SST DeviceNet CAN 2.0A Module

Firmware/Windows DLL Reference Guide

Document Edition: 1.1

Document #: 717-0026

Document Edition: 1.1

Date: November 3, 2010

This document applies to the SST DeviceNet CAN 2.0A Module and 32-Bit DLL API.

Copyright ©2010 Molex Incorporated

This document and its contents are the proprietary and confidential property of Woodhead Industries Inc. and/or its related companies and may not be used or disclosed to others without the express prior written consent of Woodhead Industries Inc. and/or its related companies.

SST is a trademark of Woodhead Industries Inc. All other trademarks belong to their respective companies.

At Woodhead, we strive to ensure accuracy in our documentation. However, due to rapidly evolving products, software or hardware changes occasionally may not be reflected in our documents. If you notice any inaccuracies, please contact us (see Appendix A).

Written and designed at Woodhead Software & Electronics, 50 Northland Road, Waterloo, Ontario, Canada N2V 1N3.

Hardcopies are not controlled.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Contents

Preface ..... **vii**

Purpose of this Guide viii

Conventions viii

Style viii

Special Terms ix

Special Notation ix

Introduction **11**

1.1 CAN2A Overview 12

1.2 Distribution Files 12

The CAN Network **13**

2.1 Introduction 14

2.2 Bus Arbitration 14

2.3 CAN 2.0A 15

2.4 CAN 2.0B 15

2.5 Mixed CAN 2.0A & 2.0B Networks 15

CAN2A Firmware Operation **17**

3.1 Module Features 18

3.2 CAN Compatibility 18

3.3 Bus Control /Status 19

3.3.1 Bus Off Recovery 19

3.4 Transmit Queue 19

3.4.1 Timestamped Transmit 19

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3.4.2 Timestamp 19
3.5 Receive Queue 20
3.5.1 Receive Timestamp 20
3.5.2 Receive Filter 21
3.5.3 COMM LED 21

Shared Memory Interface 23
4.1 Introduction 24
4.2 Host Interface Memory 24
4.3 Application Module Header 25
4.3.1 IRQ Control / Status 27
4.3.2 CAN Bus Status Word (0030h) 29
4.4 Application Host Interface 31
4.4.1 Command Word (0080h) 31
4.4.2 Status Word (0082h) 33
4.4.3 Receive Filter 33
4.4.4 Transmit & Receive Queue Format 34
4.4.5 Receive Queue 34
4.4.6 Transmit Queue 35

DeviceNet CAN 2.0A Module 32-Bit DLL 37
5.1 Introduction 38
5.2 Services 38
5.3 Application Stack 38

DeviceNet CAN 2.0A DLL API 39
6.1 Introduction 40
6.2 API Interface 41
6.2.1 C2A_CAN_COUNTERS Data Type 41
6.2.2 C2A_CloseCard 42
6.2.3 C2A_Driver 43
6.2.4 C2A_EnumDrivers 44
6.2.5 C2A_FreeDriver 45
6.2.6 C2A_GetBusStatus 46
6.2.7 C2A_GetCANCounters 47
6.2.8 C2A_GetCardStatus 48
6.2.9 C2A_GetModuleHeader 49
6.2.10 C2A_LoadDriver 50
6.2.11 C2A_MESSAGE Data Type 51
6.2.12 C2A_ModifyFilter 52
6.2.13 C2A_MODULE_HEADER Data Type 53
6.2.14 C2A_Offline 54
6.2.15 C2A_Online 55
6.2.16 C2A_OpenCard 57

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.17 C2A_Receive 60
6.2.18 C2A_RegisterBusStatusEvent....61
6.2.19 C2A_RegisterReceiveEvent....63
6.2.20 C2A_Send 64
6.2.21 C2A_SetAccessTimeout 66
6.2.22 C2A_SetEventNotificationInterval 68
6.2.23 C2A_TriggerTX....69
6.2.24 C2A_TXQueueEmpty....70
6.2.25 C2A_UnRegisterBusStatusEvent....71
6.2.26 C2A_UnRegisterReceiveEvent....72
6.2.27 C2A_VerifyCardHandle....73
6.2.28 C2A_Version....74
6.2.29 C2A_VersionEx 75
6.2.30 C2A_WriteTXQueue 77

Warranty and Support.... 79
A.1 Warranty 80
A.2 Technical Support....80
A.2.1 Getting Help 80

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Preface

Preface Contents:

- Purpose of this Guide
- Conventions

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Purpose of this Guide

This document is a reference guide for the DeviceNet™ CAN 2.0A Module firmware (CAN2A), an application module for the SST-DN family of interface cards.

Conventions

This guide uses stylistic conventions, special terms, and special notation to help enhance your understanding.

Style

The following stylistic conventions are used throughout this guide:

Bold	indicates field names, button names, tab names, and options or selections
Underlining	indicates a hyperlink
Italics	indicates keywords (indexed) or instances of new terms and/or specialized words that need emphasis
CAPS	indicates a specific key selection, such as ENTER, TAB, CTRL, ALT, DELETE
`Code Font`	indicates command line entries or text that you would type into a field
“>” delimiter	indicates how to navigate through a hierarchy of menu selections/options

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Special Terms

The following special terms are used throughout this guide:

SST-DN Card	SST interface cards/products within the SST-DN3 and SST-DN4 families
Firmware	the software running on the card
Module	a synonym for *firmware*
CAN2A	the DeviceNet CAN 2.0A Module

Special Notation

The following special notations are used throughout this guide:

Note

A note provides additional information, emphasizes a point, or gives a tip for easier operation. Notes are accompanied by the symbol shown, and follow the text to which they refer.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

1

Introduction

Chapter Contents:

- CAN2A Overview
- Distribution Files

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

1.1 CAN2A Overview

The CAN2A Module allows an SST-DN card to provide basic access to the CAN network using the 11-bit identifier, as defined in the CAN Specification, Version 2.0, Part A.

1.2 Distribution Files

CAN2A is shipped with the following distribution files:

CAN2A.SS3

This file contains the SST CAN 2.0A module firmware for use with the SST-DN3 family of DeviceNet interface cards.

CAN2A.SS4

This file contains the SST CAN 2.0A module firmware for use with the SST-DN4 family of DeviceNet interface cards.

CAN2A_USB.SS4

This file contains the SST CAN 2.0A module firmware for use with the SST-DN4-USB DeviceNet USB Interface.

SSC2A32.DLL

This file contains the exported APIs (documented in Section 6, DeviceNet CAN 2.0A DLL API).

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

2

The CAN Network

Chapter Contents:

- Introduction
- Bus Arbitration
- CAN 2.0A
- CAN 2.0B
- Mixed CAN A & B Networks

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

2.1 Introduction

The CAN network is a high-speed bus, capable of speeds up to 1 Mbps. The basic protocol provides:

- Maximum of 8 data bytes per message (packet)
- Bus arbitration (determines which station transmits next)
- Comprehensive error detection (checks transmissions and receptions)
- Fault confinement (disconnects faulty stations)

CAN does not require station (node) numbers. All stations receive all messages, discarding or filtering out those that aren't required. This can be compared to the global data transfer provided by many industrial networks.

Two variations of the CAN protocol are currently being used; both are defined in the CAN Specification, Version 2.0.

