

Simple programming of the CAN bus application in LabView with the driver interface for the PEAKCAN Driver

The driver interface in [LabView](#) allows writing a program for CAN bus communication and gives the professional user the possibility to manage CAN messages quickly. Special attention was given to speed, easy usage, and a high quality online documentation.

Therefore configuration values, for example, for connecting new hardware are given in terms of enum variables. After placing a VI in the diagram the configurable values can be generated easily in LabView with the functions *Create Constant* or *Create Control*. The corresponding *Control* is displayed in the diagram or in the front panel. All additional controls are located in the control menus as shown in figure 1.

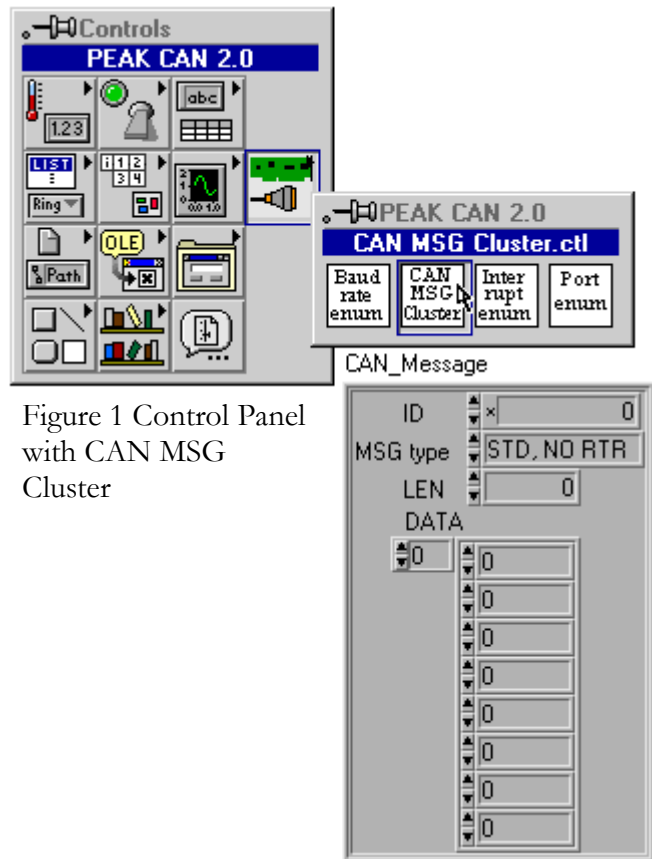


Figure 1 Control Panel with CAN MSG Cluster

The [PEAKCAN](#) driver can be addressed either via the high-level routine, shown in figure 2 or over the API calls underneath. The high-level routines enable a secure introduction to CAN bus programming. The routines provide, among other things, error clusters for signaling error states. The errors are shown in clear script and enable quick error analysis. The driver offers the professional user the possibility to perform independent error processing with VIs, which display API-calls exclusively.

The VIs are arranged in the [LabView](#) function palette just as they will be implemented later. Thus a program can erase all pre-existing driver settings (1*) using the VI **CAN Close all** (VI with the little door), in order to register new hardware on the driver via the VI **Register Hardware**.

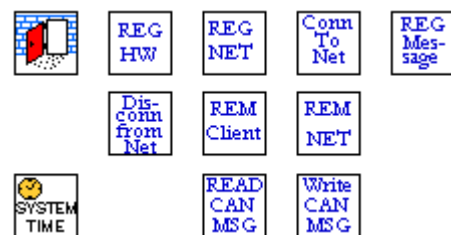


Figure 2 High-Level Routines

1* This function should not be used in connection with USB and PCI cards since otherwise the entries of the operating system will be erased. In that case these settings are lost until the PC is restarted. The examples contain a VI for locating USB and PCI cards for just this reason.

Afterwards it is necessary to build a virtual network with the VI **Register Net**. After this has been done, the program registers itself in the net over the VI **Connect to net**.

