LabVIEW® API for PCAN-Basic 4.0
by
KDI Kunze Digital Instrumentation

CONTENTS

PURPOSE AND SCOPE

This document describes the easy-to-use LabVIEW® API. This API uses the PCAN-Basic 4.0 API from PEAK-System to establish one or more CAN networks. Using the LabVIEW® API, you can exchange CAN messages via CAN 1.0/2.0B and CAN FD with the PCAN hardware. CAN FD is supported by the PEAK device driver from version 4 and the PCAN-Basic API also from version 4.0. The LabVIEW® API is strongly related to the PCAN-Basic 4.0 help files and supports LabVIEW® 2011 or any higher version.

INSTALLATION AND VI LOCATIONS

To install the LabVIEW® API the VI Package Manager is needed. It's available from National Instruments for free. To install the LabVIEW® API, just make a double click on the kdi_digital_instrumentation_lib_labview_api_for_pcan_basic_4.0-1.x.x.x.vip file.

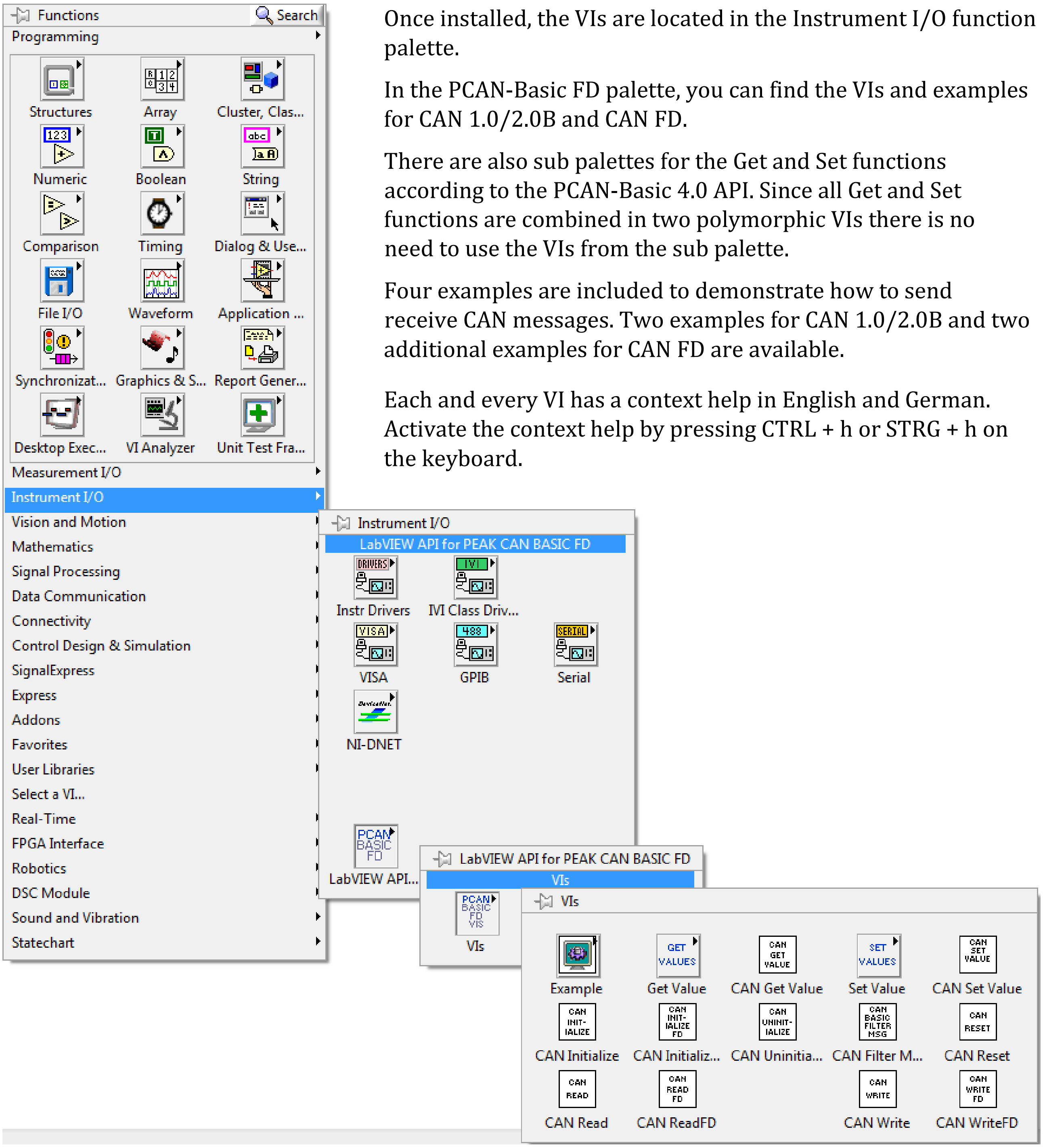
Once installed, the VIs are located in the Instrument I/O function palette.

