JNMM-4LP-XT Driver Software User Manual

Janus-MM-4LP Quad CAN
PC/104-Plus Module

For Version 1.0.0

Revision A.4 Jan 2017

Revision	Date	Comment
A.0	9/1/2016	Initial release
A.1	10/21/2016	Update for PTB/STB method and error handling
A.2	11/02/2016	Update for RW function for PCI/ISA mode, User to enter the Base address/DIO address for ISA mode.
A.3	12/02/2016	Updated with installPCI.sh/uninstallPCI.sh/InstallISA.sh/UninsatllISA.sh for updating the driver at load time for PCI/ISA mode
A.4	1/02/2017	Updated GUI monitor.

FOR TECHNICAL SUPPORT PLEASE CONTACT:

support@diamondsystems.com

Diamond Systems Corporation
555 Ellis Street
Mountain View, CA 94043 USA
Tel 1-650-810-2500
Fax 1-650-810-2525
www.diamondsystems.com

CONTENTS

1. INTRODUCTION

This user manual contains all essential information about the JNMM-4LP-XT Driver 1.0 JNMM-4LP-XT demo applications, programming guidelines and usage instructions. This manual also includes the JNMM-4LP-X Driver API descriptions with usage examples.

2. HARDWARE OVERVIEW

2.1 Description

The Janus-MM-4LP-XT family of I/O modules offers two or four opto-isolated CANbus 2.0B ports plus 16 digital I/O lines. Models are available in both the PC/104-Plus and PC/104 form factors. An FGPA core houses the CAN controller logic and digital I/O logic providing data rates up to 1Mbps. Each CAN port supports standard and extended frames as well as expanded TX and RX message queues for enhanced performance. Each port has its own combination isolator and transceiver chip.

Janus-MM-4LP-XT fits a wide variety of rugged and on-vehicle embedded serial I/O application needs. It was designed with harsh applications in mind including latching connectors to further improve reliability. Extended temperature operation of -40°C to +85°C is tested and guaranteed. The module is compatible with MIL-STD-202G shock and vibration specifications.

2.2 Specifications

- 2 or 4 CAN 2.0B compatible ports
- Data rates up to 1Mbps
- Supports standard 11-bit identifier and extended 29-bit identifier frames
- Extended TX and RX message queues for enhanced performance
- 16 8-byte transmit message queues
- 31 8-byte receive message queues
- 16 receive filters
- Galvanically isolated transceivers
- 500V port-to-host and port-to-port isolation
- Jumper selectable biased split termination for improved noise reduction
- 16 digital I/O lines
- Latching I/O connectors for increased ruggedness
- PCI and ISA bus interfaces

3. GENERAL PROGRAMMING GUIDELINES

3.1 Initialization and Exit Function Calls for Linux

All the demo applications begin with the following functions and should be called in sequence to initialize the JNMM-4LP-XT Driver .

1. JNMMCANOpen (),This function opens a handle to access the Diamond JNMM CAN driver and returns the handle value.
2. JNMMCANClose (), This function closed the access to the Diamond JNMM CAN driver.

These function calls are important in opening and closing the file descriptor.

```
#include "../CANLib/libcan.h"
int main()
{
	int can_fd;
	Int flag;
	can_fd = JNMMCANOpen(&flag);
	if(can_fd == -1)
	{
		return 0;
	}
	If(flag==1) // ISA mode
	{
		Printf("JNMMCAN4LP-XT-ISA mode ");
		Base_address = 0x6000;
		JNMMCANBaseAddrInit(base_address);
		Diobase_address =0x140;
		JNMMCANDIOAddrInit(diobase_address);

	}
	Else
	{
		Printf("JNMMCAN4LP-XT-PCI mode ");

	}

	JNMMCANClose(can_fd);
}
```

3.2 Error Handling for Linux

If file discriptor value is "-1" it means we are not able to open the file, make sure that driver has been installed or not .

```
	can_fd = JNMMCANOpen(&flag);
	if(can_fd == -1)
	{
		return 0;
	}
```

3.3 Initialization and Exit Function Calls for Windows

All the demo applications begin with the following functions and should be called in sequence to initialize the JNMM-4LP-XT Driver.

