4.4.2013

PEAK-System Technik GmbH

PCAN - Parameters

PCAN-Basic Parameters Description | Dipl.-Ing. Keneth Wagner-Rivas

Index

Introduction

The amount of configurable parameters within the PCAN-Basic has been growing in the last time. It is sometime difficult to figure out when you need to use a specific parameter or how it actually works. Added to this, there are some parameters that support a pre-initialized behavior. But for what is this intended for? This and some other question are tried to be answered here.

Take into consideration that the PCAN-Basic API version used at the moment of writing this document is the version **1.3.0. 47**. Please check your API version and, if necessary, update it.

The changes history that the API has suffered since its first release can be found in our website at http://www.peak-system.com/PCAN-Basic.126.0.html.

If you want to easily keep informed about our products, for example new releases of our free API PCAN-Basic, you can subscribe to our RSS-Feed or you can visit our support website at http://www.peak-system.com/Support.55.0.html.

Supported PCAN-Parameters

PCAN-Basic currently supports 21 parameters that can be read/configured using the functions CAN_GetValue /CAN_SetValue. Not all parameters can be configured because some of them are **read-only** parameters. Following you will find a list with the parameters and their associated value:

- PCAN_DEVICE_NUMBER 1
- PCAN_5VOLTS_POWER 2
- PCAN_RECEIVE_EVENT 3
- PCAN_MESSAGE_FILTER 4
- PCAN_API_VERSION 5
- PCAN_CHANNEL_VERSION 6
- PCAN_BUSOFF_AUTORESET 7
- PCAN_LISTEN_ONLY 8
- PCAN_LOG_LOCATION 9
- PCAN_LOG_STATUS 10
- PCAN_LOG_CONFIGURE 11
- PCAN_LOG_TEXT 12
- PCAN_CHANNEL_CONDITION 13
- PCAN_HARDWARE_NAME 14
- PCAN_RECEIVE_STATUS 15
- PCAN_CONTROLLER_NUMBER 16
- PCAN_TRACE_LOCATION 17
- PCAN_TRACE_STATUS 18
- PCAN_TRACE_SIZE 19
- PCAN_TRACE_CONFIGURE 20
- PCAN_CHANNEL_IDENTIFYING 21

Parameters Groups

In order to delimit the purpose of them, they could be arranged within 5 groups as:

Parameters for “Hardware Identification”:

- PCAN_CHANNEL_CONDITION
- PCAN_DEVICE_NUMBER
- PCAN_HARDWARE_NAME
- PCAN_CONTROLLER_NUMBER
- PCAN_CHANNEL_IDENTIFYING

Parameters for “Informational” purposes:

- PCAN_API_VERSION
- PCAN_CHANNEL_VERSION

Parameters for “Influencing Behavior”:

- PCAN_5VOLTS_POWER
- PCAN_BUSOFF_AUTORESET
- PCAN_LISTEN_ONLY

Parameters for “Data Reading and Flow Control”:

- PCAN_RECEIVE_EVENT
- PCAN_MESSAGE_FILTER
- PCAN_RECEIVE_STATUS

Parameters for “Logging and Debugging”:

- PCAN_LOG_LOCATION
- PCAN_LOG_STATUS
- PCAN_LOG_CONFIGURE
- PCAN_LOG_TEXT

Parameters for “CAN Data Recording (Tracing)”:

- PCAN_TRACE_LOCATION
- PCAN_TRACE_STATUS
- PCAN_TRACE_SIZE
- PCAN_TRACE_CONFIGURE

Pre-Initialized Parameters

The parameter configuration within the PCAN-Basic API, except of the parameters grouped as “Logging and Debugging” (these are not tied to a channel in particular), is allowed ***after*** a channel is successfully initialized. Nevertheless, there are some cases in which it is needed to do some configuration even before a channel is initialized. At the moment, the following parameters are able to be configured on a channel ***before*** it is initialized:

- PCAN_RECEIVE_STATUS
- PCAN_LISTEN_ONLY

Identifying a Hardware

First of all take into account that the first identification takes place when selecting the PCAN-Channel to be used. The channel's name already identifies the bus to use.

PCAN_USBBUS1

The name above tells the API that the PCAN hardware to connect use a kind of bus *USB*, and it is the *first ("1")* hardware ***registered*** in a system. PCAN-Basic allows connecting following interfaces:

- USB: Universal Serial Bus. Up to 8 channels.
- PCI: Peripheral Component Interconnect (including ExpressCard hardware). Up to 8 channels.
- PCC: PC-Card (PCMCIA), Personal Computer Memory Card. Up to 2 channels.
- DNG: Parallel port Dongle. Up to 1 channel.
- ISA: Industry Standard Architecture. Up to 8 channels.

Note that the way of how hardware is registered in a system depends on its controller driver and on the system itself. When several hardware of the same kind is installed in a system (USB for example), it is not guaranteed by default that connecting to PCAN_USBBUS1 after a system restart will still connect to the same hardware.

That is why parameters are used to help on the detection of the right hardware. The following parameters are used to identify the physical hardware to connect, for example when several devices are available for connection.

PCAN_CHANNEL_CONDITION

This parameter is used to identify the state of use of a PCAN-Channel. For example, a connection is only possible when a PCAN-Channel is available, which means:

- It is valid: The PCAN-Channel is one of the listed in the section "Supported by" bellow.
- It isn't initialized: The PCAN-Channel is not being used, or it is being used by a PCAN-View application.

Availability

It is available since version 1.0.0. Nevertheless, due to some bugs, it actually worked well beginning with the version 1.0.4.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter can only be read. It cannot be modified.

Possible Values

The condition of a PCAN-Channel can be one of the following defined values:

Defined Value	Description
PCAN_CHANNEL_UNAVAILABLE	The channel is invalid or is not present.
PCAN_CHANNEL_AVAILABLE	The channel can be used.
PCAN_CHANNEL_OCCUPIED	The channel was already initialized.

Default Value

Not apply.

Initialization Status

Not relevant, since this parameter is used to ask the current status of a PCAN-Channel.

When to Use

It can be used when it is need to know the availability status of a particular channel (or all channels) registered in a system at a given time.

Application – Example of Use

Imagine you want to create a Test-Application that connects to a PCAN-PCI device that allows the user to decide which PCAN-Channel should be used for data transmission. For this you have to list all available PCAN-PCI Channels. Using this parameter you can filter the channels that are occupied or unavailable:

```
Repeat From PCAN_PCIBUS1 To PCAN_PCIBUS8
{
     Get the value CHANNEL_CONDITION on Channel-X (PCAN_PCIBUSX)
     If "CHANNEL_CONDITION Equals PCAN_CHANNEL_AVAILABLE" Then
     {
          Include Channel-X to the AvailableChannels list
     }
}
Show The PCAN-Channels available for connection are:
Print List AvailableChannels
```

PCAN_CHANNEL_IDENTIFYING

This parameter is used to physically identify an USB-based PCAN-Channel being used. The identification is done using the status LED that USB devices have. At the moment PEAK-System offers USB devices of two different generations:

- First Generation: PCAN-USB, PCAN-Hub.
- Second Generation: PCAN-USB Pro, PCAN-USB2

According with the hardware used, the blinking of the LED is different in color and blink rate:

- First Generation: Blink color is RED, and the blink rate is about 300 milliseconds.
- Second Generation: Blink color is ORANGE, and the blink rate is about 250 milliseconds.

