第4章　系统设置的 C 语言程序设计

4.1 硬件结构

这一章介绍的硬件主要是与系统时钟、PLL、LVD、LVR、Watch_Dog 相关的部分。

4.1.1 系统时钟

unSP 时钟电路采用晶体振荡器电路，外接晶振采用 32768Hz。图 4.1 为 SPCE061A 时钟电路的接线图。

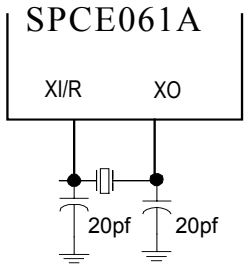


图4.1 SPCE061A 与振荡器的连接

32768Hz 实时时钟通常用于钟表、实时时钟延时以及其它与时间相关类产品。SPCE061A 通过对 32768Hz 实时时钟源分频而提供了多种实时时钟中断源。例如，用作唤醒源的中断源 IRQ5_2Hz，表示系统每隔 0.5 秒被唤醒一次，由此可作为精确的计时基准。

32768Hz 的实时时钟经过 PLL 倍频电路产生系统时钟频率(Fosc)，Fosc 再经过分频得到 CPU 时钟频率(CPUCLK)可通过对 P_SystemClock(写)($7013H)单元编程来控制。默认的 Fosc、CPUCLK 分别为 24.576MHz 和 Fosc/8。用户可以通过对 P_SystemClock 单元编程完成对系统时钟和 CPU 时钟频率的定义。

此外，32768Hz RTC 振荡器有两种工作方式：强振模式和自动弱振模式。处于强振模式时，RTC 振荡器始终运行在高耗能的状态下。处于自动弱振模式时，系统在上电复位后的前 7.5s 内处于强振模式，然后自动切换到弱振模式以降低功耗。CPU 被唤醒后默认的时钟频率为 Fosc/8，用户可以根据需要调整该值。CPU 被唤醒后经过 32 个时钟周期的缓冲时间后再进行其它的操作，这样可以避免在系统被唤醒后造成 ROM 读取错误。

4.1.2 锁相环 PLL (Phase Lock Loop)振荡器

PLL 电路的作用是将系统提供的实时时钟的基频(32768Hz)进行倍频，调整至 49.152MHz、40.96MHz、32.768MHz、24.576MHz 或 20.480MHz。系统默认的 PLL 自激振荡频率为 24.576MHz。PLL 的电路框图如图 4.2 所示。我们可以从图中看出，Fosc 是由 P_SystemClock 单元的第 5、6、7 位设定，CPUCLK 由第 0、1、2 位设定。

