



## ETH2CAN – FMS firmware

### Obsah:

<b>ZÁKLADNÍ POPIS</b>	<b>2</b>
<b>KOMUNIKACE PO ROZHRANÍ ETHERNET</b>	<b>3</b>
Paket UNKNOWN_PACKET_ID	4
Paket RUN	4
Paket MODE	5
Paket SET_TIMESTAMP	5
Paket GET_TIMESTAMP	6
Paket PACKET_REBOOT_DATA	6
Paket SHUTDOWN , RESET	7
Paket TACHOGRAPH_CONNECTION	8
Paket CONFIGURATION	9
Paket FIRMWARE VERSION	11
Paket SERIAL NUMBER	11
Paket FMS	12
Paket FMS_EXT	15
Paket DTI	17
Paket J1708	18
Paket J1708_EXT	19
<b>PŘIPOJENÍ ZAŘÍZENÍ</b>	<b>20</b>
<b>PŘÍKAZ SETTINGS U NEJČASTĚJI MONITOROVANÝCH VOZIDEL.</b>	<b>22</b>
<b>ZMĚNA FW UŽIVATELSKÝM PROGRAMEM</b>	<b>23</b>
<b>TESTOVACÍ APLIKACE</b>	<b>24</b>
<b>ZMĚNY VE VERZÍCH FIRMWARE</b>	<b>25</b>
1.13 boot	25
1.13	25
1.20 boot	25
1.20	25
1.21	25
1.23 boot	25
1.23	25
1.30 boot	25
1.30	26
1.45	26
1.60	26
2.00	26

Ing. David Španěl

Mgr. Vítězslav Rejda

CANLAB s.r.o.

## Základní popis

FMS firmware pro interface ETH2CAN je určeno pro poskytování vozidlových provozních dat z nákladních a osobních vozidel. Pro připojení do vozidla je interface vybaven těmito rozhraními:

- CAN bus (high speed)
- Interface digitálního tachografu DTCO1381
- Rozhraní pro sběrnici J1708 (starší vozidla)

Pro připojení do nadřazeného systému je pak určeno rozhraní ETHERNET o rychlosti 10Mbitu.

Zařízení neposkytuje veškerá data uvedená v datových paketech ale jen data dostupná. Dostupnost pak závisí na typu, výrobci, modifikaci a roku výroby vozidla.



## 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. Ve vývojovém stádiu se používá hodnota položky packet\_size nastavená na hodnotu 0xFFFF. Toto nastavení udává, že velikost není signalizována a aplikace klienta musí zjistit velikost paketu pomocí sizeof struktury. 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_SHUTDOWN 2
#define PACKET_MODE 3
#define PACKET_RESET 4
#define PACKET_SET_TIMESTAMP 8
#define PACKET_GET_TIMESTAMP 9
#define PACKET_FMS 32
#define PACKET_DTI 33
#define PACKET_FMS_EXT 34
#define PACKET_J1708 35
#define PACKET_J1708_EXT 36
#define PACKET_REBOOT_DATA 129
#define PACKET_GET_TACHOGRAPH_CONNECTION 249
#define PACKET_RESP_TACHOGRAPH_CONNECTION 249
#define PACKET_SET_TACHOGRAPH_CONNECTION 250
#define PACKET_SERIAL_NUMBER 253
#define PACKET_FIRMWARE_VERSION 254
#define PACKET_CONFIGURATION 255
```

\* ZMĚNA ve verzi FW od 2.00, revize HW 1.30

Podpora paketů:

Paket	Bootloader	Aplikace
PACKET_UNKNOWN_PACKET_ID	Y	Y
PACKET_RUN	Y	Y
PACKET_SHUTDOWN	Y	Y
PACKET_MODE	Y	Y
PACKET_RESET	N	Y
PACKET_SET_TIMESTAMP	N	Y

PACKET_GET_TIMESTAMP	N	Y
PACKET_FMS	N	Y
PACKET_DTI	N	Y
PACKET_FMS_EXT	N	Y
PACKET_REBOOT_DATA	Y	N
PACKET_SERIAL_NUMBER	Y (jen čtení)	Y
PACKET_FIRMWARE_VERSION	Y	Y
PACKET_CONFIGURATION	Y (jen čtení)	Y
PACKET_GET_TACHOGRAPH_CONNECTION	N	Y
PACKET_RESP_TACHOGRAPH_CONNECTION	N	Y
PACKET_SET_TACHOGRAPH_CONNECTION	N	Y

### **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;
```

Paket je vrácen pokud interface obdrží paket s neznámou hodnotou packet\_type. Například v režimu bootladeru 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;
```

Paket je určen k aktivaci zařízení. Po připojení zařízení k napájení a připojení signálu 15 se zařízení nachází v režimu bootladeru. Tento režim je určen pro snadnou změnu firmware v zařízení. Zasláním tohoto paketu dojde k aktivaci firmware. Bootlader je automaticky přepnut do režimu firmware po uplynutí intervalu 30 sekund, pokud bootlader 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;

```

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;

```

Paket je určen k vyžádání aktuálního režimu firmware.

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;

```

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 SET\_TIMESTAMP***

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

```

typedef struct _SET_TIMESTAMP {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
}

```

```

    unsigned char    id;
    unsigned __int16 timestamp;
}
SET_TIMESTAMP;

```

Paket je určen k přenastavení čítače timestampu. Timestamp je položka nastavená pro většinu měřených veličin z CAN sběrnice a udává stáří veličiny. Hodnota je inkrementována každých 100ms. Po startu je hodnota timestampu nastavena na hodnotu 0.

