**USB-CAN通讯模块——用户编程说明USB** **-** **CAN** **Communication Module** **-** **User Programming** **Instructions**

2012.03201 2.03

用户如果只是利用If the user just uses USB-CAN通讯模块进行CAN总线通信测试，可以直接利用随本产品提供的EmbededDebug工具软件，接入CAN总线。The USB - CAN communication module performs CAN bus communication test, and can directly access the CAN bus by using the EmbededDebug tool software provided with this product . 如果用户打算开发自己产品的软件程序，可以使用我公司提供的相关API，简化通讯开发过程。If you plan to develop software programs for your own products, you can use the relevant APIs provided by our company to simplify the communication development process.

下面介绍CAN通讯相关的两个类。The following describes the two classes related to CAN communication. 如果只做CAN方面的收发操作，可以只关心RT\_CAN类，使用其中的SendCANMessage函数发送CAN消息，使用RecvCANMessage或者通过SetRxPackageCallBack设置接收回调函数的方式接收CAN消息即可；如果希望详细了解我公司的通讯协议，可结合本文档与源代码进行学习。If the only aspect of the CAN transceiver operation may concern only RT\_CAN class, wherein the sending CAN messages using SendCANMessage function, using CAN messages to or receive RecvCANMessage SetRxPackageCallBack arrangement received by the callback function; If you want to learn more about the communication I agreement, the present document can be combined with the source code to learn.

**11** **串口通讯说明Serial communication instructions**

1.1串口通讯数据包的封装格式1.1 Serial communication packet encapsulation format

通讯数据包的封装格式：FrameHead +Data+CheckSum+FrameTail，控制符为0xA5, FrameHead为连续的两个0xAA, FrameTail为连续的两个0x55，如果Data中含0xA5、0xAA、0x55（即特殊字符），则在发送该字符之前添加一个控制符0xA5。The encapsulation format of the communication data packet: FrameHead + Data + CheckSum + FrameTail , the control character is 0xA5, the FrameHead is two consecutive 0xAA, and the FrameTail is two consecutive 0x55 , if the Data contains 0xA5 , 0xAA , 0x55 (ie special characters) , then add a control character 0xA5 before sending the character . CheckSum为8位校验和，即Data的所有数据之和的低八位。CheckSum is an 8 -bit checksum, which is the lower eight bits of the sum of all data of Data .

1.2串口通讯操作类1.2 serial communication operation class

1.2.1类名称1.2.1 class name

串口操作类名称是：RT\_COM（见RT\_COM.cpp和RT\_COM.h）The serial port operation class name is: RT\_COM (see RT\_COM.cpp and RT\_COM.h )

1.2.2相关定义说明1.2.2 Description of related definitions

(1) typedef void (\*RxPackageCallBackFunc)(void \*pParam, const BYTE \* byBuf, DWORD dwLen);(1) typedef void (\*RxPackageCallBackFunc)(void \*pParam, const BYTE \* byBuf, DWORD dwLen);

说明：用于定义接收数据操作回调函数Description: Used to define the receive data operation callback function

              参数：pParam：用户参数，调用SetRxPackageCallBack时设置。Parameters: pParam : User parameter, set when calling SetRxPackageCallBack .

                               byBuf：接收到的数据。byBuf : Received data.

                               dwLen：接收到数据的长度d wLen : the length of the received data

              返回值：无Return value: none

1.2.3函数说明1.2.3 Function Description

注意：以下文档中出现的com为RT\_COM类实例化的对象。Note: com following documents appear to RT\_COM class instantiated objects.

(1) RT\_COM(DWORD dwMaxPackageSize = 4096);(1) RT\_COM (DWORD dwMaxPackageSize = 4096);

              说明：类的构造函数Description: The constructor of the class

              参数：MaxPackageSize：默认值为4096，用于初始化发送和接收缓冲。Parameters: MaxPackageSize : The default value is 4096 for initializing the send and receive buffers.

              返回值：无Return value: none

(2) static DWORD FindAll(DWORD \*dwDevName);(2) static DWORD FindAll(DWORD \*dwDevName);

说明：\*当传入参数为NULL时，返回设备的数量，可利用该数量初始化dwDevNameDescription: \* When the incoming parameter is NULL , the number of devices is returned, which can be used to initialize dwDevName

                            \*当传入参数不为NULL时，dwDevName传出所有设备名称\* When the incoming parameter is not NULL , dwDevName passes all device names

              参数：dwDevName（传出）：存放设备名称，可以为NULL。Parameters: dwDevName (outgoing): Stores the device name, which can be NULL .