1. JNMMCANOpen(),This function opens a handle to access the Diamond JNMM CAN driver and returns the handle value..
2. JNMMCANClose(), This function closed the access to the Diamond JNMM CAN driver.

These function calls are important in opening and closing the handle.

```
int main()
      {
            HANDLE hHandle;
            Int flag;
            hHandle = JNMMCANOpen(&flag);
            if(hHandle == -1)
            {
                  return 0;
            }
            If(flag==1) // ISA mode
            {
                  Printf("JNMMCAN4LP-XT-ISA mode ");
                  Base_address = 0x6000;
                  JNMMCANBaseAddrInit(base_address);
                  Diobase_address =0x140;
                  JNMMCANDIOAddrInit(diobase_address);

            }
            JNMMCANClose(hHandle);
      }
```

3.4 Error Handling for Windows

If file handle value is "-1" it means we are not able to open the file ,make sure that driver has been installed or not .

```
hHandle = JNMMCANOpen(&flag);
if(hHandle == -1)
{
      return 0;
}
```

4. JNMM-4LP-XT FOR LINUX DRIVER API DESCRIPTION

4.1 JNMMCANConfig

Function Definition
BYTE JNMMCANConfig(int can_fd, CANCONFIG* config);

Function Description
This function configures the CAN controller with baud rate.

Function Parameters

Name	Description
Can_fd	Handle to Diamond JNMM CAN driver as received from JNMMCANOpen().
CANCONFIG	Structure with following member variables, Can_ch - CAN port number 0-3. BaudRate - constant indicating the baud rate [0-B1000 ─ 1000kbit/s ,1-B800 ─ 800kbit/s , 2-B500 ─ 500kbit/s, 3-B250 ─ 250kbit/s,4-B125 -125kbits/s, 5-B100 ─ 100kbit/s,6-B50 ─ 50kbit/s,7-B20 ─ 20kbit/s, 8-B10 ─ 10kbit/s]

Return Value
Error code or 0

Usage Example
To configure baudrate.

```
canconfig.BaudRate=1; //B800kbit/s
success = JNMMCANConfig(device_handle, canconfig);
```

4.2 JNMMCANTX

Function Definition
BYTE JNMMCANTX(HANDLE hHandle, CANMESSAGE* msg);

Function Description
This function transmits a message on the selected CAN port.

Function Parameters

Name	Description
Can_fd	The Can_fd is received from JNMMCANOpen().
CANMESSAGE	Pointer to CANMESSAGE structure Can_ch : CAN port numer 0-3 Msg_type : 0 : Standard 1 : Extended Msg_ID : Hexadecimel value Msg_len : length of the frames RTR : 0 : Disable 1 : Enable Data : Holds the data frame of message length

Return Value
Error code or 0.

Usage Example
For transmits the message.

```
Tx.Can_ch = 0;
Tx.Msg_type = MSG_STANDARD;
Tx.Msg_ID = 0x12;
Tx.Rtr    = 0;
Tx.Msg_len = 4;
Tx.Data[0] = 0x1A;
Tx.Data[1] = 0xAB;
Tx.Data[2] = 0x22;
Tx.Data[3] = 0x4D;
ret = JNMMCANTX(device_handle,&Tx);
if(ret == 1)
{
      printf("JNMMCANTX failure \n");
}
```

4.3 JNMMCANRX

Function Definition
BYTE JNMMCANRX(int Can_fd, CANMESSAGE* msg);

Function Description
This function receives a message on the selected CAN port.

Function Parameters

Name	Description
Can_fd	Can_fd received from JNMMCANOpen().
CANMESSAGE	Pointer to CANMESSAGE structure Can_ch : CAN port numer 0-3 Msg_type : 0 : Standard 1: Extended Msg_ID : Hexadecimel value Msg_len : length of the frames RTR : 0 : Disable 1: Enable Data : Holds the data frame of message length

Return Value
Error code or 0.

