3. 实验[3] UART 串行通讯实验

3.1 实验目的

了解 UART 串行通讯的工作原理；
掌握在 PC 端通过串口调试工具与实验板通过 UART 通讯的方法；
掌握 UART 的堵塞式与非堵塞式通讯方法。

3.2 实验主要器材和设备

电脑，TM4C1294NCPDT 实验板卡，A2000TM4 扩展板,串口调试软件，数字示波器。

3.3 关于例程 exp3_0.c 的说明

例程 exp3_0.c 通过 A2000TM4 底板按键阵列控制 UART 端口的数据读写，并进行字符的大小写转换。实现方法采用阻塞式，即在发送或接收数据过程中，始终查询状态，占用 CPU 时间，只有发送或接收结束后才退出。开机或复位后，底板上左边第 2 个数码管显示 0，LED8 点亮，系统工作在默认状态（状态 0）。系统启动后，会处于以下 4 种状态：

1)状态 0：默认状态或按下编号为 4-9 按键后进入。此时 PC 端发送的字符，实验板收到后原样返回。底板上左边第 2 个数码管显示 0，LED8 点亮。

2)状态 1：按下编号为 1 的按键后进入。此时 PC 端如果发送的是小写字符'a'-'z',实验板收到后转换为大写'A'-'Z'后返回，其他字符则原样返回。底板上左边第 2 个数码管显示 1，LED7 点亮。

3)状态 2：按下编号为 2 的按键后进入。此时 PC 端如果发送的是大写字符'A'-'Z',实验板收到后转换为小写'a'-'z'后返回，其他字符则原样返回。底板上左边第 2 个数码管显示 2，LED6 点亮。

4)状态 3：按下编号为 3 的按键后进入。此时 PC 端如果发送的是大写字符'A'-'Z',实验板收到后转换为小写'a'-'z'后返回；如果发送的是小写字符'a'-'z',实验板收到后转换为大写'A'-'Z'后返回；其他字符则原样返回。底板上左边第 2 个数码管显示 3，LED5 点亮。

图 3-1 是 exp3_0.c 程序流程示意图。

在例程 exp3_0.c 中所需定义的常量、变量（及初值）、函数全部列出，形成表 3-1 至 3-3 的表格。

表 3-1 exp3_0.c 示例程序的常量

常量名	含义	值
SYSTICK_FREQUENCY	SysTick 频率为 50Hz	50

表 3-2 exp3_0.c 示例程序的变量

变量名	类型	含义
digit[8]	uint8_t	数组变量 digit[]用于缓存数码管的显示内容，数组长度为 8，对应板上数码位从左到右序号排列为 4、5、6、7、0、1、2、3；比如，digit[4]对应最左一个数码管，digit[2]对应左数第七个数码管。
pnt	uint8_t	8 位变量 pnt 用于缓存数码管小数点的亮灭状态，0 灭 1 亮。板上数码小数点从左到右对应 pnt 的第 4、5、6、7、0、1、2、3 位；初值 0x04 会使左数第七个数码小数点被点亮。

led[8]	uint8_t	数组变量 led[]用于缓存 LED 指示灯的状态值，0 灭，1 亮。板上指示灯从左到右对应的下标序号为 7、6、5、4、3、2、1、0。初值{1，1，1，1，1，1，1，0}会使八个指示灯从左到右显示为“亮亮亮亮亮亮亮灭”。
key_code	uint8_t	变量 key_code 用于缓存最近一次读到的键值。3x3 按键阵列从左上向右下编号 1-9。比如第二行第三个按键按下，键值为 6；无键按下，则键值用 0 表示。
key_state	uint8_t	用于记录前一次按键检测时的键盘状态，0 表示无键按下，1 有键按下
run_state	uint8_t	系统运行状态,取值 0-3

程序中的函数可以分布在两个文件中。在 exp3_0.c 中的除了主程序和 SYSTICK 中断程序外，主要是用来初始化微控制器的函数。其他与 TM1638 芯片有关的函数原型声明放在 tm1638.h 中，函数定义放在 tm1638.c 中。

