**实验一**

**1. 人为修改内部时钟或外部时钟，如将内部时钟改为8M，或将外部时钟改为30M，会有什么结果？**

答：内部时钟或外部时钟的晶振频率硬件固定，不能人为修改。

**2. 在使用PIOSC及MOSC时，能否生成非晶振频率？如1M、10M 或32M？**

答：可以对PIOSC（16M）及MOSC（25M）进行 1~1024（实际为1~64，参见sysctl.h）整数分频，但不能倍频。PIOSC 16分频可以得到 1M，但无法用16M或25M 整数分频到10M 或32M。

**3. 能否将PLL时钟调整到外部时钟的频率以下？如将25M外部时钟用PLL后调整为20M？**

答：利用PLL可以对外部时钟进行倍频（倍频到320M或480M）和分频，分频系数 1~1024（实际为1, 1.5, 2, 2.5 ~ 64，参见sysctl.h）。所以对320M进行16分频或者对480M进行24分频都可以得到20M。

**4. 在使用PLL时，系统频率最小值及最大值分别为多少？**

答：利用PLL可以倍频到320M或480M，经过分频，理论上最小值312.5K（1024分频），实际最小值为5M（64分频），最大值受处理器最高频率限制为120M。

**5. 将PLL后的时钟调整为最大值120M，LED闪烁会有什么变化？为什么？**

答：若用Delay函数作软件延时，则延时时长与主频有关，LED闪烁会变快。若用定时器硬件作延时，则延时时长决定于定时时间，与时钟频率无关，LED闪烁无变化。

**6. GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_PIN\_0)此函数中，每个函数项的意义。第三个函数项为GPIO\_PIN\_0，如果改为1或改为2，或其他值，分别有什么现象？**

答：第一项是GPIO F口的基地址，第二项是屏蔽位，第三项是要写入的值，只有未屏蔽的位才能写入。GPIO\_PIN\_0表示的是常数0x00000001，所以该语句的作用是，屏蔽除bit0（即PF0）以外的所有引脚，往PF0引脚写数值1。若第三项改为1，与GPIO\_PIN\_0的值相同，也是将PF0置1；若改为2，则将PF0置0。

**7. 结合硬件说明GPIOPinConfigure行的作用。如果此行注释，在WATCH窗口中观察key\_value值会有什么变化。**

答：从原理图上看PJ0是开漏的，GPIOPinConfigure行的作用是给PJ0加弱上拉。如果没有此行，则只能读到0，读不到1。

**实验二参考答案**

本程序要求掌握I2C总线扩展GPIO芯片PCA9557及TCA6424的使用方法

ans2-1

让数码管同时显示多个字符的要点：

1）先往port2写0（防拖影），再给码管值和位选

2）循环刷新 + 控制延时（防频闪）：适当增加一点延时可以使码管亮度增加些，但延时不能太大，同时显示8个字符约2ms左右，同时显示2个字符不超过10ms

例如：

volatile uint8\_t cnt=0;

int v[]={2,0,1,9,0,6,0,1}; //同时显示20190601

.....

while(1) {

I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_OUTPUT\_PORT2, 0); //清0

I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_OUTPUT\_PORT1,seg7[v[cnt]]);

I2C0\_WriteByte(TCA6424\_I2CADDR,TCA6424\_OUTPUT\_PORT2, 0x1<<cnt);

SysCtlDelay(2\*ui32SysClock/3000); //延时2ms

cnt = (cnt+1) % 8;

}

ans2-2

将例程中LED走马灯处的 1 << cnt 改为 3<<cnt

ans2-3

按照ans2-1的方法将两位数码管显示放在10ms定时分枝，LED走马灯和 cnt 更新放在500ms分枝

ans2-4

在ans2-3的基础上，启用PJ0中断，下降沿触发，记录按键次数 btn\_cnt。用数组预设定时间隔，把btn\_cnt作为下标，以替代原500ms定时的符号常数。具体：

1）定义全局变量 uint8\_t btn\_cnt; 记录按键次数，在GPIO中断中更新。（参考实验1 ans1-5.c对应的程序）

2) 定义数组 uint32\_t I2C\_FLASHTIME[] = {500, 1000, 2000, 200}; //500ms,1s,2s,200ms

用 I2C\_FLASHTIME[btn\_cnt] 替代程序中的符号常数 I2C\_FLASHTIME。

ans2-5

参见asn2-5.c

**实验三**

**1. 实验3-2，if (UARTCharsAvail(UART0\_BASE))此行程序的作用。如果没有此行，会导致什么问题？**

答：作用是检测RxFIFO中是否有接收到的字符。由于例程采用的是阻塞式接收方式，所以如果没有此行，当PC机的超级终端没有发送数据过来时，程序会一直停在UARTCharGet()处，下面的走马灯、LED等都不能正常运行。

**2. 实验3-3，void UART0\_Handler(void)为什么没有在主函数声明？**

答：UART0\_Handler已在初始化程序中注册，是由系统直接调用，而非主函数调用，所以不需要在主函数声明。

**3. 为什么3-3的中断中需要读取中断标志并清除，而SYSTICK不需要？**

答：UART是多源中断，硬件不能确定程序当前会处理哪个中断源引起的中断，所以没有自动清除，因此需要软件完成读取中断标志并清除的任务。Systick是单源中断，它的中断标志由硬件自动清除。

**4&5. 要点**

（1）定义全局字符数组用于存放接收到的字符串。如：char uart\_receive\_char[30];

（2）在中断处理程序（或主循环中）接收字符串，（接收程序参见ARM实验3PPT）。

注意：在主循环中接收数据也要考虑加延时或识别结束标记，延时方法如：

uint8\_t k;

...

while（1）//主循环

{

k = 0;

while(UARTCharsAvail(UART0\_BASE)) // Loop while there are characters in the receive FIFO.

{

uart\_receive\_char[k++] = UARTCharGetNonBlocking(UART0\_BASE);

SysCtlDelay(ui32SysClock/30000); //延时

}

uart\_receive\_char[k] = '\0';

...//按题意要求作字符串处理

}

（3）对接收到的字符串按题意要求处理，可使用字符串处理函数，推荐使用 sscanf 分割字符串。

如：从格式为: MON+dd 的字符串 uart\_receive\_char 中提取“MON”、‘+’ 和dd

char month[4], op;

int n;

sscanf(uart\_receive\_char, "%3s%c%2d", month, &op, &n); //则month的值为“MON”，op=‘+’，n=dd

同样可以用 sprintf 将不同类型的数据组织成一个字符串，再从串口输出。（注意：UART的数据收发都必须是字符串）

如：把month, op, n三个变量组成字符串 today=MON+dd

char buff[30];

sprintf(buff, "today=%3s%c%2d\r\n", month, op, n); //则buff的值为“today=MON+dd\r\n”（加\r\n是为了显示后换行）

UARTStringPut(buff);