



JetBox 8152 User Manual

WinCE 5.0 Canbus

www.korenix.com

Copyright Notice

Copyright© 2011 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

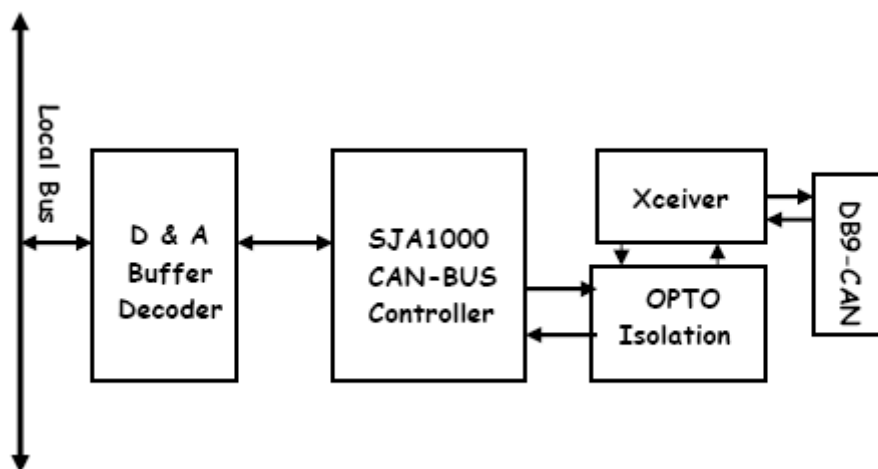
Copyright Notice	2
Acknowledgments.....	2
Table of Content	3
Chapter 1 Introduction	4
Chapter 2 Hardware Configuration.....	6
2-1 Pin Assignment.....	6
2-2 Jumper Setting: JP6.....	6
Chapter 3 Sample Program Configuration.....	7
3-1 Program Start	7
Chapter 4 Korenix SDK Reference	9
4-1 How to Install and Use Canbus SDK	9
4-2 Functions.....	11
4-3 Structure	22
Chapter 5 Appendix	24
5-1 Baud Rate Table.....	24
5-2 Error Code Table.....	25
5-3 Notes	26
5-4 Revision History.....	27
5-5 Customer Service	27

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)

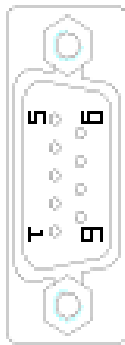
Outportb (0x301, 0x78) : Write Data of CAN Address 0 (Control Register)

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.

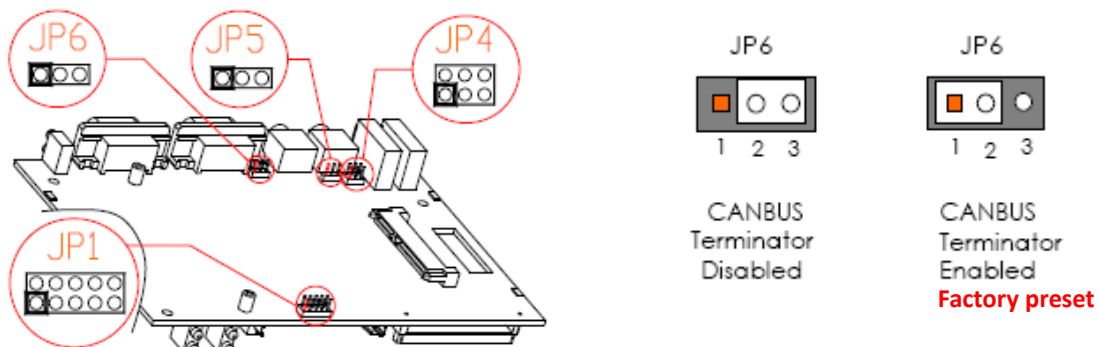



DB-9 CANBUS	CANBUS Signal	CANBUS Description
1	N.C	-
6	N.C	-
2	CAN-L	Dominant Low
7	CAN-H	Dominant High
3	CAN-Ground	Isolated Ground
8	N.C	-
4	N.C	-
9	N.C	-
5	Ground	Digital Ground
Case	Case Ground	

 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.