In the PCAN-Basic FD palette, you can find the VIs and examples for CAN 1.0/2.0B and CAN FD.

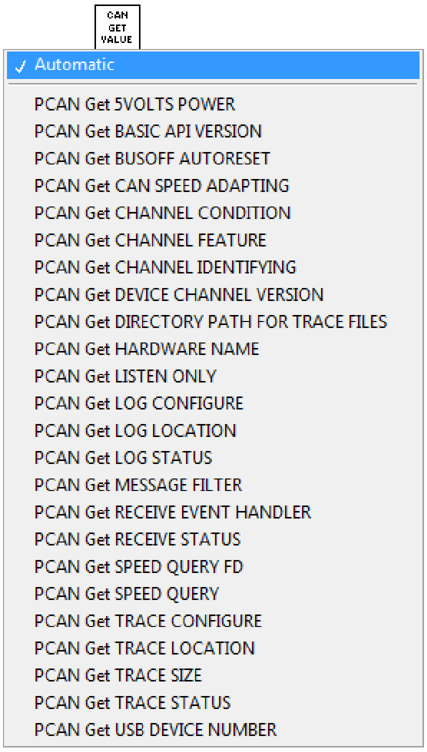
There are also sub palettes for the Get and Set functions according to the PCAN-Basic 4.0 API. Since all Get and Set functions are combined in two polymorphic VIs there is no need to use the VIs from the sub palette.

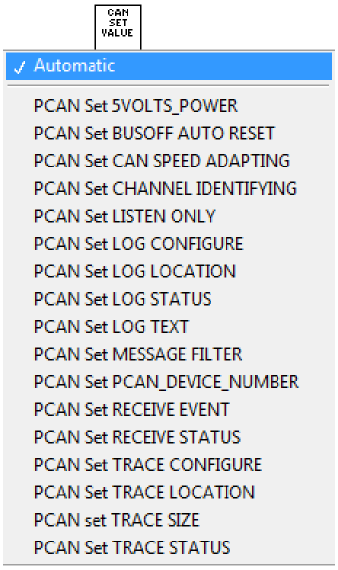
Four examples are included to demonstrate how to send receive CAN messages. Two examples for CAN 1.0/2.0B and two additional examples for CAN FD are available.

Each and every VI has a context help in English and German. Activate the context help by pressing CTRL + h or STRG + h on the keyboard.

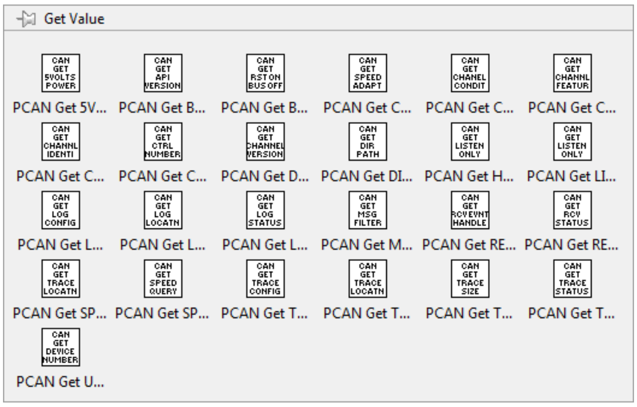


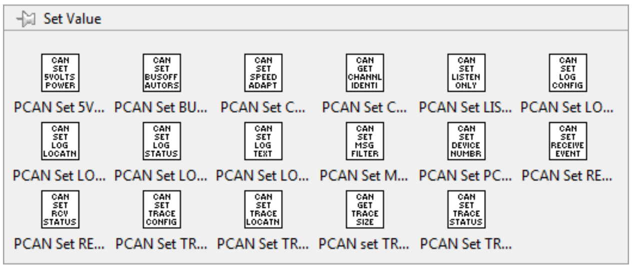
Get and Set Value VIs are accumulated in the VIs CAN Get Value.vi and CAN Set Value.vi. Both VIs are polymorphic to make programming comfortable.





