CANnes

PC CAN Interface

Manual

Version: 1.00
December 19th, 2001

Deelbögenkamp 4c,
D - 22 297 Hamburg, Germany
Phone +49-40-51 48 06 - 0,
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

Version

Version	Date	Author	Comment
0.03	11-Dec-2001	OK	First version (Preliminary)
1.00	19-Dec-2001	OK	First release version

Contents

Deelbögenkamp 4c,
D - 22 297 Hamburg, Germany
Phone +49-40-51 48 06 - 0,
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

1 Introduction

The CANnes card is a PCI card which provides a simple, easy to use CAN interface. It is equipped with a male and a female 9 pin Sub-D connector with standard CAN pin assignments. CAN bit rates between 10 and 1000 kBit/s (1 MBit/s) are possible.
The industry standard CAN controller SJA1000 is used together with the CAN bus driver 82C250 and optical isolation between the CAN controller and the CAN bus driver. A 120 ohms termination resistor is also provided. This resistor can be switched on and off either by a software controlled relais or by a jumper.
Device drivers and function libraries are provided for the Windows (98/ME/2000 and NT4) operating systems and also for the Linux operating system.
The CANnes cards is CE-approved and also meets FCC specifications. It can be used with every PC which is equipped with PCI slots fulfilling at least the PCI 2.1 standard.

2 Technical Data

2.1 General data

- PCI standard: V2.1
- CAN controller: SJA1000
- CAN bus driver: 82C250
  - CANH/CANL input/output voltage: -8V..+18V
  - Differential bus voltage: 1.5V..3.0V
  - An unpowered CANnes card will not disturb the CAN bus lines.
- Termination resistor: 120 Ohms, selectable by software or by hardware.

2.2 CAN connector pin assignments

The pins of the male and the female Sub-D connectors are connected as follows:

Pin number	Signal
1	Not connected
2	CAN Low (CANL)
3	Ground
4	Not connected
5	Shield
6	Ground
7	CAN High (CANH)
8	Not connected
9	Not connected

Both connectors are electrically identical.

Deelbögenkamp 4c,
D - 22 297 Hamburg, Germany
Phone +49-40-51 48 06 - 0,
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

2.3 CAN bus termination

The CANnes card is equipped with a 120 ohms resistor to terminate the CAN bus. This termination resistor can be switched on and off by a relais which is controlled by software.
The termination can also be switched on manually by hardware. Therefore, a jumper has to be put on the plug beneath the termination relais. The termination resistor is then always switched on and the software control function has no effect. To make the software termination control take effect again the jumper must be removed. Please see also Figure 2.1 to locate the termination jumper.

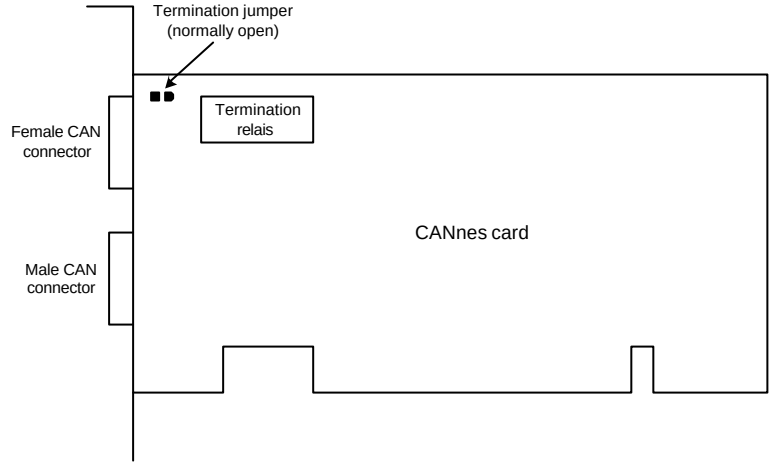

Figure 2.1: Locating the termination jumper

3 Installation

3.1 Hardware installation

To install the CANnes card into a PC, at least one free PCI slot is needed. First make sure that the PC is turned off. Then, open the PC and put the CANnes card into a free PCI slot. Make sure that all contacts fit correctly. After that you can close your PC and turn it on again.

3.2 Software installation

The software is shipped with the CANnes card on diskettes or a CD. The installation of the software depends on the operating system you use.