              返回值：返回检查到的设备数量。Return Value: Returns the number of devices checked.

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*DWORD \*dwDev, dwDevNum;DWORD \*dwDev, dwDevNum;*

*//分配空间/ /* *allocate space*

*dwDevNum = RT\_ COM::FindAll(NULL);dwDevNum = RT\_ COM::FindAll(NULL);*

*dwDev = new DWORD[dwDevNum];dwDev = new DWORD[dwDevNum];*

*//列出虚拟COM/ /* *List virtual* *COM*

*RT\_ COM::FindAll(dwDev);RT\_ COM::FindAll(dwDev);*

*for (DWORD i=0; i<dwDevNum; i++)Fo* *r (DWORD i=0; i<dwDevNum; i++)*

*{{*

*printf("COM%d\n", dwDev[i]);Printf("COM%d\n", dwDev[i]);*

*}}*

(3) BOOL Open(DWORD dwPortNo, DWORD dwBaud);(3) BOOL Open (DWORD dwPortNo, DWORD dwBaud);

              说明：打开指定串口Description: Open the specified serial port

              参数：dwPortNo：串口号Parameters: dwPortNo : serial port number

                              dwBaud：指定串口波特率dwBaud : Specify the serial port baud rate

              返回值：如果成功打开，则返回TRUE；否则返回FALSE。Return Value: Returns TRUE if opened successfully ; otherwise returns FALSE .

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*if(com.Open (6, 921600))If(com.Open (6, 921600))*

*{{*

*//打开虚拟串口成功的操作/ /* *Open the virtual serial port successfully operated*

*}}*

(4) void Close();(4) void Close();

              说明：关闭当前打开的串口Description: Close the currently open serial port

              参数：无Parameters: None

              返回值：无Return value: none

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*if(com.Close ())If(com.Close ())*

*{{*

*//关闭虚拟串口成功的操作/ /* *Close the virtual serial port successfully operated*

*}}*

(5) void SetRxPackageCallBack(RxPackageCallBackFunc func, void \*pParam);(5) void SetRxPackageCallBack(RxPackageCallBackFunc func, void \* pParam );

              说明：设置接收数据的回调函数Description: Set the callback function to receive data

              参数：pParam：用户参数。Parameters: pParam : User parameter.

func：RxPackageCallBackFunc类型的函数指针，用于在接收到数据后进行相关处理。func: RxPackageCallBackFu nc type of function pointers for performing correlation processing after receiving the data.

              返回值：无Return value: none

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*定义回调函数：Define the callback function:*

*void RecvCallback(void \* pParam, const BYTE \* buf, DWORD len)Void RecvCallback(void \* pParam, const BYTE \* buf, DWORD len)*

*{{*

*//执行回调操作/ /* *Perform callback operation*

*}}*

*设置回调函数：Set the callback function:*

*com.SetRxPackageCallBack(RecvCallback, NULL);com.SetRxPackageCallBack(RecvCallback, NULL);*

(6) DWORD WritePackage(BYTE \* byBuf, DWORD dwLen);(6) DWORD WritePackage(BYTE \* byBuf, DWORD dwLen);

              说明：向模块端发送包含数据的包Description: Send the package containing the data to the module side.

              参数：byBuf：要发送的数据Parameters: byBuf : the data to send

                              dwLen：要发送数据的长度dwLen : the length of the data to be sent

              返回值：已发送的数据长度Return value: length of data sent

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*BYTE buf[30];BYTE buf[30];*

*if(com.WritePackage(buf, sizeof(buf)))If(com.WritePackage(buf, sizeof(buf)))*

*{{*

*//数据发送成功的操作/ /* *Data transmission successful operation*

*}}*

(7) BOOL IsOpened()(7) BOOL IsOpened()

              说明：判断当前串口是否已经打开Description: Determine whether the current serial port is open.

              参数：无Parameters: None

              返回值：返回当前串口是否已经打开。Return value: Returns whether the current serial port is already open.

