嵌入式系统实验报告

实验名称：	UART 与 SHELL
姓　　名：	陈姝仪
学　　号：	2018211507
学 院(系)：	计算机学院
专　　业：	网络工程
指导教师：	刘健培

2020 年 12 月 31 日

1 实验目的

- 通过 FSM4 实验板了解实验的软硬件环境，熟悉 MDK 开发环境和 STM32CubeMx 开发工具的使用。
- 掌握基本的轮询式多软件编写与调试方式。
- 学会 STM32 USART 的基本操作方式。

2 实验环境

- FS-STM32F407 开发平台
- ST-Link 仿真器
- RealView MDK5.23 集成开发软件
- STM32CUBEMX 图形开发软件
- PC 机 Window7/8/10 (32/64bit)

3 实验要求

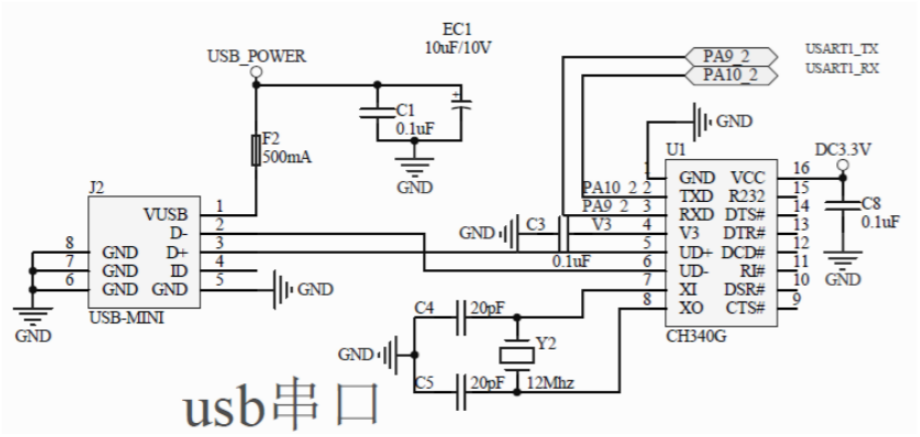
- 基本要求
  - 使用 USART3 控制 4 个 led 灯的亮灭闪烁
  - 命令格式：led n on/off/flash
  - 参数：38400-8-1（波特率-数据位-停止位，其余无）

