



JetBox 8152 XPe CANBus

User Manual

www.korenix.com

Copyright Notice

Copyright© 2009 Korenix Technology Co., Ltd.

All rights reserved.

Reproduction without permission is prohibited.

Information provided in this manual is intended to be accurate and reliable. However, the original manufacturer assumes no responsibility for its use, or for any infringements upon the rights of third parties that may result from its use. The material in this document is for product information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, Korenix assumes no liabilities resulting from errors or omissions in this document, or from the use of the information contained herein.

Korenix reserves the right to make changes in the product design without notice to its users.

Acknowledgments

Korenix is a registered trademark of Korenix Technology Co., Ltd.

All other trademarks or registered marks in the manual belong to their respective manufacturers.

Table of Content

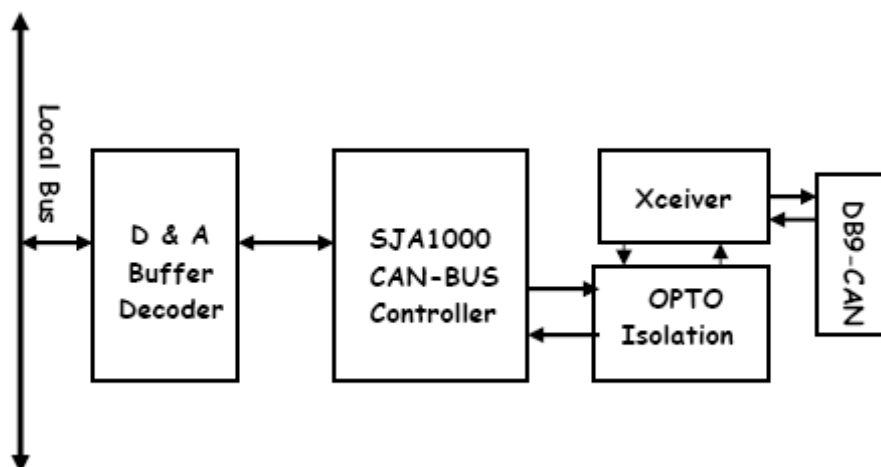
Copyright Notice	2
Acknowledgments.....	2
Table of Content	3
Chapter 1 Introduction	4
Chapter 2 Hardware Configuration.....	6
2-1 Pin Assignment.....	6
2-2 Jumper Setting: JP6	6
Chapter 3 Software Configuration	7
3-1 Installation.....	7
3-2 Program Start	9
Chapter 4 Korenix Library Reference	11
4-1 How to Use CanPort Library	11
4-2 Functions.....	13
4-3 Structure	19
Chapter 5 Appendix	20
5-1 Baud Rate Table.....	20
5-2 Error Code Table.....	21
5-3 Notes	22
5-4 Customer Service	23

Chapter 1 Introduction

The JetBox 8152 has two ports for I/O communications, One RS-232/422/485 port and one CANBUS port. The CAN (Controller Area Network) is a serial bus system especially suited for networking "intelligent" I/O devices as well as sensors and actuators within a machine or plant. Characterized by its multi-master protocol, real-time capability, error correction, high noise immunity, and the existence of many different silicon components, the CAN serial bus system, originally developed by Bosch for use in automobiles, is increasingly being used in industrial automation.

CANbus

This section describes how to program and use the CANBUS. It provides a description of the I/O memory map of the chip and discussion of the internal registers to aid you in programming your CAN controller chip.



Defined Memory Mapping and Interrupt

The CANBUS occupies 2 bytes of memory space. You can set the base address and access to the internal resources of the SJA1000 CAN controller chip. The SJA1000 chip access is multiplexed in such a way that the host must first write to **300h** the internal address of the CAN chip and after that perform a write to address **301h** with the actual data to be written into the desired memory location. Address **302h** is a hardware-reset function of the SJA1000. Performing a read or write to this address

will cause a hardware reset to the CAN controller. You may need to reset the chip in case of an unrecoverable error in the CAN controller chip. And your can use interrupt the main processor when a message is received or transmitted if interrupts are enabled on the JetBox 8152. By using interrupts you can write powerful code to CAN.

Description	Factory Setting
Base Address	300H
Data Of Address	301H
Hardware Reset Of SJA1000 Chips	302H
Interrupt Require Quest	11

Example Programming

Write 300H to the CAN controller Control byte located in the on-chip address 0.

The Example is listed below:

Outportb (0x300, 0x00) : Write CAN Address 0 (Control Register)

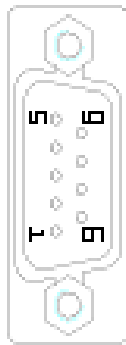
Outportb (0x301, 0x78) : Write Data of CAN Address 0 (Control Register)

And please see "SJA1000.pdf" for further information of the SJA1000 chip.


Chapter 2 Hardware Configuration

2-1 Pin Assignment

The CANBUS is use DB9 standard connector. The following tables show the CANBUS signal connections of this connector.

