实验记录.

(一)嵌入式处理器UART通讯

1. 设计完成从PC机键盘将字符输入到ARM上并通过串口输出.

(1) UART模块初始化设置.

```
void SerialInit(void){
    GPDR1  |=  0x80 ;   //设置34号引脚为输入，39号引脚为输出.
    GAFR1_L |= 0x8010 ; //34号复用输入功能1(FFRXD)，39号复用输出功能2(FFTXD).
    FFLCR  = 0x3; //通信控制8位字长，1位停止.
    FFFCR  = 0x0;
    FFIER  = 0x40;
    //设置波特率.
    FFLCR  = 0x80; //选择访问FFDLL和FFDLH.
    FFDLL  = 0x8;
    FFDLH  = 0x0; //当Divisor设置为8时，得到波特率为115200.
                  //小组任务将超级终端波特率改为4800，则将FFDLL改为0xC0;
    FFLCR  &= 0x7F;   //设置访问FFRBR, FFTHR, FFIER.
    while(!(FFLSR & 0x40));
        //当THR寄存器为空时，初始化结束，开始实验.
    return;
}
```

(2) 设置字节传入与传出.

```
void SerialOutputByte(const char c){
    while((FFLSR & 0x20)==0); //当THR寄存器有数据时，才运行后续操作.
    FFTHR = ((ulong)c & 0xFF); //处理数据放入缓冲区中.
    if(c=='\n') SerialOutputByte('\r');
}.
void SerialInputByte(char *c){
    if((FFLSR & 0x1)==0) return 0;   //若无数据则返回.
    else{
        *c = FFRBR; return 1;} //否则接受输入缓冲区数据.
```

(3) 设置将输入字符通过串口输出.

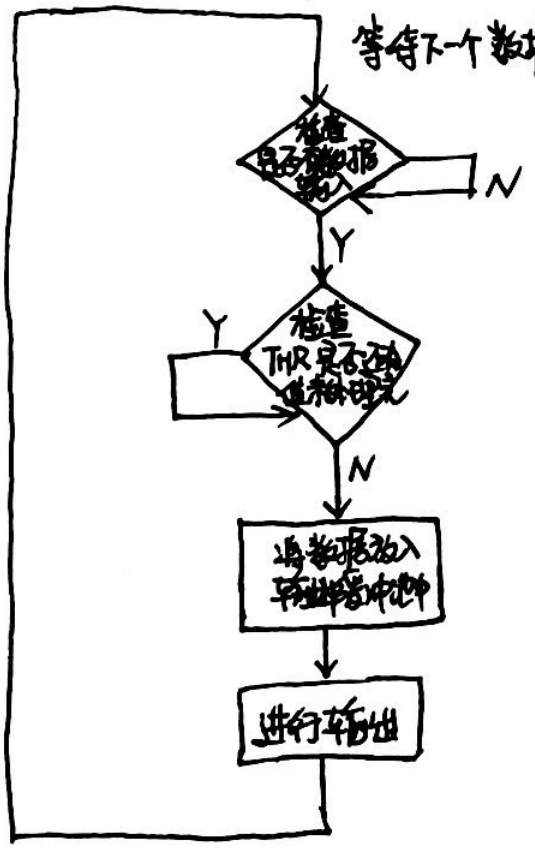
```
int main( ) {
    char cc;
    SerialInit( );
    while(1) {
        if (SerialInputByte(&cc))
            SerialOutputByte(cc);      //若存在输入数据,将其输出.
    }
    return 0;
}
```

2. 如果超级终端波特率改为"4800",要想PC与ARM平台还正常通信,如何修改程序?

由 $BaudRate = \frac{14.7456MHz}{16 \times Divisor}$, 将 BaudRate = 4800 代入得:

Divisor = 192 = 0xC0. 故将 DLL = 0xC0, DLH = 0x0 即可.

3. 使用LSR方法中的程序执行框架.



(二)嵌入式处理器的中断控制器

1. 键盘作为中断源，编写中断处理程序完成功能：键盘上5、8对应led灯，对应数码管，D余数，实现8%5的结果。

(1)编写中断初始化程序.

```
mrs     r1 , CPSR
bic     r0l , r1, #0x80
msr     CPSR_c , r1
//将状态寄存器 IRQ控制位置0清空，其余位不变，使ARM可以响应中断.
ldr     r1 , =ICMR
ldr     r2 , =init_ICMR.  0 0 : 此将4位置1，响应key请求.
str     r2 , [r1]
//配置屏蔽寄存器ICMP，设置允许键盘中断源请求.
```

(2)编写 boot.s 中的异常向量表，将预设的IRQ_Fuction的框架注释掉.

(3)编写中断服务子程序.

```
__irq    void    IRQ_Handler (void)
{
    char    key_value;
    key_value   = kPAS_VALUE;
    switch (key_value) {
        case    0x20:                   //key-press 8
            LED_Addr = (0xFF<<7)-1;     //让第8个灯亮.
            LED_CS3 = 0x00FF;           //数码管右边第一个显示数字8
        case    0x10:                   //key-press 5
            LED_Addr = (0xFF<<4)-1;     //第5个灯亮.
            LED_CS3 = 0x10FF;           //数码管显示数字5.
        case    0x56:                   //key_press D  (8%5)
            LED_Addr = (0xFF<<3)-1;     //第3个灯亮.
            LED-CS3 = 0x30FF;           //显示数字3.
```