西安交通大學

XI'AN JIAOTONG UNIVERSITY

数字电子技术与微处理器基础

微处理器实验

课程名称：	数电与微处理器实验
班级：	电气 2102
姓名：	邓朴达
学院：	电气工程学院
专业：	电气工程及其自动化
学号：	2216113167

2023 年 12 月

西安交通大學实验报告

班级：电气 2102
姓名：邓朴达
学号：2216113167
日期：2023 年 12 月
地点：东一楼

课程名称：数字电子技术与微处理器基础　实验名称：微处理器实验
实验类型：课程实验

实验一：微处理器应用编程及基本输入/输出实验

一、 实验目的

(1) 熟练掌握开发环境及 CPU、外设接口、数据的观察、调试等开发方法。

(2) 通过 LED、按键，学习、掌握 I/O 的工作原理及编程、应用方法。

(3) C 语言、机器指令相结合，观察指令、寄存器, 理解、领会微处理器系统工作。

二、 实验原理图

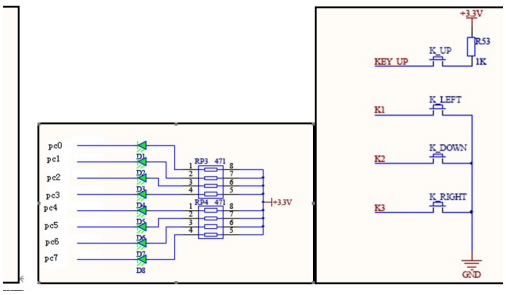

图 1: LED 和开关连线图

(1) KEY_UP 按键接 PA0 引脚，K1（PB2）、K2(PB3)、K3(PB4) 按键

(2) 8 个 LED 灯分别接 PC 口 (PC07) 的输出，当 PC 口某位输出为 0 时，相应指示灯即可点亮。

三、 目标要求

(1) 填充学号至 sBUF，通过 8 段 LED，轮流显示自己学号各位。