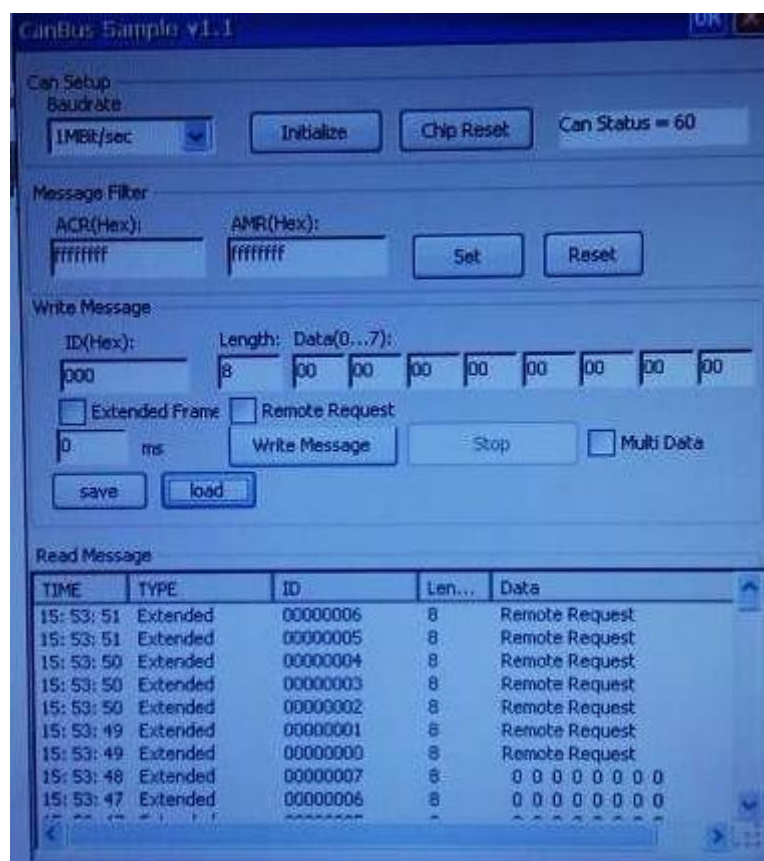
Chapter 3 Sample Program Configuration

Under WinCE 5.0 platform, a driver is needed to access the Can Port and to provide the interface for WinCE 5.0 applications. Beside the mentioned device driver, the CANBus Sample application for WinCE 5.0 can also be set up.

3-1 Program Start

CANbus Sample Program

CANbus Sample for WinCE 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.

Note.User have to Initialize baud rate after chip reset.Before doing this step,write message won't work.

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.If time filter is 0 ms,write message button can only send canbus message one time,else send message will sen messages in sequence.While user enable Multi Data function,canbus message will send ids between id+0x00 to id+0x14,and will plus 1 to data after a loop.Save/Load will save/load read message filter from canmessage.txt.

Read Messages

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

Chapter 4 Korenix SDK 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 Install and Use Canbus SDK

When Install the Canbus SDK fir Visual C++, please do the following steps:

1.Click VIACAN_SDK.msi

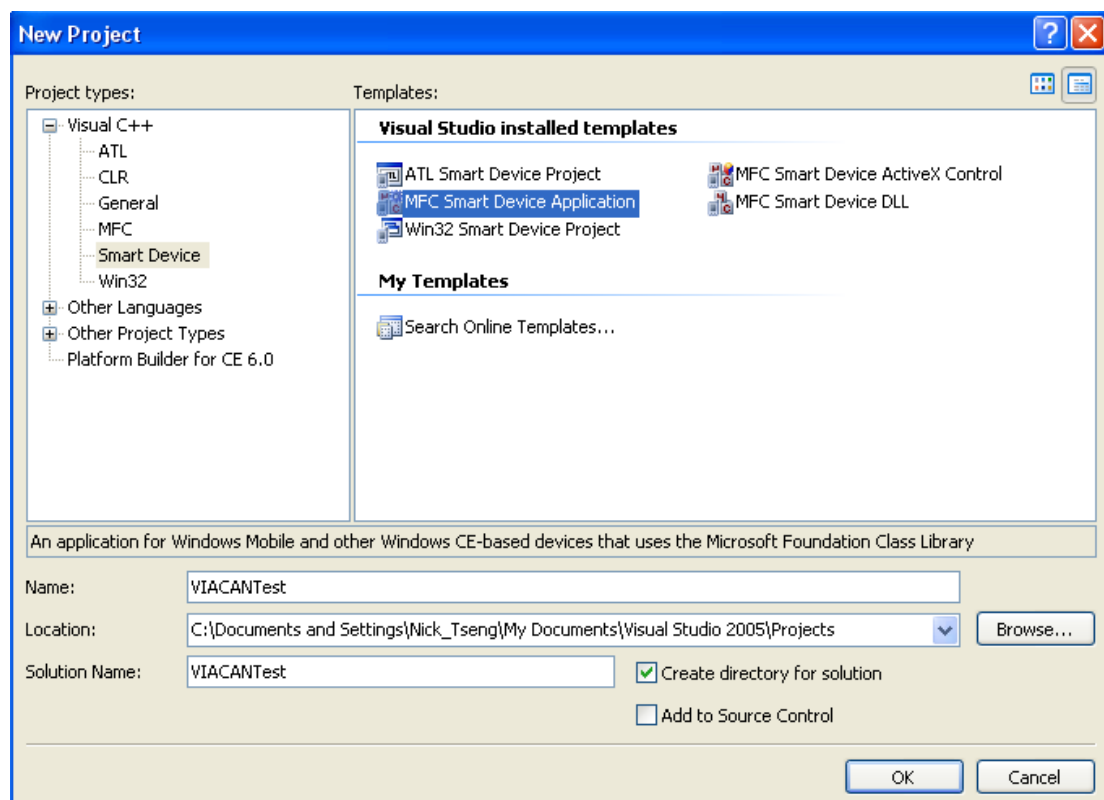


2.Click next and agree to installl VIACAN_SDK .

