Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue** | **Description** | **Author** |
| September 12, 2017 | 1.0 | Initial creation. Extracted the Problem Statement from the Vision document for purposes of scoping. | ZhangYikui |
| September 20, 2017 | 1.1 | Revise some description error  Revise Motorola big endian//Intel little endian description | ZhangYikui |
|  |  |  |  |
|  |  |  |  |

Version 1.1

Problem Statement

在现代汽车控制技术中，汽车中会使用多个电子控制装置（ECU：Electronic Control Unit）对整车进行控制。而ECU之间的信息交换更多地依赖于CAN(Controller Area Network)总线的网络连接方式来完成。为了检测和控制CAN bus的信息内容，需要使用CAN bus检测设备。CanTool装置是完成CANbus检测的工具。为了实现CAN数据的显示及控制，需要使用本文提出的CanToolApp软件。该软件需要将连接在CAN总线上的CanTool装置采集的CAN信息发送到上位机（移动终端Android、iOS、Windows PC），并由运行在上位机中的CanToolApp软件接收这些信息，显示在用户图形界面上。同时在CanToolApp的界面上还可以设定CAN信息，通过GUI按钮将信息发送给CanTool装置，CanTool装置将按照规定的信息格式，将信息发送的CAN总线上。

此外，CanToolApp可以设定CAN总线的通信参数，并通过相应的命令设置CanTool装置的CAN通信参数，以使CanTool装置能够与CAN总线上的其他被测ECU进行正常的通信。系统的总体框图见图 1所示。CanTool装置与上位机通过USB串口或蓝牙RFComm实现UART串口通信。与上位机连接的串口号需要用户选择，串口的波特率固定为115200BPS，8个数据位，1个停止位。



图 1 CanTool系统框图

上位机与CanTool装置之间的信息传送方式使用ASCII码格式+ \r（即0x0d）方式进行信息交换，当上位机发送给CanTool装置的信息被正确接收后，CanTool装置会返回\r,否则返回\BEL (即0x07)。通信格式如表 1所示。

