5136-SD-VME, -VME2

User's Guide

Version 1.03

Industrial Communication Technologies

50 Northland Road, Waterloo, Ontario N2V 1N3

(519) 725-5136 fax (519) 725-1515

Printed in Canada

Publication Name : SDVME.DOC
Publication Revision: 1.03
Date Printed: July 22, 1998

--This Document Applies To --

5136-SD-VME

5136-SD-VME2

Interface Cards

This product incorporates patented technology which is licensed by Allen-Bradley Company Inc. Allen-Bradley has not technically approved, nor does it support, this product. All warranty and support for this product is provided by S-S Technologies.

1. Introduction

1.1 Purpose of this Document

This document is a user's guide for the SST 5136-SD-VME and 5136-SD-VME2 interface cards. These cards make it possible for an application running on a VME host computer to communicate with Allen-Bradley programmable controllers over Data Highway and Data Highway Plus.

The 5136-SD-VME is a single channel card; the 5136-SD-VME2 is a two channel card. On the two channel card, the channels are completely independent of each other.

1.2 Card Overview

The 5136-SD-VME and 5136-SD-VME2 interface cards provide an intelligent front end between VMEbus master cards and Allen-Bradley programmable controller communication networks.

There are single channel (5136-SD-VME) and two channel (5136-SD-VME2) versions. On the two channel card, each channel can be configured and operated independently of the other.

Each card is a co-processor card with circuitry to provide a standard bus interface to the VMEbus backplane. The card's form factor is double-height, single-width with electrical connection to the VMEbus backplane via the P1 connector. The card acts as a slave on the VMEbus.

Each channel on the card contains a Z180 processor which is loaded with software by the host computer to enable it to perform the communication tasks on the selected network protocol.

The card contains no software in ROM; the appropriate interface software is downloaded to the card from the host computer. This design allows a single card to access Data Highway or Data Highway Plus networks and makes it easy to add new features to the card software by simply downloading a new module to the card. You simply download a new module from the host before letting the card run. On the two channel card, each channel can use a different protocol or be connected to a different network.

Interaction between the task resident on the channel and the application software on a VME host is made through shared RAM. The lower 32 Kbytes of memory on each channel are reserved for the software module which is downloaded when the card is initialized; the upper 32 Kbytes are used for data and tables.

Each channel on the card occupies a 64 Kbyte block of Standard Access space for the shared RAM and a 6-byte "control block" in Short I/O space. (By definition, short addressing applies to boards which decode A01-A15. This mode of addressing is normally used for I/O boards. A16-A23 are not decoded; therefore the I/O memory is usually, but not always, mapped into the 64 Kbyte block $FF0000 to $FFFFFF.) The address of the control block is set using DIP switches on the card.

There are three registers in the control block which affect the operation of the channel: the control/status register, the interrupt status/ID register and the memory address register. All control block registers are 1 byte wide.

The control/status register is a read-write register which allows the host to control and monitor the channel. The interrupt status/ID register is used for interrupt initialization and processing. The memory address register is used to assign the address of the 64 Kbyte block of RAM associated with the channel in the VMEbus standard access memory map.

The card supports the VMEbus Priority Interrupt Bus. The interrupt vector is software selectable using the "Interrupt status/ID" register in the control block. The interface card generates interrupt requests using an I(1) through I(7) (Stationary), ROAK interrupter. D08(O) (Stationary) status/ID transfer capability is used.

The card responds to Data D08(O) Transfer Bus (DTB) cycles in the A16 (Short) addressing mode and D16 or D08(EO) DTB cycles in the A24 (Standard) addressing mode. D08(EO) bus cycles are single-byte cycles. When 16-bit values are read from the card (i.e. buffer pointers) the values are organized in low-byte, high-byte order.

The address modifier (AM) codes can be one of standard or short supervisory data access ($3D, $2D) or standard or short non-privileged data access ($39, $29).

For diagnostic purposes the card is considered to be a "non-intelligent" card; diagnostic routines (i.e. memory test) are performed on the card by a master processor rather than by the card itself.

1.3 Manual Organization

This document is divided into several sections: part 2 describes how to install the card, part 3 describes the Data Highway and Data Highway Plus modules and the steps in writing an application. The appendices give some technical information on the card and explain how to obtain technical support.

1.4 Conventions

In this manual, a leading $ or 0x indicates a hexadecimal number.

VMEbus signals are shown in bold. An asterisk indicates an active-low signal, for example **SYSFAIL*.**

1.5 Reference Documents

Refer to the appropriate Allen-Bradley documents for information on Allen-Bradley hardware, cabling, programming and network protocols.

If you are developing an application on Data Highway or Data Highway Plus or if you need information about message formats or network status values, refer to the Allen-Bradley document "Communication Protocol and Command Set", 1770-6.5-16.

The Allen-Bradley document "Data Highway Cable", 1770-6.2.2 describes cabling of a Data Highway or Data Highway Plus network.

Refer to “IEEE Standard for a Versatile Backplane Bus: VMEbus”, ANSI/IEEE Std. 1014-1987 for explanations of VMEbus terminology.

2. Installation

The 5136-SD-VME interface card contains components that are sensitive to electrostatic discharge. Do not remove the card from its protective bag without using the following precautions:

- Before handling the card, ground yourself by touching a grounded object, such as the case of your computer.
- Never touch the backplane connectors or pins. Handle the card by its mounting bracket.
- Always store the card in its protective bag.

This chapter describes the procedures for:

- setting the switches and jumpers on the interface card
- installing the card in your computer
- downloading the firmware to the card
- getting the card to communicate on a network and verifying that it is working

It also contains information about short I/O register usage on the card.

2.1 Card Installation

Before you install the card in the VMEbus chassis, you must decide on the answers to the following questions:

- which 1 Kbyte block in the Short address space will be used for card control for each channel?
- will interrupts be used?
- will the card transmit on the network to which it is attached?
- should the card respond to supervisory mode commands and data accesses only?

In order to answer these questions, you must know the memory usage of the system and the capabilities of the application software.

The answers to these questions determine the settings of several switches and jumpers on the card. These must be set before you install the card in a VMEbus computer.

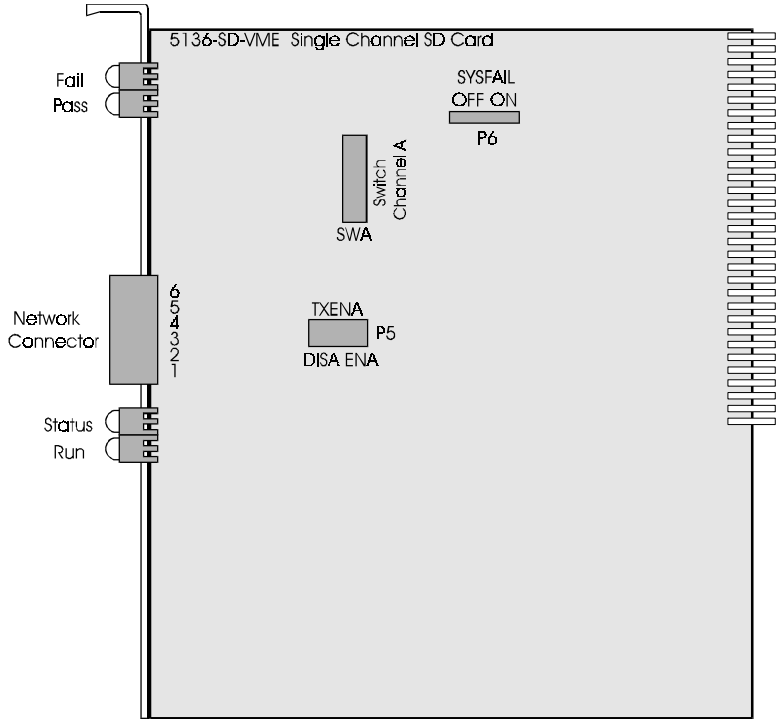
For each channel there is a 10-position switch. The table below shows the use for the various switch positions and where to look for information on how to set the switch.

Positions	Used to set	Reference section
1-6	Short I/O address	2.2.1
7	Allowed address modifiers	2.2.2
8-10	Interrupt	2.2.3

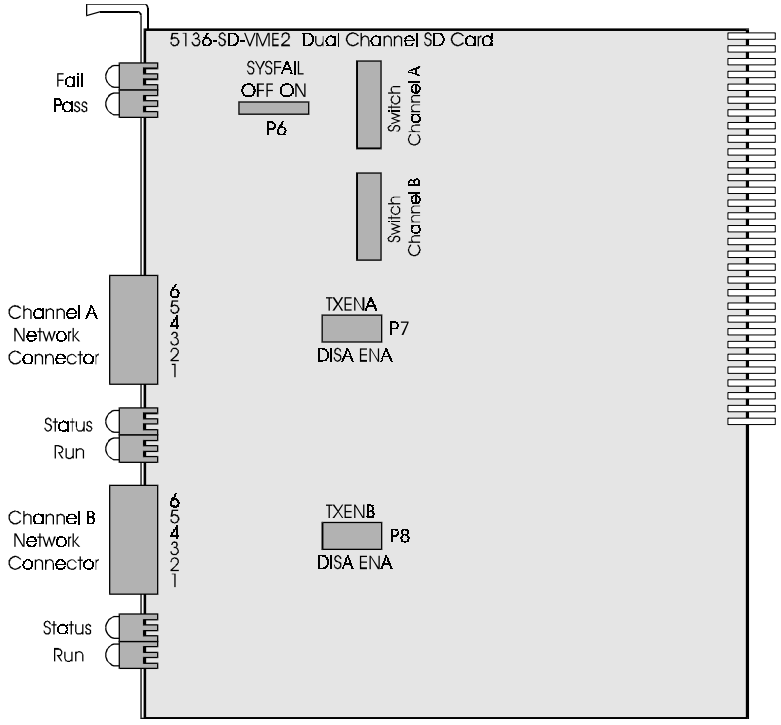
In addition there is a jumper for each channel which controls transmission from that channel. Refer to section 2.2.4.

The SYSFAIL* Jumper controls how the card handles the **SYSFAIL*** line. Refer to section 2.2.5.