Availability

It is available since version 1.3.0.

Supported By

PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter represents a procedure used for identification that can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The identifying procedure is set to OFF.
PCAN_PARAMETER_ON	The identifying procedure is set to ON.

Note that only one channel can be activated at a time. In order to switch on the identifying procedure in another channel, the previous one must be first switched off.

Default Value

The default state of this identification procedure is off (PCAN_PARAMETER_OFF). After switching it on, the LED of an USB device stays blinking until it is expressly turned off.

Initialization Status

This parameter can be used with both, initialized and uninitialized PCAN-Channels. **Note** that the activation of this identification procedure doesn't affect any communication that can occur on the device while it is being identified.

When to Use

It can be used when an application can connect to several USB devices and it is not clear which (physical) channel has to be used in a determined time, for example, before establishing a connection to a channel. It is also useful in application that communicate with several USB devices at the same time and for long time of periods (or applications used for several persons), in order to check with channels are being used in a determined time.

Application – Example of Use

Let's say you have an application communicating with several USB devices (5 for example). This application is working on a computer in which the order of the devices representing each PCAN-Channel can vary (the computer reboots automatically within a given period of time, the physical CAN networks are eventually swapped, etc.). Now, you come to the application and you need to disconnect a device, but you don't know which PCAN-Channel is associated to it, and you don't want to disturb the other channels. You can write a small application that just turns on the identifying procedure on a given channel, so that you can see which device is the one you are looking for:

```
Repeat From PCAN_USBBUS1 To PCAN_USBBUS5
{
    Set PCAN_CHANNEL_IDENTIFYING on Channel-X (PCAN_USBBUSX) to ON
    If "Identifying Procedure of Channel-X was activated"
    {
        Show Channel-X is being identified. Click OK to continue...
        Set PCAN_CHANNEL_IDENTIFYING on Channel-X (PCAN_USBBUSX) to OFF
    }
}
```

PCAN_DEVICE_NUMBER

This parameter applies ONLY to hardware of type PCAN-USB. It is used to distinguish between 2 or more hardware of this kind connected to a computer simultaneously. A Device Number is a

persistent value stored in the flash memory of each USB device, i.e. the value is not lost after disconnecting the hardware.

Note that the devices can have the same identification number. It is job of the user to guarantee that the used devices are configured with different identifiers, so that a differentiation through a device number can work.

Availability

It is available since version 1.0.0.

Supported By

PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

According with the firmware version of the PCAN-USB device, this value can have a resolution of a byte (range [0...255]) or a double-word (range [0...4294967295]). If the value wanted to set is bigger than the resolution supported for the firmware, then the value is truncated.

Default Value

If this parameter was never set before, the value is the maximum value possible for the used resolution which is 255 (FFh), or 429496729 (FFFFFFFFh).

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when it is need to differentiate between PCAN-USB devices connected to the same system at a given time.

Application – Example of Use

Let’s say you want to write an application that read data form one CAN-BUS and reply it to a second CAN-BUS (a.k.a. Gateway application). For this you would have one PCAN-USB connected to each CAN-BUS. You could set the device number of both PCAN-USBs so that you know which bus is used for writing (for example, **number 1** for the “to write to” bus), and which bus is used for reading (for example, **number 2** for the “to read from” bus). Using this parameter you would be able to know if both channels are available and also which device use for sending and which one for writing:

```
Repeat From PCAN_USBBUS1 To PCAN_USBBUS8
{
     Initialize the current Channel-X (PCAN_USBBUSX)
     If "Channel-X was initialized" Then
     {
          Get the value DEVICE_NUMBER
          If "DEVICE_NUMBER Equals 1" Then
          {
               Mark Channel-X as: WRITE_BUS
          }
          If "DEVICE_NUMBER Equals 2" Then
          {
               Mark Channel-X as: READ_BUS
          }
          Uninitialize Channel-X
     }
}
If "READ_BUS was found" AND "WRITE_BUS was found" Then
{
     Show Both Channels were found. Starting ...
     Start working
}
Else
{
     Show Error! Not all Channels found. Terminating...
     Terminate
}
```


PCAN_HARDWARE_NAME

This parameter is used to retrieve a description text from the hardware represented by a PCAN channel. This text allows the recognition of device's models that use the same interface, for example USB. A normal PCAN USB adaptor would return "PCAN-USB" while the new dual CAN/LIN channel adaptor would return "PCAN-USB Pro".

Availability

It is available since version 1.0.6.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter can only be read. It cannot be modified.

Possible Values

The value is a null-terminated string which contains the name of the hardware specified by the given PCAN channel. This string has a maximum length of 32 bytes (null-termination character included).

According with the hardware model represented by the current PCAN-Channel, the following text can be returned:

Hardware Name Value	Interface	Hardware Description
PEAK ISA-CAN	PCAN-ISA	PCAN-ISA, PCAN-PC/104
PEAK ISA-CAN SJA	PCAN-ISA	PCAN-ISA, PCAN-PC/104 with a SJA1000
PEAK Dongle-CAN	PCAN-DNG	PCAN-Dongle with a 82C200
PEAK Dongle-CAN EPP	PCAN-DNG	PCAN-Dongle with a 82C200, using EPP mode
PEAK Dongle-CAN SJA	PCAN-DNG	PCAN-Dongle with a SJA1000
PEAK Dongle-CAN SJA EPP	PCAN-DNG	PCAN-Dongle with a SJA1000, using EPP mode
PEAK Dongle-Pro	PCAN-DNG	PCAN-Dongle Pro
PEAK Dongle-Pro EPP	PCAN-DNG	PCAN-Dongle Pro in EPP mode

PCAN-PCI	PCAN-PCI	PCAN-PCI, PCAN-PCI Express, PCAN-PC/104-Plus
PCAN-ExpressCard	PCAN-PCI	PCAN-ExpressCard
PCAN-USB	PCAN-USB	PCAN-USB Adapter, PCAN-USB Hub
PCAN-USB Pro	PCAN-USB	PCAN-USB Pro
PCAN-PCCARD-CAN	PCAN-PCC	PCAN-PC Card

Default Value

Not apply.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when it is needed to differentiate between several hardware models using the same interface (e.g. PCAN-PCI, PCAN-ExpressCard)

Application – Example of Use

Suppose the following scenario: You want to develop a Diagnostic-Application using a normal PCAN-USB device for data transmission. The program should run in computers that have per default a PCAN-USB Pro attached, intended to be used from another programs (for ECU controlling, Gateway configuration purpose, etc), and therefore they shouldn't be occupied. This means that the system will have 3 PCAN channels registered (PCAN_USBBUS1 to PCAN_USBBUS3). Since the diagnostic network will be always plugged-in to your PCAN-USB, your application must be sure to connect the single channel and not one of the PCAN-USB Pro channels. Using this parameter you would be able to identify which PCAN-Channel represents a PCAN-USB and which one a PCAN-USB Pro:

```
Repeat From PCAN_USBBUS1 To PCAN_USBBUS3
{
     Initialize the current Channel-X (PCAN_USBBUSX)
     If "Channel-X was initialized" Then
     {
          Get the value HARDWARE_NAME
          If "HARDWARE_NAME EquaTs PCAN-USB" Then
          {
               Mark Channel-X as: DEBUG_BUS
               Exit Repeat
          }
          Uninitialize Channel-X
     }
}
If "DEBUG_BUS was found" Then
{
     Show DEBUG-BUS found, connected, and ready to work...
     Start working
}
Else
{
     Show Error! Single PCAN-USB Channel was not found. Terminating...
     Terminate
}
```

PCAN_CONTROLLER_NUMBER

This parameter is used to identify the physical CAN channel index of a multichannel CAN hardware (PCAN-PCI, PCAN-USB Pro, etc). This index is zero-based, so that the first channel on a device is 0, the second 1, and so on.

