ETH2CAN – FMS firmware

Content:

Ing. David Španěl

Mgr. Vítězslav Rejda

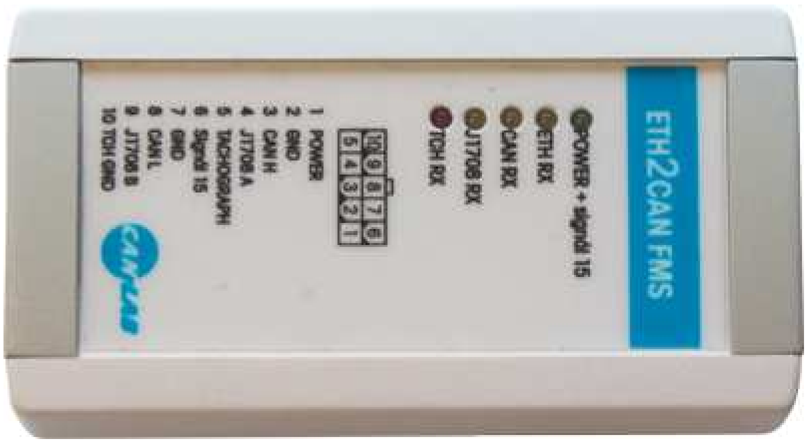
Basic description

FMS firmware ETH2CAN interface is designed for providing car operation data from trucks and cars. For connection into the car it is equipped with these interfaces:

- CAN bus (high speed)
- Digital tachograph interface DTCO1381
- Interface for J1708 bus (older cars)

ETHERNET interface (speed of 10 Mbit) is designed for connection to superior system.

The device does not provide all data stated in data packets, but only available data. Availability depends on type, manufacturer, modification, and car manufacture year.



Communication via ETHERNET interface

Device has its IP address and TCP port for entire communication. It behaves like server, it means that client connects to this device.

Several packets are used for communication. Each packet contains magic item that is meant for verifying of device endian. Its value is 0xAA123456. It also contains packet_type item that distinguishes packet type. The packet_size item determines size of packet. Within development stage, packet_size is set to 0xFFFF value. It means that the size is not signalized and client's application shall determine packet size using size of structure. The id item is designed for packet identification. For example, if query is sent to the device, it is possible to set id item to any value. Then, set value of response packet is the same. It can be used for distinguishing of two responses when two different queries are sent.

Packet types:

```
#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_SPEC_SOR                    37
#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
```

```
* CHANGE in FW version from 2.00, revision of HW 1.30
* CHANGE in FW version from 3.00, revision of HW 1.30
```

Packet support:

Packet	**Bootloader**	**Application**
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 (only for reading)	Y
PACKET_FIRMWARE_VERSION	Y	Y
PACKET_CONFIGURATION	Y (only for reading)	Y
PACKET_GET_TACHOGRAPH_CONNECTION	N	Y
PACKET_RESP_TACHOGRAPH_CONNECTION	N	Y
PACKET_SET_TACHOGRAPH_CONNECTION	N	Y

UNKNOWN_PACKET_ID packet

Packet in the direction ETH2CAN -> Client (superior system).

```
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;
```

If interface receive packet with unknown packet_type value, this packet is returned. For example, in bootloader regime it is returned after sending FMS or DTI packet, in the application regime after sending of REBOOT_DATA.

RUN packet

Packet in the direction Client (superior system) -> ETH2CAN.

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

Packet is intended for activation device activation. The device is in bootloader regime after connection of device to power and connection of signal 15. This regime is intended for easy change of firmware in the device. Firmware activation takes place after sending of this packet. After 30 seconds, bootloader is automatically switched into firmware regime, if bootloader doesn't detect reception of PACKET_REBOOT_DATA packet.

Packet in the direction ETH2CAN -> Client (superior system).

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

Packet is generated as response to incoming ETH_RUN packet. It confirms reception of this packet and in mode item, it signalizes actual firmware regime (bootloader: mode = 1, application-firmware: mode = 2)

MODE packet

Packet in the direction Client (superior system) -> ETH2CAN.

