

PISO-CM200U Series User Manual

Version 1.0.0, Oct. 2016



Service and usage information for
PISO-CM200U-D / PISO-CM200U-T

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright © 2016 by ICP DAS Co., Ltd. All rights are reserved.

Trademarks

Names are used for identification only and may be registered trademarks of their respective companies.

Contact us

If you encounter any problems while operating your device, please feel free to contact us by email at: service@icpdas.com.

Table of Contents

1. Introduction	5
1.1. Specifications	7
1.2. Features	8
1.3. Dimensions	9
1.4. Overview	11
1.4.1. Board Layout	11
1.4.2. Pin Assignments	12
1.4.3. Switch Settings	13
1.4.4. Terminal Resistor Jumper Settings	14
1.4.5. LED Indicators and Operating Modes	15
2. Getting Started	16
2.1. Installing the Windows Driver	16
2.2. Installing the Hardware	20
2.3. Wiring Connections	21
3. Windows API Function Reference	22
3.1. System Information API	22
3.1.1. CM200_GetDllVersion	23
3.1.2. CM200_GetBoardInf	24
3.1.3. CM200_TotalBoard	26
3.1.4. CM200_TotalCM200Board	27
3.1.5. CM200_GetCM200BoardSwitchNo	28
3.1.6. CM200_GetCardPortNum	29
3.1.7. CM200_ActiveBoard	30
3.1.8. CM200_CloseBoard	31
3.1.9. CM200_BoardIsActive	32
3.1.10. CM200_CheckMCUMode	33
3.1.11. CM200_HardwareReset	34
3.1.12. CM200_AdjustDateTime	35
3.2. CAN Bus API	36

3.2.1.	CM200_CANReset	38
3.2.2.	CM200_CANGetStatus	39
3.2.3.	CM200_EnableCAN.....	41
3.2.4.	CM200_DisableCAN	42
3.2.5.	CM200_SetCANConfig.....	43
3.2.6.	CM200_GetCANConfig	46
3.2.7.	CM200_AddCyclicTxMsg	48
3.2.8.	CM200_DeleteCyclicTxMsg	51
3.2.9.	CM200_EnableCyclicTxMsg.....	52
3.2.10.	CM200_DisableCyclicTxMsg	53
3.2.11.	CM200_CheckCyclicTxRestMsg	54
3.2.12.	CM200_IsTxTimeout	56
3.2.13.	CM200_CANInit	57
3.2.14.	CM200_RxMsgCount.....	58
3.2.15.	CM200_ReceiveCANMsg	59
3.2.16.	CM200_SendCANMsg	61
3.2.17.	CM200_ClearSoftBuffer	63
3.2.18.	CM200_ClearTxSoftBuffer	64
3.2.19.	CM200_ClearRxSoftBuffer	65
3.2.20.	CM200_ClearBufferStatus	66
3.3.	Error Code Definitions.....	67
4.	<i>Firmware Upgrade</i>	69
5.	<i>Appendix</i>	72
5.1.	EMI Ferrite Split/Snap-On Core	72

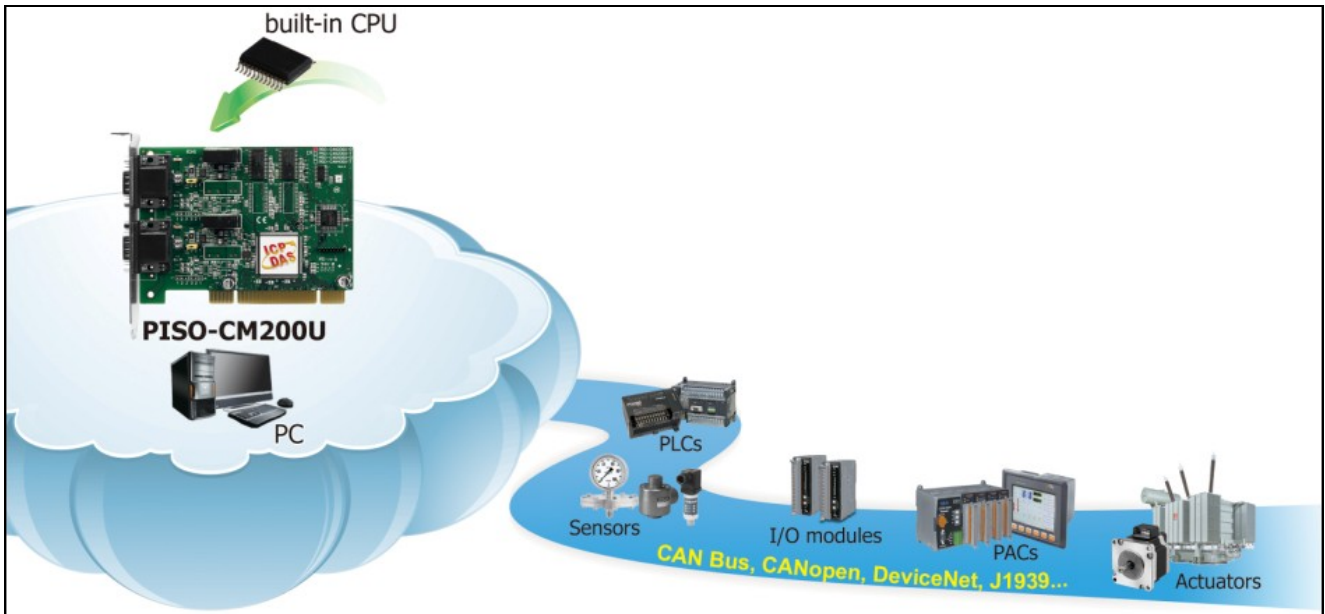
1. Introduction

A Controller Area Network (CAN) is a serial communication system based on the CAN protocol that enables efficient distributed real-time control combined with a very high level of security. The CAN protocol is especially suited for structuring "intelligent" device networks, as well as for building automatic control environments that contain sensors and actuators within a system or sub-system. In a CAN network, there is no addressing of subscribers or stations in the conventional sense, but instead prioritized messages are transmitted.

The PISO-CM200U is a very powerful and economic solution for an active CAN board, containing two CAN channels that cover a wide range of CAN applications. The 32-bit on-board microcontroller allows the filtering, pre-processing, and storage of CAN messages (with a timestamp), as well as real-time transmission of CAN messages, among many other features.

The PISO-CM200U uses Bosch C_CAN controllers and NXP 82C250 Transceivers, which provide bus arbitration and error detection features, combined with auto-correction and re-transmission functionality. And it is equipped with integrated intelligence functions that make it possible to pre-process CAN data streams, thus relieving the Host PC of a considerable burden. As a result, the real-time requirements for applications operating on the Host PC are drastically reduced. As the PISO-CM200U is state-of-the-art, it can be installed in either a 32-bit PCI bus or a Universal PCI bus.

An added benefit is that custom CAN Bus applications can be developed using the PISO-CM200U library. When the PISO-CM200U is active, data exchange between the custom application and the CAN Bus firmware is performed via the memory mapping method on the PISO-CM200U.



1.1. Specifications

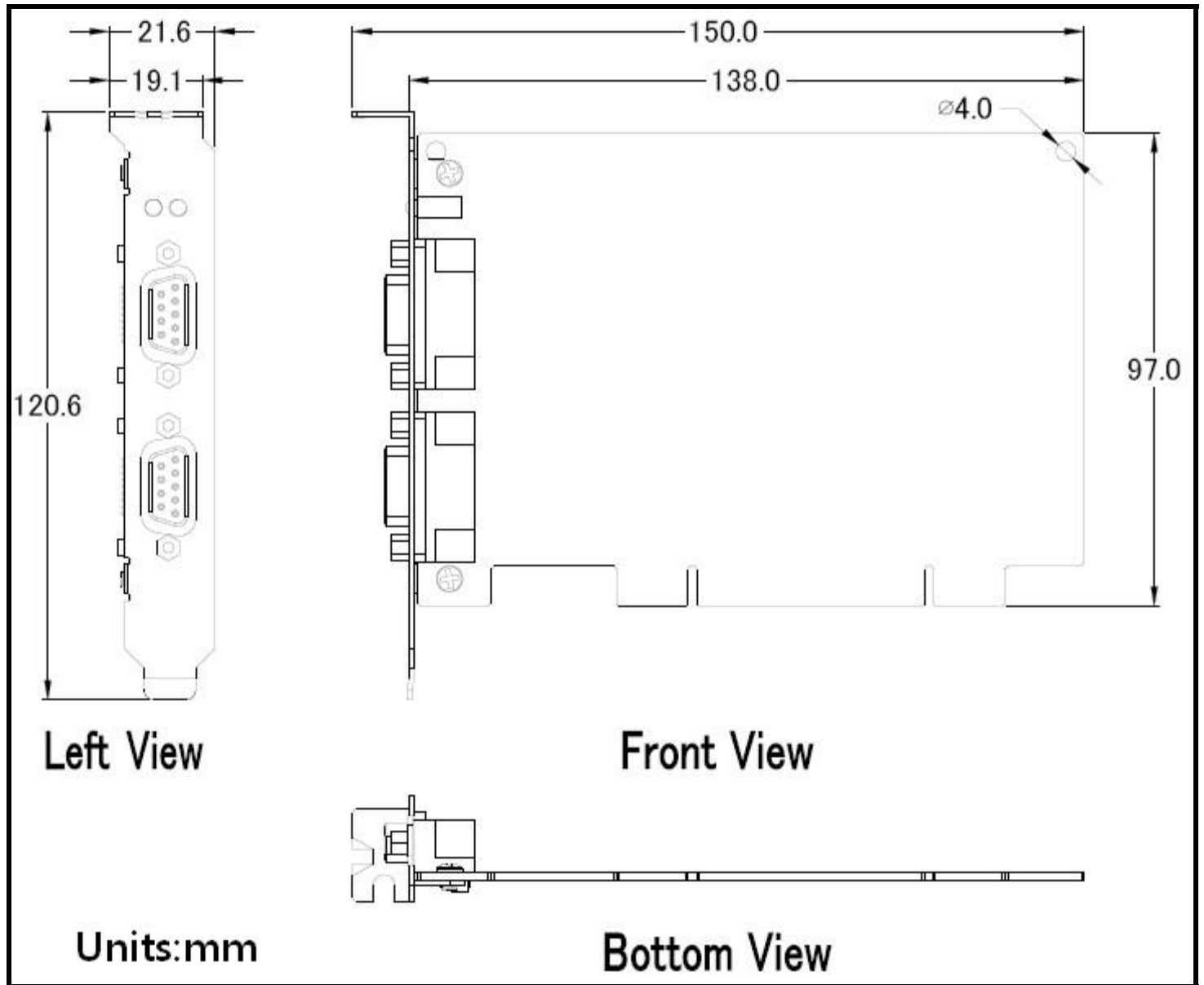
Model	PISO-CM200U-D	PISO-CM200U-T
Hardware		
CPU	32-bit MCU	
DPRAM	16 KB	
RTC (Real Time Clock)	Yes	
Bus Interface		
Type	Universal PCI, 5 V, 33 MHz, 32-bit, plug and play	
Board No.	Configurable via DIP switch	
CAN Interface		
Controller	Bosch C_CAN	
Transceiver	NXP 82C250	
Channel number	2	
Connector	9-pin Male D-Sub	5-pin screw terminal block
Baud Rate (bps)	10 k, 20 k, 50 k, 125 k, 250 k, 500 k, 800 k, 1 M (user-defined Baud Rate allowed)	
Isolation	3000 V _{DC} for DC-to-DC, 3000 V _{rms} for photo-couple	
Terminator Resistor	Jumper for 120 Ω terminator resistor	
LED Indicators/Display		
System LED Indicators	Yes, two (round) as Communication Indicators, Rx/Tx, ERR	
Power		
Power Consumption	800 mA @ 5 V	
Software		
Drivers	Windows XP/7/8.1/10 (32-bit/64-bit)	
Library/Demo Languages	C#.Net, VB.Net, VC++.Net	
Mechanism		
Dimensions (L x W x D)	150 mm x 121 mm x 22 mm	
Environment		
Operating Temperature	0 to 60°C	
Storage Temperature	-20 to 70°C	
Humidity	5 to 85% RH, Non-condensing	