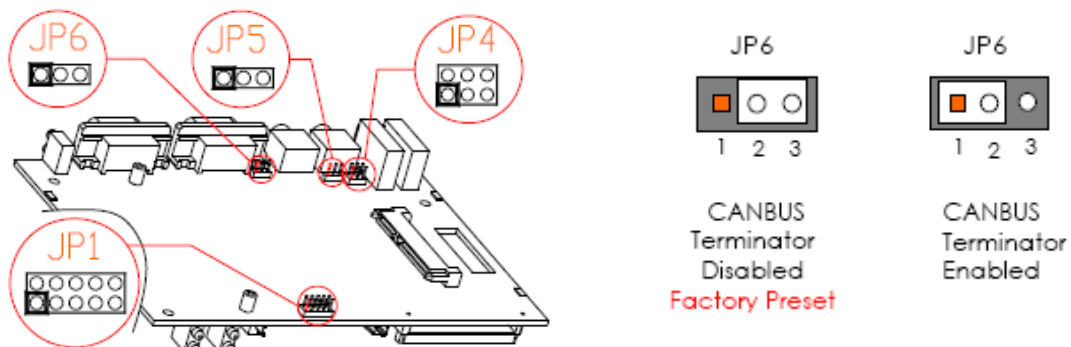



DB-9 CANBUS	CANBUS Signal	CANBUS Description
1	N.C	-
6	N.C	-
2	CAN-L	Dominant Low
7	CAN-H	Dominant High
3	CAN-Ground	Isolated Ground
8	N.C	-
4	N.C	-
9	N.C	-
5	Ground	Digital Ground
Case	Case Ground	

 Note 1: The CANBUS DB9-pin out conforms to the ISO 11898/2 standard

2-2 Jumper Setting: JP6

JP6: CANBUS Terminal Resistor Selection



 Note 2: The JP6 is the CANbus termination jumper. Only two termination jumpers should be closed at the endpoints of the CANbus. Value Terminator Resistor (120 Ω). **The minimum speed is 20k bps. The maximum speed is 1M bps.** But when CANBUS terminator is **disabled**, the maximize speed of CANBUS is **125k bps**. If you want to use high speed (1M bps), please enable terminator.

Chapter 3 Software Configuration

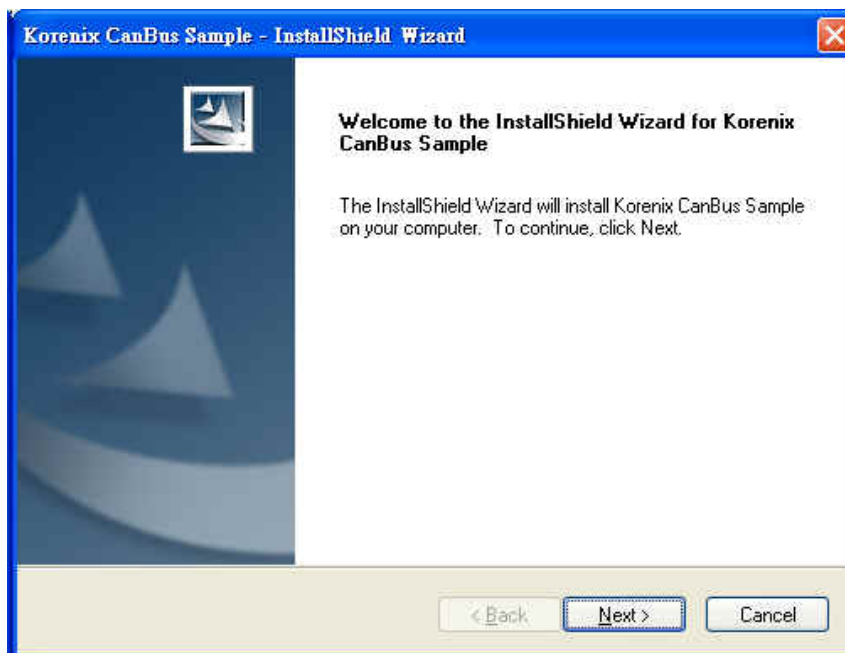
Under Windows, a driver is needed to access the Can Port and to provide the interface for Windows applications. Beside the mentioned device driver, the CANBus Sample application for Windows can also be set up.