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

Packet allows to request actual firmware regime.

Packet in the direction ETH2CAN -> Client (superior system).

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

Packet is generated as response to incoming ETH_MODE packet. It confirms reception of this packet and in mode item, it signalizes actual firmware regime (bootloader: mode = 1, application: mode = 2).

SET_TIMESTAMP packet

Packet in the direction Client (superior system) -> ETH2CAN.

```
typedef struct _SET_TIMESTAMP {
     unsigned __int32    magic;
     unsigned char       packet_type;
     unsigned __int16    packet_size;
     unsigned char       id;
     unsigned __int16    timestamp;
}    SET_TIMESTAMP;
```

Packet is designed for reconfiguration of timestamp counter. Timestamp is an item set for most of measured quantities from CAN bus and it represents age of the quantity. The value is incremented each 100 ms. After start, the timestamp value is set to 0.

Packet in the direction ETH2CAN -> Client (superior system).

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

GET_TIMESTAMP packet

Packet in the direction Client (superior system) -> ETH2CAN.

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

Packet is designed for reading of actual value of timestamp counter. Timestamp is an item set for most of measured quantities from CAN bus and it represents age of the quantity. The value is incremented each 100 ms.

Packet in the direction ETH2CAN -> Client (superior system).

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

Packet is generated as response to incoming SET_TIMESTAMP packet. It confirms reception of this packet and it returns timestamp counter value back.

PACKET_REBOOT_DATA packet

Packet in the direction Client (superior system) -> ETH2CAN.
In bootloader regime, the packet transmits 1 line of HEX file.

```
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;
```

Packet in the direction ETH2CAN -> Client (superior system).
Packet confirms reception and programming of sent line of HEX file and signalizes that it is ready for reception of following line.

```
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;
```

After sending of whole file, new FW is updated by RUN packet.

Item error_code:
0 – OK
1 – incorrect address
2 – data length incorrect
3 – flash record incorrect
4 – record verification incorrect

SHUTDOWN , RESET packet

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

Packet in the direction ETH2CAN -> Client (superior system).

Item why:
0 – on demand of client
1 – ethernet watchdog
2 – signal 15 switched off

By this packet, the device indicates termination of activity after signal 15 disconnection or restart of device.

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

Packet in the direction Client (superior system) -> ETH2CAN.
Control processor firmware restart occurs after reception of this packet. Application (not bootloader) supports RESET command that resets only the application.

TACHOGRAPH_CONNECTION packet

```
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;
```

Packet in the direction Client (superior system) -> ETH2CAN.

```
tachograph_connection =  0    - OFF
                         1    - AUTO (FW determines connection)
                         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;
```

Packet in the direction ETH2CAN -> Client (superior system).

```
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 - actually used settings
tachograph_connection_EEPROM - settings in EEPROM (set by
ETH_TCH_CONNECTION_SET) that will be used after restart.
```

CONFIGURATION packet

Packet in the direction Client (superior system) ->ETH2CAN.
It sets new configuration of device. Device responds by sending this packet back.

```
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;
```

Packet in the direction ETH2CAN -> Client (superior system).
Designed for determination of device actual configuration:

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

can_speed – CAN bus speed, values

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 normal mode (connection to FMS gate)
1 listen only mode (connection to car CAN bus, engine CAN)

st_ext
0 standard identifiers
1 extended identifiers

ip
IP address of the device. Default setting 192.168.12.150. However, it is possible to require different value from manufacturer.

port
TCP port, where the communication is running. Default 3000.

startup_timeout
Delay of device activation after connection of signal 15. The delay eliminates activation of the device within short activation of signal 15. Time is set in seconds. Range 1…200s. Default 5 s.

shutdown_timeout
Delay of device deactivation after disconnection of signal 15. The delay eliminates deactivation of device within short deactivation of signal 15. Time set in seconds. Range 1...200s. Default 5 s.

eth_watchdog
Timeout in seconds. If an activity of a client is not detected in set period of time, reset of device occurs. Function is not active, when 0 value is set. Range 20...300s.

mac
MAC address of device. Default 00-04-A3-00-00-00.

app_start_timeout
Time, after which the bootloader is automatically switched into application, if there is no packet received that changes firmware.