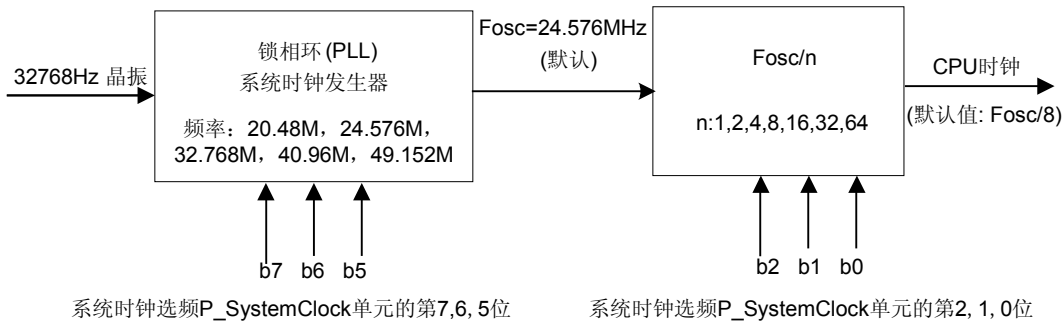


图4.2　锁相环电路框图

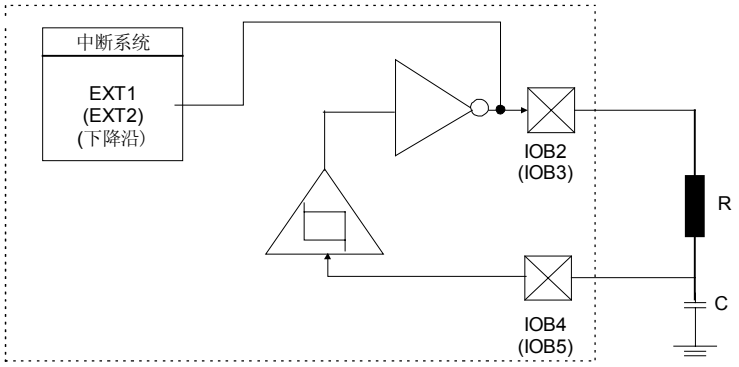
4.1.3 利用 B 口的特殊功能产生振荡信号

我们在上一章提到，B 口既可以做普通 IO 口，又可以做特殊功能口。比如，单个 IOB2 或 IOB3 口可设置为外部中断的输入口。

本小节将讲述如何为外部电路提供振荡信号。

图 4.3 为 IOB2，IOB3，IOB4 及 IOB5 的反馈结构示意图。通过在 IOB2 (IOB3)和 IOB4 (IOB5)之间增加一个 RC 电路形成反馈回路，即可在 IOB2(IOB3)端得到振荡源频率信号。此信号可作为外部中断源输入 EXT1 或 EXT2，此时所得到的中断频率与 RC 振荡器的频率是一致的。当外部电路需要用到一定频率的振荡信号时，也可以在 IOB2(IOB3)端获得。

为使反馈回路正常工作，必须将 IOB2 (IOB3) 设置成反相输出口，且将 IOB4 (IOB5)设置成悬浮式输入口。同时，必须设置 P_FeedBack 单元。



当P_FeedBack(写)($7009H)单元的FBKEN2(FBKEN3)位被置为"1"时
IOB2、IOB4或IOB3、IOB5之间的反馈结构

图4.3 反馈结构示意图

4.1.4 低电压监测/低电压复位 (LVD/LVR)

SPCE061A 具有低电压复位功能，并可通过编程设置低电压监测，目的是为了通过对系统的电源电压进行监控，而使系统运行在一个正常、可靠的工作环境，并在一旦出现电源异常的情况下能立即采取相应的措施，使系统及时恢复正常。

低电压监测功能可以提供系统内电源电压的使用情况。如果系统电压 Vcc 低于用户设定的电压监测低限电压 V_{LVD}，P_LVD_Ctrl 单元的第 15 位(LVD 监测标志位)将被置为“1”；反之，当 Vcc > V_{LVD} 时，该位被置为“0”。SPCE061A 具有 4 级电压监测低限，系统默认

的电压监测低限为 2.4V。

SPCE061A 复位电路如图 4.4 所示，在 RESB 端加上一个低电平就可令其复位。该电路具有手动和上电复位两种功能。

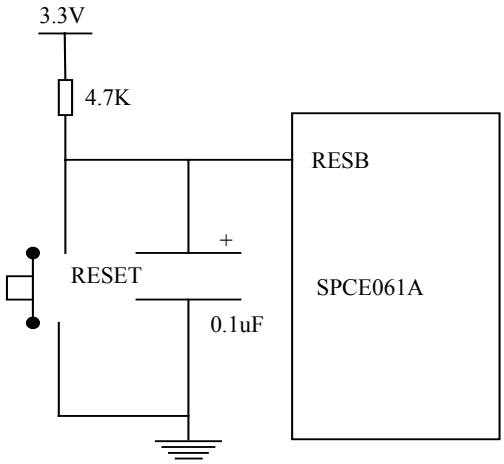


图4.4 复位电路

有很多原因可以导致电源电压过低，如电压的反跳、负载过重、电池能量不足等等。当电源电压低于 2.2V 时，系统会变得不稳定且易出故障。这时，CPU 会在 4 个时钟周期之后产生一个复位信号，使系统复位。

4.1.5 看门狗计数器（WatchDog）

WatchDog 的清除周期应该小于 0.75s。

4.2 系统设置的寄存器

P_SystemClock

在 SPCE061A 内，P_SystemClock(写)($7013H)单元（如表 4.1 所示）控制着系统时钟和 CPU 时钟。第 0~2 位用来改变 CPUCLK，若将第 0~2 位置为“111”可以使 CPU 时钟停止工作，系统切换至低功耗的睡眠状态；通过设置该单元的第 5~7 位可以改变系统时钟的频率(如表 4.3 所示)。此外，在睡眠状态下，通过设置该单元的第 4 位可以接通或关闭 32768Hz 实时时钟。

