OpenModbus使用说明

1. 概述

Modbus是一种串行通信协议，是Modicon公司（现在的施耐德电气 Schneider Electric）于1979年为使用可编程逻辑控制器（PLC）通信而发表。由于它的易用性和可靠性，它已经是一个被广泛接受的串行协议，而Modbus RTU更广泛应用于楼宇管理系统(BMS)和工业自动化系统(IAS)。

1. 介绍

OpenModbus是一个开源的Modbus RTU协议栈，支持多主/从模式配置于同一个设备当中。它由以下几部分组成：

1. Modbus.c/.h stack 核心文件，用户无需修改
2. Modbus\_Porting.c/.h 移植文件，如果用户需要切换处理器平台则需要修改
3. ModbusSlaveApp.c/.h 从站用户文件，用户需要根据自身的需求进行修改
4. ModbusMasterApp.c/.h 主站用户文件，用户需要根据自身的需求进行修改
5. ModbusUserConfig.h 用户配置文件，用户需要根据自身需求进行修改

默认情况下，OpenModbus使用NXP Kinetis KV4x作为示例平台，并在TWR-KV4x开发板上进行验证，使用IAR工程进行开发，底层驱动代码源于MCUxpresso SDK2.4.1。

OpenModbus有以下几个特性：

1. 支持多种UART工作模式: 查询、中断、DMA
2. 支持RS485模式，UART收发使能支持使用特定的RTS管脚或使用任意GPIO
3. 默认情况下，使用中断接收数据，并使用DMA发送数据以减小CPU的负荷。协议处理函数已经发送触发函数都在主循环中调用
4. 非常容易移植到其他处理器平台，仅需要移植Timer、UART相关的几个函数即可，请参考第五章内容
5. 对于从站而言，用户需要在main函数中初始化协议栈，并在主循环中调用核心文件提供的相关函数即可，协议栈会自动将g\_ucMBUF全局数组映射到Modbus中，用户仅需要对这个数组进行操作即可轻松完成从站功能。
6. 对于主站而言，ModbusMasterApp提供了一个通过状态机来切换访问不同从站的示例，用户可以在此基础上根据应用来修改不同的站地址，寄存器地址/访问长度来做修改。示例代码使用g\_u8MasterBuf数组作为从站数据缓存，协议栈会根据用户定义的偏移，来将不同的从站数据放到不同偏移的地址上，这样用户仅需要处理g\_u8MasterBuf数组的内容即可。通过状态机的方式可以节省配置从站所用的资源，如果MCU资源充足，也可以通过列表的方式来处理。
7. 用户配置
8. ModbusUserConfig.h文件相关配置：

MODBUS\_MASTER\_USED表示开启设备主站功能

MODBUS\_SLAVE\_USED 表示开始设备从站功能

MODBUS\_SLAVE\_NUMBER表示从站设备个数

MODBUS\_MASTER\_NUMBER表示主站设备个数

SLAVE\_PORT0表示从站设备端口号（从0开始）

MASTER\_PORT0表示主站设备端口号（从0开始）

1. Modbus\_Porting.h文件相关配置：

//UART

#define UART\_DMA\_USED 1 //DMA总开关

#define UART\_TX\_DMA 1 //总开关为1时，该宏表示使用DMA进行TX

#define UART\_RX\_DMA 0 //总开关为1时，该宏表示使用DMA进行RX，注意目前仅仅使用DMA完成环形缓冲区接收，按照标准的Modbus协议是需要再找利用一路DMA打时标，通过时标来寻找帧头的，该功能尚未完成，故请不要设置为1.

//TIMER

#ifndef TIMER\_MINUS

#define TIMER\_MINUS 1

#endif

默认情况下，使用递减Timer

1. 用户实例

int main(void)

{

BOARD\_InitPins();

BOARD\_BootClockRUN();

//WDOG\_Configuration();

#ifdef MODBUS\_SLAVE\_USED

ModbusSlaveInitPort(0);

#endif

#ifdef MODBUS\_MASTER\_USED

MB\_Init();

ModbusMasterInitPort(0);

#endif

while (1)

{

#ifdef MODBUS\_SLAVE\_USED

ModbusSlaveMainProcess(SLAVE\_PORT0);

ModbusSlavePollSend(SLAVE\_PORT0);

