ETH2CAN – CAN firmware

Obsah:

Ing. David Španěl

Mgr. Vítězslav Rejda

CANLAB s.r.o.

Základní popis

Interface ETH2CAN s firmwarem CAN je určen pro připojení ke sběrnici CAN prostřednictvím sítě ethernet. Rychlost ethernetového rozhraní je 10Mbitu.

Práce s tímto zařízením je možná prostřednictvím software PP2CAN, API X2CAN nebo vlastní implementací komunikace prostřednictvím socketu. Adaptér podporuje současné připojení až 4 klientů. Je tak možno například číst data z CANu paralelně na 4 PC a z jakéhokoliv tohoto PC odesílat data na CAN.

Komunikace po rozhraní ETHERNET

Zařízení má svoji IP adresu a TCP port prostřednictvím kterého probíhá veškerá komunikace. Chová se jako server, tedy klient se připojuje k tomuto zařízení.

Pro komunikaci se používá několika paketů, každý paket obsahuje položku magic, ta je určena k ověření endianu zařízení. Má hodnotu 0xAA123456. Dále pak položku packet_type, ta rozlišuje typ paketu. Položka packet_size pak určuje velikost paketu. Položka id je určena k identifikaci paketu. Je li například zaslán do zařízení dotaz, je možné nastavit položku id na libovolnou hodnotu. Paket s odpovědí pak má nastavenou stejnou hodnotu. Lze tak rozlišit dvě odpovědi od sebe při zaslání dvou požadavků.

Typy paketů:

```
#define   PACKET_UNKNOWN_PACKET_ID      0
#define   PACKET_RUN                    1
#define   PACKET_MODE                   3
#define   PACKET_RESET                  4
#define   PACKET_CLOSE                  16
#define   PACKET_DATA                   32
#define   PACKET_STATUS                 128
#define   PACKET_REBOOT_DATA            129
#define   PACKET_UNLOCK_SETTINGS        140 *
#define   PACKET_SETTINGS_ACK           141 *
#define   PACKET_FULL_SETTINGS_EEPROM   251
#define   PACKET_FULL_SETTINGS          252
#define   PACKET_SERIAL_NUMBER          253
#define   PACKET_FIRMWARE_VERSION       254
#define   PACKET_CONFIGURATION          255

   * od verze FW 1.12
```

Podpora paketů:

Paket	Bootloader	Aplikace
PACKET_UNKNOWN_PACKET_ID	Y	Y
PACKET_RUN	Y	Y
PACKET_MODE	Y	Y
PACKET_RESET	Y	Y
PACKET_CLOSE	N	Y
PACKET_DATA	N	Y
PACKET_STATUS	N	Y
PACKET_REBOOT_DATA	Y	N
PACKET_FULL_SETTINGS_EEPROM	N	Y
PACKET_FULL_SETTINGS	N	Y
PACKET_SERIAL_NUMBER	Y (jen čtení)	Y
PACKET_FIRMWARE_VERSION	Y	Y
PACKET_SETTINGS	Y (jen čtení)	Y

Struktury paketů je třeba zarovnat na 1 bajt. Tedy bez jakékoliv další výplně. V jazyce C/C++ v prostředí Microsoft Visual Studia je toho možné dosáhnout například takto:

```
#pragma pack(push, 1)
typedef struct _ETH_HEADER {
          unsigned __int32 magic;
          unsigned char packet_type;
          unsigned __int16 packet_size;
          unsigned char id;
} ETH_HEADER;
#pragma pack(pop)
```

Paket UNKNOWN_PACKET_ID

Paket ve směru ETH2CAN -> Klient (nadřazený systém).

```
typedef struct _ETH_UNKNOWN_PACKET_ID {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       unknown_packet_type;
}    ETH_ UNKNOWN_PACKET_ID;

packet_type = PACKET_UNKNOWN_PACKET_ID
```

Paket je vrácen, pokud interface obdrží paket s neznámou hodnotou packet_type. Například v režimu bootloaderu je vrácen po zaslání paketu FMS nebo DTI, v režimu aplikace při zaslání REBOOT_DATA.

