X2CAN API V3 - USB2CAN

Obsah:

Ing. David Španěl

Mgr. Vítězslav Rejda

X2CAN API

Toto API je určené pro převodníky CAN bus firmy CANLAB s.r.o. Každý převodník má vlastní sadu API funkcí. Funkce vždy začínají názvem převodníku. Tedy PP2CAN_***funkce***, USB2CAN_***funkce***, ETH2CAN_***funkce*** atd. Převodníky USB2CAN TRIPLE a COMBO mají společnou sadu funkcí USB2CAN_TRIPLE_***funkce***. V2CAN_***funkce*** je pak virtuální interface – loopback.

Pro situace, kdy je třeba integrovat více typů převodníků do jednoho software (například software PP2CAN), je k dispozici sada funkcí X2CAN. To jsou ukazatele na funkce, které se nastaví na funkce dle vybraného převodníku. Pak je možné například pro příjem CAN zpráv vždy použít funkci `X2CAN_GetCANMessage`.

Pro situace, kdy je třeba pracovat nejednou s více převodníky jednoho typu, probíhá výběr převodníku, se kterým se bude pracovat pomocí funkcí `USB2CAN_SelectActualAdapter` , `USB2CAN_TRIPLE_SelectActualAdapter` a podobně.

Varianty převodníků

Převodník USB2CAN

Převodník USB – 1x CAN bus 2.0, pro USB komunikaci používá obvod FTDI.

Převodník USB2CAN TRIPLE

Převodník USB – 2x CAN bus 2.0 + 1x CAN FD
Využívá USB osazeného MCU a vlastní PID a VID – sublicence Microchip.
VID: 0x04D8
PID: 0xF0D7

Převodník USB2CAN COMBO

Převodník USB – 1x CAN FD + 1x CAN low speed + 1x CAN single wire
Využívá USB osazeného MCU a vlastní PID a VID – sublicence Microchip.
VID: 0x04D8
PID: 0xE9B2

Detekce a nalezení

Vyhledání zařízení s FTDI chipem

Tedy i možného převodníku USB2CAN a naplnění vyhledaných sériových čísel do combo boxu.

```
FT_STATUS ftStatus;
DWORD numDevs;

m_device_list.ResetContent();
m_device_list.RedrawWindow();
UpdateData(FALSE);

ftStatus = FT_ListDevices(&numDevs, NULL, FT_LIST_NUMBER_ONLY);
if(ftStatus == FT_OK)
```

```
{
	CString str;
	ftStatus = FT_ListDevices(&numDevs, NULL, FT_LIST_NUMBER_ONLY);
	if(ftStatus == FT_OK)
	{
		DWORD d;
		char *BufPtrs[64]; // pointer to array of 64 pointers
		for(d=0; d<numDevs; d++)
			BufPtrs[d] = new char[64];

		ftStatus = FT_ListDevices(BufPtrs, &numDevs, FT_LIST_ALL|FT_OPEN_BY_SERIAL_NUMBER);
		if (FT_SUCCESS(ftStatus))
		{
			for(DWORD u=0; u<numDevs; u++)
			{
				str.Format("%s", BufPtrs[u]);
				m_device_list.AddString(str);
			}
		}
		else
		{
			str.Format("ListDevices failed");
			m_device_list.AddString(str);
		}

		//free memory
		for(d=0; d<numDevs; d++)
			delete BufPtrs[d];
	}
}
else
{
	AfxMessageBox("FT_ListDevices failed");
}
if(m_device_list.GetCount()>0)
	m_device_list.SetCurSel(0);
```

Detekce připojení nebo odpojení převodníku

Pro detekci je možné použít funkci API `USB2CAN_DeviceNotificationStart`. Od verze API 3.004 není funkce limitována pouze na klasický převodník USB2CAN ale pro jakékoliv zařízení USB. Parametrem funkce je pak callback, kam je předán PID a VID připojeného nebo odpojeného zařízení.