表4.1 设置 P_SystemClock 单元

b15-b8	b7~b5	b4[1]	b3	b2	b1	b0
---	PLL 频率选择	32768Hz 睡眠状态	32768Hz 方式选择	CPU 时钟选择		
		1：在睡眠状态下，32768Hz 时钟仍处于工作状态(默认) 0：在睡眠状态下，32768H 时钟被关闭	1：32768Hz 时钟处强振模式 0：32768Hz 时钟处自动弱振模式(默认)			

表4.2 CPU 时钟频率 (CPUCLK) 选择

b2	b1	b0	CPUCLK
0	0	0	Fosc

0	0	1	Fosc/2
0	1	0	Fosc/4
0	1	1	Fosc/8[2]
1	0	0	Fosc/16
1	0	1	Fosc/32
1	1	0	Fosc/64
1	1	1	停止(睡眠状态)

表4.3　PLL 频率 (Fosc) 选择

b7	b6	b5	Fosc
0	0	0	24.576MHz
0	0	1	20.48MHz
0	1	0	32.768MHz
0	1	1	40.96MHz
1	-	-	49.152MHz

注：

[1]只有当 b 0~b2 同时被置为“1”时(即睡眠状态)b4 设置才有效。

[2]上电复位或系统从备用状态(睡眠状态)被唤醒后，默认的 CPU 时钟频率为 Fosc/8。

P_FeedBack

B 口工作方式的控制单元，用于控制 B 口的 IOB2 (IOB3)和 IOB4 (IOB5)用作普通 I/O 口，或作为特殊功能口。其特殊功能包括以下两个部分：（1）.单个 IOB2 或 IOB3 口可设置为外部中断的输入口。（2）.设置 P_FeedBack 单元， 再将 IOB2(IOB3)和 IOB4(IOB5)之间连接一个电阻和电容(电路连接如图4.3)形成反馈电路以产生振荡信号,此信号可作为外部中断源输入EXT1或 EXT2。当然此时所得到的中断频率与 RC 振荡器的频率是一致的。由于该频率较高，所以通常情况下都是通过(1)获得外部中断信号。此特殊功能仅运用于：当外部电路需要用到一定频率的振荡信号时，可以在 IOB2(IOB3)端获得。 P_FeedBack 的设置的如表 4.4 所示。

表4.4 P_FeedBack 的设置

b15 – b4	b3	b2	b1	b0
---	FBKEN3	FBKEN2	---	---
	1：设定 IOB3 和 IOB5 之间形成反馈功能 0：IOB3、IOB5 作为普通的 I/O 口(默认)	1：设定 IOB2 和 IOB4 之间形成反馈功能 0：IOB2、IOB4 作为普通的 I/O 口(默认)		

图 4.3 为 IOB2，IOB3，IOB4 及 IOB5 的反馈结构示意图。通过在 IOB2 (IOB3)和 IOB4 (IOB5)之间增加一个 RC 电路形成反馈回路，即可在 IOB2(IOB3)端得到振荡源频率信号。为使反馈回路正常工作，必须将 IOB2 (IOB3) 设置成反相输出口，且将 IOB4 (IOB5)设置成悬浮式输入口。

P_LVD_Ctrl

SPCE061A 具有 4 级电压监测低限：2.4V、2.8V、3.2V 和 3.6V，可通过对 P_LVD_Ctrl 单元编程进行控制，参见表 4.5。假定 V_{LVD}=3.2V，当系统电压 Vcc 低于 3.2V 时，P_LVD_Ctrl 单元的第 15 位返回值为“1”，这样，CPU 可以通过可编程电压监测低限来完成低电压监测。系统默认的电压监测低限为 2.4V。