The following diagrams show the position of the switches, jumpers and LEDs on each of the cards.


5136-SD-VME Single Channel SD Card
Fail
Pass
SYSFAIL
OFF ON
P6
Switch
Channel A
SWA
Network
Connector
6
5
4
3
2
1
TXENA
P5
DISA ENA
Status
Run

5136-SD-VME


5136-SD-VME2 Dual Channel SD Card
Fail
Pass
SYSFAIL
OFF ON
P6
Switch Channel A
Switch Channel B
Channel A Network Connector
6 5 4 3 2 1
TXENA
P7
DISA ENA
Status
Run
Channel B Network Connector
6 5 4 3 2 1
TXENB
P8
DISA ENA
Status
Run

5136-SD-VME2

2.2 Setting the Switches and Jumpers

2.2.1 Setting the Short I/O Base Address

The 1 Kbyte block of short address space occupied by each channel on the card is located on a 1 Kbyte boundary at an address selected by positions 1 through 6 of the switch for that channel. These switch positions correspond to address bits A15 through A10, respectively, with a switch in the ON position matching the corresponding address signal as a 0.

Switch Position	Address bit
1	A15
2	A14
3	A13
4	A12
5	A11
6	A10

On the two-channel card, the short I/O base address must be different for each channel.

The following table shows possible addresses and the corresponding switch settings.

Address	1	2	3	4	5	6
FC00	OFF	OFF	OFF	OFF	OFF	OFF
F800	OFF	OFF	OFF	OFF	OFF	ON
F400	OFF	OFF	OFF	OFF	ON	OFF
F000	OFF	OFF	OFF	OFF	ON	ON
EC00	OFF	OFF	OFF	ON	OFF	OFF
E800	OFF	OFF	OFF	ON	OFF	ON
E400	OFF	OFF	OFF	ON	ON	OFF
E000	OFF	OFF	OFF	ON	ON	ON
DC00	OFF	OFF	ON	OFF	OFF	OFF
D800	OFF	OFF	ON	OFF	OFF	ON
D400	OFF	OFF	ON	OFF	ON	OFF

Address	1	2	3	4	5	6
D000	OFF	OFF	ON	OFF	ON	ON
CC00	OFF	OFF	ON	ON	OFF	OFF
C800	OFF	OFF	ON	ON	OFF	ON
C400	OFF	OFF	ON	ON	ON	OFF
C000	OFF	OFF	ON	ON	ON	ON
BC00	OFF	ON	OFF	OFF	OFF	OFF
B800	OFF	ON	OFF	OFF	OFF	ON
B400	OFF	ON	OFF	OFF	ON	OFF
B000	OFF	ON	OFF	OFF	ON	ON
AC00	OFF	ON	OFF	ON	OFF	OFF
A800	OFF	ON	OFF	ON	OFF	ON
A400	OFF	ON	OFF	ON	ON	OFF
A000	OFF	ON	OFF	ON	ON	ON
9C00	OFF	ON	ON	OFF	OFF	OFF
9800	OFF	ON	ON	OFF	OFF	ON
9400	OFF	ON	ON	OFF	ON	OFF
9000	OFF	ON	ON	OFF	ON	ON
8C00	OFF	ON	ON	ON	OFF	OFF
8800	OFF	ON	ON	ON	OFF	ON
8400	OFF	ON	ON	ON	ON	OFF
8000	OFF	ON	ON	ON	ON	ON
7C00	ON	OFF	OFF	OFF	OFF	OFF
7800	ON	OFF	OFF	OFF	OFF	ON
7400	ON	OFF	OFF	OFF	ON	OFF
7000	ON	OFF	OFF	OFF	ON	ON
6C00	ON	OFF	OFF	ON	OFF	OFF
6800	ON	OFF	OFF	ON	OFF	ON
6400	ON	OFF	OFF	ON	ON	OFF
6000	ON	OFF	OFF	ON	ON	ON
5C00	ON	OFF	ON	OFF	OFF	OFF
5800	ON	OFF	ON	OFF	OFF	ON
5400	ON	OFF	ON	OFF	ON	OFF
5000	ON	OFF	ON	OFF	ON	ON
4C00	ON	OFF	ON	ON	OFF	OFF
4800	ON	OFF	ON	ON	OFF	ON

Address	1	2	3	4	5	6
4400	ON	OFF	ON	ON	ON	OFF
4000	ON	OFF	ON	ON	ON	ON
3C00	ON	ON	OFF	OFF	OFF	OFF
3800	ON	ON	OFF	OFF	OFF	ON
3400	ON	ON	OFF	OFF	ON	OFF
3000	ON	ON	OFF	OFF	ON	ON
2C00	ON	ON	OFF	ON	OFF	OFF
2800	ON	ON	OFF	ON	OFF	ON
2400	ON	ON	OFF	ON	ON	OFF
2000	ON	ON	OFF	ON	ON	ON
1C00	ON	ON	ON	OFF	OFF	OFF
1800	ON	ON	ON	OFF	OFF	ON
1400	ON	ON	ON	OFF	ON	OFF
1000	ON	ON	ON	OFF	ON	ON
0C00	ON	ON	ON	ON	OFF	OFF
0800	ON	ON	ON	ON	OFF	ON
0400	ON	ON	ON	ON	ON	OFF
0000	ON	ON	ON	ON	ON	ON

2.2.2 Address Modifier Codes

The 5136-SD-VME provides 8-bit access to objects in its short address space, and 8- and 16-bit access to objects in its standard address space. Whether a particular bus cycle accesses short, standard, extended, or long (extended and long are not used on the 5136-SD-VME or 5136-SD-VME2) address spaces, and the type of access that is made, are selected by the VME master through the use of the address modifier codes. These codes are decoded by the card and used to determine the object to be accessed.

In addition to selecting from among the four spaces available on the VMEbus, address modifier codes also select whether the master is making a supervisory or non-privileged access, and (for all but short address space accesses) whether the access is to program or data space, and whether it is to be a single-object or block access.

The 5136-SD-VME can respond to address modifier codes $3D, $39, $2D, and $29. This means that supervisory and non-privileged data accesses may be made to standard address space ($3D and $39), and that supervisory and non-privileged accesses may be made to short address space ($2D and $29). An access which can be positively determined to address the card, but with an address modifier code that is not supported, causes a VMEbus error.

The selection of whether only supervisory accesses or both supervisory and non-privileged accesses are permitted is made with position 7 of the channel's DIP switch. The use of this switch is detailed in the following table. If a non-privileged access is made to the card when this switch is in the supervisory only position, a VMEbus error occurs.

Switch position 7	Permitted accesses
ON	Both Supervisory and Non-privileged
OFF	Supervisory only

2.2.3 Setting the Interrupt

If so enabled, the software module on the local processor can assert a VME interrupt on the switch-selected level. The level is set using positions 8, 9 and 10 of the corresponding switch block for the channel.

Switch position			Interrupt
8	9	10	
ON	ON	ON	none
OFF	ON	ON	1
ON	OFF	ON	2
OFF	OFF	ON	3
ON	ON	OFF	4
OFF	ON	OFF	5
ON	OFF	OFF	6
OFF	OFF	OFF	7

2.2.4 Transmit Enable Jumpers

Jumpers are used to enable or disable transmission from the card. On the single channel 5136-SD-VME card, the jumper is labelled TXENA. On the two channel 5136-SD-VME2 card, the jumpers are labelled TXENA for channel A and TXENB for channel B. If the software module does not transmit over the network then it is good practice to disable transmission using the appropriate jumper. Transmission is disabled when the jumper is placed over the pins labelled DISA of the block. Transmission is enabled when the jumper is placed over the pins labelled ENA. For all Data Highway and Data Highway Plus applications, the jumper must be in the enabled position even if your application does not send any messages.

2.2.5 SYSFAIL* Jumper

This jumper is labelled P6 on both the 5136-SD-VME and the 5136-SD-VME2. When it is in the ON (right) position, assertion of the SYSFAIL* control bit in the control register causes the VMEbus **SYSFAIL*** signal to be asserted, the PASS LED to turn off, and the FAIL LED to turn on. When the jumper is in the OFF position, the LEDs reflect the state of the **SYSFAIL*** control signal but the card does not drive the VMEbus signal. On the 5136-SD-VME2 a single jumper applies to both channels.

2.3 Short I/O Registers

Each channel occupies 1 Kbyte of the short I/O space but only three 1-byte registers are used in that 1K.

The following table gives the map of the registers contained in the short address space to which the card responds. Since the short address space of the card is essentially a D08(O) Slave, the objects in this space are not at contiguous addresses. Any attempt to access addresses other than those specified in the table (e.g. offsets 0, 2 or 4), or to access objects larger than 8 bits from the card's short address space, results in a VMEbus error.

Offset from short I/O base	Selected register
1	Control and status register
3	Interrupt vector register
5	Memory address register

2.3.1 Control and Status Register

The control and status register permits the host computer to enable/disable the local processor (Z180), to assert/deassert an interrupt to the local processor, to sense and clear an interrupt from the local processor, to control an indicator LED, to assert/deassert the VME SYSFAIL signal from the card, to enable/disable access to the channel's shared RAM, and to enable/disable write protection to the local processor's code space.

The control and status register is located at offset 1 from the selected register base address in short address space. Its bits are all high true, and their assignments and reset states are detailed in the following table.

Control/Status Register (byte at offset 1)

BIT	FUNCTION	R/W	RESET
0	local processor reset control	R/W	1
1	interrupt to local processor	R/W	0

BIT	FUNCTION	R/W	RESET
2	clear interrupt from local processor (reads as 0)	W	0
3	LED control	R/W	0
4	SYSFAIL control	R/W	0
5	shared RAM enable	R/W	0
6	low 32KB write inhibit	R/W	0
7	interrupt from local processor	R	0

Setting the **local processor reset control** bit to '1' holds the processor on the channel in reset and prevents it from running. Since this is the case following a system reset, the processor does not run until the host computer releases it, after the code has been loaded. When this bit is a '1' and the host writes a '0' to it, a reset pulse is issued to the Z180 on the channel, causing it to begin execution at its own offset 0000.