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

```

typedef struct _SET_TIMESTAMP2 {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
}
SET_TIMESTAMP 2;

```

### ***Paket GET\_TIMESTAMP***

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

```

typedef struct _GET_TIMESTAMP {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
}
GET_TIMESTAMP;

```

Paket je určen k přečtení aktuální hodnoty čítače timestampu. Timestamp je položka nastavená pro většinu měřených veličin z CAN sběrnice a udává stáří veličiny. Hodnota je inkrementována každých 100ms.

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

```

typedef struct _GET_TIMESTAMP2 {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned __int16   timestamp;
}
GET_TIMESTAMP 2;

```

Paket je generován jako odpověď na příchozí paket SET\_TIMESTAMP. Potvrzuje přijetí tohoto paketu a vrací zpět hodnotu čítače timestampu.

### ***Paket PACKET\_REBOOT\_DATA***

Paket ve směru Klient (nadřazený systém) -> ETH2CAN.  
Paket přenáší v režimu bootloderu 1 řádek HEX souboru.

```
typedef struct _ETH_REBOOT_DATA {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned char      data[64];
}    ETH_REBOOT_DATA;
```

Paket ve směru ETH2CAN -> Klient (nadřazený systém).  
Paket potvrzuje přijetí a naprogramování zasláného řádku HEX souboru, tím signalizuje připravenost k přijetí dalšího řádku.

```
typedef struct _ETH_REBOOT_ACK {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned char      error_code;
    unsigned char      dummy[68];
}    ETH_REBOOT_ACK;
```

Po zaslání celého souboru se nový FW aktivuje paketem RUN.

Položka error\_code:

- 0 – OK
- 1 – chybná adresa
- 2 – chybná délka dat
- 3 – chyba zápisu do flash
- 4 - chyba verifikace zápisu

### ***Paket SHUTDOWN , RESET***

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

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

Položka why:

- 0 – na požadavek klienta
- 1 – ethernet watchdog

## 2 – vypnutí signálu 15

Tímto paketem zařízení indikuje ukončení činnosti po odpojení signálu 15 nebo restart zařízení.

```
typedef struct _ETH_SHUTDOWN {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
}    ETH_SHUTDOWN2;
```

Paket ve směru Klient (nadřazený systém) -> ETH2CAN. Po přijetí tohoto paketu dojde k restartu firmware řídicího procesoru. Aplikace (ne bootloader) podporuje variantu příkazu RESET která resetuje pouze aplikaci.

### ***Paket TACHOGRAPH\_CONNECTION***

```
typedef struct _ETH_TCH_CONNECTION_SET {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned char      tachograph_connection;
}    ETH_TCH_CONNECTION_SET;
```

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

```
tachograph_connection = 0    - OFF
                        1    - AUTO (FW sam určí připojení)
                        2    - MAX3100
                        3    - Direct
```

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

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

```
typedef struct _ETH_TCH_CONNECTION {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned char      tachograph_connection_actual;
    unsigned char      tachograph_connection_EEPROM;
}    ETH_TCH_CONNECTION;
```



tachograph\_connection\_actual - aktuálně použité nastavení  
 tachograph\_connection\_EEPROM - nastavení v EEPROM (nastavené pomocí ETH\_TCH\_CONNECTION\_SET) které bude použito po restartu.

## **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       st_ext;
    unsigned char       ip[4];
    unsigned int        port;
    unsigned __int16    startup_timeout;
    unsigned __int16    shutdown_timeout;
    unsigned __int16    eth_watchdog;
    unsigned char       mac[6];
    unsigned char       ipmask[4];
    unsigned __int16    app_start_timeout;
    unsigned char       mask[4];
    unsigned char       tachograph_mode;
} ETH2CAN_SETTINGS;
```

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_SETTINGS_REQ;
```

**can\_speed** – rychlost CAN sběrnice, hodnoty

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

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

5	83.3k
---	-------

**listen\_only**

0 normální mód (připojení na FMS bránu)

1 listen only mód (připojení na CAN bus vozidla, motorový CAN)

**st\_ext**

0 standardní identifikátory

1 rozšířené identifikátory

**ip**

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

**port**

TCP port na kterém probíhá komunikace. Defaultně 3000.

**startup\_timeout**

Opoždění aktivace zařízení po připojení signálu 15. Zpoždění eliminuje aktivaci zařízení při krátké aktivaci signálu 15. Čas nastavován v sekundách. Rozsah 1..200s. Defaultně 5 s.

