TTCANopen

USB转CAN使用说明书

北京中铁航机电技术有限公司（监制）

目 录

1. 产品介绍

USB转CAN模块主要用于将CAN网络接口通过USB端口转换使其接入上位监控和管理系统，如：工业现场数据监控等。

其广泛应用于汽车电子，工业自动化监控，医疗仪器，机器人，纺织机械，安防系统，水文气象等领域的数据传输和监控。

TTCANopen USB转CAN产品分为三种类型。

第一类：学习型，主要面向青年学生和初学者，模块CAN端口为非隔离方式，集成多种功能，以裸板形式提供，可极大降低学习成本。

第二类：安全型，模块CAN端口采用ADM公司的磁隔离芯片ADM3053进行隔离，可有效的保护PC端设备。模块采用黑塑封装，功能同学习型。

第三类：工程型，主要面向工程应用，采用白塑封装，以单一定制功能模块提供（见附录1），模块出厂前全部通过高低温和老练等可靠性测试（测试指标要求详见附录2，也可根据用户要求添加测试内容）。

本产品为业界第一款全面支持TTCANopen应用层协议的USB转CAN专用模块。

了解TTCANopen应用层协议，请访问www.ttcanopen.com网站，本说明书不对协议内容进行讲解。见谅！

模块详细设计与测试参见：

https://www.ttcanopen.com/forum.php?mod=viewthread&tid=47

为了便于用户学习比较，学习型和安全型模块集成有常规通用USB转CAN工作模式。

2. 接口和主要指标

学习型: USB端口链接器：A型链接器

USB端口通讯：USB2.0全速（12Mbps），

USB通讯形式：虚拟串口

供电：采用USB端口+5V供电

核心ARM芯片：NXP LPC1752或1754

CAN端口链接器：3P3.81mm直脚接线端子

CAN收发器：TAJ1040（非隔离）

终端电阻：平衡式终端电阻（可选）

CAN通讯速率kbps：20、50、100、125、250、500、800、1000

外观尺寸：60×28×13 mm

模块集成：集成5种功能模块，

1).普通静听模块、

2).普通收发模块、

3).TTCANopen静听模块、

4).TTCANopen收发模块、

5).TTCANopen心跳器模块

安全型: USB端口链接器：A型链接器

USB端口通讯：USB2.0全速（12Mbps），

USB通讯形式：虚拟串口

供电：采用USB端口+5V供电

核心ARM芯片：NXP LPC1752或1754

CAN端口链接器：3P3.81mm直脚接线端子

CAN收发器：ADM3053（带磁隔离）

终端电阻：无

CAN通讯kbps:10、20、50、100、125、250、500、800、1000

外观尺寸：62×30×13 mm

封装：黑色塑封

模块集成：集成5种功能模块，

1).普通静听模块、

2).普通收发模块、

3).TTCANopen静听模块、

4).TTCANopen收发模块、

5).TTCANopen心跳器模块

工程型: USB端口链接器：A型链接器

USB端口通讯：USB2.0全速（12Mbps），

USB通讯形式：虚拟串口

供电：采用USB端口+5V供电

核心ARM芯片：NXP LPC1752或LPC1754

CAN端口链接器：3P3.81mm直脚接线端子

CAN收发器：ADM3053（带磁隔离）

终端电阻：无

CAN通讯kbps:10、20、50、100、125、250、500、800、1000

外观尺寸：62×30×13 mm

模块集成：单一定制功能，下列功能模块之一。

1).普通静听模块、

2).普通收发模块、

3).TTCANopen 静听模块、

4).TTCANopen 收发模块、

5).TTCANopen 心跳器模块

6）TTCANopen 数据回放器

可靠性测试：指标见附录 2

3. CAN 接口说明

模块 CAN 接口有三个接线端子，见图 1 所示，分别是 CAN_L（左）、CAN_H（中）总线信号和 GND（右）地线，通常总线信号线为双绞线，地线可选（多为屏蔽地）。

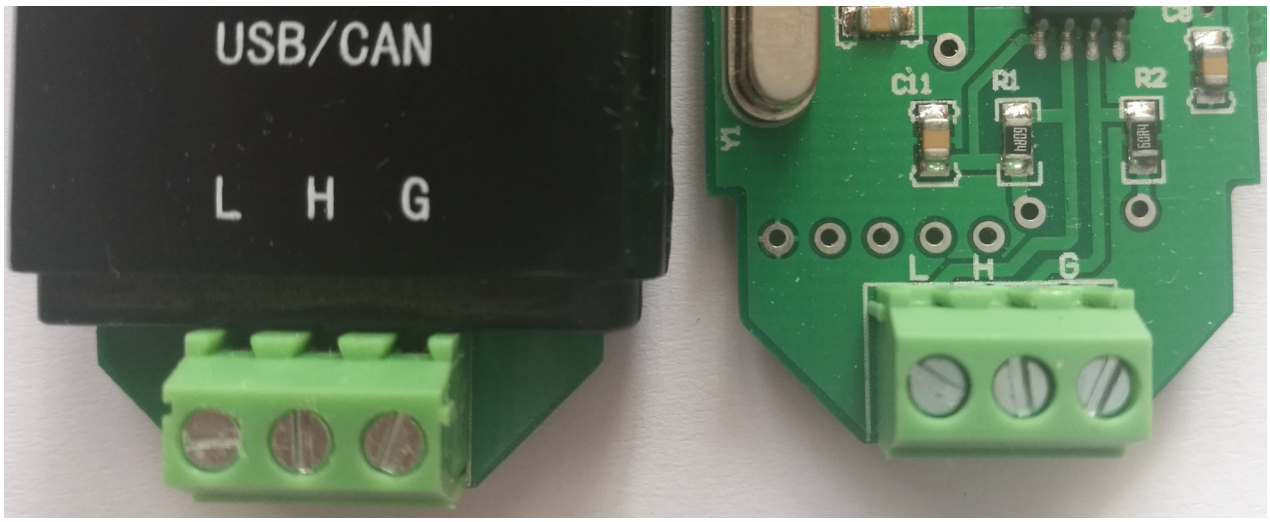


图 1 CAN 端口接线端子

4. 安装驱动（windows）

在使用模块时，PC 机上需安装 usbser.sys 驱动程序，实际上这个驱动程序是操作系统自带的，用户需要一个 xxx.inf 文件对“模块”进行说明，这里直接使用 NXP 虚拟串口配置文件 lpc17xx-vcom.inf 即可，用户可以从 internet 网上找到。注意：驱动分 32 位和 64 位两种，需要根据用户计算机和操作系统环境决定。