Usage Example
For receive the message.

```
Memset(&rx,0,sizeof(CANMESSAGE));
ret = JNMMCANRX(device_handle,&Rx);
 if(ret == -1)
 {
      printf("JNMMCANRX failure \n");
  }
```

4.4 JNMMCANErrorStat

Function Definition
BYTE JNMMCANErrorStat(int Can_fd, CANCONFIG CanParam, can_device_stats* stats);

Function Description
This function receives a message on the selected CAN port.

Function Parameters

Name	Description
Can_fd	Can_fd received from JNMMCANOpen().
CanParam	Structure with following member variables, Can_ch - CAN port number 0-3.
can_device_stats	Pointer to can_device_stats structure txerr no of Tx errors rxerr no of Rx errors txpackets no of Tx transmitted count rxpackets no of Rx transmitted count rx_over_errors no of Rx over error count error_active Error active enable stat_change stat changed from passive to active ,vice versa */ bit_error bit error stuff_error stuff error form_error Form error bus_error no of Bus errors bus_error_enable bit error enable error_warning Changes to error warning state error_passive Changes to error passive state bus_off Changes to bus off state arbitration_lost Arbitration lost errors arbitration_lost_enable arbitration lost error enable

Return Value
Error code or 0.

Usage Example

For receive the message.

```
Memset(&stats,0,sizeof(can_device_stats));
ret = JNMMCANErrorStat (device_handle,&stats);
 if(ret == -1)
 {
      printf("JNMMCANErrorStat failure \n");
  }
```

4.5 JNMMCANRead

Function Definition

BYTE JNMMCANRead(int can_ch, unsigned char offset);

Function Description

This function reads the data value from device register for a specified CAN port(0-3).

Function Parameters

Name	Description
Can_ch	CAN port(0-3)
offset	Device register offset

Return Value

Error code or 0.

Usage Example

To read CAN port.

```
Can_ch=0;//port 0
Offset =0x90; // device register offset
Value = JNMMCANRead( can_ch, Offset );
Printf("data=%d",value);
```

4.6 JNMMCANWrite

Function Definition
BYTE JNMMCANWrite((int can_ch, unsigned char offset,unsigned char data);

Function Description
This function reads the data value from a specified GPIO port.

Function Parameters

Name	Description
Can_ch	CAN port(0-3)
offset	Device register offset
data	Data to write into the device register

Return Value
Error code or 0.

Usage Example
To Write CAN port to a particular register

```
Can_ch=0;//port 0
Offset =0x90; // device register offset
Data =0x80;
ret = JNMMCANWrite( can_ch, Offset,data );
```

4.7 JNMMCANBaseAddrInit

Function Definition
BYTE JNMMCANBaseAddrInit(int base_address);

Function Description
This function will get the Base address from the user for ISA mode.

Function Parameters

Name	Description
base_address	int base_address base address received from user

Return Value
Error code or 0.

Usage Example
For getting the base address.

```
Base_address =0x6000 // please refer the J14 configuration for ISA mode
JNMMCANBaseAddrInit(Base_address);
```

4.8 JNMMCANDIOAddrInit

Function Definition
BYTE JNMMCANDIOAddrInit(int dio_base);

Function Description
This function will get the digital I/O Base address from the user for ISA mode.

Function Parameters

Name	Description
dio_address	int dio_address dio address received from user

Return Value
Error code or 0.

Usage Example
For getting the base address.

```
dio_address =0x140 // please refer the J14 configuration for ISA mode
JNMMCANDIOAddrInit(dio_address);
```

4.9 JNMMGetFPGAVersion

Function Definition
Unsigned short JNMMGetFPGAVersion();

Function Description
This function returns the FPGA Version.

Function Parameters

Name	Description
Unsigned short	Read and return the FPGA version value.

Return Value
Error code or 0.

Usage Example
For collecting the FPGA version.

```
Fpga= JNMMGetFPGAVersion();
printf("FPGA Version=%d",Fpga);
```

4.10 JNMMGetBoardVersion

Function Definition
Unsigned short JNMMGetBoardVersion();

Function Description
This function returns the board Version.

Function Parameters

Name	Description
Unsigned short	Read and return the board version value.

Return Value
Error code or 0.