表 1 CANTool装置与CanToolApp间数据通信格式

| CAN总线数据 | Dir | CanTool装置 | dir | CanToolApp | 说明 |
| --- | --- | --- | --- | --- | --- |
| 无 | - | 接收到“V\r”  返回CANtool装置的版本信息“SV2.5-HV2.0\r” | 🡸  🡺 | “V\r”  接收“SV2.5-HV2.0\r” | 上位机获得CanTool装置的版本信息 |
| 无 | - | 接收到“O1\r”  返回成功或失败信息。 | 🡸  🡺 | “O1\r”  \r或\BEL | CanTool装置Return: \r for Open OK, \BEL for Failure.  It works only after power up or if controller is in reset mode after command “C”. The following bit rates are available. |
| 无 | - | 接收到“Sn\r”  返回成功或失败信息。 | 🡸  🡺 | “Sn\r”  \r或\BEL | Sn为以下选项，设置CAN总线的通信速率。  S0 Setup 10Kbit  S1 Setup 20Kbit  S2 Setup 50Kbit  S3 Setup 100Kbit  S4 Setup 125Kbit  S5 Setup 250Kbit  S6 Setup 500Kbit  S7 Setup 800Kbit  S8 Setup 1Mbit  It works only after power up or if controller is in reset mode after command “C”. |
| 无 |  | 接收到“C\r”  返回成功或失败信息。 | 🡸  🡺 | “C\r”  \r或\BEL | Return: \r for Close OK, \BEL for Failure.  It works only if the controller was set to Operation mode with “O1\r” command before. |
| Id:iii  DLC:L 0x8  Data:DD…DD  实例1：  id:0x123  DLC;0x08,  Data:0x00,0x11,0x12,  0x13,0x14,0x15,  0x16,0x17  实例2：  id:0x3FF  DLC;0x04,  Data:0x00,0x11,0x12,  0x13 | 🡺 | tiiiLDDDDDDDDDDODDDDD\r  实例1：组成字符串如下  “t12380011121314151617\r”  实例1：组成字符串如下  “t3FF400111213\r”  注：\r=0x0d | 🡺 | 接收到CanTools装置的信息，将其解析为实际的CAN信号。解析过程中需要使用CAN信息及信号描述数据库 | CanTool装置接收到CAN总线信息后会自动将信息发送给上位机。  iii:为CAN标准ID的范围0-0x7FF.  L:表示数据长度DLC，范围0..8  DD:表示1字节(8bit)16进制数据,DD的数量由L的数值决定 |
| Id:iiiiiiii  DLC:L 0x8  Data:DD…DD  实例：  id:0x123FABCD  DLC;0x08,  Data:0x00,0x11,0x12,  0x13,0x14,0x15,  0x16,0x17 | 🡺 | TiiiiiiiiLDDDDDDDDDDODDDDD\r  实例：组成字符串如下  “T123FABCD80011121314151617\r”  注：\r=0x0d | 🡺 | 接收到CanTools装置的信息，将其解析为实际的CAN信号。解析过程中需要使用CAN信息及信号描述数据库 | This command transmits an extended 29 Bit CAN frame. It works only if controller is in operational mode after command "O". Identifier iiiiiiii in hexadecimal (00000000-1FFFFFFF) Data length code L (0-8), Data byte value in hexadecimal(00-FF). Number of given data bytes will be checked against given data length code. |
| ID: iiiiiiii  DLC:L  Data:DD DD … DD  实例：  ID：1234567F  DLC:8  DATA: 11 22 33 44 55 66 77 88 | 🡸  🡸 | CanTool装置将接收到的发送CAN扩展帧命令发送到CAN总线。  实例：接收到  “T1234567F81122334455667788\r”  返回”\r” 或”\BEL” | 🡸  🡸  🡺 | CanToolAPP向CanTool装置发送CAN信息命令“TiiiiiiiiLDD…DDmmmm\r”。此字串是由用户在GUI界面上输入的CAN信号的内容合成而来，需要参照CAN信息及信号描述数据库。  实例：单次发送  “T1234567F811223344556677880000\r”  “\r”或”\BEL” | This command transmits an extended 29 Bit CAN frame. It works only if controller is in operational mode after command "O1\r".  Identifier iiiiiiii in hexadecimal (00000000-1FFFFFFF)  Data length code L (0-8),  DD: Data byte value in hexadecimal(00-FF). Number of given data bytes will be checked against given data length code.  mmmm:range 00000-FFFF,代表周期发送的毫秒数。mmmm=0000，代表是发送一次，0001-FFFF 代表CanTool装置接收到此命令后，自动以mmmm为周期，发送该命令到CAN总线  Return: “\r”for OK, “\BEL” for Failure |
| ID: iii  DLC:L  Data:DD DD … DD  实例：  ID：12F  DLC:4  DATA:11 22 33 F4 | 🡸 | CanTool装置将接收到的发送CAN标准帧命令按制定周期发送到CAN总线。  实例：  接收到需要发送的CAN数据  返回”\r” 或”\BEL”  然后，CanTool将以  周期mmmm= 0x0110= 272ms.发送此CAN数据 | 🡸  🡸  🡺 | CanToolAPP向CanTool装置发送CAN信息命令“tiiiLDD…DDmmmm\r”。此字串是由用户在GUI界面上输入的CAN信号的内容合成而来，需要参照CAN信息及信号描述数据库。  实例：以定周期发送 “t12F4112233F40110\r”  “\r”或”\BEL” | This command transmits an standard 11 Bit CAN frame. It works only if controller is in operational mode after command "O1\r". Identifier iii in hexadecimal (000-7FF) Data length code L (0-8), Data byte value in hexadecimal(00-FF). Number of given data bytes will be checked against given data length code.  mmmm:range 00000-FFFF,代表周期发送的毫秒数。mmmm=0000，代表是发送一次，0001-FFFF 代表CanTool装置接收到此命令后，自动以mmmm为周期，发送该命令到CAN总线  Return: “\r”for OK, “\BEL” for Failure |