Paket RUN

Paket ve směru Klient (nadřazený systém) -> ETH2CAN.

```
typedef struct _ETH_RUN {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH_RUN;

packet_type = PACKET_RUN
```

Paket je určen k aktivaci zařízení. Po připojení zařízení k napájení nachází v režimu bootloaderu. Tento režim je určen pro snadnou změnu firmware v zařízení. Zasláním tohoto paketu dojde k aktivaci firmware. Bootloader je automaticky přepnut do režimu firmware po uplynutí intervalu 30 sekund, pokud bootloader nedetekuje příchod paketu PACKET_REBOOT_DATA.

Paket ve směru ETH2CAN -> Klient (nadřazený systém).

```
typedef struct _ETH_RUN2 {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       mode;
}    ETH_RUN2;

packet_type = PACKET_RUN
```

Paket je generován jako odpověď na příchozí paket ETH_RUN. Potvrzuje přijetí tohoto paketu a zároveň v položce mode signalizuje aktuální režim firmware (bootloader: mode = 1, application-firmware: mode = 2)

Paket MODE

Paket ve směru Klient (nadřazený systém) -> ETH2CAN.

```
typedef struct _ETH_MODE {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH_MODE;

packet_type = PACKET_MODE
```

Paket je určen k vyžádání aktuálního režimu firmware (bootloader/aplikace).

Paket ve směru ETH2CAN -> Klient (nadřazený systém).

```
typedef struct _ETH_MODE2 {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       mode;
}    ETH_MODE2;

packet_type = PACKET_MODE
```

Paket je generován jako odpověď na příchozí paket ETH_MODE. Potvrzuje přijetí tohoto paketu a zároveň v položce mode signalizuje aktuální režim firmware (bootloader: mode = 1, application: mode = 2).

Paket RESET

```
typedef struct _ETH_SHUTDOWN {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH_RESET;

packet_type = PACKET_RESET
```

Paket ve směru Klient (nadřazený systém) -> ETH2CAN. Po přijetí tohoto paketu dojde k restartu firmware řídicího procesoru a spuštění režimu bootloaderu.

Paket UNLOCK

Paket ve směru Klient (nadřazený systém) ->ETH2CAN. Je do FW aplikace doplněn od verze FW 1.12.
Tento paket je nutné zaslat před každým odeslání paketu PACKET_CONFIGURATION nebo PACKET_FULL_SETTINGS. Pokud převodník s FW 1.12 obdrží paket s konfigurací bez toho, aby byl použit paket UNLOCK pro odemknutí konfigurace, je požadavek odmítnut odpovědí ACK.
Paket je určen pro lepší ochranu proti přepsání konfigurace v EEPROM.

```
typedef struct {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       data[4];
} ETH2CAN_UNLOCK;

packet_type = PACKET_UNLOCK_SETTINGS
```

data[0] - pevná hodnota 0x14
data[1] - pevná hodnota 0xA4
data[2] - pevná hodnota 0x45
data[3] - pevná hodnota 0xF1

Paket ACK

Paket ve směru Klient (nadřazený systém) ->ETH2CAN. Je do FW aplikace doplněn od verze FW 1,12.

```
typedef struct {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
```

```
    unsigned char       ack_data;
} ETH2CAN_UNLOCK;

packet_type = PACKET_SETTINGS_ACK
```

- potvrzuje paket CONFIGURATION, v případě kladného potvrzení změny, odesílá v datech jediný datový bajt s hodnotou 0, při chybě má hodnotu nenulovou.
- potvrzuje paket FULL_SETTINGS, v případě chyby (při kladné odpovědi se posílá zpět celé nastavení)

Paket CONFIGURATION

Paket ve směru Klient (nadřazený systém) ->ETH2CAN. Nastavuje novou konfiguraci zařízení. Zařízení odpovídá zasláním tohoto paketu zpět.

```
typedef struct {
    unsigned __int32    magic;
    unsigned char       packet_type;
    unsigned __int16    packet_size;
    unsigned char       id;
    unsigned char       can_speed;
    unsigned char       listen_only;
    unsigned char       app_mode;
} ETH2CAN_CONFIGURATION;

packet_type = PACKET_CONFIGURATION
```