**shutdown\_timeout**

Opoždění deaktivace zařízení po odpojení signálu 15. Zpoždění eliminuje deaktivaci zařízení při krátkém vypnutí signálu 15. Čas nastavován v sekundách. Rozsah 1..200s. Defaultně 5 s.

**eth\_watchdog**

Timeout v sekundách. Pokud po uvedený čas není detekována aktivita klienta dojde k resetu zařízení. Je-li nastavena hodnota 0, funkce není aktivní. Rozsah 20..300s.

**mac**

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

**app\_start\_timeout**

Čas po kterém je bootloader automaticky přepnut do aplikace pokud není přijat paket, který provádí změnu firmware .

Po přijetí tohoto paketu je nové nastavení uloženo do interní EEPROM interface. Aby se nové nastavení uplatnilo, je nutné restartovat firmware příkazem shutdown nebo reset.

**mask**

Maska sítě. Defaultně přednastavena na 255.255.255.0.

**tachograph\_mode**

Nastavení typu připojeného digitálního tachografu, 0-VDO Siemens, 1-Stoneridge, 2-Actia.

## ***Paket FIRMWARE VERSION***

Tímto paketem jsou 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;
```

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;
```

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;
```

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;

```

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

## **Paket FMS**

Tímto paketem jsou vyžadována data čtená ze sběrnice CAN.

Požadavek klienta má tvar:

```

typedef struct _ETH_FMS_REQ {
    unsigned __int32    magic;
    unsigned char       packet_type;
    unsigned __int16    packet_size;
    unsigned char       id;
}    ETH_FMS_REQ;

```

Odpověď interface ETH2CAN má tvar:

```

typedef struct _ETH_FMS {
    unsigned __int32    magic;
    unsigned char       packet_type;
    unsigned __int16    packet_size;
    unsigned char       id;
    unsigned __int16    rpm;
    unsigned __int16    speed;
    unsigned char       acc_pedal;
    unsigned char       brake_pedal;
    unsigned __int32    total_fuel_used;
    unsigned __int32    total_engine_hours;
    unsigned char       fuel_level;
    unsigned __int16    fuel_consumption;
    unsigned char       axle_weight_captured[12];
    unsigned char       axle_weight_location[12];
    unsigned __int16    axle_weight[12];
    unsigned __int32    total_vehicle_distance;
    unsigned __int16    daily_vehicle_distance;
    unsigned __int16    service_distance;
    unsigned char       engine_coolant_temperature;
    unsigned __int16    tachograph_speed;
    unsigned char       tachograph[4];
    unsigned char       tire_pressure_captured[12];
    unsigned char       tire_pressure_location[12];
    unsigned char       tire_pressure[12];
    unsigned __int16    door;
    unsigned __int16    fuel_instantaneous;
}

```

```

unsigned __int16    fuel_rate;

//ROZSIRENI - DETEKOVAT NA ZAKLADE VELIKOSTI PAKETU
unsigned char      secondary_fuel_level;
unsigned long      hires_total_fuel_used;

unsigned char      percent_torque;
unsigned char      service_brake_air_pressure[2];
unsigned char      diesel_exhaust_fluid_level;
unsigned char      tell_tale_status[8*4];
unsigned char      clutch_brake_cruise_control;
unsigned char      engine_load_at_speed;
unsigned __int16   gross_combination_vehicle_weight;
unsigned char      retarder_torque_mode;
unsigned char      actual_retarder_percent_torque;
unsigned char      retarder_selection_non_engine;
unsigned char      air_suspension_control[8];
unsigned char      selected_gear;
unsigned char      current_gear;
unsigned char      door2[8];

} ETH_FMS;

* ZMĚNA ve verzi FW od 2.00, revize HW 1.30
* ZMĚNA ve verzi FW od 1.60, revize HW 1.20
* ZMĚNA ve verzi FW od 2.12, revize HW 1.30

```

Data ze struktury je možné převést na skutečné hodnoty pomocí této tabulky:

Data	Počet bitů	Váha 1 bitu	Offset
Rychlost	16	1/256 km/h	0
Poloha pedálu akcelerace	8	0.4 %	0
Poloha brzdového pedálu	8	0.4 %	0
Celkově spotřebované palivo	32	0.5 litru	0
Celkově spotřebované palivo - HIRES	32	0.001 litru	0
Stav palivové nádrže	8	Truck:0.4 VW:1litr	0
Otáčky motoru	16	0.125 otáčky	0
Zatížení nápravy	16	0,5 kg	0
Celkový počet motohodin	32	0,05 hod.	0
Celkově najeté kilometry	32	5 m	0
Vzdálenost v kilometrech do servisní prohlídky	16	5 km	-160 635
Teplota chladící kapaliny.	8	1°C	-40
Průměrná spotřeba.	16	1/512 km/L	0

## Axle weight

Položka `axle_weight_location[x]` udává lokaci hodnoty zatížení nápravy v položce `axle_weight[x]`. Hodnota `axle_weight_captured[x] = 0` udává, že položka neobsahuje žádnou (platnou) hodnotu, `axle_weight_captured[x] = 1` udává, že položka obsahuje platnou hodnotu.

