PCM-CAN200-D
with Windows CE

User's Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to 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.

Trademark

The names used for identification only maybe registered

trademarks of their respective companies.

Tables of Content

1. Overview

The CAN (Controller Area Network) is a serial communication protocol, which efficiently supports distributed real-time control with a very high level of security. It is especially suited for networking "intelligent" devices as well as sensors and actuators within a system or sub-system. In CAN networks, there is no addressing of subscribers or stations in the conventional sense, but instead prioritized messages are transmitted. As a stand-alone CAN controller, the PCM-CAN200 can represent a CAN solution on a high quality PCI104 hardware in industrial environment compliant with CAN 2.0A and CAN 2.0B specification. The PCM-CAN 200 has 2 independent CAN bus communication ports with 9-pin D-sub connector, and has the ability to cover a wide range of CAN applications.

1.1. Feature

- PCI104 compliant
- 2500Vrms photo-isolation protection
- Two independent CAN communication ports
- Compatible with CAN specification 2.0 parts A and B
- On-board optical isolation protection
- Programmable transfer-rate up to 1 Mbps
- Jumper select 120Ω terminator resistor for each port
- Direct memory mapping to the CAN controllers
- 9-pin D-sub connector
- 3KV galvanic isolation
- 2 independent CAN channels
- Driver supported for Windows CE 5.0

1.2. Specifications

- CAN controller: Phillips SJA1000T.
- CAN transceiver: Phillips 82C250/251.
- Signal support: CAN_H, CAN_L.
- CAN controller frequency :16 MHz
- Connector: 9-pin D-sub male connector.
- Isolation voltage: 2500Vrms.
- Power requirements:

  CAN400: 5V@640mA

  CAN200: 5V@380mA

  CAN400U:5V@300mA

  CAN200U:5V@164.2mA
- Environmental:

  Operating temp: 0~60℃

  Storage temp: -20~80℃

  Humidity: 0~90% non-condensing

  Dimensions: 130mm X 110mm

1.3. Package Checklist

Besides this manual, the package includes the following items:

- □ PCM-CAN200 card
- □ Software CD ROM

It is recommended that users read the release note first. All the important information needed will be provided in the release note as follows:

- □ Where you can find the software driver, API and demo programs.
- □ How to install driver.
- □ FAQ's and answers.

Attention !

If any of these items are missing or damaged, please contact your local field agent. Keep aside the shipping materials and carton in case you want to ship or store the product in the future.

2. **Install Driver**

This installation guide describes the procedures to install the PCM-CAN200D in Mircrosoft Windows CE.NET (ver 5.0) operation system on x86 systems.

The developer uses Platform Builder 5.0 or greater and has created a Win CE image for the PCM-CAN200D. This image should be created with the MAXALL configuration under CE 5.0 or an appropriate platform configuration under CE.NET. Users can find the driver in the path of "\CAN\PCI\PISO-CAN200_400\wince\PCM-CAN200\Driver\" in the Fieldbus_CD.

To embed PCMCAN_CE.dll in a Windows CE image follow the following steps.

1. Preparation prior to installation:
   Copy driver file into the your platform BSP "files" folder.
   Driver file:
   CD:\CAN\PCI\PISO-CAN200_400\wince\PCM-CAN200\Driver\PCMCAN200_CE.dll
   Platform BSP path:
   WINCE\PLATFORM\MyBSP\files

2. Install PCM-CAN200D Board Driver for PCI Bus
   The platform setting must meet the following requirements.
   A. Edit the configuration file project.bib, adding the following line:
      PCMCAN200_CE.dll $(_FLATRELEASEDIR)\ PCMCAN200_CE.dll NK SH

   B. Edit the configuration file project.reg:
      [HKEY_LOCAL_MACHINE\Drivers\BuiltIn\PCI\Template\ICPDASCAN]
      "Class"=dword:6
      "SubClass"=dword:80
      "DeviceArrayIndex"=dword:1
      "DeviceID"=multi_sz:"9030
      "VendorID"=multi_sz:"10B5"
      "Dll"="PCMCAN200_CE.dll"
      "FrogIF"=dword:0
      "Prefix"="ICP"
      "Order"=dword:0

3. **API Function Definition and Description**

The API and Demo of PCM-CAN200 can be used in Windows CE. Users can find them in the path of "\CAN\PCI\PISO-CAN200_400\wince\PCM-CAN200\Demo\" in the Fieldbus_CD. And users must use the EVC to develop the program.

All the functions provided in the PCM-CAN200 are listed in the following table and detailed information for every function is presented in the following sub-section. However, in order to make the descriptions more simplified and clear, the attributes for the both the input and output parameter functions are given as **[input]** and **[output]** respectively, as shown in following table.