Paket ve směru ETH2CAN -> Klient (nadřazený systém).
Slouží k zjištění aktuální konfigurace zařízení:

```
typedef struct _ETH2CAN_SETTINGS_REQ {
    unsigned __int32    magic;
    unsigned char       packet_type;
    unsigned __int16    packet_size;
    unsigned char       id;
}   ETH2CAN_CONFIGURATION_REQ;

packet_type = PACKET_CONFIGURATION
```

can_speed – rychlost CAN sběrnice, hodnoty

0	10k
1	20k
2	33.3k
3	50k
4	62.5k
5	83.3k

6	100k
7	125k
8	250k
9	500k
10	1M

listen_only
0 normální mód (připojení na FMS bránu)
1 listen only mód (připojení na CAN bus vozidla, motorový CAN)

app_mode
položka určena pro budoucí použití

Paket PACKET_FULL_SETTINGS

Paket ve směru Klient (nadřazený systém) ->ETH2CAN. Nastavuje novou plnou konfiguraci zařízení. Zařízení odpovídá zasláním tohoto paketu zpět.

```
typedef struct {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       can_speed;
     unsigned char       listen_only;
     unsigned char       app_mode;
     unsigned char       ip[4];
     unsigned char       mask[4];
     unsigned char       mac[6];
     unsigned __int16    port;
     unsigned char       max_connections;
} ETH2CAN_SETTINGS;

packet_type = PACKET_FULL_SETTINGS
```

Paket ve směru ETH2CAN -> Klient (nadřazený systém). Paket s nastavením a dotazem na nastavení mají stejnou hodnotu položky packet_type, liší se však délkou hlavičky. Paket s dotazem má pouze délku hlavičky.

Slouží k zjištění aktuální plné konfigurace zařízení:

```
typedef struct _ETH2CAN_SETTINGS_REQ {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH2CAN_SETTINGS_REQ;

packet_type = PACKET_FULL_SETTINGS
```

can_speed – rychlost CAN sběrnice, hodnoty

0	10k
1	20k
2	33.3k

6	100k
7	125k
8	250k

3	50k
4	62.5k
5	83.3k

9	500k
10	1M

listen_only
0 normální mód (připojení na FMS bránu)
1 listen only mód (připojení na CAN bus vozidla, motorový CAN)

app_mode
položka určena pro budoucí použití

ip
IP adresa zařízení. Defaultně přednastavena na 192.168.12.150. Je možné však vyžádat z výroby jinou hodnotu.

mask
Maska adresy zařízení. Defaultně přednastavena na 255.255.255.0. Je možné však vyžádat z výroby jinou hodnotu.

port
TCP port na kterém probíhá komunikace. Defaultně 3000.

max_connections
Počet paralelních spojení klientů se zařízením ETH2CAN. Povolené hodnoty 2..4. Menší počet spojení dovoluje zrychlit množství přenášených dat.

mac
MAC adresa zařízení. Defaultně 00-04-A3-00-00-00.

Paket FIRMWARE VERSION

Tímto paketem jsou je vyžadována verze firmware v zařízení interface ETH2CAN.

Požadavek klienta má tvar:

```
typedef struct _ETH_FIRMWARE_REQ {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH_FIRMWARE_REQ;

packet_type = PACKET_FIRMWARE_VERSION
```

Odpověď interface ETH2CAN má tvar:

```
typedef struct _ETH_FIRMWARE {
     unsigned __int32    magic;
     unsigned char       packet_type;
```

```
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char[20]   fw_version_string;
}    ETH_FIRMWARE;

packet_type = PACKET_FIRMWARE_VERSION
```

Paket s požadavkem na čtení verze firmware a paket s odpovědí s verzí FW mají stejnou hodnotu položky packet_type, liší se však délkou paketu. Požadavek má jen délku hlavičky paketu.

Položka obsahuje string s verzí firmware. Neobsahuje ukončovací 0 řetězce. Řetězec má tvar například CANLABsro-01.10. V režimu bootloaderu pak CANLABsro-01.10boot. Bootloader využívá jiné číslování než aplikace!

Paket SERIAL NUMBER

Tento paket je určen k přečtení sériového čísla interface ETH2CAN.

Požadavek klienta má tvar:

```
typedef struct _ETH_SERNUM _REQ {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
}    ETH_SERNUM _REQ;

packet_type = PACKET_SERIAL_NUMBER
```

Odpověď interface ETH2CAN má tvar:

```
typedef struct _ETH_SERNUM {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char[14]   serial_number;
}    ETH_SERNUM;

packet_type = PACKET_SERIAL_NUMBER
```

Položka obsahuje string se sériovým číslem interface. Sériové číslo má tvar E2Cxxxxxxxxx. První 3 znaky jsou vždy E2C. Dalších 10 znak; jsou číslice, tedy sériové číslo může obsahovat hodnotu 0000000000-9999999999. Poslední, čtrnáctý znak má hodnotu 0, tedy konec řetězce.