1.2. Features

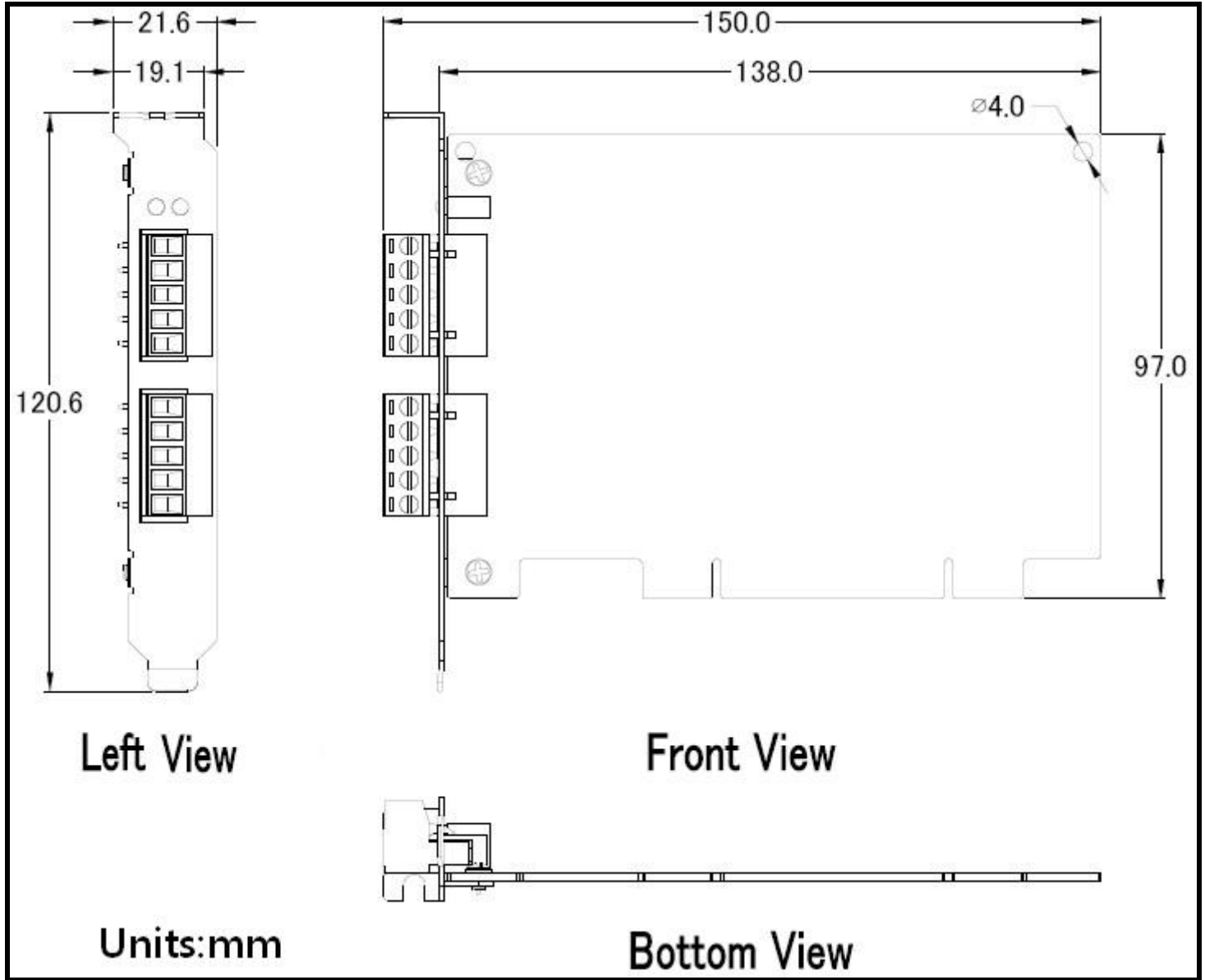
- Embedded 32-bit microprocessor
- NXP 82C250 CAN transceiver
- Bosch C_CAN controller
- Supports both CAN 2.0A and CAN 2.0B specifications
- Storage Timestamp with a precision of at least ± 1 ms
- Board number selectable via DIP switch
- Dual port RAM communication mechanism
- Embedded RTC (real time clock)
- Drivers provided for Windows XP/7/8.1/10
- Demos and libraries provided for C#.Net, VB.Net and VC++.Net
- Transmission/reception buffer for up to 256 CAN messages
- Allows a maximum of 5 sets of cyclic transmission messages
- Cyclic transmission message precision: ± 1 ms
- Easy to update firmware

1.3. Dimensions

PISO-CM200U-D



PISO-CM200U-T

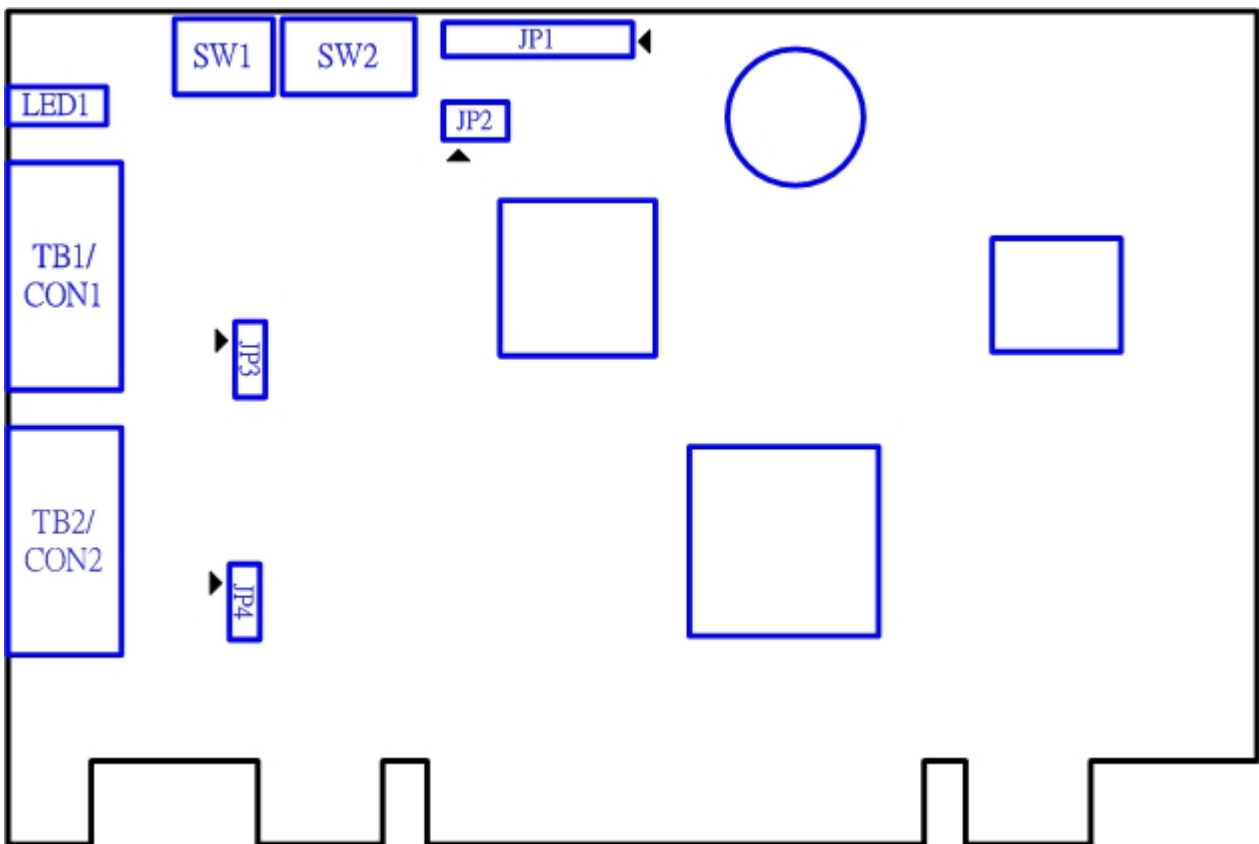


1.4. Overview

The following is a description of the hardware settings for the PISO-CM200U, including the board layout, pin assignments, jumper and switch selection, LED indicators, and the configuration for the wiring connections.

1.4.1. Board Layout

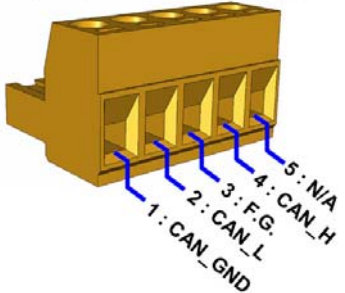
The following is the layout for the PISO-CM200U board, illustrating the positions of the various connectors, jumpers and switches on the board.