CanToolApp使用GUI界面对接收和发送的CAN信息进行显示。需要根据CAN信息及信号描述数据库对接收到的数据字符串进行解析，然后得到CAN信息中包含的各种CAN信号值，将此CAN信号值进一步进行计算，还原该信号所代表的物理量的信息，并显示在GUI界面上。要发送的CAN信息也采用同种方式将用户输入的物理值转换为CAN信号，并依据CAN信号描述数据库将属于同一个CANID的信号合成为字符串发送给CanTool装置。

首先要明确CAN信息（CAN message）是一组由ID，DLC，DATA组成的数据信息，DATA最长为8个字节。每个字节有8bit共64bit信息。这里的DATA是由多个CAN信号是组成的。CAN信号的长度最小是1bit,最长64bit。CAN信号的数值与实际它所代表的物理量的值通过phy=A\*x+B来计算。其中phy代表物理量的值，A为1LSB（Least Significant Bit）代表的物理值大小，也称Factor，x是CAN信号的值，B是物理量的偏移量。例如：x=0x10=16， A=0.1℃，B=-10℃，则此CAN信号代表的物理值phy=16\*0.1-10=-8.4℃。如果用户在GUI界面上输入了15℃，希望CanToolApp发送给CanTool装置并发送到CAN总线，那么就需要把此物理量的数值转换为CAN信号值x=(phy-B)/A=(15-(-10))/0.1=250=0xFA，将16进制0xFA

放到相应的CAN信息中，组成CAN信息字符串发送给CanTool装置。CAN信号使用Intel的Little Endian方式排列在CAN信息DATA中，排列方式如图 2所示。CAN信号也可以是按照Motorola的Big Endian方式排列在CAN信息DATA中，其排列方式如图 3所示。具体使用哪种方式，可以有用户在CAN信号描述数据库中指定。

|  |  |
| --- | --- |
| CAN信息 DATA的bit顺序（intel格式） | 当CAN信号占据多个字节时  LSB与MSB的顺序如下 |
| Msb:most significant bit |  |

图 2 CAN信号在CAN信息中的排列方式（Intel格式）

|  |  |
| --- | --- |
| CAN信息 DATA的bit顺序（Motorola格式） | 当CAN信号占据多个字节时  LSB与MSB的顺序如下 |
| Msb:most significant bit |  |

图 3 CAN信号在CAN信息中的排列方式（Motorola格式）

“CAN信息及信号描述数据库”的格式如表 2。CAN信号描述数据库格式如表 3所示。

表 2 “CAN信息Message描述数据库”格式

| 字段名 | 类型 | 样例 | 说明 |
| --- | --- | --- | --- |
| CANmessage标识符 | char[32] | BO\_ | 固定为BO\_ |
| id | uint32 | 2148606241 | 十进制数值2148606241转换为16进制为0x80112121，其中msb=bit31=1表示是CAN扩展帧，bit28~bit0是实际的CANID值=0x00112121 |
| 100 | 十进制数值100转换为16进制为0x00000064，其中msb=bit31=0表示是CAN标准帧，bit10~bit0是实际的CANID值=0x013 |
| Message Name | char[32] | EngMsg1 | 字符串，最长32字节 |
| 分隔符 | char[1] | : | 固定为 : |
| DLC | unsigned char | 8 | 范围:0—8, 表示此CAN信息的DATA的长度为8byte |
| Node Name | char[32] | BODY\_ECU | 字符串，最长32字节,发送此信息的Node名。也是ECU名 |
| 注：每个字段间间隔符除注明外，均使用空格作为分隔符，上述样例在数据库中保存的格式为。BO\_开始于新的一行第0列，信息的末尾为”\n”即DOS换行符”0x0d 0x0a”。  BO\_ 2148606241 Ext1: 8 Node\_Body | | | |