Usage Example
For getting Board version.

```
Board_version= JNMMGetBoardVersion();
Printf("Board Version=%d",Board_version);
```

4.11 JNMMGPIOConfig

Function Definition
BYTE JNMMGPIOConfig(int Port, int Config);

Function Description
This function sets the digital I/O port direction for the selected port.

Function Parameters

Name	Description
Port	0-1 for port A, B
Config	0 = input, 1 = output

Return Value
Error code or 0.

Usage Example
For configure GPIO.

```
input_port =0;//0-port A,1-port B
config = 0;//input mode
JNMMGPIOConfig(input_port,config);
```

4.12 JNMMGPIORead

Function Definition
BYTE JNMMGPIORead(int Port, int *data);

Function Description
This function reads the data value from a specified GPIO port.

Function Parameters

Name	Description
Port	0-1 for port A, B
data	read value.

Return Value
Error code or 0.

Usage Example
To read GPIO port.

```
inport=0;//port A
JNMMGPIORead( inport, &input_byte );
Printf("data=%d",*input_byte);
```

4.13 JNMMGPIOWrite

Function Definition
BYTE JNMMGPIOWrite(int Port, int data);

Function Description
This function writes a data value to a specified GPIO port.

Function Parameters

Name	Description
Port	0-1 for port A, B
data	Value.

Return Value
Error code or 0.

Usage Example
To write a data on port A

```
outport=0;//port A
output_byte=20;
JNMMGPIOWrite( outport, output_byte );
Printf("written data=%d",output_byte);
```

4.14 JNMMGPIOReadBit

Function Definition
BYTE JNMMGPIOReadBit(int Port, int Bit,int *data);

Function Description
This function outputs the desired value to a single bit. The other bits remain at their current values

Function Parameters

Name	Description
Port	0-1 for port A, B
Bit	DIO line number, 0-15
Data	read value.

Return Value
Error code or 0.

Usage Example
To read single bit.

```
port=0;//port A
bit=6;//6th bit
JNMMGPIOReadBit(port,bit,&digital_value);
```

4.15 JNMMGPIOWriteBit

Function Definition
BYTE JNMMGPIOWriteBit(int Port,int Bit, int data);

Function Description
This function reads in the specified bit and returns it in the location specified by the pointer to data.

Function Parameters

Name	Description
Port	0-1 for port A, B
Bit	DIO line number, 0-15
Data	Value.

Return Value
Error code or 0.

Usage Example
To write a single bit.

```
sscanf ( input_buffer, "%d", &intBuf );
digital_val = intBuf;
JNMMGPIOWriteBit(port,bit,digital_val);
```

5. JNMM-4LP-XT FOR WINDOWS DRIVER API DESCRIPTION

5.1 JNMMCANConfig

Function Definition
BYTE JNMMCANConfig(HANDLE hHandle, CANCONFIG* config);

Function Description
This function configures the CAN controller with baud rate and remote transmission request for the specified CAN port.

Function Parameters

Name	Description
hHandle	Handle to Diamond JNMM CAN driver as received from JNMMCANOpen().
CANCONFIG	Structure with following member variables, Can_ch - CAN port number 0-3. BaudRate - constant indicating the baud rate [0-B1000 ─ 1000kbit/s ,1-B800 ─ 800kbit/s , 2-B500 ─ 500kbit/s, 3-B250 ─ 250kbit/s,4-B125 -125kbits/s, 5-B100 ─ 100kbit/s,6-B50 ─ 50kbit/s,7-B20 ─ 20kbit/s, 8-B10 ─ 10kbit/s]

Return Value
Error code or 0

Usage Example
To configure baudrate.

```
canconfig.BaudRate=1;// B800kbit/s
success = JNMMCANConfig(device_handle, canconfig);
```

5.2 JNMMCANTX

Function Definition
BYTE JNMMCANTX(HANDLE hHandle, CANMESSAGE* msg);

Function Description
This function transmits a message on the selected CAN port.

Function Parameters

Name	Description
hHandle	The hHandle is received from JNMMCANOpen().
CANMESSAGE	Pointer to CANMESSAGE structure Can_ch : CAN port numer 0-3 Msg_type : 0 : Standard 1 : Extended Msg_ID : Hexadecimel value Msg_len : length of the frames RTR : 0 : Disable 1 : Enable Data : Holds the data frame of message length

Return Value
Error code or 0.