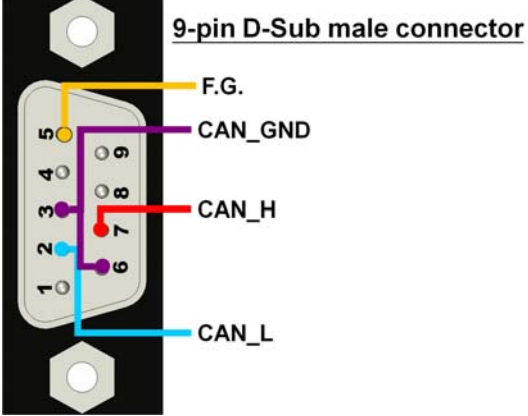
1.4.2. Pin Assignments

The pin assignments for the 5-pin screw terminal connector and 9-pin Male D-Sub connector on the PISO-CM200U board are shown below.

Pin Assignments for the 5-pin screw terminal connector		
Pin No.	Name	Description
1	CAN_GND	CAN_Gnd, signal line for the CAN port.
2	CAN_L	CAN_Low, signal line for the CAN port.
3	F.G.	Frame Ground.
4	CAN_H	CAN_High, signal line for the CAN port.
5	N/A	Not used




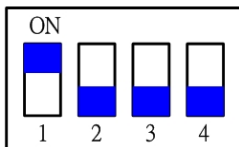
Pin Assignments for the 9-pin Male D-Sub connector		
Pin No.	Name	Description
1	N/A	Not used
2	CAN_L	CAN_Low, signal line for the CAN port.
3	CAN_GND	CAN_Gnd, signal line for the CAN port.
4	N/A	Not used
5	F.G.	Frame Ground.
6	CAN_GND	CAN_Gnd, signal line for the CAN port.
7	CAN_H	CAN_High, signal line for the CAN port.
8	N/A	Not used
9	N/A	Not used



Electronic circuits are always influenced by different levels of Electrostatic Discharge (ESD), which become worse in a continental climate area. The built-in F.G. provides a path for conducting the ESD to the earth ground. Therefore, connecting the F.G. correctly can enhance the ESD protection capabilities and improve the reliability of the module. Note that wiring of the F.G. is not necessary. The wiring configuration can be modified based on the specific application.

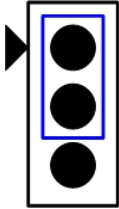
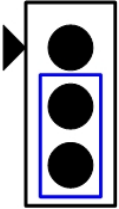
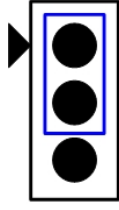
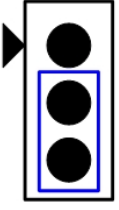
1.4.3. Switch Settings

The following provides a description of the SW1 switch and SW2 DIP switch, which can be referred to when configuring the PISO-CM200U series hardware.

Switch	Description	Status
SW1	The SW1 switch is used to set the firmware into download mode. To force the CAN board into download mode, press the switch for more than 3 seconds to reset the firmware and set the firmware to download mode.	 SW1
SW2	SW2 is a DIP switch that is used to configure the board number for the PISO-CM200U. For example, if the switch on the left-hand side, (i.e., DIP switch 1, is set to ON, as indicated in the figure, the board number will be set to 1. The board number can range from 0 to 15. Note that the board number for each PISO-CM200U installed in the Host PC must be unique.	<p>DIP Switch</p>  <p>Thus configuration indicates that the board number is set to 1.</p>

1.4.4. Terminal Resistor Jumper Settings

The following provides a description of the resistor jumpers for the CAN Bus terminal, which can be referred to when configuring the PISO-CM200U series hardware.

Jumper	Description	Status	
JP3	Used to enable or disable the 120Ω terminal resistor for CAN Bus Port1.	 JP3 Enabled	 JP3 Disabled
JP4	Used to enable or disable the 120Ω terminal resistor for CAN Bus Port2.	 JP4 Enabled	 JP4 Disabled

1.4.5. LED Indicators and Operating Modes

The PISO-CM200U can operate in one of three modes and the status of the LED indicators will change depending on the mode. The following is an overview of each mode together with a description of the LED status.

1. Boot Loader Mode

When the PISO-CM200U is operating in Boot Loader Mode, the green LED and the red LED will alternately flash once per second, i.e., when the green LED is ON, the red LED will be OFF, and vice versa. Then, the firmware upgrade software can be used to download the latest version of the default firmware. For details of how to install or upgrade to the latest version of the firmware, see Section 4.

2. Firmware Running Mode

When the PISO-CM200U is operating using the default firmware, the green LED will flash once as the PISO-CM200U successfully transmits or receives a single CAN message to or from the CAN bus. If the loading on the CAN bus is heavy, the green LED may be constantly lit. If an error occurs, the red LED will be lit. Use the “CM200_CANGetStatus” API function (see Section 3.2.2 for more details) to retrieve the status of the CAN bus.

3. Firmware Reset Mode

To reset the firmware, press switch SW1 on the PISO-CM200U for more than 3 seconds. After pressing the switch, both the red and green LEDs will be lit for about 1 second, and the PISO-CM200U will be forced into Boot Loader mode. If the PISO-CM200U malfunctions because of software errors or other problems, use this method to reset the firmware, and then download and install the latest version of the firmware. For details of how to install or upgrade to the latest version of the firmware, see Section 4.

2. Getting Started

This following is a description of how to begin using the PISO-CM200U series board, including installing the driver installing the hardware and making the wiring connections.

2.1. Installing the Windows Driver

To use the PISO-CM200U series board in a Windows environment, the correct driver for the specific version of the Windows operating system must be installed. The drivers can be found on the FieldBus CD that is included in the PISO-CM200U package in the following location:

CD:\can\pci\piso-cm200u\driver\windows.

The drivers can also be downloaded from the ICP DAS website at either of the following addresses:

ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/driver/windows

or

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/driver/windows

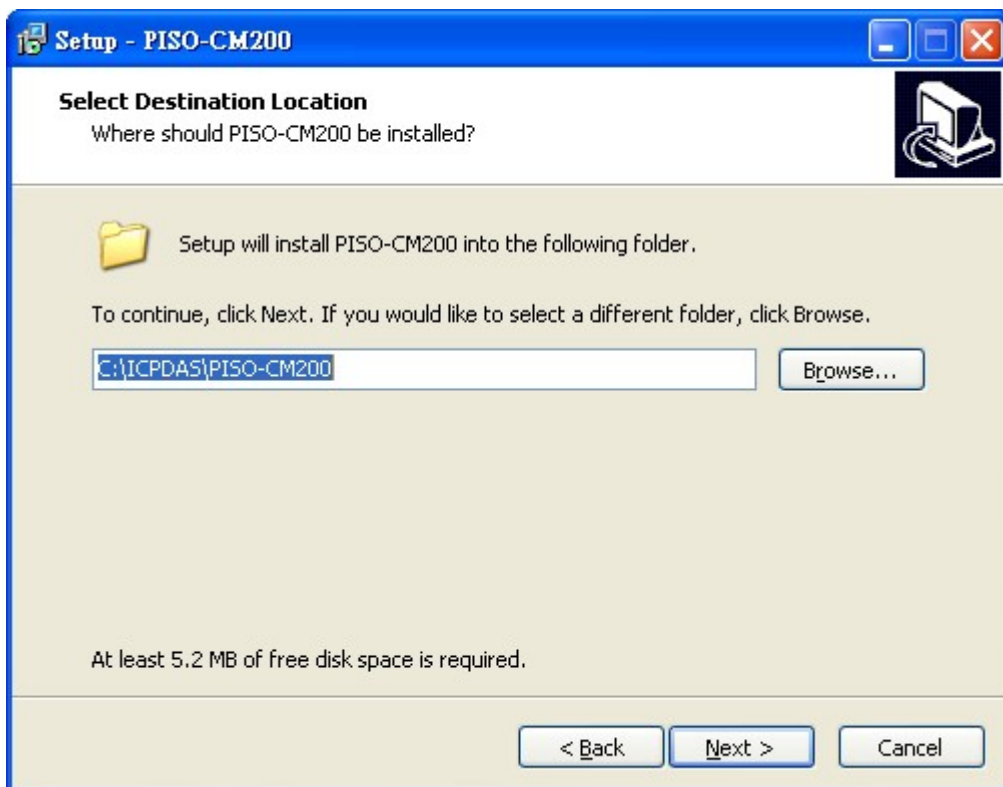
The following is a description of the installation procedure for the Windows XP operating system. The installation procedures for other versions of Windows are similar.

Step1: Execute the driver installation file, piso-cm200u_setup.exe, which can be found on the FieldBus CD that is included PISO-CM200U package or can be downloaded from the ICP DAS website.

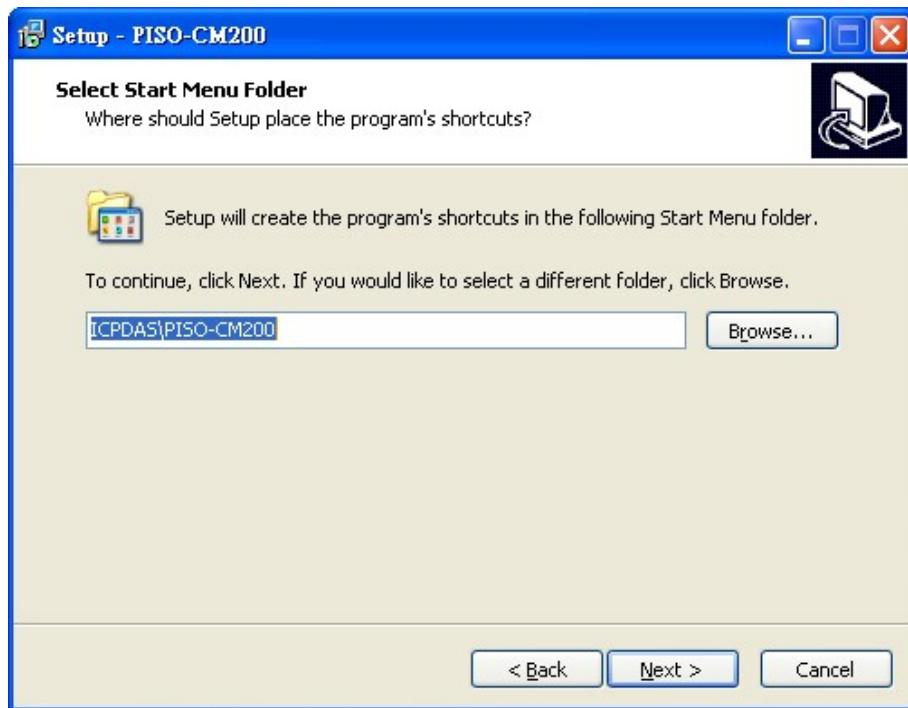
When the Setup Wizard is displayed, click the **Next >** button to continue.



