　1、eMBErrorCode为枚举类型变量，代表错误码，共有8个错误代号。常用的是MB\_ENOERR，即没有错误。  
　　2、eMBMode枚举类型变量代表设备的工作模式，分别是MB\_RTU、MB\_ASCII和MB\_TCP。  
　　3、eMBEventType枚举类型变量定义了event的类型，分别是EV\_READY，代表Startup启动完成；EV\_FRAME\_RECEIVED代表接收到帧；EV\_EXECUTE代表执行功能函数；EV\_FRAME\_SENT代表帧已发送。  
　　4、eMBParity枚举类型变量代表奇偶校验选项，分别是MB\_PAR\_NONE无校验，MB\_PAR\_ODD奇校验，和MB\_PAR\_EVEN偶校验。  
　　5、mb.c文件中的静态变量ucMBAddress存储设备地址，此变量在eMBInit函数中初始化。  
　　6、在C51Modbus中将freeModbus库中的源码进行了更改，例如尽量不使用函数指针，而是直接调用相关功能函数，根据eMBCurrentMode中的工作模式，来判断调用哪个函数。在freeModbus库中某些函数声明前加上reentrant，这是Keil编译器特有的关键词。这样做带来的一个不足是：不能动态绑定函数，从而导致库代码失去可移植性。这样做是C51编译器与ANSI标准不兼容的特殊性导致的。  
　　7、ENTER\_CRITICAL\_SECTION()和EXIT\_CRITICAL\_SECTION()宏，实际上就是关闭和打开全局中断。  
　　8、带xMBPort前缀的函数都属于port layer层，也就是独立于ModBus协议栈。  
　　9、freeModbus库中函数名称的第一个字母表示返回值类型，例如e表示返回enum枚举类型；v表示void无返回值；x表示BOOL布尔类型。注意这条规则并不是总成立，但主要函数基本上还是符合此规则的。第一个字母后的MB代表是属于ModBus协议栈的函数。  
　　10、port.h文件中宏#define F\_MCU 定义了单片机的工作频率。需要用其值计算Uart0定时器和Tick定时器的重装入值。  
　　11、在程序主函数main中，使用协议栈的方法是：  
　　eStatus = eMBInit( MB\_RTU, 0x0A, 0, 9600, MB\_PAR\_EVEN );  
　　/\* Enable the Modbus Protocol Stack. \*/  
　　eStatus = eMBEnable( );  
　　for( ;; )  
　　{  
　　( void )eMBPoll( );  
　　……  
　　}  
　　12、在port layer层的xMBPortSerialInit函数中，需要根据传入的波特率、奇偶校验、数据位长度设置来配置Uart0及其使用的定时器。  
　　13、在port layer层的vMBPortSerialEnable函数中配置接收和发送使能，由于在单片机的寄存器SCON0中只有接收使能控制位REN0，而没有发送使能控制位，所以在portserial.c文件中又定义了一个TxEnable变量，用来表示发送的使能状态。若同时关闭接收和发送，则要关闭Uart0中断，即让ES0 = 0。  
　　14、eMBRTUInit函数中的变量usTimerT35\_50us代表如果50us进行一次Tick的话，T35超时的Tick次数。这个公式很重要：  
　　usTimerT35\_50us = ( 7UL \* 220000UL ) / ( 2UL \* ulBaudRate );  
　　函数xMBPortTimersInit要以变量usTimerT35\_50us为传入参数，对T35超时定时器进行设置。  
　　15、在mbrtu.c文件中定义了两个状态变量，一个是接收状态变量eRcvState，为eMBRcvState枚举类型，有4个状态，在使能ModBus协议栈后赋予STATE\_RX\_INIT，即初始状态；另一个是发送状态变量eSndState，为eMBSndState枚举类型，有两个状态，初始化为发送idle状态，即STATE\_TX\_IDLE。  
　　16、mb.c文件中的eMBState状态变量为枚举类型，代表设备的工作状态，有3种状态，分别是“未初始化”、 “使能”和“禁止”状态。调用完eMBInit 函数后要调用eMBEnable函数来使能ModBus协议栈，在其中将eMBState状态变量从“未初始化状态”变为“使能状态”，然后使能串口和打开T35定时器  
　　17、如果T35定时器超时并产生中断，则要调用xMBRTUTimerT35Expired函数，其内部是一个状态机转换的switch，根据当前接收状态来通过xMBPortEventPost发送事件通知，然后关闭T35定时器，并将当前接收状态设置为STATE\_RX\_IDLE。  
　　18、eMBException枚举型变量表示Exception的类型，共有10种Exception，在ModBus协议中有定义。  
　　19、在eMBPoll( )中，首先通过xMBPortEventGet函数取event,如果没有则退出，若有event的话便根据event类型进行相应处理。EV\_READY是在协议栈初始化后xMBRTUTimerT35Expired函数发出来的，表示startup完成；EV\_FRAME\_RECEIVED是xMBRTUTimerT35Expired函数在T35超时后发出的，表示已经收到了一帧，需要进行成帧处理，调用eMBRTUReceive函数；EV\_EXECUTE是在处理EV\_FRAME\_RECEIVED过程中最后一步，如果此帧的地址符合本机地址，则发出EV\_EXECUTE事件，进行应用层的处理。  
　　20、在eMBRTUReceive函数中首先查看帧大小是否符合要求，然后进行CRC校验。此函数的原型是：  
　　eMBRTUReceive( UCHAR \* pucRcvAddress, UCHAR \*\* pucFrame, USHORT \* pusLength )  
　第一个参数是为了返回帧中的地址，也就是帧中第一个字节；第二个传入的参数以后要当做数组来使用，所以用了指针的指针类型；第三个参数表示PDU的长度，也就是帧中除去地址字节和CRC校验字节后的长度。  
　　21、在eMBPoll( )中处理EV\_EXECUTE事件，首先从PDU中提取出FunctionCode，然后根据FunctionCode找到相应的处理函数。xMBFunctionHandler结构体类型变量xFuncHandlers中定义了各个FunctionCode对应的处理函数pxHandler，函数的第一个参数ucMBFrame是PDU的存储地址，第二个参数usLength返回PDU的长度。如果帧不是一个广播帧，则需要设备发出一个回复，如果前面有错误发生，则要回复一个错误报告帧。  
　　22、在Keil中程序需要使用大模式编译，否则会出现error c249: 'data': segment too large的错误。