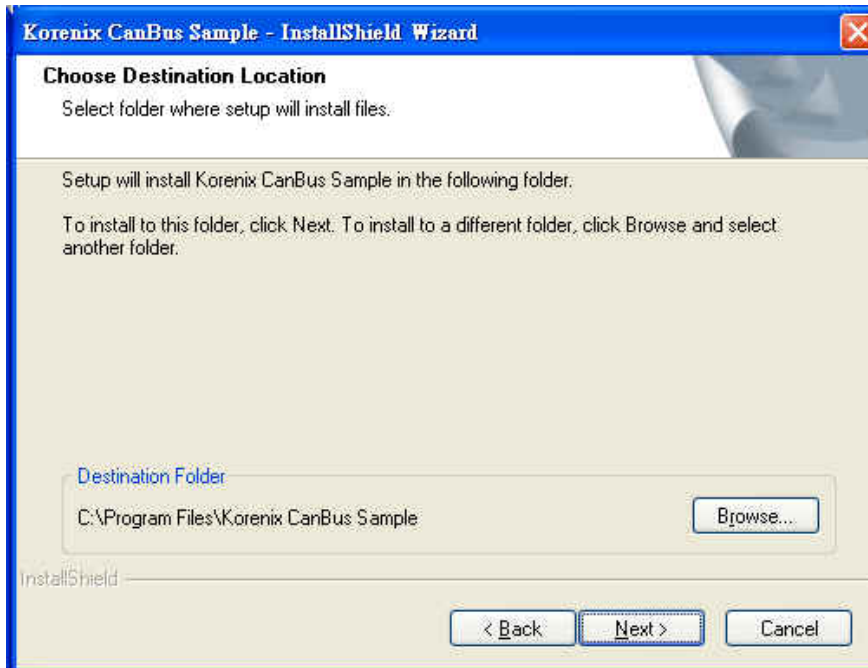
3-1 Installation

❑ **Do the following steps to setup the driver**

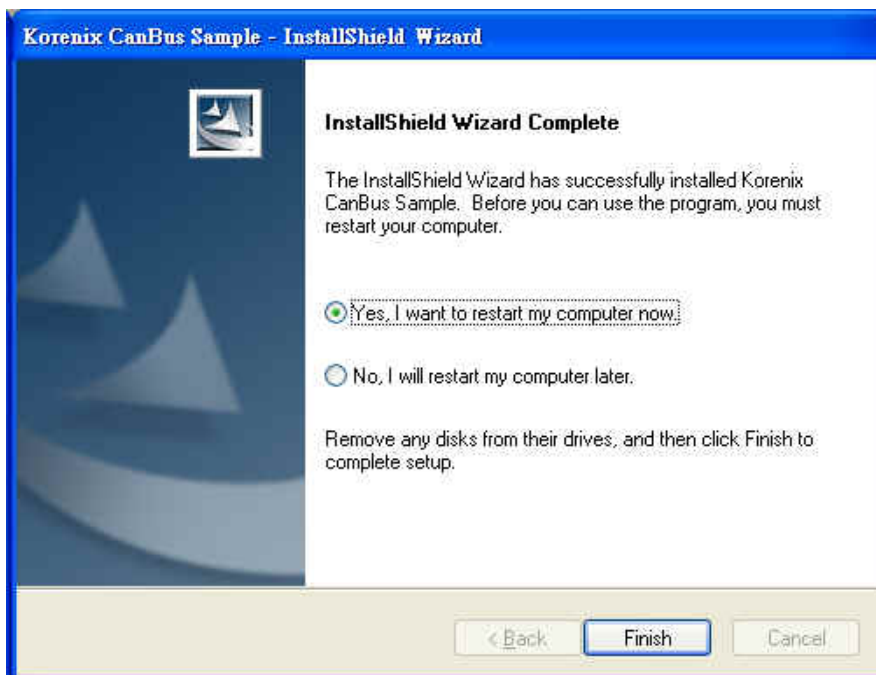
1. Make sure that you are logged in as the user with administrator privileges.
2. Click on **setup.exe** now. The setup program for the driver is executed.

And follow the instructions of the set up program.





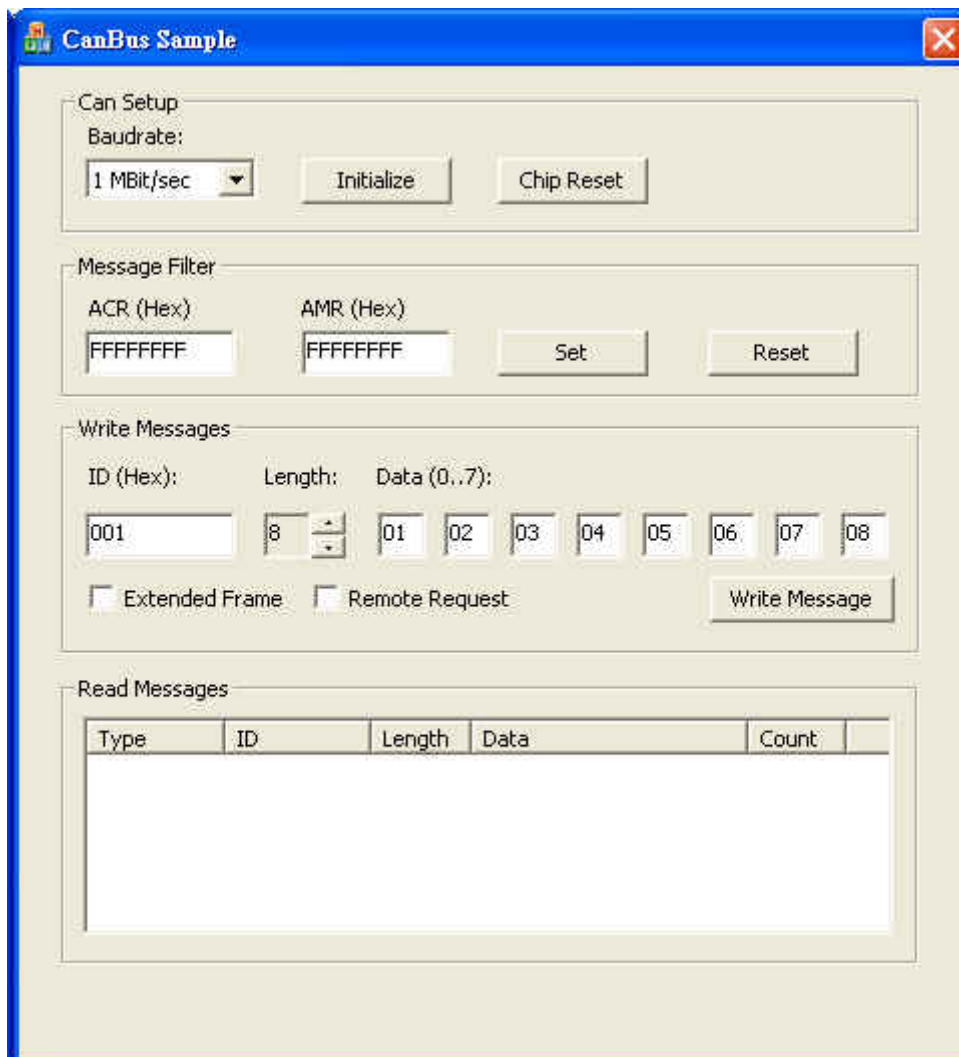
3. Please reboot the system when install finished.