Nevertheless, you can access the Get and Set Value VIs from the VI palette using the sub palettes GET VALUE and SET VALUE.

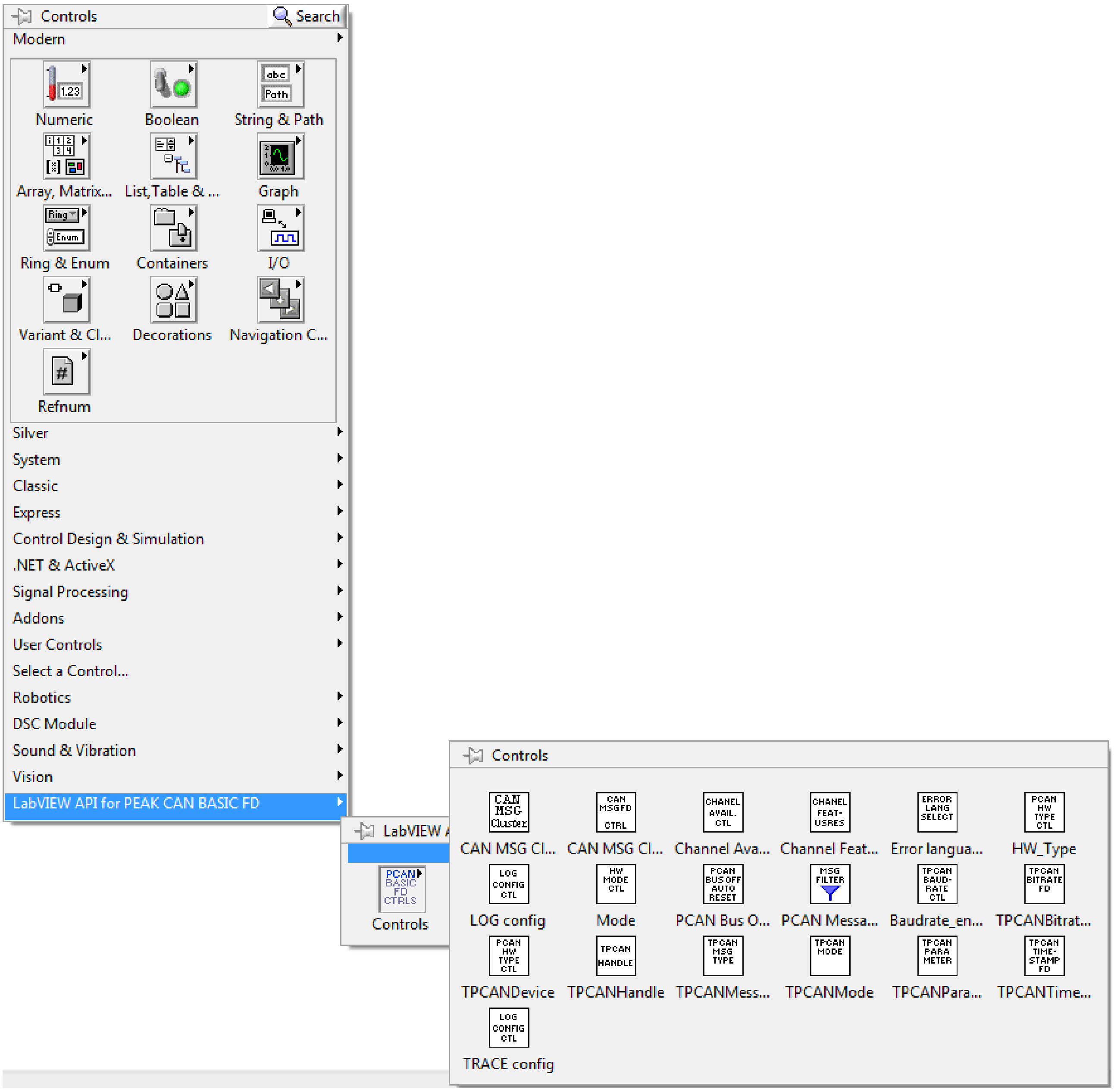




CONTROLS

All controls reside within the LabVIEW® controls palette, too.