In addition the driver must be informed which messages are to be directed to the program (network member). From this point on the messages can be sent and received with the VIs **Read CAN Msg** and **Write CAN Msg**. The registration procedure and withdrawal procedures are shown in figure 3. The use of the enum-variables can be easily recognized here.

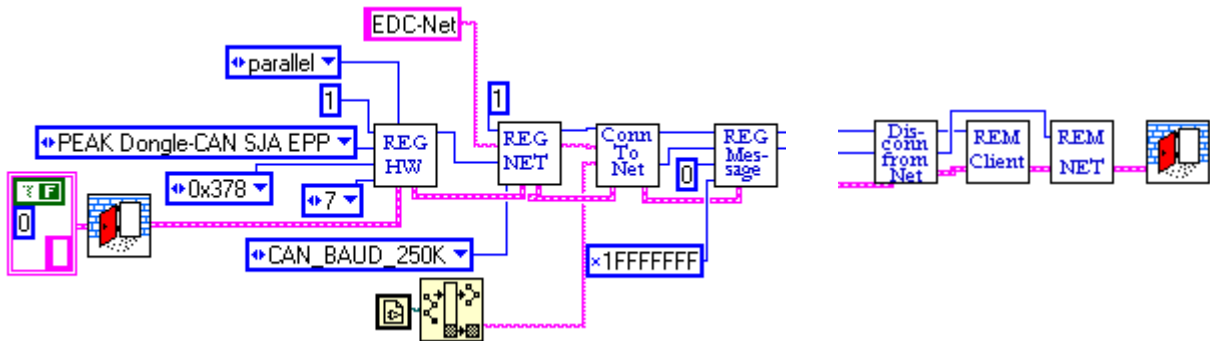


Figure 3 Registration procedure

Withdrawal procedure

Sending and reading messages is quite simple. Figure 4 shows how to send a CAN message. The incoming value for hNet and hClient are the output values of the **REG Message** VI from figure 3. The usage of **Create Constant** creates a CAN MSG cluster. This leads to a non-visible cluster in the front panel, which enables the function **Bundle by Name**. The data is given as an array. Sendtime is the time at which the driver sends the message it has been given. The time is given in the form of a 64-bit number. There are VIs for processing this number included in the [LabView](#) llb as well.

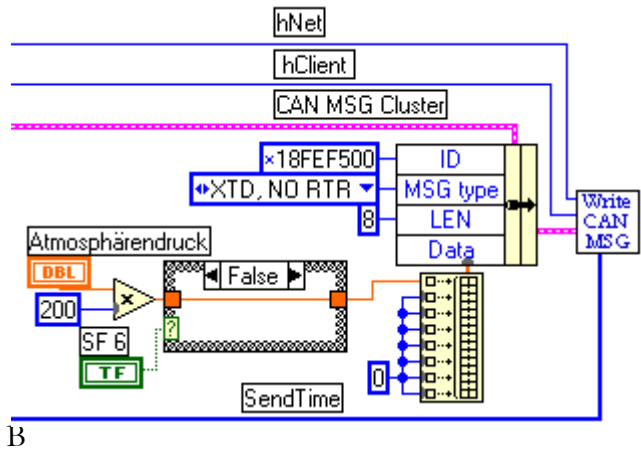


Figure 4 Sending a CAN message

Along with these high-level VIs the API calls shown in figure 5 are included as part of the package as well.

