DLL Software Manual MSSL32
DLL Device Driver

Version 2.0 March 2002

Documentation History

Date	Version	Change/Description
2002/03/05	2.0	Added LevelX Interface

Copyright

Disclaimer

The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Brand Innovators of Digital Products bv makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchant ability or fitness for any particular purpose. Furthermore, Brand Innovators of Digital Products bv reserves the right to make changes to any product herein to improve reliability, function or design, without obligation of Brand Innovators of Digital Products bv to notify any person of such revision or changes. Brand Innovators of Digital Products bv does not assume any liability arising out of applications or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

This manual describes the usage and specification of the MSSL32.

1.2 Installation

1. Install the ISA Card. The port address should be set to $200. No interrupts need to be enabled. A description of the hardware can be found in Hardware Manual AT CAN Mini Card.
2. Install the DriverLINX Port I/O Driver. This is needed to access the ISA Card CAN Interface.
3. Install the PCI Card. There are no hardware jumpers to set on the PCI card. A description of the hardware can be found in Hardware Manual PCI Intellican Card.
4. Install the IntelliCAN Driver. This is needed to access the PCI Card CAN Interface.
5. Copy the MSSL DLL to WinNt/system32/mssl32.dll. This DLL provides the generic entries to the different CAN Interfaces.
6. Install the MSSL Test Utility. This utility can be used to check if the installation of the hardware and software was successful.

