CAN 总线自定义协议使用说明

用C语言实现自己的协议

进入 EV5000 安装目录下 builddriver 目录（如图 1），这个目录里面的 fbserver.c 文件即协议程序，用户不需要了解 CAN 口的细节编程，只需要按照该框架，用 C 语言来编写自己的协议即可。不可随意更改该文件中的函数名及头文件引用。

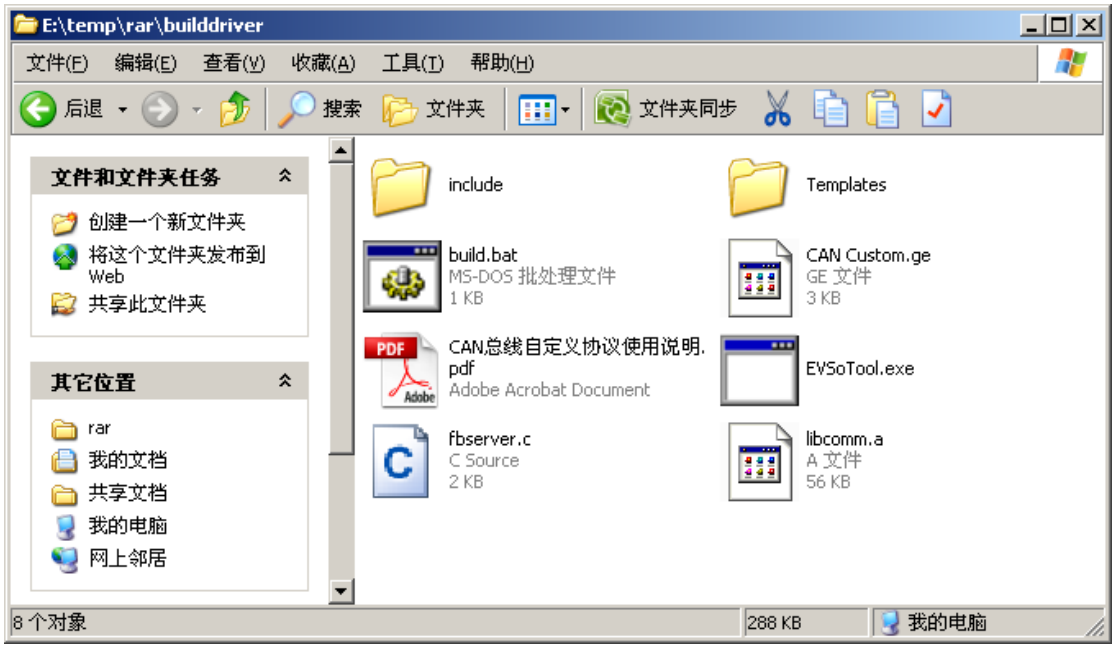
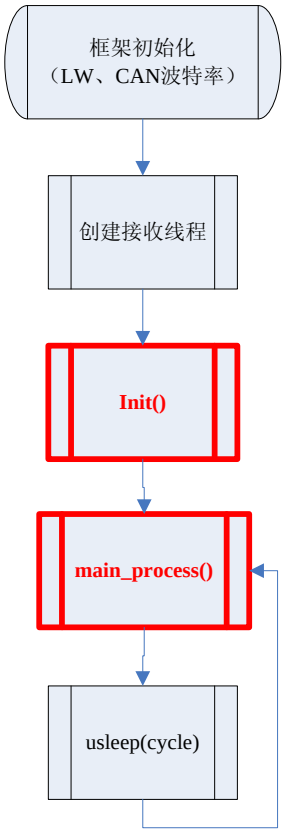