Start monitorování:

```
USB2CAN_DeviceNotificationStart(USB2CAN_DeviceNotification);
```

Ukončení monitorování:

```
USB2CAN_DeviceNotificationStop();
```

Příklad jednoduché callback funkce:

```
void USB2CAN_DeviceNotification(USB2CAN_DEVICE_NOTIFICATION notification,
USB2CAN_DEVICE_NOTIFICATION_INFO info)
{
	if(p_log->GetCount()>50)
	{
		p_log->ResetContent();
		p_log->RedrawWindow();

	}
	CString s;

	if((info.pid==0x6001)&&(info.vid==0x0403))
		p_log->InsertString(0,"USB2CAN or FTDI device");
	else if((info.pid==0xF0D7)&&(info.vid==0x04D8))
		p_log->InsertString(0,"USB2CAN TRIPLE");
```

```
        else if((info.pid==0xE9B2)&&(info.vid==0x04D8))
                p_log->InsertString(0,"USB2CAN COMBO");

        if(notification == INSERT)
                s.Format("INSERT PID:%.4X VID:%.4X", info.pid, info.vid );
        else
                s.Format("REMOVE PID:%.4X VID:%.4X", info.pid, info.vid );

        p_log->InsertString(1,s);
        p_log->InsertString(2,"---------------");
}
```

Inicializace

Inicializace převodníku USB2CAN

```
//Ukazatel kde bude vytvořena datová struktura převodníku
void *USB2CAN_adapter;
//Z combo boxu viz příklad vyhledání převodníku USB2CAN přečtu položku //vybraného
sériového čísla
char usb2can_ident[128];
m_device_list.GetLBText(m_device_list.GetCurSel(),usb2can_ident);

//Alokuji paměť pro strukturu dat nutnou pro převodník
USB2CAN_adapter = USB2CAN_PrepareAdapterEx(OPEN_BY_SERIAL_NUMBER, usb2can_ident);
//Vyberu pro práci tento adaptér
USB2CAN_SelectActualAdapter(USB2CAN_adapter);
// Otevreni adapteru
bool back = USB2CAN_Open(SPEED_125k, TRUE ,ErrorUSB2CAN, false);
if(!back)     p_log->AddString("USB2CAN Error");
else p_log->AddString("USB2CAN OK");
//Nastavím ukazatele pro případné použití univerzálních funkcí X2CAN API
X2CAN_USB2CAN_SetPointers();
```

`ErrorUSB2CAN` – toto je ukazatel na callback, kam jsou předávány chybové hlášení:

```
void ErrorUSB2CAN(int error_code,const char *text)
{
       //vypsat nebo zapsat do logu
}
```

Inicializace převodníku USB2CAN TRIPLE

```
//Ukazatel kde bude vytvořena datová struktura převodníku
void *USB2CAN_triple_adapter;
//Alokuji paměť pro strukturu dat nutnou pro převodník
//Parametrem funkce je číslo COM portu
//Je li zadána hodnota -1, vyhledá se číslo COM portu samo dle PID a VID
USB2CAN_triple_adapter = USB2CAN_TRIPLE_PrepareAdapter(-1);

//Nastavím rychlosti jednotlivých CAN portů
USB2CAN_TRIPLE_PrepareAdapterSpeedC1((void*)USB2CAN_triple_adapter,SPEED_250k,false);
USB2CAN_TRIPLE_PrepareAdapterSpeedC2((void*)USB2CAN_triple_adapter,SPEED_250k,false);
USB2CAN_TRIPLE_PrepareAdapterSpeedC3((void*)USB2CAN_triple_adapter, CAN_250K_1M,false,false, true);
USB2CAN_TRIPLE_SelectActualAdapter(USB2CAN_triple_adapter);
//Přesné časové značky přijatých zpráv
USB2CAN_TRIPLE_SetTimeStampMode(true);
//Mód upřednostňující pořadí
USB2CAN_TRIPLE_SetBufferMode(1);
//Zde se opakuje rychlost pro port 1 – kompatibilita univerzálních X2CAN funkcí
USB2CAN_TRIPLE_Open(SPEED_250k,NULL);
//Nastavím ukazatele pro případné použití univerzálních funkcí X2CAN API
X2CAN_USB2CAN_TRIPLE_SetPointers();
```

