AD-A206 979

SCIENCE INC
HORIZONS

SeisMS.NORSAR

A SEISMIC MONITORING SYSTEM

FINAL REPORT

DECEMBER 1988

sponsored by

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (DoD)

NUCLEAR MONITORING RESEARCH OFFICE

(Support and Enhancement of Scientific Workstations for Seismological Research)

ARPA Order No. 4887

Issued by DSSW under Contract MDA903-85-C-0090

DTIC
ELECTE
17 APR 1989
S E

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Goverment.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

SCIENCE HORIZONS, INC. Encinitas, California

89
89 8 15 010

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) SH-R4-88-2112	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Science Horizons, Inc. — 6b OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Defense Supply Service-Washington
6c. ADDRESS (City, State, and ZIP Code) 710 Encinitas Blvd., Suite 200 Encinitas, CA 92024	7b. ADDRESS (City, State, and ZIP Code) Rm 1D245, The Pentagon Washington, D.C. 20310
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA — 8b. OFFICE SYMBOL (If applicable) NMRO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903-85-C-0090
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. — PROJECT NO. — TASK NO. — WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification)	
12. PERSONAL AUTHOR(S)	
13a. TYPE OF REPORT Final Technical — 13b. TIME COVERED FROM 3/1/85 TO 9/30/88 — 14. DATE OF REPORT (Year, Month, Day) 88 NOV 9 — 15. PAGE COUNT	
16. SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Seismic Monitoring System Communications Interface Module, Real-time Data Acquisition, Center for Seismic Studies Workstation, Data Base, Bureau of Mineral Resources, & NORSA

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Science Horizons has developed SeisMS.NORSAR, a seismic monitoring system that is part of a joint effort by Norway and the United States to relay real-time data streams generated from the ARCESS array located at Finmark (Karasjok) to the NORSAR acquisition site located at Kjeller, Norway. SeisMS.NORSAR consists of a communications hardware/software subsystem called a Communications Interface Module (CIM) and a near-real-time software subsystem for automated acquisition and processing of the seismic data. One CIM located at ARCESS receives the real-time array (hub) and high frequency (HF) data streams, multiplexes them and retransmits the stream by satallite to the NORSAR site. At NORSAR, a receiving CIM demultiplexes the data, buffers it if necessary, and relays the hub and HF data streams to a DCP board in a Sun Workstation. Data acquisition software running in the workstation, diverts each data stream onto a circular buffer disk loop, from which all or selected portions of the raw data can be reformatted to conform to a CSS data base and archived to tape or written to a CSS data base on disk. This report includes information on CIM

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Maj. Art W. Hurt	22b TELEPHONE (Include Area Code) — 22c OFFICE SYMBOL

reference, CIM maintenance, using the data acquisition software, and manual (man) pages for user commands and required input file formats. In addition to this report, Science Horizons has delivered under separate cover a tabular listing of the final disposition of the computer hardware that was purchased under this contract plus a ½-inch tape containing the source code and on-line documentation for the seismic monitoring systems developed by Science Horizons and delivered to the Bureau of Mineral Resources (Australia) and to NORSAR (Norway).

ABSTRACT

Science Horizons has developed SeisMS.NORSAR, a seismic monitoring system that is part of a joint effort by Norway and the United States to relay real-time data streams generated from the ARCESS array located at Finmark (Karasjok) to the NORSAR acquisition site located at Kjeller, Norway. SeisMS.NORSAR consists of a communictions hardware/software subsystem called a Communications Interface Module (CIM) and a near-real-time software subsystem for automated acquisition and processing of the seismic data. One CIM located at ARCESS receives the real-time array (hub) and high frequency (HF) data streams, multiplexes them and retransmits the stream by satallite to the NORSAR site. At NORSAR, a receiving CIM demultiplexes the data, buffers it if necessary, and relays the hub and HF data streams to a DCP board in a Sun workstation. Data acquisition software running in the workstation, diverts each data stream onto a circular buffer disk loop, from which all or selected portions of the raw data can be reformatted to conform to a CSS data base and archived to tape or written to a CSS data base on disk. This report includes information on CIM reference, CIM maintenance, using the data acquisition software, and manual (man) pages for user commands and required input file formats. In addition to this report, Science Horizons has delivered under separate cover a tabular listing of the final disposition of the computer hardware that was purchased under this contract plus a ½-inch tape containing the source code and on-line documentation for the seismic monitoring systems developed by Science horizons and delivered to the Bureau of Mineral Resources (Australia) and to NORSAR (Norway).