Paket DATA

Tímto paketem jsou přenášena data z/na CAN sběrnici.

Paket má tvar:

```
typedef struct _ETH_CAN_PACKET {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       number_of_items;
     ETH_CAN_MESSAGE     item1;
     ETH_CAN_MESSAGE     item2;
     ..
     ETH_CAN_MESSAGE     itemn;
} ETH_CAN_PACKET;

typedef struct _ ETH_CAN_MESSAGE {
     unsigned __int32    id;
     unsigned char       data[8];
     unsigned char       dataLen;
     unsigned char       msgFlags;
} ETH_CAN_MESSAGE;

packet_type = PACKET_DATA
```

Položka number_of_items označuje počet CAN zpráv v paketu.
Zadaný počet zpráv následuje. Zprávy jsou přenášený ve tvaru struktury ETH_CAN_MESSAGE. Maximální počet CAN zpráv v jednom paketu je 14. Přenos více CANovských zpráv v rámci jednoho paketu zvyšuje propustnost ethernetového rozhraní.

Položka msgFlags může obsahovat bitovou masku pro zadání požadavku na přenos CAN zprávy s 29 bitovým identifikátorem a nebo přenos RTR zprávy.

```
#define ETH_CAN_XTD_FRAME     0x20 // 29 bitový ID
#define ETH_CAN_RTR_FRAME     0x40 // RTR zpráva
```

Paket STATUS

```
typedef struct _ETH_STATUS {
     unsigned __int32 magic;
     unsigned char packet_type;
     unsigned __int16 packet_size;
     unsigned char id;
} ETH_CAN_STATUS;

packet_type = PACKET_STATUS
```

Tento paket je třeba zasílat cca každých 500ms z klientské aplikace do zařízení ETH2CAN. Tímto paketem je udržováno spojení. Pokud zařízení ETH2CAN nedetekuje v intervalu 5 sekund příchod tohoto paketu, spojení ze strany ETH2CAN zrušeno.

```
typedef struct _ETH_STATUS {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned char       TEC;
     unsigned char       REC;
     unsigned __int16    RST;
     unsigned char       TXFIFO;
     unsigned char       RXFIFO;
     unsigned __int16    ETHTXFIFO;
     unsigned __int16    ETHRXFIFO;
     unsigned char       active_connections;
} ETH_STATUS;

packet_type = PACKET_STATUS
```

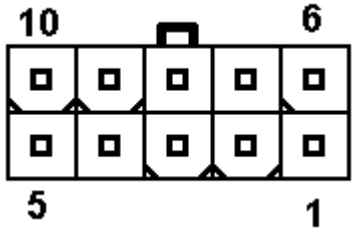
Paket je generován v intervalu cca 500ms ze zařízení ETH2CAN. Interval roste při větším zatížení, neboť generování tohoto paketu má nízkou prioritu. Klientská aplikace je tímto paketem informována o interních stavech interface ETH2CAN.

Připojení zařízení

Zařízení je uloženo v krabičce TOPTEC 102 firmy OKW. Zařízení má 2 konektory. Prvním je konektor RJ45, tedy klasický ethernetový konektor. Druhým konektorem je konektor MOLEX, který je určen při připojení napájení a vozidlových sběrnic.

Zařízení pracuje s rozsahem napájecího napětí 8-36V. Spotřeba zařízení v provozním stavu je 1.7W. V deaktivovaném stavu po odpojení signálu 15 je spotřeba rovna téměř nule. Signál 15 je aktivován cca od úrovně 1V.

