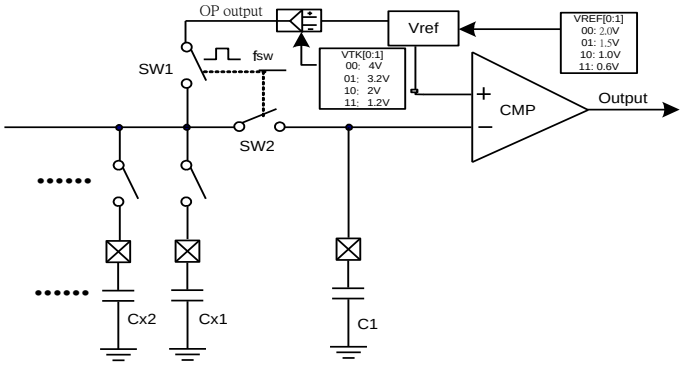
引脚配置为 P0.0 为 TXD0，P0.1 为 RXD0，P2.6 为 TXD1, P2.7 为 RXD1。

3.2. 例程

```
#include <SH79F6481A.H>
#include “intrins.h”
IO_config()
{
    _push_(INSCON);
    INSCON = 0x40;   //选择  Bank1
    UART0CR = 0x01; //P0.0  作为  TXD0, P0.1 作为 RXD0
    UART1CR = 0x01; //P2.6 作为 TXD1, P2.7 作为 RXD1
    _pop_(INSCON);
}
```