正确安装驱动后，当模块插入 PC 机 USB 接口后，模块背面当中指示灯会点亮。

打开 PC 机设备管理器，在“端口”一栏中可以看到模块的虚拟串口端口号，见图 2 所示。PC 机的多个 USB 端口，可以支持多个模块。

图 2 设备管理器界面

5. 模块串口基本配置

除了“串口号”外，模块对串口的其它配置项都进行了重新定义，在打开串口时，在串口接收窗口会显示配置帮助信息（字符方式，注：工程模块无此帮助）见图 3 所示。

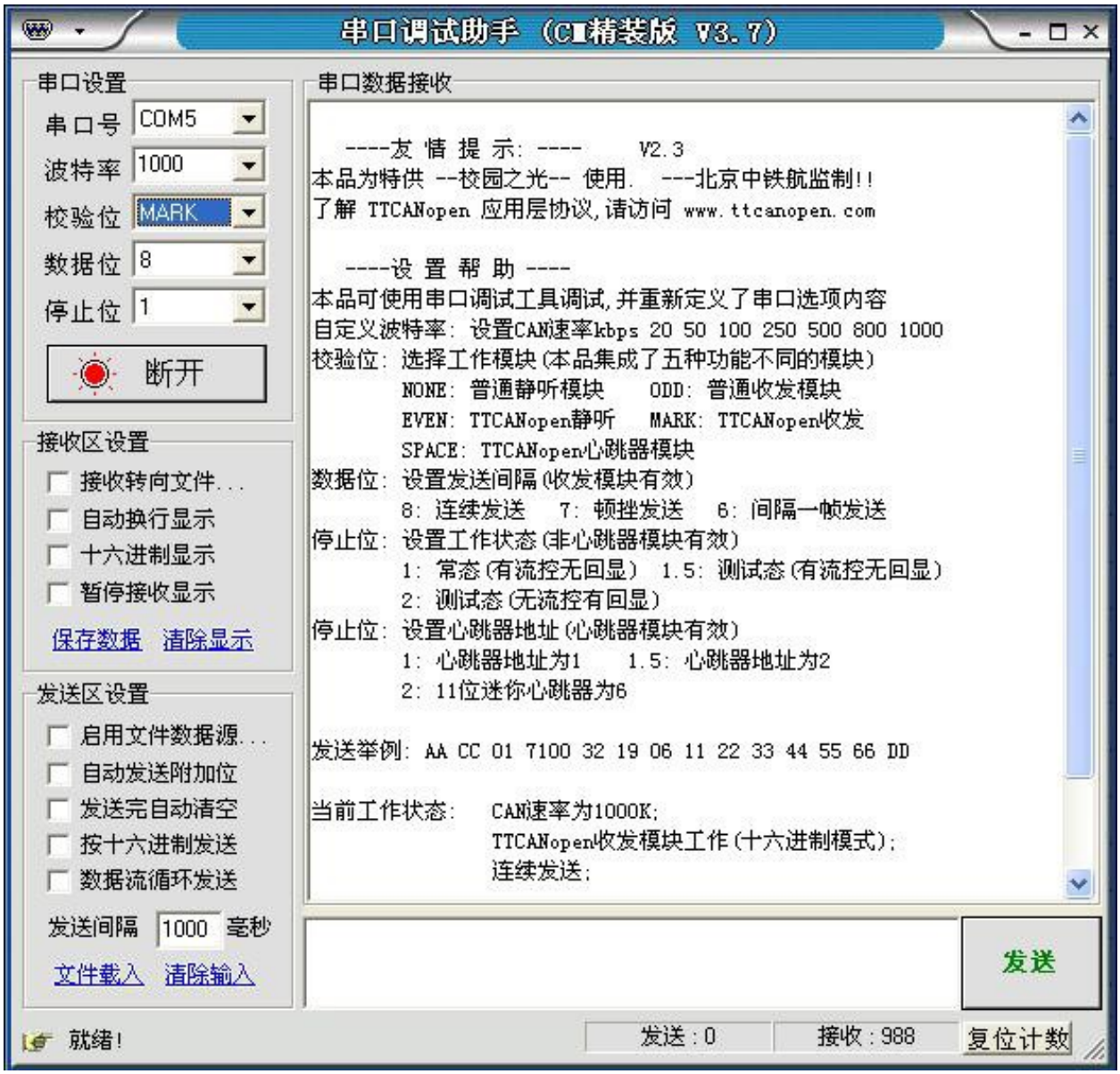


图 3 模块配置帮助信息

下面逐一进行介绍各项配置：

波特率: （通常使用自定义波特率方式）设置CAN总线通讯速率kbps，可输入10、20、50、100、125、250、500、800、和1000。

对于没有自定义波特率输入方式的串口调试工具，可以使用表1对应关系输入CAN总线通讯速率。

表1 串口波特率与CAN端口通讯速率的对应关系

序号	串口波特率	CAN端口速率kbps
1	9600	10
2	14400	20
3	19200	50
4	38400	100
5	56000	125
6	57600	250
7	115200	500
8	128000	800
9	256000	1000

校验位: 选择工作模块（注：产品集成了5种功能模块）

NONE： 普通静听模块

ODD： 普通收发模块

EVEN： TTCANopen 静听模块

MARK： TTCANopen 收发模块

SPACE：TTCANopen 心跳器模块

数据位: 选择发送指令间隔（对发送指令有效）

8：连续发送

7：顿挫发送

6：间隔一帧发送

5：间隔两帧发送

停止位: 选择 TTCANopen 心跳器地址（对心跳器有效）

1： 心跳器地址为 1

1.5：心跳器地址为 2

2： 11 位迷你心跳器地址为 6（通常不用）

6. 普通静听模块

当串口配置为：**校验位:** NONE；**数据位:** 8；**停止位:** 1 时模块处于普通静听工作状态。

静听模块，是用户观察 CAN 总线数据和状态的利器，它使用字符形式直接显示当前总线上的指令和状态。

模块不参与总线活动，只静听总线指令和进行总线出错判断，将总线上的指令和出错信息以字符串的形式传送到虚拟串口数据接收区，支持 CAN1.0A 和 CAN2.0B 协议，支持 4 种帧的显示，其格式如下：

1）显示扩展数据帧指令格式：

帧计数	接收时标	帧标识	ID 字	DLC	帧标识	DLC 个字节数据
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	[	00000000 至 1FBFFFFF	0 至 8	]	xx ～ xx

2）显示扩展遥控帧指令格式：

帧计数	接收时标	帧标识	ID 字	DLC	帧标识
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	<	00000000 至 1FBFFFFF	0 至 8	>

3）显示标准数据帧指令格式：

帧计数	接收时标	帧标识	ID 字	DLC	帧标识	DLC 个字节数据
00000000至FFFFFFFF	时:分:秒:毫秒.微妙	[	0000至07EF	0至8	]	xx ～ xx