V položce `axle_weight_location[x]` je zakódována informace o čísle měřené nápravy a kole této nápravy. Dolní 4 bity udávají index kola, horní 4 bity udávají index nápravy. Jsou-li všechny 4 bity nastaveny na 1, lokace není známa.

### **Položka tachograph[4]**

Tato položka obsahuje informace, které je možno dekodovat dle následujícího popisu:

#### ***tachograph[0]***

*Bit 2..0 :Driver 1 working state*

- 000 = Rest
- 001 = Driver available
- 010 = Work
- 011 = Drive
- 110 = Error
- 111 =

*Bit 5..3 :Driver 2 working state*

- 000 = Rest
- 001 = Driver available
- 010 = Work
- 011 = Drive
- 110 = Error
- 111 = not available

*Bit 7..6 :Drive recognize*

- 00 = Vehicle motion not detected
- 01 = vehicle motion

#### ***tachograph[1]***

*Bit 3..0 : Driver 1 time rel states*

- 0000 = normal
- 0001 = 15 min bef. 4 ½ h
- 0010 = 4 ½ h reached
- 0011 = 15 min bef. 9 h
- 0100 = 9 h reached
- 0101 = 15 min bef. 16 h
- 0110 = 16h reached
- 1110 = Error
- 1111 = not available

*Bit 5..4 :Driver 1 card*

- 00 = Card not present
- 01 = Card present

*Bit 7..6 :Overspeed*

- 00 = No overspeed
- 01 = Overspeed

#### ***tachograph[2]***

*Bit 3..0 : Driver 2 time rel states*

- 0000 = normal

0001 = 15 min bef. 4 ½ h  
 0010 = 4 ½ h reached  
 0011 = 15 min bef. 9 h  
 0100 = 9 h reached  
 0101 = 15 min bef. 16 h  
 0110 = 16h reached  
 1110 = Error  
 1111 = not available

*Bit 5..4 :Driver 2 card*

00 = Card not present  
 01 = Card present

*Bit 7..6 :Not used*

### ***tachograph[3]***

*Bit 0..1 :System event*

00 = no tachogr. Event  
 01 = tachogr. Event

*Bit 2..3 :Handling information*

00 = no handling information  
 01 = handling information

*Bit 5..4 :Tachgraph performance*

00 = Normal performance  
 01 = Performance

*Bit 7..6 :Direction indicator*

00 = Forward  
 01 = Reverse

## ***Paket FMS\_EXT***

Tímto paketem jsou vyžadována data čtená ze sběrnice CAN.

Požadavek klienta má tvar:

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

Odpověď interface ETH2CAN má tvar:

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

```

unsigned char      id;
unsigned __int16   rpm;
unsigned __int16   speed;
unsigned char      acc_pedal;
unsigned char      brake_pedal;
unsigned __int32   total_fuel_used;
unsigned __int32   total_engine_hours;
unsigned char      fuel_level;
unsigned __int16   fuel_consumption;
unsigned char      axle_weight_captured[12];
unsigned char      axle_weight_location[12];
unsigned __int16   axle_weight[12];
unsigned __int32   total_vehicle_distance;
unsigned __int16   daily_vehicle_distance;
unsigned __int16   service_distance;
unsigned char      engine_coolant_temperature;
unsigned __int16   tachograph_speed;
unsigned char      tachograph[4];
unsigned char      tire_pressure_captured[12];
unsigned char      tire_pressure_location[12];
unsigned char      tire_pressure[12];
unsigned __int16   door;
unsigned __int16   fuel_instantaneous;
unsigned __int16   fuel_rate;
//TIMESTAMP
unsigned __int16   rpm_captured;
unsigned __int16   speed_captured;
unsigned __int16   acc_pedal_captured;
unsigned __int16   brake_pedal_captured;
unsigned __int16   total_fuel_used_captured;
unsigned __int16   total_engine_hours_captured;
unsigned __int16   fuel_level_captured;
unsigned __int16   fuel_consumption_captured;
unsigned __int16   total_vehicle_distance_captured;
unsigned __int16   daily_vehicle_distance_captured;
unsigned __int16   service_distance_captured;
unsigned __int16   engine_coolant_temperature_captured;
unsigned __int16   tachograph_speed_captured;
unsigned __int16   tachograph_captured;
unsigned __int16   fuel_instantaneous_captured;
unsigned __int16   fuel_rate_captured;

//ROZSIRENI - DETEKOVAT NA ZAKLADE VELIKOSTI PAKETU
unsigned char      secondary_fuel_level;
unsigned __int32   hires_total_fuel_used;
unsigned char      percent_torque;
unsigned char      service_brake_air_pressure[2];
unsigned char      diesel_exhaust_fluid_level;
unsigned char      tell_tale_status[8*4];
unsigned char      clutch_brake_cruise_control;
unsigned char      engine_load_at_speed;