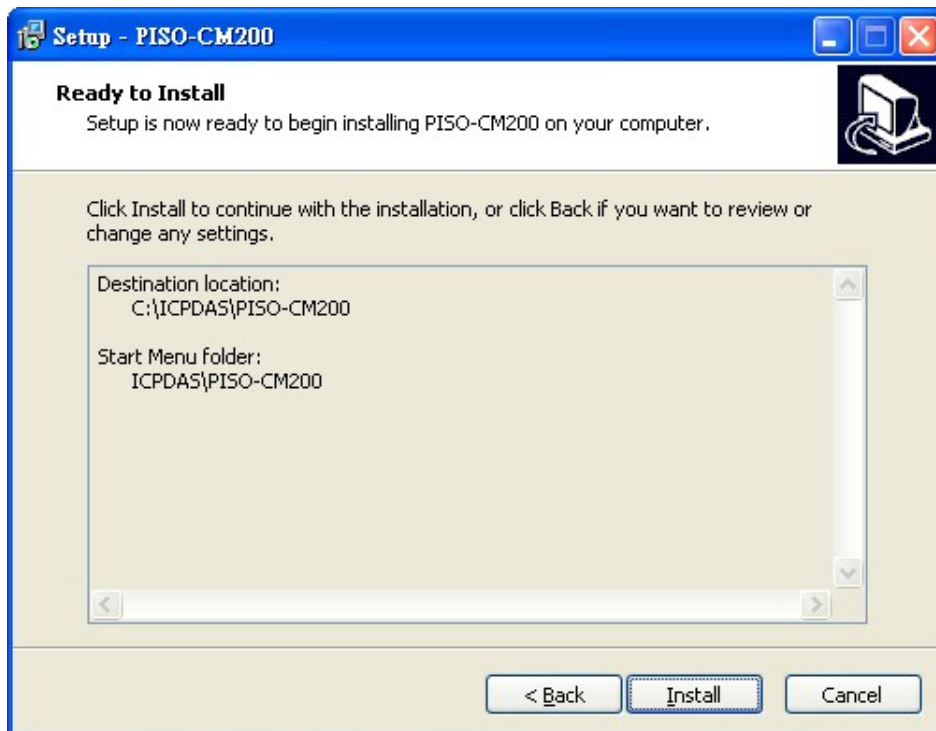
Step 2: On the **Select Destination Location** screen, either click the **Next >** button to install the driver into the default folder, or click the **Browse...** button to select an alternate folder, and then click the **Next >** button to continue.



Step 3: On the **Select Start Menu Folder** screen, either click the **Next >** button to allow the shortcuts to be created in the default Start Menu folder, or click the **Browse...** button to select an alternate folder, and then click the **Next >** button to continue.



Step 4: On the **Ready to Install** screen, verify that the settings are correct, and then click the **Install** button to begin the installation. The **Setup Wizard** will display a progress bar to indicate the status of the installation process. Click the **Cancel** button to stop the installation if necessary.



Step 5: Once the installation has been completed, click the Finish button to exit the Setup Wizard and automatically restart the computer.



2.2. Installing the Hardware

Step 1: Shut down and power off the computer.

Step 2: Remove all the covers from the computer.

Step 3: Select an unused PCI slot.

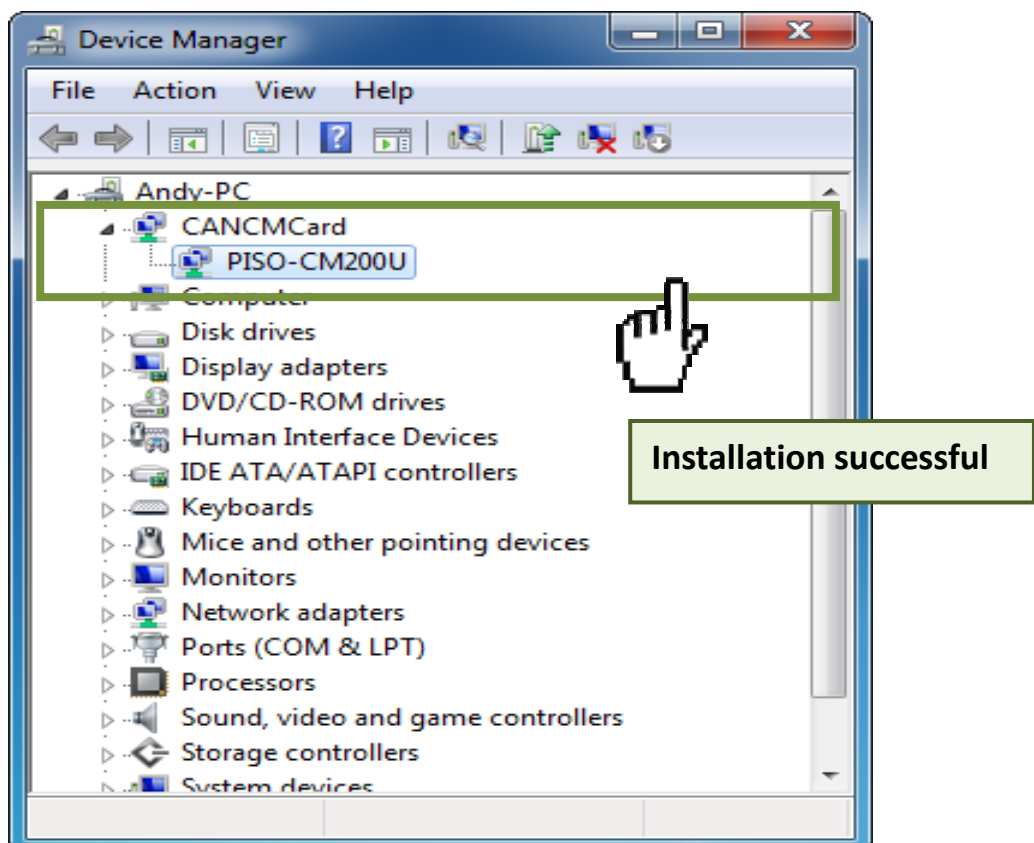
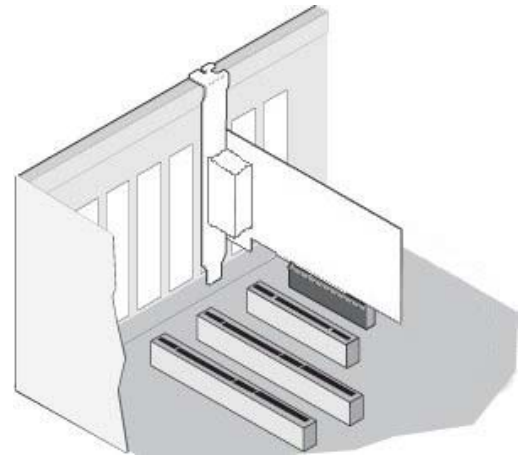
Step 4: Carefully insert the PISO-CM200U board into the PCI slot and secure the board in place.

Step 5: Replace the covers on the computer.

Step 6: Reconnect the power supply and power on the computer.

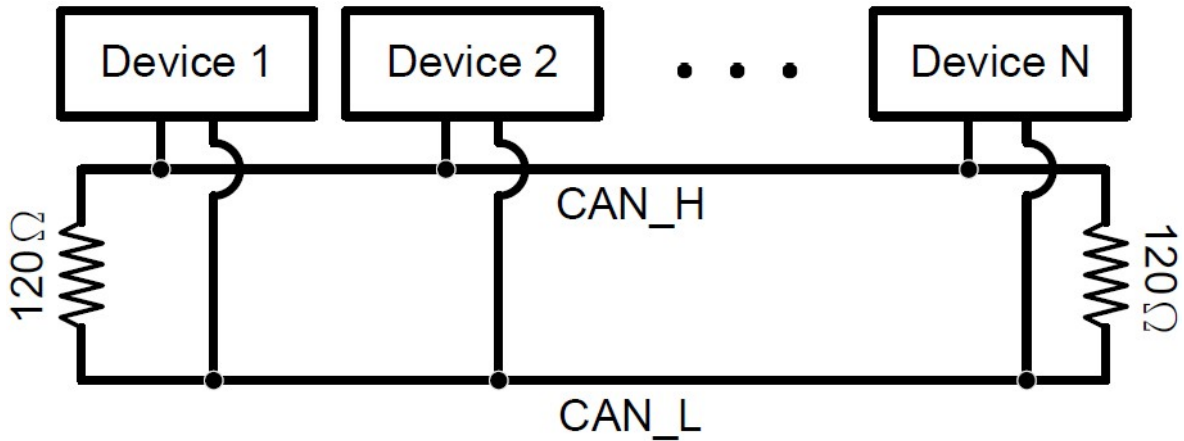
Step 7: Once the computer reboots, follow any messages that may be displayed to complete the Plug and Play installation procedure.

Step 8: Open the “**Device Manager**” in the Control Panel and verify that the PISO-CM200U board is listed correctly, as illustrated below.



2.3. Wiring Connections

In order to minimize any reflection effects on the CAN bus line, it must be terminated at each end using a terminator resistor, as illustrated in the diagram below. The specifications provided in ISO 11898-2 state that each terminator resistor must be 120 Ω (or between 108 Ω and 132 Ω). The bus topology and the positions of these terminator resistors are shown below.



To minimize any voltage drop over long distances, the terminal resistance should be greater than the resistance value defined in the ISO 11898-2. The following table can be used as a good reference.

Bus Length (meter)	Bus Cable Parameters		Terminal Resistance (Ω)
	Length Related Resistance (mΩ/m)	Cross Section (Type)	
0~40	70	0.25(23AWG)~ 0.34mm ² (22AWG)	124 (0.1%)
40~300	<60	0.34(22AWG)~ 0.6mm ² (20AWG)	127 (0.1%)
300~600	<40	0.5~0.6mm ² (20AWG)	150~300
600~1K	<20	0.75~0.8mm ² (18AWG)	150~300

3. Windows API Function Reference

This chapter describes the APIs that relate to the default firmware, including the System Information API, the CAN Bus API and an overview of the error codes, which can be helpful when developing custom applications.

3.1. System Information API

The following is an overview of the System Information API functions provided on the PISO-CM200U series board. A detailed description of each API is presented in subsequent sections.