The **interrupt to local processor** bit is used by application software to get the attention of the module running on the Z180. This bit cannot be cleared by the local processor, so the host must clear it when it detects that the local processor has handled the interrupt. This interrupt is edge-sensitive. No current modules for the card use interrupts to the local processor.

A vector is placed on the bus during the interrupt acknowledge cycle, as specified in the Interrupt vector register. The local processor generates an interrupt by setting the interrupt from local processor bit, which causes the selected VME interrupt signal to be asserted. This signal is released when the interrupt acknowledge cycle takes place, but the interrupt from local processor bit that caused that interrupt remains set until cleared explicitly by an interrupt service routine. The routine must do this by writing a '1' to the **clear interrupt from local processor** bit, which clears and re-enables the hardware for further interrupts.

The **LED control** bit turns on (1) and off (0) the RUN LED for the channel. (The STATUS LED is controlled by the local processor directly and is used by the software as an indicator of its current state.)

Setting the SYSFAIL control bit to '1' asserts the **SYSFAIL*** signal on the VMEbus if the SYSFAIL jumper is in the "ON" position. This bit also controls the state of the PASS and FAIL LEDs on the card.

Following reset, the **shared RAM enable** bit assures that the card will not drive the VMEbus for any standard address cycles. This bit must not be set to '1', enabling access to the shared RAM, until the Memory address register has been set to the required base in standard address space.

Since all of the Z180's memory is shared with the VME host, it may be desirable to prevent the host from writing to the lower 32 Kbyte of Z180 memory. SST software uses the lower 32 Kbyte for Z180 code and unintentional writes to this area by the host could cause the Z180 software module to crash. The **low 32KB write inhibit** bit, when '1', prevents host-initiated write cycles from altering shared RAM in the range $0000 to $7FFF (offset from the selected standard access base). Such cycles do not cause VMEbus errors and appear to complete normally. Nevertheless, as with all shared RAM cycles, they should be minimized as they rob cycles from the local processor.

Whenever the card software generates an interrupt on a channel, it sets the **interrupt from local processor** bit.

2.3.2 Interrupt Vector Register

When an interrupt request is made from the card on one of the VME interrupt lines, an interrupt handler acquires the bus and executes an interrupt acknowledge cycle. The card recognizes the acknowledgment and places an 8-bit interrupt vector on the odd half of the data bus. The value for this vector is taken from the Interrupt vector register, located at offset 3 from the selected register base address in short address space.

This register is zeroed by system reset and may be read or written at any time. If interrupts are used from the card, this register should be initialized before the local processor is allowed to run.

2.3.3 Memory Address Register

Following system reset, the shared RAM enable bit in the Control/Status register is reset, preventing the card from driving the bus during any standard address cycles. This is done in order that the location in standard address space occupied by the shared RAM may be selected. Standard address space accesses use 24 bits of address (16MB) while the card uses only 16 bits of address (64 Kbytes) for generating addresses into the shared RAM. For standard address bus cycles, the upper 8 bits from the VMEbus are compared to the bits in the 8-bit Memory address register, located at offset 5 from the selected register base address in short address space. If these match exactly, and the shared RAM enable bit is set, then a shared RAM access is performed.

For example, if it is required that the shared RAM occupy the 64 Kbyte block between $0D0000 and $0DFFFF in standard address space, then the Memory address register should be set to $0D.

The memory address register is zeroed by system reset, and may be read or written at any time. It should be initialized with the upper 8 bits of the 24-bit address at which the shared RAM area is to begin.

2.4 Standard Address Space

Each channel on the card contains 64 Kbytes of static RAM that is used as the code and data memory for the local processor (Z180) and the buffers and registers for all network activity in which the card is involved. This RAM is accessible at any time to a master on the VMEbus, regardless of whether or not the local processor is running. As necessary, the local processor pauses while the VMEbus master accesses the memory.

This memory is located at offsets 0000 through $FFFF from the 24-bit base address, whose upper 8 bits are determined from the Memory Address register contents. A VME master may access the memory either 8 or 16 bits at a time. If a 16-bit object is accessed, it must be aligned on a 16-bit boundary. That is, an access to a 16-bit object, 8 bits of which are in the odd byte of one address, and whose other 8 bits are in the even byte of the next address, is not possible. If this is attempted by host software, the hardware on the master will probably take care of the details, performing two single-byte accesses invisibly to the software.

Byte addresses on the VMEbus access locations that correspond exactly to addresses on the local processor. Also, 16-bit accesses on the VMEbus access the expected bytes from the local processor's space. For example, if $1234 is written by a VMEbus master to offset 0 in the shared RAM, the local processor sees $12 at address 0, and $34 at address 1. Since the processor on the card puts low bytes at low addresses and VMEbus masters put low bytes at high addresses, care must be taken when accessing 16-bit objects in the shared RAM to ensure that the byte ordering is what the software (and local processor) expect.

2.5 Diagnostic LEDs

The PASS and FAIL LEDs provide information on the overall operation of the card. They follow the state of the SYSFAIL line when the SYSFAIL jumper is in the ON position. See section 2.2.5.

In addition, there are two diagnostic LEDs for each channel.

The red RUN LED is controlled by application software using the card's control register. This allows a host processor to perform diagnostics on the card and manipulate the state of the LED to give an indication of the results.

The green STATUS LED is under control of the microprocessor on the card. Its function varies depending on the software module which has been loaded onto the card. Refer to the documentation for each module.

2.6 Connecting to the Communication Network

Allen-Bradley specifies Belden 9463 twinaxial cable ("Blue Hose") for their network installations. The Blue Hose is wired into the card using the Phoenix Combicon™ connector supplied with the card. See page 65 for the part number of the connector.

The wiring details depend on the type of network being used. The documentation for each software module includes the specific details for connecting to the various networks. For wiring purposes, pin 1 of each Phoenix connector is identified on the faceplate of the card and is the bottom pin. Pin 1, pin 2 and pin 3 are connected internally to pin 4, pin 5 and pin 6, respectively to make it easy to daisy-chain connections. Any connections to pins 1, 2 or 3 can also be made to pins 4, 5 or 6 respectively.

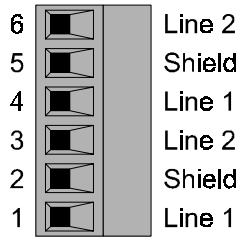
Note:If you daisy chain in this fashion and then remove the connector from the card, the connection through the card is broken.

2.6.1 Connecting to a DH/DHP Network

This section contains information on how to connect the card to an Allen-Bradley Data Highway or Data Highway Plus communication network.

DH/DHP Connection

When you are connecting to a Data Highway or Data Highway Plus network, the card is a single station on the network. The cabling requirements for the card are the same as for any other station on the network. A station is connected to the network trunk-line using a drop-line constructed of Belden 9463 twin-axial cable ("Blue Hose"). The standard rules for cable routing and termination apply.



For Data Highway/Data Highway Plus communication, you connect Line 1 of the network to pin 1 or pin 4 on the card, shield on the network to pin 2 or pin 5 on the card, and Line 2 on the network to pin 3 or pin 6 on the card. Data Highway and Data Highway Plus networks are usually wired with the CLEAR wire as Line 1 and the blue wire as Line 2. In general, if you look at the PLC, whatever color is at the top on the three-pin connector is the color you connect to pin 1 on the card. For modules other than PLCs, pin 1 on the card connects to the pin labelled Line 1 on the module.

Termination

The nodes at the two physical ends of the network should have terminating resistors. All other nodes should not. Every network should have exactly two terminators.

The card does not have an onboard terminator. If you require a terminator, it consists of a resistor between the blue and clear wires. Allen-Bradley recommends a 150 ohm resistor at 57.6 and 115.2 kbaud and an 82 ohm resistor at 230.4 kbaud.

Effects of Cabling Errors

If you are attaching to an existing network, you should be aware that if you make an incorrect connection, you can disrupt the entire network. This applies to any device you attach to the network. Things to watch for are interchanging wires, incorrect termination, shorts to signal lines, or leads on terminating resistors shorting to the shield line. Watch also for the fine braid wires shorting to signal wires.

2.7 Loading a Program on the Card

An application must take the following steps to initialize a channel on the interface card.

1. Make sure the processor is reset by writing to the control/status register for the channel to be loaded. Select the card address by writing to the address register for the channel.
2. Enable card memory by setting the enable bit in the control register.
3. If interrupts are to be used, initialize the interrupt status/ID register for the channel.
4. Perform memory test etc. as required by the application.
5. Set or clear the RUN LED according to the result of the memory test.
6. Load the software module onto the card at the standard memory address set for the channel. Verify that the module has been correctly loaded.
7. Write $A5 into offset $8000 from the start of memory for that channel.
8. Set the run bit in the control register for the channel.
9. Look at the byte at offset $8000. Wait up to 7 seconds for it to change. If it doesn't change, the module did not run on the card. If it becomes 00, everything has run correctly. If it becomes 01, there is a null-terminated string starting at offset $8001 which describes the error. If it has any other value, the result is invalid and the processor did not run correctly.
10. Run your application. Refer to the description for each software module to determine how to use the module and how to tell when you can begin to access the module.

The distribution disk contains a sample loader for a Xycom XVME computer.

2.8 VME Programming Notes (READ THIS!!!!!)

The following information should always be kept in mind when reading the module descriptions.

The structures by which you access the card are all memory mapped. Within the 64K of card memory per channel, the lower 32K is reserved for the card software; the upper 32K is used by the tables by which the host application communicates with the card.

The card processor organizes data in low-byte/high-byte format. Your application must take this into consideration. For example, messages are placed into buffers at locations given by the card software. When your application reads the card to determine the location of the next free buffer and the bytes read are sequentially $00 and $0A, the next free buffer is at offset $0A00 from the start of the data area for that channel (rather than $000A as you might expect in a VMEbus environment.)

The structures on the card contain some elements that are one byte wide. Some compilers by default align structure elements on word boundaries; you must tell your compiler to use byte alignment for structure elements that refer to data on the card.

2.9 Software Modules

The following modules exist for the 5136-SD-VME. Not all modules are shipped with the standard cards.