Availability

It is available since version 1.2.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter can only be read. It cannot be modified.

Possible Values

A number in the range [0…**n-1**], where **n** is the number of physical channels on the device being used. The correspondence between an index number and the CAN channel description on the hardware etiquette is:

Channel Index	Channel Etiquette
0	CAN 1
1	CAN 2
n-1	CAN n

Default Value

Not apply.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used to determine which physical channel of a multichannel PCAN device has to be connected.

Application – Example of Use

The easy case, let's say you want to write an application that should work only with the second channel of any PCAN-USB device. You could just ask for the PCAN_CONTROLLER_NUMBER on each available USB channel until you find the channel you are looking for:

```
Repeat From PCAN_USBBUS1 To PCAN_USBBUS8
{
     Initialize the current Channel-X (PCAN_USBBUSX)
     If "Channel-X was initialized" Then
     {
          Get the value CONTROLLER_NUMBER
          If "CONTROLLER_NUMBER Equals 1" Then
          {
               Mark Channel-X as: CHANNEL_CAN2
          }
     }
     Uninitialize Channel-X
}
If "CHANNEL_CAN2 was found" Then
{
     Show CAN-Channel Two was found
     Start working
}
Else
{
     Show Error! CAN-Channel Two not found.
     Terminate
}
```

The complicated case, you want to use the second channel of a specific PCAN-USB Pro hardware, device number 7 for example, and there exists the possibility to have several multi-channels devices attached to the computer at a time. Using the parameter PCAN_HARDWARE_NAME let you find any PCAN-USB Pro connected. Using the parameter PCAN_DEVICE_NUMBER let you find the right Device (number 7). Finally, using the PCAN_CONTROLLER_NUMBER let you find the right CAN channel to use:

```
Repeat From PCAN_USBBUS1 To PCAN_USBBUS8
{
     Initialize the current Channel-X (PCAN_USBBUSX)
     If "Channel-X was initialized" Then
     {
          Get the value HARDWARE_NAME
          If "HARDWARE_NAME EquaTs PCAN-USB Pro" Then
          {
               Get the value DEVICE_NUMBER
               If "DEVICE_NUMBER EqUals 7" Then
               {
                    Get the value CONTROLLER_NUMBER
                    If "CONTROLLER_NUMBER EqUals 1" Then
                    {
                         Mark Channel-X as: DEV7_CAN2
                    }
               }
          }
          Uninitialize Channel-X
     }
}
If "DEV7_CAN2 was found" Then
{
     Show PCAN-USB Pro (Device Nr. 7 / CAN-Channel Two) was found
     Start working
}
Else
{
     Show Error! PCAN-USB Pro (Device Nr. 7 / CAN-Channel Two) not found.
     Terminate
}
```

Using Informational Parameters

These parameters are intended to give versioning information about the API itself, as well as about the Hardware (e.g. device driver version). This is important since different features can or cannot be available according with the versions being used.

To be sure that a PCAN-Basic software works properly with a specific hardware, it is a good idea to check version parameters at the beginning (after connect). In this way, you can ensure that the software will work by the user as it was working by you at develop time.

Note that when dependences between a PCAN-Parameter and the API and/or driver/firmware Version appear, they will be notified and remarked in the Online-Help of the PCAN-Basic, as well as in our Website (e.g. Forum).

PCAN_API_VERSION

This parameter is used to get the API implementation version.

Availability

It is available since version 1.0.0.

Supported By

All channels: Due to the API structure, a channel value is needed in order to get a PCAN-Parameter when using the function CAN_GetValue. But since the API version doesn't depend on a specific channel, any channel value can be used, including PCAN_NONEBUS.

Access Mode

This parameter can only be read. It cannot be modified.

Possible Values

The API version value is represented as a string of the form "a,b,c,d", where:

a: represents the major version number.
b: represents the minor version number.
c: represents the release version number.
d: represents the build number.

All four values have a maximum size of 16 bits that allows a value of 65535 per each. The returned value is a null terminated string with a maximum length of 24 bytes. It is recommended to use a buffer that large to guaranty success in any case.

Default Value

Not apply.

Initialization Status

Not relevant, since this parameter is not channel dependant.

When to Use

It can be used to determine if a feature is available or not to be used, or just as informative output in an application.

Application – Example of Use

Let's say that you want to show from your application a list of the APIs and libraries being used with their versions, so that if any problem appears then a user can get back to you with versioning information.

```
Get the value PCAN_API_VERSION on PCAN_NONEBUS
Show The PCAN-Basic version used is:
Print PCAN_API_VERSION
```

PCAN_CHANNEL_VERSION

This parameter is used to obtain information about the underlying device driver of a PCAN device being used as well as to obtain copyright information.

Availability

It is available since version 1.0.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter can only be read. It cannot be modified.

Possible Values

The information about driver version and copyright is represented as a multiline string (4 lines) offering the following information in each line:

1) Device driver name and driver version.
2) Architecture implemented on the driver and targeted platform.
3) Year of Copyright.
4) Name from the company and from the city where its head office is located.

Note that this format is available beginning with the device driver version 3.x. The returned value is a null terminated string with a maximum length of 256 bytes (null termination included). It is recommended to use a buffer that large to guaranty success in any case.

Default Value

Not apply.

Initialization Status

Not relevant, since this parameter refers to device driver used for a given channel. Device drivers are loaded on Windows start and unloaded again on Windows shutdown.

When to Use

It can be used as informative output about the used driver in an application.

Application – Example of Use

Let's say that your application is distributed without hardware, so that there is the possibility that a user can have a device with a version you did not tested. Using this parameter avoids

time losing, looking for an error that actually is not in your software but in the use of a wrong or old driver.

```
If "An unspected error occurred on Channel-USED" Then
{
     Get the value PCAN_API_VERSION on Channel-USED
     Get the value PCAN_CHANNEL_VERSION on Channel-USED
     Show Unknown error while working with Channel-USED
     Show Contact our support indicating the following data:
     Print PCAN_API_VERSION
     Print PCAN_CHANNEL_VERSION
     Terminate
}
```

Using Special Behaviors

These parameters are intended to activate some modes on the devices being used that cause those devices to react or work in an exceptional way.

Note that not all modes are supported by all kind of devices.