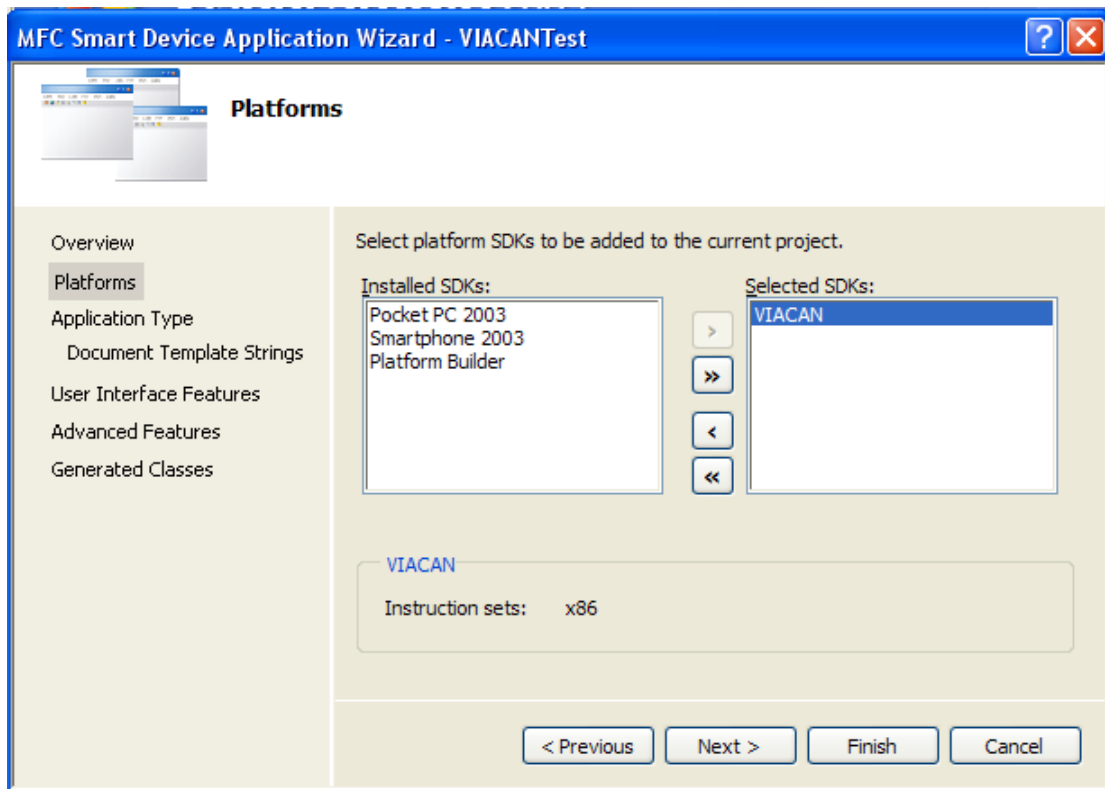
3. Open Visual Stdio 2005/2008

4. Create a New Project

5. Choose Visual C++ => Smart Device=>Project Type



6. Choose VIACAN as Target Platform.



7. Include CANBUS.H as CANBUS headfile.

4-2 Functions

The CanPort IO control provides the following functions

❑ **CreateFile**

```
HANDLE CreateFile(  
LPCTSTR lpFileName,  
DWORD dwDesiredAccess,  
DWORD dwShareMode,  
LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
DWORD dwCreationDisposition,  
DWORD dwFlagsAndAttributes,  
HANDLE hTemplateFile  
);
```

This function creates, opens, or truncates a file, CAN port, device, service, or console. It returns a handle that you can use to access the object.

Coding Examples:

```
hCANBUSDevice = CreateFile(TEXT("CAN1:."),  
GENERIC_READ | GENERIC_WRITE,  
FILE_SHARE_READ | FILE_SHARE_WRITE,  
NULL,  
OPEN_EXISTING,  
0,  
NULL);
```

❑ **DeviceIoControl**

```
BOOL DeviceIoControl(  
HANDLE hDevice,  
DWORD dwIoControlCode,  
LPVOID lpInBuffer,  
DWORD nInBufferSize,  
LPVOID lpOutBuffer,  
DWORD nOutBufferSize,  
LPDWORD lpBytesReturned,  
LPOVERLAPPED lpOverlapped  
);
```

This function sends an IOCTL directly to a specified device driver, causing the corresponding device to perform the specified operation.

❑ **IOCTL_INIT_CAN**

Parameters	Description
IOCTL_INIT_CAN	This IOCTL allows the user to sets configuration parameters to initialize the CAN controller.
lpInBuffer	An array to a UCHAR BTR[2] Structure BTR[0] BUS TIMING REGISTER 0 BTR[1] BUS TIMING REGISTER 1
nInBufferSize	Size of UCHAR * 2
lpOutBuffer	NULL.
nOutBufferSize	0.

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.

❑ **IOCTL_CAN_CHIP_RESET**

Parameters	Description
IOCTL_CAN_CHIP_RESET	This IOCTL resets the CAN controller to default state.
lpInBuffer	NULL
nInBufferSize	0
lpOutBuffer	NULL.
nOutBufferSize	0.

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.

Note .User must have to Initialize baud rate after reset chip because cac chip won't work before **IOCTL_INIT_CAN** .

❑ IOCTL_CAN_SEND_MESSAGE


Parameters	Description
IOCTL_CAN_SEND_MESSAGE	This IOCTL set canbus message to send.
lpInBuffer	CAN_MSG MsgToSend Please refer to Structure for details
nInBufferSize	Sizeof CAN_MSG structure
lpOutBuffer	NULL.
nOutBufferSize	0.

❑ IOCTL_CAN_RECEIVE_MESSAGE

Parameters	Description
IOCTL_CAN_RECEIVE_MESSAGE	This IOCTL returns a CAN message from the receive queue.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	CAN_MSG MsgToRead
nOutBufferSize	Sizeof CAN_MSG structure.


❑ IOCTL_CAN_STATUS_REPORT

Parameters	Description
IOCTL_CAN_STATUS_REPORT	This IOCTL returns a the status of CAN controller. Get the current status of the CAN controller.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 pCanStatusValue
nOutBufferSize	Sizeof UINT8.

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

❑ IOCTL_CAN_SET_FILTER

Parameters	Description
IOCTL_CAN_SET_FILTER	This IOCTL sets the Acceptance Code and Acceptance Mask of the CAN controller. The CAN controller must set to reset mode when calling the function.
lpInBuffer	UCHAR CSF[2] CSF[0] Acceptance Code Register CSF[1] Acceptance Mask Register
nInBufferSize	sizeof(UCHAR)*2
lpOutBuffer	NULL
nOutBufferSize	0

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

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 0x007FFFFFF.