4）显示标准遥控帧指令格式：

帧计数	接收时标	帧标识	ID 字	DLC	帧标识
00000000至FFFFFFFF	时:分:秒:毫秒.微妙	<	0000至07EF	0至8	>

显示帧举例：

1）扩展数据帧：

00000012 23:18:56:566.211 [01234567 3] 11 22 33

2）扩展遥控帧：

00000013 23:18:56:566.623 < 01234567 8 >

3）标准数据帧：

00000014 23:18:56:567.135 [0012 4] 11 22 33 44

4）标准遥控帧：

00000015 23:18:56:567.376 < 0012 4 >

上例中:

第一个字符串“00000012”：

为接收帧计数，计数范围 0x0～0xFFFFFFFF;

第二个字符串“23:18:56:566.211”：

为接收时标，分辨率为1微秒;

第三个[扩起来]的字符串“[01234567 4]”：

为帧ID和DLC，(ID为32位十六进制数,低29位有效);

<尖括号>则表述遥控帧;

第四个字符串“11 22 33 ”：

为数据帧中0～8个字节数据内容。

（注：对于标准帧，ID为16位十六进制数，低11位有效）

图4为在串口调试工具接收窗口（字符形式）观察到的扩展数据帧举例，

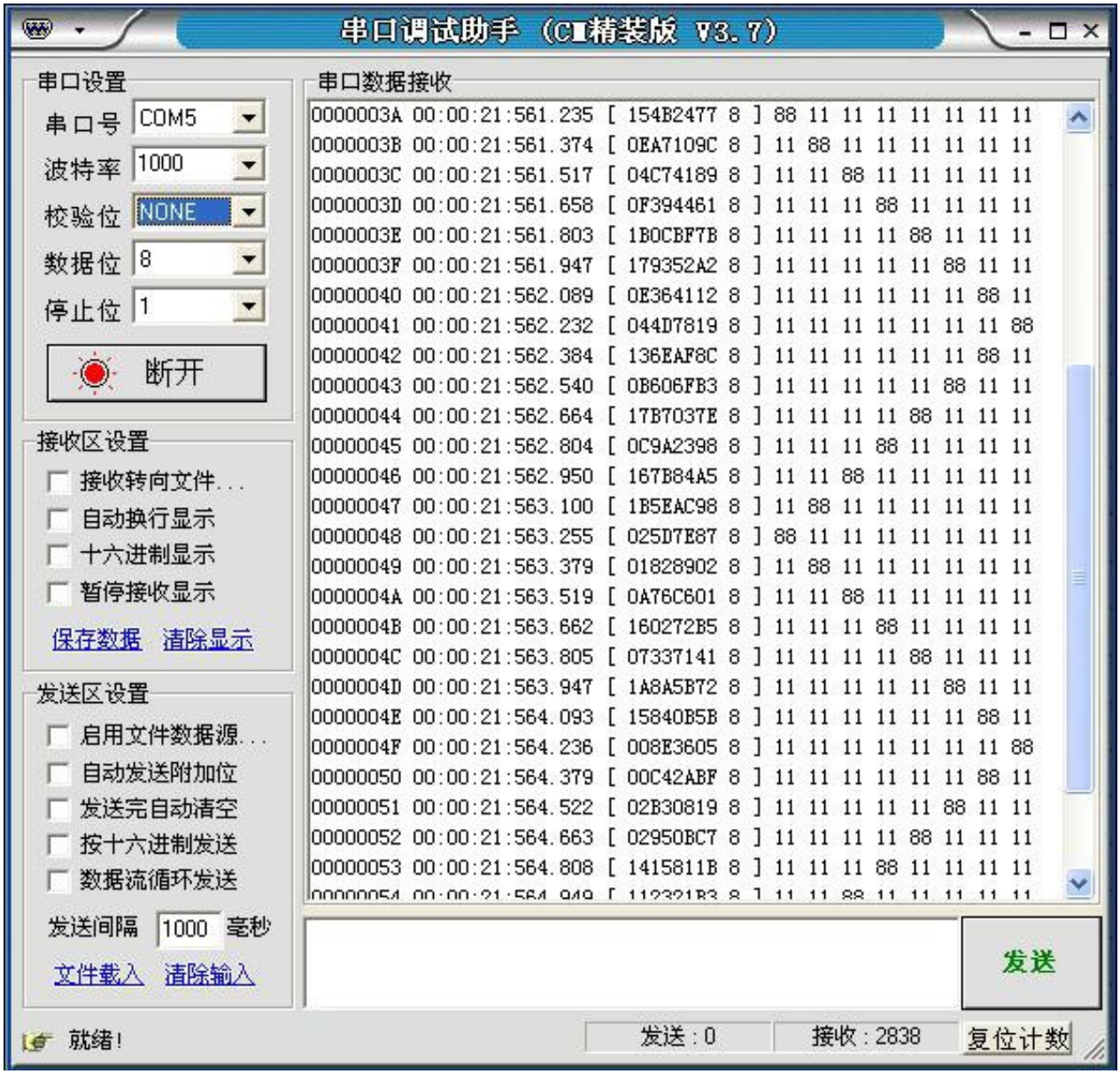


图 4 普通静听模块接收显示举例

7. 普通收发模块

当串口配置为：**校验位:** ODD；**数据位:** 8、7、6；**停止位:** 1 时，模块处于普通收发工作状态，USB 端虚拟串口支持流控功能。

收发模块，是 PC 端应用程序与 CAN 网络数据交互的工具。

模块将接收到的 CAN 总线指令按格式要求转换为十六进制字节流发送到虚拟串口数据接收区，将虚拟串口发送区字节流转换为 CAN 指令发送到总线上。

模块支持 4 种指令的发送和接收。

发送指令格式（十六进制）：

1）发送扩展数据帧指令格式：

帧头	帧标识	ID 字	DLC	DLC 个字节数据	帧尾
AA	CC	00000000 至 1FBFFFFF	00 至 08	xx ～ xx	DD

2）发送扩展遥控帧指令格式：

帧头	帧标识	ID 字	DLC	帧尾
AA	FF	00000000 至 1FBFFFFF	00 至 08	DD

3）发送标准数据帧指令格式：

帧头	帧标识	ID 字	DLC	DLC 个字节数据	帧尾
AA	55	0000 至 07EF	00 至 08	xx ～ xx	DD

4）发送标准遥控帧指令格式（十六进制）：

帧头	帧标识	ID 字	DLC	帧尾
AA	77	0000 至 07EF	00 至 08	DD

在串口调试工具中发送指令举例：

1）扩展数据帧：

AA CC 01234567 08 11 22 33 44 55 66 77 88 DD

2）扩展遥控帧：

AA FF 01234567 08 DD

3）标准数据帧：

AA 55 0012 08 11 22 33 44 55 66 77 88 DD

4）标准遥控帧：

AA 77 0012 08 DD

接收指令格式（十六进制）：

1）接收扩展数据帧指令格式：

帧头	接收时标	帧标识	ID 字	DLC	DLC 个字节数据	帧尾
BB	tt tt tt tt	CC	00000000 至 1FBFFFFF	00 至 08	xx ～ xx	EE