When this packet is received, new setting is saved into internal EEPROM interface. It is necessary to restart firmware by shutdown order or reset to apply new setting.

mask
Mask of net. Default setting 255.255.255.0.

tachograph_mode

Connected digital tachograph type setting, 0-VDO Siemens, 1-Stoneridge, 2-Actia.

FIRMWARE VERSION packet

Firmware version in interface ETH2CAN device is requested by using this packet.

Client's request form:

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

Interface ETH2CAN response form:

```
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;
```

This item contains string with firmware version. It does not contain ending 0 strings. Form of the string is e.g. CANLABsro-01.10. In bootloader regime e.g. CANLABsro-01.10boot. Bootloader uses different numeration than applications!

SERIAL NUMBER packet

This packet is designed for reading of interface ETH2CAN serial number.

Client's request form:

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

Interface ETH2CAN response form:

```
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;
```

The item contains string with interface serial number. The form of serial number is E2Cxxxxxxxxxx. First three signs are always E2C. Following 10 signs are numbers, thus value of the serial number can be 0000000000-9999999999. The last (fourteenth) sign is 0, thus the end of the string.

FMS packet

Data read out of CAN bus are requested by this packet.

Client's request form:

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

Interface ETH2CAN response form:

```
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;

    //EXTENSION - DETECT BASED ON PACKET SIZE
    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;

    * CHANGE in FW version from 2.00, HW revision 1.30
    * CHANGE in FW version from 1.60, HW revision 1.20
    * CHANGE in FW version from 2.12, HW revision 1.30
```

Structure data can be converted into real values using this chart:

Data	Number of bits	Weight of 1 bit	Offset
Speed	16	1/256 km/h	0
Position of acceleration pedal	8	0.4 %	0
Position of brake pedal	8	0.4 %	0
Total amount of consumed fuel	32	0.5 litre	0
Total amount of consumed fuel - HIRES	32	0.001 litre	0
State of fuel tank	8	Truck:0.4 VW:1litre	0
Engine revolutions	16	0.125 rev	0
Axle load	16	0.5 kg	0
Total amount of operating hours	32	0,05 h	0
Total amount of covered kilometers	32	5 m	0
Distance to service (in kilometers)	16	5 km	-160 635
Temperature of coolant	8	1°C	-40
Average consumption	16	1/512 km/L	0

Axle weight

The axle_weight_location[x] means location of axle weight value in axle_weight[x] item. The value axle_weight_captured[x] = 0 means that this item does not contain any (valid) value, axle_weight_captured[x] = 1 means that the item contains valid value.

Information regarding number of measured axle and wheel of this axle is encoded in the axle_weight_location[x] item. Lower 4 bits mean wheel index, upper 4 bits mean axle index. If all 4 bits are set to 1, location is unknown.

Item tachograph[4]

This item contains information that can be decoded according to following description:

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 :Tachograph performance
00 = Normal performance
01 = Performance

Bit 7..6 :Direction indicator
00 = Forward
01 = Reverse

FMS_EXT packet

Data red from CAN bus are requested by this packet.

Client's request form:

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

Interface ETH2CAN response form:

```
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;

          //EXTENSION - DETECT BASED ON PACKET SIZE
     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;
* CHANGE in FW version from 2.00, HW 1.30 revision
* CHANGE in FW version from 1.60, HW 1.20 revision
* CHANGE in FW version from 2.05, HW 1.30 revision

* CHANGE in FW version from 2.12, HW 1.30 revision
```

Item xxx_captured means age of the quantity from the time it was read from CAN bus in hundreds of milliseconds. Value of quantity that was not read out of the CAN bus is 65535.

Timestamp of secondary_fuel_level item is the same as fuel_level timestamp.

DTI packet

Data read from digital tachograph are requested by this packet.

Client's request form:

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

Interface ETH2CAN request form:

```
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;
```

J1708 packet

Packet added in FW version from 2.00, HW 1.30 revision.

```
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;
```

Attention! Data from J1708 bus are stated in Anglo-Saxon units (based on standards).

