**1.**  **FreeModbus协议分析**

协议必须首先调用初始化功能eMBinit()函数。后调用eMBEnable()，最后，在循环体或者单独一个任务中调用eMBPoll()函数。

**2.**  **应用层协议**

**2.1.**    **系统的启动**

**2.1.1.**  **eMBInit()函数的源码分析**

以RTU方式为例，首先，检查调用的地址是否合法。如不合法，返回错误。如果合法则继续执行，

首先，针对RTU方式还是ASCII方式，选择不同的编译模块。

对需要调用的函数指针进行复制。如果移植需要改变其他用途，则要修改相应的指针，包括如下赋值：

            pvMBFrameStartCur = eMBRTUStart;

            pvMBFrameStopCur = eMBRTUStop;

            peMBFrameSendCur = eMBRTUSend;

            peMBFrameReceiveCur = eMBRTUReceive;

            pvMBFrameCloseCur = MB\_PORT\_HAS\_CLOSE ? vMBPortClose : NULL;

            pxMBFrameCBByteReceived = xMBRTUReceiveFSM;

            pxMBFrameCBTransmitterEmpty = xMBRTUTransmitFSM;

            pxMBPortCBTimerExpired = xMBRTUTimerT35Expired;

然后调用eStatus = eMBRTUInit( ucMBAddress, ucPort, ulBaudRate, eParity );具体初始化通讯端口。

**2.1.2.**  **eMBRTUInit**

eMBRTUInit这个函数主要干两件事：

第一，   初始化串口：

if( xMBPortSerialInit( ucPort, ulBaudRate, 8, eParity ) != TRUE )

{

   eStatus = MB\_EPORTERR;

}

这个函数在portserial.c中，需要用户在移植的时候根据自己的处理器编写。

第二，   初始化计时器：首先要根据波特率计算一下是3.5~5.0个字节周期的时间，然后再调用xMBPortTimersInit( ( USHORT ) usTimerT35\_50us )，初始化计时器。这个函数在porttimer.c中，需要用户在移植的时候根据自己的处理器编写。

**2.1.3.**  **eMBEnable源码分析**

首先，看看Modbus功能是否是被关闭的，如果不是被关闭（可能是没有被初始化或者已经打开），就返回错误。

如果是disable状态，就干下面两件事：

  调用pvMBFrameStartCur()。由于这是个函数指针，在模块eMBInit中，指向了eMBRTUStart函数

  在源代码中有这样一段注释：，意思是，首先设置成STATE\_RX\_INIT，然后打开计时器，等待t3.5以后，进入STATE\_RX\_IDLE状态。

  看源代码中，首先有设置Receiver的状态，后调用vMBPortSerialEnable，设置接收状态，然后打开定时器。

  当定时器中断后，自动调用中断服务程序，在中断服务程序中，只调用了pxMBPortCBTimerExpired，而这是一个函数指针，在RTU方式初始化时，被指向了xMBRTUTimerT35Expired()函数。

  xMBRTUTimerT35Expired函数在mbrtu.c中，在这里，我们只看第一种方式，就是进入初始化状态，在t35时间以后，只调用了一个xNeedPoll = xMBPortEventPost( EV\_READY );

  xMBPortEventPost函数就是在事件队列里加了一个EV\_RDY事件。

  然后，将eMB状态改为使能状态，

  初始化结束。

**2.2.**    **总线侦听eMBPoll()**

首先，判断系统是否被使能，如果没有，则返回错误值。

然后，检查是否有事件发生，如果有，则根据不同类型的事件响应：

  如果是EV\_RDY，表示系统刚刚进入侦听状态，则什么都不做；

  如果状态为EV\_FRAME\_RECEIVED，也就是接收到完整的帧，做下面两件事情：

  调用eStatus=peMBFrameReceiveCur( &ucRcvAddress, &ucMBFrame, &usLength )。这是一个函数指针，在eMBInit中，被初始化指向eMBRTUReceive。