Module	Purpose
sddhp.bin	standard Data Highway Plus module
sdhdhp.bin	115 kbaud Data Highway Plus module, forces the baud rate to 115 kbaud
sdudhp.bin	230 kbaud Data Highway Plus module, forces the baud rate to 230 kbaud
sddh.bin	standard Data Highway module
sddopt.bin	module to display card options
sddem.bin	remote I/O emulation module, requires a licence from Allen-Bradley. Used on the 5136-SD-VME-R card.

Module	Purpose
sdrmas.bin	module to scan remote I/O. Used on the 5136-SD-VME-S card.

2.10 Card Options

The 5136-SD-VME contains an EEPROM that is programmed before the card is shipped to select which software modules the card can run. The options include:

DH	Data Highway
DHP	Data Highway Plus
MON	Data Highway Plus monitoring (for DH+ Network Analyzer)
NET	network monitor (for DH+ Network Analyzer)
KTEMU	KT emulation
EXEC	the exec module
RIO	for emulating remote I/O. For use on the 5136-SD-VME-R card. Requires a licence from Allen-Bradley. Contact SST for details.
RMAS	for scanning remote I/O. For use on the 5136-SD-VME-S card

By default, cards are shipped with the DH, DHP, KTEMU, MON, NET and EXEC options enabled.

To determine what options are enabled on your card, load the card module SDDOPT.BIN.

The module simply reads the options enabled and displays them at offset $8001, then sets $8000 to 01 (same as an error).

If you try to load a module for which the option is not enabled, the module puts the message "*** Fatal Error ; Option Not Enabled on this Card ***" at location $8001, $8000 gets set to 01 (same as an error) and the module does not run.

2.11 Troubleshooting Installation

This section describes what to do if the card cannot communicate on a network. It also provides more detailed information on some common sources of problems.

- Check cabling for correct wiring to the card.
- Check for shorted wires, leads on terminating resistors shorted to cables, strands from the shield shorting to the other wires.
- Check baud rate
- Check network termination. Only the two nodes at the physical ends of the network should have terminating resistors.
- Check the card station number
- If the network is Data Highway Plus, look at the active station list.

2.12 Related Products

SST roduces the following products:

5136-SD	ISA bus card for DH/DH+
5136-SD-VME-R	VMEbus card which emulates and monitors remote I/O
5136-SD-VME-S	VMEbus remote I/O scanner
5136-SD-MUL	Multibus I card for DH/DH+

The Data Highway/Data Highway Plus cards listed above use the same memory interface as the 5136-SD-VME.

3. Programming the DH/DHP Modules

This section is a guide for writing applications to interface to the SST Data Highway and Data Highway Plus modules.

The programming interfaces for the Data Highway and Data Highway Plus modules are identical. An application that uses the Data Highway Plus software module can use the Data Highway software module without modification.

The Data Highway module is referred to as "DH", the Data Highway Plus module is referred to as "DHP", and references that apply to both the DH and DHP modules are referred to as "DH/DHP".

This section starts with some general information, then describes the modules in detail, with information about the interface between your application and the module and the organization of the tables for the module.

3.1 Comparison of Protocols

Data Highway is a "Floating Master" network. Nodes pass mastership according to the need to send messages. The advantage of this approach is that if there are many nodes, the network is still as fast as if there were few nodes, as long as only a few nodes need to send messages at a given time.

Data Highway Plus is a token passing network. When there are few nodes, Data Highway Plus is much faster than Data Highway because the time-consuming polling scheme used for Data Highway is not necessary.

However, as the number of Data Highway Plus nodes increases, the token passing between nodes degrades network performance.

However, there are other factors to consider when choosing a network.

3.2 DH/DHP Software Overview

The Data Highway/Data Highway Plus software modules (SDDH/SDDHP):

- send and receive messages
- handle all the details of the network with no intervention by the host
- maintain diagnostic counters
- can be configured to reply automatically to diagnostic commands from remote nodes

In addition, the Data Highway Plus module:

- maintains an active station list for the network
- maintains a global data table
- contains a terminal name for the card

The software modules know nothing about message formats. It's up to your application to format messages and put them on the card. Similarly, when the card receives a message from another station, it notifies your application. The application is responsible for reading the message from the card and sending the appropriate reply.

To send a message, the host application software places a message in a transmit buffer on the interface card, then sets a byte to notify the DH/DHP software that there is a message to send. The card sends the message when it next gets the token. When the card receives a reply, it notifies the host application, either by setting a byte which the host polls, or by generating an interrupt to the host. The host reads the remote station's response from a reply queue maintained in memory on the card and processes the contents.

The DH/DHP module maintains the transmit and reply queues, enabling the card to buffer up to 126 Data Highway/Data Highway Plus messages for the application software. You can program the Data Highway Plus module to use 512 byte message buffers, which reduces by half the number of available buffers.

The DH/DHP modules can receive unsolicited messages from remote nodes. The host application must process these messages and send appropriate replies.

The DH/DHP software also provides host interrupt capabilities. Interrupts allow for more efficient use of host CPU time, but require more complex application programming to implement. The card modules can generate interrupts on:

- message received
- transmit buffer available

The modules can automatically respond to diagnostic messages from remote stations with no intervention by the host. The modules maintain their own diagnostic counters.

The Data Highway Plus module supports baud rates of 57.6, 115.2 and 230.4 kbaud. For Data Highway, the only possible baud rate is 57.6 kbaud.

3.3 Programming Overview

The DH/DHP module maintains queues of transmit and receive buffers for applications to send and receive messages. The card maintains a pointer to the next available buffer in a designated memory location.

An application wishing to send a message reads this location to find the next available buffer, then writes the message into the card memory using this pointer to access the buffer. To send the message, the application sets a flag in memory to indicate to the DH/DHP software that the message is ready to send. The DH/DHP software schedules the message on the DH/DHP network, then initializes a pointer to the next transmit buffer and sets the appropriate flags so that the application knows that the card is ready to accept another message.

Upon reception of a message from the Data Highway/Data Highway Plus network, the DH/DHP module initializes a pointer to the received message buffer and sets a flag to indicate to the application that it has received a message. The application can poll the flag location to determine when the card receives a message or the card can notify it by means of an interrupt. After the application has extracted the message from the buffer, the application acknowledges receipt of the message by setting an acknowledgment flag.

The interface between application and card module requires that the application software write into and read from specific memory locations. The examples in this chapter illustrate how to do this, using pointers and structures in the C programming language.

Familiarity with C is helpful to understand the code fragments in this manual. The sample programs were developed for the card using Borland C++ version 3.1.

The file "abdh.h" contains structure declarations and constant definitions. Include this file in any C program that uses the same structures, names, etc. as the sample programs.

It is preferable to use defined constants and masks rather than hard-coded constants. Any changes in future versions of the card software will be easier to accommodate.

Some type definitions have been used:

```
typedef unsigned char  uchr;
typedef unsigned int   uint;
typedef unsigned long  ulng;
```

Note:

One function that is specific to DOS and Borland C is "MK_FP" - make far pointer. This macro is defined by Borland as follows:

```
#define MK_FP(seg,ofs)          ((void far *) \
   (((unsigned long)(seg) << 16) | (unsigned)(ofs)))
/* combine segment and offset into longword of format SSSSOOOO */
```

The MK_FP macro creates a far pointer to an absolute memory location. Other compilers have an equivalent method for assigning pointers to memory locations. Remember that byte ordering is important!

You can avoid much of this simply by using large model to compile your code.

Note:

The sample programs use structures to access specific byte addresses on the card. Some compilers (such as Microsoft C) automatically word-align components of structures unless you tell it to use byte-aligned ("PACKED") structures.

Some changes to the code are required to compile the samples with a different compiler or to run it under a different operating system. To use other languages, you must define structures and constants. The size of the various data types is as follows:

uchr: 8 bits
uint: 16 bits
ulng: 32 bits

If the programming language does not support structures (for example BASIC), then you must define absolute memory locations for each of the structure components using the above data type sizes. Most assemblers allow for the creation of suitable structures.

3.4 DH/DHP Interface Structure

The interface structure controls access to the transmit and receive queues. It also contains various flags and variables that affect overall operation of the DH/DHP module. It is through the interface structure that an application communicates with the DH/DHP module. The structure is defined as follows:

```
typedef struct zif
{
   uint   PC_RXB;       /* 00 ptr to receive buffer     */
   uchr   PC_RXC;       /* 02 RX control                */
   uchr   PC_IFL;       /* 03 interface flags           */
   uint   PC_TXB;       /* 04 ptr to transmit buffer    */
   uchr   PC_TXC;       /* 06 TX control                */
   uchr   STNAD;        /* 07 Station address           */
   uchr   INT_EN;       /* 08 Interrupt Enable byte     */
   uchr   MOD_ID;       /* 09 Module ID                 */
   uchr   DHP_LIST[8]; /* 0a List of active nodes on DH+  */
                        /*    DHP_LIST is valid only for    */
                        /*    Data Highway Plus             */
   uchr   MSG_TOUT;   /* 12 msg timeout, for Data Highway only  */
   uchr   rsrvd1[4];    /*    do not use                   */
   uchr   PC_RX;        /* 17 # of msgs on PC Queue     */
   uchr   rsrvd2;       /*    do not use                   */
   char   term_name[9]; /* 19 terminal name for node    */
   uchr   status;       /* 22 card status               */
   uchr   ext_options;  /* 23 extended options          */
   uint   wr_glob_data;        /* 24 global data to write from this stn    */
   uchr   rsrvd3[0x30-0x26];  /* do not use                */
   uchr   diag_ctrs[35];       /* 30 dh+ diagnostic counters for this stn  */
   uchr   rsrvd4[0x100-0x53]; /* do not use                */
   uint   glob_data[0100];     /* 100 global data (station 0-77 octal)  */
} ZIF;
```

This structure is located at offset 0x0000 from the card base address. (On the VME versions of the card, the structure for a channel is located at offset 0x8000 from the start of memory for the channel). To reference the zif structure, declare a far pointer to it.

```
struct zif far * zp;

zp = MK_FP(CARDADR, 0);
```

To reference components of this structure, use the "zp->component" syntax, i.e.:

```
zp->STNAD = station;   /* set station number */
```

The following sections describe the various members of this structure.

3.4.1 Module ID Byte

An application can use the module ID byte to identify the software module running on the card and the type of card on which it is running.