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

SCIENCE HORIZONS INC

SeisMS.NORSAR

A SEISMIC MONITORING SYSTEM

OCTOBER 1988

USER GUIDE

SCIENCE HORIZONS, INC. Encinitas, California

NOTICE

The information in this manual has been checked and is believed to be accurate and reliable. However, no responsibility is assumed by Science Horizons for its use or for any inaccuracies. Specifications are subject to change without notice. Science Horizons does not assume any liability arising out of use or other application of any product, circuit or program described herein. This document does not convey any license under Science Horizons' patents or the rights of others.

SUN-3 and SUN Workstation are trademarks of Sun Microsystems, Inc.

UNIX is a trademark of ATT Bell Labratories.

Copyright © 1988
Science Horizons, Inc.
Suite 200
710 Encinitas Boulevard
Encinitas, California 92024

TABLE OF

CONTENTS

FIGURES

TABLES

PREFACE

This manual is divided into six chapters that describes the Science Horizons participation in the seismic monitoring system currently installed in Norway. The installation is referenced as the NORSAR System in this manual.

This documentation includes an overview chapter of the NORSAR system, a chapter describing the acquisition software developed by Science Horizons plus the *man* page documentation of the user software programs, a CIM reference chapter, and a CIM maintenance chapter.

The OVERVIEW chapter introduces the NORSAR system, describes its major configurations and subsystems, and provides information that is generic to the system. Succeeding chapters present an in-depth look at the major subsystems.

The CIM chapter (Chapter 2) is a generic system reference that discusses the CIM capabilities, its communications and internal configuration.

Chapter 3 describes the acquisition software that processes the data streams and writes them to a Center data base for analysis.

Chapter 4 is intentionally omitted because it does not apply to this installation.

The fifth chapter includes the user level documentation *man* pages for the software.

Chapter 6 contains the *man* page documentation for the file that must be read into the system in order to convert the acquired data into a usable Center data base.

Chapter 7 provides CIM maintenance instructions and diagrams.

Many details of the hardware and software depend on site-specific requirements and vary from one installation to another. Therefore, the manual may include information on optional hardware or software not in use at your site; however, an effort has been made to tailor this document specifically to the Norway installation.

Details of the firmware and proprietary design of any of the system components and subsystems are beyond the scope of this manual.

CHAPTER 1

SeisMS.NORSAR OVERVIEW

October 1988

TABLE OF

CONTENTS

FIGURES

TABLE OF CONTENTS

CHAPTER 1

NORSAR OVERVIEW

1.1. Introduction

Science Horizons' participation in a joint research and development effort by Norway and the United States consists in establishment of the following items:

1. A *communications* hardware/software subsystem, called a Communications Interface Module (CIM), designed to capture the real-time seismic data stream at the array sites and transmit the data to the analysis sites.
2. A *near-real-time* software subsystem for automated acquisition and processing of the seismic data.

The completed project was to have provided high quality data in near-real-time from two array sites to two analysis sites, and the means at the anaylsis sites to automatically analyze and archive the data. The array sites comprise the NORESS array at Hamar (Løten) and the ARCESS array at Finmark (Karasjok). The analysis sites are NORSAR in Kjeller, Norway, and the Center for Seismic Studies in Rosslyn, Virginia, USA.

To date, Science Horizons has installed a CIM multiplexer at the ARCESS site and a CIM demultiplexer and acquisition software at the NORSAR analysis site. Figure 1-1 describes the ARCESS configuration at Finmark, where both high frequency (HF) and array (hub) data streams are initiated.

- At the ARCESS site, the two data streams are multiplexed by a CIM for satellite transmission to NORSAR.
- At NORSAR, a CIM receives and demultiplexes the data, and relays the array and HF data streams to a SUN Workstation for acquisition processing.

Figure 1-2 shows the communications configuration at the NORSAR acquisition site.