十五、SH79F6481A Touch Key 应用指南

1. Touch Key 触摸按键功能



触摸框图

2. Touch Key 启动流程

(1) 选择需要扫描的按键通道；

(2) 寄存器 TKCON 位置 1，允许触摸按键模块工作；

(3) 设置开关频率、参考电压 V_{REF} 和按键采样次数和扫描顺序；

(4) 设置 28 位放大系数寄存器；

(5) 软件延时 10uS；

(6) 寄存器 TKGO/$\overline{DONE}$位置 1，启动按键扫描；

(7) 中断产生，TKGO 硬件自动清 0；

(8) 中断标志位判断：IFERR，IFGO，IFAVE，IFCOUNT

如果 IFAVE = 1，读数据寄存器 500H – 52FH，程序保存数据结果，执行步骤 9；

如果 IFERR = 1，数据寄存器运算溢出错误，清 IFERR 和标志位，重新设置放大系数寄存器值，减小放大系数值，返回步骤 5 重新启动扫描；

如果 IFGO = 1，按键控制器启动错误，清 IFGO 和标志位，返回步骤 5 重新启动扫描；

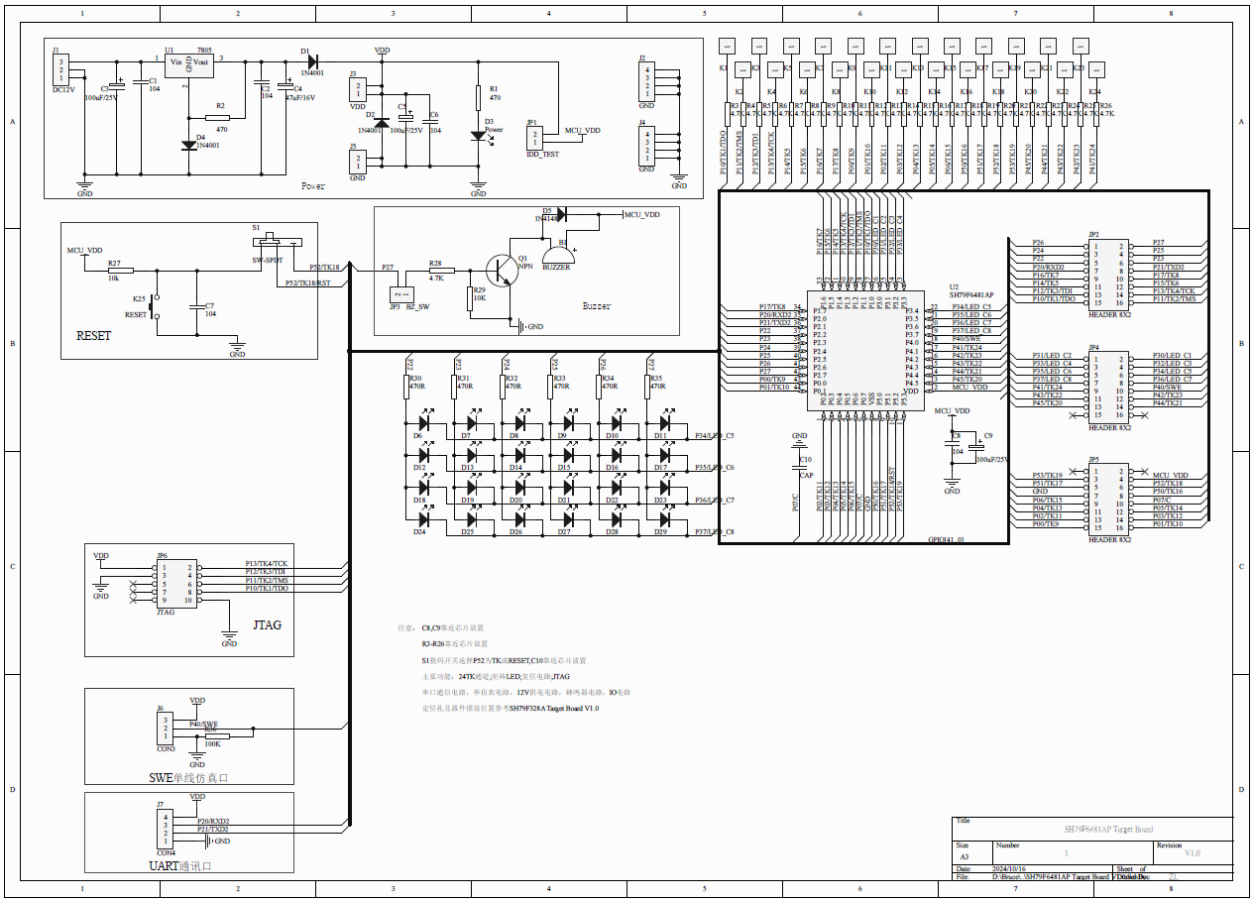
如果 IFCOUNT = 1，按键扫描计数溢出错误，清 IFCOUNT 和标志位，减小电容 C1。返回步骤 5，重新启动扫描。