表4.5 P_LVD_Ctrl(读/写)($7019H)

b15	b14 - b2	b1	b0
Result_of_LVD	---	LVD_level_define(写)	
0：$V_{DD} > V_{LVD}$ 1： $V_{DD} < V_{LVD}$			

b1	b0	电压监测低限 (V_{LVD})
0	0	2.4V* (默认)
0	1	2.8V
1	0	3.2V
1	1	3.6V

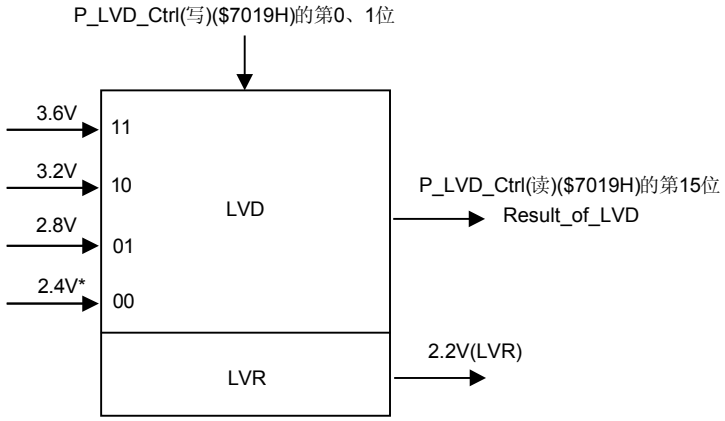


图4.6 低电压监测/低电压复位

P_Watchdog_Clear

表 4.6 列出了 WatchDog 配置单元 P_WatchDog_Clear 及其内 B0 对 WatchDog 清除的控制。清除 WatchDog 只需写入 P_WatchDog_Clear 单元‘0x0001’即可。此外，若 32768Hz 振荡器被打开，则在空闲方式期间 WatchDog 功能是被激活的。

表4.6　WatchDog 的配置及 WatchDog 的清除

配置单元	读写属性	存储地址	配置单元功能说明
P_WatchDog_Clear	写	7012H	清除 WatchDog 单元
B15~B1	B0	控制位功能解释	
――――	WatchDog_Clear		
0～0	0	不清除 WatchDog	
0～0	1	清除 WatchDog	

4.3 系统设置的 C 函数

SPCE061.lib 中提供了相应的 API 函数如下所示：

函数原型

void Watchdog_Clear(void);

功能说明　清看门狗，Watchdog Reset Timer = 0.75 second

用法　　Watchdog_Clear()；
参数　　无

函数原型

void Set_SystemClock(unsigned int);
功能说明　　设置 System 信息
用法　　Set_SystemClock(System_Information);
参数　　System Information = CPU Clock + 32K Mode + 32K Sleep Status + OSC Frequency

CPU Clock 的取值可以是以下几种：

C_Fosc	→ Fosc
C_Fosc_Div_2	→ Fosc/2
C_Fosc_Div_4	→ Fosc/4
C_Fosc_Div_8	→ Fosc/8 (默认值)
C_Fosc_Div_16	→ Fosc/16
C_Fosc_Div_32	→ Fosc/32
C_Fosc_Div_64	→ Fosc/64
C_Sleep	→ Stop (睡眠)

32K Hz Mode 的取值可以是以下几种：

C_StrongMode	→强振模式
C_AutoMode	→弱振模式　(默认值)

32K Hz Sleep Status 的取值可以是以下几种：

C_32K_Work	→ 睡眠时, 32K 时钟仍然工作　(默认值)
C_32K_Off	→ 睡眠时, 32K 时钟停止工作

OSC Frequency 的取值可以是以下几种：

C_Fosc_24M	→ 24.576M (默认值)
C_Fosc_20M	→ 20.480M
C_Fosc_32M	→ 32.768M
C_Fosc_40M	→ 40.960M
C_Fosc_49M	→ 49.152M

函数原型

void Set_LVD_Ctrl(unsigned int);
功能说明　　设置电压监测低限
用法　　Set_LVD_Ctrl(Voltage_Level)；
参数　　Voltage Level

Voltage Level 的取值可以是以下几种：

C_LVD24V	→	LVD = 2.4V　(默认值)
C_LVD28V	→	LVD = 2.8V
C_LVD32V	→	LVD = 3.2V

函数原型

unsigned int Get_LVD_Ctrl(void)；
功能说明　　获取电压是否到达监测低限的信息
用法　　LVD_Result = Get_LVD_Ctrl();