2.2 Bus Arbitration

CAN uses a non-destructive bitwise arbitration scheme. This means that all stations may begin to transmit whenever the bus is free, but only the highest priority message will complete. The message priority is based on the identifier, which is transmitted first.

The CAN protocol's physical layer requires that the electrical interface have two levels: dominant (0), and recessive (1). Any station transmitting a dominant level overrides any stations transmitting a recessive level.

Each station monitors its own transmission during arbitration, and if it transmits a recessive level (1) and reads back a dominant (0) level, it has lost arbitration and stops transmitting immediately.

Stations that lose arbitration will try again after the bus is free (after the higher-priority message is transmitted).

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

2.3 CAN 2.0A

Part A of the CAN Specification describes a protocol that uses an 11-bit value to identify each message on the wire. This allows for 2032 different message identifiers (16 potential identifiers are defined as invalid). During arbitration, message priority is determined by the message identifier, with 0 being the highest priority.

2.4 CAN 2.0B

Part B of the CAN Specification describes a protocol that uses the basic 11-bit identifier described in Part A, and defines an additional 18-bit identifier. This (29 bits in total) identifier allows for 532,676,608 different message identifiers. During arbitration, message priority is determined by the message identifier, with 0 being the highest priority.

2.5 Mixed CAN 2.0A & 2.0B Networks

CAN 2.0B messages may be transmitted on the same network as CAN 2.0A messages.
The stations configured for CAN 2.0A messages ignore CAN 2.0B messages and vice-versa.

Bus arbitration in a mixed network can be divided into three steps:

1. The messages are arbitrated, based on the 11-bit portion of the identifier (most significant 11 bits of a 29-bit identifier). This step eliminates all lower-priority 11-bit messages, but may pass one 11-bit message and multiple 29-bit messages. All of these messages will have an identical 11-bit identifier.

2. If a 29-bit message and an 11-bit message both pass step 1, the 11-bit message loses arbitration.

3. The remaining 29-bit messages are arbitrated on the remaining 18 bits, with 0 having the highest priority.

This arbitration method provides a flexible priority scheme independent of message type.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3

CAN2A Firmware Operation

Chapter Contents:

- Module Features
- CAN Compatibility
- Bus Control /Status
- Transmit Queue
- Receive Queue

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3.1 Module Features

CAN2A provides a low-level interface to the CAN bus. The main features of the module are:

- Maximum of 254 messages stored in the circular transmit queue, and up to 128 more stored internally
- Receive filter with individual message identifier resolution
- Maximum of 743 messages stored in the circular receive queue, and up to 128 more stored internally
- Message time stamping for receive and synchronous transmit
- Asynchronous and synchronous transmission modes
- On network initialization, predefined baud rates of 125K, 250K, 500K and 1M
- Support for custom bit timing parameters
- Bus statistics counters
- 3 logical interrupt sources: host command, bus status change and message received

3.2 CAN Compatibility

CAN2A is compatible with the CAN 2.0A Specification, with the following exceptions:

- CAN remote frames are not supported
- Message priority has no effect on transmit queue. Messages are transmitted in the order they are placed into the transmit queue.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3.3 Bus Control /Status

A command interface is used to allow the host application to initialize and shut down the CAN interface. Bus status is maintained in the Application Module Header and may be read by the host application at any time.

3.3.1 Bus Off Recovery

If the CAN interface detects excessive errors on the bus, it enters a Bus Off state. This may be caused by a faulty transceiver, incorrect wiring, or an incorrect baud rate. Recovery from the bus off condition is performed by re-initializing the CAN interface (taking the card offline and putting it back online).

3.4 Transmit Queue

A circular queue provides the interface between the host application and the CAN transmitter. Messages are placed into the queue by the host application and removed by CAN2A. An interrupt is required to transmit any messages you have placed in the transmit queue, and messages are transmitted in the order that they were placed into the queue.

3.4.1 Timestamped Transmit

In Timestamped Transmit mode, messages in the transmit queue are timestamped upon successful transmission. This mode causes a slightly slower transmission to occur.

3.4.2 Timestamp

After being transmitted in Timestamped Transmit mode, the message that was placed in the queue is timestamped, and may be extracted from the queue in order to use the timestamp information. It is recommended that a local pointer to the oldest timestamped message be used to prevent the new messages placed in the queue from wrapping around and overwriting the timestamped messages before they have been read.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3.5 Receive Queue

A circular queue and receive filter provide the interface between the host application and the CAN receiver.

Messages received from the CAN bus are first checked against the filter. If the message is enabled (meets the filter criteria), it is placed into the receive queue. Messages must be removed from the receive queue by the host application. An interrupt from CAN2A to the host application indicates that messages have been placed into the queue.

Messages are removed from the receive queue in the order they were received from the bus.

3.5.1 Receive Timestamp

Messages in the receive queue are timestamped.

Note

Calculation of the TimeStamp can be done using the following formula:

TimeStamp (ms) = (TimeH * MajorTickInterval + ((MajorTickInterval / MinorTickCount) * (MinorTickCount - TimeL)))

See Section 4.3, Application Module Header, for a description of the MajorTickInerval and MinorTickCount.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

3.5.2 Receive Filter

The receive filter is implemented as an array of 2032 bits. Each bit corresponds to a CAN message identifier. Only messages received from the bus that have the corresponding filter bit set are placed into the receive queue.

Note

As illustrated in Section 4.4, Application Host Interface, the receive filter is actually composed of 2048 bits. The last 16 identifiers are invalid and are not used.

3.5.3 COMM LED

Indicates the communication status of the CAN2A application.

COMM Led State	Meaning
Off	Offline/Online (no network activity)
Solid Green	Network activity detected
Solid Red	Bus Off

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4

Shared Memory Interface

Chapter Contents:

- Introduction
- Host Interface Memory
- Application Module Header
- Application Host Interface

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.1 Introduction

The interface between CAN2A and the host application is implemented using shared memory and one interrupt signal in each direction. The following sections describe the nature of this interface.

4.2 Host Interface Memory

The SST-DN card uses 16 Kbytes of shared RAM to communicate with the host application. The layout of this memory is illustrated in the following table.

Address		Description
Hex	**Decimal**	
00000h 03FFFh	0 16383	Not used at runtime
04000h 0407Fh	16384 16511	Application Module Header
04080h 07FFFh	16512 32767	Application Host Interface
08000h 3FFFFh	32768 262143	Not used at runtime

Each type of SST-DN card has 256K of RAM conforming to this memory map. However, the way this memory is mapped into the host system's address space different for each type of card. Please refer to the Hardware Reference Guide for memory access details.

Note

Only 16K of the 256K memory (Host Interface Block) is used for the runtime interface between the card and the host. The remainder (shaded in the above table) should not be accessed at runtime.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.3 Application Module Header

Each application for the SST-DN card is based on an event-driven kernel. This kernel provides low-level hardware interface, startup self-diagnostics, and common services such as timers and event management.

The kernel reserves the first 128 bytes of the Host Interface Block for loader interface and run-time status information. The majority of this header area is defined and maintained by the application kernel.