图 1

CAN自定义协议程序的流程图

主程序流程



接收线程流程

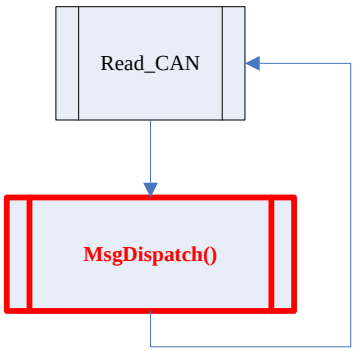


图 2

需要用户实现的函数

void Init(CAN_PORT canport)
调用方式：仅在组态程序运行的时候执行一次
功能：用户程序的初始化

void main_process(CO_Data* d, UNS32 id)
调用方式：周期性执行，默认周期为 10ms，周期可以在 void Init(CAN_PORT canport)中调用 Set_Cycle 来设定，最小周期为 10ms
功能：用户程序的“main 函数”

void MsgDispatch(CO_Data* d, Message *m)
调用方式：每接收到一帧 CAN 数据，就执行一次
功能：常用于对接收到的数据做解析，或者做出响应。与 main_process 没有关联。

供用户调用的API_V0.1

除了 C 语言标准库中的 API 可以使用外，还提供以下 API

void Set_Cycle(ms)

功能：用于设定 main_process 的运行周期，

参数：ms 的单位为毫秒，非零

UNS8 Send_Msg(CAN_PORT port, Message *m,UNS8 bExtended)

功能：向 CAN 总线发送一帧 CAN 数据

参数：port 指向已打开 CAN 口的句柄，m 指向 Message 结构体的指针，bExtended 为 1 时按扩展帧发送，为 0 时按标准帧发送

void Write_LW8K (UNS32 n,UNS16 val)

功能：将 val 写入 LW8000＋n 的寄存器

参数：n 偏移量、最大 999，val 待写入的值

UNS16 Read_LW8K (UNS32 n)

功能：读取 LW8000＋n 的寄存器的值

参数：n 偏移量、最大 999

void Set_Timer(TimerCallback_t callback,TIMEVAL value, TIMEVAL period)

功能：使用定时器，经过设定的时间后，调用 callback 函数

参数：callback 回调函数指针，value 单次定时时间，period 周期定时时间

int CopyToLW(UNS32 addr,const void *src, size_t n)

功能：由 src 所指内存区域复制 n 个字节到 LW +addr 所在内存区域

参数：addr 为 LW 寄存器地址、最大 8999；**src** 待写入数据缓冲的指针；**n** 写入的字节数

说明：src 和 LW8000＋addr 所在内存区域不能重叠

返回值：失败，返回-1

成功，返回写入的字节数

int CopyFromLW(UNS32 addr,const void *src, size_t n)

功能：由 LW8000＋addr 所在内存区域复制 n 个字节 src 所指内存区域

说明：addr 为 LW 寄存器地址、最大 9999；**src** 存储读取数据缓冲的指针；**n** 读取的字节数

返回值：失败，返回-1

成功，返回读取的字节数

早期的版本只能使用 LW8000~8999 这段寄存器，现在已取消该限制，CopyToLW，CopyFromLW 这个两个 API 的参数意义有所变化，现在 CopyToLW，CopyFromLW 的第一个参数表示 LW 的地址，而之前用 0~999 表示 LW8999~LW8999。因此，之前的代码需要做小的改动，如：

老版本

CopyFromLW(100,readbuff, 8)

表示将从 LW8100 开始的 8 个字节读取到 readbuff 中

现在需要改成 CopyFromLW(8100,readbuff, 8)才具有相同的效果。

UNS16 Read_LW(UNS32 n,short *perror)
功能：读取 LWn 的寄存器的值
参数：n 为 LW 寄存器地址、最大 9999，**perror** 返回错误码的指针
返回值：当*perror=-1 时，地址超出范围，返回 0
当*perror=0 是，成功，返回 LWn 的值