2）接收扩展遥控帧指令格式：

帧头	接收时标	帧标识	ID 字	DLC	帧尾
BB	tt tt tt tt	FF	00000000 至 1FBFFFFF	00 至 08	EE

3）接收标准数据帧指令格式：

帧头	接收时标	帧标识	ID 字	DLC	DLC 个字节数据	帧尾
BB	tt tt tt tt	55	0000 至 07EF	00 至 08	xx ～ xx	EE

4）接收标准遥控帧指令格式：

帧头	接收时标	帧标识	ID 字	DLC	帧尾
BB	tt tt tt tt	77	0000 至 07EF	00 至 08	EE

注：tt tt tt tt为32位接收时间计数（0.1ms），其低字节在前，高字节在后；而ID字是高字节在前，低字节在后，解帧时需特别留意。另外，数据指令帧的帧长，是根据数据场传输字节数量改变的，即：变帧长，这是协议以后兼容CAN-FD所需。

在串口调试工具中接收指令举例：

1）扩展数据帧：

BB 15 00 00 00 CC 01 23 45 67 04 11 22 33 44 EE

2）扩展遥控帧：

BB 16 00 00 00 FF 01 23 45 67 08 EE

3）标准数据帧：

BB 18 00 00 00 55 00 12 06 11 22 33 44 55 66 EE

4）标准遥控帧：

BB 19 00 00 00 77 00 12 08 EE

图5 为普通收发状态，串口调试工具示例

图 5 普通收发状态串口调试工具演示

8. TTCANopen 静听模块

当串口配置为：**校验位: EVEN；数据位:** 8；**停止位:** 1 时模块处于 TTCANopen 静听工作状态。

静听模块，是用户观察 CAN 总线数据和状态的利器，它使用字符形式直接显示当前总线上的 TTCANopen 指令和状态。

模块不参与总线活动，只静听总线指令和总线出错判断，将总线上的指令和出错信息以字符串的形式传送到虚拟串口数据接收区，支持 CAN1.0A 和 CAN2.0B 协议，支持 4 种 TTCANopen 帧的显示，格式如下：

1）显示 TTCANopen 扩展数据帧指令格式：

帧计数	接收时标	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	帧标识
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	[	0 或 1	0000 至 FBFF	00 至 7F	00 至 1F	00 至 08	]

DLC 个字节数据
xx ～ xx

2）显示 TTCANopen 扩展遥控帧指令格式：

帧计数	接收时标	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	帧标识
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	<	0 或 1	0000 至 FBFF	00 至 7F	00 至 1F	00 至 08	>

3）显示 miniTTCANopen 标准数据帧指令格式：

帧计数	接收时标	帧标识	寄存器地址	段地址	DLC	帧标识	DLC 个字节数据
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	[	00 至 FD	00 至 07	00 至 08	]	xx ～ xx

4）显示 miniTTCANopen 标准遥控帧指令格式：

帧计数	接收时标	帧标识	寄存器地址	段地址	DLC	帧标识
00000000 至 FFFFFFFF	时:分:秒:毫秒.微妙	<	00 至 FD	00 至 07	00 至 08	>

显示帧举例：

1）TTCANopen扩展数据帧：

00000012 23:18:56:566.211 [1 7100 32 19 2] 11 22

2）TTCANopen扩展遥控帧：

00000013 23:18:56:566.623 < 1 7100 32 19 8 >

3）miniTTCANopen标准数据帧：

00000014 23:18:56:567.135 [35 3 4] 11 22 33 44

4）niniTTCANopen标准遥控帧：

00000015 23:18:56:567.376 < 35 3 8 >

上例中:

第一个字符串“00000012”：

为接收帧计数，计数范围 0x0～0xFFFFFFFF;

第二个字符串“23:18:56:566.211”：

为接收时标，分辨率为1微秒;

第三个[扩起来]的字符串“[1 7100 32 19 2]”：

分别为 1位优先级、16位寄存器基地址、7位设备地址、5位指令和DLC； <尖括号>则表述遥控帧;

第四个字符串“11 22”：

为数据帧中0～8个字节数据内容。

（注：对于标准帧，支持 miniTTCANopen，尚未推出。）

图 6 为在串口调试工具接收窗口（字符形式）观察到的扩展数据帧举例，

图 6 TTCANopen 静听模块接收显示举例

9. TTCANopen 收发模块

当串口配置为：**校验位: MARK；数据位:** 8、7、6；**停止位:** 1 时，模块处于 TTCANopen 收发工作状态。USB 端虚拟串口支持流控功能。

收发模块，是 PC 端应用程序与 CAN 网络数据交互的工具。

模块将接收到的 CAN 总线指令按 TTCANopen 格式要求转换为十六进制字节流发送到虚拟串口数据接收区，将虚拟串口发送区字节流转换为 CAN 指令发送到总线上。

模块支持 4 种 TTCANopen 指令帧的发送和接收。

发送指令格式（十六进制）：

1）发送 TTCANopen 扩展数据帧指令格式：启用

帧头	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	DLC 个字节数据	帧尾
AA	CC	00 或 01	0000 至 FBFF	00 至 7F	00 至 1F	00 至 08	xx ～ xx	DD

2）发送 TTCANopen 扩展遥控帧指令格式：未用

帧头	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	帧尾
AA	FF	00 或 01	0000 至 FBFF	00 至 7F	00 至 1F	00 至 08	DD

3）发送 miniTTCANopen 标准数据帧指令格式：未用

帧头	帧标识	寄存器地址	段地址	DLC	DLC 个字节数据	帧尾
AA	55	00 至 FD	00 至 07	00 至 08	xx ～ xx	DD

4）发送 miniTTCANopen 标准遥控帧指令格式：未用

帧头	帧标识	寄存器地址	段地址	DLC	帧尾
AA	77	00 至 FD	00 至 07	00 至 08	DD

注：寄存器基地址为 16 位十六进制数，高地址在前，低地址在后。支持标准帧的 miniTTCANopen 协议格式尚未推出。

在串口调试工具中发送指令举例：

1）扩展数据帧：（注：TTCANopen 协议只使用扩展数据帧）

AA CC 01 7100 32 19 08 11 22 33 44 55 66 77 88 DD

2）扩展遥控帧：（注：未启用）

AA FF 01 7100 32 19 08 DD

3）标准数据帧：（注：未启用）

AA 55 35 03 08 11 22 33 44 55 66 77 88 DD

4）标准遥控帧：（注：未启用）

AA 77 35 03 08 DD

接收指令格式（十六进制）：

1）接收TTCANopen扩展数据帧指令格式：启用

帧头	接收时标	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	DLC个字节数据	帧尾
BB	tt tt tt tt	CC	00或01	0000至FBFF	00至7F	00至1F	00至08	xx ～ xx	EE