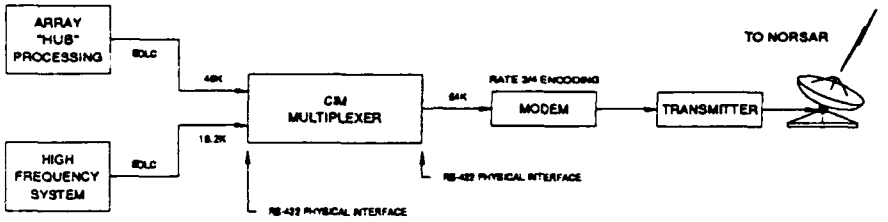


Figure 1-1. ARCESS communications configuration

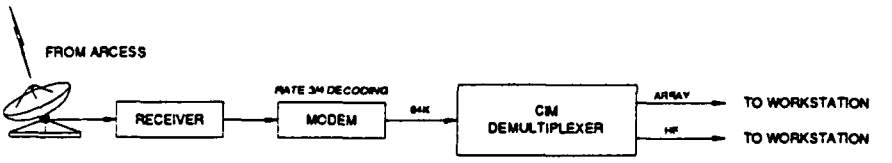


Figure 1-2. NORSAR communications configuration

1.2. CIM Communications Subsystem

The communications subsystem of the Seismic Monitoring System (SeisMS) has the capability to receive, buffer, multiplex/demultiplex and retransmit up to eight streams of seismic data in near-real time. Each installation is composed of at least one and sometimes two CIM units.

Instrumentation data is received by a CIM either already multiplexed or as separate streams of raw data. If the input data is multiplexed the receiving CIM may demultiplex the streams or just perform quality control, reformating the data if necessary for transmission, and buffer the data for relay to the workstation for acquisition processing. A multiplexing CIM may also transmit direct to the workstation or to another CIM for demultiplexing and/or buffering. The specific function and number of CIM units for an installation varies with the local requirements of the installation.

1.3. Seismic Acquisition Subsystem

Each instrumentation site for seismic monitoring may have a different number and type of seismometers, which creates different data management requirements. These site dependent specifications impact the number and type of data streams to be processed, the speed of the transmission, the length of a data frame, and the configuration of the hardware required to process the streams. These variables, in turn, determine the requirements and operating procedures at the analysis site.

At NORSAR, the seismic acquisition software creates a circular buffer disk loop for each data stream from ARCESS, writes the data to its circular buffer, then formats and stores the data in a Center data base for access and analysis. Once the process is begun, it proceeds automatically.

1.4. Applicable Documents

- Center for Seismic Studies Database Structure, Version 2.8 Technical Report C87-04; Mary Ann Brennan; Center for Seismic Studies, Arlington, VA.

CHAPTER 2

COMMUNICATIONS INTERFACE MODULE

TABLE OF

CONTENTS

CHAPTER 2 – COMMUNICATIONS INTERFACE MODULE

FIGURES

TABLES

CHAPTER 2

COMMUNICATIONS INTERFACE MODULE

2.1. Introduction

The Communications Interface Module (CIM) is a hardware subsystem that captures real-time seismic data streams at an instrumentation site, buffers and retransmits the streams to an analysis site (data center). In any installation, a CIM may also perform multiplexing and/or demultiplexing of the data streams. The CIM, also called the *White Box*, is an automated, low-maintenance, microprocessor-based module, designed for unattended operation. This chapter describes the full features of a CIM as it is configured for multiplexing or demultiplexing and buffering. Figure 2-1 shows the exterior of a CIM.

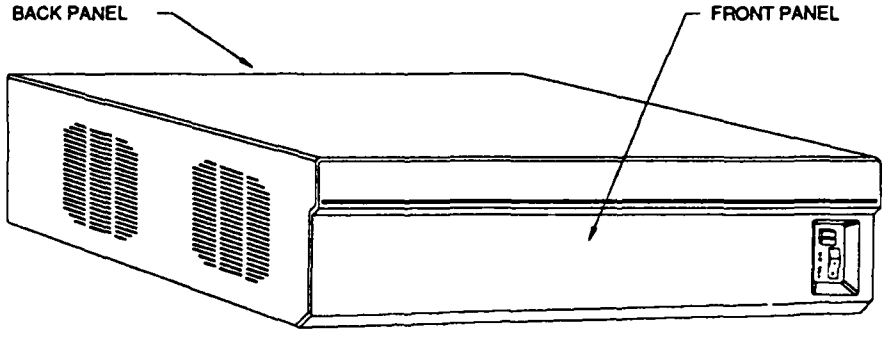


Figure 2-1. Exterior of CIM cabinet

2.2. Features

The basic CIM unit is contained in a cabinet measuring 7-inches high, 19-inches wide, and 19.75-inches deep. The unit includes a 7-slot cage to house the microprocessor and memory boards. The actual number and configuration of these boards is function and site dependent. The CIM chassis provides the following features:

- Front panel mounted power switch
- Lighted front panel mounted reset switch
- Up to eight rear panel DB25S serial communications connectors
- 325W multiple output switching power supply
- UL/VDE/CSA approved line filter
- 115 or 230 VAC operation
- Dual 78 cfm cooling fans
- VME 7-slot card cage
- Optional rack mount slides and extended front panel

Figures 2-2 and 2-3, respectively, show details of the CIM front and rear panels. Figure 2-4 shows a CIM configured with a workstation, and a disk/tape subsystem as freestanding units; while Figure 2-5 shows a rack configuration that encloses a CIM, a disk drive and a ½-inch tape drive into one unit.

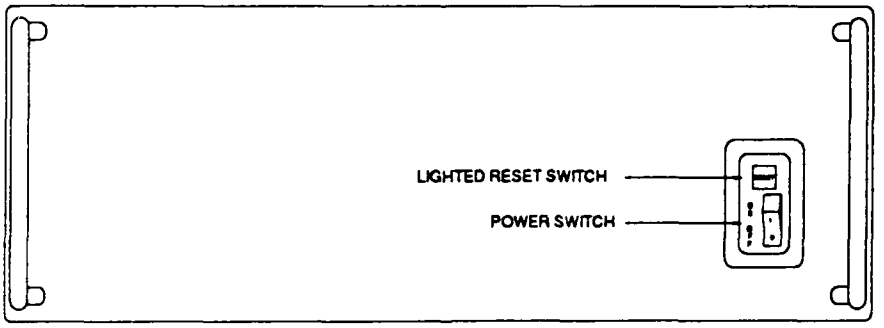


Figure 2-2. Front panel of CIM cabinet

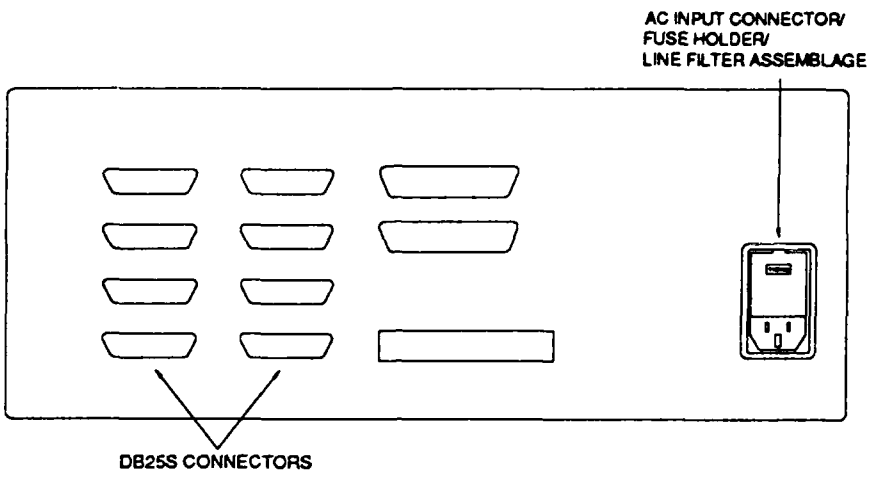