Usage Example
For transmits the message.

```
Tx.Can_ch = 0;
Tx.Msg_type = MSG_STANDARD;
Tx.Msg_ID = 0x12;
Tx.Rtr    = 0;
Tx.Msg_len = 4;
Tx.Data[0] = 0x1A;
Tx.Data[1] = 0xAB;
Tx.Data[2] = 0x22;
Tx.Data[3] = 0x4D;
ret = JNMMCANTX(device_handle,&Tx);
if(ret == 1)
{
      printf("JNMMCANTX failure \n");
}
```

5.3 JNMMCANRX

Function Definition
BYTE JNMMCANRX(**(**HANDLE hHandle, CANMESSAGE* msg);

Function Description
This function receives a message on the selected CAN port.

Function Parameters

Name	Description
hHandle	HANDLE Handle received from JNMMCANOpen().
CANMESSAGE	Pointer to CANMESSAGE structure Can_ch : CAN port numer 0-3 Msg_type : 0 : Standard 1 : Extended Msg_ID : Hexadecimel value Msg_len : length of the frames RTR : 0 : Disable 1 : Enable Data : Holds the data frame of message length

Return Value
Error code or 0.

Usage Example
For receive the message.

```
ret = JNMMCANRX(device_handle,&Rx);
if(ret == 1)
{
      printf("JNMMCANRX failure \n");
 }
```

5.4 JNMMCANBaseAddrInit

Function Definition
BYTE JNMMCANBaseAddrInit(int base_address);

Function Description
This function will get the Base address from the user for ISA mode.

Function Parameters

Name	Description
base_address	int base_address base address received from user

Return Value
Error code or 0.

Usage Example
For getting the base address.

```
Base_address =0x6000 // please refer the J14 configuration for ISA mode
JNMMCANBaseAddrInit(Base_address);
```

5.5 JNMMCANDIOAddrInit

Function Definition
BYTE JNMMCANDIOAddrInit(int dio_base);

Function Description
This function will get the digital I/O Base address from the user for ISA mode.

Function Parameters

Name	Description
dio_address	int dio_address dio address received from user

Return Value
Error code or 0.

Usage Example
For getting the base address.

```
dio_address =0x140 // please refer the J14 configuration for ISA mode
JNMMCANDIOAddrInit(dio_address);
```

5.6 JNMMGetFPGAVersion

Function Definition
Unsigned short JNMMGetFPGAVersion();

Function Description
This function returns the FPGA Version.

Function Parameters

Name	Description
Unsigned short	Read and return the FPGA version value.

Return Value
Error code or 0.

Usage Example
For collecting the FPGA version.

```
Fpga= JNMMGetFPGAVersion();
printf("FPGA Version=%d",Fpga);
```

5.7 JNMMGetBoardVersion

Function Definition
Unsigned short JNMMGetBoardVersion();

Function Description
This function returns the board Version.

Function Parameters

Name	Description
Unsigned short	Read and return the board version value.

Return Value
Error code or 0.

Usage Example
For getting Board version.

```
Board_version= JNMMGetBoardVersion();
Printf("Board Version=%d",Board_version);
```

5.8 JNMMGPIOConfig

Function Definition
BYTE JNMMGPIOConfig(int Port, int Config);

Function Description
This function sets the digital I/O port direction for the selected port.

Function Parameters

Name	Description
Port	0-1 for port A, B
Config	0 = input, 1 = output

Return Value
Error code or 0.

Usage Example
For configure GPIO.

```
input_port =0;//0-port A,1-port B
config = 0;//input mode
JNMMGPIOConfig(input_port,config);
```

5.9 JNMMGPIORead

Function Definition
BYTE JNMMGPIORead(int Port, int *data);

Function Description
This function reads the data value from a specified GPIO port.

Function Parameters

Name	Description
Port	0-1 for port A, B
data	read value.

Return Value
Error code or 0.

Usage Example
To read GPIO port.

```
inport=0;//port A
JNMMGPIORead( inport, &input_byte );
Printf("data=%d",*input_byte);
```

5.10 JNMMGPIOWrite

Function Definition
BYTE JNMMGPIOWrite(int Port, int data);

Function Description
This function writes a data value to a specified GPIO port.