3.2.1 Windows 98, Windows ME and Windows 2000

A WDM (Windows Driver Model) driver is provided for use with these operating systems. All these operating systems support the Windows Driver Model (WDM) and a plug-and-play installation. After booting the first time with the CANnes card, the "Add new hardware" wizard will appear (you will first have to log in as administrator when using Windows 2000). Then, just insert the installation diskette or CD into the drive and select the "Have disk" button. After that, a file selection dialogue will appear. From there, select the "CannesWDM.inf" file. The device driver will then be installed automatically. It is recommended to reboot the PC after the driver installation has finished. After that the CANnes card can be used.

3.2.2 Windows NT4

As Windows NT4 neither supports the plug-and-play nor the WDM standard, a special driver with an extra installation program is provided for this operating system. After you have logged in as administrator, just start the program "DrvInstNT.exe" from the installation disk by double clicking on its icon. After a short while a dialogue will appear. Then just click the button "Install the CANnes driver". When the installation has finished, you will be prompted for rebooting the computer. It is recommended to do that. The CANnes card can be used then.

3.2.3 Linux

The device driver for Linux is provided open source. To install it, just copy the archive CannesLinux.tar from the installation disk to your hard disk. Then, unpack it using "tar". After that, you can compile the driver. Just change to the directory "module" and run "make". The driver will be compiled for kernel 2.4 then, generating the module file "cannes_dev.o". If you are using kernel 2.2, edit the Makefile and remove the switch "-DKERNEL_24" from the "CFLAGS" line before compiling the driver.
The driver can be loaded by executing the shell script "cannes_load" which is also provided in the "module" directory. It must be executed by a root user. The script loads the driver and creates the necessary nodes in the "/dev" directory (can1, can2 etc.). You can install the script so that it will be executed automatically at boot time. To achieve this, copy the script and the driver to the appropriate directories. This depends a bit on the Linux distribution you use.
The programming interface is provided by a library named "Cannes.o" which can be found in the "library" directory. To use it, just link it to your application. Please see chapter 5 for a detailed description of the library.

4 The CAN Monitor “CANnesMon”

The program “CANnesMon” is an easy to use CAN monitor running on Windows 98/ME/2000 and NT4. This program is mainly provided for testing purposes and to try out other CAN devices attached to the CANnes card. Its installation is simple: Just copy the file “CANnesMon.exe” from the installation disc to any directory of the hard disk. Then, the program can be started from the hard disk by simply double clicking the file “CANnesMon.exe”. It will then display its main window with three pages: The “Setup” page, the “Send/Receive” page and the “Macro” page. These are described in detail in the following sections.

4.1 The “Setup” page

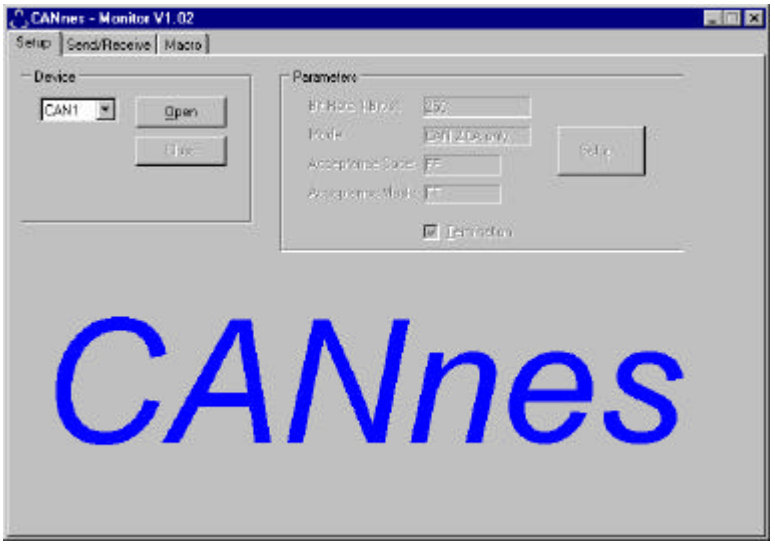

Figure 4.1: The "Setup" page

On this page (Figure 4.1), the desired CANnes card can be selected and basic setup can be done. First, select the device (which mostly will be CAN1) in the “Device” section. Then, click the “Open” button which will open the device. After that, the currently set parameters are displayed in the “Parameters” section. The “Termination” check box controls the termination resistor: If this box is checked the 120 ohms termination resistor will be switched on.
Click the “Setup” button to change the other parameters. The setup dialogue (Figure 4.2) will then appear.