3-2 Program Start

CANbus Sample Program

CANbus Sample for Windows is a simple CAN monitor for viewing and transmitting CAN messages.



❑ Initialize

Select which baud rate you want to connect to CANbus device and click initialize button.

❑ **Chip Reset**

Hardware reset of SJA1000 chip. You may need to reset the chip in case of an unrecoverable error in the CAN controller chip.

❑ **Message Filter**

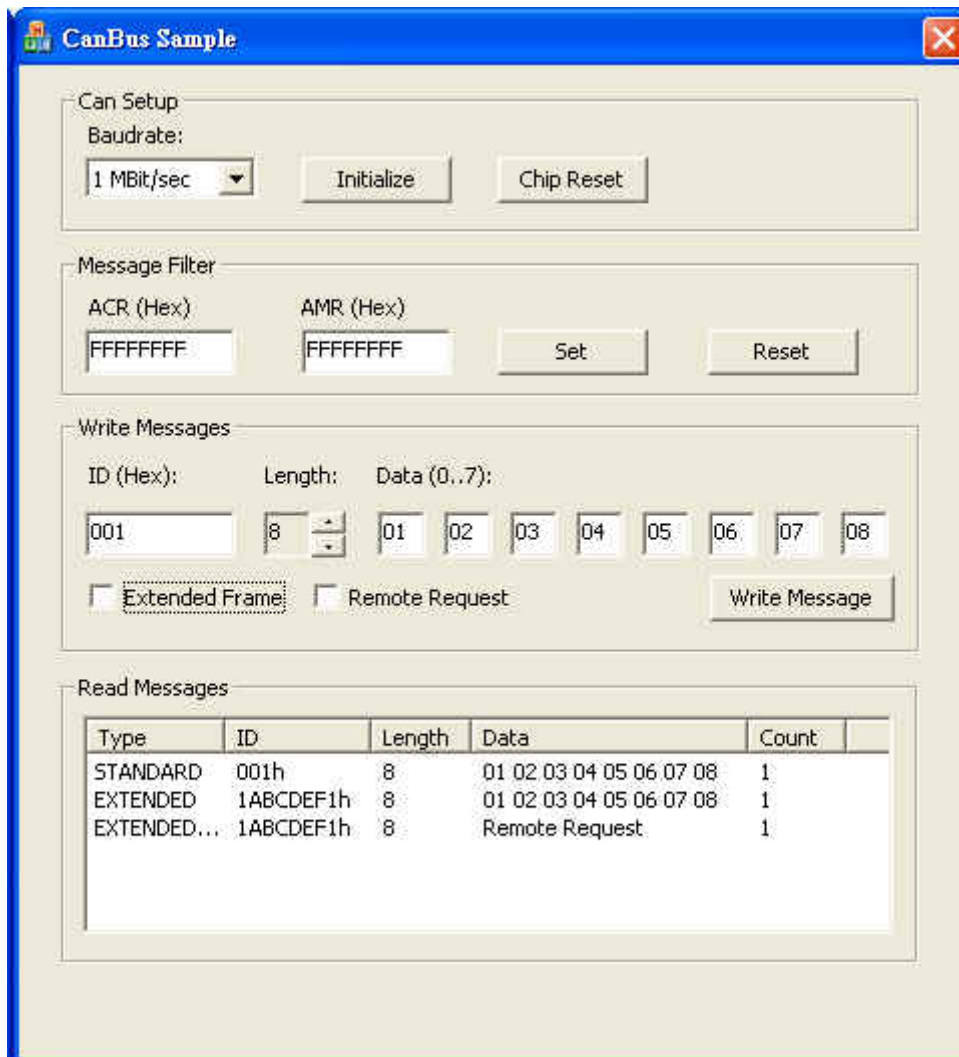
You can set up the ACR and AMR register to decide which message you want to receive it. Click "Reset" to use default value. (ACR = 0, AMR = 0xFFFFFFFF)

❑ **Write Messages**

In this filed, you can sent CAN message to your CAN device. It also support CAN 2.0B (Extended Frame) and RTR frame.