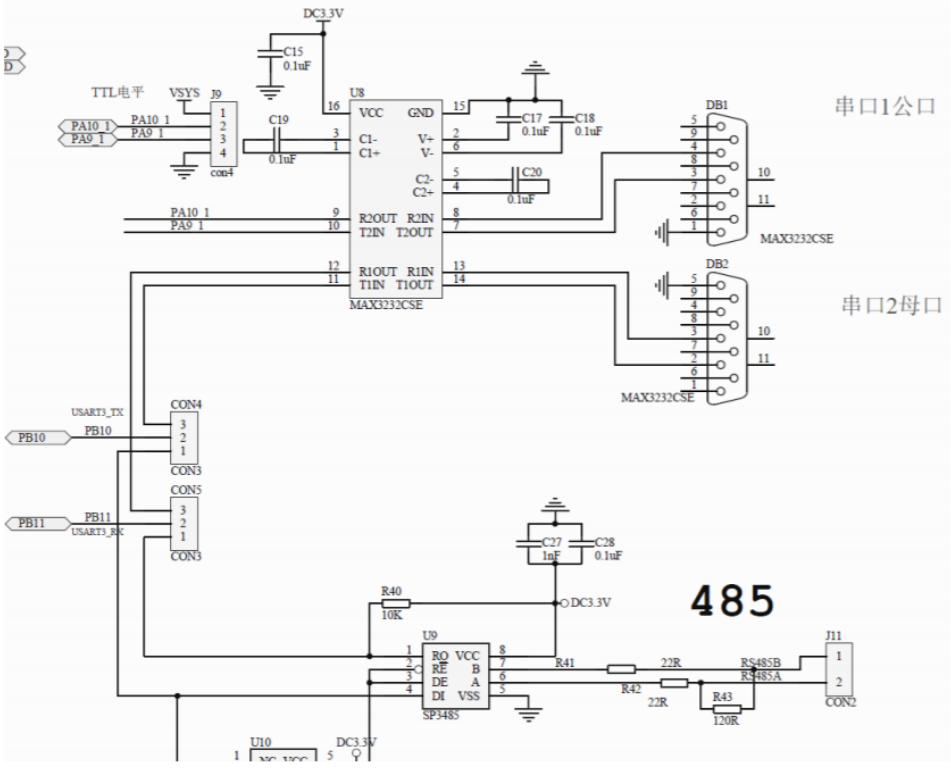
4 实验原理

《FS_STM32F4 底板原理图 V1.pdf》

USART1 使用 PA9、PA10

USART3 使用 PB10、PB11





《STM32F4x7-Datasheet.pdf》

42	68	101	E15	120	PA9	FT	PA9	USART1_TX/ TIM1_CH2 / I2C3_SMBA / DCMI_D0/OTG_FS_VBUS
43	69	102	D15	121	PA10	FT	PA10	USART1_RX/ TIM1_CH3/ OTG_FS_ID/DCMI_D1

29	47	69	R12	79	PB10	FT	PB10	SPI2_SCK / I2S2_CK / I2C2_SCL/ USART3_TX / OTG_HS_ULPI_D3 / ETH_MII_RX_ER / OTG_HS_SCL / TIM2_CH3
30	48	70	R13	80	PB11	FT	PB11	I2C2_SDA/USART3_RX/ OTG_HS_ULPI_D4 / ETH_RMII_TX_EN/ ETH_MII_TX_EN / OTG_HS_SDA / TIM2_CH4

USART 波特率的计算

26.3.4 小数波特率生成

对 USARTDIV 的尾数值和小数值进行编程时，接收器和发送器（Rx 和 Tx）的波特率均设置为相同值。

公式 1：适用于标准 USART（包括 SPI 模式）的波特率

$$\text{Tx/Rx 波特率} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

USARTDIV 是一个存放在 USART_BRR 寄存器中的无符号定点数。

- 当 OVER8=0 时，小数部分编码为 4 位并通过 USART_BRR 寄存器中的 DIV_fraction[3:0] 位编程。
- 当 OVER8=1 时，小数部分编码为 3 位并通过 USART_BRR 寄存器中的 DIV_fraction[2:0] 位编程，此时 DIV_fraction[3] 位必须保持清零状态。

注意： *对 USART_BRR 执行写操作后，波特率计数器更新为波特率寄存器中的新值。因此，波特率寄存器的值不应在通信时发生更改。*

OVER8=0 时如何从 USART_BRR 寄存器值中获取 USARTDIV

示例 1：

如果 DIV_Mantissa = 0d27 且 DIV_Fraction = 0d12 (USART_BRR = 0x1BC)，则

尾数 (USARTDIV) = 0d27

小数 (USARTDIV) = 12/16 = 0d0.75

因此 USARTDIV = 0d27.75

要获知 fck，首先需要知道 USART 接在哪个 APB 上

USART1-APB2，USART3-APB1

0x4001 3000 - 0x4001 33FF	SPI1		*第 769 页的第 27.5.10 节：SPI 寄存器映射*
0x4001 2C00 - 0x4001 2FFF	SDIO		*第 819 页的第 28.9.16 节：SDIO 寄存器映射*
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		*第 286 页的第 11.13.18 节：ADC 寄存器映射*
0x4001 1400 - 0x4001 17FF	USART6	APB2	*第 720 页的第 26.6.8 节：USART 寄存器映射*
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		*第 390 页的第 14.4.21 节：TIM1 和 TIM8 寄存器映射*
0x4001 0000 - 0x4001 03FF	TIM1		

0x4000 5000 - 0x4000 53FF	UART5		
0x4000 4C00 - 0x4000 4FFF	UART4		第 720 页的第 26.6.8 节：USART 寄存器映射
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		第 769 页的第 27.5.10 节：SPI 寄存器映射
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	APB1	