　　23、若使用波特率为9600，则 t3.5= ( 11 \* 3.5 ) / 9600 = 4.01 ms。 不能使用8位模式的Timer，因为11.0590MHz主频在最大48分频后，最长的超时时间为1.11ms，不能满足T35的超时要求。

24、freeModbus 1.5库，在使用过程中发现了一个bug，即如果在PDU中发送的寄存器数据长度与要读写的寄存器的数量不符，只要CRC校验正确，freeModbus便不会检测出来。例如写多个寄存器命令中，标明写寄存器的数量为2，也就是后面接的数据长度为4，但随后的数据只为2个字节，即一个寄存器的数据，freeModbus不会发现此错误，同时会将后面的CRC校验值认作是写第二个寄存器的数据。这个bug实际上是通过Modbus调试精灵1.024的一个写多寄存器bug发现的。

**协议栈以及定时器初始化T35第一次超时—>eMBPoll STATE\_RX\_IDLE—>收到数据中断—>prvvUARTRxISR—>pxMBFrameCBByteReceived—>xMBRTUReceiveFSM接收数据 —> STATE\_RX\_RCV—>T35超时—> eMBPollEV\_FRAME\_RECEIVED（peMBFrameReceiveCur->eMBRTUReceive）提取完整数据帧—> eMBPoll case EV\_EXECUTE:xFuncHandlers[i].pxHandler（eMBRegInputCB）对接收的数据进行处理—> peMBFrameSendCur—>eMBRTUSend（&STATE\_RX\_IDLE）—>STATE\_TX\_XMIT**

**串口发送完成中断—> prvvUARTTxReadyISR—> FSMpxMBFrameCBTransmitterEmpty—>xMBRTUTransmitFSM（& STATE\_TX\_XMIT）—>xMBPortSerialPutByte—>发送数据。**