```



```

unsigned __int16    gross_combination_vehicle_weight;
unsigned char      retarder_torque_mode;
unsigned char      actual_retarder_percent_torque;
unsigned char      retarder_selection_non_engine;
unsigned char      air_suspension_control[8];
unsigned char      selected_gear;
unsigned char      current_gear;
unsigned char      door2[8];
//TIMESTAMP
unsigned __int16    hires_total_fuel_used_captured;
unsigned __int16    service_brake_air_pressure_captured;
unsigned __int16    diesel_exhaust_fluid_level_captured;
unsigned __int16    tell_tale_status_captured;
unsigned __int16    gross_combination_vehicle_weight_captured;
unsigned __int16    retarder_captured;
unsigned __int16    air_suspension_control_captured;
unsigned __int16    gear_captured;
unsigned __int16    door2_captured;

} ETH_FMS;
* ZMĚNA ve verzi FW od 2.00, revize HW 1.30
* ZMĚNA ve verzi FW od 1.60, revize HW 1.20
* ZMĚNA ve verzi FW od 2.05, revize HW 1.30

* ZMĚNA ve verzi FW od 2.12, revize HW 1.30

```

Položky xxx\_captured udávají stáří veličiny od jejího přečtení z CAN bus sběrnice ve stovkách milisekund. Veličina která nebyla z CAN sběrnice přečtena má hodnotu 65535.

Položka secondary\_fuel\_level; má timestamp stejný jako fuel\_level.

### **Paket DTI**

Tímto paketem jsou vyžadována data čtená z digitálního tachografu.

Požadavek klienta má tvar:

```

typedef struct _ETH_DTI_REQ {
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16    packet_size;
} ETH_DTI_REQ;

```

Odpověď interface ETH2CAN má tvar:

```

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

```

```

unsigned char    seconds;
unsigned char    minutes;
unsigned char    hours;
unsigned char    month;
unsigned char    day;
unsigned char    year;
unsigned char    local_minute_offset;
unsigned char    local_hour_offset;
unsigned char    work_states;
unsigned char    driver_1_states;
unsigned char    driver_2_states;
unsigned char    tachograph_status;
unsigned __int16 tachograph_vehicle_speed;
unsigned __int32 total_vehicle_distance;
unsigned __int32 trip_distance;
unsigned __int16 k_factor;
unsigned __int16 engine_speed;
unsigned __int16 additional_information;
unsigned char    vehicle_id_len;
unsigned char    vehicle_id[20];
unsigned char    vehicle_reg_len;
unsigned char    vehicle_reg[20];
unsigned char    driver_1_len;
unsigned char    driver_1[20];
unsigned char    driver_2_len;
unsigned char    driver_2[20];
} ETH_DTI;

```

## **Paket J1708**

Paket přidán ve verzi FW od 2.00, revize HW 1.30.

```

typedef struct _ETH_J1708{
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16    packet_size;
    unsigned char      id;
    unsigned char      road_speed;
    unsigned char      fuel_level;
    unsigned char      engine_temperature;
    unsigned __int16    fuel_rate;
    unsigned __int16    fuel_economy;
    unsigned __int16    aver_fuel_economy;
    unsigned __int16    engine_speed;
    unsigned __int32    total_fuel;
    unsigned __int32    total_km;
    unsigned __int32    total_hours;
    unsigned __int32    manuf_total_fuel;
}ETH_J1708;

```

Pozor, data ze sběrnice J1708 jsou normou udávána v anglosaských jednotkách.

Data	Počet bitů	Váha 1 bitu	Offset
Rychlost	8	0.805 km/h	0
Stav palivové nádrže	8	0.5 %	0
Teplota motoru	8	1°C	0
Průtok paliva	16	16,428 x 10 <sup>-6</sup> l/s	0
Okamžitá spotřeba	16	1,66072 x10 <sup>-3</sup> km/l	0
Průměrná spotřeba	16	1,66072 x10 <sup>-3</sup> km/l	0
Otáčky motoru	16	0,25 rpm	0
Celkově spotřebované palivo	32	0,473 l (0,125 gal)	0
Celkové kilometry	32	0,05 h	0
Celkové motohodiny	32	0,161 km (0,1 mi)	0
Celkově spotřebované palivo-manufactured specific	32	0,01 l	0