表 3 “CAN信号Signal描述数据库”格式

| 字段名 | 类型 | 样例 | 说明 |
| --- | --- | --- | --- |
| CAN signal 标识符 | char[32] | SG\_ | 固定为SG\_ |
| Signal Name | char[32] | EngMsg1 | 字符串，最长32字节 |
| 分隔符 | char[1] | : | 固定为 : |
| 起始位| bit长度@bit格式 | unsigned char[10] | 23|12@0+ | 23表示起始位编号  12 表示此CAN信息的DATA的长度为12bit  0+表示数据按Motorola的bit顺序排列,数据最高位MSB放在bit编号23的位置，即23bit为MSB,bit28为LSB，数据是左对齐的，即放置不在bit编号bit23,bit22…bit16; bit31,bit30,bit29,bit28的位置，注意此编号还是规定使用intel格式时的编号。从数据库的编号开始从左到右，并向高位数据字节依次排列MSB🡺LSB |
| 16|12@1+ | 16表示起始位编号  12 表示此CAN信息的DATA的长度为12bit  0+表示数据按Intel的bit顺序排列,数据最低位LSB放在bit编号16的位置，即16bit为LSB,bit27为MSB，数据是右对齐的，及从CAN数据库的编号开始依次从右向左,并向高位字节放置信号的LSB🡺MSB。 |
| (A,B) | Double | (0.1, -10) | A:分辨率，1LSB的物理值精度，B：物理值的偏移量offset。  Phy=A\*x+B, x为CAN信号的数值，phy为CAN信号对应的物理值 |
| [C|D] | double | [-41|368.5] | 物理值的范围：Min=C到MAX=D |
| 物理单位 | Char[32] | “℃” | 带有双引号的字符串，可以为空:””, |
| Node Name | char[255] | BODY\_ECU | 接收该信号的节点Node名列表（也是ECU名）字符串，最长32字节。如果多个ECU接收此信号，则用逗号将多了节点名隔开，例如：BCM,PEPS,ICM,CDU |
| 注：每个字段间间隔符除注明外，均使用空格作为分隔符，上述样例在数据库中保存的格式为。SG\_开始于新的一行第1列，信息的末尾为”\n”即DOS换行符”0x0d 0x0a”。  SG\_ sig1bittable : 8|1@1+ (1,0) [0|1] "" AC\_ECU  SG\_ Sig\_moto1 : 7|12@0+ (0.1,-41) [-41|368.5] "DegC" Node\_Body  SG\_ CDU\_NMWakeupOrignin : 23|8@0+ (1,0) [0|255] "ms" BCM,PEPS,ICM,CDU | | | |