Figure 4.2: The setup dialogue

Here, the following parameters can be set:

- Bit rate: This value is given in kBit/s. Possible values are 10, 20, 50, 100, 125, 250, 500, 800 and 1000 kBits (1000 kBit/s = 1 MBit/s).
- Mode:
  - CAN 2.0A: only standard frames (with 11 bit identifiers) can be sent and received.
  - CAN 2.0A/2.0B: standard frames (with 11 bit identifiers) and extended frames (with 29 bit identifiers) can be used.

- Acceptance Code and Acceptance Mask: These hexadecimal values form a filter for incoming CAN messages (see section 5.13 for further explanation). In CAN 2.0A mode, set both values to FF to get all incoming CAN messages. In CAN 2.0A/2.0B mode, set both values to FFFFFFFF to get all incoming CAN messages.

When you are satisfied with your settings click the "OK" button to make all changes take effect.

4.2 The "Send/Receive" page

This page provides the main functionality of the program. Here, CAN messages can be sent and incoming CAN messages can be seen.

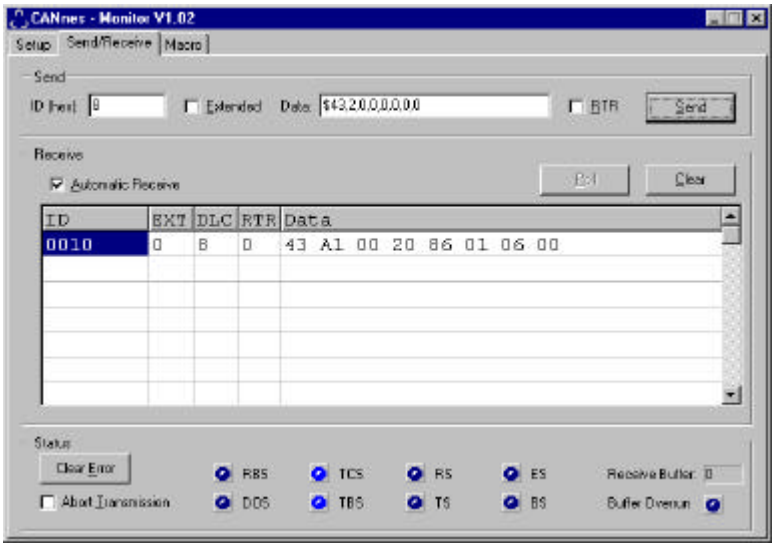

Figure 4.3: The "Send/Receive" page

In the "Send" section, CAN messages can be entered and sent out. To do this, enter the following things:

- ID: enter any hexadecimal number as the message identifier.
- Extended: check this box if the frame is to be sent as an extended frame. This is only possible when CAN 2.0B mode (s. 4.1) is active.
- Data: Here you can enter up to eight data bytes. These must be sperated by commas and can be any combination of decimal and hexadecimal numbers. Hexadecimal numbers must be preceeded by a "$" sign. Example: $43,2,0,0,0,1,$FF,5.
- RTR: Check this box to set the RTR (Remote Transmission Request) flag of the message.

To send out the data you have entered, just click the "Send" button. The message frame will then be transmitted. An error message will be displayed if the message frame could not be transmitted.

In the "Receive" section, incoming messages are displayed. If the "Automatic Receive" check box is checked, every incoming frame will be displayed automatically. If that box is not checked, the "Poll" button must be clicked for every incoming message to be displayed. This can be useful to watch the status flags when a message frame comes in and to see how the receive buffer is being filled.
Clicking the "Clear" button will clear the received message display. In the received message display the following data is displayed:

- ID: The identifier of the message as a hexadecimal value.
- EXT: 1 if the message is an extended frame, 0 if it is a standard frame.
- DLC: Data length code: Number of data bytes in this message.
- RTR: 1 if the RTR flag of the message is set, 0 if not.
- Data: The data bytes of the message, shown as hexadecimal values.

In the "Status" section the status flags of the CAN controller are shown. Clicking the "Clear Error" button clears the error flags. If "Abort Transmission" is checked, any outstanding message transmission will be aborted when the "Clear

Error" button is clicked. The "Receive Buffer" display on the right side of the window shows how many CAN messages are in the receive buffer of the device driver, and the "Buffer Overrun" LED shows if that buffer has run out of space.