❑ **Read Messages**

The CanBus Sample will receive can message automatically when you click **Initialize** button.



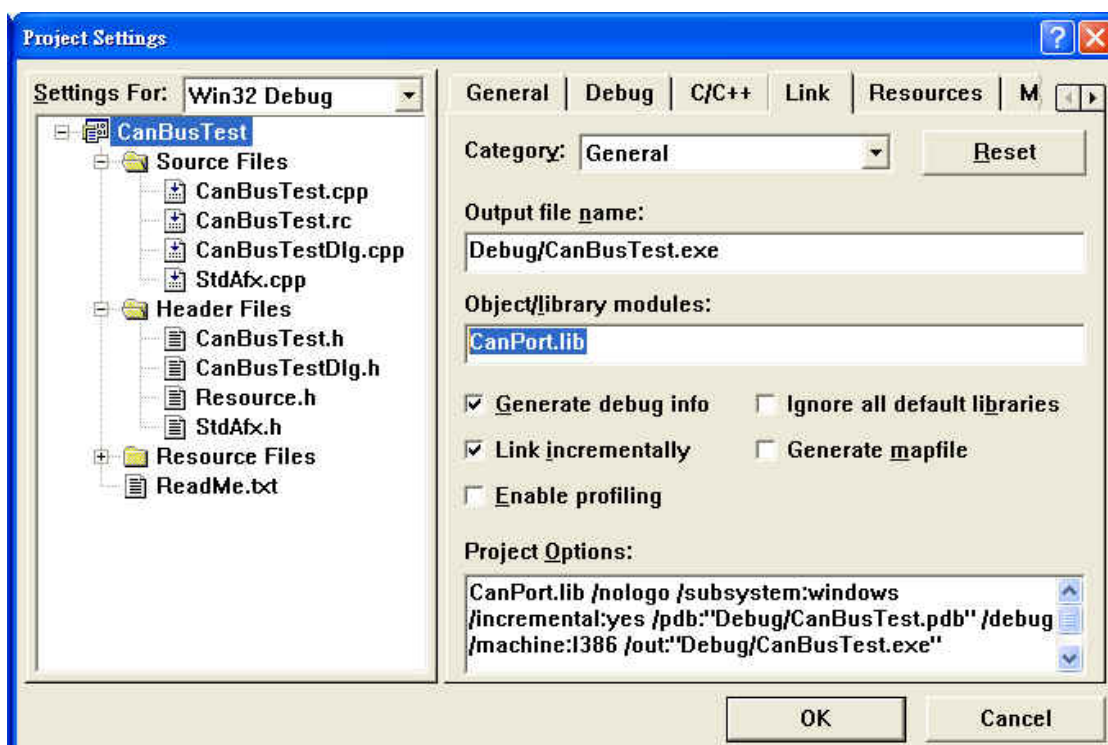
Chapter 4 Korenix Library Reference

This section shows how to use Korenix CANbus Library to develop your program. It is for VC6 library. These files are in the **C:\Program Files\Korenix CanBus Sample**.