The upper nibble contains the module type, 0xA0 for Data Highway Plus or 0x50 for Data Highway.

The lower nibble contains the card type which is usually 4.

If the value is 3, the module is running on the 5136-SD revision 1 or the 5136-SD-MCA revision 1 cards.

If the value is 2, the module is running on the 5136-AB-VME card (not the 5136-SD-VME or the 5136-SD-VME2).

3.5 Resetting the Interface

3.5.1 Startup State

When the module is loaded, the DH/DHP module initializes the PC_TXC byte to IFT_IRS (interface requires reset). The card does not go online on the Data Highway/Data Highway Plus network until the host issues a reset command (IFT_RES) to PC_TXC. This lets you initialize the station address (STNAD), the interface flags (PC_IFL) register and the extended options register before you put the card online.

3.5.2 Transmit Control Byte

The transmit control byte is used to reset the interface between the host and the card and also to control access to the transmit queue. The transmit control byte can have the following values:

```
#define IFT_WT  0x04  /* waiting for message to send  Set by Card */
#define IFT_RES 0x0c  /* interface reset Set by host */
#define IFT_SM  0x10  /* send message    Set by host */
#define IFT_IRS 0x1c   /* interface requires reset,Set by Card or by host  */
```

3.5.3 Station Address

The station address is the address that the DH/DHP module uses on the Data Highway/Data Highway Plus network. For Data Highway, the valid range is 0-376 octal (0-0xFE). (377 is a broadcast address and should not be used.) For Data Highway Plus, the valid range is 0-77 octal (0-0x3F).

To set the station address:

```
zp->STNAD = SRC; /* configure station address */
```

There are two important points regarding selection of a station address.

First, every station must have a unique station address. If the card uses the same station address as another Data Highway/Data Highway Plus station, unpredictable communication results.

Do not change the station address while you are online. Otherwise, the DH/DHP module loses track of current station information and the results are unpredictable.

We recommend that you do not use station address 0 for any device on the network because some communication errors are undetectable by the CRC hardware on the device.

3.5.4 Interface Flags

The interface flags (PC_IFL) control various features of the card modules. Set these option bits before you issue the reset command. The flags are defined as follows:

```
#define PFL_57K      0x00       /* 57.6  Kbaud DHP   */
#define PFL_115K     0x01       /* 115.2 Kbaud DHP   */
#define PFL_230K     0x02       /* 230.4 Kbaud DHP   */
#define PFL_DUP      0x04        /* duplicate message check    */
#define PFL_XDC      0x08        /* execute diagnostic commands */
#define PFL_RTN_MSGS 0x10 /* return msgs to offline stns */
#define PFL_FRCE_32K 0x20   /* force 32k map even if 16k   */
```

```
#define PFL_USE_REM  0x40   /* allow remote addressing     */
#define PFL_MULT_MSG 0x80 /*send multiple message on token*/
```

The lower two bits determine the baud rate. For Data Highway, the only possible baud rate is 57.6 kbaud. For Data Highway Plus, possible baud rates are 57.6, 115.2 and 230.4 kbaud.

Baud rate	bit 1	bit 0
57.6	0	0
115.2	0	1
230.4	1	0

PFL_DUP indicates to the DH/DHP module that it should check incoming and outgoing messages for duplicate messages and ignore them if they occur. Every DH/DHP message has a transaction number that increments on every new message. This ensures that no two packets in a given period are the same. If, for any reason, the DH/DHP module receives a message identical to an immediately previous message, it ignores the second message if the PFL_DUP flag is 1. If your application writes duplicate messages to the card and you have enabled duplicate message detection, the card discards the duplicate messages and does not send them out on the network. If you are retransmitting a message for whatever reason, remember to change the transaction number.

Data Highway Plus only: If you set bit 4, **PFL_RTN_MSGS**, the card first checks the active station list before it attempts to send a message. If the destination station is offline, the card returns the message to the sending application with an error status of 7, without trying to send the message on the network. This minimizes disruptions to the network. If the bit is zero, the card sends the message without first checking the active station list to see if the station is online. If the station is offline, the module sends the message three times, then returns the message with a status of 2.

PFL_XDC controls whether the DH/DHP module responds to any diagnostic commands it receives. If this bit is 1, the DH/DHP module processes diagnostic commands. It does not forward them to the host application. If you clear this bit to 0, the DH/DHP module passes diagnostic messages to the host system. The host application is then responsible for sending a reply to the diagnostic command. Under almost all circumstances you should set this bit to 1 and let the DH/DHP module process diagnostic commands, unless a specific application needs access to the incoming diagnostic commands. Refer to "Data Highway/Data Highway Plus Protocol and Command Set" and section 3.13 on page 50 for information on diagnostic commands.

PFL_FRCE_32K forces the DHP module to use 32K mapping even though the card occupies 16K in the host memory space. You must do your own bank switching if you use this flag/option. It should be used in very few cases. Note that you can use this only on the 5136-SD revision 2; the revision 1 card doesn't support the 16K mapping. It serves no purpose on the 5136-SD-VME.

PFL_USE_REM enables the DHP module to send remote messages (DHII Links) using the value in the "rem" field of the "mu" structure to form the remote message. If this bit is 0, the card ignores the "rem" element of the message. Use of remote link messages allows bridging between separate Data Highway Plus networks that are interconnected with a Data Highway or Data Highway II backbone.

Refer to section 3.9 for more information on remote messages.

Data Highway Plus only: If you set bit 7, **PFL_MULT_MSG**, the card sends as many messages as it can (the number depends on the byte count in the messages) before it passes the token. If the bit is zero, it sends just one message and then passes the token.

3.5.5 Extended Options Byte

The extended options (ext_options) register controls various features of the Data Highway Plus module. The flags are defined as follows:

```
#define SND_GLOB 0x01            /* send global data for local station */
#define LARGE_BUFFERS  0x02   / * use 512 byte buffers */
```

SND_GLOB enables global data on Data Highway Plus. You can set and reset this bit at any time. Since other bits in the byte must not be affected, read the current values and mask the other bits before you change the value of this bit.

LARGE_BUFFERS tells the DHP module to use 512 byte message buffers. You must set this bit before you put the card online.

Note: If you use 512 byte buffers, you must write to the len_hi member of the message buffer every time you send a message.

3.5.6 Summary of Interface Reset

You must reset the DH/DHP after you load the card or when you have taken the card offline and are putting it back online. This frees all buffers and allows a station address to be set. Initiate interface reset as few times as possible since there is some disturbance to the Data Highway/Data Highway Plus Network when the DH/DHP Module goes off line and then comes back on again.

You must be careful with the timing of the reset command. If PC_TXC is in a state that the DH/DHP software is responsible for changing (such as IFT_SM) it can miss the reset command. For this reason, issue the reset command only when PC_TXC is in the IFT_WT or IFT_IRS state.

The sample programs supplied with the card show how to reset the card.

On a Data Highway Plus network, you can determine if you are online by looking at the Active Station list (see section 3.11). On a Data Highway network, there is no way to determine if you are online since nodes transmit only when they have messages to send.

3.5.7 Interrupt Enable Byte

If your application uses host interrupts, you must set the bits in the interrupt enable byte to allow receive and/or transmit interrupts. Set this byte after you issue a card reset and the card is waiting for a message but before you send any messages. The various flags are defined as follows:

```
#define EN_RX    0x01    /* enables receive interrupts  */
#define EN_TX    0x02    /* enables transmit interrupts */
#define INT_PEND 0x80  /* interrupt pending          */
```

For example, the following code fragment enables both receive and transmit interrupts.

```
/* allow card software module to generate interrupts now */
zp->INT_EN = EN_RX + EN_TX;
```

The card generates a receive interrupt when it receives a message. The card generates a transmit interrupt when a transmit buffer becomes available. Section 4.17 describes interrupts in more detail.

3.5.8 Status Byte

Data Highway Plus: When you put the SDDHP module online, it first listens to the network to determine what nodes are already on the network. If it finds that the node number you have assigned it already exists on the network, it does not go online.

You can monitor this process by reading the status byte.

```
/* constants used by status */
#define TESTING   0x00              /* testing, wait for non-zero */
#define CARD_OK   0x80             /* card is online ok         */
#define DUP_STATION 0x01        /* duplicate station found */
#define INVALID_STATION 0x02 /* invalid station found   */
```

While the card is listening to the network, the status byte has the value TESTING. If the card finds that the node already exists on the network, it sets the status byte to the value DUP_STATION. Otherwise, it sets the status byte to the value CARD_OK and goes online.

5136-SD/MCA only: If you are using the dual-application interface with the KT emulation modules, the card returns INVALID_STATION if the station number does not match that set in the other part of the interface.

Data Highway: The SDDH module presets the status byte to CARD_OK since it has no way of determining if this node number is a duplicate.

DHP: If you try to send a message while your station number is a duplicate of that used by another node, the card immediately returns the message with a status (STS) of 6.

Note: If the only other node on the Data Highway Plus network is an enhanced PLC-5 (PLC-5/20, PLC-5/40, etc.), a SLC 5/04, or a PLC-5/250 you should be aware that these PLCs check the network for new nodes only approximately once every 5 seconds. It's possible to put the card online at the same node number as one of these PLCs if the card goes online when the PLC isn't checking the network at the moment you go online. When the PLC next looks for new nodes, it sees the card at the same node number and doesn't go online.

3.5.9 MSG_TOUT (DH only)

This byte applies only to Data Highway. It is required only in special circumstances and should not be modified unless you are told to do so by SST personnel!

3.5.10 Taking the Card Offline (DHP only)

When you are using the Data Highway Plus module, you can take the card offline by placing IFT_IRS command in PC_TXC. Before you issue the IFT_IRS command, PC_TXC must be in the IFT_WT state.

```
for (tme = 100; tme && (zp->PC_TXC != IFT_WT); tme--)
   delay(10);
if (zp->PC_TXC == IFT_WT)
    zp->PC_TXC = IFT_IRS;    /* take card off line */
else
  error("Bad Card State");
```

The card is offline when the value in PC_RXC is IFT_IRS (0x1C). Taking the card offline restores the module to the startup state.

The time for the card to go offline depends on such factors as whether the card is sending a message, who has the token, etc.