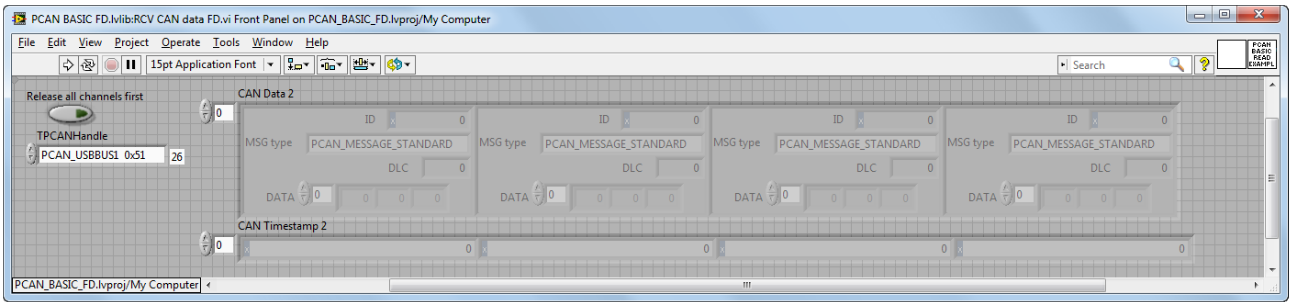
Access the controls via the controls palette to use them on the front panel:



EXAMPLES FOR CAN FD AND CAN 2.0B

Get a first impression of the functionality by having a look into the examples coming with the LabVIEW® API for PCAN-Basic 4.0 (Instrument I/O >> LabVIEW® API for PCAN-BASIC FD >> VIs >> EXAMPLES).

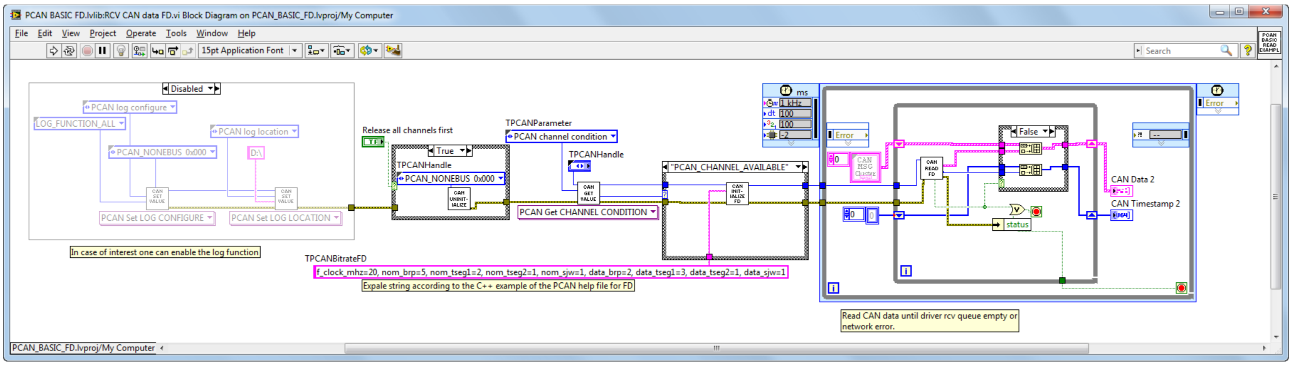
On the left hand side of the front panel is the TPCANHandle control. TPCANHandle must be used to specify the interface, while the two arrays on the right hand side cyclically show the content of the received CAN messages once the VI is running. Since this VI is an FD example, we have to take a look on the block diagram, too, in order to see what is different to CAN 2.0B. The block diagram (from left to right) shows a disabled part of code. This is just an example how to use the log function of the PCAN-Basic 4.0 driver. You can enable this function if needed.