## Paket J1708\_EXT

Paket přidán ve verzi FW od 2.00, revize HW 1.30.

```
typedef struct _ETH_J1708_EXT{
    unsigned __int32    magic;
    unsigned char      packet_type;
    unsigned __int16   packet_size;
    unsigned char      id;
    unsigned char      road_speed;
    unsigned char      fuel_level;
    unsigned char      engine_temperature;
    unsigned __int16   fuel_rate;
    unsigned __int16   fuel_economy;
    unsigned __int16   aver_fuel_economy;
    unsigned __int16   engine_speed;
    unsigned __int32   total_fuel;
    unsigned __int32   total_km;
    unsigned __int32   total_hours;
    unsigned __int32   manif_total_fuel;

    //TIMESTAMP
    unsigned __int16   road_speed_captured;
    unsigned __int16   fuel_level_captured;
    unsigned __int16   engine_temperature_captured;
    unsigned __int16   fuel_rate_captured;
    unsigned __int16   fuel_economy_captured;
    unsigned __int16   aver_fuel_economy_captured;
    unsigned __int16   engine_speed_captured;
    unsigned __int16   total_fuel_captured;
    unsigned __int16   total_km_captured;
    unsigned __int16   total_hours_captured;
    unsigned __int16   manif_total_fuel_captured;
} ETH_J1708_EXT;
```

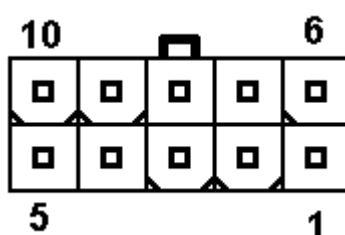
\* ZMĚNA ve verzi FW od 2.10, revize HW 1.30

## 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



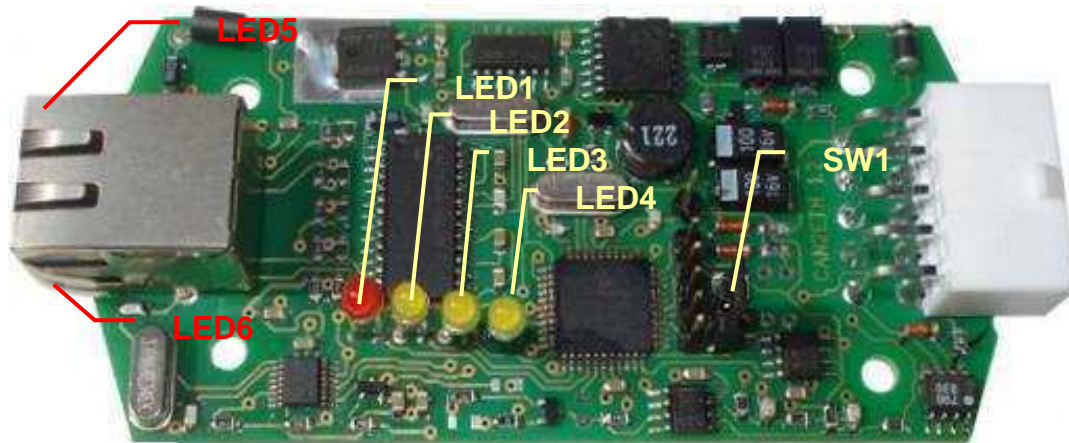
Konektor na PCB.

Pin	Popis
1	Napájecí napětí 8-36V
2	GND
3	CAN H
4	J1708 A
5	Tachograf A – signál
6	Signál 15 (startup-shutdown)
7	GND
8	CAN L
9	J1708 B
10	Tachograf B – GND

### Funkce indikačních LED

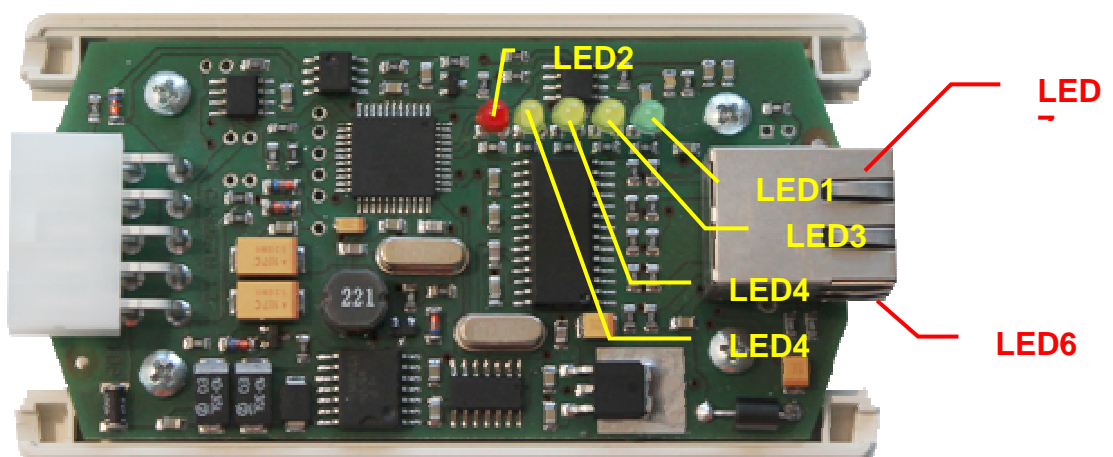
HW <= rev.1.2

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.
3	YELLOW	Indikuje činnost interface digitálního tachografu, LED mění stav.
4	YELLOW	Indikuje činnost sběrnice J1708, LED mění stav.
5	GREEN	Indikuje příchod paketu (TCPIP paket, ping apod.)
6	YELLOW	Indikuje připojení ethernetového kabelu.