(2) 按下 UP 键（PA0），倒序（或暂停）显示自己学号。

(3) 根据学号个位数，调整更新间隔 (0.5s+ 学号个位 *0.1s)。

(4) 对 sBUF 前 10 个数据累加、结果存至 sBUF[15]。

(5) 最后设断点，在 UP 键按下时，可暂停至断点。

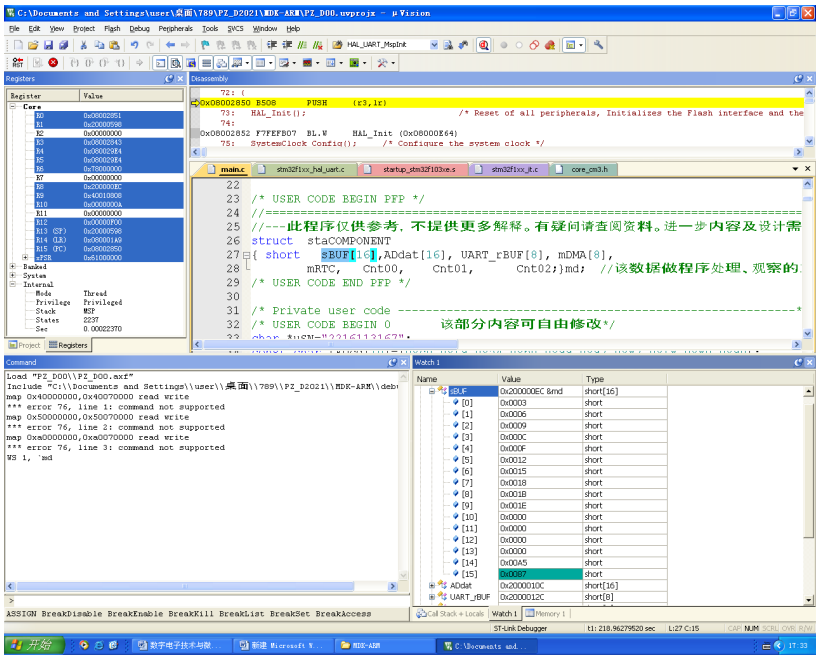

图 2: 数据存储结构体

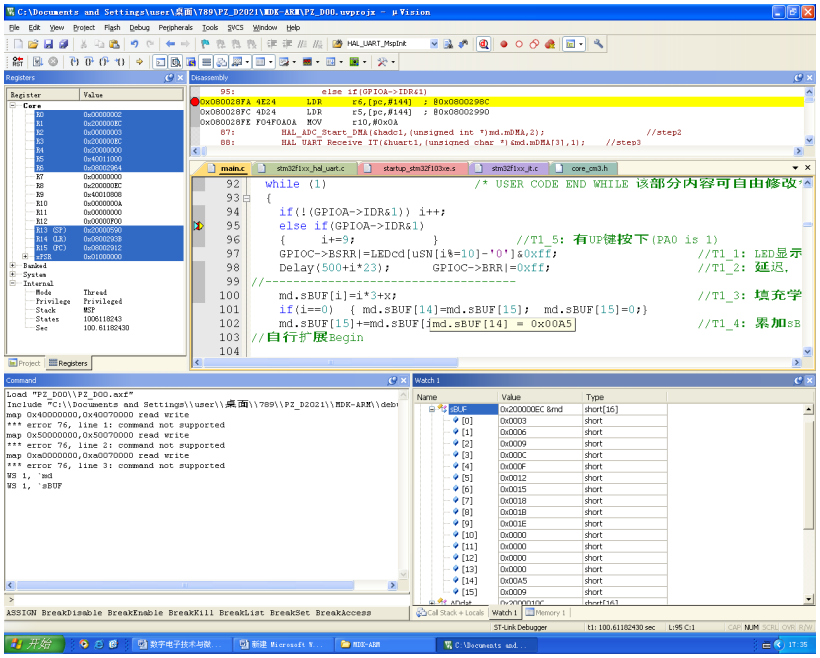

图 3: 程序界面截图

在 Debug 界面进行设置断点调试功能，发现按下 UP 按键以后，数码管显示初始电平设置，即低电平，现象为七段数码管全部点亮。

实验二：定时器及中断实验

一、 实验要求

(1) 了解 STM32-F1 系列处理器定时器及定时中断的工作原理及编程方法。

(2) 编写定时中断服务程序，完成周期性工作，并为其他模块提供时间控制。

二、 实验内容

设定定时器周期，设计定时中断服务程序。

三、 目标要求

不依靠软件延时 Delay(unsigned tDly)，在主程序实现 1Hz 及 10Hz 周期性简单处理任务（可通过计数变量如 Cntx 观察）。

四、 原理简述

配置定时器中断：

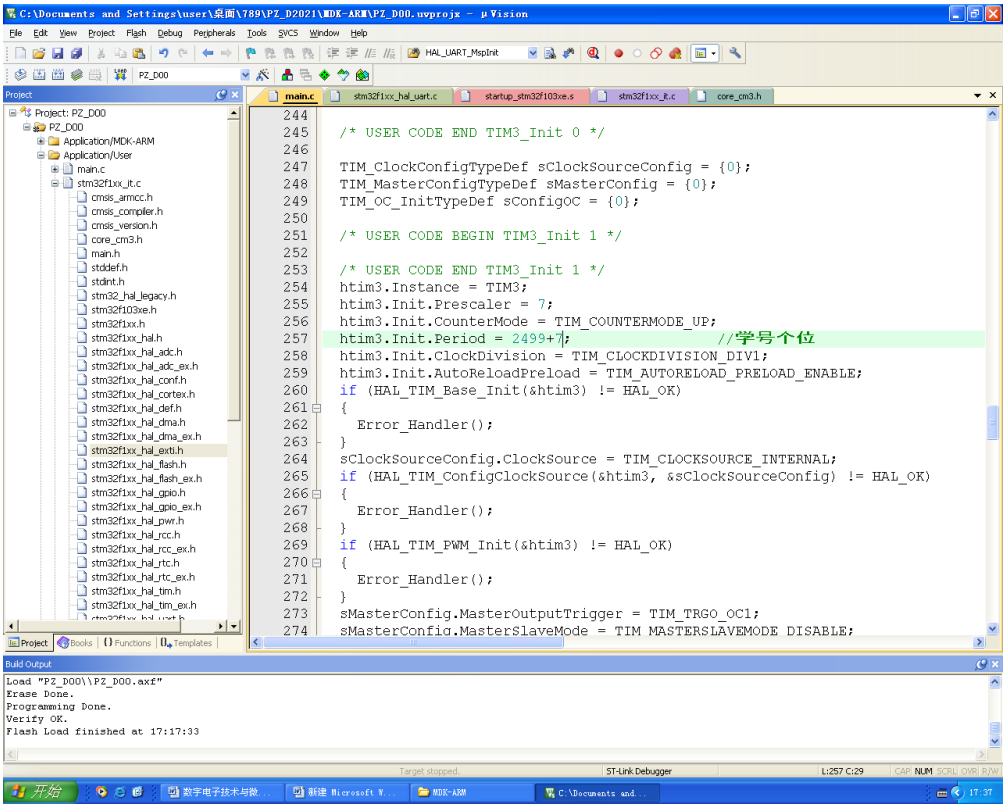

图 1: 定时器基本配置