*Example:Example:*

*#include "RT\_COM.h"#include "* *RT\_COM.h"*

*if(com.IsOpened())If(com.IsOpened())*

*{{*

*//串口已经打开时的操作//* *Operation when the serial port is already open*

*}}*

(8) DWORD GetBaudRate()(8) DWORD GetBaudRate()

              说明：获得当前串口的波特率Description: Obtain the baud rate of the current serial port.

              参数：无Parameters: None

              返回值：当前串口的波特率Return value: the baud rate of the current serial port

*Example:Example:*

*#include " RT\_COM.h"#include " RT\_COM.h"*

*DWORD baud;DWORD baud;*

*baud = com.GetBaudRate();Baud = com.GetBaudRate();*

(9) DWORD GetPortNo();(9) DWORD GetPortNo();

              说明：获得当前串口号Description: Get the current serial port number

              参数：无Parameters: None

              返回值：当前串口号Return value: current serial port number

*Example:Example:*

*#include "RT\_COM.h"#include "* *RT\_COM.h"*

*DWORD PortNO;DWORD PortNO;*

*PortNO = com.GetPortNo ();PortNO = com.GetPortNo ();*

1.3串口通讯库使用方式1.3 Serial communication library usage

(1)创建一个RT\_COM实例(1) Create an RT\_COM instance

(1)调用Open打开虚拟串口(1) Call Open to open the virtual serial port

(2)调用WritePackage函数往串口写数据(2) call the WritePackage function to write data to the serial port

(3)用户自定义RxPackageCallBackFunc类型数据接收回调函数，调用SetRxPackageCallBack函数注册该函数。(3) User-defined RxPackageCallBackFunc type data receiving callback function, call SetRxPackageCallBack function to register the function.

**2 CAN通讯说明2 CAN** **communication instructions**

2.1类名称2.1 class name

              CAN操作类名称是RT\_CAN，该类继承自RT\_COM类，重载了OnRecvPackage函数。The CAN operation class name is RT\_CAN . This class inherits from the RT\_COM class and overrides the OnRecvPackage function. （见RT\_CAN.cpp和RT\_CAN.h）(see RT\_CAN.cpp and RT\_CAN.h )

2.2 CAN的消息格式2.2 CAN message format

              CAN消息的格式如下：The format of the CAN message is as follows:

typedef struct {Typedef struct {

              DWORD id;DWORD id;                                           //消息ID，统一使用扩展帧，29bit/ / Message ID , unified use of extended frames, 29bit

              BYTE data[8];BYTE data[8];                             //数据域/ / Data field

              BYTE len;BYTE len;                             //数据域长度/ / Data field length

              BYTE ch;BYTE ch;                             //消息通道(0xff为配置通道)// Message channel (0xff is the configuration channel )

              BYTE format;BYTE format;               //消息格式(取值见相关定义中的CAN\_FORMAT)/ / message format ( see the value of CAN\_FORMAT in the relevant definition )

              BYTE type;BYTE type;                             //消息类型(取值见相关定义中的CAN\_FRAME)/ / message type ( see the value of CAN\_FRAME in the relevant definition )

} CAN\_msg;} CAN\_msg;

2.3相关定义2. 3 related definitions

(1) CAN\_FORMAT(1) CAN\_FORMAT

              说明：CAN消息帧格式Description: CAN message frame format

              成员：STANDARD\_FORMAT：标准帧Members: STANDARD\_FORMAT : Standard frame

                              EXTENDED\_FORMAT：扩展帧EXTENDED\_FORMAT : extended frame

(2) CAN\_FRAME(2) CAN\_FRAME

              说明：CAN消息帧类型Description: CAN message frame type

              成员：DATA\_FRAME：数据帧Members: DATA\_FRAME : data frame

                              REMOTE\_FRAME：远程帧REMOTE\_FRAME : Remote frame

(3) CAN\_ERROR(3) CAN\_ERROR

              成员：CAN \_ERR\_TRANS：传输错误Member: CAN \_ERR\_TRANS : Transmission error

CAN \_OK：没有错误CAN \_OK : no error

CAN \_ERR\_ACK：没有返回应答CAN \_ERR\_ACK : No response is returned

CAN \_ERR\_CHANNEL：CHANNEL错误CAN \_ERR\_CHANNEL : CHANNEL error

2.4函数说明2.4 function description

注意：以下文档中出现的can为RT\_CAN类实例化的对象。Note: you can appear in the following documents for RT\_CAN class instantiated objects.