表 3-3 exp3_0.c 示例程序的函数

函数名	功能	位置
int main(void)	主程序	exp3_0.c
void SysTick_Handler(void)	SYSTICK 中断服务程序	exp3_0.c
void GPIOInit(void)	I/O 口初始化	exp3_0.c
void SysTickInit(void)	SysTick 计时器初始化	exp3_0.c
void UARTInit(void)	UART 初始化	exp3_0.c
void DevicesInit(void)	初始化 I/O 口、SysTick 计时器等	exp3_0.c
void UARTStringPut(uint32_t ui32Base,const char *cMessage)	向 UART 模块发送字符串	exp3_0.c
uint8_t TM1638_DigiSegment (uint8_t digit)	数码管显示数据的译码	tm1638.c
void TM1638_Serial_input (uint8_t DATA)	TM1638 键盘数码管控制芯片的串行数据输入	tm1638.c
uint8_t TM1638_Serial_output(void)	TM1638 键盘数码管控制芯片的串行数据输出	tm1638.c
uint8_t TM1638_Readkeyboard (void)	读取 TM1638 获取按键状态	tm1638.c
void TM1638_RefreshDIGIandLED(uint8_t digit_buf[8], uint8_t pnt_buf, uint8_t led_buf[8])	刷新 8 位数码管（含小数点）和 8 组 LED 指示灯	tm1638.c
void TM1638_Init(void)	初始化 TM1638	tm1638.c

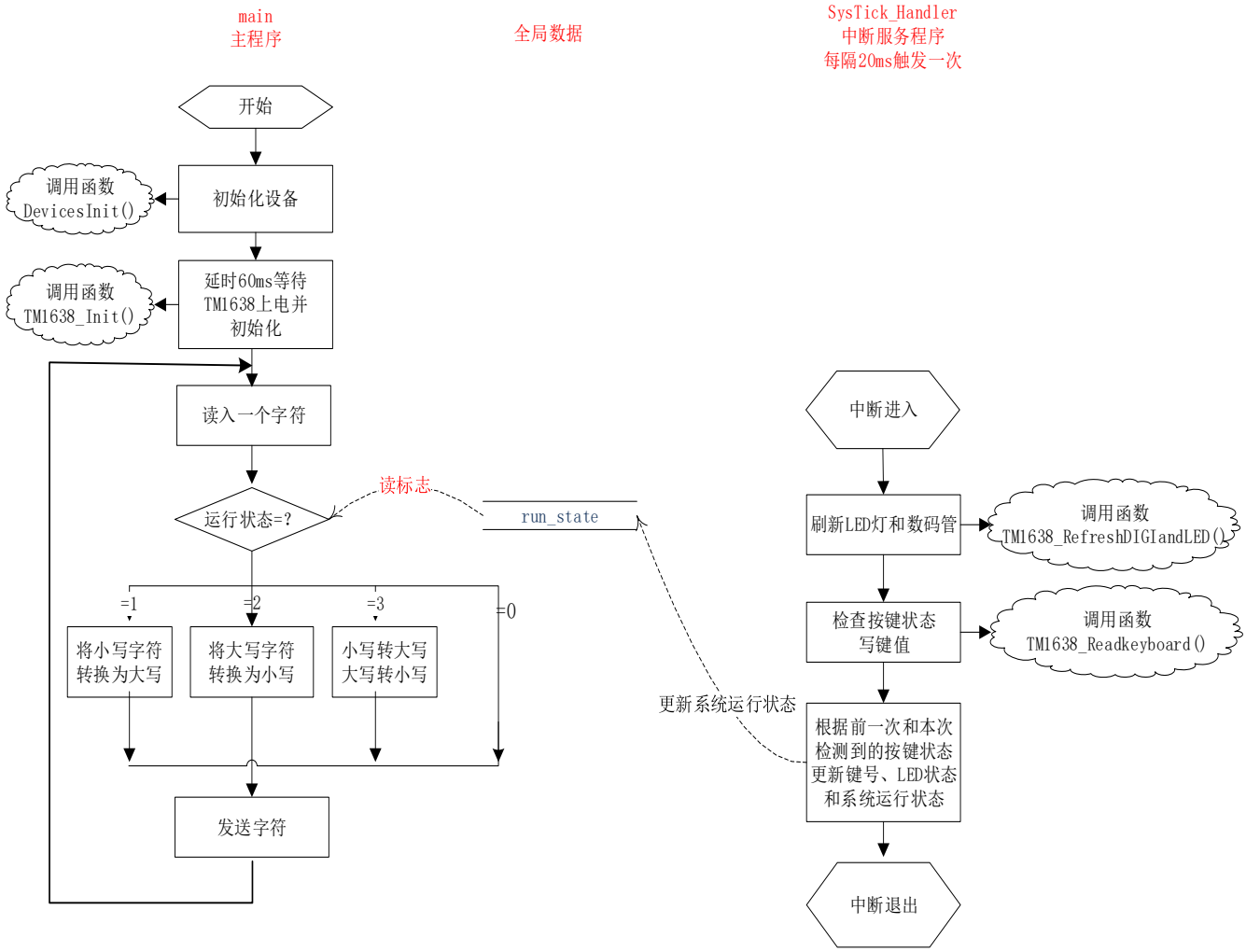

图 3-1 exp3_0.c 程序流程示意图

3.4 实验任务要求

3.4.1 实验任务 3_1

修改程序 exp3_0.c，同时另存命名为 exp3_1.c，完成实验任务 3_1。实验要求把例程 exp3_0.c 改写成非阻塞方式，即中断方式进行发送与接收。