(1) 定时器时钟。TIM3 时钟接在 APB1 上面，使用 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE) 打开 TIM3 时钟。

(2) 清除中断挂起位。在固件库中使用：void TIM_ClearITPendingBit(TIM_TypeDef*TIMx, u16 TIM_IT) 来清除中断挂起位，中断程序在开始前可能会被置位，因此需要消除中断挂起位。

(3) 初始化定时器基本配置。主要使用固件库中的 TIM_TimeBaseInit() 函数进行操作。

(4) 使能定时器 TIMx。直接使用 TIM_Cmd() 函数就可以。

(5) 使能 TIMx 中断。在库函数里面定时器中断使能是通过 TIM_ITConfig 函数来实现。

void TIM_ITConfig(TIM_TypeDef* TIMx, uint16_t TIM_IT, FunctionalState NewState);

该函数的形参：第一个参数是选择定时器号，取值为 TIM1-TIM17；第二个参数是用来指明使能的定时器中断的类型，定时器中断的；类型有很多种，包括更新中断 TIM_IT_Update，触发中断 TIM_IT_Trigger，以及输入捕获中断等等；第三个参数就是失能还是使能。

(6) 配置中断优先级。函数的原型为 void NVIC_Init(NVIC_InitTypeDef* NVIC_InitStruct)