PCAN_5VOLTS_POWER

This parameter is used for switching the external 5V on the D-Sub connector of a PCAN-Device. This is useful when connecting external bus converter modules to the card (AU5790 / TJA1054)).

Availability

It is available since version 1.0.0.

Supported By

PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

*PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8). **Note** that only the devices of type “PCAN-USB Hub” can support this parameter.

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter represents an extra voltage that can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The external 5V on the D-Sub connector is inactive.
PCAN_PARAMETER_ON	The external 5V on the D-Sub connector is active.

Default Value

The default state of extra voltage is inactive (PCAN_PARAMETER_OFF). After activating it, the extra 5V stays on the D-Sub until it is expressly deactivated, or the device is reinitialized (plugged-out and plugged-in again, or PC-reboot).

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when connecting external bus converter modules to a device, so that it is also supplied with power.

Application – Example of Use

Let’s say that your application is connected to a Single-Wired CAN network using a PC-Card Channel. A Bus-Converter (e.g. High-speed to Single-Wire CAN) is also connected to the channel used. It will be used only in special cases when you want to transfer software or

diagnostic data. You will need to use the PCAN_5VOLTS_POWER to allow the adapter to work.

```
Set the value PCAN_5VOLT_POWER of Channel-USED to ON.
If "Channel-USED 5V Power is active" Then
{
     Show Channel-USED has now 5V power in D-Sub
     Do needed work/communication.
     Set the value PCAN_5VOLT_POWER of Channel-USED to OFF
     If "Channel-USED 5V Power is inactive" Then
          Show The 5V power on Channel-USED is now deactivated.
     Else
     {
          Show Warning: the 5V Power couldn't be disabled.
          Show ....Risk of damage if short circuit....
     }
}
Else
     Show 5V Power couldn't be enabled.
```


PCAN_BUSOFF_AUTORESET

This parameter instructs the PCAN driver to reset automatically the CAN controller of a PCAN Channel when a bus-off state is detected.

Availability

It is available since version 1.0.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The automatic Hardware reset is OFF.
PCAN_PARAMETER_ON	The automatic Hardware reset is ON.

Default Value

The default state of the automatic reset on bus-off is inactive (PCAN_PARAMETER_OFF). After activating it, the automatic reset stays active until it is expressly deactivated, or the channel is disconnected (e.g. using the function CAN_Uninitialize).

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when it is needed to avoid resetting an application manually connecting external bus converter modules to a device, so that it is also supplied with power.

Application – Example of Use

Let's say that your application makes some diagnostic on an Electronic Control unit (ECU) of a car, and this ECU is battery powered (car switch on and off). Having an application communicating to the same CAN Network and having the ECU switching on and off can causes the PCAN-Channel (hardware, CAN Controller) to reach the OFF status. No communication can be achieved until the OFF status disappears. To avoid the need to manually reset the application/PCAN-Channel each time the car is switch on or off, you can use this parameter to do this automatically for you:

```
Set the value PCAN_BUSOFF_AUTORESET of Channel-USED to ON.
If "Autoreset on Bus-OFF on Channel-USED is active" Then
{
     Show Channel-USED will reset itself automatically on Bus-OFF
     Do needed work/communication.
}
Else
     Show Autoreset on Bus-OFF couldn't be enabled.
```


PCAN_LISTEN_ONLY

This parameter allows the user to set the CAN device represented by a PCAN-Channel in Listen-Only mode. When this mode is set, the CAN controller doesn't take part on active events (e.g. transmit CAN messages) but stays in a passive mode (CAN monitor), in which it can analyze the traffic on the CAN bus used by a PCAN channel. See also the Philips Data Sheet "SJA1000 Stand-alone CAN controller".

This parameter is a so called "**pre-initialized**" parameter, which means that it can be set before a PCAN-Channel is initialized in order to activate/deactivate the parameter as fast as possible, avoiding in this way problems that can appears within sensitive operations.

Availability

It is available since version 1.0.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The Listen-only mode is OFF.
PCAN_PARAMETER_ON	The Listen-Only mode is ON.

Default Value

The default state of the Listen-Only mode is deactivated (PCAN_PARAMETER_OFF). After activating it, the Listen-Only mode stays active until it is expressly deactivated, or the channel is disconnected (e.g. using the function CAN_Uninitialize).

Initialization Status

This parameter can be used in initialized or uninitialized channels.

When to Use

It can be used when an application want to passively inspect the data being transferred within a CAN network, without causing any perturbation on it.

Application – Example of Use

Let's say that your application has to work in an environment where only 4 different baud rates are used. Since the 4 baud rates are known you want to offer the possibility to auto detect the baud rate that is currently configured in a CAN network at connection time. You could use this parameter to passively connect to a network using different baud rates without causing errors when connecting with a wrong baud rate. In this way your application can recognizes the baud rate being used, and the communication is not affected while this procedure is done:

```
Repeat From BaudRate1 To BaudRate4
{
     Set PCAN_LISTEN_ONLY on Channel-ToUse to ON
     If "Listen Only mode on Channel-ToUse was activated" Then
     {
          Initialize the Channel-ToUse with BaudRate-X
          If "Channel-ToUse was initialized" Then
          {
               If "Any message received" Then
               {
                    Mark BaudrateX as: BaudRate-X
                    Exit Repeat
               }
               Uninitialize the Channel-ToUse
          }
          Else
          {
               Show Channel cannot be initialized. Terminating...
               Terminate
          }
     }
}
If "BaudrateX was found" Then
{
     Show Baud Rate found, connected, and ready to work...
     Start working
}
Else
{
     Show Error! Baud Rate couldn't be determined. Terminating...
     Terminate
}
```

Controlling the Data Flow

These parameters are intended to control the data being received through a PCAN-Channel, how it is received, and even how/when an application should check for new incoming data. According with the amount of information being transmitted within a CAN network it will reasonable to delimit the data being accepted by an application in order to facilitate the work with it.

Receiving a lot of data but having to process just a part of it can cause unnecessary use of memory and CPU processing, slowing down a system. In the same way, the reaction time for reading incoming data is also the key for successful processing of incoming information.

PCAN_RECEIVE_EVENT

This parameter passes an event handle (Windows Event Objects) to the underlying API. This event will be triggered (it states is set to "signaled") when CAN data is placed into the receive queue of a PCAN-Channel.

Events are normally used when an application separates processing in different execution threads. In a thread, that waiting for an event to occur doesn't affect the normal execution of an application.

Note that the event is not triggered each time a message is included into the queue, but only when it states was "not signaled" and data is received. When an event is signaled, then you have to read the queue until emptiness and eventually reset the event (if you are using a manual reset event).

Availability

It is available since version 1.0.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be enabled or disabled.

Status	Value needed
ENABLED	Valid event object handle, returned by the Windows function CreateEvent.
DISABLED	0, or NULL, or IntPtr.Zero (managed

environments).

Default Value

The default state is disabled (0). After enabling this parameter (by configuring an event handle), the PCAN-Basic API will try to signal the handle until it is disabled (by setting as handle a value of 0), or the channel is disconnected (e.g. using the function CAN_Uninitialize).