UNS16 Write_LW(UNS32 n,UNS16 val,short *perror)
功能：将 val 写入 LWn 寄存器中
参数：n 为 LW 寄存器地址、最大 8999；**val** 待写入的数据；**perror** 返回错误码的指针
返回值：当*perror=-1 时，地址超出范围，返回 0
当*perror=0 是，成功，返回 1

int SetLB(unsigned int n,unsigned char val)
功能：将 val 的值写入 LBn
参数：n 为 LB 寄存器地址、最大 8999；**val** 待写入的值（0 或 1）
返回值：地址超出范围，返回-1
成功，返回 LBn 的值

int GetLB(unsigned int n)
功能：读取 LBn 的值
参数：n 为 LB 寄存器地址、最大 9999
返回值：地址超出范围，返回-1
成功，返回 LBn 的值

int SetLWB(unsigned int n, unsigned int offset,unsigned char val)
功能：将 **val** 的值写入 LW.Bn
参数：n 为 LW 寄存器地址、最大 8999；**offset** 该位在 LWn 中的偏移量；**val** 待写入的值（0 或 1）
返回值：地址超出范围，返回-1
成功，返回 LBn 的值

int GetLWB(unsigned int n, unsigned int offset)
功能：读取 LW.Bn 的值
参数：n 为 LW 寄存器地址、最大 9999；**offset** 该位在 LWn 中的偏移量；
返回值：地址超出范围，返回-1
成功，返回 LBn 的值

相关的结构体定义

```
typedef struct {
  UNS32 w; /* 32 bits */
```

```
} SHORT_CAN;


/** Can message structure */
typedef struct {
   SHORT_CAN cob_id;   /* l'ID du mesg */
   UNS8 rtr;                  /* remote transmission request. 0 if not rtr,
                              1 for a rtr message */
   UNS8 len;                  /* message length (0 to 8) */
   UNS8 data[8]; /* data */
} Message;

typedef void* CAN_PORT;
```

编译、生成驱动

编写好程序后，只需运行图 1 中的“build.bat”批处理文件即可生成驱动文件，生成的驱动文件位于 EV5000 安装目录下\lib\fieldbus driver 中，名称为“CAN Custom.so” “CAN Custom.ge”