❑ **IOCTL_CAN_RESET_FILTER**

Parameters	Description
IOCTL_CAN_RESET_FILTER	This IOCTL reset the Acceptance Code and Acceptance Mask Register of the CAN to default value (It means accept all CAN message).
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	NULL
nOutBufferSize	0.

❑ **IOCTL_CAN_INTERRUPT_STATUS**


Parameters	Description
IOCTL_CAN_RESET_FILTER	This IOCTL returns The value of Interrupt Register .
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanValue The value of Interrupt Register
nOutBufferSize	Size of UINT8.

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

❑ **IOCTL_CAN_MODE_SET**

Parameters	Description
IOCTL_CAN_MODE_SET	This IOCTL sets the operation mode of canbus
lpInBuffer	UINT8 bMode operation mode
nInBufferSize	Size of UINT8.
lpOutBuffer	NULL
nOutBufferSize	0.

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).

IOCTL_CAN_SET_BTR

Parameters	Description
IOCTL_CAN_SET_BTR	This IOCTL only sets the baud rate to CAN controller.
lpInBuffer	An array to a UCHAR BTR[2] Structure BTR[0] BUS TIMING REGISTER 0 BTR[1] BUS TIMING REGISTER 1
nInBufferSize	Size of UCHAR * 2
lpOutBuffer	NULL.
nOutBufferSize	0.

IOCTL_CAN_GET_BTR

Parameters	Description
IOCTL_CAN_SET_BTR	This IOCTL only sets the baud rate to CAN controller.
lpInBuffer	NULL
nInBufferSize	0
lpOutBuffer	An array to a UCHAR BTR[2] Structure BTR[0] BUS TIMING REGISTER 0 BTR[1] BUS TIMING REGISTER 1
nOutBufferSize	Size of UCHAR * 2

❑ IOCTL_CAN_SET_COMMAND

Parameters	Description
IOCTL_CAN_SET_COMMAND	This IOCTL sets the command operation mode of canbus.
lpInBuffer	UINT8 bCmd Set Command mode
nInBufferSize	Size of UINT8
lpOutBuffer	NULL.
nOutBufferSize	0.


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).

❑ IOCTL_CAN_GET_ECR

Parameters	Description
IOCTL_CAN_GET_ECR	This IOCTL returns the value of Error Code Capture Register.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanECRValue The value of Error Code Capture Register
nOutBufferSize	Size of UINT8.

 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](#).

❑ IOCTL_CAN_GET_ALC

Parameters	Description
IOCTL_CAN_GET_ALC	This IOCTL returns the value of Arbitration Lost Capture Register.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanALCValue The value of Arbitration Lost Capture Register
nOutBufferSize	Size of UINT8.

❑ IOCTL_CAN_GET_EWL


Parameters	Description
IOCTL_CAN_GET_EWL	This IOCTL returns the value of Error Warning Limit Register.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanEWLValue 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.
nOutBufferSize	Size of UINT8.

 Note 10: More information of Status is given in the data sheet of SJA 1000 in [section 6.4.10](#).

❑ IOCTL_CAN_TXERROR_COUNTER


Parameters	Description
IOCTL_CAN_TXERROR_COUNTER	This IOCTL returns the TX error counter register reflects the current value of the transmit error

	counter.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanTXERRORValue The TX error counter register reflects the current value of the transmit error counter.
nOutBufferSize	Size of UINT8.

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

IOCTL_CAN_RXERROR_COUNTER

Parameters	Description
IOCTL_CAN_RXERROR_COUNTER	This IOCTL returns the value of receive error. The RX error counter register reflects the current value of the receive error counter.
lpInBuffer	NULL
nInBufferSize	0.
lpOutBuffer	UINT8 CanALCValue The value of receive error.
nOutBufferSize	Size of UINT8.