(9) 一组按键扫描完成。

注：*1. 进出中断时需对 INSCON 寄存器进行压栈出栈操作。*

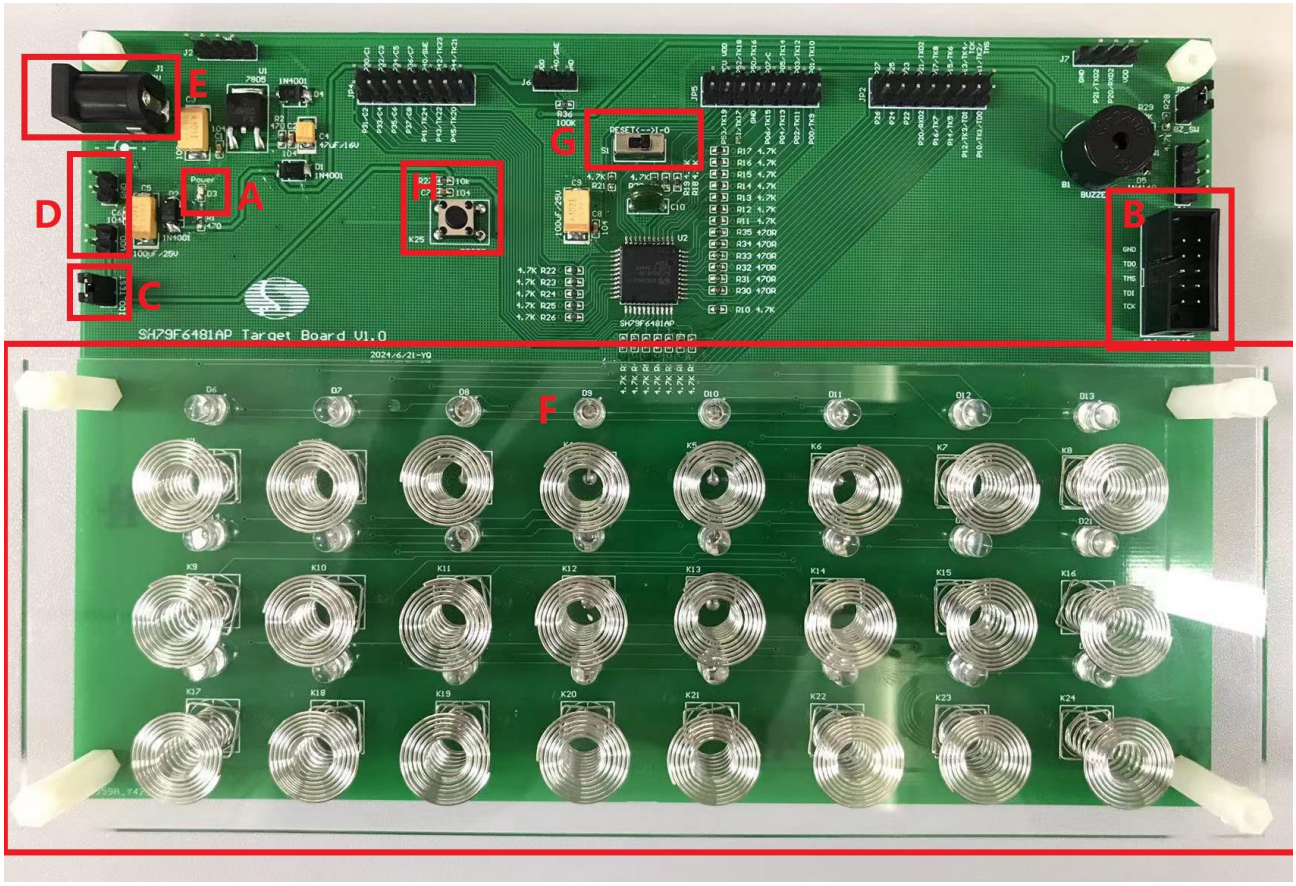
具体应用请参考 SH79F6481A Touch 库。

附件一：SH79F6481A Target Board V1.0 原理图



附件二：SH79F6481A 演示板（Target Board）

SH79F6481A 提供了一套演示板，供客户熟悉芯片功能。如下图：



板中各部分功能说明：

A. 电源指示灯

B. JTAG 仿真接口

JTAG 仿真接口和 JET51 仿真器一一对应，可用 10 芯扁平线直接连接。

C. JTAG VDD 和 MCU VDD 的接口

下载程序选择 JTAG 供电时，必须用跳线将该两个 PIN 脚短接起来，此时整个电源系统是联通的。

D. 外部供电接口 1（VDD/GND）

Target Board 板供电电源接口，分别与芯片的 VDD 和 GND 管脚相连，采用外部供电接口 1 方式供电，芯片供电电压可调。

E. 外部供电接口 2

Target Board 板供电电源接口，经过板上的 U1 电路，提供稳定的 5V 电压。

F. Touch Key 按键

G. 拨码开关（RST PIN）

P5.2 作为 PIN RST 时，拨码开关拨左和板子上的 I 部分组成 RST 电路，I 部分的按钮被按下一次就会发生一次 RESET，P5.2 作为 IO 时，拨码开关拨右，和外部的 RST 电路断开。

H. 复位按键

规格更改记录

版本	记录	日期
1.0	初始版本	2024 年 10 月

目录