HW &gt;= rev.1.3

LED#	Barva	Popis
1	GREEN	Power LED, aktivní je li připojen signál „15“.
2	RED	Indikuje činnost ETHERNETU, při příjmu paketu LED mění stav.
3	YELLOW	Indikuje činnost interface CANu, LED mění stav.
4	YELLOW	Indikuje činnost sběrnice J1708, LED mění stav.
5	YELLOW	Indikuje činnost interface digitálního tachografu, LED mění stav.
6	GREEN	Indikuje příchod paketu (TCP/IP 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 zakončovacím odporem 120 ohmu. Při připojení na motorový CAN není třeba zakončování odpor aktivovat, v případě připojení na FMS bránu je to obvykle nutné. Existenci správného počtu zakončovacích odporů je možné ověřit ve vypnutém vozidle pomocí ohmmetru. Mezi vodiči CAN H a CAN L je ve správném stavu odpor cca 60 ohmu. Propojka **SW2** je určena pro reset převodníku do výrobního nastavení.

## **Příkaz SETTINGS u nejčastěji monitorovaných vozidel.**

### **Nákladní vozidla – páteřní CAN bus**

- listen only
- rozšířený CAN ID
- rychlost 250k

### **Nákladní vozidla – FMS gateway (FMS brána)**

- normální mód
- rozšířený CAN ID
- rychlost 250k

### **Vozidla Škoda/VW, motorový CAN bus**

- listen only
- standardní CAN ID
- rychlost 500k

## Změna FW uživatelským programem

Interface podporuje možnost update firmware. Po připojení interface k napájení dojde vždy automaticky ke spuštění bootloderu. Bootloder je ukončen příkazem z nadřazeného systému (zasláním paketu **ETH\_RUN**), nebo po uplynutí doby nakonfigurované v parametru **app\_start\_timeout**.

Firmware je uložen v souborech s příponou HEX v textovém formátu jako řada hexadecimálních číslic.

Řádek tohoto souboru má následující tvar:

```
:1034B00029F070C30CF371C30DF3000E0DBFFF0EA6
```

Jednotlivé řádky jsou do zařízení odesílány celé s výjimkou počáteční dvojtečky. K zaslání je použit paket **PACKET\_REBOOT\_DATA**. Data řádku HEX souboru bez počáteční dvojtečky jsou uložena v sekci data.

Po provedení zápisu zasláního řádku vygeneruje interface zpět odpověď pomocí paketu **ETH\_REBOOT\_ACK**. Je-li zápis proveden správně, obsahuje položka **error\_code** hodnotu 0. Protože HEX soubor obsahuje i některá automaticky vygenerovaná data na adresách mimo povolený rozsah, vrací v některých případech interface v položce **error\_code** hodnotu 1 (chybná adresa). Tento chybový kód ignorujte a pokračujte v bootování následujícím řádkem stejně jako v případě návratové hodnoty 0. Je-li vrácen návratový kód 2 (chybná délka dat), je chyba ve formátu HEX souboru. U návratových kódů 3 (chyba zápisu do flash) a 4 (chyba verifikace zápisu) je možné se pokusit o opakovaný zápis zasláního řádku HEX souboru.

Další řádek HEX souboru není možné zaslat před obdržáním odpovědi **ETH\_REBOOT\_ACK** se stavem zápisu předchozího řádku.

Paket **ETH\_REBOOT\_ACK** obsahuje pole dummy ve kterém je uložen obraz paměťového segmentu, do kterého byl zápis prováděn. Tato položka je určena pouze pro verifikaci chování firmware bootloderu. Během bootování ji ignorujte.

Jakmile jsou zaslány všechny řádky HEX souboru, pokračujte v činnosti spuštěním aplikace paketem **ETH\_RUN**.