如何使用驱动

1. 在组态工程中，打开 HMI 属性对话框，如图 3

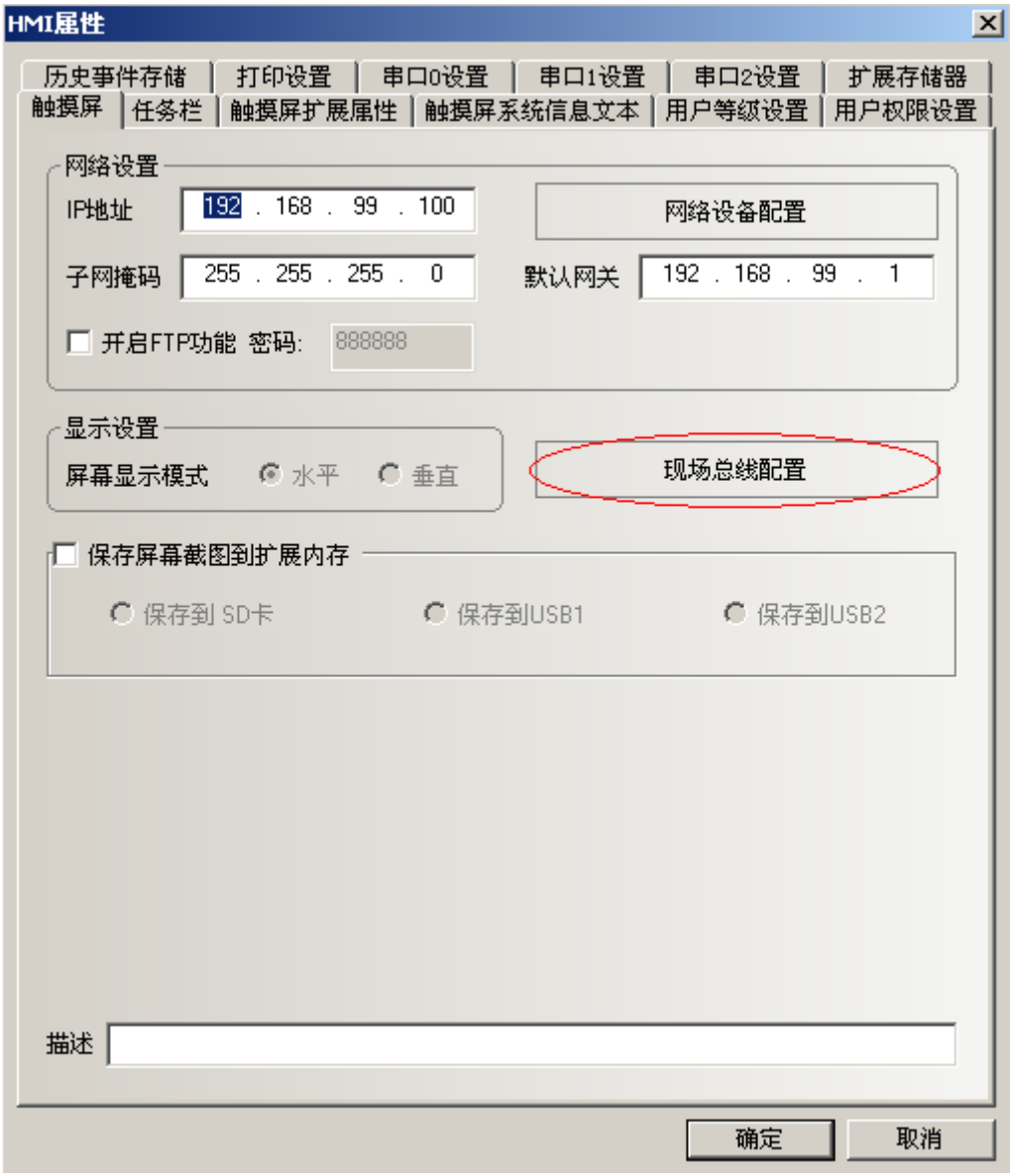


图 3

2. 点击“现场总线配置”，进入现场总线配置对话框，点击“增加”选择使用何种协议，如图 4