图 3-2 是建议修改后的 exp3_1.c 程序流程示意图。

具体的实验步骤建议如下：

1) 观察 UART 初始化函数 void UARTInit(void)，了解 UART0 的波特率及帧格式是如何设置的；
修改 UART 初始化函数 void UARTInit(void)，在函数尾部添加下面两条语句：

```
IntEnable(INT_UART0);      // UART0 中断允许
UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);// 使能 UART0 RX,RT 中断
```

2) 增加 UART0 中断服务程序 void UART0_Handler(void)，内容如下：

```
void UART0_Handler(void)
```

```
{
    int32_t uart0_int_status;
    uint8_t uart_receive_char;

    uart0_int_status = UARTIntStatus(UART0_BASE, true); // 取中断状态
    UARTIntClear(UART0_BASE, uart0_int_status);         // 清中断标志

    while(UARTCharsAvail(UART0_BASE))           // 重复从接收 FIFO 读取字符
    {
      uart_receive_char = UARTCharGetNonBlocking(UART0_BASE); // 读入一个字符
      switch (run_state)
      {
        case 1:                              // 小写转大写
          if(uart_receive_char >= 'a' && uart_receive_char <= 'z')
          {
             uart_receive_char = uart_receive_char - 'a' + 'A';
          }
          break;
        case 2:                              // 大写转小写
          if(uart_receive_char >= 'A' && uart_receive_char <= 'Z')
          {
             uart_receive_char = uart_receive_char - 'A' + 'a';
          }
          break;
        case 3:                              // 大小写互换
          if(uart_receive_char >= 'a' && uart_receive_char <= 'z')
          {
             uart_receive_char = uart_receive_char - 'a' + 'A';
          }
          else
          {
             if(uart_receive_char >= 'A' && uart_receive_char <= 'Z')
             {
                uart_receive_char = uart_receive_char - 'A' + 'a';
             }
          }
           break;
         default:
           break;
      }
      // 发送转换好的字符
      UARTCharPutNonBlocking(UART0_BASE, uart_receive_char);

      if(uart_receive_char == '\r')  // 如果发现'\r'补发一个回车
```

```
            {
              UARTCharPutNonBlocking(UART0_BASE, '\n');
            }
          }
        }
```