The meanings of the status flags are as follows:

- RBS: Receive Buffer Status. When on: At least one message is available in the hardware receive buffer.
- DOS: Data Overrun Status. When on: Data was lost because of no more space in the hardware receive buffer.
- TCS: Transmission Complete Status. When on: The last message was transmitted successfully.
- TBS: Transmission Buffer Status. When on: The transmit buffer is empty and a new message can be transmitted.
- RS: Receive Status. When on: A message is just being received.
- TS: Transmit Status. When on: A message is just being transmitted.
- ES: Error Status. When on: Too many errors have occurred.
- BS: Bus Status. When on: The CAN controller is currently **not** involved in CAN bus activities.

4.3 The "Macro" page

On this page (Figure 4.4) you can define CAN messages that will be transmitted just by clicking one button. Up to 44 macros can be defined. All macros can be saved to a file and loaded again later.

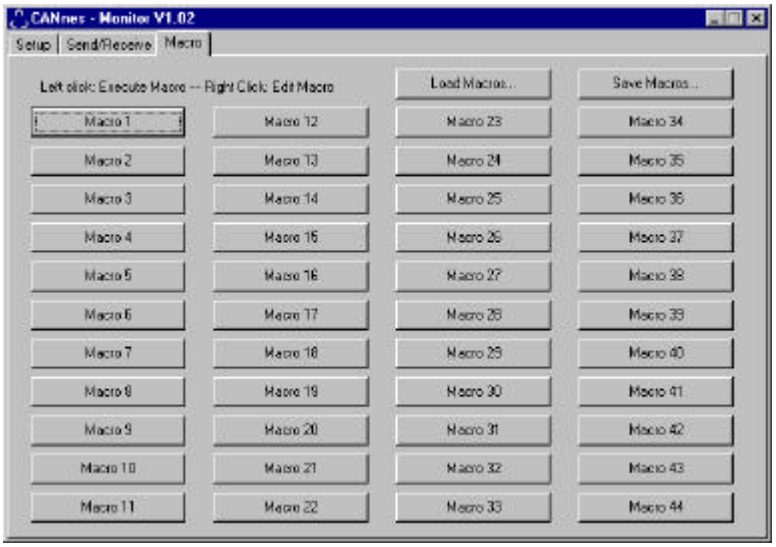

Figure 4.4: The "Macro" page

If you click a macro button with the left mouse key the pre-defined message will be transmitted. To change a macro just click the corresponding button with the right mouse key. Then, the "Edit Macro" dialogue (Figure 4.5) will be shown.

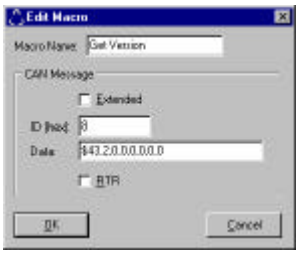

Figure 4.5: The "Edit Macro" dialogue

Here you can set the following things:

- Macro Name: This text will appear on the macro button.
- Extended: Check this box if this message is to be sent as an extended message frame (29 bit identifier).
- ID: The message identifier as a hexadecimal number.

- Data: The data bytes of this message. These bytes can be entered in the same way as in the "Send" section of the "Send/Receive" page (s. 4.2).
- RTR: Check this box to set the RTR (Remote Transmission Request) flag of this message.

The changes will take effect after clicking the "OK" button.
Click the "Save Macros..." button to save your macros to a file. To load a previously saved macro file just click the "Load Macros..." button.

Deelbögenkamp 4c,
D - 22 297 Hamburg, Germany
Phone +49-40-51 48 06 - 0,
FAX: +49-40-51 48 06 - 60
http://www.trinamic.com

5 The Programming Interface

For all supported Windows operating systems a DLL named "Cannes.dll" (which is copied to the Windows system directory automatically during installation (s. 3.2)) and for the Linux operating system a library named "Cannes.o" is provided. The functions of these libraries are the same for all supported operating systems. For ease of use, C/C++ header files and a Delphi unit are also provided on the installation disc.
The functions are described in detail in the following sections. C syntax is used for the function definitions.

5.1 Open the interface

```
BOOLEAN __stdcall CannesOpen(UCHAR DeviceNumber)
```

Parameter:

- DeviceNumber: Number of the device to be opened (1 for CAN1, 2 for CAN2 etc.).

Return value: TRUE if successful.

This function opens a CANnes card, which always has o be done before any other CANnes functions can be used. When only one CANnes card is installed the device number is always 1. If more than one card is installed, 1 stands for the first card (CAN1), 2 stands for the second card (CAN2) and so forth.