1.3 Overview

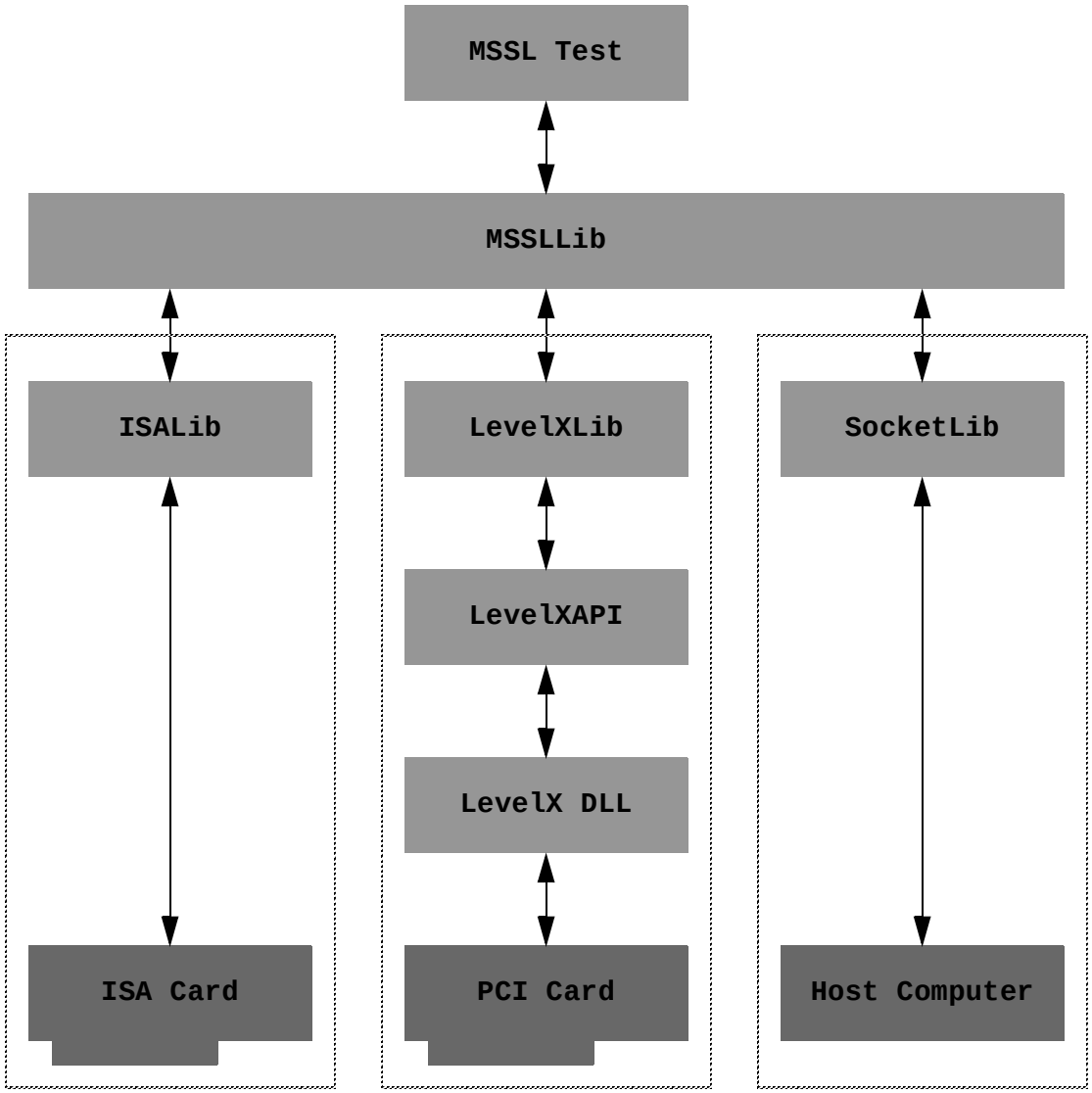

Figure 1-1 Overview

Chapter 2

DLL Usage

2.1 Introduction

This chapter explains the usage of the MSSL DLL.

2.2 Return Codes

The following table gives the return codes of the DLL functions.

Table 2-1 Return Codes

Mnemonic	Value	Description
rcDeviceOutOfRange	-4	device index out of range
rcDeviceNotFound	-3	device not found
rcDeviceNotAssigned	-2	device not assigned
rcDeviceAssigned	-1	device already assigned
rcSuccessfull	0	device found and initialized
rcRxEmpty	1	receive buffer empty
rcTxFull	2	transmit buffer full

2.3 State Flags

The following table gives the State Flag values.

Table 2-2 State Flags

Mnemonic	Value	Description
sfBusOff	$00000001	hardware bus off
sfError	$00000002	hardware error, can not be cleared
sfOverrun	$00000004	hardware receiver overrun
sfRxOverrun	$00000010	software receive buffer overrun

2.4 DLL Function Calls

The DLL Functions must be called by name.

2.4.1 GetVersion

Parameters: none

Returns: Word

This function returns the version of this DLL as $0200.

2.4.2 FindISACard

Parameters: Device: Longword; ABaseAddress, ABaudrate: Word

Returns: Longint

This function tries to find a CAN ISA Card at the specified address. The CAN baud rate will be set to the requested value in kBits. Only 500 kBit and 1 MBit are allowed. The baud rate is specified as 500 or 1000.

The Device value should be 0 to 3. This Device value should be used for subsequent calls to the ISA Card. This function returns a Return Code as specified earlier.

2.4.3 FindLevelXCard

Parameters: Device: Longword; DLLName: PChar; ABaudrate: Word

Returns: Longint

This function tries to find a LevelX Card using the specified DLL Name. The CAN baud rate will be set to the requested value in kBits. Only 500 kBit and 1 MBit are allowed. The baud rate is specified as 500 or 1000.

The Device value should be 0 to 3. This Device value should be used for subsequent calls to the LevelX Card. This function returns a Return Code as specified earlier.

2.4.4 FindSocket

Parameters: Device: Longword; AHost: PChar; APort: Word

Returns: Longint

This function tries to find the Host machine using the specified Host Name and Port.

The Device value should be 0 to 3. This Device value should be used for subsequent calls to the Socket. This function returns a Return Code as specified earlier.

2.4.5 GetState

Parameters: Device: Longword; var State: Longword

Returns: Longint

This function will fill the parameter State to the current state of this Device. This function returns a Return Code as specified earlier.

2.4.6 ClrState

Parameters: Device: Longword; Mask: Longword

Returns: Longint

This function will clear the states of this Device as specified by the parameter Mask. This function returns a Return Code as specified earlier.

2.4.7 ReadMsg

Parameters: Device: Longword; var Rec

Returns: Longint

This functions tries to read a CAN message from the specified Device. The received message will be stored in the Rec, which layout is specified in TMessageRec. This function returns a Return Code as specified earlier.

2.4.8 WriteMsg

Parameters: Device: Longword; var Rec

Returns: Longint

This functions tries to send a CAN message using the specified Device. The message to be sent must be contained in the Rec, which layout is specified in TMessageRec. This function returns a Return Code as specified earlier.

2.4.9 Close

Parameters: Device: Longword

Returns: Longint

The specified Device will be closed. The Device value should be 0 to 3. This function returns a Return Code as specified earlier.