Function Parameters

Name	Description
Port	0-1 for port A, B
data	Value.

Return Value
Error code or 0.

Usage Example
To write a data on port A

```
outport=0;//port A
output_byte=20;
JNMMGPIOWrite( outport, output_byte );
Printf("written data=%d",output_byte);
```

5.11 JNMMGPIOReadBit

Function Definition
BYTE JNMMGPIOReadBit(int Port, int Bit,int *data);

Function Description
This function outputs the desired value to a single bit. The other bits remain at their current values

Function Parameters

Name	Description
Port	0-1 for port A, B
Bit	DIO line number, 0-15
Data	read value.

Return Value
Error code or 0.

Usage Example
To read single bit.

```
port=0;//port A
bit=6;//6th bit
JNMMGPIOReadBit(port,bit,&digital_value);
```

5.12 JNMMGPIOWriteBit

Function Definition
BYTE JNMMGPIOWriteBit(int Port,int Bit, int data);

Function Description
This function reads in the specified bit and returns it in the location specified by the pointer to data.

Function Parameters

Name	Description
Port	0-1 for port A, B
Bit	DIO line number, 0-15
Data	Value.

Return Value
Error code or 0.

Usage Example
To write a single bit.

```
sscanf ( input_buffer, "%d", &intBuf );
digital_val = intBuf;
JNMMGPIOWriteBit(port,bit,digital_val);
```

6. JNMM-4LP-XT DRIVER APPLICATION DESCRIPTION

- DIO
- JNMMCANLoopBack
- JNMMCANDemo
- JNMMRWFunctions
- JNMMFPGAVersion

6.1 DIO

Digital I/O supports four types of direct digital I/O operation: input bit, input byte, output bit, and output byte.

6.2 JNMMCANLoopBack

User to select the CAN Port (0-3) in transmit or receive mode.

It will perform the continuously transmit the data to selected CAN (0-3) and will receive the data from the selected CAN (0-3)

6.3 JNMMCANDemo

User to select the CAN Port (0-3) in transmit or receive mode

It will perform the user to enter the transmit the data to the selected CAN Port (0-3) and will receive the data from the selected CAN(0-3)

6.4 JNMMRWFunctions

User to select the CAN Port (0-3) for read/write the device register

It will perform the user to read the device register from the selected CAN Port (0-3) and will write the data to the device register from the selected CAN(0-3)

6.5 JNMMFPGAVersion

It will read the updated FPGA/board version for JNMM-CAN4LP-XT board.

7. JNMM-4LP-XT DRIVER APPLICATION USAGE INSTRUCTIONS

The following section details about how to use the JNMM-4LP-XT Driver applications and to confirm the output which may be obtained through the application.

7.1 DIO Description

The DIO application provides various operations on a DIO channel such as input byte, output byte, input bit, output bit, and DIO loopback. This section describes the input byte and output byte DIO operations. The DIO port must be configured in either input or output mode based on the DIO operation to be performed.

Output Byte:

- Select Write a Byte to port option from main menu
- Enter value 0-255 or q to quit

To set Port 0 all the pins to high except pin 3 and pin 7, run the DIO application and provide input as follows:

- Select Write a Byte to port option from main menu :4
- Enter value 0-255 or q to quit: 119

The Byte value 119 is sent to port 0.

Measure the voltage on all pins of Port 0 using a multi-meter. They should show 3.3v on all the pins except pin 3 and pin 7, the application generated expected voltages.

Input Byte:

- Select Read a Byte from port option from main menu:
- Enter port number (0-2):
- Press ENTER key to stop reading

Provide 3.3V to Port 0 pin 0 from VCC. It should read and display 0x01. To see the output, run the DIO application and provide input as follows:

- Select Read a Byte from port option from main menu:1
- Enter port number (0-2):0

The application should show 0x01 on the screen.

7.2 JNMMCANLoopBack Description

JNMMCANLoopBack demo application will provides the user to enter the mode of operation for CAN (0-3) whether in transmit or receive mode.