Data Type Descriptions

Data Type	Description
CHAR	8-bit ASCII character
UINT1	Unsigned integer, 1 byte
SINT1	Signed integer, 1 byte
UINT2	Unsigned integer, 2 bytes
SINT2	Signed integer, 2 bytes
UINT4	Unsigned integer, 4 bytes

Byte Ordering

The SST-DN cards use Intel-style byte ordering for multi-byte entities (LSB - low address, MSB - high address). If your host system uses Motorola byte ordering (MSB - low address, LSB - high address) you must compensate for the byte ordering in software.

The following macro will compensate for byte ordering in a 16 bit data entity:

```
#define SWAP_WORD (WordData)  ((WordData<<8) | (WordData>>8))
```


©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

The following table provides information on the application-specific portions of the header. All offsets in the table are from the start of the Host Interface Block. Unshaded items are common to all SST-DN application modules. Only the shaded items have certain attributes specific to the CAN2A application module.

Offset	Name	Data Type	Description
0000h	ModuleType	CHAR[2]	Contains "DN" (0444eh) or "ER" (04552h) if a kernel error is detected.
0002h	WinSize	UINT2	Set by loader to indicate host interface window size. 0 = 16K
0004h	CardId	UINT2	Reserved for use by host software.
0006h	Kernel Id	UINT2	Kernel identification. 0x01 = CAN 2.0A kernel
0008h	Kernel Rev	UINT2	Kernel Revision.
000ah	ModuleId	UINT2	Module ID. 0x11 = CAN2A
000ch	ModuleRev	UINT2	Module revision. 4 hex digits XX.XX (i.e. rev 1.0 = 0x0100)
000eh	NetSerial	UINT4	DeviceNet serial number.
0012h	CardType	CHAR[16]	Card type (i.e. "SST-DN3").
0022h	CardSerial	CHAR[8]	Card serial number. (i.e. "9409001")
002ah	IrqControl	UINT2	Card interrupt control word. See Section 4.3.1.
002ch	IrqStatusA	UINT1	Card interrupt status.
002dh	IrqStatusB	UINT1	See Section 4.3.1.
002eh	MainCode	UINT2	Main Application Error Code.
0030h	CanStatus	UINT2	CAN bus status word. See Section 4.3.2.
0032h	CanTx	UINT2	CAN transmit counter. Incremented when messages are submitted to the CAN controller.
0034h	CanAck	UINT2	CAN ack error counter. Incremented when a transmit message is aborted due to lack of acknowledgment from other stations. When CanAck is incremented, CanTx is decremented to compensate for a message not actually transmitted.
0036h	CanRx	UINT2	CAN receive counter. Incremented when messages are received. Messages that fail the receive filter still increment CanRx.
0038h	CanError	UINT2	CAN communication error counter. Incremented when a CAN frame error is detected.
003ah	CanLost	UINT2	CAN lost messages counter. Incremented when a CAN message is received before the previous message is placed into the receive queue.
003ch	CanOverrun	UINT2	CAN receive queue overrun counter. Incremented when a CAN message is lost due to a full receive queue.
003eh	AddCode	UINT2	Additional Application Error Code.
0040h	Message	CHAR[60]	When ModuleType is "DN", contains the module identification string. When ModuleType is "ER", contains the kernel error string.
007ch	MajorTickInterval	UINT2	Major Tick Interval (equivalent of system timebase).
007eh	MinorTickCount	UINT2	Number of minor ticks per major tick interval.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.3.1 IRQ Control / Status

The IRQ control and status areas are used to implement up to 8 logical interrupts using only 1 physical interrupt signal.

4.3.1.1 IRQ Control Word (002Ah)

The IRQ Control Word contains 8 bits, which are used to enable the generation of a physical interrupt for each of up to 8 logical interrupt sources. This word is written by the host and not written by the scanner.

Offset	Bit							
	7	6	5	4	3	2	1	0
002Ah	Reserved					RX	BS	CM
002Bh	Reserved							

RX Message receive interrupt
BS CAN bus status (CanStatus, offset 30h) change interrupt
CM Command acknowledge interrupt

4.3.1.2 IRQ Status Byte A/B (002Ch / 002Dh)

The IRQ Status Bytes each contain 8 bits, which are set when each of up to 8 logical interrupts occur. The IRQ Status Bytes reflect logical interrupts even if the physical interrupt has been disabled via the IRQ Control Word.

These bytes are written by the module and the host application. The IRQ access protocol described below must be followed to avoid missed interrupts.

Offset	Bit							
	7	6	5	4	3	2	1	0
002Ch	Reserved					RX	BS	CM
002Dh	Reserved					RX	BS	CM

RX Message receive interrupt
BS CAN bus status (CanStatus, offset 30h) change interrupt
CM Command acknowledge interrupt

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.3.1.3 IRQ Status Access Protocol

To avoid inadvertently clearing interrupt status flags, the host application must:

1. Read IrqStatusA and store the result in temporary variable ‘A’.
2. Clear relevant bits in IrqStatusA.
3. Read IrqStatusB and store the result in temporary variable ‘B’.
4. Clear relevant bits in IrqStatusB.
5. Logically OR temporary variables ‘A’ and ‘B’ to determine IRQ status flags.

Note

The only possible result of an access collision between the module and the host application is an interrupt status flag remaining set after the host has attempted to clear it.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.3.2 CAN Bus Status Word (0030h)

The CAN Bus Status Word contains bit flags that indicate the current status of the CAN bus interface and low-level interrupt handlers. When the CAN Status word changes, a Bus Status interrupt is generated, providing it is enabled in the IRQ Control Word.

Offset	Bit							
	7	6	5	4	3	2	1	0
0030h	ML	RO	TO	TA	A	BO	BW	OL
0031h	Reserved						BP	ER

4.3.2.1 BP - Bus Power Detect

The BP bit indicates the presence or absence of network power. The BP bit is clear if the physical bus interface in not powered.

4.3.2.2 ER - CAN Bus Error

ER is set each time a CAN communication error is detected.

An excessive number of errors indicates a faulty physical medial component (cable, connector etc.) or excessive noise from external sources (check cable routing and shield connection).

4.3.2.3 ML - Message Lost

ML is set when a message is received from the bus while the previous message is still in the receive buffer.

ML indicates a lower-layer application error (in the kernel interrupt handler). Report this condition to Technical Support.

4.3.2.4 RO - Receive Buffer Overrun

RO is set when messages are received from the bus faster than the application can process them.

RO indicates an upper-layer application error (in the application module).

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.3.2.5 TO - Transmit Timeout

TO is set when a pending transmission is incomplete within 25-50ms.

TO indicates excessive message traffic at a higher priority than the aborted message.

4.3.2.6 TA - Transmit Ack Error

TA is set when a pending transmission is not acknowledged within 25-50ms.

TA indicates that no other nodes are present (or online) on the network.

4.3.2.7 A - Network activity detected

A is set when any message is transmitted or received.

4.3.2.8 BO - Bus off

BO is set when an excessive number of communication errors are detected and the CAN chip automatically goes off-line. BO is cleared when the CAN interface is re-initialized.

BO indicates a serious communication fault such as incorrect baud rate or physical layer error (short, open etc).

4.3.2.9 BW - Bus warning

BW is set when an abnormal number of communication errors is detected and the CAN chip stops transmitting error frames. BW is cleared when the error count returns to normal levels or the CAN interface is re-initialized.