Inicializace převodníku USB2CAN COMBO

Tento převodník používá stejný MCU, avšak jiné budiče CANu a modifikovaný FW. Používají se stejné funkce jako pro převodník TRIPLE: Pro práci s tímto převodníkem se pouze volá funkce

```
void* USB2CAN_TRIPLE_PrepareAdapter(int port_number, bool combo = false);
```

s dodatečným parametrem. Například takto:

```
void *USB2CAN_combo_adapter;

USB2CAN_combo_adapter = USB2CAN_TRIPLE_PrepareAdapter(-1, true);
```

Parametr combo nastavený na true specifikuje, že se bude pracovat s COMBO převodníkem.

Práce se zprávami

Pro příjem i odeslání CAN zpráv je používána struktura CAN_MESSAGE.

Pro CAN zprávy ve formátu CAN 2.0 s 8 datovými bajty se používá pro uložení těchto dat položka pole `data`. V případě portu převodníku, který podporuje CAN FD je v případě delší zprávy než 8 datových bajtů vyplněno prvních 8 datových bajtů také do pole `data`. Pro další bajty se alokuje paměť a tento ukazatel se nastavuje do položky `fd_data`.
Při odesílání alokuje paměť a nastavuje ukazatel buď uživatel, nebo je možné použít funkci `CAN_MESSAGE_FD_prepare` nebo `CAN_MESSAGE_FD_alocate`. První funkce vynuluje i položky zprávy, druhá jen případně alokuje paměť (dle délky).

Pro dealokaci při přijetí zprávy po jejím zpracování je pak k dispozici funkce `CAN_MESSAGE_delete` případně uživatel provede dealokaci sám.

Odeslání CAN zprávy

```
CAN_MESSAGE message;

// Vyplneni datovych polozek zpravy
message.Id = 0x18DA17F1;
message.st_ext = true;
message.fd_can = false;
CANMsgUpdateFrom29(&message);

message.rtr = false;
message.length = 8;

message.data[0] = 1;
message.data[1] = 2;
message.data[2] = 3;
message.data[3] = 4;
message.data[4] = 5;
message.data[5] = 6;
message.data[6] = 7;
message.data[7] = 8;

//USB2CAN_SelectActualAdapter(USB2CAN_adapter);
X2CAN_SendCANMessage(message);
```

Identifikátor CAN zprávy je možné zadat jako Id (29b) nebo Id1+Id2 (11+18b). Pro vnitřní synchronizaci obou variant jsou určené funkce `CANMsgUpdateFrom29` a `CANMsgUpdateFrom11_18`.

Čtení přijatých zpráv

```
CAN_MESSAGE message;
bool msg = X2CAN_GetCANMessage(&message);
if(msg)
{
}
```

Případně je take možné použít funkci:

```
X2CAN_DLLMAPPING bool USB2CAN_TRIPLE_WaitForRxMessage(const unsigned int timeout);
```

Parametrem je doba čekání na příchod zprávy. Nekonečné čekání je možné zadat parametrem `WAIT_INFINITE`. Pokud je čekání ukončeno, vrací v případě přijetí zprávy hodnotu true. Pokud vypršel timeout, je vrácena hodnota false.

Uložení dat do zprávy a jejich dekódování zpět

Obvykle se pro přenos dat používají celočíselné typy dat. Pokud se používá proměnná, kdy jsou třeba přenášet desetiny, setiny nebo je třeba přenášet data ještě jemněji, nepoužívá se zpravidla typ float ale integer se zadanou váhou bitu – rozlišením, případně offsetem.

Taktéž není pravidlem, že se používají datové typy s bitovou délkou v násobku 8, ale v praxi mohou mít data nejrůznější délku, například 12 nebo 15 bitový integer a podobně.