(7) 编写中断服务程序。先清除中断挂起位，接着再编写中断处理内容即可。

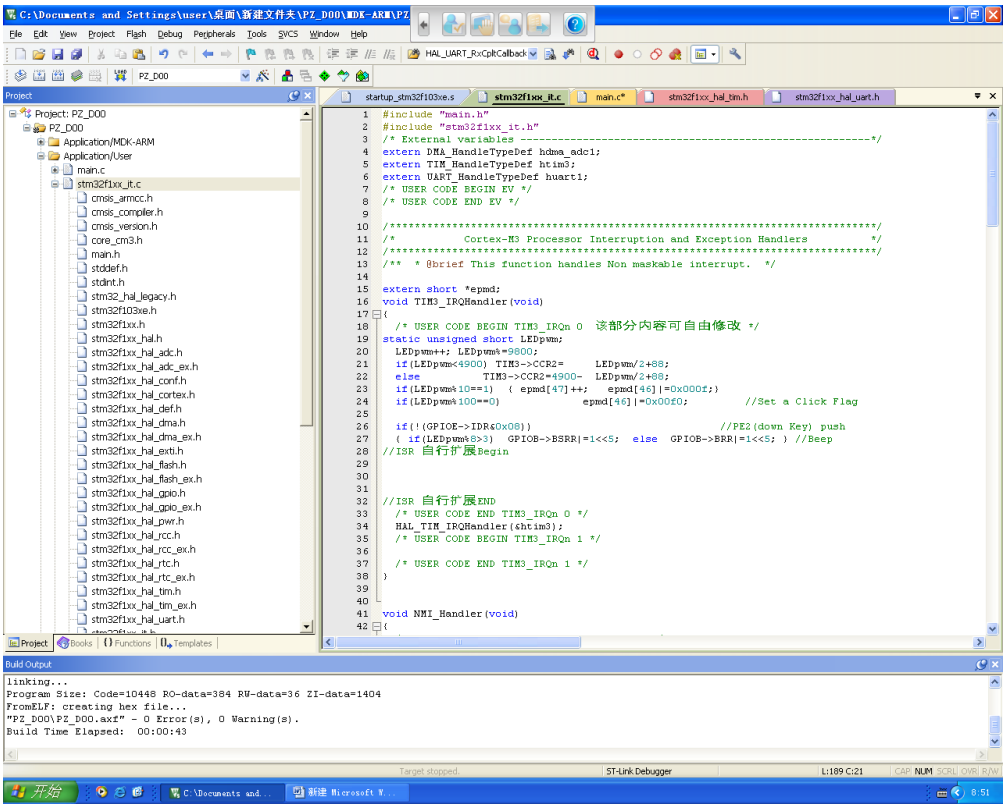

图 2: 中断服务函数界面

五、 实验程序和现象

按下 Down 按键之后，蜂鸣器按照一定频率开始响声。

```
void TIM3_IRQHandler(void)
{
 /* USER CODE BEGIN TIM3_IRQn 0 */
static unsigned short LEDpwm;
LEDpwm++; LEDpwm%=9800;
if(LEDpwm<4900) TIM3->CCR2= LEDpwm/2+88;
else TIM3->CCR2=4900- LEDpwm/2+88;
if(LEDpwm%10==1) { epmd[50]++; epmd[46]|=0x000f;}
if(LEDpwm%100==0) {
epmd[49]++;
epmd[46]|=0x00f0; } //Set a Click Flag
if(!(GPIOE->IDR&0x08))
//PE2(down Key) push
{ if(LEDpwm%8>3) GPIOB->BSRR|=1<<5; else GPIOB->BRR|=1<<5; }
//Beep
/* USER CODE END TIM3_IRQn 0 */
 HAL_TIM_IRQHandler(&htim3);
 /* USER CODE BEGIN TIM3_IRQn 1 */
 /* USER CODE END TIM3_IRQn 1 */
}
```

实验三：ADC 与 DMA 实验

一、 实验目的

(1) 了解 STM32-F1 系列处理器定时器 +ADC+DMA 工作原理及使用方法。

(2) 对 ADC 数据进行简单的处理、计算。

二、 实验内容

通过定时器 3 定时启动 ADC，自 DMA 缓冲区读取 ADC1.1 结果，保存、计算。

三、 目标要求

(1) 设定恰当采样率（如 2499+ 学号个位 7），以此采样率得到的 ADC 采样结果（PA1 通道），陆续保存至循环缓冲区 md.ADdat[0-7], 并在定时中断处理程序计算 8 点数据平均值保存到 ADdat[15]。