BW indicates a potentially serious communication fault, such as out-of-tolerance baud rate or physical layer error (electrical noise, signal attenuation, intermittent connections, etc.).

4.3.2.10 OL - Online

OL indicates that the CAN interface has been initialized and is ready to communicate.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.4 Application Host Interface

The kernel reserves the first 128 bytes of the Host Interface Block for loader interface and run-time status information. The remainder of the Host Interface Block is defined by the application module.

The following table defines the format of the Host Interface Block's application-specific portion. Note that all offsets are from the start of the Host Interface Block.

Offset	Name	Data Type	Description
0080h	Command	int	Command Word
0082h	Status	int	Status Word
0084h	Wparam	int	Command Parameter
0086h	RxqIn	int	Receive queue in offset
0088h	RxqOut	int	Receive queue out offset
008ah	RxqEnd	int	Receive queue end offset
008ch	TxqIn	int	Transmit queue in offset
008eh	TxqOut	int	Transmit queue out offset
0090h	Txq	MSG[255]	Transmit Queue
1080h	RxFilter	Char[256]	Receive filter (bitmap)
1180h	Rxq	MSG[...]	Receive Queue

See Section 4.4.4 for a description of the MSG structure (used in the queues)

4.4.1 Command Word (0080h)

The Command Word is the main control interface between the host application and CAN2A. Commands are written to the Command Word to initialize and shut down the CAN bus interface.

Sending a Command to CAN2A

To send a command to CAN2A:

1. Clear the Command Acknowledge Interrupt flag (CM) in the Interrupt Status Word (see Section 4.3.1). Make sure the Command Acknowledge Interrupt (CM) is enabled in the Interrupt Control Word (if either a physical interrupt or interrupt polling is used).

2. Write the command to the Command Word and the parameter (if required) to Wparam.

3. Interrupt the card to execute the command. Refer to the Hardware Reference Guide for details on how to interrupt the card.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4. Wait for the CM Interrupt flag to indicate completion of the command. This may be implemented with a physical interrupt, by polling the hardware interrupt flag, or by polling the Interrupt Status Register in the Application Module Header (not recommended).

5. Read the command status from the Status Word.

The following table lists all commands supported by CAN2A.

CAN2A-Supported Commands

Command	Value	Description
INIT125	0x01	(Re)initialize the CAN interface for 125K baud, flush transmit and receive queues.
INIT250	0x02	(Re)initialize the CAN interface for 250K baud, flush transmit and receive queues.
INIT500	0x04	(Re)initialize the CAN interface for 500K baud, flush transmit and receive queues.
INIT1M	0x08	(Re)initialize the CAN interface for 1M baud, flush transmit and receive queues.
INITCUSTOM	0x10	(Re)initialize the CAN interface for a custom baud rate, flush transmit and receive queues. Wparam must contain the baud rate word (see below).
STOPCAN	0x20	Shut down the CAN interface, flush transmit and receive queues.
TXTSTAMP	0x40	Causes messages to be timestamped in the transmit queue after they have been transmitted.

Custom Baud Rate

The INITCUSTOM command initializes the CAN interface to a custom baud rate. Wparam must contain the baud rate word prior to executing the INITCUSTOM command.

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">Sjw</td><td colspan="6">Brp</td><td>S</td><td colspan="3">Tseg2</td><td colspan="4">Tseg1</td></tr>
</table>

SJW Sync jump width (0-3), the actual value is Sjw+1
Brp Baud rate prescaler (0-63), the actual prescaler is Brp+1
S Sample control
0 = sample once per bit time
1 = sample three times per bit time
Tseg2 Tseg2 (1-7), the actual interval is Tseg2+1
Tseg1 Tseg1 (2-15), the actual interval is Tseg1+1

Sjw, Tseg1 & Tseg2 are all in units of time quanta.
One time quantum is 125nS X (Brp+1)
Refer to the CAN Specification for bit timing calculations.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.4.2 Status Word (0082h)

The Status Word contains bit flags that indicate various status items. The first 3 bits (bits 0 - 2) indicate the status of the last command and are cleared or set after each command.

Status	Bit	Description
OK	0	Command complete, no errors.
BADCMD	1	Invalid command.
BADPARM	2	Invalid command parameter.
ONLINE125	3	CAN interface has been initialized to 125K baud. Set by INIT125 cmd, cleared by STOPCAN or INIT? cmd.
ONLINE250	4	CAN interface has been initialized to 250K baud. Set by INIT250 cmd., cleared by STOPCAN or INIT? cmd.
ONLINE500	5	CAN interface has been initialized to 500K baud. Set by INIT500 cmd., cleared by STOPCAN or INIT? cmd.
ONLINE1M	6	CAN interface has been initialized to 1M baud. Set by INIT1M cmd., cleared by STOPCAN or INIT? cmd.
ONLINEC	7	CAN interface has been initialized to custom baud rate. Set by INITC cmd., cleared by STOPCAN or INIT? cmd.
TXTSTAMP	8	Transmission of messages are synchronous, sent messages are timestamped in the transmit queue, and secondary transmit queue is disabled. Can only be set in conjunction with an INIT command
Reserved	9-15	Reserved

4.4.3 Receive Filter

The Receive Filter is an array of bits. Each bit corresponds to a CAN message identifier. When a bit is 1, the corresponding CAN message is enabled and will be placed into the Receive Queue. When a bit is 0, the message is disabled and will be ignored.

The bitmap is packed into bytes in lsb - msb order.

Example

- Bit 0 in byte 0 corresponds to CAN identifier 0
- Bit 7 in byte 3 corresponds to CAN identifier 31
- Bit 7 in byte 253 corresponds to CAN identifier 2031

Note

The Receive Filter is never modified by CAN2A.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

4.4.4 Transmit & Receive Queue Format

The Transmit and Receive Queues are implemented as arrays of message structures. The format of each message structure is:

Offset	Name	Data Type	Description
0	Id	Int	CAN message Identifier
2	Length	Char	# of data bytes in message body
3-10	Data	char[8]	Message body
11-12	TimeL	Int	Timestamp low word
13-14	TimeH	Int	Timestamp high word
15	Filler	Char	Adjusts the message block size to 16 bytes

4.4.5 Receive Queue

The Receive Queue is a circular queue that contains messages received from the CAN bus. Messages are placed into the Receive Queue by CAN2A and removed from the queue by the host application.

The Receive Queue size is determined when the application is loaded, based on the host interface window size. RxqEnd is set to one past the end of available memory. All Receive Queue operations should use the value in RxqEnd as the end of the Receive Queue.

Retrieving a Message from the Receive Queue

To retrieve a message from the receive queue:

1. Examine RxqOut (0088h). If it is equal to RxqIn (0086h), the queue is empty.
2. Using RxqOut as an offset from the card base address, copy the message to a buffer or process it in place. Do not proceed to step 3 until the message is stored or processing is complete.
3. Copy RxqOut to a temporary variable (don't skip this step).
4. Add 16 (10h) to RxqOut(temp) to point to the next message buffer in the receive queue.
5. If RxqOut(temp) is greater than or equal to the value in RxqEnd, set RxqOut(temp) to the value of Rxq (1180h).
6. Write RxqOut(temp) to RxqOut.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Steps 3-5 are critical. If a temporary variable is not used to update RxqOut, the Receive Queue may become corrupted.