(1) RT\_CAN()(1) RT\_CAN()

说明：初始化CAN操作类Description: Initialize the CAN operation class

参数：无Parameters: None

              返回值：无Return value: none

(2)CAN\_ERROR SetConfigValue(DWORD dwID, void \*pValueBuf, DWORD \*pdwLen, DWORD dwTimeOut);(2) CAN\_ERROR SetConfigValue(DWORD dwID, void \*pValueBuf, DWORD \*pdwLen, DWORD dwTimeOut);

说明：发送配置数据包Description: Send configuration packet

              参数：dwID:数据包IDParameters: dwID: packet ID

                              pValueBuf:数据pValueBuf: data

                              pdwLen:数据长度pdwLen: data length

                              dwTimeOut:超时时间dwTimeOut: timeout

              返回值：错误信息Return value: error message

*Example:Example:*

*#include " RT\_CAN.h"#include " RT\_CAN.h"*

*DWORD Len = sizeof(BYTE);DWORD Len = sizeof(BYTE);*

*BYTEby =1;BYTE* *by =* *1;*

*//发送自动重发的配置/ /* *Send* *automatic resend configuration*

*if(CAN.SetConfigValue(CAN\_ART\_ID, &If(* *CAN* *.SetConfigValue(CAN\_ART\_ID, &* *by, &Len,1000) != CAN\_OK)By* *, &Len,1000) != CAN\_OK)*

*{{*

*ShowMessageBox("配置失败！", MB\_ICONSTOP);ShowMessageBox("* *Configuration failed!* *", MB\_ICONSTOP);*

*}}*

(2)CAN\_ERRORGetConfigValue(DWORD dwID, void \*pValueBuf, DWORD \*pdwLen, DWORD dwTimeOut);(2) CAN\_ERROR GetConfigValue (DWORD dwID, void \*pValueBuf, DWORD \*pdwLen, DWORD dwTimeOut) ;

说明：发送配置数据包Description: Send configuration packet

              参数：dwID:数据包IDParameters: dwID: packet ID

                              pValueBuf:数据(传出)pValueBuf: data ( outgoing )

                              pdwLen:数据长度pdwLen: data length

                              dwTimeOut:超时时间dwTimeOut: timeout

              返回值：错误信息Return value: error message

*Example:Example:*

*#include " RT\_CAN.h"#include " RT\_CAN.h"*

*DWORD Len = sizeof(BYTE);DWORD Len = sizeof(BYTE);*

*BYTE by;BYTE by* *;*

*//获取自动重发的配置/ /* *Get* *the configuration of automatic retransmission*

*if(CAN.GetConfigValue(CAN\_ART\_ID, &If(* *CAN* *.* *G* *etConfigValue(CAN\_ART\_ID, &* *by, &Len,1000) != CAN\_OK)By* *, &Len,1000) != CAN\_OK)*

*{{*

*ShowMessageBox("配置失败！", MB\_ICONSTOP);ShowMessageBox("* *Configuration failed!* *", MB\_ICONSTOP);*

*}}*

(3)CAN\_ERRORSendCANMessage(CAN\_msg \*pMsg, DWORD dwTimeout = INFINITE);( 3 ) CAN\_ERROR SendCANMessage(CAN\_msg \*pMsg, DWORD dwTimeout = INFINITE);

说明：发送CAN消息Description: Send CAN message

              参数：pMsg:CAN消息Parameters: pMsg : CAN message

                              dwTimeout:发送超时时间（默认为无限超时）dwTimeout : Send timeout (default is infinite timeout)

              返回值：错误信息Return value: error message

*Example:Example:*

*#include " RT\_CAN.h"#include " RT\_CAN.h"*

*//发送CAN消息/ /* *Send* *CAN* *message*

*CAN\_msg msg;CAN\_msg msg;*

*int val = 1;Int val = 1;*

*msg.id =0x01223211;Msg.id =* *0x01223211;* *//配置消息ID/ /* *Configure the message* *ID*

*msg.len = 8;Msg.len = 8;* *//配置消息长度/ /* *Configure the message length*

*msg.format =EXTENDED\_FORMAT;Msg.format =* *EXTENDED\_FORMAT* *;* *//配置消息格式/ /* *Configure the message format*

*msg.type =DATA\_TYPE;Msg.type =* *DATA* *\_* *TYPE* *;* *//配置消息类型/ /* *Configure the message type*