Data	Number of bits	Weight of 1 bit	Offset
Speed	8	0.805 km/h	0
State of fuel tank	8	0.5 %	0
Engine temperature	8	1°C	0
Fuel flow	16	16.428 x 10-6 l/s	0
Actual consumption	16	1.66072 x10-3 km/l	0
Average consumption	16	1.66072 x10-3 km/l	0
Engine revolutions	16	0.25 rpm	0
Total amount of consumed fuel	32	0.473 l (0.125 gal)	0
Total amount of kilometers	32	0.05 h	0
Total operating hours	32	0.161 km (0.1 mi)	0
Total amount of consumed fuel - manufactured specific	32	0.01 l	0

J1708_EXT packet

```
Packet added in FW version from 2.00, HW 1.30 revision.
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	manuf_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   manuf_total_fuel_captured;

} ETH_J1708_EXT;

* CHANGE in FW version from 2.10, HW 1.30 revision
```

SPEC_SOR packet

Added in FW 3.00 version.

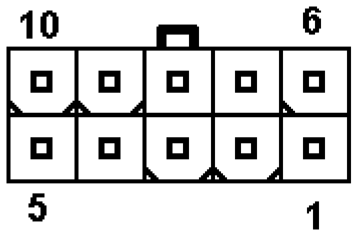
```
typedef struct  {
     unsigned __int32   magic;
     unsigned char      packet_type;
     unsigned __int16   packet_size;
     unsigned char      id;

     unsigned char      interior_temperature;
     unsigned char      outdoor_air_temperature;
     unsigned char      air_conditioning;
     unsigned char      heating;

     //TIMESTAMP
     unsigned __int16   interior_temperature_captured;
     unsigned __int16   outdoor_air_temperature_captured;
     unsigned __int16   air_conditioning_captured;
     unsigned __int16   heating_captured;
} ETH_FMS_SPEC_SOR;

interior_temperature     - resolution 0.5 °C, offset -40
degrees
outdoor_air_temperature - resolution 0.5 °C, offset -40
degrees
air_conditioning - bit information, b0=1, air conditioning
activation
heating_captured - bit information, b0=1, heating in the space
form passengers activation
heating_captured - bit information, b1=1, activation of
outlets of independent heater
```

Connection of the device

The device is placed in TOPTEC 102 box by OKW. The device has two connectors. The first is RJ45, thus classic ethernet connector. The second one is MOLEX that is set for connection of power and can bus.

The device works with power range 8-36 V. The consumption of the device in operation is 1.7 W. When deactivated after disconnection of signal 15, the consumption is almost zero. Signal 15 is activate approx. from 1V.