PCAN BASIC FD.lvlib:RCV CAN data FD.vi Front Panel on PCAN_BASIC_FD.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Release all channels first
TPCANHandle
CAN Data 2
ID
MSG type
DLC
DATA
CAN Timestamp 2
PCAN_BASIC_FD.lvproj/My Computer

TPCANHandle: PCAN_USBBUS1 0x51 26

MSG type: PCAN_MESSAGE_STANDARD

The front panel shows the TPCANHandle (channel) and two arrays for messages and timestamps.



In case of interest one can enable the log function

f_clock_mhz=20, nom_brp=5, nom_tseg1=2, nom_tseg2=1, nom_sjw=1, data_brp=2, data_tseg1=3, data_tseg2=1, data_sjw=1

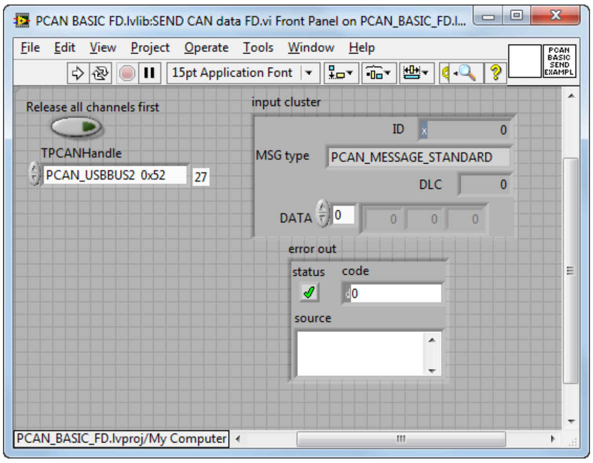
Expale string according to the C++ example of the PCAN help file for FD

Read CAN data until driver rcv queue empty or network error.

The disabled code segment is followed by the initialisation of the TPCANHandle (channel) and the hardware initialisation. Here is the biggest difference between CAN 2.0B and FD. FD is initialised by a string and not by API VIs.

The initialisation is followed by a timed loop. The outer timed loop displays the received CAN messages and according timestamps in two arrays while the inner loop polls the driver queue until its empty. The implemented build array function with shift registers can be replaced by auto indexing. In this case the last array element is always empty. The example just demonstrates how to avoid that the last array element is empty. Other solutions are possible, too.

The functionality of the SEND CAN data FD.vi is slightly similar to the Receive CAN data FD.vi.


PCAN BASIC FD.lvlib:SEND CAN data FD.vi Front Panel on PCAN_BASIC_FD.l...
File Edit View Project Operate Tools Window Help
15pt Application Font
Release all channels first
TPCANHandle
input cluster
ID
MSG type
DLC
DATA
error out
status
code
source
PCAN_BASIC_FD.lvproj/My Computer