3.6 DH/DHP Messages

For a complete discussion of Data Highway Plus messages, refer to "Data Highway/Data Highway Plus Protocol and Command Set", Allen-Bradley publication 1770-6.5.16. The messages can take on various forms, but all messages begin like this:

DEST | SOURCE | CMD | STATUS | TNS LOW | TNS HIGH | ...

(TNS is the transaction number associated with the message.)

3.6.1 Message Queues

The DH/DHP modules implement two queues. The transmit queue is used for transmitting messages to Data Highway/Data Highway Plus, the receive queue is used for receiving messages from the Data Highway/Data Highway Plus.

Applications written to interface with the DH/DHP modules place messages into and remove replies from these queues. The DH/DHP software can maintain up to 126 buffers for use by the receive and transmit queues.

An application program should monitor the progress of each outstanding message using a timer. A typical time is 5 seconds but it depends on how busy the network is. If the timer expires before the card receives a reply, flag an error condition and perhaps re-transmit the message. The sample programs show how to implement this.

Each message transmitted has a transaction number. It is up to the application to check that the transaction number in a reply message matches the transaction in the command message. If they do not match, indicate an error condition. The sample programs show how to do this.

3.6.2 Message Buffers

The card uses most of the available memory as a pool of message buffers for storing messages received or messages to be transmitted.

The message unit buffer structure, MU, describes the format of the buffer used by the DH/DHP software to store messages. Each message unit is large enough to hold Data Highway/Data Highway Plus messages of up to 250 bytes (larger if you are using 512 byte buffers).

```
typedef struct tagMU
{
uchr len_hi;        /* hi byte of length if LARGE_BUFFERS is on */
```

```
uchr rem;          /* used for remote addressing, DH+ only*/
uchr len;          /* length of message data            */
uchr dst;          /* destination                    */
uchr src;          /* source address                 */
uchr cmd;          /* message command/reply          */
uchr sts;          /* reply status code (cmd = 0)    */
uint tns;          /* sequence number                */
uchr var[244];   /* variable data                  */
uchr crc[2];       /* sdlc crc (reserved)            */
uchr spare;      /* spare byte                     */
} MU;
```

This structure is a generic Data Highway/Data Highway Plus message template. If messages fit a more specific format (e.g. PLC-5 word range read/write) then declare a structure corresponding to the specific type of addressing.

```
struct PLC5_MSG     /* PLC-5 Word Range Read/Write message */
{
   uchr len_hi;
   uchr rem;
   uchr len;
   uchr dst;
   uchr src;
   uchr cmd;
   uchr sts;
   uint tns;
   uchr fnc;
   uint pack_ofs;
   uint total_trans;
   uchr data[239]; /* address and data here, together */
};                      /* because address length is variable */
```

3.6.3 Accessing Message Buffers

To access message buffers, use the transmit buffer pointer (PC_TXB) or the receive buffer pointer (PC_RXB).

The DH/DHP module has no control over the sequence of replies when it sends multiple messages over the communications network. An application program can use the transaction number and source station address to match replies with transmitted messages. The following code fragment demonstrates how to retrieve a reply from the receive queue and match it with the transmitted message. Use this example in conjunction with the transmit routine described above. It assumes that both the transmit and receive queues are one message long.

```
struct MU far *reply;
reply = MK_FP(CARDADR, zp->PC_RXB);
/* check if reply matches transaction number and destination
     station of last transmitted msg */
if (reply.tns == last_tns && reply.src == last_station)
```

```
   {
   /* yes, correct reply msg          */
   /* application can use msg reply now */
   correct_reply(reply);
   }
zp->PC_RXC = IFR_MR;    /* acknowledge message received */
```

If you send multiple messages before checking for responses, the tracking of transaction numbers becomes more complex. You must save the transaction number of each message sent and start a timeout for each message. When you receive a reply, stop the message timer associated with the transaction number and process the reply appropriately.

It may be useful to use some of the bits in the transaction number to indicate the host application's internal storage of the status of the message. For example an application could use 16 "mailboxes" to store timeout, status, and task ID information for a message. When you send a message, open a mailbox and use 4 bits of the transaction number to indicate the number of the mailbox used. When you receive the reply, extract these 4 bits from the transaction number of the reply and use them to access the proper mail box.

3.6.4 Message Length

The "len" byte is used to store the length of the command or reply. Note that "len" is the length of the Data Highway/Data Highway Plus message only. It does not include the first three parameters required for interfacing with DH/DHP ("len_hi", "rem", “len”). It includes all data in the message, starting with the "destination station" field.

When you are sending a message, you must write the value of the message length. When the card receives a message, the card writes the length, which you can use to extract the message from the buffer on the card.

If you are using 512 byte message buffers, you must write the high byte of the length to the len_hi byte of the message structure every time you send or receive a message.

3.7 Sending Messages

To send a message, an application must:

- wait for the state IFT_WT in PC_TXC
- get the offset to the buffer from PC_TXB
- write the message to the buffer
- write the length
- set PC_TXC to IFT_SM

First, monitor PC_TXC for the state IFT_WT. This state indicates that the DH/DHP software is ready to transmit a message. The application should set up a timer while waiting for IFT_WT. The following example shows a procedure that waits for the IFT_WT state:

```
int clear_tx(void)
   {
   int tme;
   /* wait for ready to transmit    */
   for (tme = 0; (zp->PC_TXC != IFT_WT) && (tme < 2000); tme++)
      {
      delay(1);       /* delay(n) waits n milliseconds */
      }
   return (zp->PC_TXC == IFT_WT);
   }
```

The application can also use an interrupt to signal that a transmit buffer is available. Refer to section 3.17 for more information on using interrupts.

When the status is IFT_WT, put the message in the next available transmit buffer. Initialize a pointer to a message buffer using the transmit buffer pointer (PC_TXB) as follows:

```
mu = (struct MU far * )MK_FP(CARDADR, zp->PC_TXB);
```

Now use this pointer to place the message into the buffer:

```
mu->src = src;          /* Source station (same as STNAD) */
mu->dst = dst;          /* Destination station          */
mu->cmd = cmd;          /* Command                     */
mu->sts = 0;            /* Status                   */
mu->tns = tns;          /* Transaction Number          */
mu->len = msg_size;  /* Message Length              */
movedata( _DS, (unsigned)msg, CARDADR, zp->PC_TXB + 3, msg_size);
```

If you are using 512 byte message buffers, you must write the high byte of the length to the len_hi byte of the message buffer every time you send a message.

When the message is in place, write the message length, then tell DH/DHP to send the message by setting PC_TXC to IFT_SM:

```
zp->PC_TXC = IFT_SM;      /* send msg */
```

Remember that PC_TXC controls access to the transmit queue; it does not directly cause transmission of the message. The message is sent from the queue under control of the DH/DHP software. For example, the Data Highway Plus module cannot send the message until it gets the token. The card places responses in the receive queue as they arrive.

Once you tell the card to send the message, it changes the contents of the message buffer and you can no longer look at the message buffer to verify the contents.

If you send a message and haven't received an error reply within approximately 1 second, the destination station received the message. The time is not precise since it depends on the number of nodes on the network, network traffic, etc.

3.8 Receiving Messages

The receive control byte (PC_RXC) controls access to the receive queue. It can have the following values:

```
/*  constants used by PC_RXC */
#define IFR_WT  0x04  /* interface inactive (set by card) */
#define IFR_MP  0x08  /* message present    (set by card) */
#define IFR_MR  0x10  /* message received   (set by host) */
#define IFR_SR  0x14  /* send reply to message rcv'd      */
#define IFR_IRS 0x1c  /* interface requires reset (set by card) */
```

You detect reception of a message by polling the PC_RXC byte for the IFR_MP state:

```
int clear_rx(void)
   {
   int tme;
   /* wait for message present */
   for (tme = 0; (zp->PC_RXC != IFR_MP) && (tme < 2000); tme++)
      delay(1);
   return (zp->PC_RXC == IFR_MP);
   }
```

An application can also use an interrupt to signal when the card receives a message. Refer to section 3.17 for more information on using interrupts.

When a message is present, create a pointer to the receive buffer using PC_RXB as follows:

```
mu = MK_FP(CARDADR, zp->PC_RXB);
```

Extract all the information from the buffer before you acknowledge reception of the message. Use the mu->len to determine the message size. If you are using 512 byte buffers, the high byte of the length is in the len_hi byte of the message buffer.

For example, to extract a message using Borland C:

```
movedata(CARDADR, zp->PC_RXB + 3,  _DS,(unsigned)reply, mu->len);
```

Next, acknowledge the message so that the DH/DHP module can reuse the buffer:

```
zp->PC_RXC = IFR_MR;       /* acknowledge msg */
```

When you acknowledge reception of the message, the buffer is free in a worst case time of approximately 500 microseconds.

The PC_RX byte in the zif structure indicates the number of receive messages on the queue to the host PC. If PC_RXC indicates there is a message present and PC_RX contains 3, then there are 4 messages present on the card.

3.8.1 Unsolicited Messages

Unsolicited messages are messages sent from another station to the DH/DHP station. For example, you can create them using the MSG instruction in a PLC.

To receive an unsolicited message, you must poll the PC_RXC location in the zif structure for a message present (IFR_MP) or use an interrupt to signal the arrival of a message. Retrieve the contents of the message, then acknowledge the receipt of the message by setting PC_RXC to IFR_MR as with any other message.

If the message is a diagnostic command and you have set the PFL_XDC bit in the interface flags byte (page 33), the DH/DHP replies to the sender. Otherwise, your application is responsible for formatting and sending an appropriate reply to the message. If you don't reply, the message instruction in the PLC eventually times out and the PLC cannot send more messages until the error is cleared.

If your application doesn't remove and acknowledge unsolicited messages, eventually the card runs out of buffers and returns a no memory NAK to the sender in response to any further unsolicited messages.

3.8.2 Using "Send Reply"

You can build the reply to a message in the same buffer in which the message was received. The only advantage to this method is that you don't have to wait for a transmit buffer, you just use the buffer you already have.

You must swap destination and source, provide a suitable status value, and make any other changes necessary such as inserting the data if the message received was a read request. You must also change the value of the length.