2）接收TTCANopen扩展遥控帧指令格式：未用

帧头	接收时标	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	帧尾
BB	tt tt tt tt	FF	00或01	0000至FBFF	00至3F	00至1F	00至08	EE

3）接收miniTTCANopen标准数据帧指令格式：未用

帧头	接收时标	帧标识	寄存器地址	段地址	DLC	DLC个字节数据	帧尾
BB	tt tt tt tt	55	00至FD	00至07	00至08	xx ～ xx	EE

4）接收miniTTCANopen标准遥控帧指令格式：未用

帧头	接收时标	帧标识	寄存器地址	段地址	DLC	帧尾
BB	tt tt tt tt	77	00 至 FD	00 至 07	00 至 08	EE

注：tt tt tt tt为32位接收时间计数（0.1ms），其低字节在前，高字节在后；而ID字是高字节在前，低字节在后，解帧时需特别留意。另外，数据指令帧的帧长，是根据数据场传输字节数量改变的，即：变帧长，这是协议以后兼容CAN-FD所需。

在串口调试工具中接收指令举例：

1）TTCANopen扩展数据帧：

BB 15 23 00 00 CC 01 71 00 32 19 03 11 22 33 EE

2）TTCANopen扩展遥控帧：（注：未启用）

BB 16 23 00 00 FF 01 71 00 32 19 08 EE

3）miniTTCANopen标准数据帧：（注：未启用）

BB 18 23 00 00 55 35 03 06 11 22 33 44 55 66 EE

4）miniTTCANopen标准遥控帧：（注：未启用）

BB 19 23 00 00 77 35 03 08 EE

图7 为TTCANopen收发状态，串口调试工具演示

图 7 TTCANopen 收发状态串口调试工具演示

10. TTCANopen 心跳器模块

当串口配置为：**校验位: SPACE；数据位: 8；停止位:** 1、1.5、2 时，模块处于 TTCANopen 心跳器工作状态。

心跳器模块，是为 TTCANopen 网络提供系统心跳的设备，网络上的设备收到系统心跳会将其本地时间与心跳时间同步，完成总线时间统一。

心跳器，完成一次心跳分为两帧发送，第一帧为时间预报帧，第二帧为心跳同步帧，两帧通常间隔 4～5 帧时间。

时间预报帧发送格式：

帧头	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	8 个字节数据				帧尾
							4 字节预报时间	1 字节	1 字节	心跳周期	
AA	CC	00	0A02	01 至 03	19	08	00000000 至 337F97FF	CAN 接收错误计数	CAN 发送错误计数	(ms) 000A 至 FFFF	DD

“预报时间”是下一个心跳同步帧的结束时刻，以 0.1ms 为记时单位，0x00000000～0x337F97FF（863999999）是一天中 0.1ms 的计数个数，即：所谓 TTCANopen 时间。