Keyword	Set parameter by user before calling this function?	Get the data from this parameter after calling this function?
[input]	Yes	No
[output]	No	Yes

Table 4.1 DLL function definition

Function definition	Page
WORD CAN_GetDllVersion();	26
int CAN_TotalBoard();	26
int CAN_Reset(BYTE BoardNo, BYTE Port);	32
int CAN_Init(BYTE wBoardNo, BYTE Port);	33
int CAN_Config(BYTE BoardNo, BYTE Port, ConfigStruct *CanConfig);	34
int CAN_SendMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket);	46
int CAN_ReceiveMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket);	50

Table 4.2 Interpretation of the return code

Return Code	Error ID	Comment
0	CAN_NoError	OK
1	CAN_DriverError	Driver error
2	CAN_ActiveBoardError	This board can't be activated.
3	CAN_BoardNumberError	The board number exceeds the maximum board number (7).
4	CAN_PortNumberError	The port number exceeds the maximum port number.
5	CAN_ResetError	CAN chip hardware reset error
6	CAN_SoftResetError	CAN chip software reset error
7	CAN_InitError	CAN chip initiation error
8	CAN_ConfigError	CAN chip configure error
9	CAN_SetACRError	Set to Acceptance Code Register error
10	CAN_SetAMRError	Set to Acceptance Mask Register error
11	CAN_SetBaudRateError	Set Baud Rate error
17	CAN_TransmitIncomplete	Previously transmission is not yet completed
20	CAN_ReceiveError	Receive fail

5.1. CAN_GetDllVersion

- **Description:**

  Obtain the version information of PISOCAN.dll driver.

- **Syntax:**

  WORD CAN_GetDllVersion(viod)

- **Parameter:**

  None

- **Return:**

  DLL version information. For example: If 101(hex) is return, it means driver version is 1.01.

5.2. CAN_TotalBoard

- **Description:**

  Obtain the amount of all PISO-CAN or PCM-CAN boards installed in the PCI bus.

- **Syntax:**

  int CAN_TotalBoard(void)

- **Parameter:**

  None

- **Return:**

  Return the amount of all board.

5.3. CAN_Init

- **Description:**

  Initiate CAN controller.

- **Syntax:**

  int CAN_Init(BYTE BoardNo, BYTE Port)

- **Parameter:**

  BoardNo: [input] PISO-CAN or PCM-CAN board number (0~3).

  Port: [input] CAN port number (1~2)

- **Return:**

  CAN_NoError: OK

  CAN_PortNumberError: Port number is not correct.

5.4. CAN_Reset

- **Description:**

  To close the CAN Port.

- **Syntax:**

  int CAN_Reset(BYTE BoardNo, BYTE Port)

- **Parameter:**

  BoardNo: [input] PISO-CAN or PCM-CAN board number (0~3).

  Port: [input] CAN port number (1~2)

- **Return:**

  CAN_NoError: OK

  CAN_PortNumberError: Port number is not correct.

5.5. CAN_Config

- **Description:**

  Configure CAN controller. After calling this function, the CAN controller will enter operating mode.

- **Syntax:**

  int CAN_Config(BYTE BoardNo, BYTE Port,ConfigStruct *CanConfig);

- **Parameter:**

  BoardNo: [input] PISO-CAN or PCM-CAN board number (0~3).

  Port: [input] CAN port number (1~2)

  *ConfigStruct: [input] The point of structure for ConfigStruct is defined as following,

```
typedef struct config
{
        BYTE AccCode[4];
        BYTE AccMask[4];
        BYTE BaudRate;
} ConfigStruct;
```

  AccCode[4]: Acceptance code for CAN controller.

  AccMask[4]: Acceptance mask for CAN controller.

  BaudRate: 1→125Kbps, 2→250Kbps, 3→500Kbps, 4→800Kbps, 5→1Mbps.

Return:

CAN_NoError: OK

CAN_SetACRError: Set Acceptance code to CAN controller error

CAN_SetAMRError: Set Acceptance mask to CAN controller error

CAN_SetBaudRateError: Set baud rate to CAN controller error

5.6. CAN_SendMsg

- **Description:**

  Send a CAN message immediately.

- **Syntax:**

  int CAN_SendMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket)

- **Parameter:**

  BoardNo: [input] PISO-CAN or PCM-CAN board number (0~3)

  Port: [input] CAN port number (1~2)

  *CanPacket: [input] The point of structure for CanPacket is defined as following,

```
typedef struct packet
{
    BYTE    mode;
    DWORD id;
    BYTE    rtr;
    BYTE    len;
    BYTE    data[8];
} PacketStruct;
```

  mode: 0→ 11-bit identifier, 1 → 29-bit identifier.

  id: Identifier

  rtr: Remote transmission request

  len: Data length

  data[8]: data byte

- **Return:**

  CAN_NoError: OK

  CAN_TransmitIncomplete: Transmission is not yet completed.

5.7. CAN_ReceiveMsg

- **Description:**

  The CAN_ReceiveMsg uses “Polling” to receive the CAN Message. So obtain receive message from CAN controller’s RXFIFO.

- **Syntax:**

  int CAN_ReceiveMsg(BYTE BoardNo, BYTE Port, PacketStruct *CanPacket)

- **Parameter:**

  BoardNo: [input] PISO-CAN or PCM-CAN board number (0~3)

  Port: [input] CAN port number (1~2)

  *CanPacket: [output] The structure for CanPacket is defined below,

```
typedef struct packet
{
    BYTE    mode;
    DWORD id;
    BYTE    rtr;
    BYTE    len;
    BYTE    data[8];
} PacketStruct;
```

  mode: 0→ 11-bit identifier, 1 → 29-bit identifier.

  id: Identifier

  rtr: Remote transmission request

  len: Data length

  data[8]: data byte

- **Return:**

  CAN_NoError: OK

  CAN_ReceiveError: Receiver fail.