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

WriteFile

```
WriteFile(HANDLE hDevice,
          CAN_MSG g_TXmsg,
          DWORD cCount,
          DWORD dwBytesWritten,
          OVERLAPPED ovWrite)
```

Parameters

`g_TXmsg`

CANBUS Message Data

`cCount`

`sizeof(g_TXmsg)`

Return

`dwBytesWritten`

Number of CANBUS Message Bytes Written

`ovWrite`

Write overlapped

Sample code:

```
OVERLAPPED ovWrite;
HANDLE hWriteEvent=CreateEvent(NULL,TRUE,FALSE
,TEXT("pHwHead->hTxEvent"));
memset(&ovRead,0,sizeof(ovWrite));
ovWrite.hEvent = hWriteEvent;
WriteFile(hDevice, &g_txMsg, dwCount, &dwBytesWritten, &ovWrite);
```

ReadFile

```
ReadFile(HANDLE hDevice,
CAN_MSG &g_RXmsg,
DWORD cCount,
DWORD dwBytesReading,
OVERLAPPED ovRead)
```

Parameters

`g_RXmsg`

CANBUS Message Data

`cCount`

`sizeof(g_RXmsg)`

Return

`dwBytesReading`

Number of CANBUS Message Bytes reading


`ovRead`

Read overlapped

Sample code:

```
OVERLAPPED ovRead;
HANDLE hReadEvent=CreateEvent(NULL,TRUE,FALSE
,TEXT("pHwHead->hRxEvent"));
memset(&ovRead,0,sizeof(ovRead));
ovRead.hEvent = hReadEvent;
if (!ReadFile(hDevice, &g_rxMsg, dwCount, &dwBytesReading, &ovRead))
{
    printf("\n IOCTL_CAN_SEND_MESSAGE NG!\t\n");
}else{
    if(dwBytesReading!=0){
        printf("\nID          Entended Request Length\t\n");
        if(g_rxMsg.format!=1){
            printf("%2x%2x %x %x !\t\n"
                ,g_rxMsg.id[1],g_rxMsg.id[0],g_rxMsg.format,g_rxMsg.
                remote,g_rxMsg.length);
        }else{
            printf("%2x%2x%2x%2x %x %x %x!\t\n",
                g_rxMsg.id[3],g_rxMsg.id[2],g_rxMsg.id[1],g_rxMsg.id[
                0],g_rxMsg.format,g_rxMsg.remote,g_rxMsg.length);
        }

        printf("\nData\t\n");
        if(g_rxMsg.remote != 1){
            for(int i=0;i< g_rxMsg.length;i++){
                printf("%2x ",g_rxMsg.data[i] );
            }
        }
    }
}
```

 **Note 133:** If user want to use polling mode with ReadFile, User have to read `IOCTL_CAN_INTERRUPT_STATUS` to analyze if Interrupt Status is under "Receive Interrupt Bit" than read can message.

4-3 Structure

The CanPort API defines the following structures

`CAN_MSG` : Defines a CAN message

`PCAN_MSG` : Defines a pointer as CAN message

```
typedef struct _CAN_MSG{ /* CAN_MSG */
    union {
        UINT8 id[4];
        UINT32 identifier;
    };
    union {
        struct {
            UINT8 length :4; /* data length */
            UINT8 resbit :2;
            UINT8 remote :1; /* remote transmission request */
            UINT8 format :1; /* frame format */
        };
        UINT8 info;
    };
    UINT8 reserve[3];
    UINT8 data[8]; /* data field */
} CAN_MSG,*PCAN_MSG;
```

- `id[0]~[3]`:

standard mode:11bits CAN identifier.`id[0]`:3 bits(8~11) `id[1]`:8 bits(0~7)

extended mode:29bit CAN identifier.`id[0]`(24~28)`id[1]`(16~23)`id[2]`(8~15)`id[3]`(0~7)

- `format`:

Bit mask indicating the type of the message. Several message types can be combined.

Identifier	Value	Description
Standard Mode	0	Data Frame. CAN message with data contents and an 11-bit CAN ID.
Extended Mode	1	Data Frame. CAN message with data contents according to CAN 2.0B standard (29-bit CAN ID).

- `remote`:

To enable Remote Transmit Request (RTR).

- length
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 14: 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)..... 13

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).It is 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: If user want to use polling mode with ReadFile,User have to read IOCTL_CAN_INTERRUPT_STATUS to analyze if Interrupt Status is under “Receive Interrupt Bit” than read can message.	21
Note 13: More information on setting the bit rate is given in the data sheet of SJA 1000 in section 6.5.	25

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 Revision History

V0.4	by 2011/ 4/ 8
	1. Update chapter 3 and 4
	2. Change JP6 setting to enable
V0.3	by 2011/ 3/ 1
	Initial release

5-5 Customer Service

Korenix Technologies Co., Ltd.

Business service: sales@korenix.com

Customer service: koreCARE@korenix.com