Function	Description
CM200_GetDIIVersion	Used to retrieve the version number for the function library file currently installed on the PISO-CM200U series board
CM200_GetBoardInf	Used to retrieve information related to a selected PISO-CM200U series board, including the vendor ID, the device ID, etc.
CM200_TotalBoard	Used to retrieve the total number of PISO-CM200U series boards installed in the Host PC
CM200_TotalCM200Board	Used to retrieve the total number of PISO-CM200U boards installed in the Host PC.
CM200_GetCM200BoardSwitchNo	Used to retrieve the switch (SW2) number for the selected PISO-CM200U series board
CM200_GetCardPortNum	Used to retrieve the CAN port number being used by the selected PISO-CM200U series board
CM200_ActiveBoard	Used to activate the selected PISO-CM200U series board
CM200_CloseBoard	Used to deactivate the selected PISO-CM200U series board
CM200_BoardIsActive	Used to check whether or not the selected PISO-CM200U series board is active
CM200_CheckMCUMode	Used to Check the operating mode currently being used by the selected PISO-CM200U series board
CM200_HardwareReset	Used to reset the hardware for the selected PISO-CM200U series board
CM200_AdjustDateTime	Used to adjust the date and time for the selected PISO-CM200U series board

3.1.1. CM200_GetDllVersion

Description

This function is used to retrieve the version number of the CM200.dll driver currently installed for the PISO-CM200U series board.

Syntax

C#

```
-----  
uint16 CM200_GetDllVersion();
```

Parameters

This function has no parameters.

Return Value

If the function succeeds, the return value will be the version information for the CM200.dll driver currently installed for the PISO-CM200U series board in hexadecimal format. For example, a return value of “100” indicates that the driver version is “1.00”.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.2. CM200_GetBoardInf

Description

This function is used to retrieve information of related to a specified PISO-CM200U series board, including the vendor ID, the device ID and the interrupt number, etc.

Syntax

C#

```
Int16 CM200_GetBoardInf(  
    Byte BoardNo,  
    out UInt32 dwVID,  
    out UInt32 dwDID,  
    out UInt32 dwSVID,  
    out UInt32 dwSDID,  
    out UInt32 dwSAuxID,  
    out UInt32 dwIrqNo  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

dwVID

[out] Indicates the address of a variable used to receive the vendor ID.

dwDID

[out] Indicates the address of a variable used to receive the device ID.

dwSVID

[out] Indicates the address of a variable used to receive the sub-vendor ID.

dwSDID

[out] Indicates the address of a variable used to receive the sub-device ID.

dwSAuxID

[out] Indicates the address of a variable used to receive the sub-auxiliary ID.

dwIrqNo

[out] Indicates the address of a variable used to receive the logical interrupt number.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.3. CM200_TotalBoard

Description

This function is used to retrieve the total number of PISO-CM200U series boards currently installed in the Host PC.

Syntax

```
C#  
-----  
Int16 CM200_TotalBoard (  
    out Byte BoardNo  
);
```

Parameters

BoardNo

[out] Indicates the total number of PISO-CM200U series boards that were scanned.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.4. CM200_TotalCM200Board

Description

This function is used to retrieve the total number of PISO-CM200U boards currently installed in the Host PC.

Syntax

```
C#  
-----  
Int16 CM200_TotalCM200Board (  
    out Byte BoardNo  
);
```

Parameters

BoardNo

[out] Indicates the total number of PISO-CM200U boards that were scanned.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.5. CM200_GetCM200BoardSwitchNo

Description

This function is used to retrieve the current configuration of the DIP switch (SW2) on the specified PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_GetCM200BoardSwitchNo(  
    Byte BoardNo,  
    out Byte BoardSwitchNo  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

BoardSwitchNo

[out] Indicates the address of a variable used to retrieve the current configuration of the DIP switch (SW2) on the PISO-CM200U series board.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.6. CM200_GetCardPortNum

Description

This function is used to retrieve the number of the CAN port on a specified PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_GetCardPortNum(  
    Byte BoardNo,  
    out Byte PortNum  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

PortNum

[out] Indicates the address of a variable used to retrieve the number of the CAN port on the PISO-CM200U series board.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.7. CM200_ActiveBoard

Description

This function is used to activate a specified PISO-CM200U series board. Note that this function **MUST** be called before using any other API functions.

Syntax

```
C#  
-----  
Int16 CM200_ActiveBoard(  
    Byte BoardNo  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.8. CM200_CloseBoard

Description

This function is used to deactivate a specified PISO-CM200U series board. This function MUST always be called at the end of a program in order for the system to release resources before exiting the program.

Syntax

C#

```
Int16 CM200_CloseBoard(  
    Byte BoardNo  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.9. CM200_BoardIsActive

Description

This function is used to check whether or not a specified PISO-CM200U series board is active.

Syntax

```
C#
-----
Int16 CM200_BoardIsActive(
    Byte BoardNo,
    out Byte IsActive
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

IsActive

[out] Indicates the address of a variable used to retrieve the status of the selected PISO-CM200U series board, where:

0: The selected board is not active

1: The selected board has been activated

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.10. CM200_CheckMCUMode

Description

This function is used to check the mode that a specified PISO-CM200U series board is currently operating in.

Syntax

C#

```
Int16 CM200_CheckMCUMode (  
    Byte BoardNo,  
    out Byte Mode  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Mode

[out] Indicates the address of a variable used to retrieve the mode that the selected PISO-CM200U series board is currently operating in.

If this value is 0, the board is currently operating in “Boot Loader” mode, and if the value 1, the board is currently operating in “Firmware Running” mode. If the board is operating in “Boot Loader” mode, only the firmware can be updated and the firmware will not work in this mode. Use the `CM200_HardwareReset()` function to switch the board to “Firmware Running” mode.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.11. CM200_HardwareReset

Description

This function is used to reset the hardware on a specified PISO-CM200U series board, including the CAN controller, the firmware, and the DPRAM functions on the selected board.

Syntax

```
C#
-----
Int16 CM200_HardwareReset (
    Byte BoardNo
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be reset. The valid range is 0 to 15.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.1.12. CM200_AdjustDateTime

Description

This function is used to adjust the date and time for a specified PISO-CM200U series board based on the current time on the Host PC.

Syntax

C#

```
-----  
Int16 CM200_AdjustDateTime (  
    Byte BoardNo  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be set. The valid range is 0 to 15.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2. CAN Bus API

The following is an overview of the CAN Bus API functions provided on the PISO-CM200U series board. A detailed description of each function is presented in subsequent sections.

Function	Description
CM200_CANReset	Used to reset the CAN controller on the selected PISO-CM200U series board
CM200_CANGetStatus	Used to retrieve the status of the CAN port on the selected PISO-CM200U series board
CM200_EnableCAN	Used to enable the CAN port function on the selected PISO-CM200U series board
CM200_DisableCAN	Used to disable the CAN port function on the selected PISO-CM200U series board
CM200_SetCANConfig	Used to configure the CAN port on the selected PISO-CM200U series board
CM200_GetCANConfig	Used to retrieve the current configuration for the CAN port on the selected PISO-CM200U series board
CM200_AddCyclicTxMsg	Used to add a cyclic transmission message to the CAN port on the selected PISO-CM200U series board
CM200_DeleteCyclicTxMsg	Used to delete a cyclic transmission message from the CAN port on the selected PISO-CM200U series board
CM200_EnableCyclicTxMsg	Used to enable the cyclic transmission message function for the CAN port on the selected PISO-CM200U series board
CM200_DisableCyclicTxMsg	Used to disable the cyclic transmission message function for the CAN port on the selected PISO-CM200U series board
CM200_CheckCyclicTxRestMsg	Used to check the number of cyclic transmission messages remaining in the CAN port on the selected PISO-CM200U series board
CM200_IsTxTimeout	Used to check whether or not the transmission message was successfully transmitted
CM200_CANInit	Used to initialize the CAN port function on the selected PISO-CM200U series board
CM200_RxMsgCount	Used to retrieve the message counter of the CAN port on the selected PISO-CM200U series board
CM200_ReceiveCANMsg	Used to retrieve the CAN message of the CAN port on the selected PISO-CM200U series board
CM200_SendCANMsg	Used to send the CAN message of the CAN port on the selected

	PISO-CM200U series board
CM200_ClearSoftBuffer	Used to clear all the data buffer for the CAN port on the selected PISO-CM200U series board
CM200_ClearTxSoftBuffer	Used to clear the transmitted data buffer for the CAN port on the selected PISO-CM200U series board
CM200_ClearRxSoftBuffer	Used to clear the received data buffer for the CAN port on the selected PISO-CM200U series board
CM200_ClearBufferStatus	Used to clear the data buffer status for the CAN port on the selected PISO-CM200U series board

3.2.1. CM200_CANReset

Description

This function is used to reset the CAN controller for the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#  
-----  
Int16 CM200_CANReset (  
    Byte BoardNo,  
    Byte Port  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.2. CM200_CANGetStatus

Description

This function is used to retrieve the status register value for the CAN controller of the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#
-----
Int16 CM200_CANGetStatus (
    Byte BoardNo,
    Byte Port,
    out Byte Status
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Status

[out] Indicates the address of a variable used to retrieve the status register value for the CAN controller.

Bit	Symbol	Value	Description
2:0	LEC		Last error code The LEC field holds a code which indicates the type of the last error to occur on the CAN bus.
		0x0	No error has occurred
		0x1	Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.

		0x2	Form error: The format of a fixed format part of a received frame is wrong.
		0x3	AckError: The message that was transmitted by this CAN controller was not acknowledged
		0x4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device attempted to send a HIGH/recessive signal level (bit of logical value '1'), but the monitored bus value was LOW/dominant.
		0x5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device attempted to send a LOW/dominant signal level (data or identifier bit logical value '0'), but the monitored Bus value was HIGH/recessive.
		0x6	CRCErrror: The CRC checksum in the message received was incorrect
		0x7	Unused: No CAN bus event was detected
3	TXOK		The message was transmitted successfully
		0	No messages have been transmitted successfully
		1	A message has been transmitted successfully.
4	RXOK		A message was received successfully
		0	No messages have been received successfully
		1	A message has been received successfully independent of the result of acceptance filtering.
5	EPASS		assive Error
		0	The CAN controller is in the error active state.
		1	The CAN controller is in the error passive state as defined in the CAN 2.0 specification.
6	EWARN		Warning status
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters in the Error Counter Register has reached the error warning limit of 96.
7	BOFF		Busoff status
		0	The CAN module is not in the busoff state.
		1	The CAN controller is in the busoff state.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.3. CM200_EnableCAN

Description

This function is used to enable the transmit/receive interrupt function for the CAN controller of the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#
-----
Int16 CM200_EnableCAN (
    Byte BoardNo,
    Byte Port
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.4. CM200_DisableCAN

Description

This function is used to disable the transmit/receive interrupt function for the CAN controller of the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#
-----
Int16 CM200_DisableCAN (
    Byte BoardNo,
    Byte Port
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.5. CM200_SetCANConfig

Description

This function is used to configure the Baud Rate, bit sample point and message filter parameters for the CAN controller of the CAN port on a specified PISO-CM200U series board. The new configuration will be saved into the EEPROM for use by the firmware on the PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_SetCANConfig(  
    Byte BoardNo,  
    Byte Port,  
    UInt32 BaudRate,  
    UInt32 SamplePoint,  
    Byte FilterMode,  
    UInt32 FilterArbitBit,  
    UInt32 FilterMaskBit  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port1
2: CAN port 2.

BaudRate

[in] Specifies the Baud Rate to be configured for the CAN Bus. The units are bps, and the valid range is from 5000 bps to 1000000 bps.

SamplePoint

[in] Specifies the bit sample point as a percentage. The valid range is from 0.00% to 100.00% in intervals of 0.01%.

FilterMode

[in] Specifies the Filter mode based on the CAN Bus specifications, where:
0: Specifies the CAN2.0A specification, which uses the 11-bit CAN ID filter.
1: Specifies the CAN2.0B specification, which uses the 29-bit CAN ID filter

FilterArbitBit

[in] Specifies the FilterArbitBit that is used to determine which CAN IDs will be accepted by the CAN controller.

FilterMode	FilterArbitBit (hexadecimal) Range
11-bit CAN ID	000 to 7FF
29-bit CAN ID	00000000 to 1FFFFFFF

FilterMaskBit

[in] Specifies the FilterMaskBit that is used to determine which bit from the CAN ID will be checked by the CAN controller based on the value specified for the FilterArbitBit parameter. If the bit of the FilterMaskBit parameter is set to 1, it means that the bit in the CAN ID in the same position needs to be checked, and the ID bit value needs to match the value specified for the bit of the FilterArbitBit parameter in the same position.

FilterMode	FilterMaskBit (hexadecimal) Range
11-bit CAN ID	000 to 7FF
29-bit CAN ID	00000000 to 1FFFFFFF

Example using a 29-bit CAN ID

	(MSB)			(LSB)
CAN ID Bit	Bit 31 to 24	Bit 23 to 16	Bit 15 to 8	Bit 7 to 0
FilterArbitBit (hexadecimal)	00	00	00	A0
FilterMaskBit (hexadecimal)	00	00	00	E0
Accepted CAN ID (x: don't care bit value)	xxxx xxxx	xxxx xxxx	xxxx xxxx	101x xxxx

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.6. CM200_GetCANConfig

Description

This function is used to retrieve the configuration for the Baud Rate, the bit sample point, and the message filter on the CAN controller for a specified PISO-CM200U CAN port.

Syntax