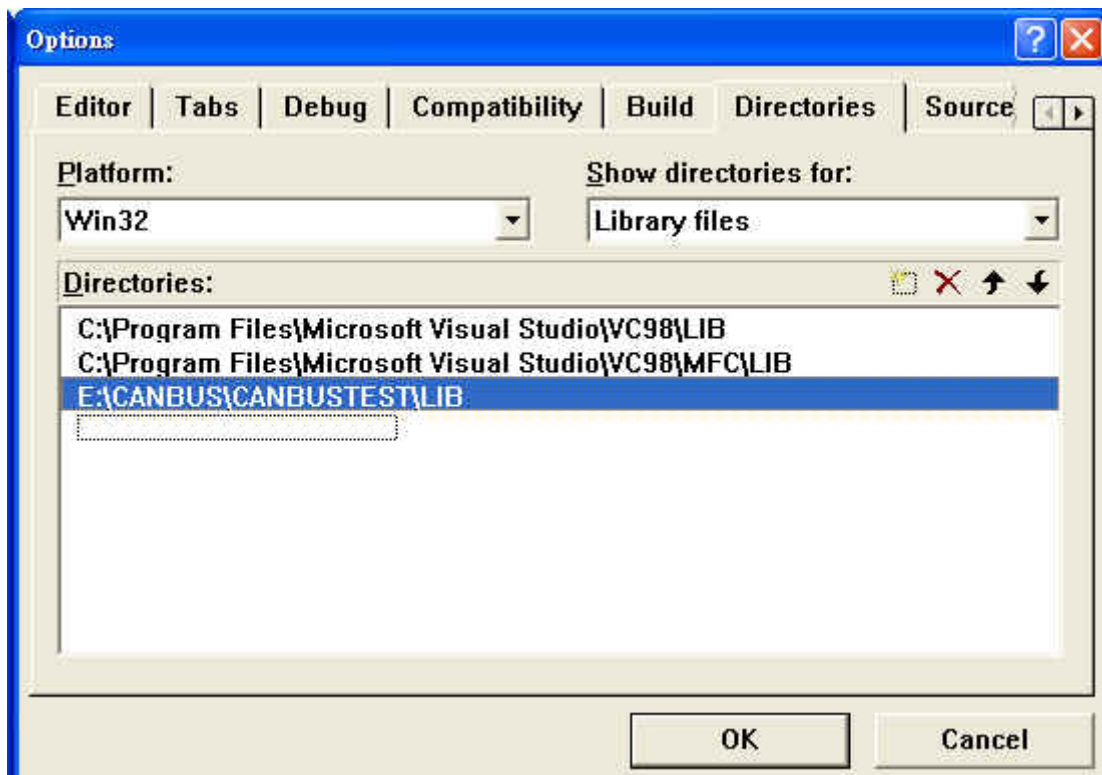
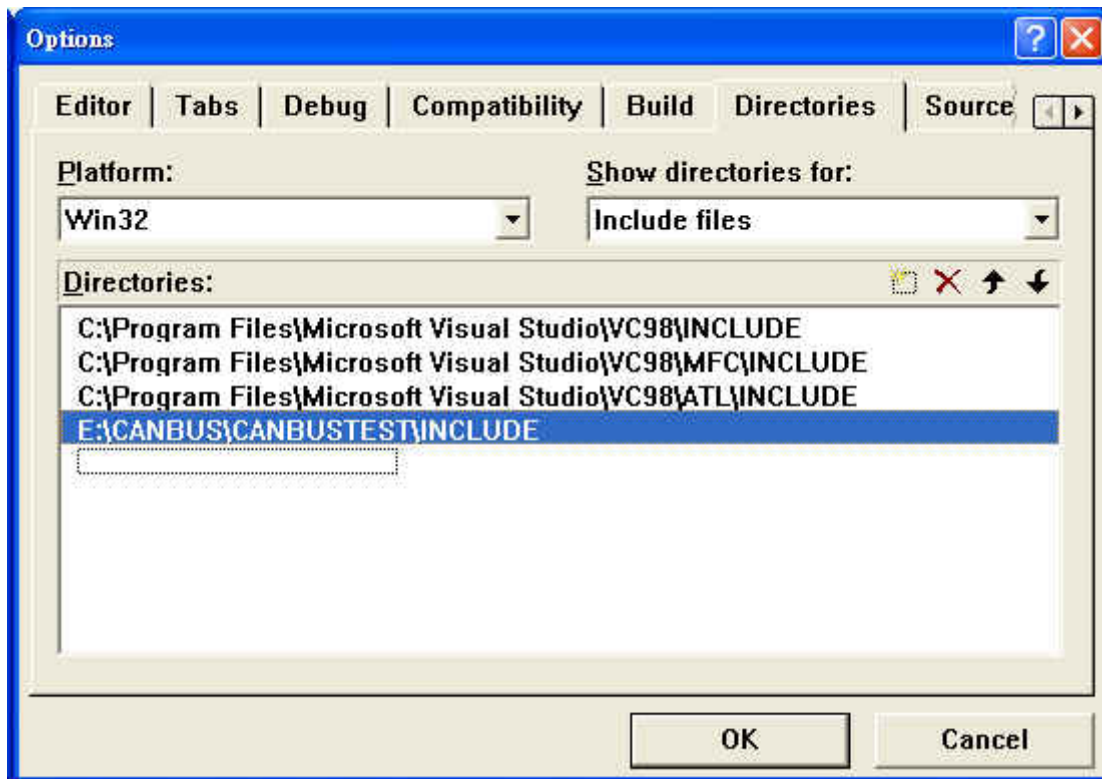
4-1 How to Use CanPort Library

When developing a program using the CanPort Library with Visual C++, you must do the following steps:

1. Add the CanPort Library C/C++ include files (CanPort.h) to your program:
`#include "CanPort.h"`
2. Add the export library file (CanPort.lib) to your link library list.



3. Make sure that CanPort.h and CanPort.lib are in the search path.



4-2 Functions

The CanPort library provides the following functions

❑ **Init_Can**

```
CANPORT_API void APIENTRY Init_Can(BYTE BTR0, BYTE BTR1)
```

Parameters

BTR0

BUS TIMING REGISTER 0

BTR1

BUS TIMING REGISTER 1

This function sets configuration parameters to initialize the CAN controller. Configuration parameters include baud rate. Valid Baud rate codes can be taken from the [Baud Rate Table](#).

This function will also set up the Interrupt Enable Register, Acceptance Code Register, Acceptance Mask Register and Output Control Register.

❑ **Can_Chip_Reset**

```
CANPORT_API void APIENTRY Can_Chip_Reset(void)
```

This function resets the CAN controller to default state. The transmitting and receiving of messages will be canceled, and messages in the driver buffer will be cleared as well.