Note

The Receive Queue is flushed upon execution of any INIT or STOP commands. See Section 4.4.1 for more details.

4.4.6 Transmit Queue

The Transmit Queue is a circular queue that contains messages to be transmitted on the CAN bus. Messages are placed into the Transmit Queue by the host application and removed from the queue by CAN2A.

The circular Transmit Queue size is fixed at 254 messages.

Placing a Message into the Transmit Queue

To place a message into the Transmit Queue:

1. Using TxqIn as an offset from the card base address, write the message to the card.
2. Copy TxqIn to a temporary variable (don't skip this step).
3. Add 16 (10h) to TxqIn(temp) to point to the next message buffer in the Transmit Queue.
4. If TxqIn(temp) is greater than or equal to 1080h, set TxqIn(temp) to the value of Txq (90h).
5. If TxqIn(Temp) is equal to TxqOut, the queue is full. Skip the remaining steps and take any action necessary to handle the error.
6. Write TxqIn(temp) to TxqIn.
7. Interrupt the card to send the message (see the Hardware Reference Guide).

Steps 3-6 are critical. If a temporary variable is not used to update TxqIn, the Transmit Queue may become corrupted.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Non-Timestamped Transmit Mode

When messages are removed from the Transmit Queue, they are placed into a secondary transmit queue maintained by the application kernel. Do not use removal of a message from the Transmit Queue as a signal that the message has actually been transmitted on the CAN bus.

Timestamped Transmit Mode

Removal of messages from the Transmit Queue in this mode indicates that the message has been sent, and that the message timestamp has been written to the Queue.

Note

The Transmit Queue is flushed upon execution of any INIT or STOP commands. See Section 4.4.1 for more details.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

5

DeviceNet CAN 2.0A Module 32-Bit DLL

Chapter Contents:

- Introduction
- Services
- Application Stack

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

5.1 Introduction

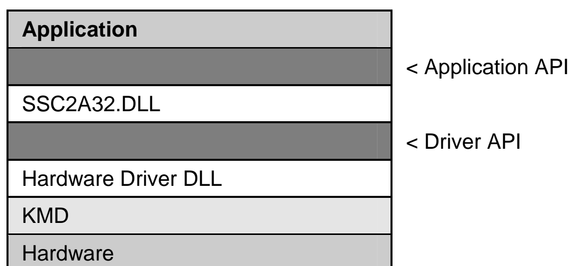
SSC2A32.DLL is a 32-bit DLL that provides a common interface that abstracts physical hardware to allow an application or DLL to interface with the DeviceNet CAN 2.0A module on different hardware platforms.

5.2 Services

SSC2A32.DLL provides the following services:

- DLL revision information
- Hardware driver management
- Card management (multiple cards, multiple clients)
- Card abstraction using handles to eliminate the need for the application to be aware of memory or I/O addresses
- Event notification

5.3 Application Stack



©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6

DeviceNet CAN 2.0A DLL API

Chapter Contents:

- Introduction
- API Reference

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.1 Introduction

This section defines the API *(Application Programming Interface)* for the DeviceNet CAN 2.0A DLL. All functions and data types are listed alphabetically.

Note

For Remote Link Devices (such as the DeviceNet USB Interface), certain API calls can return before the command is actually processed on the remote server, while others require information from the remote server prior to returning. Therefore, the standard SSC2A32 APIs are divided into two groups: “local”, and “remote”.

Remote: the API call will block (does not return) until the command executes and a response is received from the remote server.

Local: the API does not block waiting for a response from the remote server. Data is retrieved from a local buffer, or the parameters of the call can be validated locally.

The SSC2A32 APIs for remote cards will be implemented as either “local” or “remote”, as defined in the following APIs.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2 API Interface

6.2.1 C2A_CAN_COUNTERS Data Type

6.2.1.1 Declaration

```
typedef struct
{
   WORD CanTx;               // CAN transmit counter
   WORD CanAck;              // CAN ack error counter
   WORD CanRx;               // CAN recieve counter
   WORD CanError;            // CAN communictaion error Counter
   WORD CanLost;             // CAN lost messages counter
   WORD CanOverrun;          // CAN receive queue overrun counter
} C2A_CAN_COUNTERS;
```


©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.2 C2A_CloseCard

6.2.2.1 Description

Closes a card connection. The specified card will be shut down and disabled if no other applications are connected.

6.2.2.2 Prototype

```
BOOL WINAPI C2A_CloseCard( DWORD CardHandle )
```

6.2.2.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.2.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.2.5 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout

6.2.2.6 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.3 C2A_Driver

6.2.3.1 Description

Retrieves hardware driver version information. The driver version is returned in both numeric and human-readable string format.

6.2.3.2 Prototype

```
BOOL WINAPI C2A_Driver( TCHAR *Buffer, WORD *Version, DWORD Size )
```

6.2.3.3 Arguments

Argument	Description
Buffer	Driver identification string buffer
Version	Minor revision (LSB) Major revision (MSB)
Size	Buffer size. The Identification string is truncated to fit.

6.2.3.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.3.5 Error Codes

Value	Description
$2000\ 0003_{hex}$	Driver not loaded
$2000\ 0007_{hex}$	Null pointer

6.2.3.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.4 C2A_EnumDrivers

6.2.4.1 Description

Enumerates available driver configurations from the registry, allowing an application to determine the system's available SST-DN card configurations. This function can be called multiple times to determine all SST-DN cards configured on the system.

6.2.4.2 Prototype

```
BOOL WINAPI C2A_EnumDrivers( DWORD Index, char *lpName, DWORD *Len )
```

6.2.4.3 Arguments

Argument	Description
Index	Index number of the driver to enumerate
lpName	The driver name set by the function call
Len	Length of the lpName string

6.2.4.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.4.5 Error Codes

Value	Description
2000 0003h	Driver not loaded
2000 0007h	Null pointer
2000 0100h	General failure

6.2.4.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.5 C2A_FreeDriver

6.2.5.1 Description

Unloads the hardware driver DLL. If no other card connections (CardHandles) are active, the Driver DLL is unloaded.

6.2.5.2 Prototype

```
BOOL WINAPI C2A_FreeDriver( void )
```

6.2.5.3 Arguments

This function has no arguments.

6.2.5.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.5.5 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0008_{hex}	Connection exists

6.2.5.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.6 C2A_GetBusStatus

6.2.6.1 Description

Gets the CAN bus status word.

6.2.6.2 Related Topics

See Section 4.3.2, CAN Bus Status Word (0030h).

6.2.6.3 Prototype

```
BOOL WINAPI C2A_GetBusStatus( DWORD CardHandle, WORD *BusStatus )
```

6.2.6.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
BusStatus	Pointer to word to receive bus status flags

6.2.6.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.6.6 Error Codes

Value	Description
$2000\ 0003_{hex}$	Driver not loaded
$2000\ 0007_{hex}$	Null pointer
$2000\ 0102_{hex}$	Invalid CardHandle
$2000\ 011a_{hex}$	Card Access Timeout

6.2.6.7 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.7 C2A_GetCANCounters

6.2.7.1 Description

Gets the CAN bus counters.

6.2.7.2 Related Topics

See Sections 4.3, Application Module Header, and 6.2.1, C2A_CAN_COUNTERS Data Type.