Note that when you need to reinitialize a PCAN-Channel, you will need to set the event again each time after initializing the channel, since the event will have again its default value of 0 after initialization. **Note** too that it is strongly recommended to close the handle (using CloseHandle) ***after*** a PCAN-Channel has been uninitialized, since the API could try to set an invalid handle and this can cause undesired behavior.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used to avoid timeouts: when an application wants to react and process information as fast as possible. It can be used to avoid unnecessary data polling: when an application should check for specific messages that are seldom received and/or it is unknown when they can arrive.

Application – Example of Use

Let's say you have written a diagnostic application used for data update on a device (e.g. Electronic Control Unit). The application must wait until the device is initialized and then has to send a message to set the device in maintenance mode. The device has to response within the first 10 milliseconds after receiving the maintenance message, otherwise means it cannot enter the desired mode. For this, you would start a thread that send the request and wait for a response:

```
InitializeFunction
{
    Initialize the Channel-ToUse
    Create AutoResetEven with CreateEvent
    Mark AppMode: Normal-mode
    Start ThreadFunction
}

ThreadFunction
{
    Set PCAN_RECEIVE_EVENT on Channel-ToUsed to AutoResetEvent
    If "Receive event on Channel-ToUsed was set" Then
    {
        Send Diagnostic-Message
        If "Diagnostic-Message sent" Then
        {
            Wait until AutoResetEvent is signaled or timeout(10)...
            Read Message
            If "Message is maintenance-confirmation" Then
                Mark AppMode: Maintenance-mode
        }
        Set PCAN_RECEIVE_EVENT on Channel-ToUsed to 0
    }
}

MainFunction
{
    If "AppMode is Maintenance-Mode" Then
        Show Application is in MAINTENANCE mode
    Else
        Show Application is in NORMAL mode
}
```

PCAN_MESSAGE_FILTER

This parameter instructs a PCAN-Channel to receive or not messages by modifying the acceptance mask and acceptance code of its CAN chip.

Note that an internal hardware reset is done when the acceptance mask and code have to be modified. If other application is using the same device, its communication could be affected in some scenarios.

Availability

It is available since version 1.0.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

In a setting operation, this parameter can be opened or closed.

Defined Value	Description
PCAN_FILTER_OPEN	The CAN filter allows all messages to pass.
PCAN_FILTER_CLOSE	The CAN filter discards all messages.

In a getting operation, a third value can be received.

Defined Value	Description
PCAN_FILTER_CUSTOM	The CAN filter allows a custom range of messages to pass.

Default Value

The default state of the filter is to receive all messages (PCAN_FILTER_OPEN). **Note** that a PCAN-Channel starts receiving any message being transmitted with a CAN network immediately after the channel is initialized. **Note** also that using the function CAN_FilterMessages will cause the filter to be closed automatically before registering the desired message range, if the filter state before calling the function was PCAN_FILTER_OPEN.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used for switching the acceptance of messages in a given time, for example to avoid receiving unwanted messages during a defined period of time.

Application – Example of Use

Let's say you have an application reading and interpreting a considerable amount of information from a CAN network and showing it in some visual controls. Because the data fluctuate too fast you would require checking the general status of the data at some time, but you don't have the possibility to freeze the information being sent within the network. You could close the CAN filter for a while, so that the last received information stays on the visual controls let you time to check it:

```
Set the value PCAN_MESSAGE_FILTER of Channel-USED to CLOSE.
If "Filter is closed" Then
{
     Show Filter is closed.
     Do needed checking
     Show Check is finished. Enabling communication again...
     Set the value PCAN_MESSAGE_FILTER of Channel-USED to OPEN.
     If "Filter is opened" Then
          Show Filter is open.
     Else
          Show Error: Filter couldn't be restablished.
}
Else
     Show Error: Filter couldn't be closed.
```


PCAN_RECEIVE_STATUS

This parameter helps the user to allow / disallow the reception of messages within a PCAN - Channel, regardless of the value of its reception filter. The acceptance filter of the PCAN-Channel remains unchanged (other applications working with the same PCAN-Hardware will not be disturbed).

This parameter is a so called "**pre-initialized**" parameter, which means that it can be set before a PCAN-Channel is initialized in order to activate/deactivate the parameter as fast as possible, avoiding in this way problems that can appears within sensitive operations.

Availability

It is available since version 1.1.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The message receiving status is OFF.
PCAN_PARAMETER_ON	The message receiving status is ON.

Default Value

The default value of the receive status is activated (PCAN_PARAMETER_ON). After deactivating it, the receiving status stays inactive until it is expressly reactivated, or the channel is disconnected (e.g. using the function CAN_Uninitialize).

Initialization Status

This parameter can be used in initialized or uninitialized channels.

When to Use

It can be used on applications that want to discard messages for a while, without having to take modification on the message filter, avoiding disturbances within the device being used.

Application – Example of Use

Let's say you have an application that uses a complicated filter, for example, twelve different message ranges. In a certain time you need to stop receiving messages for a while without needing to configure the filter again, an intrinsically avoid a reset of the CAN controller (done when the filter must be re configured):

```
Set the value PCAN_RECEIVE_STATUS of Channel-USED to OFF.
If "Not Receiving Messages" Then
{
     Show Message receiving is disabled
     Do needed operations
     Set the value PCAN_RECEIVE_STATUS of Channel-USED to ON.
     If "Receiving Messages" Then
          Show Normal operation restablished. Message Receiving enabled.
     Else
          Show Error: Receiving status couldn't be restablished.
}
Else
     Show Error: Receiving status couldn't be set to OFF
```

Using Logging Parameters

These parameters are intended to support the developing phase of a PCAN-Basic project by helping with debug operations. Using the logging system can help finding logic problems within the use of the API, detecting problems with the data being sent or received, checking parameter data, commands order, etc.

It is also possible to activate / deactivate and configure the logging functionality without having to change the code of an application, which allows later debugging session after an application is already released. More information about this can be found in the online forum, Activate debug-logging over Windows Registry, or in Appendix A.

The logging functionality is not tied to a PCAN-Channel in particular but to the use of the PCAN-Basic library itself. This implies three important points:

- The PCAN-Channel handle to use in any CAN_GetValue / CAN_SetValue must be PCAN_NONEBUS, if any PCAN_LOG_* parameter is used. Any other value will cause the function to fail.
- The data logged corresponds to the API calls issued by the process that has loaded the PCAN-Basic dll.
- You cannot start a debug session for different threads of the same application.

PCAN_LOG_LOCATION

This value is used to set the folder on a computer in where the Log-File will be stored, within a debug session.

Note that setting this value starts recording debug information automatically. You could include calls to this parameter in any part of your code that normally shouldn't has to be executed, so you will be notified through the log file if this point was reached (as a kind of assert).

If a debug session is running (a log file is being written), PCAN_LOG_LOCATION instructs the API to close the current log file and to start the process again with the new folder information. **Note** too that the name of the log file cannot be specified. The name of the log file is always **PCANBasic.log**.

Availability

It is available since version 1.0.0.

Supported By

PCAN-NONBUS: Logging parameters are used globally, i.e. they are not tied to a specific PCAN-Channel, but to a specific process.

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This value is a string containing a fully-qualified and valid directory path on the executing computer. In order to use the default path (calling process path) an empty string must be set.