ModbusNet1SlaveAPP();

#endif

#ifdef MODBUS\_MASTER\_USED

ModbusNet1MasterAPP();

ModbusMasterSendMessage(MASTER\_PORT0);

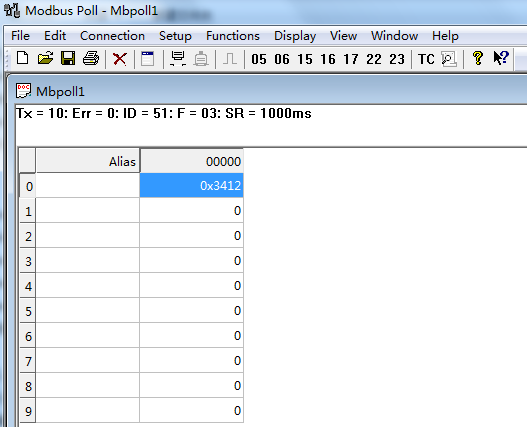
ModbusMasterMainReceive(MASTER\_PORT0);

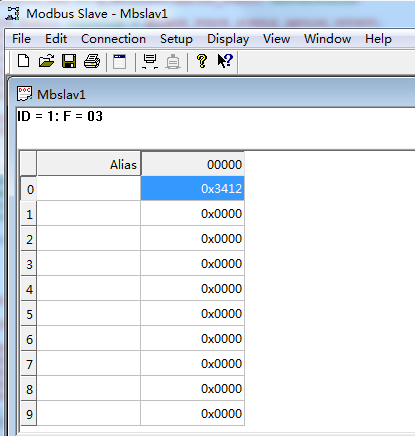
#endif

//WDOG\_Refresh(WDOG);

}

}





1. 协议栈移植

OpenModbus使用Modbus\_Port结构体来抽象Modbus端口，用户可以根据自己的需要来实例化不同的端口：

typedef struct \_modbus\_port

{

uint8\_t SendFlag;

uint8\_t ReceiveFlag;

uint8\_t ucSize;

uint8\_t g\_ucSendLen;

uint8\_t Local\_Address;

uint32\_t SendStartTime;

uint32\_t TimeOutLimit;

RxBuffer s\_RxBuf;

TxBuffer s\_TxBuf;

void (\*Timer0\_Init)(void);

uint32\_t (\*Timer0\_Value\_Get)(void);

uint32\_t (\*Timer0\_Wait3\_5char)(void);

void (\*UART\_SendData)(uint8\_t \*pbyData, uint16\_t uCount, uint8\_t port);

void (\*UART\_BaudrateSet)(uint32\_t buadrate);

}

Modbus\_Port;

假设用户所选MCU支持3路Modbus，其中两路为从站，一路为主站，需要在ModbusUserConfig.h中做如下定义：

#define MODBUS\_MASTER\_USED

#define MODBUS\_SLAVE\_USED

#ifdef MODBUS\_SLAVE\_USED

#define SLAVE\_PORT0 0

#define SLAVE\_PORT1 1

#define MODBUS\_SLAVE\_NUMBER 2

#endif

#ifdef MODBUS\_MASTER\_USED

#define MASTER\_PORT0 0

#define MODBUS\_MASTER\_NUMBER 1

#endif

配置完成后，需要移植结构体中相应的指针函数以及接收中断服务函数：

需要注意：默认使用的定时器PIT是递减的，如果需要使用递增定时器，需要#define TIMER\_MINUS 0

void (\*Timer0\_Init)(void); // init the timer for decide the FH(frame header)

uint32\_t (\*Timer0\_Value\_Get)(void); //get the timer value

uint32\_t (\*Timer0\_Wait3\_5char)(void);// get the 3.5char timer value.

void (\*UART\_SendData)(uint8\_t \*pbyData, uint16\_t uCount, uint8\_t port); //Send the DATA

void (\*UART\_BaudrateSet)(uint32\_t buadrate); //Set the UART baudrate.

void MODBUS\_SLAVE\_Rx\_ISR(); //RX interrupt.

对于定时器而言，没有必要每个Port都分配一个实际的timer，所有Port可以共享一个timer，但是Timer0\_Wait3\_5char这个函数根据不同端口的波特率不同而不同。