实际某汽车的CAN信息和信号数据库内容如表 4所示，供参考。CanToolApp应能正常完成对上述各种数据的解析及合成。

表 4实际数据库样式

| 实际数据库样式 | 说明 |
| --- | --- |
| BO\_ 856 CDU\_1: 8 CDU  SG\_ CDU\_HVACOffButtonSt : 0|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACOffButtonStVD : 1|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACAutoModeButtonSt : 2|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACAutoModeButtonStVD : 3|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACFDefrostButtonSt : 6|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACFDefrostButtonStVD : 7|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACDualButtonSt : 10|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACDualButtonStVD : 11|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACIonButtonSt : 12|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACIonButtonStVD : 13|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACCirculationButtonSt : 18|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACCirculationButtonStVD : 19|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACACButtonSt : 20|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACACButtonStVD : 21|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACACMaxButtonSt : 22|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACACMaxButtonStVD : 23|1@0+ (1,0) [0|1] "" HVAC  SG\_ CDU\_HVACModeButtonSt : 26|3@0+ (1,0) [0|7] "" HVAC  SG\_ HVAC\_WindExitSpd : 30|4@0+ (1,0) [0|15] "" Vector\_\_XXX  SG\_ CDU\_HVAC\_DriverTempSelect : 36|5@0+ (0.5,18) [18|32] "°C" Vector\_\_XXX  SG\_ HVAC\_PsnTempSelect : 44|5@0+ (0.5,18) [18|32] "" Vector\_\_XXX  SG\_ CDU\_HVACCtrlModeSt : 54|3@0+ (1,0) [0|7] "" HVAC  SG\_ CDU\_ControlSt : 55|1@0+ (1,0) [0|1] "" HVAC  BO\_ 61 CDU\_4: 8 CDU  SG\_ CDU\_HVACACCfg : 1|2@0+ (1,0) [0|3] "" HVAC  SG\_ CDU\_HVACAirCirCfg : 3|2@0+ (1,0) [0|3] "" HVAC  SG\_ CDU\_HVACComfortCfg : 5|2@0+ (1,0) [0|3] "" HVAC  BO\_ 1067 CDU\_NM: 8 CDU  SG\_ CDU\_NMDestAddress : 7|8@0+ (1,0) [0|255] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMAlive : 8|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMRing : 9|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMLimpHome : 10|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMSleepInd : 12|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMSleepAck : 13|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMWakeupOrignin : 23|8@0+ (1,0) [0|255] "" BCM,PEPS,ICM,CDU  SG\_ CDU\_NMDataField : 31|40@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  BO\_ 1056 BCM\_NM: 8 BCM  SG\_ BCM\_NMDestAddress : 7|8@0+ (1,0) [0|255] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMAlive : 8|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMRing : 9|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMLimpHome : 10|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMSleepInd : 12|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMSleepAck : 13|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMWakeupOrignin : 23|8@0+ (1,0) [0|255] "" BCM,PEPS,ICM,CDU  SG\_ BCM\_NMDataField : 31|40@0+ (1,0) [0|1] "" BCM,PEPS,ICM,CDU  BO\_ 792 BCM\_BCAN\_1: 8 BCM  SG\_ BCM\_KeySt : 1|2@0+ (1,0) [1|3] "" PEPS,ICM,AVM,CDU,HVAC  BO\_ 837 BCM\_ESC\_2: 8 BCM  SG\_ ESC\_VehSpdVD : 37|1@0+ (1,0) [0|1] "" BCM,PEPS,ICM,AVM,CDU  SG\_ ESC\_VehSpd : 36|13@0+ (0.05625,0) [0|240] "" BCM,PEPS,ICM,AVM,CDU  BO\_ 915 BCM\_VCU\_2: 8 BCM  SG\_ VCU\_CompressorPwrLimit : 21|6@0+ (100,0) [0|6000] "w" HVAC  SG\_ VCU\_CompressorPwrLimitAct : 32|1@0+ (1,0) [0|1] "" HVAC  SG\_ VCU\_PTCPwrLimit : 