5.2 Close the interface

```
void __stdcall CannesClose(UCHAR DeviceNumber)
```

Parameter:

- DeviceNumber: Number of the device to be closed (s. 5.1).

This function closes a CANnes card with the given device number. This should be done before the application ends. After a CANnes device has been closed, it can not be accessed until it is opened again.

5.3 Set up the CAN parameters

```
BOOLEAN __stdcall CannesSetBaudrate(UCHAR DeviceNumber, UINT Baudrate,
                                    ULONG AcceptanceCode, ULONG AcceptanceMask,
                                    BOOLEAN CanMode)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Baudrate: The CAN bit rate given in kBit/s; supported values are 10, 20, 50, 100, 125, 250, 500, 800 and 1000 (1000 kBit/s = 1 MBit/s).
- AcceptanceCode: The acceptance code of the acceptance filter (see section 5.13).
- AcceptanceMask: The acceptance mask of the acceptance filter (see section 5.13).
- CanMode: FALSE: Only CAN 2.0A mode; TRUE: CAN 2.0A and 2.0B mode.

Return value: TRUE if successful.

This function sets all CAN bus parameters. This should always be done right after opening a CANnes card.

5.4 Set the termination resistor

```
BOOLEAN __stdcall CannesSetTermination(UCHAR DeviceNumber, BOOLEAN Flag);
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Flag: TRUE: Switch on the termination resistor; FALSE: Switch off the termination resistor.

Return value: TRUE if successful.

This function switches the 120 ohms termination resistor on an off. This only works if the termination jumper is open!

5.5 Send a CAN message

```
BOOLEAN __stdcall CannesSendMessage(UCHAR DeviceNumber, BOOLEAN Extended,
                        ULONG Identifier, BOOLEAN Rtr, UCHAR *Data, UCHAR Length)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Extended: TRUE = send an extended frame (29 bit identifier); FALSE = send a standard frame (11 bit identifier).
- Identifier: Message identifier.
- Rtr: TRUE: Set the RTR flag of the message.
- Data: Pointer to the message data (an array of up to eight bytes).
- Length: Number of bytes to send (0..8).

Return value: TRUE if successful.

This function sends a CAN message on the CAN bus. Please note that extended frames can only be sent when this has been configured in the CannesSetBaudrate() function (s. 5.3). The function fails if there is just another message being transmitted by this device.

5.6 Read a CAN message

```
BOOLEAN __stdcall CannesGetMessage(UCHAR DeviceNumber, BOOLEAN *Extended,
                        ULONG *Identifier, BOOLEAN *Rtr, UCHAR *Data, UCHAR *Length)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Extended: Pointer to BOOLEAN variable to hold the extended flag of the received message.
- Identifier: Pointer to ULONG variable to hold the identifier of the received message.
- Rtr: Pointer to variable to hold the RTR flag of the received message.
- Data: Pointer to array for the data bytes. This array must be long enough to hold eight data bytes.
- Length: Pointer to ULONG variable to hold the length of the received message.

Return value: TRUE if successful.

This function reads a CAN message from the receive buffer. The function fails if there is no message available.

5.7 Read the input queue length

```
BOOLEAN __stdcall CannesGetInQueueCount(UCHAR DeviceNumber, WORD *InQueueCount)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- InQueueCount: Pointer to WORD (USHORT) variable to hold the number of CAN messages in the input queue (bit 0..14: number of messages, bit 15: overrun flag).

Return value: TRUE if successful.

The CANnes device driver has a buffer for incoming CAN messages. This function reads the number of messages which are currently in the buffer. The most significant bit of this 16-bit value is the overrun flag. If this bit is set an overrun of the receive buffer has occurred. The overrun flag can be cleared by reading out a CAN message (s. 5.6).

5.8 Read the CAN controller status

```
BOOLEAN __stdcall CannesGetStatus(UCHAR DeviceNumber, UCHAR *Status)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Status: Pointer to UCHAR variable to hold the status code.

Return value: TRUE if successful.