“CAN 接收错误计数”和“CAN 发送错误计数”是 CAN 端

口收发错误寄存器中的计数值，是心跳器CAN端口的通讯质量的表征。

“心跳周期”以ms为计数单位。默认为1（0x3E8）秒。

心跳同步帧发送格式：

帧头	帧标识	优先级	寄存器基地址	设备地址	指令	DLC	帧尾
AA	CC	00	0A03	01至03	19	00	DD

心跳同步帧为固定帧，帧发送结束时刻为时标，网络上的设备，接收此帧后立即将其本地时间值同步到预报时间上，完成与系统时间的同步。

在同一网络中只能有一个心跳器工作，其它心跳器处于备机状态，并将本地时间自动同步到主心跳器上，当检查到主心跳器十次没有发出正常心跳时，备机会自动启动替换主心跳器。

心跳器每发送一次心跳都会在USB端仿真串口接收窗口显示当前发送的时间和CAN端口的收发错误计数。见图8所示。

图 8 心跳器显示窗口

用户可以通过 USB 仿真串口的发送端，控制心跳器的启动和停止，也可以设置心跳器的起始时间和心跳周期。

设置串口调试工具的发送窗口为字符方式，然后输入？

问号并发送，这时在接收窗口会出现帮助信息如图 9 所示。

图 9 心跳器帮助信息

用户可以根据该提示在发送窗口进行所需操作，出错时，会有相关提示。

如果用户需要将心跳周期改为2秒，就可以在发送窗口输入：L2000回车，然后点“发送键”；

如果用户需要将起始时间改为12:12:12:000.0， 可输入：T12:12:12:000.0回车，然后发送。

如果用户需求启动或停止心跳，可输入：Q， 然后发送。

关于心跳更多内容详见：《CAN应用层协议TTCANopen开发与实践》第六章 “步调一致”。

11. 模块 TTCANopen 特性（只针对 TTCANopen 状态）

1）模块遵从“TTCANopen 应用层协议 CAN 标识的分配方法”，见附录 3；

2）模块遵从“TTCANopen 应用层协议设备内部多条指令的发送规则”，见附录 4；

3）模块接受系统心跳，并与之同步，图 10 为 TTCANopen 静听模块的时间同步过程。

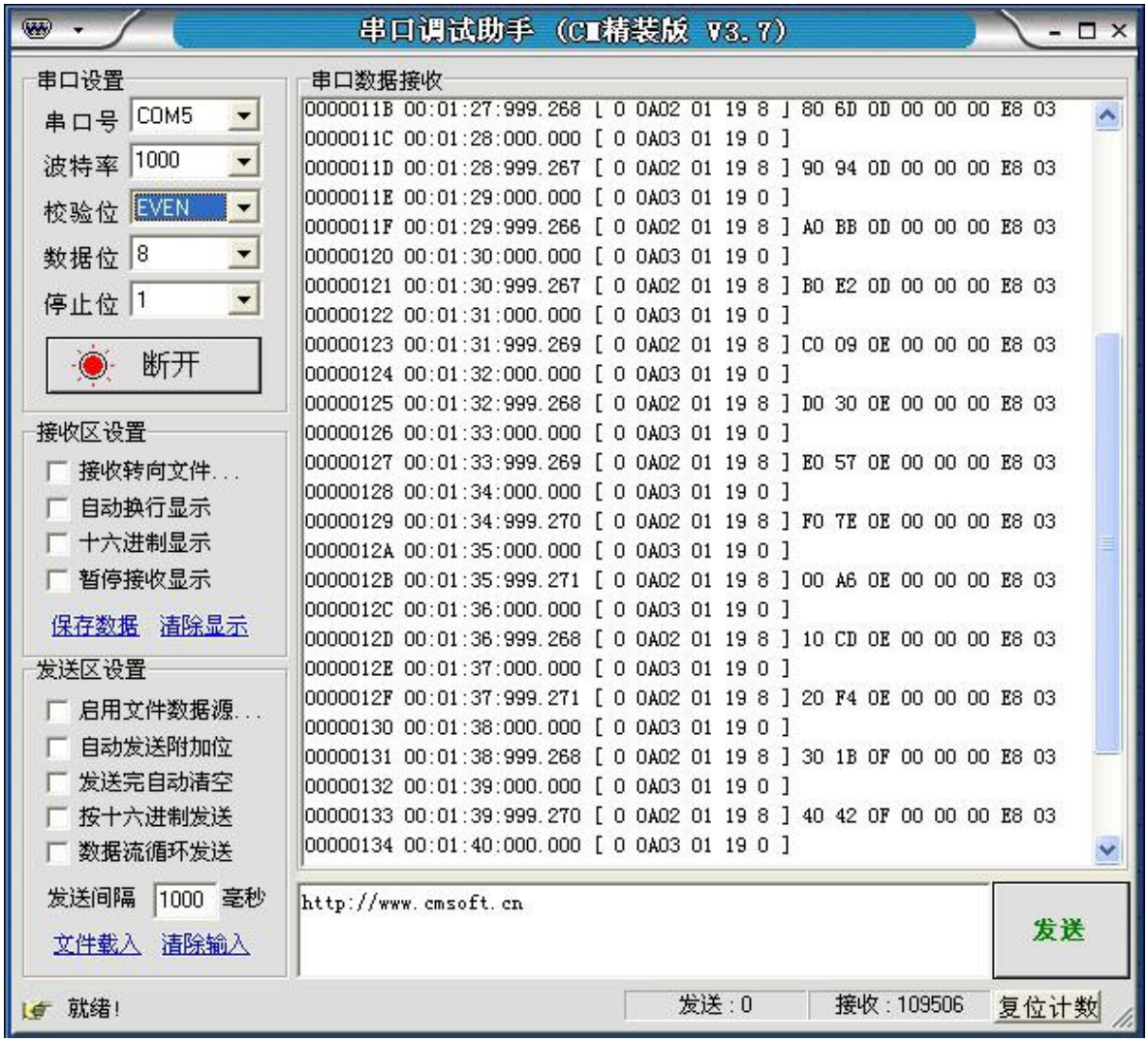


图 10 TTCANopen 静听模块与系统时间同步

附录 1：定制工程型模块

为了提高工程型模块在使用过程中的可靠性，减少配置错误，工程型模块不再集成多种功能，也不提供模块帮助信息，每种模块只包含单一定制功能和配置参数，用户在使用工程型模块时，只需设置好端口号即可。

用户需对各种模块填写相应的定制单。

1）普通静听模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000

2）普通收发模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000
2	发送间隔	连续、顿挫、间隔帧数（1～3）
3	流控	有、没有

3）TTCANopen 静听模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000

4）TTCANopen 收发模块定制内容：

序号	名称	可选参数内容（单选）

1	通讯速率（kbps）	20、50、100、250、500、1000
2	发送间隔	连续、顿挫、间隔帧数（1～3）
3	流控	有、没有

5）简易TTCANopen心跳器模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000
2	心跳周期（ms）	50～50000

6）高稳TTCANopen心跳器模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000
2	心跳周期（ms）	50～50000
3	外时统输入信号	秒脉冲、B000码、GPS、北斗
4	本地晶体	温补、恒温
5	外壳定制尺寸要求	