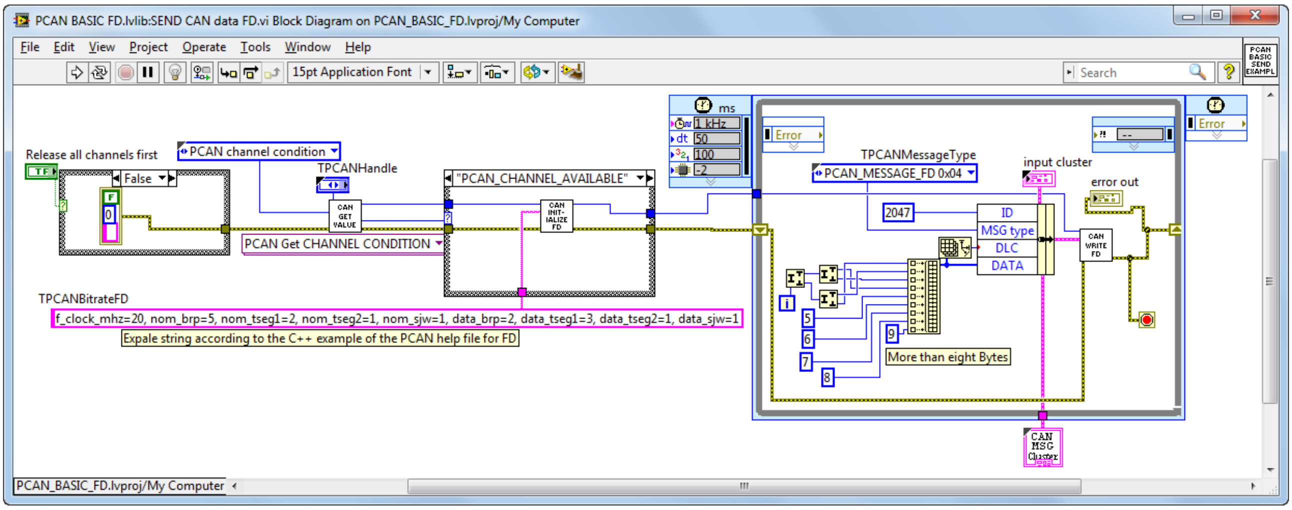
TPCANHandle: PCAN_USBBUS2 0x52 27

MSG type: PCAN_MESSAGE_STANDARD

On the front panel, you can see the TPCANHandle Enum, the send cluster and an error cluster.

The block diagram includes the TPCANHandle initialisation for CAN FD, too, followed by a timed send loop. This loop sends the loop counter value and some additional bytes. As you can see in this case, the CAN message data is nine bytes long while CAN FD massages can be as long as 64 Bytes.

Using two FD devices or one PCAN-USB Pro FD device from PEAK-System allows you to test the example VIs by creating a network to send and receive CAN FD data. Don't forget a proper termination to ensure good results.



PCAN BASIC FD.lvlib:SEND CAN data FD.vi Block Diagram on PCAN_BASIC_FD.lvproj/My Computer

f_clock_mhz=20, nom_brp=5, nom_tseg1=2, nom_tseg2=1, nom_sjw=1, data_brp=2, data_tseg1=3, data_tseg2=1, data_sjw=1

Expale string according to the C++ example of the PCAN help file for FD

More than eight Bytes

The other two examples for CAN 1.0/2.0B are similar except the initialisation and the CAN Read and CAN Write functionality. Have a look into these examples, too. They look like the VIs for the previous PCAN-Basic API but they aren't compatible!

Remark:

Setting the Boolean button TRUE, releases all channels first. This can be useful but should not be set to TRUE in both VIs. Doing this will remove one of the VIs from the connected bus.

Starting and Connecting PCAN-View before the VIs are running may cause a 'Channel occupied error'. To avoid this, connect PCAN-View when the VIs are already running.

MIGRATION FROM LABVIEW® PCAN-BASIC API TO LABVIEW® PCAN-BASIC 4.0 API

Remark:
Skip this if you never have used the PCAN-Basic API before. All the described changes are just related to the LabVIEW® API. The changes have been implemented to achieve compatibility between the LabVIEW® API and the PCAN-Basic 4.0 API.

Implementation of the PCAN-Basic API was done in 2010 with LabVIEW® 2009. Since this time there have been made some minor changes.

The new PCAN-Basic 4.0 API for LabVIEW® 2011 or higher works with CAN 1.0/2.0B and CAN FD devices. The current version is very strongly related to the PCAN-System documentation. All VI names and control names are similar to the PEAK-System documentation. Implementing the PCAN FD functionality forces some changes. According to these changes it is required to change VIs. Unfortunately this leads to incompatibility to the former version of the LabVIEW® API.

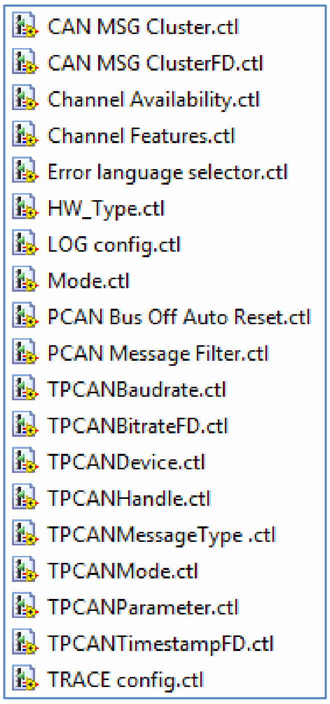
Main differences are:

- Controls
- VIs
- Wrappers (Enum to numeric value)
- Addition of CAN FD VIs

CHANGED CONTROLS

Many controls have been changed or renamed. Only the changes are described in the following.