1. For ISA mode : Enter the Base address and DIO address(PCI not required)
2. Enter the CAN Port number(0-3).
3. Enter the baud rate for CAN Port(0-3).
4. Enter the mode of operation for CAN Port(0-3) : Tx(0)/Rx(1).

Enter the mode of operation for particular CAN for RX mode.
Mode of operation for selected CAN Port(0-3)

Rx mode:

1. Press any key to terminate the application

TX mode:

1. Enter the message type(0-STD,1-EXT).
2. Enter the Transfer buffer type(0-PTB,1-STB)
3. Enter the message ID.
4. Enter the RTR.(0-Disable,1-Enable)
5. Enter the data length.
6. Enter the Tx data.
7. Press “Q” to terminate application.

7.3 JNMMCANDemo description

JNMMCANDemo demo application will provides the user to enter the mode of operation for CAN (0-3) whether in transmit or receive mode.

Enter the mode of operation for particular CAN for RX mode.

TX mode:

1. For ISA mode : Enter the Base address and DIO address(PCI not required)
2. Enter the CAN Port number (0-3).
3. Enter the baud rate for CAN Port (0-3).
4. Enter the mode of operation for CAN Port (0-3): TX(0).
5. Enter the message type (0-STD, 1-EXT).
6. Enter the Transfer buffer type(0-PTB,1-STB)
7. Enter the message ID.
8. Enter the RTR.(0-Disable,1-Enable)
9. Enter the data length.
10. Enter the Tx data.
11. Press enter key to repeat or 'q' to quit.
12. If enter key to repeat, it will start from step 2
13. If user entered 'q' means it will exit the application.

Rx mode:

1. For ISA mode : Enter the Base address and DIO address(PCI not required)
2. Enter the CAN Port number (0-3).
3. Enter the baud rate for CAN Port (0-3).
4. Enter the mode of operation for CAN Port (0-3): Rx(1).
5. It will display the received data in terminal and press enter key to repeat or 'q' to quit.
6. If enter key to repeat, it will start from step 2
7. If user entered 'q' means it will exit the application.

7.4 JNMMRWFunction description

JNMMRWfunctions demo application will provides the user to to enter for read and write operation to access the device register value for CAN Port(0-3)

READ MODE :

1. For ISA mode : Enter the Base address and DIO address(PCI not required)
2. Enter the CAN Port number (0-3).
3. Enter the Offset to read from the device register
4. Press "Q" to exit the operation or enter to continue
5. Repeat the from the step 1

WRITE MODE :

1. For ISA mode : Enter the Base address and DIO address(PCI not required)
2. Enter the CAN Port number (0-3).
3. Enter the Offset to write to the device register
4. Enter the data to write into the device register
5. Press "Q" to exit the operation or enter to continue
6. Repeat the from the step

8. INTERFACE CONNECTOR DETAILS

8.1 Digital I/O – J4

The sixteen digital I/O lines are brought out on a 20-pin pin header, J4, with the pin out shown below.

Digital Connector Pinout

DIO A0	**1**	**2**	DIO A1
DIO A2	**3**	**4**	DIO A3
DIO A4	**5**	**6**	DIO A5
DIO A6	**7**	**8**	DIO A7
DIO B0	**9**	**10**	DIO B1
DIO B2	**11**	**12**	DIO B3
DIO B4	**13**	**14**	DIO B5
DIO B6	**15**	**16**	DIO B7
V fused	**17**	**18**	V fused
Ground	**19**	**20**	Ground

Connector Part Number / Description

OUPIIN 2112-2210G00R 2mm pitch dual row right angle pin header with ejector latches

8.2 CAN(J5, J6, J7, J8)

Each of the four CAN ports has its own 4-pin latching connector with the same pinout as shown below. These connectors are located on the right side of the board. CAN1 (J5) is the topmost connector and CAN4 (J8) is the bottommost connector.

1	Ground Iso
2	CAN L
3	CAN H
4	Ground Iso

Connector Part Number / Description

JST SM04B-GHS-TB 4 pos, 1.25mm, right angle, latching, SMD

APPENDIX: REFERENCE INFORMATION

http://www.diamondsystems.com/products/janusmm4lp