返回值　返回值的最高位 (b15)为 0 代表 $V_{DD} > V_{LVD}$，为 1 代表 $V_{DD} < V_{LVD}$。

函数原型

void Set_FeedBack(unsigned int);

功能说明　设置 FeedBack 信息

用法　Set_FeedBack(FeedBack_Data)

参数　FeedBack_Data 包含了以下 3 项信息：

C_FBKEN3　→IOB3 和 IOB5 间形成反馈

C_FBKEN2　→IOB2 和 IOB4 间形成反馈

C_IRTxEN　→IOB8 和 IOB10 IRTx / Tx 使能

C_FBKEN3、C_FBKEN2、C_IRTxEN 是任选项，使用者可以根据需要来选择其中一项或多项。如：

FeedBack Data　= C_FBKEN3 + C_FBKEN2 + C_IRTxEN

FeedBack Data　= C_FBKEN3 + C_IRTxEN

FeedBack Data　= C_IRTxEN

FeedBack Data　= 0

4.4 系统设置的应用实例

例 4.1

本例主要是观察 Watchdog 的作用结果。

上文讲过，清 WatchDog 的周期要小于 0.75s，不然，系统会被重启。硬件连接图如图 3.2 所示。我们来看这个程序，当把 Delay 设为 count1 时，连接在 A 口低八位的 LED 亮一下就灭了；当把 Delay 设为 count0 时，连接在 A 口低八位的 LED 会不停地闪。为什么会这样呢？按照程序的流程，程序把 A 口的 LED 灭了之后应该一直在 while（1）中死循环，不会再点亮 LED。现在，把 Delay 设为 count0 时，连接在 A 口低八位的 LED 会不停地闪，证明系统重新启动了。这就是 WatchDog 作用的结果。

以下为本例的代码。

```
#include "SPCE061.h"
#define count0 0xffffff;
#define count1 0xfff;
int main()
{
    unsigned long int    Delay;
                                        //设置 B 口为同相低电平输出
    *P_IOA_Dir =0xffff;
    *P_IOA_Attrib =0xffff;
    *P_IOA_Data =0;                     //点亮 LED

    Delay =5000;
    while(Delay --);
```

```
        *P_IOA_Data =0xffff;              //熄灭 LED
        while(1)                          //死循环
        {
            *P_Watchdog_Clear=1;
            Delay =count0;
            while(Delay --);
        }
    }
```

例 4.2

这个例子是进行系统时钟设置。读者可以选择不同的 Fosc_CLK 和 CPU_CLK，观察发光二极管亮灭快慢。

说明：

1、 硬件连接图如图 3.2 所示。

2、 读者只需改变预编译处理的两行

```
#define Fosc_CLK_RATE        C_Fosc_49M;          //select Fosc
#define CPU_CLK_RATE         C_Fosc_Div_2;        //select CPUClk
```

就可以改变 Fosc_CLK 和 CPU_CLK。其中 Fosc_CLK 和 CPU_CLK 的取值范围在 4.3 小节中有介绍。

以下为本例的代码。

```
#include "SPCE061.H"
void Delay();
#define Fosc_CLK_RATE        C_Fosc_49M;          //select Fosc
#define CPU_CLK_RATE         C_Fosc_Div_2;        //select CPUClk
main()
{
    unsigned int Fosc_CLK,CPU_CLK;
    *P_IOA_Dir = 0x00FF;                          // IOA: [7..0] output
    *P_IOA_Attrib = 0x00FF;
    *P_IOA_Data = 0;
    Fosc_CLK = Fosc_CLK_RATE;
    CPU_CLK = CPU_CLK_RATE;
    *P_SystemClock = Fosc_CLK|CPU_CLK;

    while(1)
    {
        *P_IOA_Data = 0x00FF;                     //LED off
        Delay();
        *P_IOA_Data = 0;                          //LED on
        Delay();
    }
}
```

```
void Delay(void)
{
    unsigned int j;
    for(j=0x7fff;j>0;j--);

    *P_Watchdog_Clear=1;
}
```