(2) 调整电位器，观察实验结果。

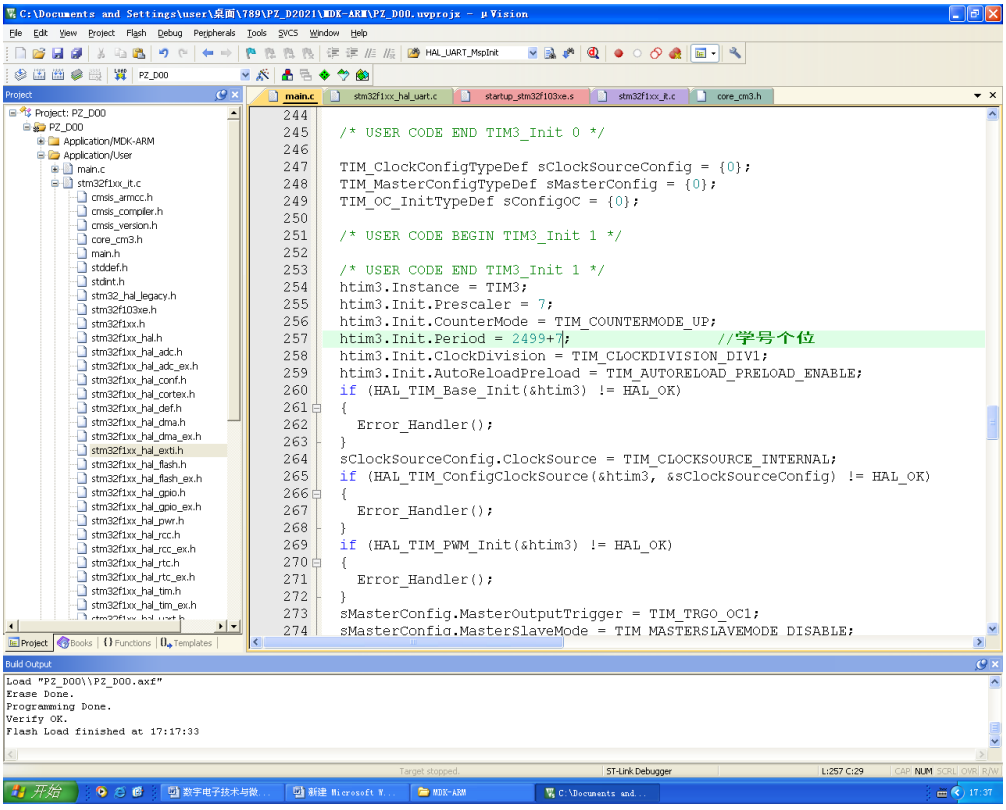

图 1: 采样率预设

四、 原理简述

ADC 模数转换模块是一种将模拟信号转换为数字信号进行处理的模块。在存储或传输时，模数转换器几乎必不可少。

ADC 经常和 DMA 共同使用，用于模拟到数字信号的获取和转换。

配置 ADC：对于 ADC 通道，每个 ADC 通道对应一个 GPIO 引脚端口，ADC 的输入需要使用模拟信号输入，所以，把 ADC1 的通道使用的 GPIO 引脚配置成模拟输入模式，GPIO 的引脚在设为模拟输入模式后可用于模拟电压的输入。

配置 DMA：内存直接访问。使用 DMA 的通道 1，数据从 ADC 的数据寄存器（ADC1_DR_Address）转移到内存（ADC_ConvertedValue 变量）当中, 内存外设地址都固定，使用 DMA 循环传输模式。

ADC 设置成连续转换模式，对应的 DMA 通道开启循环模式，ADC 就一直在进行数据采集然后通过 DMA 把数据搬运至内存。

加入定时中断，来定时读取内存中的数据。设置好定时器的触发间隔，就能实现 ADC 定时采样转换。

五、 实验程序

```
static void MX_TIM3_Init(void)
{
 /* USER CODE BEGIN TIM3_Init 0 */
 /* USER CODE END TIM3_Init 0 */
 TIM_ClockConfigTypeDef sClockSourceConfig = {0};
 TIM_MasterConfigTypeDef sMasterConfig = {0};
 TIM_OC_InitTypeDef sConfigOC = {0};
 /* USER CODE BEGIN TIM3_Init 1 */
 /* USER CODE END TIM3_Init 1 */
 htim3.Instance = TIM3;
 htim3.Init.Prescaler = 7;
 htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
 htim3.Init.Period = 2499+7;
 htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
 htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
 if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
 {
 Error_Handler();
 }
 sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
 if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
 {
 Error_Handler();
 }
 if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
 {
 Error_Handler();
 }
 sMasterConfig.MasterOutputTrigger = TIM_TRGO_OC1;
```

```
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
{
Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 6;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM3_Init 2 */
/* USER CODE END TIM3_Init 2 */
HAL_TIM_MspPostInit(&htim3);
}
```