This function reads the status register of the CAN controller. The bits of this value are assigned as follows:

Bit	Name	Description
0	RBS	Receive Buffer Status: 1 = at least one message is available in the hardware receive buffer.
1	DOS	Data Overrun Status: 1 = data was lost because of no more space in the hardware receive buffer.
2	TBS	Transmission Buffer Status: 1 = the transmit buffer is empty and a new message can be transmitted.
3	TCS	Transmission Complete Status: 1 = the last message was transmitted successfully.
4	RS	Receive Status: 1 = a message is just being received.
5	TS	Transmit Status: 1 = a message is just being transmitted.
6	ES	Error Status: 1 = too many errors have occurred.
7	BS	Bus Status: 1 = the CAN controller is currently **not** involved in CAN bus activities.

Please note the following: As incoming messages are automatically put into an internal buffer by the device driver, the RBS flag will also be cleared automatically right after a message has been completely received. So, this flag can not be used to detect if a message has been received. Instead, use the function CannesGetInQueueCount() (s. 5.7) to see if there are CAN messages available or just try reading a CAN message using CannesGetMessage() (s. 5.6) and see if the function succeeds or fails.

5.9 Clear errors

```
BOOLEAN __stdcall CannesClearError(UCHAR DeviceNumber, BOOLEAN Abort)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Abort: If set to TRUE, any outstanding CAN transmission will be aborted.

Return value: TRUE if successful.

This function resets the error flags and error counters of the CAN controller. It can also abort any outstanding transmissions in case of errors.

5.10 Reset the CANnes card

```
BOOLEAN __stdcall CannesResetCard(UCHAR DeviceNumber)
```

Parameter:

- DeviceNumber: Number of the device (s. 5.1).

Return value: TRUE if successful.

This function issues a hardware reset of the CANnes card. After that, all parameters must be set up again using the appropriate functions.

5.11 Read a CAN controller register directly

```
BOOLEAN __stdcall CannesReadRegister(UCHAR DeviceNumber, UCHAR Address, UCHAR *Data)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).

- Address: Address of the register to be read (0..255, see SJA1000 manual for the register addresses).
- Data: Pointer to UCHAR variable to hold the data.

Return value: TRUE if successful.

This function directly reads a register of the SJA1000 CAN controller. This is normally not needed. Please see the SJA1000 manual for explanation of the CAN controller registers.

5.12 Write to a CAN controller register directly

```
BOOLEAN __stdcall CannesWriteRegister(UCHAR DeviceNumber, UCHAR Address, UCHAR Data)
```

Parameters:

- DeviceNumber: Number of the device (s. 5.1).
- Address: Address of the register to be written to (0..255, see the SJA1000 manual for the register addresses).
- Data: Value to be written.

This function directly writes to a register of the SJA1000 CAN controller. This is normally not needed. Please see the SJA1000 manual for explanation of the CAN controller registers. The registers 8, 9 and 31 are not accessible by user applications, as writing wrong values to these registers could either lead to malfunctions of the CAN interface or crash the computer.

5.13 Acceptance filtering

Acceptance filtering can be used to automatically check identifiers and data bytes of CAN messages against a filter condition. CAN messages are stored in the receive buffer only when they meet this condition. Acceptance filtering is different in the CAN 2.0A only mode and CAN 2.0A/2.0B mode.

5.13.1 Acceptance filtering in CAN 2.0A mode

In this mode, there is one acceptance mask byte and one acceptance code byte. The most significant bits (bit 11..4) of the CAN message identifier are compared against the acceptance filter. At the bit positions containing '1' in the acceptance mask, any value is allowed in the message ID. At the bit positions containing '0' in the acceptance mask, the value in the message ID must be the same as the value in the acceptance code. Otherwise, the message will not be put into the receive buffer.
Hint: To get all incoming messages, just set both the acceptance mask and the acceptance code to 0xff.

5.13.2 Acceptance filtering in CAN 2.0A/2.0B mode

Acceptance filtering in this mode works similar to CAN 2.0A mode, but now four bytes are used for the acceptance mask and four bytes are used for the acceptance code.
For extended message frames, all 29 bits of the message identifier are compared against the bits 0..28 of the acceptance mask/acceptance code, and the RTR bit is compared against bit 29 of the acceptance mask/acceptance code.
For standard message frames, the following comparisons are done:

- Bit 0..10 of the message ID against bit 21..31 of the acceptance mask/code.
- The RTR bit against bit 20 of the acceptance mask/code.
- The first data byte against bit 8..15 of the acceptance mask/code.
- The second data byte against bit 0..7 of the acceptance mask/code.

The message will be put into the receive buffer only if all values match.
Hint: To get all incoming messages, just set both the acceptance mask and the acceptance code to 0xffffffff.