7）TTCANopen数据回放模块定制内容：

序号	名称	可选参数内容（单选）
1	通讯速率（kbps）	20、50、100、250、500、1000
2	按系统时间回放	是、否
3	加速回放规则	有、无

附录 2：定制工程型模块可靠性实验

1）高温实验

+50±2℃持续工作 72 小时。

2）低温实验

-20±2℃持续工作 48 小时。

3）老练实验

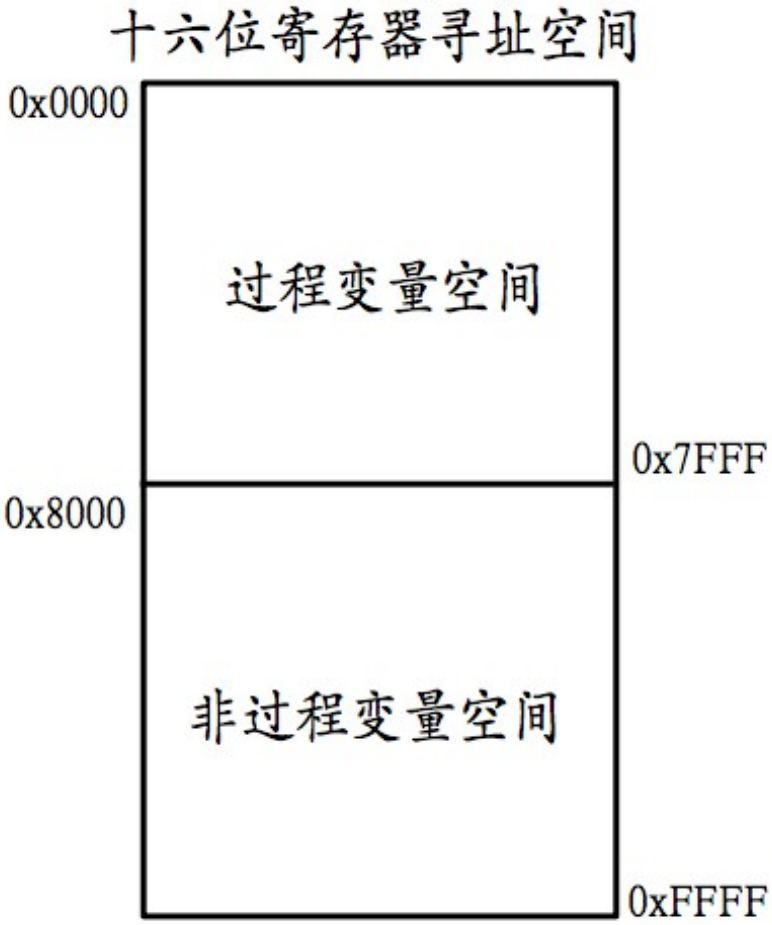
总折合老练时间不小于 200 小时。

（注：用户可根据应用条件，添加相关测试。）

附录 3： TTCANopen 应用层协议

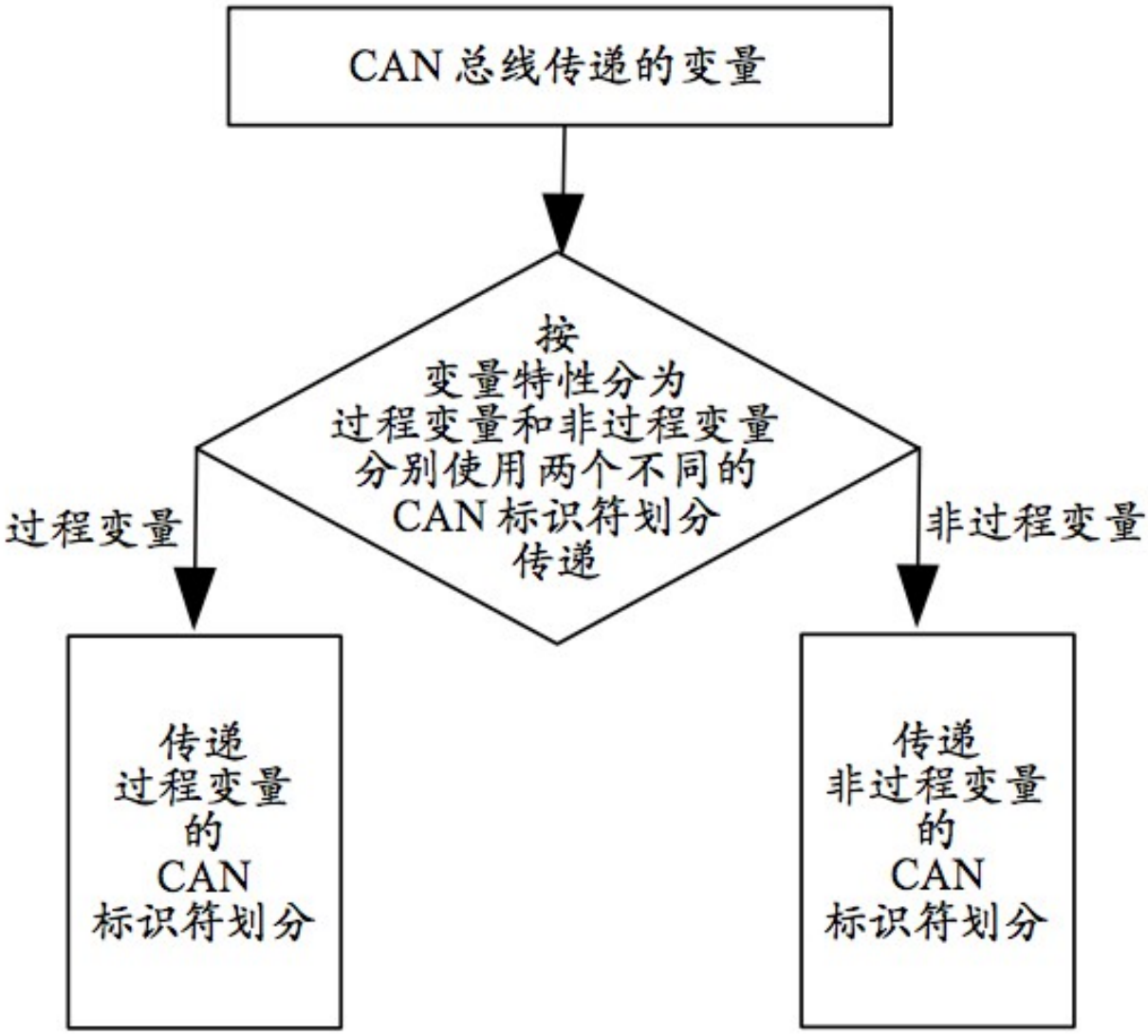
CAN 标识的分配方法

TTCANopen 应用层协议以连续寄存器空间为信息载体，并将其十六位寄存器寻址空间平分为两部分，见附图 1 所示。其中：0x0000～0x7FFF 用于存放、传递过程变量；0x8000～0xFFFF 用于存放、传递非过程变量。



附图 1 寄存器空间的划分

TTCANopen 应用层协议根据 CAN 总线传递变量特性的不同，即过程变量或非过程变量，使用两个不同的 CAN 标识符划分分别传递上述两种变量，见附图 2 所示，从而为两种变量提供了不同的总线竞争途径，并为应用层协议实施生产者-消费者模型和主机-从机模型提供了明确的对象指向。



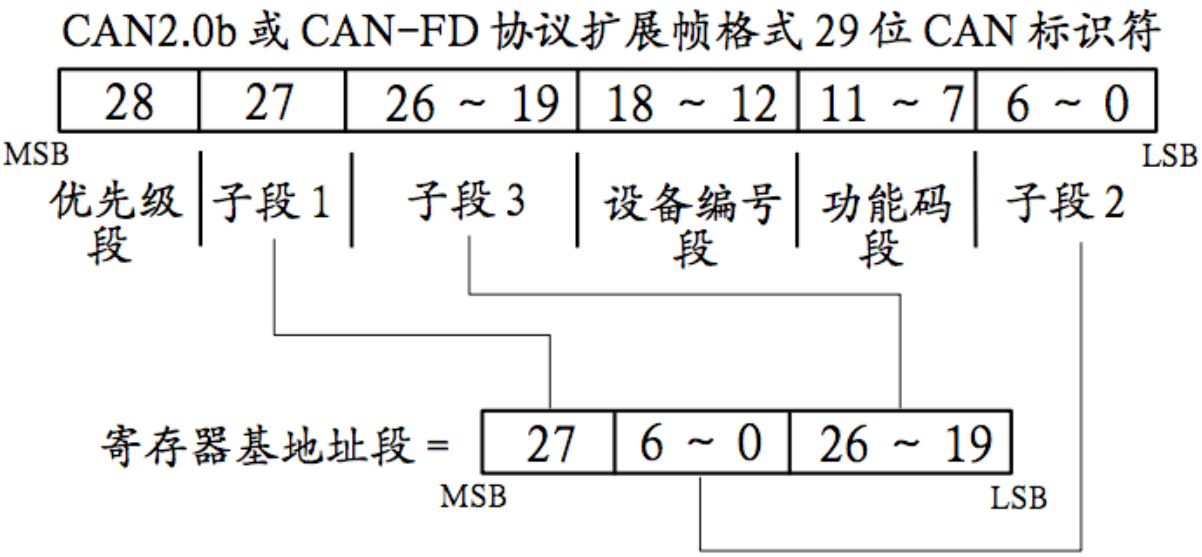
附图 2 使用不同的 CAN 标识符划分传递两种不同的变量

传递过程变量的CAN标识符划分，采用了在CAN标识符中抽取多个位片段，并重新排列拼接组合的方式，构成应用层传递过程变量的寄存器基地址段，从而可以在过程变量寄存器基地址空间构成多个等价竞争层，很好的解决了变量分层管理与总线竞争分配不平衡的矛盾。

传递过程变量的CAN标识符划分，见附图3所示，其将CAN2.0b或CAN-FD协议扩展帧格式中29位CAN标识符划分为六段，即:优先级段、子段1、子段3、设备编号段、功能码段和子段2，其中:子段1、子段2和子段3排列拼接组合构成寄存器基地址段，各段组成说明如下:

优先级段: 由CAN标识符的最高位第28位组成;
子段1: 由CAN标识符的第27位组成;
子段3: 由CAN标识符的第26～19位组成;
设备编号段: 由CAN标识符的第18～12位组成;
功能码段: 由CAN标识符的第11～7位组成;
子段 2: 由CAN标识符的第6～0位组成;