6.2.7.3 Prototype

```
BOOL WINAPI C2A_GetCANCounters( DWORD CardHandle, C2A_CAN_COUNTERS *CANCounters )
```

6.2.7.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
CANCounters	Buffer to contain CAN counters

6.2.7.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.7.6 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout

6.2.7.7 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.8 C2A_GetCardStatus

6.2.8.1 Description

Retrieves the card status.

6.2.8.2 Prototype

```
BOOL WINAPI C2A_GetCardStatus( DWORD CardHandle, BOOL *CardOk )
```

6.2.8.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
CardOk	Contains card status upon return. True = OK, False = Error

6.2.8.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.8.5 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout

6.2.8.6 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.9 C2A_GetModuleHeader

6.2.9.1 Description

Gets the module header information.

6.2.9.2 Related Topics

See Sections 4.3, Application Module Header, and 6.2.13, C2A_MODULE_HEADER Data Type.

6.2.9.3 Prototype

```
BOOL WINAPI C2A_GetModuleHeader( DWORD CardHandle, C2A_MODULE_HEADER *ModuleHeader )
```

6.2.9.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
ModuleHeader	Buffer to contain Module Header information

6.2.9.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.9.6 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout

6.2.9.7 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.10 C2A_LoadDriver

6.2.10.1 Description

Loads the hardware driver DLL (SSDN32.DLL).

6.2.10.2 Prototype

```
BOOL WINAPI C2A_LoadDriver( TCHAR *DriverName )
```

6.2.10.3 Arguments

Argument	Description
DriverName	Name and optional path to the hardware driver DLL i.e. "c:\windows\system\ssdn32.dll'

6.2.10.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.10.5 Error Codes

Value	Description
2000 0000_{hex}	Driver loaded
2000 0001_{hex}	Driver not found
2000 0002_{hex}	Invalid Driver
2000 0007_{hex}	Null pointer
2000 $000B_{hex}$	Corrupted installation – re-install required

6.2.10.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.11 C2A_MESSAGE Data Type

6.2.11.1 Description

Used to transmit and receive CAN packets.

6.2.11.2 Declaration

```
typedef struct
{
   WORD CANID;                // CAN Identifier
   BYTE Length;               // # of data bytes in message
   BYTE Data[8];              // Message data
   WORD TimeL;                // Message timestamp low word*
   WORD TimeH                 // Message timestamp high word*
   BYTE Reserved;             // reserved
} C2A_MESSAGE;
```

* Calculation of the TimeStamp can be done using the following formula:

```
TimeStamp (ms) = (TimeH * MajorTickInterval + ((MajorTickInterval / MinorTickCount) *
(MinorTickCount - TimeL)))
```

6.2.11.3 Related Topics

See Section 4.3, Application Module Header, for information on the MajorTickInterval and MinorTickCount.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.12 C2A_ModifyFilter

6.2.12.1 Description

Enables/disables receipt of messages with the specified CAN identifier.

6.2.12.2 Prototype

```
BOOL WINAPI C2A_ModifyFilter( DWORD CardHandle, WORD CANID, BOOL Enable )
```

6.2.12.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
CANID	The CAN identifier of the message to be enabled. 0xFFFF - enables/disables the receipt of all CAN messages.
Enable	TRUE = Enable receipt of message FALSE = Disable receipt of message

6.2.12.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.12.5 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0004_{hex}	Command timeout
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0300_{hex}	Invalid CAN identifier

6.2.12.6 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.13 C2A_MODULE_HEADER Data Type

6.2.13.1 Declaration

```
typedef struct
{
   WORD ModuleType;     // Module type identifier
   WORD WinSize;        // Host interface window size
   WORD CardId;         // Reserved
   WORD KernelId;       // Kernel identifier
   WORD KernelRev; // Kernel revision
   WORD ModuleId;       // Module identifier
   WORD ModuleRev; // Module revision
   DWORD NetSerial;     // DeviceNet serial number
   BYTE CardType[16];   // Card type (i.e. "SST-DN3")
   BYTE CardSerial[8];  // Card serial number
   WORD IrqControl;     // Card interrupt control word
   BYTE IrqStatusA;     // Card interrupt status byte A
   BYTE IrqStatusB;     // Card interrupt status byte B
   WORD MainCode;       // Main application error code
   WORD CanStatus; // CAN bus status word
   WORD CanTx;          // CAN transmit counter
   WORD CanAck;         // CAN ack error counter
   WORD CanRx;          // CAN receive counter
   WORD CanErr;         // CAN communication error counter
   WORD CanLost;        // CAN lost messages counter
   WORD CanOverrun;     // CAN receive queue overrun counter
   WORD AddCode;        // Additional application error code
   BYTE ModuleString[60];    // Module status string
   WORD MajorTickInterval;// Timestamp 'major tick interval'
   WORD MinorTickCount;      // Timestamp 'minor tick count'
} C2A_MODULE_HEADER;
```


©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.14 C2A_Offline

6.2.14.1 Description

Sets the physical connection offline.

6.2.14.2 Prototype

```
BOOL WINAPI C2A_Offline( DWORD CardHandle )
```

6.2.14.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.14.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.14.5 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0004_{hex}	Command timeout
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0302_{hex}	Not online

6.2.14.6 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.15 C2A_Online

6.2.15.1 Description

Puts the physical connection online.

6.2.15.2 Related Topics

See Section 4.4.1, Command Word, for more information on custom baud rate bit timings.

6.2.15.3 Prototype

```
BOOL WINAPI C2A_Online( DWORD CardHandle, BYTE BaudRate, WORD BitTiming, BOOL
EnableTimestamp )
```

6.2.15.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
BaudRate	Baud rate at which to go online 0 = 125K 1 = 250K 2 = 500K 3 = Custom 4 = 1M
BitTiming	Bit timing for custom baud rates
EnableTimestamp	Enables message TX and RX timestamps

6.2.15.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.15.6 Error Codes

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0004_{hex}	Command timeout
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0301_{hex}	Invalid baud rate
2000 0303_{hex}	Bus is not offline
2000 0304_{hex}	Bus fault encountered
2000 0308_{hex}	Invalid bit timing

6.2.15.7 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.16 C2A_OpenCard

6.2.16.1 Description

Opens a card connection. The card handle returned by this function is used by the DLL client to identify the card to all other services.