六、 实验结果

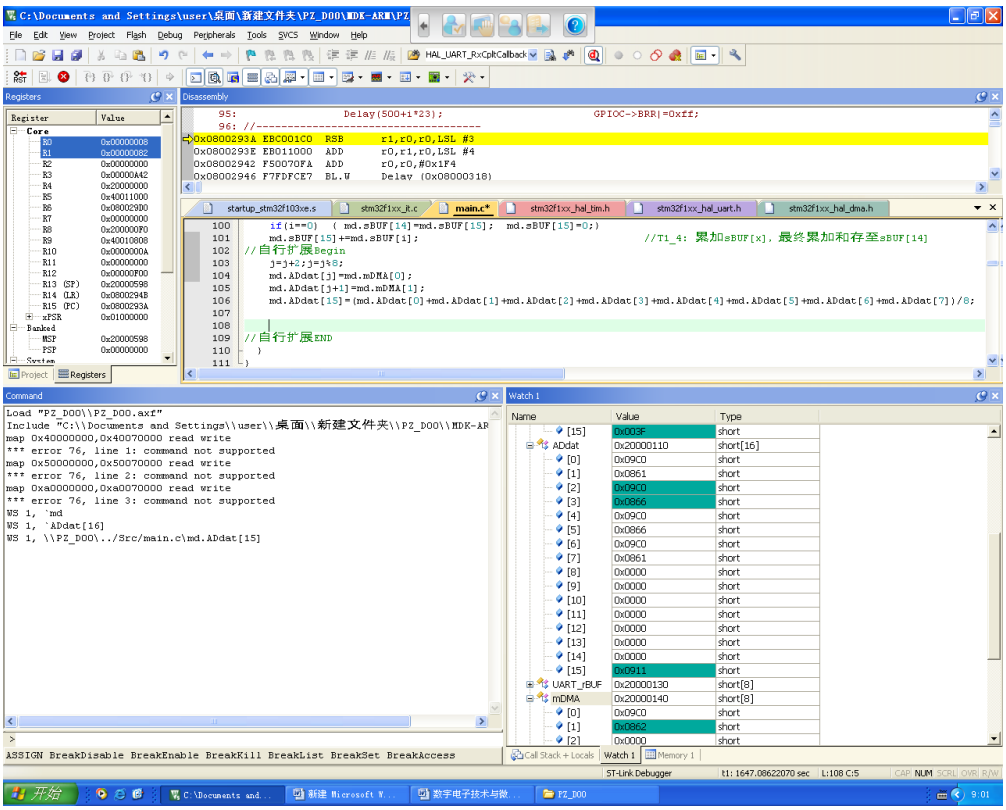

图 2: DMA 数据搬运调试

实验四:UART 串行通信实验

一、 实验目的

(1) 了解 STM32-F1 系列处理器 UART 的工作原理及编程方法。

(2) 简单处理收、发内容。

二、 实验内容

实现 UART 收、发功能。对收到的命令做处理，通过 UART 发送相应内容。

三、 目标要求

根据接收的自行约定命令代码，通过 UART 分别发送学号或 ADC 结果（2 字节）, 在 PC 串口观察相应内容。

四、 原理简述

STM32F10x 系列芯片，分别有 3 个 USART 和 2 个 UART。

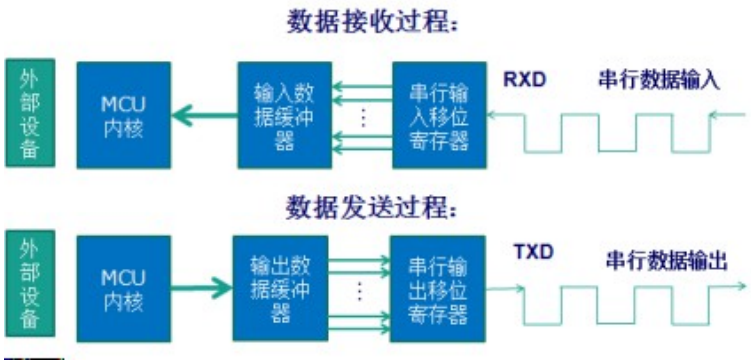

图 1: UART 原理图

STM32 的 UART 特点：

(1) 可编程的数据字长度 (8 位或者 9 位)