  eMBRTUReceive这个函数首先校验帧的长度和CRC，然后从协议中解析出地址、数据和长度。

  然后检查地址，如果是广播地址或者是本机地址，就调用xMBPortEventPost( EV-EXECUTE)，将接收器的状态更改为EV\_EXECUTE。

  如果状态为EV\_EXECUTE，就在函数列表中检查，有没有与命令字段相符合的函数来解析相应则执行该函数，否则返回非法功能代码。

**2.3.** **数据发送**

发送数据通过指针eMBRTUSend，调用eMBRTUSend函数。

**2.3.1.**  **eMBRTUSend函数**

这个函数的作用就是打包，将数据打包成帧。

  首先，检查接收状态。因为MODBUS是基于RS-485半双工通讯，所以当正在接收数据时，不发送该帧。

  如果总线空，就将数据打包，将地址和CRC加入数据帧

  将总线状态改为发送。

**2.4.** **功能注册**

  对于指定的功能代码，需要一个功能回调函数来处理，格式如下。

eMBException eMXXXXXX ( UCHAR \* pucFrame, USHORT \* usLen )

  需要通过函数eMBRegisterCB(功能代码，函数名)加到处理代码中。具体源码分析从略。

**2.4.1.**  **prvvUARTTxReadyISR()**

总线状态改为发送后，会在发送缓冲时，自动调用prvvUARTTxReadyISR()中断服务程序。prvvUARTTxReadyISR()只调用了一个函数，就是pxMBFrameCBTransmitterEmpty ()。

**2.4.2.**  **pxMBFrameCBByteReceived ()**

pxMBFrameCBTransmitterEmpty ()是一个指针，指向了xMBRTUTransmitFSM函数。

**3.**  **数据链路层协议**

数据链路层是最基本的打包部分，将数据打包成帧，送到应用层。在数据链路层协议中，使用中断方式来接受。那么每次接收到字符就自动调用接收字符的ISR程序。按照规定，应该将中断服务程序安装给prvvUARTRxISR(void)函数。实际上这个函数只调用了一个函数：

pxMBFrameCBByteReceived()，这个指针调用了xMBRTUReceiveFSM函数。

**3.1.** **xMBRTUReceiveFSM()函数**

函数首先检查是不是处于发送状态。如果处于发送状态，直接退出。

  首先调用xMBPortSerialGetByte( ( CHAR \* ) & ucByte )，获取从串口读到的字符。

  然后检查接受状态：

  如果是错误状态或者处于初始化状态，那么直接等待，错过该帧。

  如果是STATE\_RX\_IDLE空闲状态，则将指针重置，将收到的第一个字节存储到缓冲区，并将状态改为STATE\_RX\_RCV状态。

  如果处于接收状态，就判断，如果缓冲区未满，就将收到的字节放入缓冲区，否则改为错误状态。

  不管在任何状态，最后都开启了t35计时器。在t35结束的时候，通过指针调用了xMBRTUTimerT35Expired()函数。

  xMBRTUTimerT35Expired()函数检查状态，如果是接收状态那就表明，已经有t35这么长的时间里，没有收到任新字节，当前的帧结束。在队列里增加一个EV\_FRAME\_RECEIVED事件。

  如果是错误状态，什么都不做。

  然后关掉计时器，将状态改为空闲。

**3.2.** **xMBRTUTransmitFSM()函数**

xMBRTUTransmitFSM首先判断总线是否忙，如果忙，则终止。如果不忙，则继续，根据发送状态变量：

  如果当前为STATE\_TX\_IDLE（空闲）状态，则打开端口发送

  如果当前状态为STATE\_TX\_XMIT，则进一步判断发送队列是否为空，

  如果不空，则发送下一个字符

  如果空，说明发送完成，关闭发送端口，改为侦听，并将状态改为空闲。

**4.**  **传输控制**

除了传输控制以外，还有传输控制的若干函数。通过下面几个指针来调用：

         pvMBFrameStopCur()

         pvMBFrameCloseCur()

**4.1.** **pvMBFrameStopCur()函数**

pvMBFrameStopCur是一个函数指针，在RTU方式下，它指向eMBRTUStop()函数。该函数做下面几件事情:

  关闭侦听和发送

  关闭定时器

**4.2.** **pvMBFrameCloseCur()函数**

这个指针指向一个叫做vMBPortClose()的函数，该函数目前只有在mbport.h中的声明，而没有实现。需要等到后面的版本再实现。