6.2.16.2 Prototype

```
BOOL WINAPI C2A_OpenCard( DWORD *CardHandle, TCHAR *CardName, void *Module, WORD
ShareFlags, WORD LoadFlags )
```

6.2.16.3 Arguments

Argument	Description
CardHandle	Upon successful completion of C2A_OpenCard, CardHandle is used for subsequent API calls.
CardName	The name assigned to the SST-DN card.
Module	Embedded binary application module pointer. See SS_EMBEDDED loader flag. {This argument is ignored for remote link devices.}
ShareFlags	Connection share flags (see Share Flags table below) {This argument is ignored for remote link devices. Exclusive write access is always used.}
LoadFlags	Application module loader flags (see Load Flags table below) {This argument is ignored for remote link devices. A valid LoadFlags value should always be specified though, to enable compatibility with local devices.}

6.2.16.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.16.5 Errors

Value	Description
2000 0003$_{hex}$	Driver not loaded
2000 0007$_{hex}$	Null pointer
2000 0008$_{hex}$	Connection already exists
2000 0009$_{hex}$	Resource unavailable
2000 000a$_{hex}$	Permission denied
2000 0100$_{hex}$	General failure
2000 0101$_{hex}$	Card handle not available
2000 0103$_{hex}$	Write Access denied (sharing violation)
2000 0104$_{hex}$	Card Irq Access denied (sharing violation)
2000 0105$_{hex}$	App Irq Access denied (sharing violation)
2000 0106$_{hex}$	Hardware Control Access denied (sharing violation)
2000 0107$_{hex}$	Overlap conflict
2000 0108$_{hex}$	Configuration not found (invalid CardName)
2000 0109$_{hex}$	Card not found
2000 010a$_{hex}$	Memory conflict
2000 010b$_{hex}$	Invalid / damaged application module
2000 010c$_{hex}$	Card not responding (application did not run)
2000 010d$_{hex}$	Card self-diagnostic failure
2000 0115$_{hex}$	Invalid interrupt level
2000 0116$_{hex}$	Invalid I/O address
2000 0117$_{hex}$	Invalid I/O length
2000 0118$_{hex}$	Invalid memory address
2000 0119$_{hex}$	Invalid memory length
2000 011a$_{hex}$	Card Access Timeout (default 1 second)
2000 011b$_{hex}$	Driver access denied
2000 011c$_{hex}$	Memory test failure
2000 011d$_{hex}$	Module load request denied
2000 022d$_{hex}$	Remote Transport Link Error

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.16.6 ShareFlags

This argument controls multiple application arbitration for a single card.

Bit	Name	Description
0	SS_WRITE	Card opened for shared write access[1]
1	SS_XWRITE	Card opened for exclusive write access[1]
2	SS_HCONTROL	Card opened for exclusive hardware control access
3	SS_CARDIRQ	Card opened for exclusive card interrupt access (interrupts from the application to the card)
4	SS_APPIRQ	Card opened for exclusive application interrupt access (interrupts from the card to the application)
5-7	Reserved	Reserved, set to 0
8	SS_USER1**	Exclusive user access 1
9	SS_USER2**	Exclusive user access 2
10	SS_USER3**	Exclusive user access 3
11	SS_USER4**	Exclusive user access 4
12-13	Reserved	Reserved, set to 0
14	SS_OVERLAP	Overlapped cards are supported
15	Reserved	Reserved, set to 0

** Exclusive user access flags are used to avoid conflicts between applications sharing access to the same card. The meaning of these flags is card / module specific.

[1] Write access is required for module loading

6.2.16.7 LoadFlags

This argument controls the application module loader and also selects some connection modes of operation.

Bit	Name	Description
0	SS_REPLACE	Replace already loaded module if no other connections have write access
1	SS_RELOAD	Reload the specified module if no other applications have write access
2	SS_EMBEDDED	Module is embedded in application. Module argument points to start of module binary.
3-15	Reserved	Always 0

6.2.16.8 Remote Link Device API Behavior

The behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.17 C2A_Receive

6.2.17.1 Description

Retrieves a message from the receive queue.

6.2.17.2 Prototype

```
BOOL WINAPI C2A_Receive( DWORD CardHandle, C2A_MESSAGE *Message )
```

6.2.17.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Message	The buffer to contain the message

6.2.17.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.17.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0302_{hex}	Not online
2000 0305_{hex}	No messages available

6.2.17.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.18 C2A_RegisterBusStatusEvent

6.2.18.1 Description

Registers for bus status change event notification. Upon notification of a registered event, *wParam* contains the CAN bus status word.

6.2.18.2 Related Topics

See Section 4.3.2, CAN Bus Status Word (0030h).

6.2.18.3 Prototype

```
BOOL WINAPI C2A_RegisterBusStatusEvent( DWORD CardHandle, DWORD ThreadId, DWORD MsgId,
LPARAM lParam )
```

6.2.18.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
ThreadId	Thread Id of thread to be notified of events
MsgId	Message Identifier
lParam	Thread-specific data (not modified by SSC2A32.DLL)

6.2.18.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.18.6 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 0307_{hex}	Event already registered

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.18.7 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.19 C2A_RegisterReceiveEvent

6.2.19.1 Description

Registers for event notification upon receipt of a message.

6.2.19.2 Prototype

```
BOOL WINAPI C2A_RegisterReceiveEvent( DWORD CardHandle, DWORD ThreadId, DWORD MsgId,
LPARAM lParam )
```

6.2.19.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
ThreadId	Thread Id of thread to be notified of events
MsgId	Message Identifier
lParam	Thread-specific data (not modified by SSC2A32.DLL)

6.2.19.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.19.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 0307_{hex}	Event already registered

6.2.19.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.20 C2A_Send

6.2.20.1 Description

Transmits the specified message. If timestamping is enabled, the calling thread is blocked until the message is transmitted. Upon return, the TimeL and TimeH parameters within the Message parameter contain the time the message was transmitted.

6.2.20.2 Related Topics

See Sections 6.2.11, C2A_MESSAGE Data Type, and 6.2.15, C2A_Online.

6.2.20.3 Prototype

```
BOOL WINAPI C2A_Send( DWORD CardHandle, C2A_MESSAGE *Message )
```

6.2.20.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Message	Buffer containing message to send

6.2.20.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.20.6 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0300_{hex}	Invalid CAN identifier
2000 0302_{hex}	Not Online
2000 0309_{hex}	Tx queue full
2000 0310 $_{hex}$	Invalid data size

6.2.20.7 Remote Link Device API Behavior

The behavior is remote if timestamping is enabled; otherwise, the behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.21 C2A_SetAccessTimeout

Note

This API applies only to interface cards that share system memory (ISA and 104).

6.2.21.1 Description

Sets the card access timeout delay. Indicates the maximum access time for use with overlapped cards.

6.2.21.2 Prototype

```
BOOL WINAPI C2A_SetAccessTimeout( DWORD CardHandle, DWORD Timeout )
```

6.2.21.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Timeout	The access timeout delay in milliseconds (default 1000 ms).

6.2.21.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.21.5 Errors

Value	Description
2000 0003h	Driver not loaded
2000 0102h	Invalid CardHandle

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.21.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

Note

For Remote Link Devices, this API always returns TRUE.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.22 C2A_SetEventNotificationInterval

6.2.22.1 Description

Sets the event notification interval for all registered events.

6.2.22.2 Prototype

```
BOOL WINAPI C2A_SetEventNotificationInterval( DWORD CardHandle, DWORD Interval )
```

6.2.22.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Interval	Event notification interval in milliseconds (minimum 10ms)

6.2.22.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.22.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle

6.2.22.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.23 C2A_TriggerTX

6.2.23.1 Description

Transmits all messages in the transmit queue. It is strongly recommended that transmit timestamping be disabled when using this call.

6.2.23.2 Related Topics

See Section 6.2.30, C2A_WriteTXQueue.

6.2.23.3 Prototype

```
BOOL WINAPI C2A_TriggerTX( DWORD CardHandle )
```

6.2.23.4 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.23.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.23.6 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0302_{hex}	Not Online

6.2.23.7 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.24 C2A_TXQueueEmpty

6.2.24.1 Description

Checks the status of the transmit queue.