29|6@0+ (100,0) [0|6000] "w" HVAC  SG\_ VCU\_PTCrPwrLimitAct : 33|1@0+ (1,0) [0|1] "" HVAC  SG\_ VCU\_AirCompressorReq : 36|1@0+ (1,0) [0|1] "" HVAC  SG\_ VCU\_AirCompressorReqVD : 37|1@0+ (1,0) [0|1] "" HVAC  BO\_ 800 HVAC\_1: 8 HVAC  SG\_ HVAC\_AirCompressorSt : 2|3@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_CorrectedExterTempVD : 3|1@0+ (1,0) [0|1] "" BCM,CDU  SG\_ HVAC\_RawExterTempVD : 4|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_EngIdleStopProhibitReq : 5|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_ACSt : 6|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_ACmaxSt : 7|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_CorrectedExterTemp : 15|8@0+ (0.5,-40) [-40|87.5] "°C" BCM,CDU  SG\_ HVAC\_RawExterTemp : 23|8@0+ (0.5,-40) [-40|87.5] "°C" CDU  SG\_ HVAC\_TempSelect : 28|5@0+ (0.5,18) [18|32] "°C" CDU  SG\_ HVAC\_DualSt : 29|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_AutoSt : 30|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_Type : 31|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_WindExitMode : 34|3@0+ (1,0) [0|7] "" CDU  SG\_ HVAC\_SpdFanReq : 36|2@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_TelematicsSt : 42|3@0+ (1,0) [0|7] "" CDU  SG\_ HVAC\_AirCirculationSt : 46|2@0+ (1,0) [0|3] "" CDU  SG\_ HVAC\_PopUpDisplayReq : 47|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_DriverTempSelect : 53|5@0+ (0.5,18) [18|32] "°C" CDU  SG\_ HVAC\_IonMode : 55|2@0+ (1,0) [0|3] "" CDU  SG\_ HVAC\_WindExitSpd : 59|4@0+ (1,0) [0|15] "" CDU  SG\_ HVAC\_PsnTempSelect : 48|5@0+ (0.5,18) [18|32] "" CDU  BO\_ 801 HVAC\_2: 8 HVAC  SG\_ HVAC\_RawCabinTemp : 7|8@0+ (0.5,-40) [-40|87.5] "°C" CDU  SG\_ HVAC\_CorrectedCabinTemp : 15|8@0+ (0.5,-40) [-40|87.5] "°C" CDU  SG\_ HVAC\_RawCabinTempVD : 19|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_CompressorComsumpPwr : 17|10@0+ (10,0) [0|8000] "w" BCM  SG\_ HVAC\_PTCPwrAct : 33|10@0+ (10,0) [0|8000] "w" BCM  SG\_ HVAC\_stPTCAct : 55|3@0+ (1,0) [0|1] "" BCM  SG\_ HVAC\_CorrectedCabinTempVD : 18|1@0+ (1,0) [0|1] "" CDU  BO\_ 797 HVAC\_3: 8 HVAC  SG\_ HVAC\_ACCfgSt : 0|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_AirCirCfgSt : 1|1@0+ (1,0) [0|1] "" CDU  SG\_ HVAC\_ComfortCfgSt : 3|2@0+ (1,0) [0|1] "" CDU  BO\_ 864 HVAC\_4: 8 ACP  SG\_ HVAC\_ACPCommandVD : 0|1@0+ (1,0) [0|1] "" ACP  SG\_ HVAC\_ACPCommand : 2|2@0+ (1,0) [0|3] "" ACP  SG\_ HVAC\_ACPSpeedSet : 14|7@0+ (100,0) [0|8600] "" ACP  SG\_ HVAC\_ACPHighSidePress : 21|6@0+ (0.5,0) [0|31] "" ACP  SG\_ HVAC\_PTCPowerRatio : 31|8@0+ (1,0) [0|100] "" PTC  SG\_ HVAC\_Checksum : 39|8@0+ (1,0) [155|255] "" PTC  BO\_ 867 ACP\_1: 8 ACP  SG\_ ACP\_Speed : 6|7@0+ (100,0) [0|8600] "" HVAC  SG\_ ACPComsumpPwr : 15|10@0+ (10,0) [0|8000] "" HVAC  SG\_ ACP\_Current : 16|9@0+ (0.1,0) [0|51] "" HVAC  SG\_ ACP\_MotorTemp : 39|8@0+ (1,-40) [-40|140] "" HVAC  SG\_ ACP\_HearBeat : 55|4@0+ (1,0) [0|15] "" HVAC  SG\_ ACP\_ExtState : 58|3@0+ (1,0) [0|7] "" HVAC  SG\_ ACP\_FailGrade : 60|2@0+ (1,0) [0|3] "" HVAC  SG\_ ACP\_BaseState : 63|3@0+ (1,0) [0|7] "" HVAC  BO\_ 868 PTC\_1: 8 PTC  SG\_ PTC\_ElementError : 7|4@0+ (1,0) [0|15] "" HVAC  SG\_ PTC\_TemperatureOver : 3|1@0+ (1,0) [0|1] "" HVAC  SG\_ PTC\_VoltageFault : 2|1@0+ (1,0) [0|1] "" HVAC  SG\_ PTC\_InternalError : 1|2@0+ (1,0) [0|3] "" HVAC  SG\_ PTC\_Current : 15|8@0+ (0.2,0) [0|25.4] "A" HVAC  SG\_ PTCPwrAct : 23|10@0+ (10,0) [0|8000] "w" HVAC  SG\_ PTCActst : 26|3@0+ (1,0) [0|7] "" HVAC | 每个CAN的Message后都跟着改Message要发送的Can信号Signal及其参数。 |