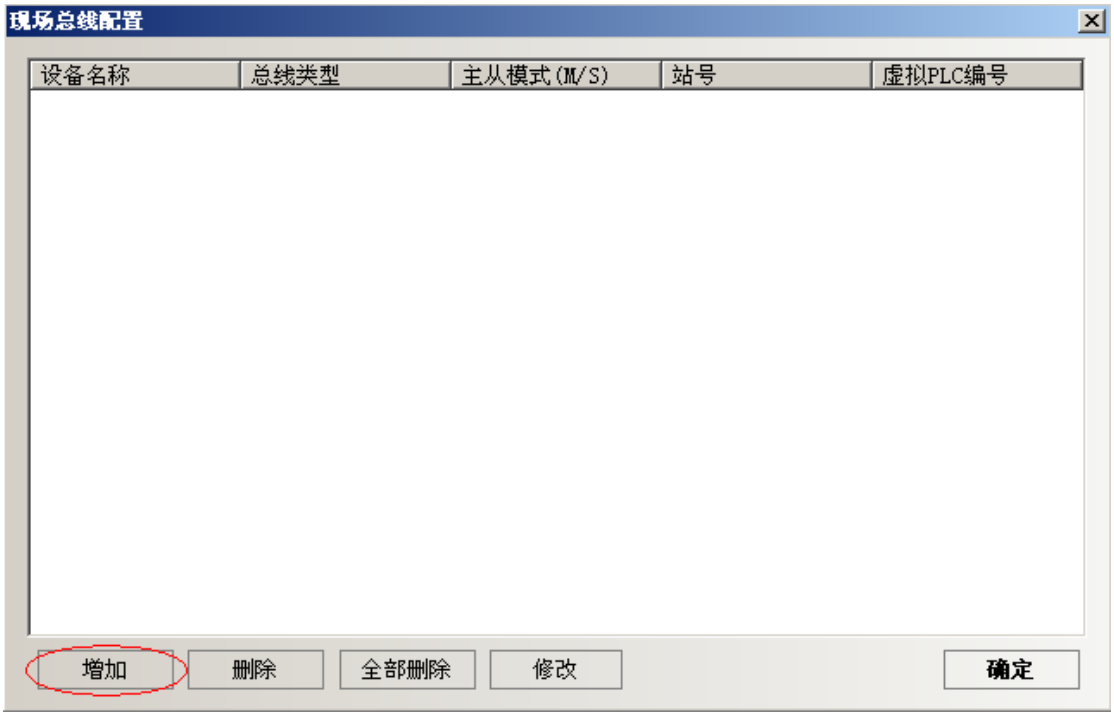


图 4

3. 选择“CAN Custom Protocol”，如图 5

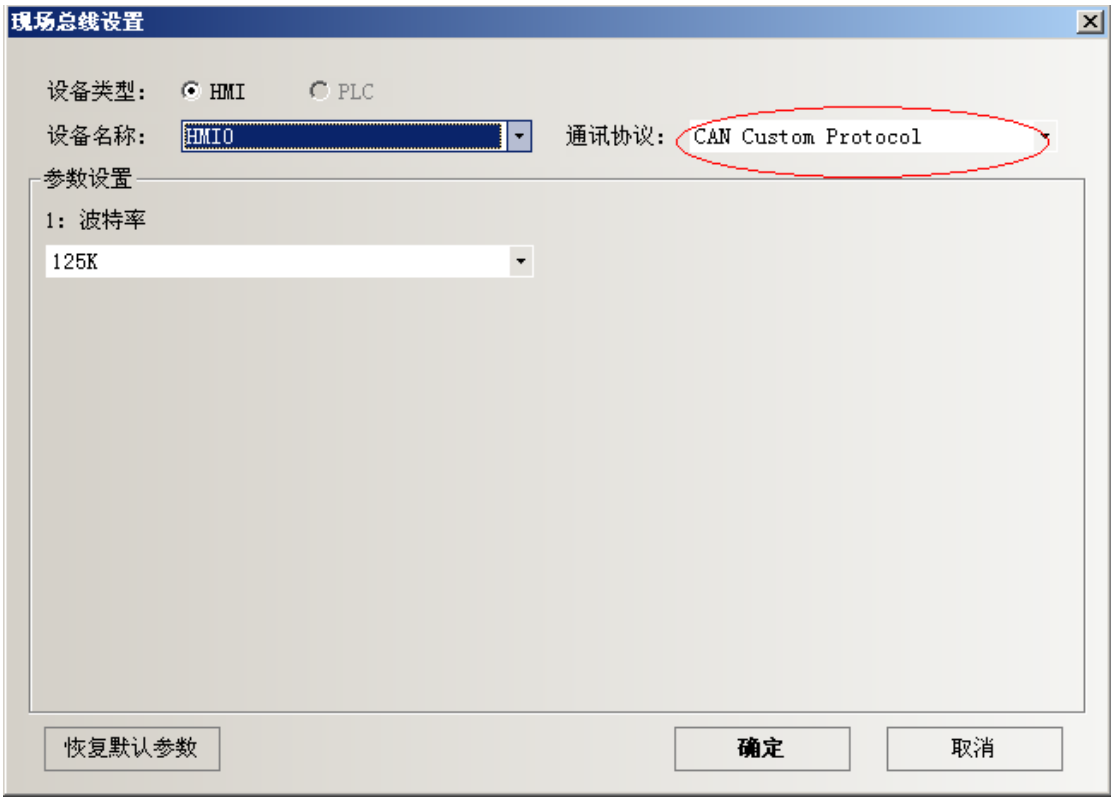


图 5

4. 制作自己的组态画面。