3) 修改 main 函数，删除 while 循环的循环体。

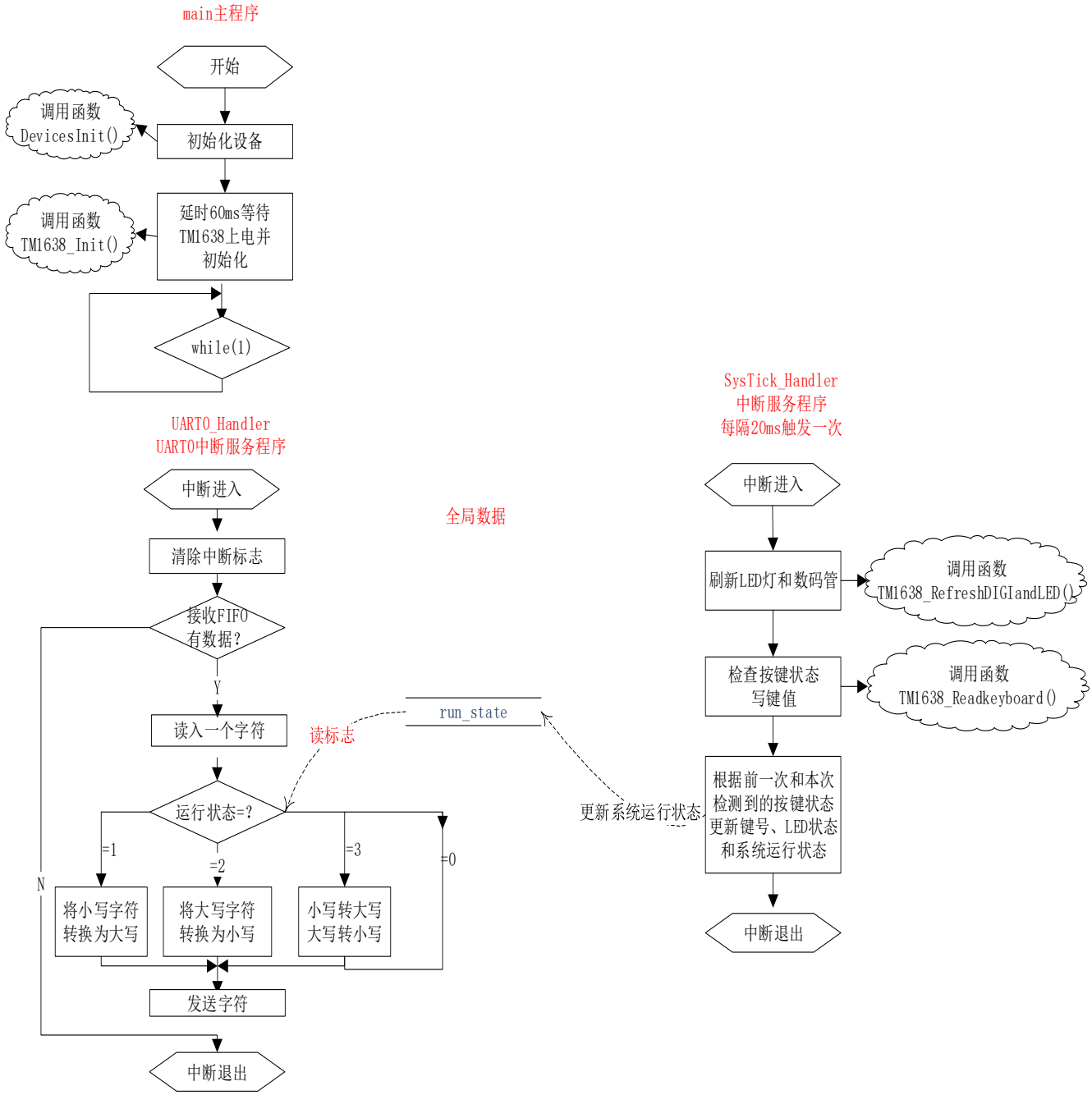

图 3-2 exp3_1.c 程序流程示意图

3.4.2 实验任务 3_2

请编写实现一个时钟（24 小时制）：

1) 开机或复位后，从左到右 6 位数码管显示 00.00.00，表示 00:00:00(零点零分零秒)。
2) 当 PC 端发送绝对对时命令，如 AT+SET15:04:34，自动将当前时间同步到 15:04:34,数码管显示为 15.04.34。
3) 当 PC 端发送相对对时命令，如 AT+INC00:00:34，自动将当前时间加 34 秒，并回之以当前时间。PC 端显示格式为 hh:mm:ss。要求检测时间格式的有效性。
4) 当 PC 端发送查询时间命令，如 AT+GET，自动回之以当前时间。PC 端显示格式为 hh:mm:ss。
5) 对 PC 端发送的命令要进行格式有效性的检测,要求 hh 取值 00-23；mm 取值 00-59；ss 取值 00-59。如果输入错误命令或格式不对的命令，微控制器回之以 Error Command!。

3.4.3 实验任务 3_3

请编写实现一个时钟（24 小时制）：

1) 开机或复位后，从左到右 6 位数码管显示 00.00.00，表示 00:00:00(零点零分零秒)。
2) MCU 每隔 1 秒钟向 PC 端发送“现在是中华人民共和国北京时间”和当前时间。如当前时间为 15:04:34，则 PC 端显示为“现在是中华人民共和国北京时间 15:04:34”。
3) 按照表 3-4 修改 UART 的波特率，观察 PC 端的显示结果是否正确，并分析原因。

表 3-4 不同的波特率对 UART 通信的影响

序号	UART 波特率（Baud）	显示结果是否正确	原因
1	230400		
2	115200		
3	9600		
4	1200		
5	300		
6	110		

3.5 实验结果的考评

课程组织现场考评，检查学习者对实验任务的完成情况。

- ✧ 对实验任务规定的功能，学习者应能熟练操作和展示；
- ✧ 学习者应能熟练掌握实验步骤中的具体操作和设计能力，根据评测官现场要求（工作参数或功能设计要求，可能略微有别于已做内容）展示这些技能。

课程还对实验报告进行评价。

3.6 实验报告要求

根据自己的实验结果记录，编写实验报告。课程提供有报告模板作为参考，其中内容允许学习者根据自己情况灵活调整。