❑ **Can_Send_Message**

```
CANPORT_API void APIENTRY Can_Send_Message(CANMsg *MsgToSend)
```

Parameters

MsgToSend

Message to transmit. Please refer to [Structure](#) for details.

❑ Can_Receive_Message

CANPORT_API void APIENTRY Can_Receive_Message(CANMsg *MsgToRead)

Parameters

MsgToRead

Returns a CAN message from the receive queue.


❑ Can_Status_Report

CANPORT_API BYTE APIENTRY Can_Status_Report(void)

Return

Status of CAN controller


Get the current status of the CAN controller.

 **Note 3: More information of Status is given in the data sheet of SJA 1000 in section 6.4.5 Status Register (SR).**

❑ Can_Set_Filter

CANPORT_API void APIENTRY Can_Set_Filter(DWORD dwACR, DWORD dwAMR)

This function sets the Acceptance Code and Acceptance Mask of the CAN controller. The CAN controller must set to reset mode when calling the function.

 **Note 4: More information of Status is given in the data sheet of SJA 1000 in section 6.4.15 Acceptance Filter.**

Parameters

dwACR

Acceptance Code Register

dwAMR

Acceptance Mask Register

Example

The following example shows the parameter values for *dwACR* and *dwAMR* in order to accept only the Data messages of standard frame in the range 110h to 113h.

dwACR:	001 0001 0000
dwAMR:	000 0000 0011
Valid IDs:	001 0001 00xx
ID 110h:	001 0001 0000
ID 111h:	001 0001 0001
ID 112h:	001 0001 0010
ID 113h:	001 0001 0011

So the value of *dwACR* is 0x221FFFFFF and the value of *dwAMR* is 0x007FFFFF.

Can_Reset_Filter

CANPORT_API void APIENTRY Can_Reset_Filter(void)


This function reset the Acceptance Code and Acceptance Mask Register of the CAN to default value (It means accept all CAN message).

Can_Interrupt_Status

CANPORT_API BYTE APIENTRY Can_Interrupt_Status(void)

Return

The value of Interrupt Register

 **Note 5: More information of Status is given in the data sheet of SJA 1000 in section 6.4.6 Interrupt Register (IR).**

❑ Can_Mode_Set


CANPORT_API void APIENTRY Can_Mode_Set(BYTE bMode)

Parameters

bMode

operation mode

Description	Value
SLEEP_MODE	0x10
ACCEPT_FILTER_MODE	0x08
SELF_TEST_MODE	0x04
LISTEN_ONLY_MODE	0x02
RESET_MODE	0x01
NORMAL_MODE	0x00

 **Note 6:** More information of Status is given in the data sheet of SJA 1000 in section 6.4.3 Mod Register (MOD).

❑ Can_Set_BTR

CANPORT_API void APIENTRY Can_Set_BTR(BYTE BTR0, BYTE BTR1)

Parameters

BTR0

BUS TIMING REGISTER 0

BTR1

BUS TIMING REGISTER 1

This function only sets the baud rate to CAN controller.

❑ Can_Set_Command


CANPORT_API void APIENTRY Can_Set_Command(BYTE bCmd)

Parameters

bCmd

Command mode

Description	Value
CLEAR_DATA_OVERRUN	0x08
RELEASE_RECEIVE_BUFFER	0x04
ABORT_TRANSMISSION	0x02
TRANSMISSION_REQUEST	0x01


 **Note 7:** More information of Status is given in the data sheet of SJA 1000 in section 6.4.4 Command Register (CMR).


❑ Can_Get_ECR

CANPORT_API BYTE APIENTRY Can_Get_ECR(void)

Return

The value of Error Code Capture Register.

 Note 8: Please refer to [Error Code Table](#) for details.

 Note 9: More information of Status is given in the data sheet of SJA 1000 in section 6.4.9.

Can_Get_ALC

CANPORT_API BYTE APIENTRY Can_Get_ALC(void)

Return

The value of Arbitration Lost Capture Register.


Can_Get_EWL

CANPORT_API BYTE APIENTRY Can_Get_EWL(void)

Return

The value of Error Warning Limit Register.

The error warning limit can be defined within this register. The default value (after hardware reset) is 96.

 Note 10: More information of Status is given in the data sheet of SJA 1000 in section 6.4.10.


Can_Get_TXERROR_COUNTER

CANPORT_API BYTE APIENTRY Can_Get_TXERROR_COUNTER(void)

Return

The value of transmit error.

The TX error counter register reflects the current value of the transmit error counter.

 **Note 11:** More information of Status is given in the data sheet of SJA 1000 in section 6.4.12.


❑ Can_Get_RXERROR_COUNTER

CANPORT_API BYTE APIENTRY Can_Get_RXERROR_COUNTER(void)

Return

The value of receive error.

The RX error counter register reflects the current value of the receive error counter.

