# 2.协议栈中串口基础实验

## 实验内容

1. 掌握串口的使用

实现现象：

模块通过串口发送“UartInit OK”给电脑串口调试助手显示

串行简介

串行接口（Serial Interface）是指数据一位一位地顺序传送，其特点是通信线路简单，只要一对传输线就可以实现双向通信，从而大大降低了成本，特别适用于远距离通信，但传送速度较慢。一条信息的各位数据被逐位按顺序传送的通讯方式称为串行通讯。串行通讯的特点是：数据位的传送，按位顺序进行，最少只需一根传输线即可完成；成本低但传送速度慢。串行通讯的距离可以从几米到几千米；根据信息的传送方向，串行通讯可以进一步分为单工、半双工和全双工三种。

串口在嵌入式开发中非常重要，一般都要使用串口通讯、调试，所以学会串口使用也是必须的。实际上这个实验非常简单，和上个实验大部分一样，增加三个语句就可使串口工作，是不是信心十足啊。

实验详解： 使

用串口步骤： 1.

串口初始化

2. 注册串口任务任务
3. 串口发送

打开\软件资料\开发例程\5.zigbee 协议栈应用与组网\2.协议栈中串口基础实验

\ZStack-CC2530-2.3.0-1.4.0\Projects\zstack\Samples\SampleApp\CC2530DB\SampleApp.eww 工程。在左边 workspace 目录下比较重要的两个文件夹分别是 Zmain 和 App。我们开发主要在 App 文件夹进行，这也是用户自己添加自己代码的地方。主要修改 SampleApp.c 和 SampleApp.h 即可，如果增加传感器则增加相应的模块驱动到 App 里面，在 SampleApp.c 中调用就行。

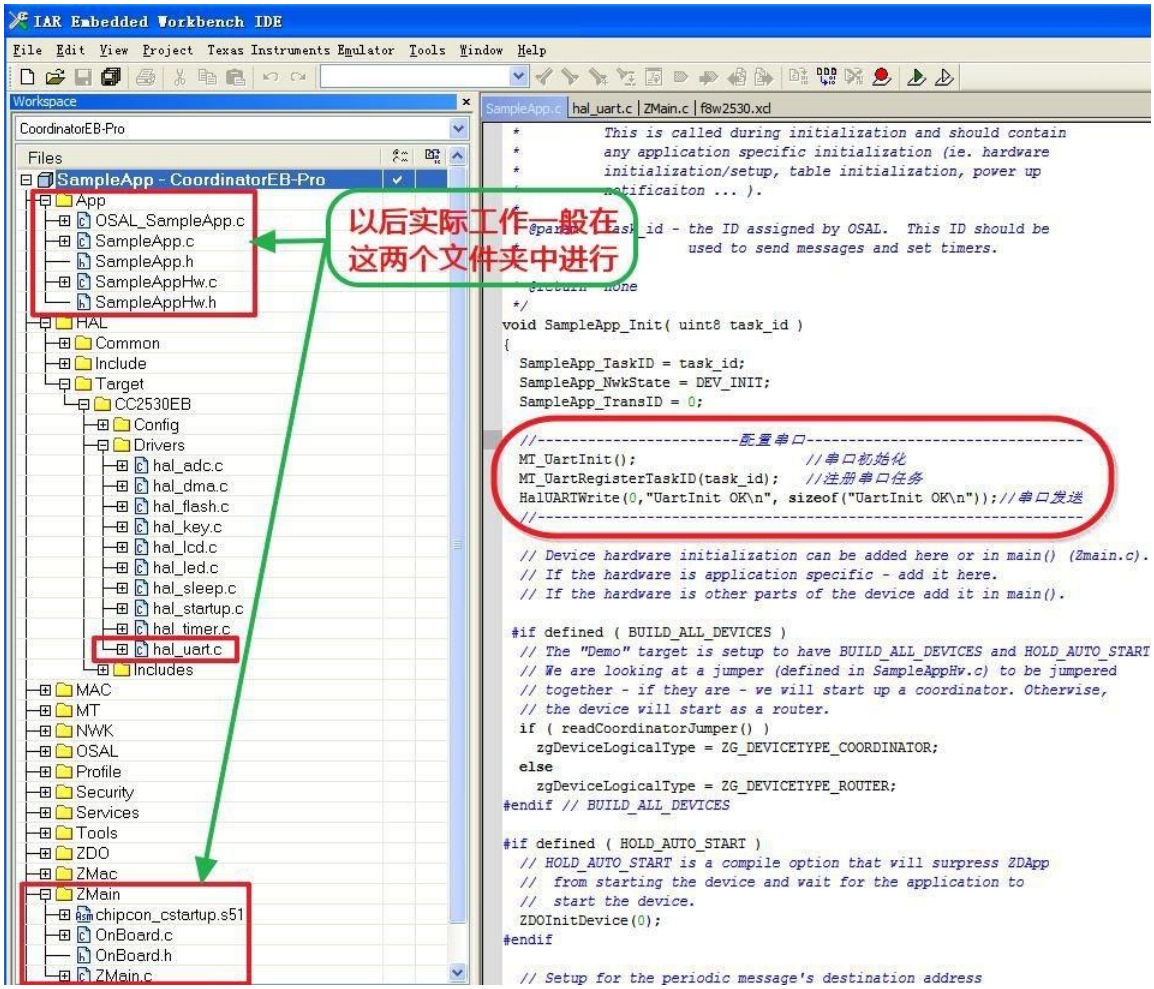
第一步：串口初始化 串口初始化相信大家很熟悉，就是配置串口号、波特率、校验位、数据位、停止位等等。在基础实验我们都是配置好寄存器然后使用。现在我们在 workspace 下找到 **HAL\Target\CC2530EB\drivers** 的 **hal_uart.c** 文件，我们可以看到里面已经包括了串口初始化、发送、接收等函数，全都封装好了；我们只需根据自己需要修改相关配置，调用相应的接口函数就可使用串口了，是不是觉得很方便？如图 1 所示。

