Basic requirements

1. Hardware:
   a.) Digital EPS device with a CAN interface card.
   b.) Required CAN settings on the device (refer to the device user guide for details):
      • Device node, for example 8 (this will determine the two CAN IDs)
      • RID, for example 10 (this will determine the two CAN IDs)
      • CAN baudrate, for example 100k
      • Bus termination (very important if the device is the last one on the bus or the only one)
   c.) CAN controller card (internal or external) on the PC side, set to the same settings as b.), at least regarding bus termination and baudrate.
2. Software:
   a.) Drivers of CAN controller installed

Preparations

1. Determine CAN IDs

The EPS devices with CAN card of type IF-C1 or IF-C2 use two CAN IDs each. The IDs are determined by the device node and the RID settings in the device. The **first ID** is used to send data to the device, like a control byte (output on/off) or a set value. The **second ID** is used to query data from the device, such as actual values, set values, condition etc.

The IDs calculate as this: RID*64 + device node * 2 and RID*64 + device node * 2 + 1.

For the given examples in „1. Hardware“ above, the two IDs will calculate as 656 (hex: 0x290) and 657 (hex: 0x291).

2. Setting up the CAN controller

The CAN controller on the PC side is either set up by the user application or, for general use, by a config tool. It is imperative to set the same baudrate as with the device and to determine, if bus termination is required.

Accessing the device

The examples given here were made using an external CAN dongle from PeakSystems. In order to send a telegram via CAN to a specific device, it requires to give:

• **CAN ID** of the specific device (single cast mode) (later on referred as ID)

• **Data length** (later on referred as DL)

• **Data** (see further explanations below)

The **data length** will automatically result from the data we want to sent. The data is the important part. Since CAN has it‘s own checksum system, we don‘t need to include a checksum.

Data depends on whether you send/receive something with a single message or send/receive something with a split message. Split messages are used to send/receive data with a length of more than 8 bytes, because CAN will only transfer up to 8 bytes per message. See the examples below.

Example 1: Querying the device type string

According to the object lists that can be found in the IF-C1/IF-C2 user guide (one for every device series), the device type string is object number 0. Since we are about to query data from the device, we need to use the second CAN ID (0x291).

Querying of data only requires to send the object number. So the resulting telegram would look like this:

ID: 0x291, DL: 0x01 (because we just send 1 byte) , Data: 0x00

This is normally sufficient for the CAN driver. If the message is transferred correctly and the device is able to respond, the responses could look like this:

0x291 0x08 0x01 0xFF ..... (first message) and **0x291 0x07 0x01 0xFE ....** (second message).

0x291 is the responding CAN ID, the 0x08 the returned data length, the 0x01 the returned object number, the 0xFF is a marker that indicates that this is a split message (because length of returned bytes is >8) and the rest are the device type string bytes. The markers are given in descending order, means in the first message it‘s 0xFF, in the second one 0xFE etc. In those split messages, they‘re located always after the object number and before the first actual data byte that is returned by or sent to the device.

Example 2: Set the device into remote mode

According to the object lists that can be found in the IF-C1/IF-C2 user guide (one for every device series), the device can be set into remote control by object 54 (0x36). Since we are about to send control data to the device, we need to use the first CAN ID (0x290). Object 54 has a mask that varies depending on the bit that is going to be changed. In this case, bit 4 is about to be set. So the mask has to be 0x10 and the control byte also has to be 0x10.

By to the given scheme of data to be sent, the telegram is now:

ID: 0x290, DL: 0x03, Data: 0x36 0x10 0x10

where the object is again given as first byte in the data and then the rest.

If transferred correctly and accepted by the device for the current device condition, the device should change into remote mode. This is normally indicated on the front by different means (display or LED). Since this was a send telegram, the device does not respond anything.

Example 3: Setting the user text

The user can write and store a text of up to 15 ASCII characters length into the device. In order to show the usage of split messages, the example string shall be „System-ID: 774“, without the quotation marks. It has a length 1 characters.

Setting something like this user text requires the device to be in remote control mode, so the command from example 2 above should already been successfully sent.

According to the object lists that can be found in the IF-C1/IF-C2 user guide (one for every device series), the user text is set or read by object 7 (0x07). Since we are about to send control data to the device, we need to use the first CAN ID (0x290). And since the data length of the user text string is 15 (including EOL), we need to use three split messages:

1st message: 0x290 0x08 0x07 0xFF 0x53 0x79 0x73 0x74 0x65 0x6D

2nd message: 0x290 0x08 0x07 0xFE 0x2D 0x49 0x44 0x3A 0x20 0x37

3rd message: 0x290 0x05 0x07 0xFD 0x37 0x34 0x00

If transferred correctly and accepted by the device for the current device condition, the device store the string permanently, until changed. Since this was a send telegram, the device does not respond anything. The user text can be read back again with the message 0x291 0x01 0x07.