MOLEX connector



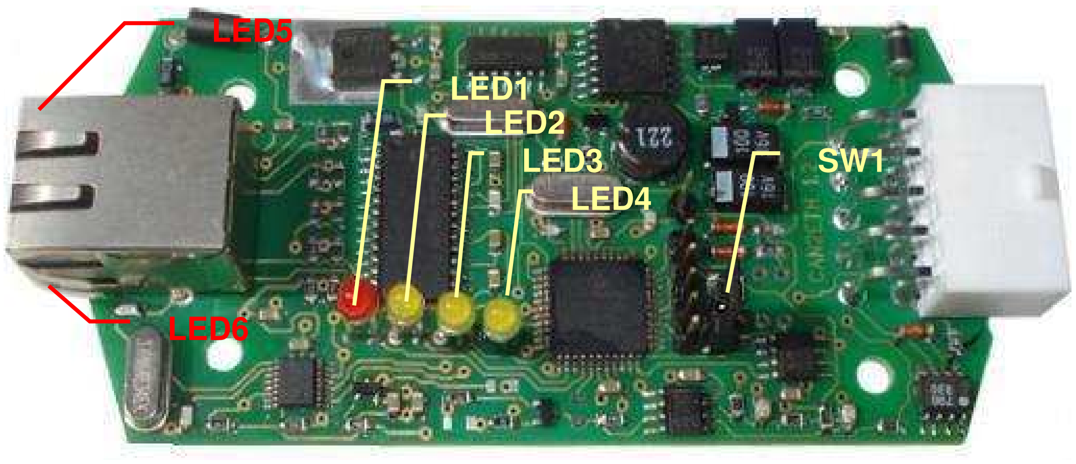
PCB connector

Pin	Description
1	Power 8-36V
2	GND
3	CAN H
4	J1708 A
5	Tachograph A – signal
6	Signal 15 (startup-shutdown)
7	GND
8	CAN L
9	J1708 B
10	Tachograph B – GND

Indicative LED function

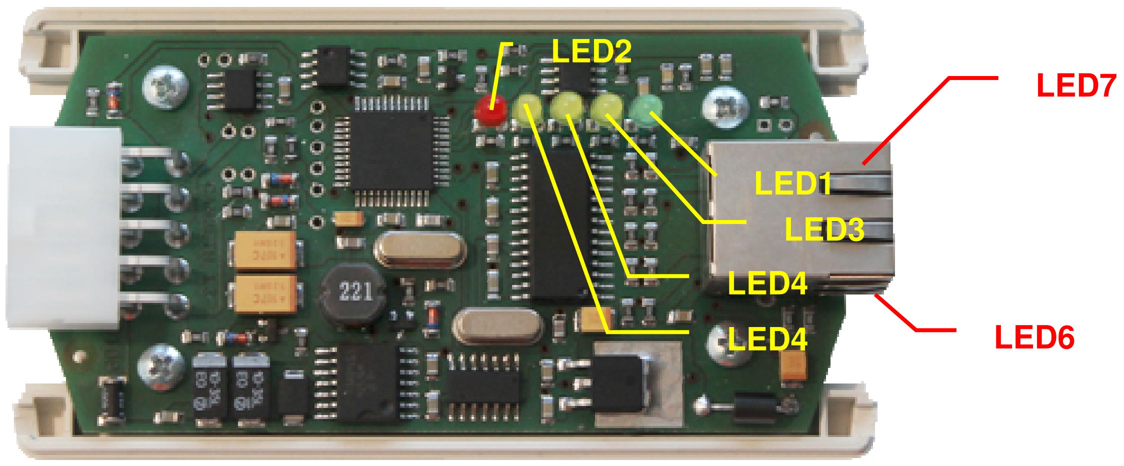
HW <= rev.1.2

LED#	Color	Description
1	**RED**	Fault of CAN bus – bus off (e.g. bus speed is incorrectly set, it is not functional in Listen only regime). In bootloader regime, this LED fleshes in the interval of 1 s.
2	**YELLOW**	Indicates activity of CAN bus, LED changes state.
3	**YELLOW**	Indicates activity of digital tachograph interface, LED changes state.
4	**YELLOW**	Indicates activity of J1708 bus, LED changes state.
5	**GREEN**	Indicates incoming packet (TCPIP packet, ping and so on)
6	**YELLOW**	Indicates connection of ethernet cable.



HW >= rev.1.3

LED#	Color	Description
1	GREEN	Power LED, active when signal 15 is connected.
2	RED	Indicates activity of ETHERNET, LED changes state during acceptance of packet.
3	YELLOW	Indicates activity of CAN interface, LED changes state.
4	YELLOW	Indicates activity of J1708 bus, LED changes state.
5	YELLOW	Indicates activity of digital tachograph interface, LED changes state.
6	GREEN	Indicates incoming packet (TCPIP packet, ping and so on)
7	YELLOW	Indicates connection of ethernet cable.



Short circuit jumper **SW1** is designed for activation of 120 ohm terminator on CAN bus. CAN bus is always ended on both sides with 120 ohm terminators. It is not necessary to activate terminator after connection to engine CAN; in case of connection to FMS gate it is usually necessary. It is possible to verify existence of correct number o terminators in switched off car by ohmmeter. Ideally, the correct resistance between CAN H and CAN L conductors is approximately 60 ohm.
SW2 jumper is designed for reset of converter into default settings.