*memcpy(&msg.data[0], &val, sizeof(int));Memcpy(* *&msg.data[0], &val, sizeof(* *int* *)* *);*

*CAN.SendCANMessage(&msg);CAN* *.SendCANMessage(&msg);*

(4)CAN\_ERRORRecvCANMessage(CAN\_msg \*pMsg, DWORD dwTimeout = INFINITE);(4) CAN\_ERROR RecvCANMessage (CAN\_msg \*pMsg, DWORD dwTimeout = INFINITE);

说明：发送CAN消息Description: Send CAN message

              参数：pMsg:CAN消息(传出)Parameters: pMsg : CAN message ( outgoing )

                              dwTimeout:发送超时时间（默认为无限超时）dwTimeout : Send timeout (default is infinite timeout)

              返回值：错误信息Return value: error message

*Example:Example:*

*#include " RT\_CAN.h"#include " RT\_CAN.h"*

*//发送CAN消息/ /* *Send* *CAN* *message*

*CAN\_msg msg;CAN\_msg msg;*

*msg.id =0x01223211;Msg.id =* *0x01223211;* *//配置消息ID/ /* *Configure the message* *ID*

*msg.len = 8;Msg.len = 8;* *//配置消息长度/ /* *Configure the message length*

*msg.format =EXTENDED\_FORMAT;Msg.format =EXTENDED\_FORMAT;* *//配置消息格式/ /* *Configure the message format*

*msg.type = REMOTE\_FORMAT;Msg.type = REMOTE\_FORMAT;* *//配置消息类型/ /* *Configure the message type*

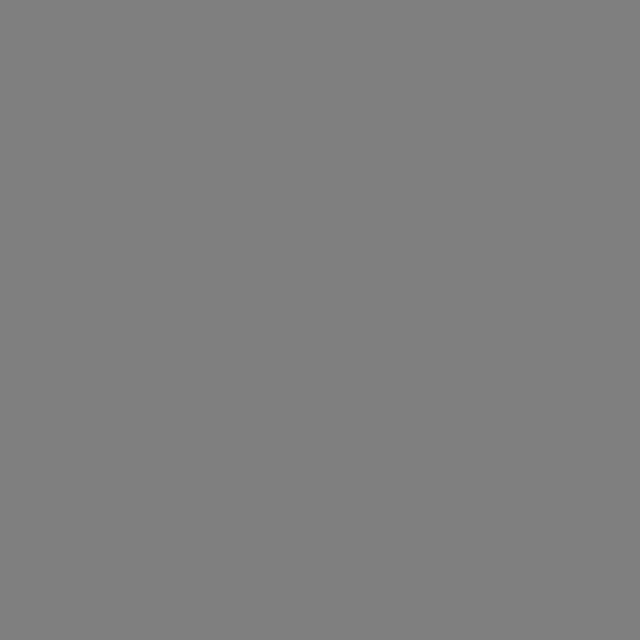
*CAN.RecvCANMessage(&msg);CAN* *.* *Recv* *CANMessage(&msg);*

2.5. CAN通讯库使用方式2.5. How to use the CAN communication library

(1)创建RT\_CAN的实例(1) Create an instance of RT\_CAN

(2)调用Open打开想要操作的对应于模块端的串口(2) Call Open to open the serial port corresponding to the module end that you want to operate.

(3)调用相应的函数设置和获取CAN的信息。(3) Call the corresponding function to set and get the CAN information.

https://www.gstatic.com/translate/infowindow/iws_n.pnghttps://www.gstatic.com/translate/infowindow/iws_n.pnghttps://www.gstatic.com/translate/infowindow/iws_w.pnghttps://www.gstatic.com/translate/infowindow/iws_e.pnghttps://www.gstatic.com/translate/infowindow/iws_s.pnghttps://www.gstatic.com/translate/infowindow/iws_s.png

https://www.gstatic.com/translate/infowindow/iw_n.pnghttps://www.gstatic.com/translate/infowindow/iw_n.pnghttps://www.gstatic.com/translate/infowindow/iw_w.pnghttps://www.gstatic.com/translate/infowindow/iw_e.pnghttps://www.gstatic.com/translate/infowindow/iw_s0.pnghttps://www.gstatic.com/translate/infowindow/iw_s0.png