Kind of Path	Value needed
CUSTOM Path	A valid directory string (Files and Paths).
DEFAULT Path	Empty string (calling process folder).

Default Value

The default value is the path to the calling process folder.

Initialization Status

Not apply. It is not needed to have any PCAN-Channel initialized in order to use this parameter.

When to Use

It can be used when you want to differentiate on debug or logging session by assigning different paths and creating several PCANBasic.log files.

Application – Example of Use

Let's say you have started several instances of the same program and you want to debug all of them at the same time. Additionally you want to separate the log files per application. You could create a folder for each and configure the path on each application, so that each of them can create its own log file:

```
Do In each Application
{
     Set the value PCAN_LOG_LOCATION to NEW_PATH.
     If "Path was successfully set" Then
     {
          Show Logging is active. Path for Log is
          Print NEW_PATH
          Do needed operations
     }
     Else
          Show Error: Log's Path couldn't be configured.
}
```

PCAN_LOG_STATUS

This parameter helps the user to control the activity status of a debug session within the PCAN-Basic API.

Note that if the logging status is set to ON without having configured a destination path for the log file or without having configured the information to be logged, then the session process will start with the default values, that is the log file will be placed in the folder where the calling process is located and only exceptions will be logged.

Availability

It is available since version 1.0.0.

Supported By

PCAN-NONBUS: Logging parameters are used globally, i.e. they are not tied to a specific PCAN-Channel, but to a specific process.

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The Logging is OFF.
PCAN_PARAMETER_ON	The Logging is ON.

Default Value

The default value of the Logging mode is deactivated (PCAN_PARAMETER_OFF). After activating it, the logging functionality stays active until it is expressly deactivated.

Initialization Status

Not apply. It is not needed to have any PCAN-Channel initialized in order to use this parameter.

When to Use

It can be used to interrupt debug sessions (start, stop, restart, etc).

Application – Example of Use

Let's say you want to debug your application. You already noted that you have an intermittent problem. In order to get only logged data that potentially contains information about the issue being investigated, you could activate the debug session only in those moments when the anomaly takes place:

```
Function ActivateLogging
{
      Set the value PCAN_LOG_STATUS to ON.
      If "Log was activated" Then
            Show Logging is active
      Else
            Show Error: Log's Path couldn't be activated.
}
Function DeactivateLogging
{
      Set the value PCAN_LOG_STATUS to OFF.
      If "Log was deactivated" Then
            Show Logging is now inactive
      Else
            Show Error: Log's Path couldn't be deactivated.
}
```

PCAN_LOG_CONFIGURE

This value is used to configure the debug information to be included in the log file generated in a debug session within the PCAN-Basic API.

Availability

It is available since version 1.0.0.

Supported By

PCAN-NONBUS: Logging parameters are used globally, i.e. they are not tied to a specific PCAN-Channel.

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be configured with one of the following values or a combination of those:

Defined Value	Description
LOG_FUNCTION_DEFAULT	This value is always active.
LOG_FUNCTION_ENTRY	Logs when a function is entered.
LOG_FUNCTION_PARAMETERS	Logs the parameters passed to a function.
LOG_FUNCTION_LEAVE	Logs when a function is leaved and its return value.
LOG_FUNCTION_WRITE	Logs the parameters and CAN data passed to the CAN_Write function.
LOG_FUNCTION_READ	Logs the parameters and CAN data received through the CAN_Read function.

Default Value

The default value of this parameter is to log only internal exceptions (LOG_FUNCTION_DEFAULT). **Note** that having only this default value can cause to log no data at all, since the appearance of exceptions are very rare (we do our best to maintain this API bugs free ☺).

Initialization Status

Not apply. It is not needed to have any PCAN-Channel initialized in order to use this parameter.

When to Use

It can be used when only specific debug information is desired.

Application – Example of Use

Let's say you have an application that have a problem with the sequence in which some API functions are called, and you want to know which function is being called too early or too late. You could configure the debug session to only log the calling of the functions, so that you can see the order in which those functions are processed:

```
Set the value PCAN_LOG_CONFIGURE to LOG_FUNCTION_ENTRY.
If "Log was configured" Then
{
     Set the value PCAN_LOG_STATUS to ON.
     If "Log was started" Then
     {
          Do needed operation
          Set the value PCAN_LOG_STATUS to OFF.
          Show Debug is finished. Please check the log file
     }
     Else
          Show Error: Logging couldn't be started.
}
Else
     Show Error: Logging cannot be configured.
```

PCAN_LOG_TEXT

This parameter helps the user to insert custom text into the log file generated in a debug session.

Note that using this parameter starts recording debug information automatically, if the logging functionality was inactive. You could include calls to this parameter in parts of your code that normally shouldn't have to be executed, so that any unwanted behavior triggers the start of a debug session (as a kind of watch dog).

Availability

It is available since version 1.0.0.

Supported By

PCAN-NONBUS: Logging parameters are used globally, i.e. they are not tied to a specific PCAN-Channel.

Access Mode

This parameter can only be written.

Possible Values

This parameter must be a string containing the data to be inserted in the log file. There is no limit for the length of the string but it is recommended to use a length not bigger than MAX_PATH (255 bytes).

Default Value

Not apply. This is a value that can only be written.

Initialization Status

Not apply. It is not needed to have any PCAN-Channel initialized in order to use this parameter.

When to Use

It can be used when you want to use the log functionality for your own purposes, i.e. to debug own processes, behavior, to mark executed code places, etc.

Application – Example of Use

Let's say you are writing an application and want to include debug information of other processes being done inside of it, e.g. to log when any access violation occur, or when the user makes any configuration changes, etc. Instead of implementing your own debug logging, you could use this parameter and so save implementation time, since this logging file works, has been tested already, and it include already information as, when an entry was done, and from which thread it was done:

```
Function FunctionLogger(Message_to_log)
{
      Set the value PCAN_LOG_TEXT to Message_to_log.
      If "Log couldn't be written" Then
            Show Error: Log couldn't be written.
}

..... Any_Part_Of_The_Application
{
      Do "Some Operation"
      If "Some Operation was executed well" Then
            FunctionLogger( "MyAPP: "Some Operation" OK"
      Else
            FunctionLogger( "MyAPP: "Some Operation" FAILURE"
}
```

Using Tracing Parameters

These parameters are intended to minimize the developing time and cost of CAN applications using the PCAN-Basic API, by allowing the recording and storing of all CAN communication in an ASCII formatted file that can be loaded by any text editor. Thanks to the structured stored data, it can be easily parsed into own applications too (see Appendix B).

Since the trace format is officially used by several Peak-System applications, there are already several tools that are able to load and process those traces files, minimizing so the investment in own software programming. For example, the information recorded can be inspected using PCAN-Explorer, and can even be played back for simulation purposes using the PCAN-Trace application.

Consider that the trace functionality is available for each PCAN-Channel. This implies three important points:

- The PCAN-Channel must be first initialized before a trace session can be started.
- You can start as many trace sessions as used/initialized PCAN-Channels within your application, simultaneously.
- The data traced corresponds to the data successfully transmitted through a PCAN-Channel, using the functions CAN_Read and CAN_Write. **Note** that if an application never calls those functions then no data will be traced.