```
C#
-----
Int16 CM200_SetCANConfig(
    Byte BoardNo,
    Byte Port,
    out UInt32 BaudRate,
    out UInt32 SamplePoint,
    out Byte FilterMode,
    out UInt32 FilterArbitBit,
    out UInt32 FilterMaskBit
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

BaudRate

[out] Indicates the address of a variable used to retrieve the Baud Rate that is currently configured for the CAN Bus. The units are bps, and the valid range is from 5000 bps to 1000000 bps.

SamplePoint

[out] Indicates the address of a variable used to retrieve the Bit sample point

that is currently configured for the CAN Bus. The valid range is from 0.00% to 100.00% in intervals of 0.01%.

FilterMode

[out] Indicates the address of a variable used to retrieve the Filter mode that is currently configured for the CAN Bus based on the CAN Bus specifications, where:

0: Denotes the CAN2.0A specification, which uses an 11-bit CAN ID filter.

1: Denotes the CAN2.0B specification, which uses a 29-bit CAN ID filter.

FilterArbitBit

[out] Indicates the address of a variable used to retrieve the value currently configured for the FilterArbitBit, which is used to determine the CAN IDs that will be accepted by the CAN controller.

FilterMode	FilterArbitBit (hexadecimal) range
11-bit CAN ID	000 to 7FF
29-bit CAN ID	00000000 to 1FFFFFFF

FilterMaskBit

[out] Indicates the address of a variable used to retrieve the value of the FilterMaskBit that is used to determine which bit from the CAN ID will be checked by the CAN controller based on the value specified for the FilterArbitBit parameter. If the bit of the FilterMaskBit parameter is set to 1, it means that the bit in the CAN ID in the same position needs to be checked, and the ID bit value needs to match the value specified for the bit of FilterArbitBit parameter in the same position.

FilterMode	FilterMaskBit (hexadecimal) range
11-bit CAN ID	000 to 7FF
29-bit CAN ID	00000000 to 1FFFFFFF

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.7. CM200_AddCyclicTxMsg

Description

This function is used to add a cyclic transmission message to the firmware for the CAN port on a specified PISO-CM200U series board. Subsequently, the `CM200_EnableCyclicTxMsg()`, `CM200_DisableCyclicTxMsg()` and `CM200_DelectCyclicTxMsg()` functions can be used to enable, disable, or delete these cyclic transmission messages. A maximum of 5 cyclic transmission messages can be added.

The *Handle* variable indicates the variable number for the cyclic transmission message, and is generated automatically. After adding a cyclic transmission message, the handle for this data will be returned. A lower value for the handle indicates a higher priority for the cyclic transmission message. If two cycle data transmissions need to be sent at the same time, the data with the higher priority will be sent first.

Syntax

```
C#
-----
Int16 CM200_AddCyclicTxMsg(
    Byte BoardNo,
    Byte Port,
    Byte Mode,
    UInt32 MsgID,
    Byte RTR,
    Byte DataLen,
    Byte[] Data,
    UInt32 TimePeriod,
    UInt32 TransmitTimes,
    out Byte Handle
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The

valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Mode

[in] Specifies the CAN Bus specification to be used for the CAN message ID.
0: Denotes the CAN2.0A specification, which uses an 11-bit CAN message ID.
1: Denotes the CAN2.0B specification, which uses a 29-bit CAN message ID.

MsgID

[in] Specifies the CAN message ID.

RTR

[in] Specifies the mode that will be used to transmit the message, where:
0: Indicates that the message will be transmitted as a remote-transmit-request message
1: Indicates that the message will be transmitted as a normal message

Note that as this parameter will be set to 1, the *Data* parameter will be useless.

DataLen

[in] Specifies the length of the CAN message data. The maximum value is 8.

Data

[in] Specifies the starting address of the data array used for a CAN message. The maximum length of the Data array is 8. Useless for the RTR parameter is setting to 1.

TimePeriod

[in] Specifies the time period to be used for the cyclic transmission message in increments interval of 0.1 milli-seconds. The minimum value is 5 milli-seconds.

TransmitTimes

[in] Specifies the number of times the cyclic transmission message is to be transmitted. After enabling transmission, the message will be sent for the number of times specified. If the value of the parameter is 0, transmission will be repeated until the cyclic transmission function is disabled.

Handle

[out] Indicates the address of a variable that is used to retrieve the handle for a cyclic transmission message. This value is needed in order to enable, disable, or delete a specific transmission using the `CM200_EnableCyclicTxMsg()`, `CM200_DisableCyclicTxMsg()`, or `CM200_DelectCyclicTxMsg()` functions.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.8. CM200_DeleteCyclicTxMsg

Description

This function is used to delete the specified cyclic transmission message that was added to the firmware for the CAN port on a specified PISO-CM200U series board using the function CM200_AddCyclicTxMsg().

Syntax

```
C#
-----
Int16 CM200_DeleteCyclicTxMsg (
    Byte BoardNo,
    Byte Port,
    Byte Handle
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

Handle

[in] Specifies the handle for the cyclic transmission message, which is obtained using the CM200_AddCyclicTxMsg() function.

Return Value

If the function succeeds, the return value will be 0.
If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.9. CM200_EnableCyclicTxMsg

Description

This function is used to enable the cyclic transmission message that was added to the firmware for the CAN port on a specified PISO-CM200U series board using the CM200_AddCyclicTxMsg() function. After calling this function, the specified cyclic transmission message will be transmitted.

Syntax

```
C#
-----
Int16 CM200_EnableCyclicTxMsg (
    Byte BoardNo,
    Byte Port,
    Byte Handle
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

Handle

[in] Specifies the handle for the cyclic transmission message, which is obtained using the CM200_AddCyclicTxMsg() function.

Return Value

If the function succeeds, the return value will be 0.
If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.10. CM200_DisableCyclicTxMsg

Description

This function is used to disable the cyclic transmission message that was added to the firmware for the CAN port on a specified PISO-CM200U series board using the CM200_AddCyclicTxMsg() function. After calling this function, transmission of the specified cyclic transmission message will stop.

Syntax

```
C#
-----
Int16 CM200_DisableCyclicTxMsg (
    Byte BoardNo,
    Byte Port,
    Byte Handle
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

Handle

[in] Specifies the handle for the cyclic transmission message, which is obtained using the CM200_AddCyclicTxMsg() function.

Return Value

If the function succeeds, the return value will be 0.
If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.11. CM200_CheckCyclicTxRestMsg

Description

This function is used to check how many cyclic transmission messages remain unsent, as specified using the *TransmitTimes* parameter of the `CM200_AddCyclicTxMsg()` function.

Syntax

```
C#
-----
Int16 CM200_CheckCyclicTxRestMsg (
    Byte BoardNo,
    Byte Port,
    Byte Handle,
    out UInt32 RestCount
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

Handle

[in] Specifies the handle for the cyclic transmission message, which is obtained using the `CM200_AddCyclicTxMsg()` function.

RestCount

[out] Specifies the address of a variable used to retrieve the number of cyclic transmissions remaining, as specified using the *TransmitTimes* parameter of the `CM200_AddCyclicTxMsg()` function.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.12. CM200_IsTxTimeout

Description

This function is used to check whether or not the CAN controller has successfully sent the CAN message.

Syntax

```
C#  
-----  
Int16 CM200_IsTxTimeout (  
    Byte BoardNo,  
    Byte Port,  
    out Byte Status  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

Status

[out] Indicates the transmission status. A value of 1 indicates that the CAN controller was not successful when attempting to send the CAN message to the CAN Bus network, and a value of 0 indicates that there was no error and that the message was transmitted successfully.

Return Value

If the function succeeds, the return value will be 0.
If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.13. CM200_CANInit

Description

This function is used to initialize the CAN controller of the CAN port on a specified PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_CANInit (  
    Byte BoardNo,  
    Byte Port  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.14. CM200_RxMsgCount

Description

This function is used to retrieve the count of the received CAN message on the CAN controller for a specified PISO-CM200U CAN port.