Chapter 3

MSSL Library

3.1 Introduction

In this chapter an overview of the DLL for the MSSL32 is given.

3.2 Overview

TISADevice.Create calls TDevice.Create calls TThread.Create.

TISADevice.Destroy calls TDevice.Destroy calls TThread.Destroy.

3.3 TMessageRec

This record holds an individual CAN message.

```
TMessageRec = record
        Identifier: Word;
        RTR: Bytebool;
        DataLen: Byte;
        Data: TDataArray;
```

3.4 TMessageBuffer

This class provides the means to exchange messages between the DLL thread, called by the user program and the Device thread.

```
TMessageBuffer = class
procedure Add(const Rec: TMessageRec);
procedure Delete;
function First: TMessageRec;
function Count: Integer;
```

3.5 TDevice Class

This class is used as a parent for the different Devices that will be supported. Existing and new MSSL interfaces will have the following entries.

TDevice = class(TThread). This class inherits from the TThread class, therefore execution of the Device Interface code is not dependent on the cooperation of the application calling this DLL.

constructor Create(CreateSuspended: Boolean); This entry is called from the Device in the Create call. This entry first calls the inherited Create of the TThread, then creates the MessageBuffers and clears the State.

destructor Destroy; override; This entry destroys the MessageBuffers and then calls the inherited Destroy of TThread.

function ReadMsg(var Rec): Longint; This entry gets a Message from the MessageBuffer, when available. This entry is used to communicate with the running thread of the Device.

function WriteMsg(var Rec): Longint; This entry stores a Message in the MessageBuffer, when possible. This entry is used to communicate with the running thread of the Device.

function GetState: Longword; virtual; abstract; This entry is defined by the Device.

procedure ClrState(Mask: Longword); virtual; abstract; This entry is defined by the Device.

Chapter 4

ISA Card Interface

4.1 Introduction

4.2 Installation

The port address should be set to $200. No interrupts need to be enabled.

4.3 Device Library

```
TISADevice = class(TDevice)
public
constructor Create(ABaseAddress, ABaudrate: Word);
destructor Destroy; override;
function GetState: Longword; override;
procedure ClrState(Mask: Longword); override;
```

Chapter 5

LevelX Interface

5.1 Introduction

This library enables access to all LevelX interfaces.

The DLLName parameter must be set for the specific interface card.

- PCI Card: DLLName = LXN4pi2j

5.2 Installation

There are no hardware jumpers to set on the PCI card.

5.3 Device Library

TLevelXDevice = class(TDevice)

public

constructor Create(DLLName: PChar; ABaudrate: Word);

destructor Destroy; override;

function GetState: Longword; override;

procedure ClrState(Mask: Longword); override;

Chapter 6

Socket Interface

6.1 Introduction

6.2 Installation

6.3 Device Library

```
TSockDevice = class(TDevice)
public
constructor Create(AHost: PChar; APort: Word);
destructor Destroy; override;
function GetState: Longword; override;
procedure ClrState(Mask: Longword); override;
```

Chapter 7

MSSL Test Program

7.1 Introduction

The MSSL Test program is able to connect to the ISA Card Interface, LevelX Interface and the Socket Interface. More than one interface may be opened and used at the same time, as long as unique Device numbers are used for each opened interface. The CAN baud rate is always set fixed to 500 kBit per second according to the MSSL standard.

The Connect button tries to open the interface. When the interface is successfully opened, the Device identifier is associated with this device and used with subsequent Read and Write actions.

The Read button will try to get a CAN message from the interface. When a CAN message is available and successfully received, the Received Check button is checked. The received message is displayed in the ModuleID, RTR and Data fields.

The Write button tries to send a CAN message to the interface. The CAN message is constructed from the ModuleID, RTR and Data fields. Hexadecimal values should be preceded by 0x.

The Close button closes the interface.

7.2 ISA

Two parameters can be set before opening the ISA card.

Base Address: this is the address at which the ISA card should be found.

Device Number: this is the identifier for this connection.

7.3 LevelX

Two parameters can be set before opening the LevelX card.

DLL Name: this is the identifier for the LevelX product to find.

Device Number: this is the identifier for this connection.

7.4 Socket

Three parameters can be set before opening the Socket.

Port Number: this is the TCP/IP port number to use.

Host Name: this is the IP address or Host name to use.

Device Number: this is the identifier for this connection.