CanToolApp需要完成的主要功能如下，可选平台Android应用、Windows应用、可使用Java、C#、C++、JavaScript等计算机语言实现：

1. 能够搜索到本机所有可使用的COM口，并在弹出式ComboBox中以列表方式让用户选择CanTool装置在上位机中映射的COM口。并设置相应COM口波特率115200、数据位数8、停止位数1。这些设定内容可保存到CanToolApp设定文件中，供下次使用。基本信息如图 4所示。

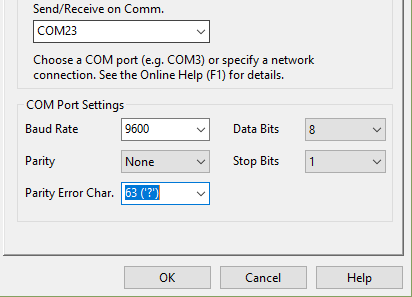


图 4 COM口基本设定信息

1. 能够实现CANtool装置的CAN速率设置、进入CAN工作状态（Open）、进入CAN初始化状态（ Close）。这些设定内容可保存到CanToolApp设定文件中，供下次使用。

略

图 5 CAN基本设定信息

1. 能够对接收到的多个CAN信息，通过CAN信息及CAN信号数据库进行解析，将CAN信息原始数据进行显示。并能对CAN信息中的CAN信号的物理值实时数据进行显示。样式如图 6所示。

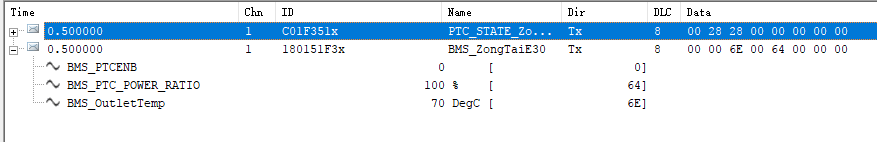


图 6 CAN信息原始数据及物理数据实时显示

1. 显示时可以让用户选择仪表盘方式显示接收到CAN信号物理值。这些用户选择的显示方式可保存到CanToolApp设定文件中，供下次使用。仪表盘样式如图 7所示。仪表盘、LED等需要自制控件。

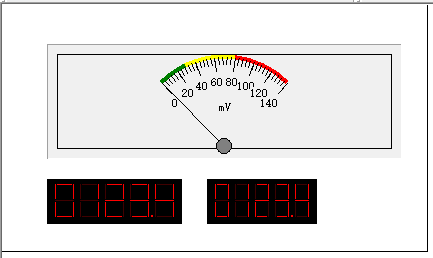


图 7 CAN信号的仪表盘

1. 可以让用户选择某些接收到的CAN信号，显示其变化的实时物理值曲线。样式如图 8所示。

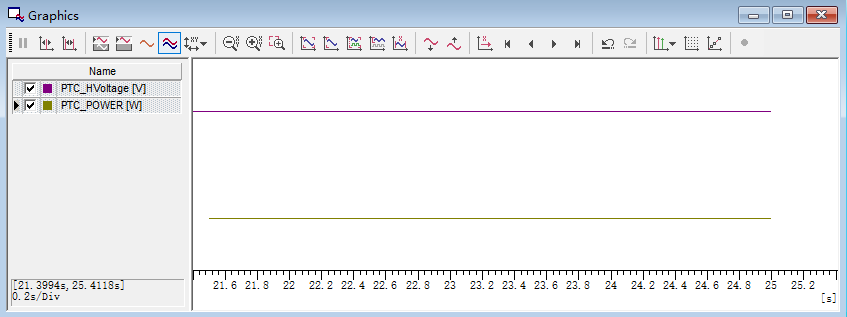


图 8 CAN信号实时曲线

1. 可以将接收到的所有CAN信息数据，实时保存为数据文件。格式为CSV格式，或自定义。
2. 能够指定要发送的多个CAN信息，并允许用户设定CAN信息中的CAN信号物理值。可以指定CAN信息的发送周期（0-65535ms即0x0000-0xFFFF）。
3. App可将用户设定的物理值转换为CAN信号值，将CAN信息中包含的所有CAN信号合成完整的CAN信息后，发送给CanTool装置，发送到CAN总线上。
4. 可以加载用户提供的CAN信息和信号数据库，完成CAN信号数据的解析以及CAN发送信息的组装。可以显示CAN信号在CAN信息的布局，样例如图 9所示。未加载数据库的情况下，只显示CAN信号的原始值，即phy=A\*x+B中的A=1，B=0。加载的数据库文件相关信息，可保存到CanToolApp设定文件中，供下次使用。