Syntax

```
C#  
-----  
Int16 CM200_RxMsgCount (  
    Byte BoardNo,  
    Byte Port,  
    out UInt16 MsgCount  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:
1: CAN port 1
2: CAN port 2.

MsgCount

[out] Indicates the count of the received CAN messages.

Return Value

If the function succeeds, the return value will be 0.
If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.15. CM200_ReceiveCANMsg

Description

This function is used to retrieve a CAN message on the CAN controller for a specified PISO-CM200U CAN port.

Syntax

C#

```
Int16 CM200_ReceiveCANMsg (  
    Byte BoardNo,  
    Byte Port,  
    out Byte Mode,  
    out UInt32 MsgID,  
    out Byte RTR,  
    out Byte DataLen,  
    Byte[] Data,  
    out UInt32 UpperTime,  
    out UInt32 LowerTime  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Mode

[out] Indicates the address of a variable used to retrieve the the CAN Bus specification to be used for the CAN message ID.

0: Denotes the CAN2.0A specification, which uses an 11-bit CAN message ID.

1: Denotes the CAN2.0B specification, which uses a 29-bit CAN message ID.

MsgID

[out] Indicates the address of a variable used to retrieve the CAN message ID.

RTR

[out] Indicates the address of a variable used to retrieve the mode of the message, where:

0: Indicates that the message is a remote-transmit-request message

1: Indicates that the message is a normal message

Note that as this parameter will be set to 1, the *Data* parameter will be useless.

DataLen

[out] Indicates the address of a variable used to retrieve the length of the CAN message data.

Data

[out] Indicates the address of a variable used to retrieve the starting address of the data array used for a CAN message. The maximum length of the Data array is 8. Useless for the RTR parameter is 1.

UpperTime

[out] Indicates the address of a variable used to retrieve the upper timestamp of the received CAN message in increments interval of 4,294,967,295.5 milli-seconds. The maximum value is 4,294,967,295.

LowerTime

[out] Indicates the address of a variable used to retrieve the lower timestamp of the received CAN message in increments interval of 0.1 milli-seconds. The maximum value is 4,294,967,295.5 milli-seconds

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.16. CM200_SendCANMsg

Description

This function is used to transmit a CAN message on the CAN controller for a specified PISO-CM200U CAN port to the CAN network.

Syntax

```
C#
-----
Int16 CM200_SendCANMsg (
    Byte BoardNo,
    Byte Port,
    Byte Mode,
    UInt32 MsgID,
    Byte RTR,
    Byte DataLen,
    Byte[] Data,
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Mode

[in] Specifies the CAN Bus specification to be used for the CAN message ID.
0: Denotes the CAN2.0A specification, which uses an 11-bit CAN message ID.
1: Denotes the CAN2.0B specification, which uses a 29-bit CAN message ID.

MsgID

[in] Specifies the CAN message ID.

RTR

[in] Specifies the mode of the CAN message, where:

0: Indicates that the message is a remote-transmit-request message

1: Indicates that the message is a normal message

Note that as this parameter will be set to 1, the *Data* parameter will be useless.

DataLen

[in] Specifies the length of the CAN message data.

Data

[in] Specifies the starting address of the data array used for a CAN message.

The maximum length of the Data array is 8. Useless for the RTR parameter is 1.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.17. CM200_ClearSoftBuffer

Description

This function is used to clear the transmit/receive data buffer of the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#
-----
Int16 CM200_ClearSoftBuffer (
    Byte BoardNo,
    Byte Port
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.18. CM200_ClearTxSoftBuffer

Description

This function is used to clear the transmit data buffer of the CAN port on a specified PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_ClearTxSoftBuffer (  
    Byte BoardNo,  
    Byte Port  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.19. CM200_ClearRxSoftBuffer

Description

This function is used to clear the transmit data buffer of the CAN port on a specified PISO-CM200U series board.

Syntax

C#

```
Int16 CM200_ClearRxSoftBuffer (  
    Byte BoardNo,  
    Byte Port  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.2.20. CM200_ClearBufferStatus

Description

This function is used to clear the data buffer status of the CAN port on a specified PISO-CM200U series board.

Syntax

```
C#  
-----  
Int16 CM200_ClearBufferStatus (  
    Byte BoardNo,  
    Byte Port  
);
```

Parameters

BoardNo

[in] Specifies the number of the PISO-CM200U series board to be read. The valid range is 0 to 15.

Port

[in] Specifies the number of the CAN port, where:

- 1: CAN port 1
- 2: CAN port 2.

Return Value

If the function succeeds, the return value will be 0.

If the function fails, refer to Section 3.3 Error Code Definitions.

3.3. Error Code Definitions

The following are the definitions for the error codes that may be encountered while operating the PISO-CM200U.

Error Code (hexadecimal)	Error ID	Error Description
0x000	ERR_NO_ERR	No error
0x001	ERR_INIT_DRIVER_ERROR	An error occurred while initializing the driver
0x002	ERR_COMM_DRIVER_ERROR	An error occurred while communicating with the driver
0x003	ERR_DRIVER_UNINIT_ERROR	The driver is not initialized
0x004	ERR_ACTIVE_BOARD_ERROR	An error occurred while activating the board
0x005	ERR_BOARD_ALREADY_ACTIVE_ERROR	The board has already been activated (detected by driver)
0x006	ERR_BOARD_UNACTIVE_ERROR	The board is not activated
0x007	ERR_BOARD_NUMBER_ERROR	The board value exceeds the valid range
0x008	ERR_PORT_NUMBER_ERROR	The Port value exceeds the valid range (detected by driver)
0x009	ERR_DPRAM_ADDR_OVER_RANGE	The DPRAM address value exceeds the valid range
0x00A	ERR_FIRMWARE_MODE_ERROR	An error occurred while switching to Firmware mode (detected by driver)
0x00B	ERR_SIZE_OF_NAME_OVER_RANGE	The size of the Name exceeds the valid range
0x00C	ERR_SIZE_OF_DATA_OVER_RANGE	The size of the firmware data exceeds the valid range (detected by driver)
0x00D	ERR_ALLOC_DATA_BUFFER_ERROR	An error occurred while allocating the firmware data buffer
0x101	ERR_SYS_COMM_ACK_TIMEOUT	An timeout occurred while attempting to communicate with the firmware
0x102	ERR_SYS_BOARD_ALREADY_ACTIVE	The board has already been activated (detected by firmware)
0x103	ERR_SYS_SET_DATETIME_ERROR	An error occurred while setting the

		date and time for the firmware
0x104	ERR_SYS_FIRMWARE_MODE_ERROR	An error occurred while switching to Firmware mode (detected by firmware)
0x105	ERR_SYS_INSTALL_IRQ_ERROR	An error occurred while installing the system interrupt
0x106	ERR_SYS_ENABLE_PLX_INT_ERROR	An error occurred while enabling the system interrupt
0x107	ERR_SYS_DISABLE_PLX_INT_ERROR	An error occurred while disabling the system interrupt
0x108	ERR_SYS_PORT_NUMBER_ERROR	The Port value exceeds the valid range (detected by firmware)
0x109	ERR_SYS_CAN_CONFIG_ERROR	An error occurred while configuring the CAN port
0x10A	ERR_SYS_SET_CYCLIC_MSG_ERROR	An error occurred while setting the cyclic transmission message
0x10B	ERR_SYS_SET_DEBUG_MSG_ERROR	An error occurred while setting the debug message
0x10C	ERR_SYS_SOFTBUFF_IS_EMPTY	The CAN software buffer is empty
0x10D	ERR_SYS_SOFTBUFF_IS_FULL	An overflow occurred in the CAN software buffer
0x10E	ERR_SYS_USERISR_ALREADY_INSTALL	The user-defined interrupt has already been installed
0x10F	ERR_SYS_NO_DPRAM_COMMDAND	There was no response to the DPRAM command
0x110	ERR_SYS_DOWNLOAD_DATA_ERROR	An error occurred while downloading data to the firmware
0x111	ERR_SYS_GET_FW_INFO_ERROR	An error occurred while retrieving the firmware file information
0x112	ERR_SYS_SET_FW_INFO_ERROR	An error occurred while setting the firmware file information
0x113	ERR_SYS_NO_FW_EXIST_ERROR	No firmware exists
0x114	ERR_SYS_SIZE_OF_DATA_OVER_RANGE	The size of the firmware data exceeds the valid range (detected by firmware)

4. Firmware Upgrade

The following describes how to upgrade the firmware for the PISO-CM200U series board.

Step 1: Locate and install the “Firmware Update Tool” and firmware file.

The Firmware Update Tool software can be found on the Fieldbus CD at:
CD:\can\pci\piso-cm200u\software\tools

The Firmware Update Tool software can also be downloaded from the ICP DAS website at either of the following addresses:

ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/software/tools

or

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/software/tools

The firmware file can be found on the Fieldbus CD at:

CD:\can\pci\piso-cm200u\firmware

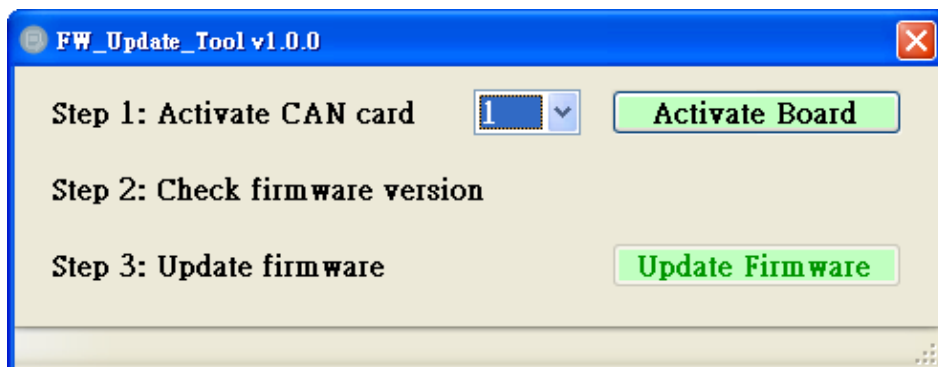
The firmware file can also be downloaded from the ICP DAS website at either of the following addresses:

ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/firmware

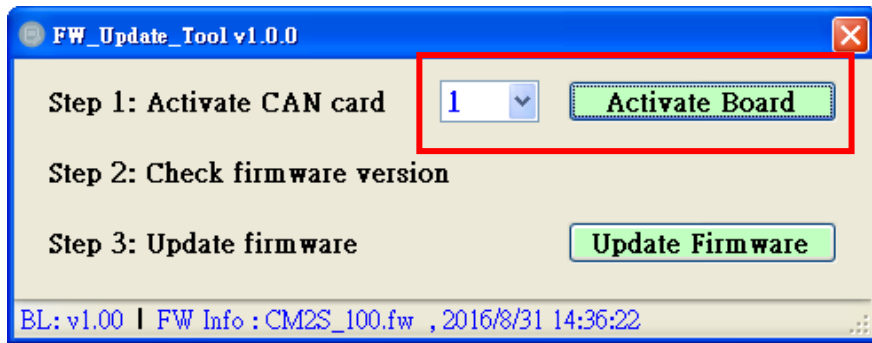
or

http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm200u/firmware

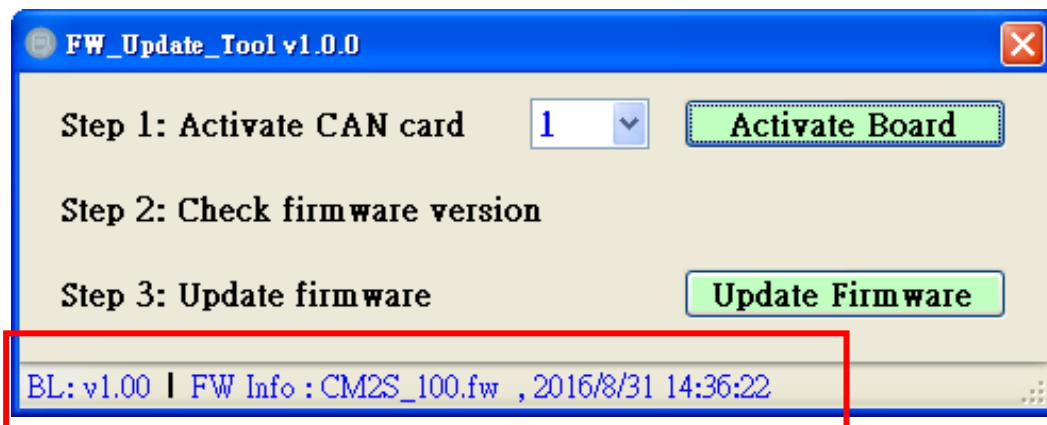
Step 2: The “Firmware Update Tool” requires the .NET Framework 3.5 component to be installed on the Host PC. Install the “Firmware Update Tool”. If the [.NET Framework 3.5](#) component does not exist on the Host PC, it will be automatically downloaded from the Microsoft web site and installed after executing the Firmware Update Tool installation file.



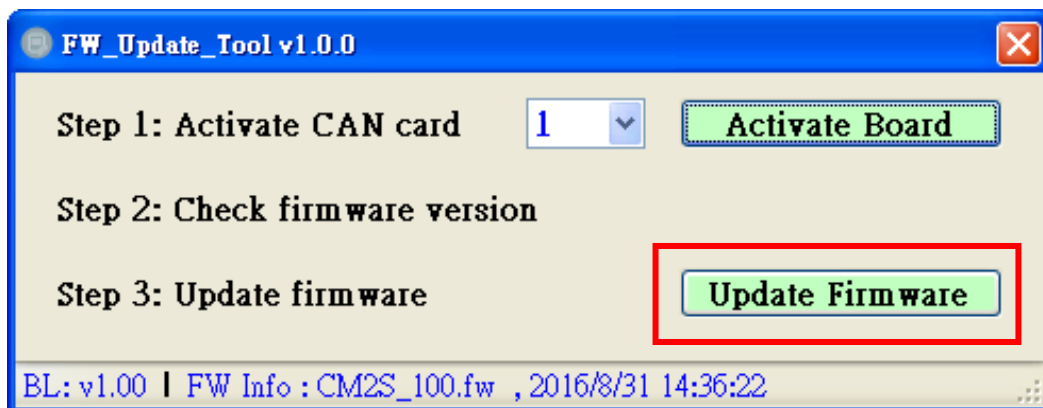
Step 3: Select the ID for the PISO-CM200U series board to be updated from the drop-down menu, and then click the “Activate Board” button to activate the selected board.

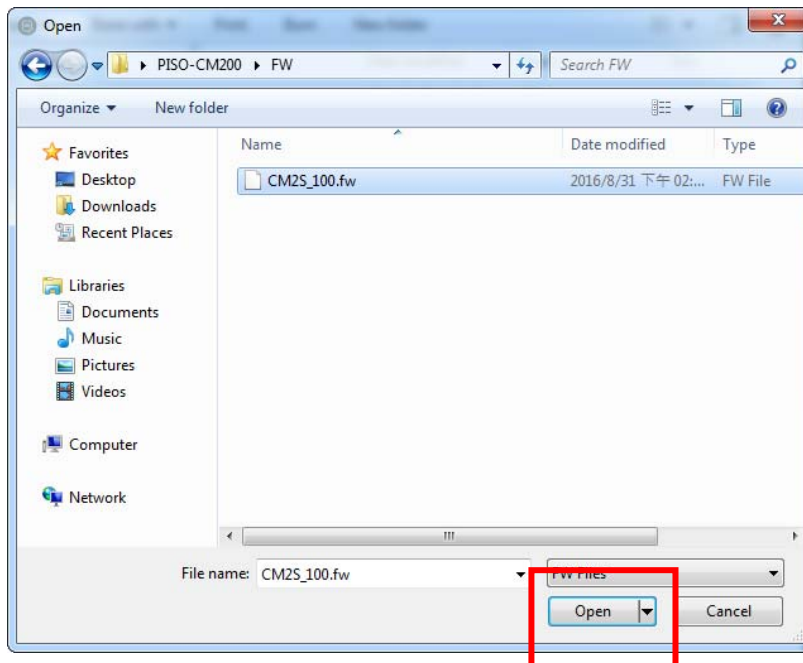


Step 4: In the status bar at the bottom of the dialog box, check the firmware information (FW_Info). In this example, “CS2S_xxx.fw” is the name of the firmware file, where “xxx” denotes the firmware version number. The build date and time of the file is also indicated.

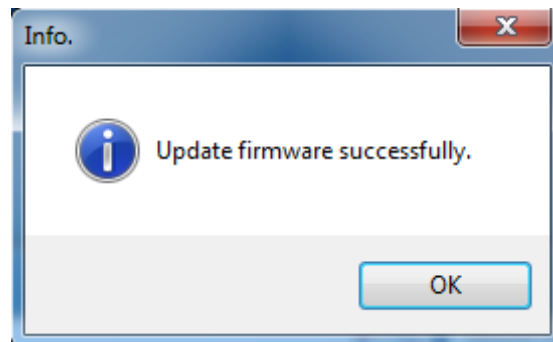


Step 5: Click the “Update Firmware” button to select the necessary firmware file and download it to the PISO-CM200U series board.





Step 6: After the firmware has been updated, a dialog will be displayed to indicate that the update was successful. Click the OK button to dismiss the dialog and then exit the “FW_Update_tool” to begin using the PISO-CM200U series board.



5. Appendix

5.1. EMI Ferrite Split/Snap-On Core



EMI Ferrite Split/Snap-On Core

Features

- Aimed to suppress low frequency noise generated by engine control units, inverters, and motors
- Split type
- Operation Temperature: -25°C ~ 75°C



Introduction

The split ferrite cable cores are designed to significantly reduce EMI/RFI on round cables.

The hinged plastic case surrounding the split core is designed to clamp onto the cable to provide a secure fixture of the ferrite onto the cable. The cores can be retrofitted onto existing installations or used in post-assembly operations on the data and power cables of electronic equipment. Ferrite cores are important for ensuring strong electronic signals through cables in environments where EMI or RFI can be an issue.

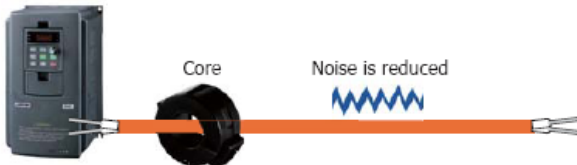
Applications

RS-232, RS-422, RS-485, CAN bus, FRnet, PROFIBUS, Ethernet, USB, AC/DC Power line..etc

Inverter



Inverter

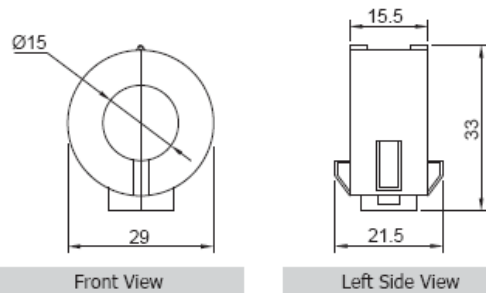


Specifications

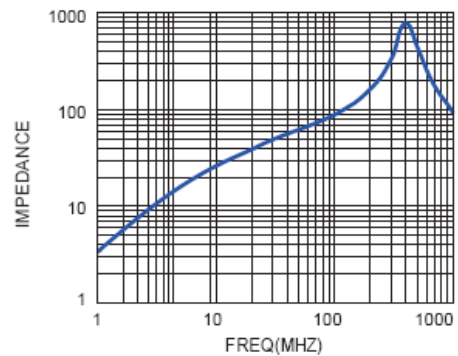
Mechanical

Max. Cable Diameter	Ø15 mm
Material Type	Board Band Material
Additional Description	Plastic Case
Case Color	Black

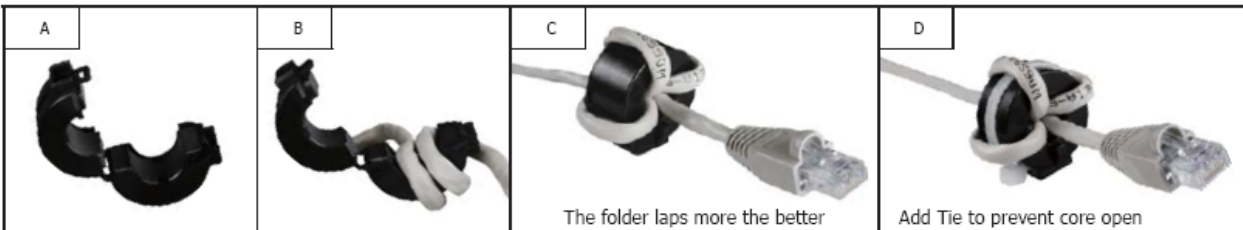
Dimensions (Units: mm)



Characteristic



Installation



Clip-on Ferrite Core Installation