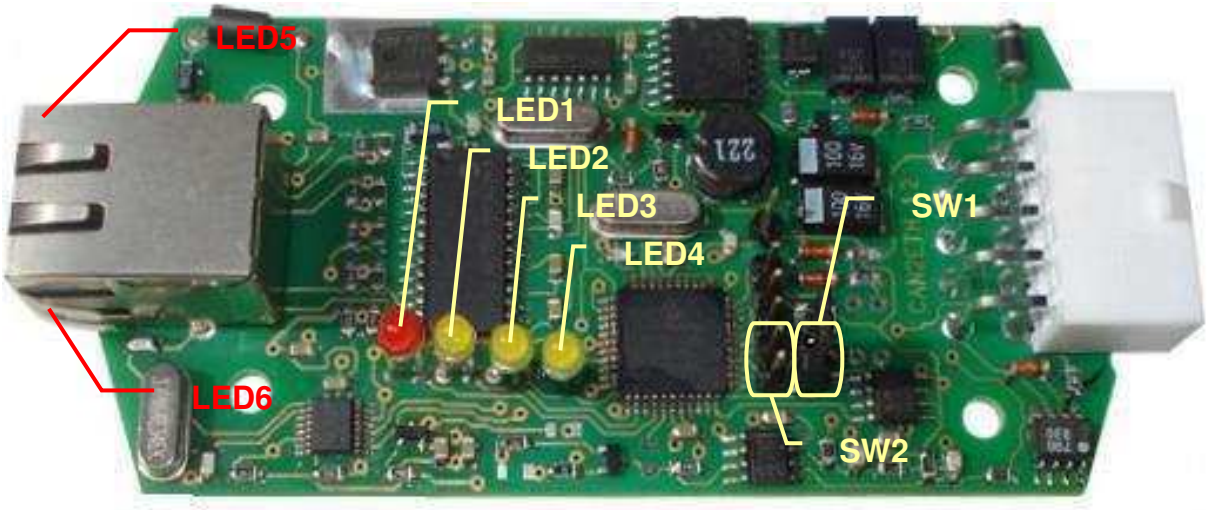
Konektor MOLEX



Konektor na PCB.

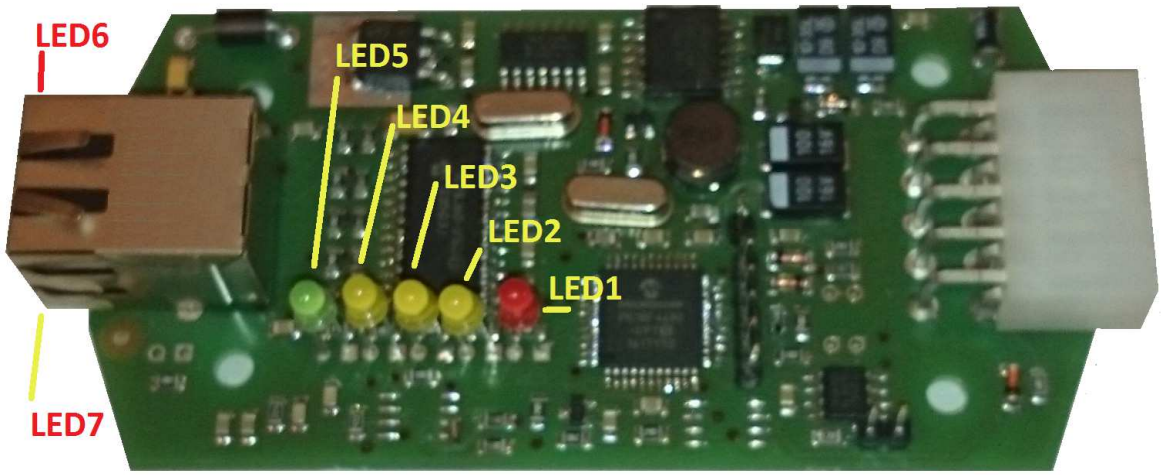
Pin	Popis
1	Napájecí napětí 8-36V
2	GND
3	CAN H
4	
5	
6	Signál 15 (startup-shutdown) – spojit s pinem 1
7	GND
8	CAN L
9	
10	

Funkce indikačních LED verze HW 1.2



LED#	Barva	Popis
1	RED	Chyba CAN sběrnice – bus off (například chybně nastavena rychlost sběrnice, není funkční v režimu Listen only). V režimu bootloaderu tato LED bliká v intervalu 1s.
2	YELLOW	Indikuje činnost CAN sběrnice, LED mění stav při příchodu CAN zprávy.
3	YELLOW	Indikuje činnost CAN sběrnice, LED mění stav při odeslání CAN zprávy.
4	YELLOW	LED mění stav při příchodu paketu STATUS.
5	GREEN	Indikuje příchod paketu (TCPIP paket, ping apod.)
6	YELLOW	Indikuje připojení ethernetového kabelu.