Pro snadné vkládání dat jsou v API obsaženy funkce:

```
void storeSignal(uint8_t* frame, uint64_t value, const uint8_t startbit, const uint8_t
length, bool is_big_endian, bool is_signed);

void storeSignalFD(uint8_t* frameDataClassic8, uint8_t* frameDataFD56, uint64_t value,
const uint16_t startbit, const uint16_t length, bool is_big_endian, bool is_signed);
```

Pro vyčtení dat jsou v API obsaženy funkce:

```
uint64_t extractSignal(const uint8_t* frame, const uint8_t startbit, const uint8_t
length, bool is_big_endian, bool is_signed);

uint64_t    extractSignalFD(const    uint8_t*    frameDataClassic8,    const    uint8_t*
frameDataFD56, const uint16_t startbit, const uint8_t length
, bool is_big_endian, bool is_signed);
```

`frameDataClassic8` - ukazatel na prvních 8 datových bajtů CAN zprávy, v případě struktury `CAN_MESSAGE` tedy `&message.data[0]`

`frameDataFD56` – ukazatel na dalších 56 datových bajtů CAN FD zprávy, v případě struktury `CAN_MESSAGE` tedy `message.fd_data`
`startbit` – pozice prvního bitu kam se vkládají data, pozor na big endian

`length` – delka dat (maximálně 64 bitů)

`is_big_endian` – true pokud se jedná o big endian

`is_signed` – pokud je true, jedná se o integer se znaménkem

Rychlosti CANu podporované převodníky

Převodníky podporují jak běžně používané rychlosti CAN sběrnice, tak i některé méně rozšířené. Nicméně některé převodníky nemusí konkrétní hodnoty nestandardních CAN rychlostí podporovat.

X2CAN API podporuje funkce pro zjištění seznamu podporovaných rychlostí.

Příklad naplnění Combo boxu seznamem podporovaných rychlostí:

```
//Parametrem je ukazatel na combo box, typ interface a číslo CAN portu
//pro který se seznam čte.
void FillSpeedComboBox(CComboBox* combo, CAN_INTERFACE iface, int port)
{
	combo->ResetContent();
	//Celkový počet podporovaných rychlostí.
	int size = CANSpeedListSize(iface,port);
	for(int i=0;i<size;i++)
	{
		CString s;
		//vrátí podporovanou rychlost - převedu na řetězec a ten
		//vloží do combo boxu.
		switch(Int2CANSpeed(i,iface,port))
		{
			case SPEED_10k:		s="10k";break;
			case SPEED_12_5k:	s="12.5k";break;
			case SPEED_20k:		s="20k";break;
			case SPEED_25k:		s="25k";break;
			case SPEED_33_3k:	s="33.3k";break;
			case SPEED_40k:		s="40k";break;
			case SPEED_50k:		s="50k";break;
			case SPEED_62_5k:	s="62.5k";break;
			case SPEED_83_3k:	s="83.3k";break;
			case SPEED_100k:	s="100k";break;
			case SPEED_125k:	s="125k";break;
			case SPEED_200k:	s="200k";break;
			case SPEED_250k:	s="250k";break;
			case SPEED_333k:	s="333.3k";break;
			case SPEED_500k:	s="500k";break;
			case SPEED_667k:	s="666.6k";break;
			case SPEED_800k:	s="800k";break;
			case SPEED_833k:	s="833.3k";break;
			case SPEED_1M:		s="1M";break;
			case SPEED_1_25M:	s="1.25M";break;
			case SPEED_1_50M:	s="1.50M";break;
			case SPEED_2M:		s="2M";break;
			case SPEED_USR:		s="USER TM";break;
		}
```

```
                combo->InsertString(i,s.GetBuffer(0));
        }
}
```

CAN bus 2.0 podporuje maximální rychlost 1Mbit/s. Nicméně ještě pro rychlost 2Mbit/s je jej možné používat na krátkou vzdálenost. Nastavení 2M se prak hodí v případech kdy je potřeba odsimulovat si velké zatížení vyvíjeného zařízení.