SETTINGS order for the most frequently monitored cars.

Trucks – backbone CAN bus

- listen only
- extended CAN ID
- speed 250k

Trucks – FMS gateway

- normal mode
- extended CAN ID
- speed250k

Škoda/VW, engine CAN bus

- listen only
- standard CAN ID
- speed 500k

Change of FW by user's program

Interface supports possibility of firmware update. After connection of interface to power, bootloader is always automatically activated. Bootloader is ended by command of superior system (by sending of **ETH_RUN** packet) or after lapse of time configured in **`app_start_timeout`** parameter.

Firmware is saved in files with HEX suffix in text format as string of hexadecimal numbers.

Line of this file has following form:

```
:1034B00029F070C30CF371C30DF3000E0DBFFF0EA6
```

Individual lines are sent to the device as a whole without initial colon. **PACKET_REBOOT_DATA** packet is used for sending. Line data of HEX file without initial colon are saved in data section.

After inscription of sent line interface generates response using **ETH_REBOOT_ACK** packet. If the inscription is correct, **error_code** value is 0. HEX file contains also some automatically generated data on addresses out of allowable range, therefore interface sometimes returns value 1 in error_code (incorrect address). Ignore this error code and continue in booting of following line as in case of return value 0. If return code 2 is returned (incorrect length of data), the error occurs in HEX file format. In case of return codes 3 (incorrect record into flash) and 4 (incorrect record verification) it is possible to try repeated record of sent HEX file line.

Following line of HEX file cannot be sent before reception of **ETH_REBOOT_ACK** response with state of record of previous line.

ETH_REBOOT_ACK packet contains "dummy" field, where image of memory segment with record is saved. This item is designed only for verification of bootloader firmware operation. Ignore it during booting.

When all lines of HEX file are sent, continue by initiation of application by **ETH_RUN** packet.