PCAN_TRACE_LOCATION

This value is used to set the folder on a computer in where the PCAN-Trace file will be stored. If a session is running (a PCAN-Trace file is being written), PCAN_TRACE_LOCATION instructs the API to close the current PCAN-Trace file and to start the process again with the new folder information.

Note that the name of the trace file cannot be freely specified. The base name of the trace file is always the name of the PCAN-Channel being used (**PCAN_USBBUS1.trc**, for example). It is only possible to enhance the name with the date and/or time of creation of the file.

Availability

It is available since version 1.3.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This value is a string containing a fully-qualified and valid directory path on the executing computer. In order to use the default path (calling process path) an empty string must be set.

Kind of Path	Value needed
CUSTOM Path	A valid directory string (Files and Paths).
DEFAULT Path	Empty string (calling process folder).

Default Value

The default value is the path to the calling process folder.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when you want to sort trace sessions being done.

Application – Example of Use

Let's say you have an application that operates in different modes (flashing, diagnostic, custom, user, etc). You could have a folder for each mode, so that trace files are automatically sorted by the application's mode used:

```
Function SetTracingPath(Current_App_Mode)
{
     According to Current_App_Mode
          Mode1: Mark DirectoryName: Mode1Dir
          Mode2: Mark DirectoryName: Mode2Dir
          ModeN: Mark DirectoryName: ModeNDir

     Create DirectoryName with CreateDirectory
     Set the value PCAN_TRACE_LOCATION to DirectoryName
     If "Trace location was set" Then
     {
          Show Trace location sucessfully set to
          Print DirectoryName
     }
     Else
          Show Error: Trace location couldn't be changed.
}
```

PCAN_TRACE_STATUS

This parameter helps the user to control the activity status of a trace session within the PCAN-Basic API.

Note that if the tracing status is set to ON without having configured a destination path for the trace file or without having configured the tracing mode, then the session process will start with the default values, that is:

- The PCAN-Trace file will be placed in the folder where the calling process is located.
- The file name to use is the name of the used PCAN-Channel (**PCAN_USBBUS1.trc**, for example).
- Existent files will not be overwritten, i.e. starting the trace process will fail.
- The API will create one PCAN-Trace file, and will fill it with data until the file reaches a size of **10 megabytes**.

Availability

It is available since version 1.3.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be activated or deactivated.

Defined Value	Description
PCAN_PARAMETER_OFF	The Tracing is OFF.
PCAN_PARAMETER_ON	The Tracing is ON.

Default Value

The default value of the Tracing mode is deactivated (PCAN_PARAMETER_OFF). After activating it, the tracing functionality stays active until one of these possibilities happens:

- The tracing session is expressly deactivated.
- The used PCAN-Channel is disconnected (e.g. using the function CAN_Uninitialize).
- The configuration of the tracing session instructs the API to stop tracing (e.g. the maximum size for a trace file is reached).

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used to control a tracing sessions (start, stop, restart, etc).

Application – Example of Use

Let's say you want to allow the user of your application to decide when data should be traced. You could allow this by simply invoking this parameter through a function that a user could reaches using a button click:

```
Function ActivateTracing()
{
     Set the value PCAN_TRACE_STATUS to ON
     If "Trace status was actIvated" Then
          Show Trace session started sucessfully
     Else
          Show Error: Couldn't start a trace session
}

Function DeactivateTracing()
{
     Set the value PCAN_TRACE_STATUS to OFF
     If "Trace status was deactivated" Then
          Show Trace session finished
     Else
          Show Error: Couldn't stop the trace session
}
```

PCAN_TRACE_SIZE

This parameter is used to set the maximum size in megabytes that a single PCAN-Trace file can have. **Note** that trying to set the size for a file will fail, if a tracing session is active.

Availability

It is available since version 1.3.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This value is an integer representing the amount of megabytes a file can store. In order to use the default size (10 megabytes) the value of 0 must be set.

Kind of Size	Valid Value
CUSTOM Size	A value between 1 and 100 megabytes.
DEFAULT Size	A value of 0 (defaults to 10 megabytes).

Default Value

The default size value is 10 Megabytes. This allows to record about 166.000~ CAN messages (Standard frames, with 8 data bytes).

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used to control the amount of data to be stored in a single file. According with the tracing configuration, this parameter can be used to automatically stop a trace session (e.g. to record data until a given limit is reached).

Application – Example of Use

Let's say you want to allow the user of your application to decide how big should be a trace. You could allow this by simply invoking this parameter through a function that a user could reaches using a button-click:

```
Function SetMaximumTraceSize(Size_To_Set)
{
      Set the value PCAN_TRACE_SIZE to Size_To_Set
      If "Trace Size was set" Then
      {
            Show Trace size set to
            Print Size_To_Set
      }
      Else
            Show Error: Couldn't configure the size for the trace file
}
Function SetdefaultTraceSize()
{
      Set the value PCAN_TRACE_SIZE to 0
      If "Trace Size was set" Then
            Show The default Trace file size was sucessfully set
      Else
            Show Error: Couldn't set the default size for the trace file
}
```


PCAN_TRACE_CONFIGURE

This parameter is used to configure the trace process and the file generated in a trace session. **Note** that trying to configure the trace process will fail, if a tracing session is active.

Availability

It is available since version 1.3.0.

Supported By

PCAN-ISA (Channels PCAN_ISABUS1 to PCAN_ISABUS8).
PCAN-DNG (Channel PCAN_DNGBUS1).
PCAN-PCI (Channels PCAN_PCIBUS1 to PCANPCIBUS8).
PCAN-USB (Channels PCAN_USBBUS1 to PCAN_USBBUS8).
PCAN-PCC (Channels PCAN_PCCBUS1 to PCAN_PCCBUS2).

Access Mode

This parameter is read/write. It can be set and read.

Possible Values

This parameter can be configured with one of the following values or a combination of those:

Defined Value	Description
TRACE_FILE_SINGLE	A trace session is stored in a single and stays active until the file reaches the maximum configured file size, or it is deactivated, or the PCAN-Channel used is disconnected.
TRACE_FILE_SEGMENTED	A trace session is stored in several files. A new file is created when a previous file reaches the maximum configured size. The tracing session stays active until it is deactivated, or the PCAN-Channel used is disconnected.
TRACE_FILE_DATE	The name of the trace file also includes the start-date of the tracing session. The date is expressed using 8 digits with the form **YYYYMMDD**, where YYYY are four digits for the year, MM two digits for the month, and DD two digits for the day, e.g. "**20130228**_PCAN_USBBUS1.trc" for the 28th February 2013. If both, TRACE_FILE_DATE and TRACE_FILE_TIME are configured, the file name starts always with the date: "**20130228**140733_PCAN_USBBUS1_1.trc".
TRACE_FILE_TIME	The name of the trace file also includes the start-time of the tracing session. The time is expressed using 6 digits with the

	form **HHMMSS**, where HH are two digits for the hour in 24 hours format, MM two digits for the minutes, and SS two digits for the seconds, e.g. "**140733**_PCAN_USBBUS1.trc" for the 14:07:33 (02:07:33 PM). If both, TRACE_FILE_DATE and TRACE_FILE_TIME are configured, the file name starts always with the date: "20130228**140733**_PCAN_USBBUS1_1.trc".
TRACE_FILE_OVERWRITE	It causes the overwriting of a existence trace file when a new trace session is started. If this value is not configured, trying to start a tracing process will fail, if the file name to generate is the same as one used by an existing file.