New CAN FD Controls:

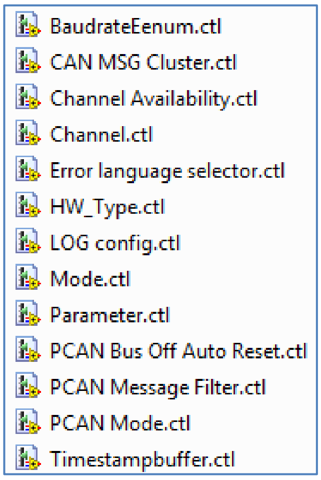


CAN MSG Cluster.ctl
CAN MSG ClusterFD.ctl
Channel Availability.ctl
Channel Features.ctl
Error language selector.ctl
HW_Type.ctl
LOG config.ctl
Mode.ctl
PCAN Bus Off Auto Reset.ctl
PCAN Message Filter.ctl
TPCANBaudrate.ctl
TPCANBitrateFD.ctl
TPCANDevice.ctl
TPCANHandle.ctl
TPCANMessageType .ctl
TPCANMode.ctl
TPCANParameter.ctl
TPCANTimestampFD.ctl
TRACE config.ctl

CAN MSG Clusters for CAN 2.0B and CAN FD are new implemented. They have several changes compared to the old CAN Basic Controls.

Channel Availability:	Not changed
Channel Features:	Return value available in CAN FD
Error language selector:	Not changed
HW_Type:	Not changed
LOG config:	Not changed
Mode:	Not changed
PCAN Bus Off Auto Reset:	Not changed
PCAN Message Filter:	Not changed
TPCANBaudrate:	Previously BaudrateEenum, changed
TPCANBitrateFD:	New
TPCANDevice:	Previously called HW_Type, changed
TPCANHandle:	Previously called channel, changed
TPCANMessageType:	Prev. in CAN MSG Cluster, new, changed
TPCANMode:	New
TPCANParameter:	Prev. called Parameter, changed, enhanced
TPCANTimestamp:	New, format changed to U64
TRACE config:	New

Old CAN Basic Controls:



BaudrateEenum.ctl
CAN MSG Cluster.ctl
Channel Availability.ctl
Channel.ctl
Error language selector.ctl
HW_Type.ctl
LOG config.ctl
Mode.ctl
Parameter.ctl
PCAN Bus Off Auto Reset.ctl
PCAN Message Filter.ctl
PCAN Mode.ctl
Timestampbuffer.ctl

Timestampbuffer:	Array, U16 micros, U32 millis, U16 millis overflow

RECOMMENDATIONS

Migrating from an existing PCAN-Basic LabVIEW® API project to the PCAN-Basic FD LabVIEW® API can be achieved by:

- Uninstall the LabVIEW® API for PCAN-Basic.
- Be sure that no old PCAN-Basic DLL resides on your system.
- Install LabVIEW® API for PCAN-Basic 4.0.
- Install PCAN-Basic 4.0 and copy the DLLs into the right folders:
  - Windows 64-bit: put the 32-bit DLL into the Windows/SysWOW64/ folder.
  - Windows 64-bit: put the 64-bit DLL into the Windows/System32/ folder.
  - Windows 32-bit: put the 32-bit DLL into the Windows/System32/ folder.
- Start LV and load your project.
- Replace all the old VIs and controls by the new ones.

May be easier:

- Uninstall the LabVIEW® API for PCAN-Basic.
- Be sure that no old PCAN-Basic DLL resides on your system.
- Install PCAN-Basic 4.0 and copy the DLLs into the right folders.
  - Windows 64-bit: put the 32-bit DLL into the Windows/SysWOW64/ folder.
  - Windows 64-bit: put the 64-bit DLL into the Windows/System32/ folder.
  - Windows 32-bit: put the 32-bit DLL into the Windows/System32/ folder.
- Load your project and don't try to find the uninstalled VIs.
- Open the examples and copy the required VIs and Controls into your project to replace the old ones.

REFERENCES

LabVIEW® is a Trademark of Notional Instruments

COPYRIGHT

Dipl.-Ing. (FH) Martin Kunze

Im Bühl 5
D-74670 Forchtenberg-Sindringen
E-Mail: labview@peak-system.com