浏览一下关于串口的操作函数还是挺全的。我们看看 workspace 上的 MT 层，发觉有很多基本函数，前面带 MT。包括 MT_UART.C，我们打开这个文件。看

到 MT_UartInit()函数，这里也有一个串口初始化函数的，没错 Z-stack 上有一个 MT 层，用户可以选用 MT 层配置和调用其他驱动。进一步简化了操作流程。我们已经知道串口配置的方法，那么应该在那里初始化呢？当然是 SampleApp_Init()，如果你还有疑问说明你上个实验还要复习一下，我们大概再说下吧。 用户自己添加的应用任务程序在 Zstack 中的调用过程是:

main() ---> osal_init_system()---> osalInitTasks()---> SampleApp_Init()

我们打开 APP 目录下的 SampleApp.c 发现 SampleApp_Init()函数。我们在这里加入串口初始化代码。




图一

**MT_UartInit();**设置串口参数

```
1.  void MT_UartInit ()
2.  {
3.  halUARTCfg_t uartConfig;
4.  /* Initialize APP ID */
5.  App_TaskID = 0;
6.  /* UART Configuration */
7.  uartConfig.configured = TRUE;
8.uartConfig.baudRate = MT_UART_DEFAULT_BAUDRATE;
9.uartConfig.flowContro l = MT_UART_DEFAULT_OVERFLOW;
```

```
10.  uartConfig.flowControlThreshold     =   MT_UART_DEFAULT_THRESHOLD;
11.          uartConfig.rx.maxBufSize                                          =
MT_UART_DEFAULT_MAX_RX_BUFF;
12.  uartConfig.tx.maxBufSize              =   MT_UART_DEFAULT_MAX_TX_BUFF;
13.          uartConfig.idleTimeout                                            =
MT_UART_DEFAULT_IDLE_TIMEOUT;
14.  uartConf ig.intEnable                    = TRUE;
15.  #if defined (ZTOOL_P1) || defined (ZTOOL_P2)
16.  uartConfig.callBackFunc                 = MT_UartProcessZToolData;
17.  #elif defined (ZAPP_P1) || defined (ZAPP_P2)
18.  uartConfig.callBackFunc                 = MT_UartProcessZAppData;
19.  #else
20.  uartConfig.callBackFunc                 = NULL;
21.  #endif
22.  /* Start UART */
23.  #if defined (MT_UART_DEFAULT_PORT)
24.  HalUARTOpen (MT_UART_DEFAULT_PORT, &uartConfig);
25.  #else
26.  /* Silence IAR compiler warning */
27.  (void)uartConfig;
28.  #endif
29.  /* Initialize for ZApp */
30.  #if defined (ZAPP_P1) || defined (ZAPP_P2)
31.  /* Default max bytes that ZAPP can take */
32.  MT_UartMaxZAppBufLen     = 1;
33.  MT_UartZAppRxStatus      = MT_UART_ZAPP_RX_READY;
34.  #endif
35.  }
```

第 8 行：uartConfig.baudRate = MT_UART_DEFAULT_BAUDRATE;是配置波特率，我们 go to definition of MT_UART_DEFAULT_BAUDRATE,  可以看到：

```
#define MT_UART_DEFAULT_BAUDRATE HAL_UART_BR_38400
```

默认的波特率是 38400bps,现在我们修改成 115200bps,修改如下：

```
#define MT_UART_DEFAULT_BAUDRATE HAL_UART_BR_115200
```