## Testovací aplikace

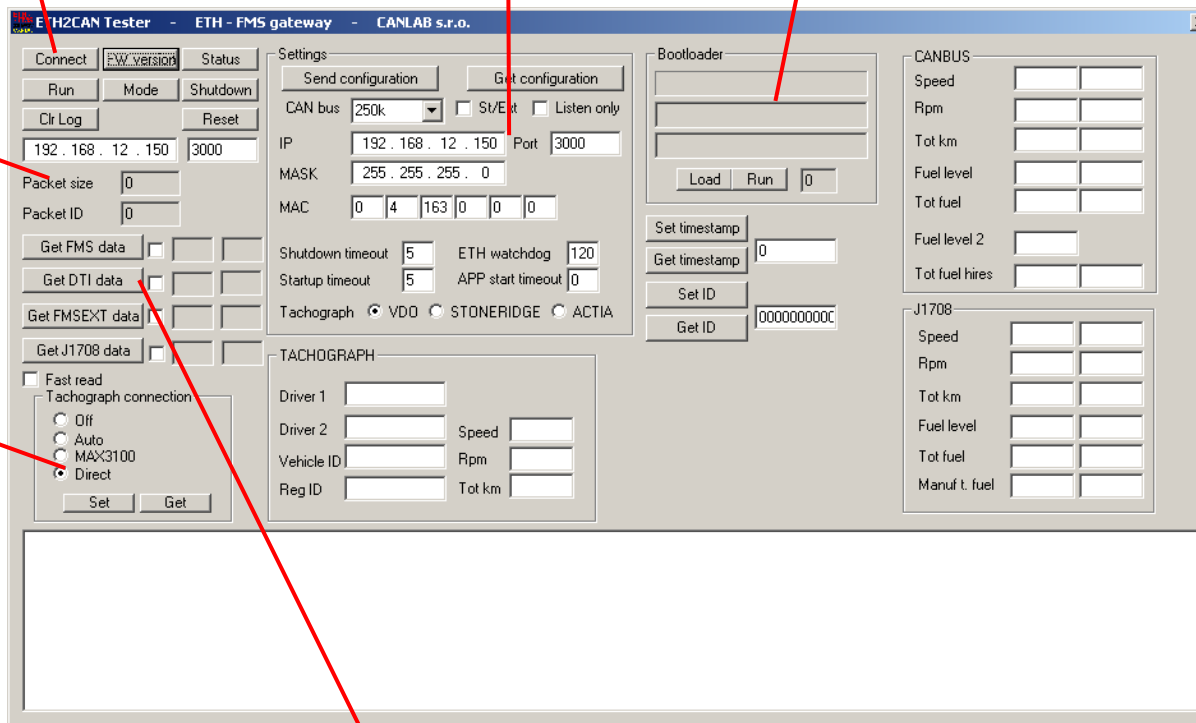
Tlačítko pro navázání spojení se zařízením

Nastavení zařízení. Doporučený postup je použít Get, provést změny a nastavit pomocí Send. Pro nákladní vozidla nastavení 250k, zatrhnout St/Ext a při přímém připojení na CAN aktivovat Listen only, u FMS brány Listen only vypnout.

Změna FW. Nelze kombinovat FW pro HW verzi 1.2 a 1.3 !!! Režim Bootloaderu je nastaven několik sekund po připojení jednotky k napájení. Čas je nastaven dle parametru APP start timeout v Settings. Manuálně lze aplikaci resetovat do Bootloaderu příkazem Shutdown (vlevo). Tlačítko Reset restartuje pouze aplikaci. Tlačítko Mode zjistí, zda je zařízení v režimu bootloader nebo aplikace.

IP adresa a port zařízení. Standardně 192.168.12.150, port 3000. Zkratovací propojkou v zařízení lze tuto adresu aktivovat.

Tento režim je určen pouze pro HW 1.2 a dovoluje nastavit typ připojení tachografu v závislosti na variantě osazení HW. Nepoužívat bez znalosti varianty osazení!!!



Tlačítka dovolují poslat požadavek na jednorázové čtení dat. Zatržítka aktivují periodické čtení dat.



## Změny ve verzích firmware

### 1.13 boot

- doplněna podpora paketu PACKET\_UNKNOWN\_PACKET\_ID

### 1.13

- doplněna podpora paketu PACKET\_UNKNOWN\_PACKET\_ID
- doplněna podpora paketu PACKET\_RESET
- doplněna podpora paketu PACKET\_SET\_TIMESTAMP
- doplněna podpora paketu PACKET\_GET\_TIMESTAMP
- doplněna podpora paketu PACKET\_FMS\_EXT
- paket PACKET\_CONFIGURATION již neprovádí automaticky restart firmware. Nutný je restart paketem SHUTDOWN (do bootloaderu) nebo RESET (restart aplikace firmware).

### 1.20 boot

- přechod na Microchip TCPIP Stack verze 4.55

### 1.20

- přechod na Microchip TCPIP Stack verze 4.55

### 1.21

- doplněna možnost čtení sériového čísla
- konfigurační paket doplněn o možnost nastavení masky sítě
- konfigurační paket doplněn o možnost volby připojeného tachografu (vlastní čtení z tachografu Stoneridge a Acta není ještě implementováno, funkční pouze VDO).

### 1.23 boot

- přidána podpora čtení konfigurace zařízení
- znovu aktivován funkce automatického spuštění aplikace po uplynutí zadaného intervalu
- zrychlena rychlost načítání firmware-aplikace

### 1.23

- sériové číslo rozšířeno na 10 míst

### 1.30 boot

- podpora PIC 18F4680 Rev7.

**1.30**

- podpora PIC 18F4680 Rev7.

**1.45**

- Podpora připojení tachografu přímo na PIC bez použití extenderu MAX3100.

**1.60**

- Podpora secondary fuel level a hires total fuel used ve FW pro HW revizi 1.2.

**2.00**

- Redesign HW na verzi 1.3. Podpora připojení J1708, Změna struktury paketu FMS.
- FW nelze použít pro HW revize 1.2 a starší.

**2.12**

- Rozšíření množství čtených dat u paketu FMS a FMS EXT