The final step is to set the PC_RXC byte to IFR_SR (send reply) instead of using IFR_MR.

3.9 Offlink Addressing using 1785-KAs (DHP only)

To use offlink addressing, before you go online you must first set bit 6 (PFL_USE_REM) in the PC_IFL register.

Messages can be routed between Data Highway Plus networks by bridging the networks with an intermediate Data Highway network, using 1785-KA modules.

The routing information is added as 11 extra bytes between the SRC and CMD bytes of the normal Data Highway Plus message.

The general format for offlink messages is:

| DST | SRC | 0x24 | DID_LO | DID_HI | DNDE_LO | DNDE_HI | 0x80 | SID_LO | SID_HI | SNDE_LO | SNDE_HI | 00 | CMD | STS | ...

where:

DST	DHP station address of local bridge node
SRC	DHP station address of this node
DID	destination node's link id, see A-B documentation (0 for KA bridge)
DNDE	station address of destination on destination link high 2 bits of remote KA DH address OR'ed with low 6 bits of final destination DHP address
SID	link id of local network, see A-B documentation (0 for KA bridge)
SNDE	DHP station address of this station (same as SRC)

so the message format can be simplified as:

| ka_loc | SRC | 0x24 | 00 | 00 | ((ka_rem & 0xC0) | dst) | 00 | 0x80 | 00 | 00 | SRC | 00 | 00 | CMD | STS | ...

where:

ka_loc	local KA DHP address
SRC	DHP address of this station
ka_rem	DH address of remote KA
dst	DHP address of remote final destination

You can use the REM_MU structure in abdh.h for remote messages.

Other bridge networks can also be used such as DH-485 (KA5 bridge modules) or Data Highway II (KP5 bridge modules). If you need to use one of these bridge networks, contact technical support at SST for more information.

3.10 Queueing Messages

The examples provided show the basic operation of the DH/DHP module. You can greatly increase DH/DHP network throughput by sending several messages at once. You can detect replies to these queued messages by polling the card or by using the DH/DHP module's host interrupt capabilities. This requires more application software overhead to perform reply matching and timeouts. The DH/DHP module's queueing capability allows for management of 126 message units. Since the card releases a message unit after it transmits the message, you can send up to 119 messages at once and the round up replies as processing time permits. Note that the DH/DHP module releases message buffers for sending messages only if there are more than 7 buffers available. This guarantees that the card can buffer up to 7 unsolicited messages even if the transmit queue is full. This dictates the limit of 119 message queue-ahead.

If you use 512 byte message buffers, the number of buffers is reduced to 63.

3.11 Active Node List (DHP only)

For Data Highway Plus only, the card maintains an active station list. This active station list is read only and is located at offset 0x0a from the start of card memory. The host application may determine if a particular station is present by checking the bit that corresponds to the station in the active station list. If the bit is 1, the station is present, otherwise the station is not present. The active station list is arranged as follows:

Byte #	0	1	2	3	4	5	6	7
0	#00	#10	#20	#30	#40	#50	#60	#70
1	#01	#11	#21	#31	#41	#51	#61	#71
2	#02	#12	#22	#32	#42	#52	#62	#72
3	#03	#13	#23	#33	#43	#53	#63	#73
4	#04	#14	#24	#34	#44	#54	#64	#74
5	#05	#15	#25	#35	#45	#55	#65	#75
6	#06	#16	#26	#36	#46	#56	#66	#76
7	#07	#17	#27	#37	#47	#57	#67	#77

If your application has set the PFL_RTN_MSGS bit in the interface flags byte (page 33), the card checks the active station list before it sends the message on the network.

3.12 Terminal Name (DHP only)

This applies to SDDHP version 5.04 and above. Do NOT use terminal names if you are using an earlier version - it disrupts the software on the card. If you are using a version earlier than 5.04, upgrade by contacting SST.

Nine bytes are reserved in the zif structure for a terminal name, eight bytes for data and one byte for a null terminator. (The card software reads either the first 8 bytes of data or until it encounters a null character.) Initially the module sets these bytes to zero.

You can write a terminal name here after you issue the card reset command and the card acknowledges by setting PC_TXC to the IFT_WT state. As soon as the card module sees a non-zero value in the first byte, it responds to diagnostic reads as a terminal (FE1B in the module type) rather than as a computer (FF). This applies only to Data Highway Plus and not to Data Highway. There are no station names on Data Highway.

3.13 Diagnostic Commands

If you set the PFL_XDC flag in the interface flags register, the DH/DHP modules automatically respond to diagnostic commands from remote stations. Otherwise the card passes diagnostic commands to the host application as unsolicited messages. The host application is then responsible for sending the appropriate replies to these commands.

3.13.1 Diagnostic Counters

The internal diagnostic counters record information such as the number of bad transmissions and the cause of transmission errors. These counters are useful for diagnosing sources of communication problems on the network. To read the counters, issue a diagnostic read command. If the destination for the read is the local station (the card), the DH/DHP module traps the diagnostic read command and returns a reply that contains the state of the error counters. The document "Data Highway/Data Highway Plus Protocol and Command Set" describes the format of this command.

The card uses the same counter format as the 1770-KF2, except that the KF2 maintains counters related to asynchronous communication on the serial link between the computer and the 1770-KF2. Since there is no serial link using the card, you can ignore these counters. The card returns them in response to a diagnostic read command but they are always zero.

To monitor the counters of a remote station, issue a diagnostic status command to determine the base address of the diagnostic counters in the remote station. Then send a diagnostic read command to extract the information from that address. The sample diagnostic program dhpd.c supplied with the card shows how to monitor diagnostic counters.

To clear the diagnostic counters, use:

cmd = 6

fnc = 7

The tables on the next two pages summarize the diagnostic counters.

Data Highway Diagnostic Counters (KF2 format)

Offset from start of data	Description	Length in bytes
0	Rec'd ACK/NAK bad CRC	1
1	ACK timeout	1
2	Contention	1
3	Bad ACK status	1
4	Returned Messages	1
5	Transmit memory full	1
6	Poll timeout	1
7	False Poll	1
8	Receiver heard status	1
9	Frame too small	1
10	Wrong destination address	1
11	Receiver Memory Full	1
12	Bad frame status	1
13	Buffer overflow	1
14	Memory overflow	1
15	Re-transmits	1
16	Aborts	1
17	Transmitted Messages	1
18	No counter	1
19	Received Messages	1
20	No counter	1

Data Highway Plus Diagnostic Counters (KF2 Format)

Offset from start of data	Description	Length in bytes
0	Rec'd ACK/NAK bad CRC	1
1	ACK timeout	1
2	TX re-tries Exhausted	1
3	NAK-Ileg.Prot. Rec'd	1
4	NAK-Bad LSAP Rec'd	1
5	NAK-No Mem. Rec'd	1
6	Rec'd ACK/NAK too short	1
7	Rec'd ACK/NAK too long	1
8	Unrecognized ACK/NAK	1
9	Token Pass Timeout	1
10	Token Pass Re-tries Exhausted	1
11	Token Claim Sequence Entered	1
12	Token Claimed	1
13	Bad CRC in Rec'd Frame	1
14	NAK-Ileg.Prot. Sent	1
15	NAK-Bad LSAP Sent	1
16	NAK-No Mem. Sent	1
17	Rec'd Frame too Small	1
18	Rec'd Frame too Large	1
19	Rec'd a Duplicate Frame	1
20	Rec'd Frame Aborted	1
21	Message Successfully Sent	2
23	Message Successfully Rec'd	2
25	Command Successfully Sent	2
27	Reply Successfully Rec'd	2
29	Command Successfully Rec'd	2
31	Reply Successfully Sent	2
33	Reply Could Not Be Sent	1
34	Active Stations	1

3.13.2 Diagnostic Status

Data Highway Plus

When the DHP module receives a diagnostic status command (CMD 06, FNC 03), the format of the reply depends on whether you have assigned a station name. If there is no station name, the reply format is:

Byte	Typical Value	Meaning
1	00	this byte is always 0
2-3	FF 00	this station is a computer
4	A4	Module ID from MOD_ID
5-6	800A	Offset to active station list
7-8	8030	Offset to diagnostic counters
9	4	Software revision
10	1E	contents of PC_IFL

If you have assigned a station name, the format is:

Byte	Typical Value	Meaning
1	00	this byte is always 0
2-3	FE 1B	this station is a terminal
4	A4	Module ID from MOD_ID
5-6	800A	Offset to active station list
7-8	8030	Offset to diagnostic counters
9	4	Software revision
10	1E	contents of PC_IFL
11-18		terminal name

Data Highway

On Data Highway, the reply uses the first format shown above.

3.13.3 Other Diagnostic Commands

The DH/DHP modules respond to the remaining diagnostic commands if you have set the PFL_XDC bit in the interface flags register.

The modules respond appropriately to the Diagnostic Counters Reset (CMD 06 FNC 07) and the Diagnostic Loop (CMD 06 FNC 00) commands. The modules send replies to the Set ENQs (CMD 06 FNC 06), Set NAKs (CMD 06 FNC 05), Set Timeout (CMD 06 FNC 04) and Set Variables (CMD 06 FNC 02) commands but these commands have no effect since they relate to the asynchronous interface, which doesn't exist for the DH/DHP modules.

3.14 Status (STS) Errors

The DH/DHP modules return the following local status (STS) values:

Definition	Value	Meaning
STS_NOMEM	0x01	destination out of memory and could not accept message
STS_NOACK	0x02	destination did not send ACK, usually caused by non-existent destination
STS_CONTENTION	0x03	Contention. Unrecognized response from destination. Caused by a duplicate station or just general problems on the network.
STS_DISCON	0x04	local port is disconnected (DH+ only), usually caused by no network.
STS_DUPL_STAT	0x06	duplicate station detected
STS_OFFLINE	0x07	station is off-line, returned if PFL_RTN_MSGS=1
STS_DUPL_STS	0x0E	card received a duplicate transaction

The following status values are reserved to be used by an application. These values are used by the sample programs supplied with the card.

Definition	Value	Meaning
STS_TOUT	0x05	timeout waiting for reply
STS_TNS_MISM	0x0D	Reply contained a transaction number that did not match the TNS in the command.