Testing applications

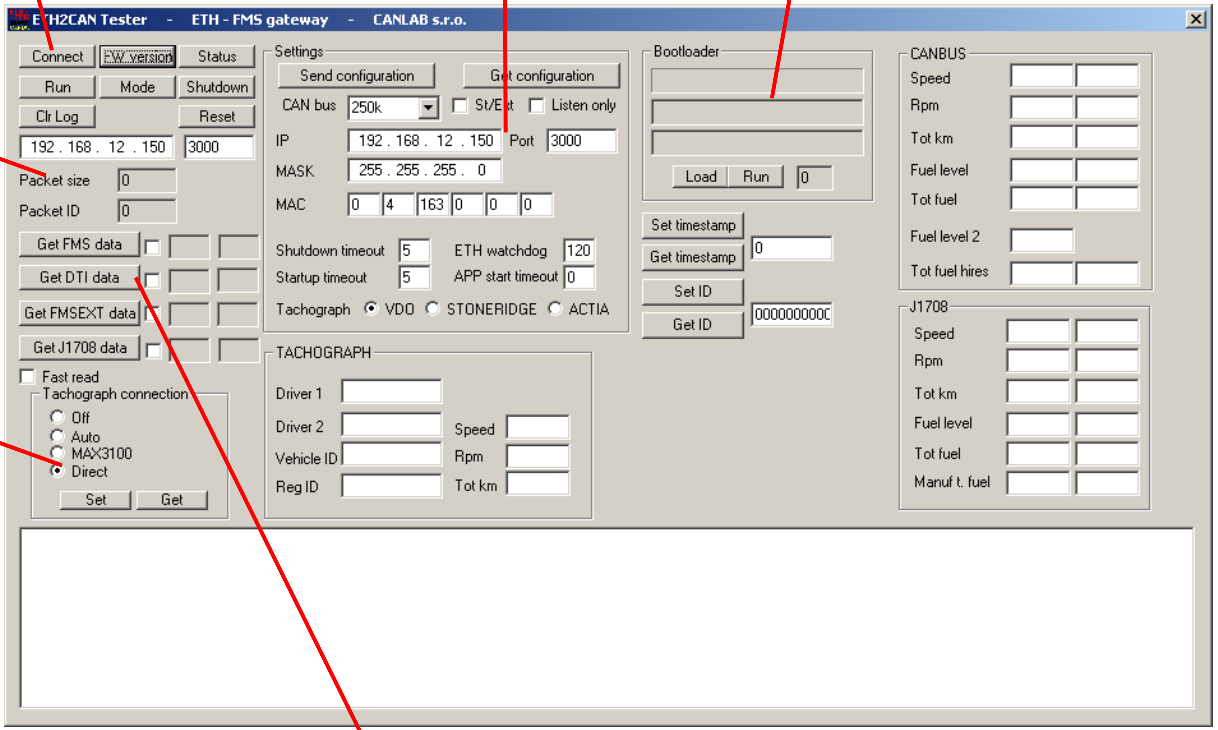
Button for connection with the device

Device settings. Recommended procedure: use Get, make changes and set them using Send.
For trucks, set 250k, check St/Ext and activate Listen only regime when directly connected to CAN, deactivate Listen only regime in case of FMS gate.

Change of FW. It is not possible to combine FW for HW 1.2 and 1.3 !!! Bootloader regime is set to several seconds after connection of the unit to power. Time is set according to APP start timeout parameter in Settings. It is possible to reset the application manually to Bootloader by Shutdown command (on the left). Reset button resets only application. Mode button determines if the device is in bootloader regime or application.

IP address and port of the device. Standard: 192.168.12.150, port 3000. This address can be activated by short-circuit jumper in the device.

This regime is designed only for HW 1.2 and it allows setting of connection type of tachograph depending on HW shoulder type. Do not use without knowledge of shoulder type!!!



192 . 168 . 12 . 150 | 3000
Packet size: 0
Packet ID: 0
Tachograph connection: Off / Auto / MAX3100 / [✓ Direct]

Settings
CAN bus: 250k
St/Ext: ☐
Listen only: ☐
IP: 192 . 168 . 12 . 150
Port: 3000
MASK: 255 . 255 . 255 . 0
MAC: 0 | 4 | 163 | 0 | 0 | 0
Shutdown timeout: 5
ETH watchdog: 120
Startup timeout: 5
APP start timeout: 0
Tachograph: [✓ VDO] STONERIDGE ACTIA

Bootloader: 0
Timestamp: 0
ID: 000000000C

Buttons allow sending request for one-shot reading of data. When checked, periodic data reading is activated.

Changes in firmware versions

1.13 boot

- PACKET_UNKNOWN_PACEKT_ID packet support added

1.13

- PACKET_UNKNOWN_PACEKT_ID packet support added
- PACKET_RESET packet support added
- PACKET_SET_TIMESTAMP packet support added
- PACKET_GET_TIMESTAMP packet support added
- PACKET_FMS_EXT packet support added
- PACKET_CONFIGURATION packet does not automatically restart firmware. It is necessary to use SHUTDOWN packet (into bootloader) or RESET (restart of firmware application).

1.20 boot

- Transition to Microchip TCPIP Stack version 4.55

1.20

- Transition to Microchip TCPIP Stack version 4.55

1.21

- Serial number reading option added
- Setting of net mask option added into configuration packet
- Selection of connected tachograph option added into configuration packet (reading from tachograph Stoneridge and Acta not implemented yet, only VDO is functioning).

1.23 boot

- Reading of device configuration support added
- Reactivated function of automatic launch of application after lapse of set interval
- Accelerated speed of firmware application loading

1.23

- Serial number extended into 10 numbers

1.30 boot

- PIC 18F4680 Rev7 support

1.30

- PIC 18F4680 Rev7 support

1.45

- Support of tachograph connection directly to PIC without using of MAX3100 extender.

1.60

- Support of secondary fuel level and hires total fuel used in FW for HW 1.2 revision.

2.00

- HW redesign into 1.3 version. Support of J1708 connection. Change of FMS packet structure.
- FW cannot be used for HW 1.2 revision and older.

2.12

- Amount of read data at FMS and FMS EXT packet extended

3.00

- SPEC_ SOR packet added