Funkce indikačních LED verze HW 1.3



LED#	Barva	Popis
1	RED	LED mění stav při příchodu paketu STATUS
2	YELLOW	Indikuje činnost CAN sběrnice, LED mění stav při odeslání CAN zprávy.
3	YELLOW	Indikuje činnost CAN sběrnice, LED mění stav při příchodu CAN zprávy.
4	YELLOW	Bliká v intervalu 1s v režimu bootloaderu. V režimu aplikace indikuje chybu CAN sběrnice.
5	GREEN	Indikace napájecího napětí
6	GREEN	Indikuje příchod paketu (TCPIP paket, ping apod.)
7	YELLOW	Indikuje připojení ethernetového kabelu.

Zkratovací propojka **SW1** je určena pro aktivaci zakončovacího odporu 120 ohmu na CAN sběrnici. CAN sběrnice je vždy zakončena na obou stranách zakončovacími odpory 120 ohmu.
Zkratovací propojka **SW2** je určena k resetu nastavení. Po aktivaci s touto propojkou je nastavena defaultní IP adresa 192.168.12.150.

Práce s využitím X2CAN API

API X2CAN dovoluje pracovat s adapterem ETH2CAN dvěma způsoby. První možností
je použít skupiny funkcí s předponou X2CAN která dovoluje pracovat stejně se všemi adaptéry které API X2CAN podporuje. Druhou variantou jsou funkce s předponou ETH2CAN které obsahují stejné funkce jako první skupina, navíc však obsahují i další specializované funkce.

```
X2CAN_DLLMAPPING bool X2CAN_Open_ETH2CAN(CAN_SPEED speed, void
(*error_function)(int err_code, const char * error_string),
unsigned char ip[4], unsigned int port);
```

Funkce vytvoří spojení s ETH2CAN.

```
X2CAN_DLLMAPPING void* ETH2CAN_PrepareAdapter(unsigned char
ip[4], unsigned int port);
```

Vytvoří strukturu s konfiguračními daty adaptéru a vrátí ukazatel na ni.

```
X2CAN_DLLMAPPING void ETH2CAN_DestroyAdapter(void* adapter);
```

Odstraní strukturu konfiguračních dat adaptéru.

```
X2CAN_DLLMAPPING void ETH2CAN_SelectActualAdapterAccess(void*
adapter);
```

Funkce slouží k vybrání adaptéru pokud se pracuje s několika paralelně. Taktéž v případě práce s jedním adaptérem je nutno před voláním Open zavolat tuto funkci a nastavit tento adaptér ETH2CAN.

```
X2CAN_DLLMAPPING                                        void
ETH2CAN_UnselectActualAdapterAccess(void);
```

Zrušení výběru adaptéru.

```
X2CAN_DLLMAPPING bool ETH2CAN_Open(CAN_SPEED speed, void
(*error_function)(int err_code, const char * error_string) ,
unsigned char ip[4] , unsigned int port);
```

Funkce vytvoří spojení s ETH2CAN.

```
X2CAN_DLLMAPPING bool ETH2CAN_Close();
```

Uzavře spojení s adaptérem.

```
X2CAN_DLLMAPPING bool ETH2CAN_SendCANMessage(CAN_MESSAGE
message);
```

Odeslání zprávy na CAN.

```
X2CAN_DLLMAPPING bool ETH2CAN_GetCANMessage(CAN_MESSAGE *message);
```

Čtení dat z CANu.

```
X2CAN_DLLMAPPING unsigned char ETH2CAN_GetREC(void)
```

Čtení chybového registr REC.

```
X2CAN_DLLMAPPING unsigned char ETH2CAN_GetTEC(void)
```

Čtení chybového registr TEC.

```
X2CAN_DLLMAPPING void ETH2CAN_SendSetFullSettings(CAN_SPEED speed,bool listen_only, unsigned char ip[4], unsigned char mask[4], unsigned char mac[6], unsigned int port, unsigned char max_connections);
```

Nové plné nastavení adaptéru včetně IP adresy, MAC adresy a podobně.

```
X2CAN_DLLMAPPING void ETH2CAN_SendInterfaceReset(void);
```

Resetuje adaptér. API jej opět automaticky připojí pokud není zavoláno Close.

Změny ve verzích firmware

1.13 boot

- první distribuční verze bootloadru

1.00

- první distribuční verze FW