(2) 分数波特率发生器系统，提供精确的波特率。发送和接受共用的可编程波特率，最高可达 4.5Mbit-s/s

(3) 可配置的使用 DMA 多缓冲器通信

(4) 检测标志： 接受缓冲器 发送缓冲器空 传输结束标志

(5) 多个带标志的中断源，触发中断

(6) 全双工异步通信

(7) 其他：校验控制，四个错误检测标志

设备通过自身的 TxD 接口传输到接收设备的 RxD 接口，通信双方的数据包格式要规约一致才能正常收发数据。内容包括：起始位、数据位、奇偶校验位、停止位、波特率设置。

五、 实验结果

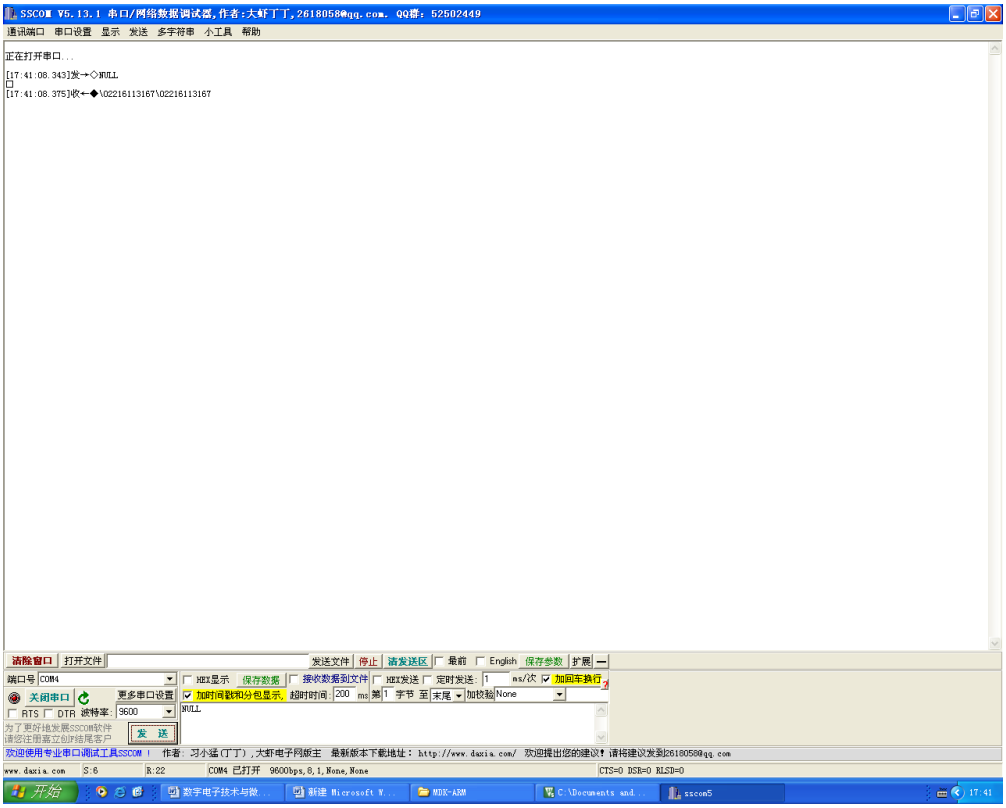

图 2: SSCOM 串口通信

通过发送特定指令 NULL，返回至 PC 学号信息。