Default Value

The default value of this parameter is TRACE_FILE_SINGLE, which means a single file is created and filled out until the maximum configured file size is reached.

Note that the name of the file to use is the name of the PCAN-Channel being traced (e.g. PCAN_USBBUS1.trc). If a file with the same name already exists, then the activation of the tracing session will fail.

Initialization Status

The PCAN-Channel has to be initialized before using this parameter.

When to Use

It can be used when the trace behavior desired is other than the default.

Application – Example of Use

Let's say you want to trace CAN data but you don't know how many bytes you will trace, or you know that the trace information will be more than the maximum file size allowed (100 megabytes). You could configure the trace process to use several files (segmentation) so that the only limit is the storing unit used. In this way the application stays tracing data in different files until you stop the process or an error on file creation occurs:

```
Set the value PCAN_TRACE_SIZE to 20
If "Trace Size was set" Then
{
     Mark TraceConfig: TRACE_FILE_SEGMENTED Or TRACE_FILE_OVERWRITE
     Set the value PCAN_TRACE_CONFIGURE to TraceConfig
     If "Trace was configured" Then
     {
          Set the value PCAN_TRACE_STATUS to ON
          If "Trace status was activated" Then
               Show Trace configured and started sucessfully.
          Else
               Show Error: Couldn't start a trace session
     }
     Else
          Show Error: Couldn't configure the size for the trace file
}
Else
     Show Error: Couldn't configure the size for the trace file
```

Appendix A: Debug-log over Registry

These steps will guide you activating/deactivating the Logging functionality of PCAN-Basic using the registry of Windows.

Activating a Log Session

1. Stop all applications using the PCAN-Basic.
2. Open the Windows's Registry (e.g. using the Windows Start menu / "Execute…" and typing "**regedit**").
3. Create the following registry key under the [**HKEY_CURRENT_USER**] hive:
   \Software\PEAK-System\PCAN-Basic\Log
4. To specify the data to be logged, add a new **DWORD** value to the key created before, and call it "**Flags**".
5. Sets the value for "**Flags**" according to your needs. This value is the numerical value of any LOG_FUNCTION_* define or a logic-OR combination of them.
6. To specify the directory where the log file should be created, add a new **STRING** value to the key created before, and call it "**Path**".
7. Sets the value for "**Path**" with the full path to the directory you want.

At this point, starting any application that use the PCAN-Basic API will cause the automatic generation of a debug session.

Deactivating a Log Session

1. Stop all applications using the PCAN-Basic.
2. Open the Windows's Registry (e.g. using the Windows Start menu / "Execute…" and typing "**regedit**").
3. Locate the registry hive [**HKEY_CURRENT_USER**].
4. Search for the following registry key:
   \Software\PEAK-System\PCAN-Basic\Log
5. Delete the key and its values.

At this point, starting any application that use the PCAN-Basic will not cause logging operations anymore.

VERY IMPORTANT NOTE

Please don't forget to delete the created key after your debug session is done. If you leave the key, all PCAN-Basic applications running under your Windows account will remain writing data to their log files, generating in this way huge text files that consume hard-disk space unnecessarily.

Appendix B: PCAN-Trace Format 1.1

The PCAN-Basic API uses the PCAN-Trace format 1.1 which is used by PCAN-Explorer 3.0.2, PCAN-Explorer 4, PCAN-Trace 1.5, PCAN-View 3, and the Peak-Converter.

Example

```
;$FILEVERSION=1.1
;$STARTTIME=37704.5364870833
;
;   C:\TraceFile.trc
;
;   Start time: 24.03.2003 12:52:32.484
;   PCAN-Net: TestNet
;
;   Columns description:
;   ~~~~~~~~~~~~~~~~~~~~
;   +-Message Number
;   |          +Time Offset (ms)
;   |          |         +Type
;   |          |         |          +ID (hex)
;   |          |         |          |     +Data Length Code
;   |          |         |          |     |   +Data Bytes (hex) ...
;   |          |         |          |     |   |
;---+--   -----+----  --+--  -----+---  +   -+ -- -- -- -- -- -- --
     1)       1059.9  Rx          0300  7   00 00 00 00 04 00 00
     2)       1283.2  Rx          0300  7   00 00 00 00 04 00 00
     3)       1298.9  Tx          0400  2   00 00
     4)       1323.0  Rx          0300  7   00 00 00 00 06 00 00
     5)       1346.8  Warng   FFFFFFFF  4   00 00 00 04  BUSLIGHT
     6)       1349.2  Error       0008  4   00 19 08 08
```

Description

File Coding:

The Trace file is ASCII coded.

Comment Lines:

Lines prefixed with a Semicolon are “Comments” and are ignored while loading Trace files, except for $-Keywords.

$-Keywords:

These are defined information that gives different information about the Trace file. They appear as a comment line. Possible keywords are:

- $FILEVERSION: contains the major and minor version of the file format, i.e. “1.1” for this version.
- $STARTIME: contains the absolute start time of the trace file:
  - Format: Floating point, point as decimal separator.
  - Value: the integral part represents the number of days that have passed since 30th December of 1899. The fractional part, the fraction of a 24 hour day that has elapsed, resolution is 1 millisecond.

Columns:

The information contained in a Trace file is accommodated within 5 columns:

- Message Number: Index of a recorded message (ignored while loading the trace file).
- Time Offset (ms): Time offset since start of the trace session. The time has a resolution of 1/10 milliseconds.

  - Format: Floating point, point as decimal separator.
  - Value: the integral part represents the milliseconds offset. The fractional part is 1/10 milliseconds (1 digit).
- Type: Represents the kind of message recorded. Possible message types are:
  - “Rx”: Message was received (in PCAN-Basic, using the function CAN_Read).
  - “Tx”: Message was sent (in PCAN-Basic, using the function CAN_Write).
  - “Warng”: Message represents a received Warning-Frame.
  - “Error”: Message represents an Error-Frame (**not supported** by PCAN-Basic).
- ID (hex): Represents the CAN-ID in hexadecimal notation. Possible values are:
  - 4 digits for 11-bit CAN-IDs (0000-07FF).
  - 8 digits for 29-bit CAN-IDs (00000000-1FFFFFFF).
  - Special case: “FFFFFFFF” for Warning-Frames.
- Data Length Code: It is a number between 0-8 representing the amount of data contained within the message recorded.
- Data Bytes (hex): represents the data of a recorded message. According with the message type, the data can be:
  - If the message represents common CAN data: so many data bytes, in hexadecimal notation, as the Data Length Code indicates.
  - If the message represents a remote request frame: “RTR”
  - If the message represents a Warning-Frame: 4 data bytes expressed in hexadecimal notation, using Motorola format. At the end of this line, the short name of the Warning (ignored while loading the Trace file). Example: “00 00 00 04 BUSLIGHT”.
  - If the message represents an Error-Frame: 4 data bytes expressed in hexadecimal notation. Error-Frames are not supported by PCAN-Basic.