寄存器基地址段:由子段1、子段2和子段3排列拼接组合而成，即:由CAN标识符的第27位、6～0 位和 26～19位组成十六位的寄存器基地址段。



附图 3 传递过程变量的 CAN 标识符划分

当 CAN 标识符的第 27 位(即:寄存器基地址段的最高位)为 0 时，寄存器基地址空间指向过程变量空间(即:0x0000～0x7FFF)，由于十六位寄存器基地址段的低八位(26～19 位)处于 CAN 标识符的较高端 MSB，而其次高七位(6～0 位)处于 CAN 标识符的低端 LSB，由此过程变量寄存器空间变量的竞争能力由其十六位寄存器基地址段的低八位决定，从而在过程变量寄存器地址空间形成了 128 个等价竞争层，每层包含 256 个地址空间，在每层相同位置上的过程变量具有相同的总线竞争能力，在同层中，低地址变量的竞争能力优于高地址变量。

传递非过程变量的 CAN 标识符划分，见附图 4 所示，其将 CAN2.0b 或 CAN-FD 协议扩展帧格式中 29 位 CAN 标识符划分

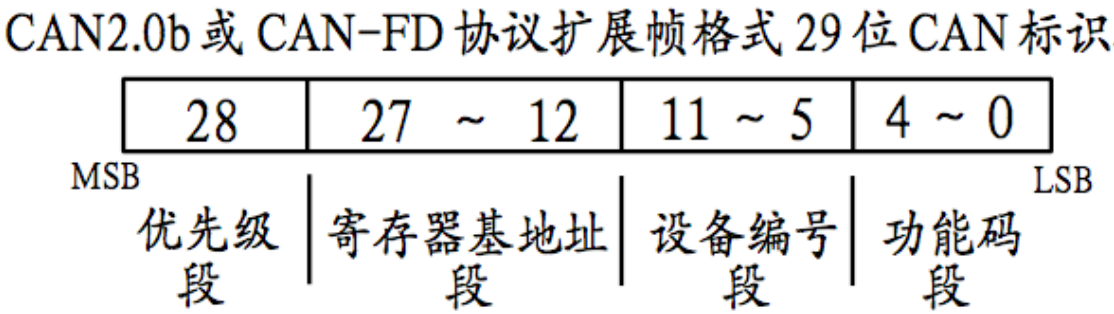
为四段，即:优先级段、寄存器基地址段、设备编号段和功能码段，各段组成说明如下:

优先级段: 由 CAN 标识符的最高位第 28 位组成;

寄存器基地址段:由 CAN 标识符第 27～12 位组成;

设备编号段: 由 CAN 标识符的第 11～5 位组成;

功能码段: 由 CAN 标识符的第 4～0 位组成;



附图 4 传递非过程变量的 CAN 标识符划分

一般非过程变量的实时性要求不高，在应用层通常适用主机-从机模型，主机和从机间采用问答方式通讯，变量分段管理与其总线竞争能力没有突出的矛盾，因此从 CAN 标识符的高端直接截取十六位(27～12 位)构成寄存器基地址段，当 CAN 标识符的第 27 位(即: 寄存器基地址段的最高位)为 1 时，寄存器基地址空间指向非过程变量空间(即:0x8000～0xFFFF)，

非过程变量的总线竞争能力整体低于过程变量，这是应用层所期望的，由于CAN标识符的高七位不能连续出现高电平1，当优先级位(28位)为1时，非过程变量实际有效的地址空间为0x8000～0xFBFF，也就是在0xF000段的尾部集中有1024个地址空间(0xFC00～0xFFFF)不能使用，如果对非过程变量采用和过程变量一样的寄存器拼接组合的方法构成寄存器基地址段，势必将这1024个不可用地址空间分散到各等价层中，这对非过程变量寄存器空间的使用和管理无益，因此应用层对过程变量和非过程变量分别采用不同的CAN标识符划分进行总线传递是非常必要的。

TTCANopen应用层协议对CAN标识符的划分及其物理含义的分配，如:优先级段、寄存器基地址段、设备编号段、功能码段都具有明确的应用层协议指令含义，据此设计的应用层协议具有很好的直读性和开放性，使应用层协议易于解析，便于用户在框架内构建适合其使用环境的应用层子协议。

TTCANopen应用层协议以连续寄存器空间为信息载体，其应用层协议指令是对该连续寄存器空间的读写访问，由此使其能够天然的兼容CAN-FD协议对数据场长度的扩充。

附录4： TTCANopen应用层协议

设备内部多条指令的发送规则

通常CAN通讯协议只规范了多个设备在总线上同时发送指令时的竞争顺序，而对于单个设备内部产生的多条指令的发送顺序却很少涉及。

在纯主从模式下，设备内部产生多条指令的可能性并不大，相关问题很难暴露，而当系统发展演绎到全触发方式后情况就不同了，一个设备可能具有多个触发变量，它们可以是周期触发、变化触发、越界触发等等，而总线的发送速率是有限的，且同时还存在其他设备对总线的竞争，因此在同一设备中就有可能积累多条指令等待发送。

通常设备会默认以“先生优势”规则处理这些指令，即先产生的指令优先发送，这样处理的最大好处是指令动作先后逻辑规范，这对那些具有先后关联的系列指令尤为重要，但事物总是两方面的，这种“先生优势”规则，会延迟和阻碍后产生的高优先级指令的发送，如：告警指令，尤其是当总线上存在竞争时，情况尤为严重，如:当设备指令发送端口被一条先生成的低优先级指令占据时，其在总线上的竞争处于劣势，从而使该设备中后产生的告警指令无法参与竞争而

被推迟发送，这种延迟有时可能是致命的。 因此我们需要重新规范设备内部多条指令的发送规则，最粗暴简单的方法就是对这多条指令进行优先级排队，将ID字小的指令排在前面，每当设备产生一条新指令时都要重新进行一次指令排序(注意:这种重新排序应当包括那条已经处在发送端口的未发指令)，这样设备便具有了最优的发送竞争力，但同时可能会破坏前面所说的某些指令的先后逻辑关联，可能使系统运转出现偏差。

为此TTCANopen应用层协议做出如下规范，当一设备同时具有多条指令等待发送时：

1)优先级位为0的指令优先于优先级位为1的指令发送;

2)优先级位相同的指令按其产生的先后顺序发送。

虽然只有短短的两条规定，但它却完成了其它应用层协议很难规范完成的内容，而这对于 TTCANopen协议来说却是自然天成的，它的第一条规定确保了低优先级(1)的指令不能阻碍高优先级(0)指令的发送，使设备中诸如告警指令得以顺利抢先通过总线发送；第二条规定保障了具有先后逻辑关系的系列指令的顺利完成，因为我们一般不会将这些相关指令定义在不同的优先级位上。

更多内容请关注：www.ttcanopen.com