6.2.24.2 Prototype

```
BOOL WINAPI C2A_TXQueueEmpty( DWORD CardHandle, BOOL *Empty )
```

6.2.24.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Empty	The status of the transmit queue. TRUE = The queue is empty FALSE = The queue is not empty

6.2.24.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.24.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle

6.2.24.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.25 C2A_UnRegisterBusStatusEvent

6.2.25.1 Description

Cancels bus status change notification.

6.2.25.2 Prototype

```
BOOL WINAPI C2A_UnRegisterBusStatusEvent( DWORD CardHandle )
```

6.2.25.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.25.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.25.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 0306_{hex}	Event not registered

6.2.25.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.26 C2A_UnRegisterReceiveEvent

6.2.26.1 Description

Cancels receive event notification.

6.2.26.2 Prototype

```
BOOL WINAPI C2A_UnRegisterReceiveEvent( DWORD CardHandle )
```

6.2.26.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.26.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.26.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0102_{hex}	Invalid CardHandle
2000 0306_{hex}	Event not registered

6.2.26.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.27 C2A_VerifyCardHandle

6.2.27.1 Description

Verifies the validity of the card handle.

6.2.27.2 Prototype

```
BOOL WINAPI C2A_VerifyCardHandle( DWORD CardHandle )
```

6.2.27.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard

6.2.27.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.27.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded

6.2.27.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.28 C2A_Version

6.2.28.1 Description

Retrieves DLL version information. The DLL version is returned in both numeric and human-readable string format.

6.2.28.2 Prototype

```
BOOL WINAPI C2A_Version( TCHAR *Buffer, WORD *Version, DWORD Size )
```

6.2.28.3 Arguments

Argument	Description
Buffer	DLL identification string buffer
Version	Minor revision (LSB) Major revision (MSB)
Size	Buffer size. The Identification string is truncated to fit.

6.2.28.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.28.5 Errors

Value	Description
2000 0007_{hex}	Null pointer

6.2.28.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.29 C2A_VersionEx

6.2.29.1 Description

Retrieves version information for every component in the card name tree. For example, calling this function on a local card name would return the version information of Ssc2a32.dll, SSDn32.dll, and the hardware driver; calling it on a remote card name would return the version information of Ssc2a32.dll, RemoteProxy, the remote server firmware tasks, and the remote server hardware version.

The versions are returned in an array of VersionEx structures, beginning at *Buffer. If all the VersionEx structures fit into “Buffer”, the API will return TRUE, and *Size will be set to the total size (in bytes) of all VersionEx structures in the “Buffer” data. If the VersionEx structures will not all fit into “Buffer” (because *Size is not large enough), the API will return FALSE, and *Size will be set to the buffer size (in bytes) required for the API to succeed.

6.2.29.2 Prototype

```
BOOL WINAPI C2A_VersionEx( CHAR* CardName, CHAR *Buffer, DWORD *Size, unsigned short
*VersionCount )
```

6.2.29.3 Arguments

Argument	Description
CardName	The name assigned to the SST-DN card
Buffer	Version information buffer
Size	Buffer Size
VersionCount	Number of VersionEx structures returned in “Buffer”

6.2.29.4 VersionEx Structure

The fields of the VersionEx data structure shall be defined as listed below:

Name	Data Type	Description
ComponentName	Unsigned char[80]	ASCII human-readable NULL terminated string format
Version	Unsigned short	Upper byte ->Major revision. Lower byte -> Minor revision

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.29.5 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.29.6 Errors

Value	Description
2000 0007h	Null pointer
2000 0109h	Card not Found
2000 0110h	Invalid Size – The length specified by “DWORD *Size” was not large enough to hold all the requested version information

6.2.29.7 Remote Link Device API Behavior

The behavior is local if the card is already open; otherwise, the behavior is remote. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

6.2.30 C2A_WriteTXQueue

6.2.30.1 Description

Writes a message into the CAN2A transmit queue. Messages written to the transmit queue using this function will be transmitted when C2A_Send or C2A_TriggerTX is called.

6.2.30.2 Prototype

```
BOOL WINAPI C2A_WriteTXQueue( DWORD CardHandle, C2A_MESSAGE *Message )
```

6.2.30.3 Arguments

Argument	Description
CardHandle	The handle returned by C2A_OpenCard
Message	Buffer containing message to write

6.2.30.4 Returns

Value	Description
TRUE	Success
FALSE	Error. Use GetLastError to retrieve error code.

6.2.30.5 Errors

Value	Description
2000 0003_{hex}	Driver not loaded
2000 0007_{hex}	Null pointer
2000 0102_{hex}	Invalid CardHandle
2000 $011a_{hex}$	Card Access Timeout
2000 0300_{hex}	Invalid CAN identifier
2000 0302_{hex}	Not Online
2000 0309_{hex}	Tx queue full
2000 0310 $_{hex}$	Invalid data size

6.2.30.6 Remote Link Device API Behavior

The behavior is local. For more information, see Section 6.1, Introduction.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

A

Warranty and Support

Appendix Contents:

- Warranty
- Technical Support

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

A.1 Warranty

For warranty information, refer to http://www.mysst.com/warranty.asp.

A.2 Technical Support

Please ensure that you have the following information readily available before calling for technical support:

- Card type and serial number
- Computer's make, model, CPU speed and hardware configuration (other cards installed)
- Operating system type and version
- Details of the problem you are experiencing: firmware module type and version, target network and circumstances that may have caused the problem

A.2.1 Getting Help

Technical support is available during regular business hours by telephone, fax or email from any Woodhead Software & Electronics office, or from http://www.woodhead.com. Documentation and software updates are also available on the website.

Note

If you are using the SST-DN card with a third-party application, refer to the documentation for that package for information on configuring the software for the card.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

North America

Canada:
Tel: +1-519-725-5136
Fax: +1-519-725-1515
Email: WoodheadSupportNA@molex.com

Europe

France:
Tel: +33 2 32 96 04 22
Fax: +33 2 32 96 04 21
Email: WoodheadIC.SupportEU@molex.com

Germany:
Tel: +49 7252 9496 555
Fax: +49 7252 9496 99
Email:mailto: WoodheadIC.SupportEU@molex.com

Italy:
Tel: +39 010 5954 052
Fax: +39 02 664 00334
Email: WoodheadIC.SupportIT@molex.com

Other countries:
Tel: +33 2 32 96 04 23
Fax: +33 2 32 96 04 21
Email: WoodheadIC.SupportEU@molex.com

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.

Asia-Pacific

Japan:
Tel: +81 52 221 5950
Fax: +81 46 265 2429
Email: WoodheadIC.SupportAP@molex.com

Singapore:
Tel: +65 6268 6868
Fax: +65 6261 3588
Email: WoodheadIC.SupportAP@molex.com

China:
Tel: +86 21 5835 9885
Fax: +86 21 5835 9980
Email: WoodheadIC.SupportAP@molex.com

For the most current contact details, please visit http://www.woodhead.com.

©2010 Molex Incorporated
Document Edition: 1.1, Document #: 717-0026, Template Edition: 1.1, Template #: QMS-06-045
Use, duplication or disclosure of this document or any of the information contained herein is subject to the restrictions on page ii of this document.