然后，需要知道 APB 输出的时钟频率，根据我们的配置：

APB1-42M APB2-84M

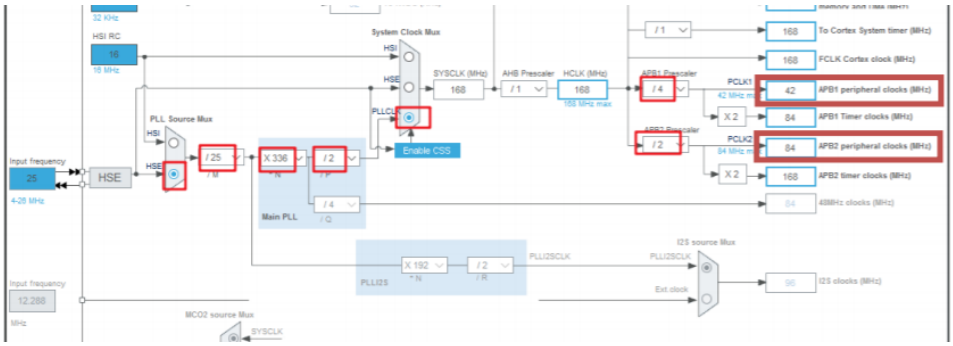


表 116. 采用 16 倍过采样时，在 f_{PCLK} = 42 MHz 或 f_{PCLK} = 84 Hz 下编程波特率时的误差计算(1)(2)

16 倍过采样时 (OVER8=0)							
波特率		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
序号	所需值	实际值	波特率寄存器中编程的值	误差 % =（计算值 - 所需值）/所需波特率	实际值	波特率寄存器中编程的值	误差 %
1	1.2 KBps	1.2 KBps	2187.5	0	1.2 KBps	4375	0
2	2.4 KBps	2.4 KBps	1093.75	0	2.4 KBps	2187.5	0
3	9.6 KBps	9.6 KBps	273.4375	0	9.6 KBps	546.875	0
4	19.2 KBps	19.195 KBps	136.75	0.02	19.2 KBps	273.4375	0
5	38.4 KBps	38.391 KBps	68.375	0.02	38.391 KBps	136.75	0.02
6	57.6 KBps	57.613 KBps	45.5625	0.02	57.613 KBps	91.125	0.02
7	115.2 KBps	115.068 KBps	22.8125	0.11	115.226 KBps	45.5625	0.02
8	230.4 KBps	230.769 KBps	11.375	0.16	230.137 KBps	22.8125	0.11

代码计算方式

stm32f4xx_hal_uart.c

```
#else
    if (huart->Instance == USART1)
    {
      huart->Instance->BRR = UART_BRR_SAMPLING16(HAL_RCC_GetPCLK2Freq(), huart->Init.BaudRate);
    }
#endif /* USART6 */
    else
    {
      huart->Instance->BRR = UART_BRR_SAMPLING16(HAL_RCC_GetPCLK1Freq(), huart->Init.BaudRate);
    }
```

stm32f4xx_hal_uart.c 是 uart 的 stm32 固件库，可以使用其中的函数完成本实

Shell 控制字：

Tab	'\t'
回车	'\r'，linux 终端'\r'+ '\0'
上下左右	up key : 0x1b 0x5b 0x41 down key: 0x1b 0x5b 0x42 right key:0x1b 0x5b 0x43 left key: 0x1b 0x5b 0x44
光标后退/删除	'\b'