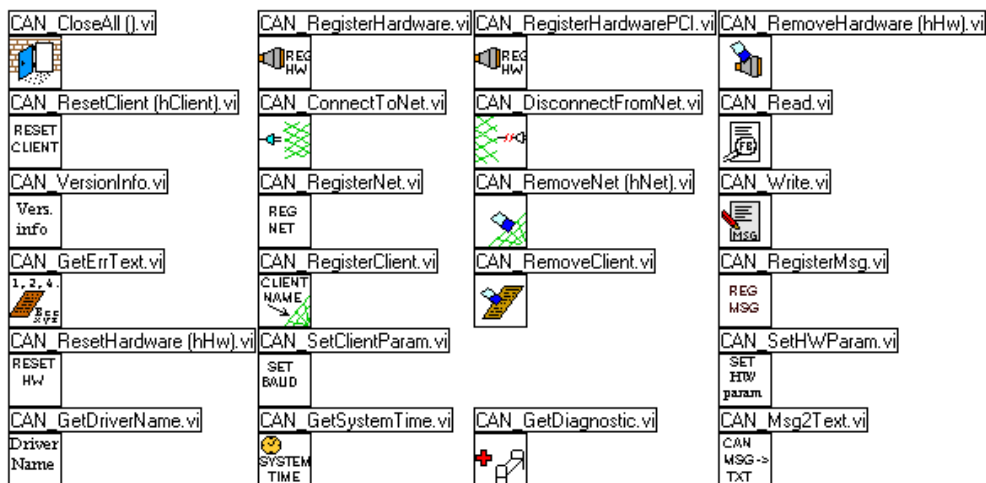


Figure 5 API calls (Low Level Functions)

All VIs are documented via the context help from [LabView](#). The documentation is based on the documentation of the CANAPI2.dll by [PEAK-System Technik GmbH](#). As an example the documentation to the VI **Register Hardware** is shown below in figure 6.

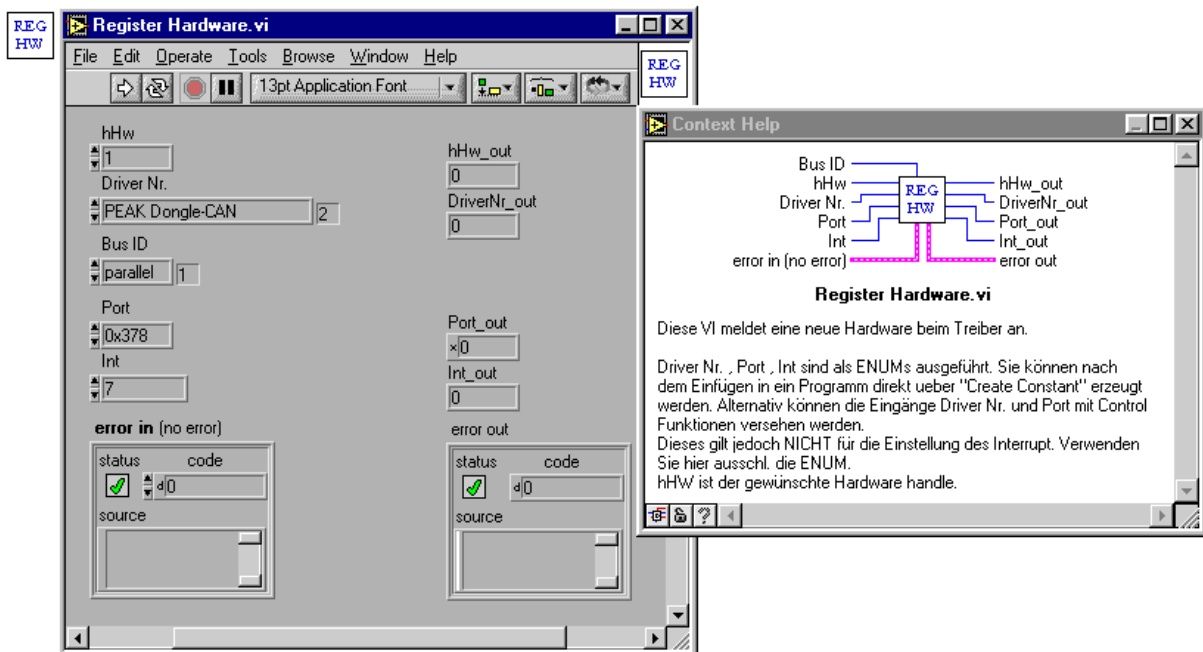


Figure 6 Register Hardware.vi with opened front panel and context help

The LabView interface is available from [PEAK-System Technik GmbH](#). Along with the driver, the package includes examples which enable an easier startup. Please do not hesitate to contact us if you require additional help or information.

Please contact us at:

Dipl.-Ing. (FH) Martin Kunze
labview@peak-system.com