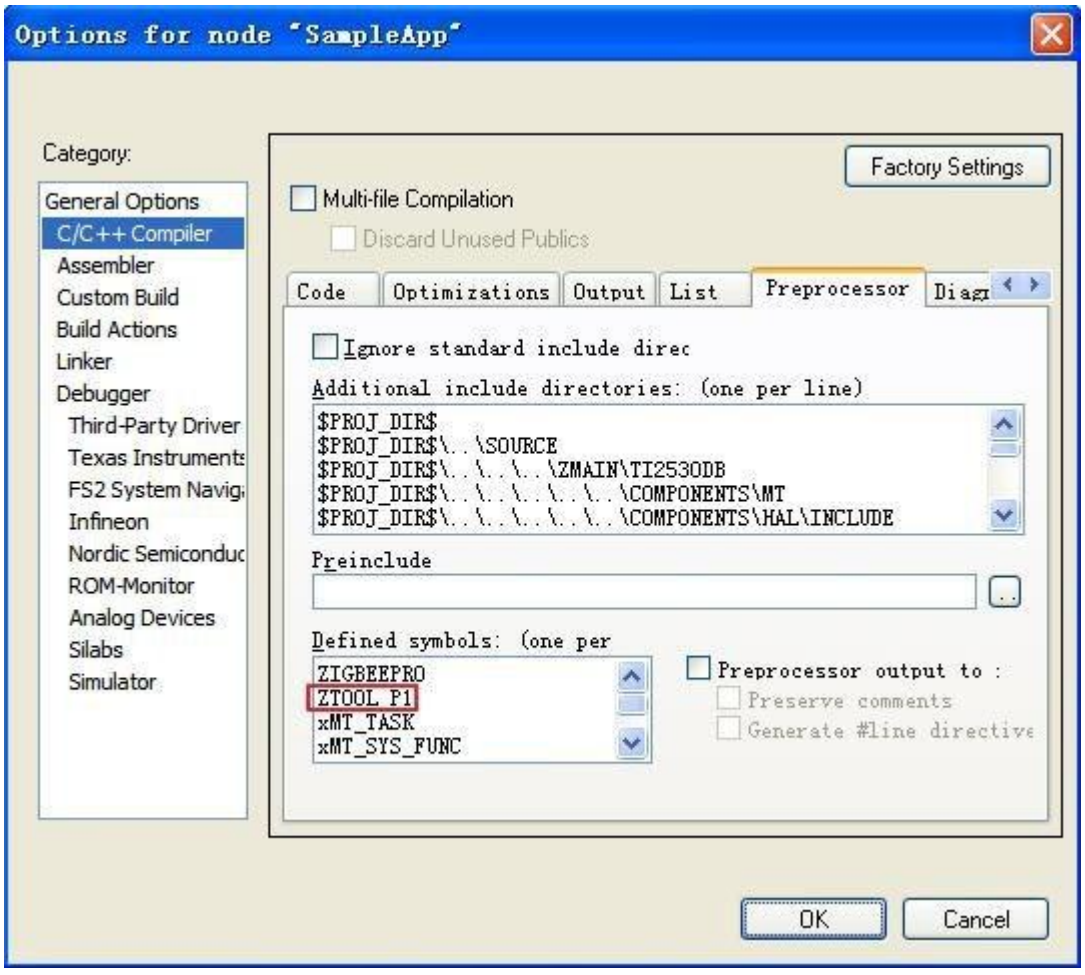
第 9 行：uartConfig.flowControl = MT_UART_DEFAULT_OVERFLOW;  语句是配置流控的，我们进入定义可以看到：

```
#define MT_UART_DEFAULT_OVERFLOW TRUE
```

默认是打开串口流控的，如果你是只连了 TX/RX 2 根线的方式务必关流控。注意：2 根线的通讯连接一定要关流控，不然是永远收发不了信息的，现在大部产品很少用流控。

```
#define MT_UART_DEFAULT_OVERFLOW FALSE
```

第 16~22 行：这个是预编译，根据预先定义的 ZTOOL 或者 ZAPP 选择不同的数据 处理函数。后面 的 P1 和 P2 则是串口 0 和 串口 1。我们用 ZTOOL，串 口 0。我们可 以在 option——C/C++ 的 CompilerPreprocessor 地方加入。如图 2 所示。至此初始化配置完了。


第二步：注册 串口任务任务　在 **SampleApp_Init();** 刚添加的串口初始画语句下面加入语句：

**MT_UartRegisterTaskID(task_id);**　　//注册串口任务任务

第三步：串口 发送　经过前面两个步骤，现在串口已经可以发送信息了,增加代码如图1所示。

**HalUARTWrite(0,"UartInit OK\n",　sizeof("UartInit OK\n"));** //串口发送

在项目配置选项卡中预编译处加入以下一些内容，如图3所示。

ZIGBEEPRO

ZTOOL_P1

xMT_TASK

xMT_SYS_FUNC

xMT_ZDO_FUNC

LCD_SUPPORTED=DEBUG

连接仿真器和 USB 转串口线,选择 CoordinatorEB-Pro,编译完成后  下载和调试。配 置串口点调试助手为：115200 8N1 并打开串口,(串口请选择自己的端口号)。在 IAR 点 
全速运行，可以看到串口调试助手收到模块发过来的字符串。


也许仔细的朋友会发现 xMT_TASK，xMT_SYS_FUNC，xMT_ZDO_FUNC 前面都有个 x，事实上真正的宏是 MT_TASK，MT_SYS_FUNC，MT_ZDO_FUNC，加了 x 表示不定义它们了。给大家布置一个小实验，去掉上面的 x,编译后下载看串口会收到什么？会在“UartInit OK”前面出现一段乱码，如果用 16 进制显示为 FE 开头的字符串，这是 Z-stack MT 层定义的串口发送格式，更具体的介绍会在后面的实验提到。