 **Note 12:** More information of Status is given in the data sheet of SJA 1000 in section 6.4.11.

4-3 Structure

The CanPort API defines the following structures

CANMsg : Defines a CAN message

```
typedef struct {  
    DWORD ID;  
    BYTE  MSGTYPE;  
    BYTE  LEN;  
    BYTE  DATA[8];  
} CANMsg;
```

- ID
11/29-bit CAN identifier.
- MSGTYPE
Bit mask indicating the type of the message. Several message types can be combined.

Identifier	Value	Description
MSGTYPE_STANDARD	00h	Data Frame. CAN message with data contents and an 11-bit CAN ID.
MSGTYPE_RTR	01h	Remote Transmit Request (RTR).
MSGTYPE_EXTENDED	02h	Data Frame. CAN message with data contents according to CAN 2.0B standard (29-bit CAN ID).

- **LEN**
Number of data bytes in a data message (Data Length Code).
- **DATA**
Data bytes of a CAN message. The size can be 0 to 8 bytes.

Chapter 5 Appendix

5-1 Baud Rate Table

Baud Rate (bps)	Predefined BTR0 BTR1	BTR0	BTR1
5K	BAUD_5K	7F	7F
10K	BAUD_10K	67	2F
20K	BAUD_20K	53	2F
50K	BAUD_50K	47	2F
100K	BAUD_100K	43	2F
125K	BAUD_125K	03	1C
250K	BAUD_250K	01	1C
500K	BAUD_500K	00	1C
1000K	BAUD_1M	00	14

The following Baud Rates are also common:

Baud Rate (bps)	BTR0	BTR1
33.33K	1D	14
47.6K	14	14
83.33K	4B	14
95.23K	C3	4E
800K	00	16



Note 13: More information on setting the bit rate is given in the data sheet of SJA 1000 in section 6.5.

5-2 Error Code Table

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Description
0	0							bit error
0	1							form error
1	0							stuff error
1	1							other type of error
		0						error occurred during reception
		1						error occurred during transmission
			0	0	0	1	1	start of frame
			0	0	0	1	0	ID.28 to ID.21
			0	0	1	1	0	ID.20 to ID.18
			0	0	1	0	0	bit SRTR
			0	0	1	0	1	bit IDE
			0	0	1	1	1	ID.17 to ID.13

			0	1	1	1	1	ID.12 to ID.5
			0	1	1	1	0	ID.4 to ID.0
			0	1	1	0	0	bit RTR
			0	1	1	0	1	reserved bit 1
			0	1	0	0	1	reserved bit0
			0	1	0	1	1	data length code
			0	1	0	1	0	data field
			0	1	0	0	0	CRC sequence
			1	1	0	0	0	CRC delimiter
			1	1	0	0	1	acknowledge slot
			1	1	0	1	1	acknowledge delimiter
			1	1	0	1	0	end of frame
			1	0	0	1	0	intermission
			1	0	0	0	1	active error flag
			1	0	1	1	0	passive error flag
			1	0	0	1	1	tolerate dominant bits
			1	0	1	1	1	error delimiter
			1	1	1	0	0	overload flag

5-3 Notes

Note 1: The CANBUS DB9-pin out conforms to the ISO 11898/2 standard ..6

Note 2: The JP6 is the CANbus termination jumper. Only two termination jumpers should be closed at the endpoints of the CANbus. Value Terminator Resistor (120 Ω). **The minimum speed is 20k bps. The maximum speed is 1M bps.** But when CANBUS terminator is **disabled**, the maximize speed of CANBUS is **125k bps**. If you want to use high speed (1M bps), please enable terminator.6

Note 3: More information of Status is given in the data sheet of SJA 1000 in

section 6.4.5 Status Register (SR).....	14
Note 4: More information of Status is given in the data sheet of SJA 1000 in section 6.4.15 Acceptance Filter.	14
Note 5: More information of Status is given in the data sheet of SJA 1000 in section 6.4.6 Interrupt Register (IR).....	15
Note 6: More information of Status is given in the data sheet of SJA 1000 in section 6.4.3 Mod Register (MOD).	16
Note 7: More information of Status is given in the data sheet of SJA 1000 in section 6.4.4 Command Register (CMR).	17
Note 8: Please refer to Error Code Table for details.	18
Note 9: More information of Status is given in the data sheet of SJA 1000 in section 6.4.9.....	18
Note 10: More information of Status is given in the data sheet of SJA 1000 in section 6.4.10.	18
Note 11: More information of Status is given in the data sheet of SJA 1000 in section 6.4.12.	19
Note 12: More information of Status is given in the data sheet of SJA 1000 in section 6.4.11.	19
Note 13: More information on setting the bit rate is given in the data sheet of SJA 1000 in section 6.5.	21

Note: You can get the SJA1000 datasheet from following website:

[http://www.nxp.com/#/pip/pip=\[pip=SJA1000_3\]|pp=\[t=pip,i=SJA1000_3\]](http://www.nxp.com/#/pip/pip=[pip=SJA1000_3]|pp=[t=pip,i=SJA1000_3])

5-4 Customer Service



Korenix Technologies Co., Ltd.

business service: sales@korenix.com

Customer service: koreCARE@korenix.com