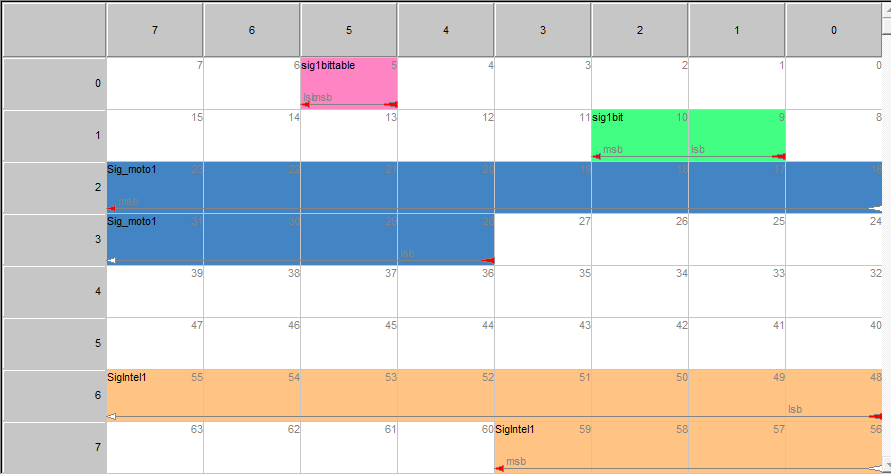


图 9 CAN信号在CAN信息中的布局

1. 加载用户提供的CAN信息和信号数据库，可以树状结构显示在GUI界面中。树状结构如图 10所示，供参考。

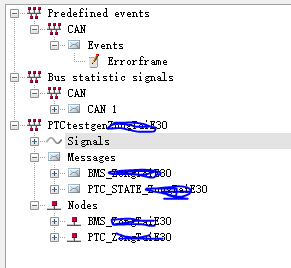


图 10 CAN信息树状结构

1. 可以将用户提供的CAN信息和信号数据库另存为xml和JSON (JavaScript Object Notation)格式。也可以已将xml或Json格式的数据库，转换为CAN信息和信号数据库格式。
2. 可以将所有CAN信息实时数据、CAN设定信息等 通过WEB API方式更新到远程数据库。此时CanToolApp作为客户端与远程的Web API服务进行数据交换。此功能需要完成WebAPI服务器端的get\post等服务，实现数据的增删改查、可视化数据显示，实现数据共享。（此功能可以独立一个项目完成）。
3. 功能可能随时增加或修改，需要做好变更管理。

## 问题反馈

1. 如何判断CAN扩展帧与标准帧？

Ans:在表 1中描述的CANToolApp接收和发送给CANTool装置的CAN信息中，以大写T开始的CAN信息是CAN扩展帧信息，以小写t开始的CAN信息是CAN标准帧信息。

1. 虚拟CANtool装置与CANToolAPP之间的数据如何产生？

Ans:可以通过自己定义的数据模拟CAN总线的信息。但需要能够满足测试CAN信号的不同类型、排列方式及bit信息长度的Little Endian/Big Endian的需要。例如：1bit , 2bit, 4bit, 5bit, 1byte，12bit，2byte，3byte，4byte的信号的排列方式。大于1bit的CAN信号需要考虑signed, unsigned的数据类型。

1. 是否需要CANTool装置与CANToolAPP之间实现实时的通信？

Ans:是，需要CANTool装置与CANToolAPP之间实现实时通信。

1. CANTool装置使用什么方式实现？

Ans:可是使用Arduino的硬件配合串口/蓝牙来实现虚拟的CANTool装置，需要编写Arduino的C/C++语言程序，此方式可以适用于Android/Windows开发。或使用纯软件的com0com0虚拟串口方式来模拟，此方法只适用于Windows APP开发。