本实验主要使用轮询式软件流程的技术：

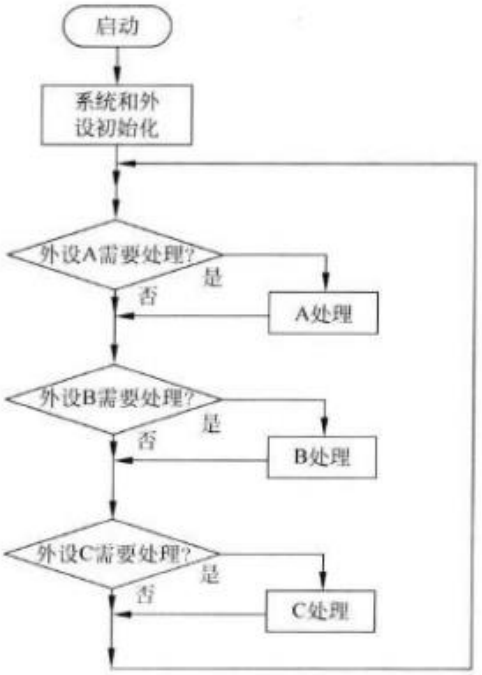


图 2.8 轮询方式的应用中存在多个需要处理的设备

5 实验步骤

5.1 基础部分

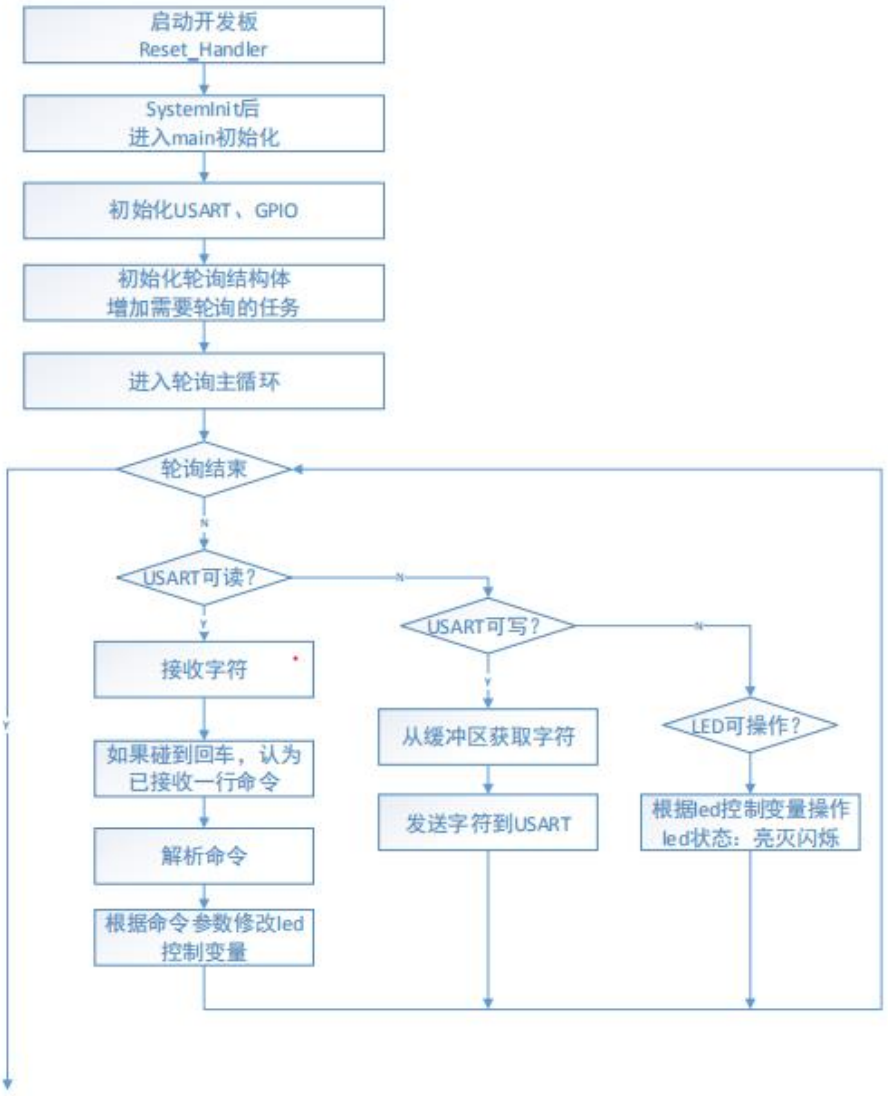
1. 下载并打开老师给的代码 emlab2020-lab3.rar

2. 连接好实验板的相关线路，打开电源，打开串口工具并打开响应串口。

3. 在 lab_main.c 中找到实验入口 lab3_a_main()，根据函数调用，理解一个个按顺序调用的函数，观察程序的执行过程和现象。

6 实验方案与实现

6.1 软件结构



6.2 源代码

- Lab3_a:

```
void uart_recv_process(void) {
  //# error "Not Implemented!"
  int num=0;
  char *ins[30];
  char c=board_getc();
  FIFO_PUSH_VAR(&sendfifo,c);

  command[cmd_idx++]=c;
  if (c=='\r'){
    trace_printf(command);
    num =split_cmdline(command,cmd_idx,ins);
    led_cmd_exec(num,ins);
    cmd_idx=0;
    memset(command,0,sizeof(command));
  }
}
```

7 实验结果与分析

基础部分的实验结果是将代码烧录到实验板后，在 sscom 的输入栏发送相应的命令，如 led 1 on，led 2 flash，led 3 off 等，led 灯作出相应反应。

8 实验总结

本次实验碰到的主要问题在于，串口收发显示在屏幕上的时候乱码，这证明采样率不对，经过多次实验，把波特率改成 9600，把代码里的所有跟波特率有关的数字改成 9600，才实现了正确的采样，解决了乱码问题。

在进行本次实验时，主要时间都花费在了调试环境上面，换了两块板子，重新在 ftp 上下了两次代码，才解决问题。