Some common remote status errors and their meanings are:

Remote Status	Meaning
F007	file is wrong size. Also occurs if start address + length refers to an address that doesn't exist.

Remote Status	Meaning
F011	illegal data type
F012	invalid parameter or invalid data

3.15 Global Data (DHP Only)

On Data Highway Plus, each node can pass one word of global data with the token. This is supported by the Data Highway Plus module, SDDHP, starting with version 5.13

The global read table, 64 words starting at offset 0x100, one word per station, shows the global data for each station on the network, regardless of whether the local node (SD card) is sending global data.

To enable sending global data, set bit 0 in the extended options register (offset 0x23) to 1. Then write the value to pass with the token in the word at location wr_glob_data in the ZIF structure.

Global data from the card can be enabled or disabled at any time.

When a remote node with global data goes offline, the DHP module clears the global data for that node to zero.

3.16 Summary of Memory Locations

The DH/DHP interface structure starts at the installed address of the card plus the offset and contains the following items.

Item	Description	Offset	Size in bytes
*PC_RXB	offset to receive buffer	0x00	2
PC_RXC	receive control	0x02	1
PC_IFL	interface flags	0x03	1
*PC_TXB	offset to transmit buffer	0x04	2
PC_TXC	transmit control	0x06	1
STNAD	station address	0x07	1
INT_EN	interrupt interface	0x08	1
*MOD_ID	module ID	0x09	1
*DHP_LIST	active DHP nodelist	0x0A	8
MSG_TOUT	DH only, message timeout	0x12	1
*PC_RX	# of RX msgs on PC Q	0x17	1
term_name	terminal name	0x19	9
status	card status	0x22	1
ext_options	extended options	0x23	1
wr_glob_data	global data for this station	0x24	2
diag_ctrs	DHP diagnostic counters for this station	0x30	35
*glob_data	global data	0x100	128

* These bytes should not be modified by an application. They should only be read by the application.

The 8 bytes set aside for the DHP_LIST are used to maintain a list of active Data Highway Plus stations. The DHP_LIST does not apply to Data Highway. They are not used in the DH software.

Pointers to the message unit buffers (MU) are stored in PC_RXB and PC_TXB. Each MU contains the following members:

Item	Description	Offset	Size in bytes
len_hi	used by DHP	0	1
rem	used for remote messages	1	1
len	length of message data	2	1
dst	destination station	3	1
src	source station	4	1
cmd	command code	5	1
sts	reply status	6	1
tns	transaction sequence number	7	2
var	variable length message data	9	maximum 244
crc	used by DHP	253	2
spare	used by DHP	255	1

3.17 Interrupts

The DH/DHP module can generate host computer interrupts when it receives a message or when a transmit buffer is available. This relieves the host of having to poll the card and allows for more efficient use of host CPU time. The use of interrupts does, however, increase the complexity and overhead required in the host application software.

Interrupts are enabled by setting the appropriate flags in the INT_EN byte of the zif structure. The location of the zif structure and the bit patterns to enable transmit and receive interrupts are listed in the abdh.h header (EN_TX and EN_RX respectively).

The first step in setting up interrupts is to choose the hardware interrupt level on which interrupts are to operate. See section 2 "Installation" for information on setting an interrupt level.

The ISR must follow a specific sequence to ensure that no interrupts are missed. It is assumed that the compiler handles register preservation and other necessary low level steps.

A typical interrupt sequence is listed below.

- A channel receives a message generating an interrupt request, or the card is ready to accept a new message to send out.
- the interrupt handler checks the interrupt pending bit of the status register to determine which channel has generated the interrupt.
- the interrupt handler clears the interrupt pending bit by writing a 1 to bit 2 of the control/status register (offset 1)
- the handler checks the status of PC_RXC and PC_TXC to determine if the interrupt was caused by a message being placed in the receive queue or a buffer becoming available in the transmit queue.
- the handler services interrupt as required.
- after interrupt servicing is completed the handler clears the INT_PEND bit of the INT_EN byte to enable interrupts in software.

DH/DHP Message Processing

If you have enabled receive interrupts, the interrupt service routine should check to see if any messages are available. If a message is available, copy it out of the buffer and then acknowledge it using "IFR_MR". The card does not pass another receive message to the host CPU until it receives the message acknowledgement.

If transmit interrupts are enabled, the ISR should check to see if the card is ready to transmit (PC_TXC == IFT_WT). If it is, and the application has a message to send, the application should load the message into the appropriate buffer and send the message using "IFT_SM". If no message is ready to send, the host should set a flag to indicate to the main loop of the application software that it must initiate transmission of the next message rather than placing the message on the application's transmit interrupt queue.

The final step in processing DH/DHP interrupts is to tell the card that current card interrupt servicing is complete and the card may generate new interrupts.

Reset the "INT_PEND" bit (bit 7) of the "INT_EN" software register in the DH/DHP module. Do not change any other bits in this register.

The two channel card operates as two separate and independent channels so if the same interrupt number is used for both channels, it must be serviced as any other chained interrupt i.e. the interrupt handler must check all channels on a given interrupt.

To inhibit further card interrupts, clear the "TX_EN" and "RX_EN" bits of the "INT_EN" register.

3.18 Sample Programs

The sample programs provided with the card provide simple examples of the use of the DH/DHP modules. The sample programs are each stored in their own directories on the distribution disk, along with any programs they need to be linked with to create the executable program.

DHPD

The program initializes the DHP module, displays a list of all active stations on the network, and monitors and displays diagnostic counters of any Data Highway Plus station. Program syntax is as follows:

DHPD <local node address> <memory address> [<terminal name>]

An 'h' in front of the local node address sets the baud rate to 115.2 kbaud. A 'u' in front of the local node address sets the baud rate to 230.4 kbaud.

Examples:

DHPD 01 C800

DHPD u4 D000

This program is especially useful for determining that the card is working correctly on the network and for understanding the A-B diagnostic command system

SB

This is a simple program that simply writes out blocks of data to a PLC-5 integer file over Data Highway/Data Highway Plus using the DH/DHP Module. Consider the consequences of writing to the PLC integer file before running this program. If you are writing to the data table in a running processor, you may be overwriting real data and disrupting the normal operation of the system.

The file to which the program is writing must exist in the PLC or the program returns an error.

SBB

This is a simple program that writes out a block of data to a PLC-5 integer file over Data Highway/Data Highway Plus using the DH/DHP Module. It uses multiple packet reads and writes. Consider the consequences of writing to the PLC integer file before running this program. If you are writing to the data table in a running processor, you may be overwriting real data and disrupting the normal operation of the system.

The file to which the program is writing must exist in the PLC or the program returns an error.

SBBAS

This is a modified version of the SB program that reads and writes integers using the basic command set.

SBTYP

This is a modified version of the SB program that reads and writes integers using PLC-5 typed read and write commands and logical ASCII addressing.

SBLOGTYP

This is a modified version of the SB program that reads and writes integers using PLC-5 typed read and write commands and logical binary addressing.

SBSLC

This is a modified version of the SB program that reads and writes integers to a SLC 5/04 using SLC protected typed logical read and write commands.

UNSOL

This is a sample program that demonstrates processing unsolicited commands.

Technical Data

Card Type	5136-SD-VME
	5136-SD-VME2
Function	VMEbus interface card for Allen-Bradley networks
Description	IEEE 1014, 6U height, P1 compatible
	Memory SD16, SD08(EO), SADO24
	Registers SD08(O),SADO16
	Standard Addressing: 64 Kbytes on any 64 Kbyte boundary, per channel
	Short addressing: 6 bytes on any 1 Kbyte boundary, per channel
	Interrupt capability: switch selected level 1-7, software set 8-bit vector, release on acknowledge (ROAK)
Current Consumed	1 A at 5V, 5136-SD-VME
	2 A at 5V, 5136-SD-VME2
	from pins 32 of rows A, B, and C of the P1 connector
Environmental	operating temperature 0-50 degrees Celsius
	storage temperature 0 -70 degrees Celsius
	Operating Humidity 5 to 95 % non-condensing
	storage humidity 0 to 95%
Card connector	Phoenix MSTB1.5/6ST-5.08
Cable	Belden 9463, twinaxial, 20 AWG

Acknowledgements

X-Link and PICS Simulation are registered trademarks of S-S Technologies Inc.

PLC is a registered trademark of Allen-Bradley.

IBM and Micro Channel are registered trademarks of International Business Machines.

All other trade names referenced are trademarks or registered trademarks of their respective companies.

Technical Support

Before you call for help ...

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

- Card type and serial number
- Computer make and model and hardware configuration (other cards installed)
- Operating system type and version
- Details of the problem you are experiencing; application module type and version, target network, circumstances that caused the problem.

Getting Help

Technical support is available during regular business hours (eastern standard time) or by fax or mail.

Technical Support

SST

50 Northland Road

Waterloo, Ontario N2V 1N3

Voice: (519) 725-5136

Fax: (519) 725-1515

email: techsupport@sstech.on.ca

website: sstech.on.ca

Software Updates

The current distribution software for the 5136-SD-VME is available from our website ate www.sstech.on.ca

Warranty

SST warrants all new products to be free of defects in material and workmanship when applied in the manner for which they were intended and according to SST's published information on proper installation. The Warranty period is one year from the date of shipment for all cards except the following which carry a 10 year warranty from date of purchase: 5136-SD, 5136-SD-104, 5136-SD-VME, 5136-SD-VME/2, 5136-DN, 5136-DN-PCM, 5136-DN-VME, 5136-DNP, 5136-PFB, 5136-PFB-104, 5136-PFB-PCI, 5136-PFB-VME and 5136-CN.

SST will repair or replace, at its option, all products returned to factory freight prepaid, which prove upon examination to be within the Warranty definitions and time period.

The Warranty does not cover costs of installation, removal or damage to user's property or any contingent expenses or consequential damages. Maximum liability of SST is the cost of the product(s).

Product Returns

If it should be necessary to return or exchange items, please contact SST for a Return Authorization number.

SST

50 Northland Road

Waterloo, Ontario N2V 1N3

Voice: (519) 725-5136

Fax: (519) 725-1515

—T—

—U—

—W—

—Z—