Figure 2-3. Rear panel of CIM cabinet

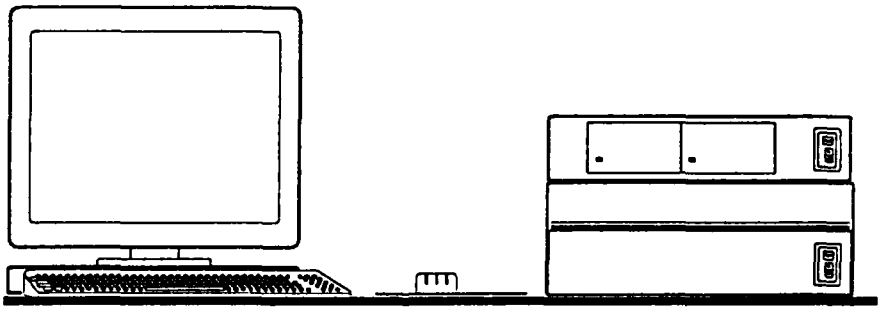


Figure 2-4. CIM, workstation, disk/tape configuration

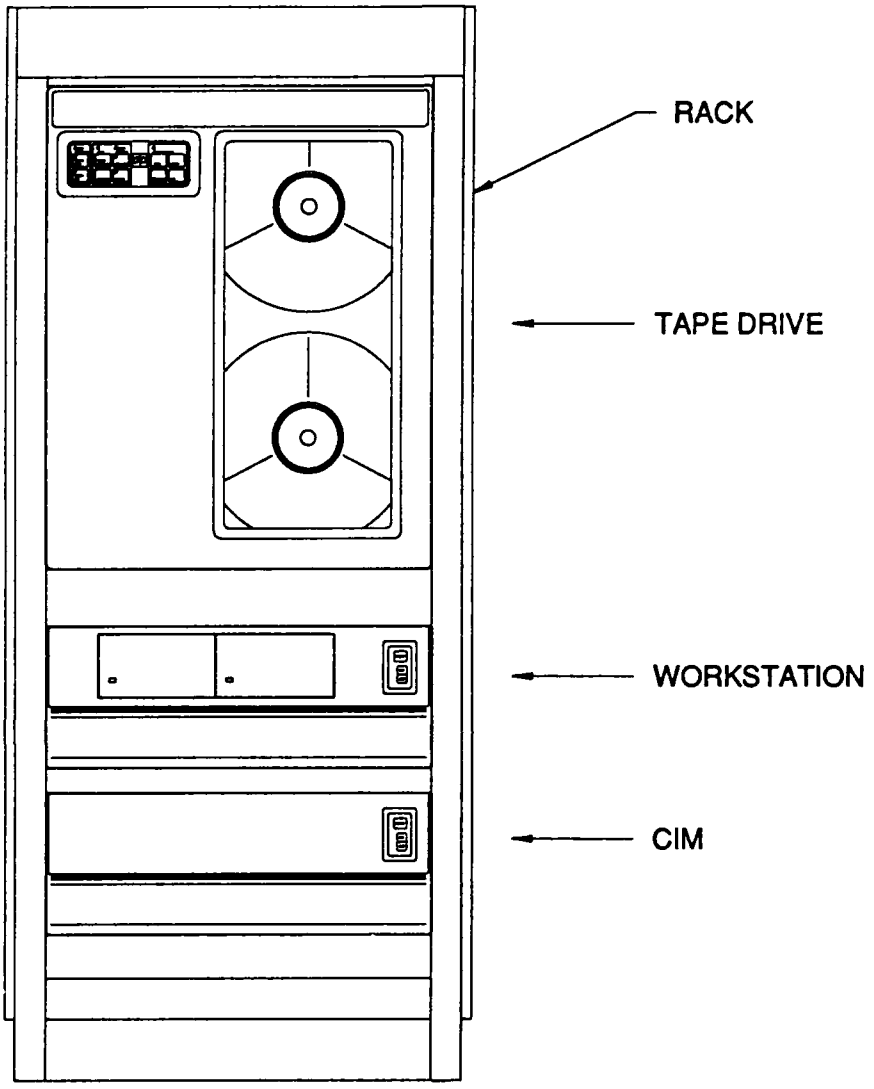


Figure 2-5. CIM, disk, tape rack configuration

2.3. Specifications

With an empty card cage, a CIM weighs 38 lbs. Installed weight depends on the configuration.

2.3.1. Environment

Intake air temperature must be within 0° to 50° centigrade and intake humidity must be within 20% to 90% non-condensing.

Ample air flow for cooling the power supply and card cage is provided by the two 78 cfm fans. Be certain that unobstructed air paths exist for both the inlet to and exhaust from the chassis.

2.3.2. Power Requirements

For sites using 115 VAC power, the chassis should be fitted with a 5 amp fuse. For sites using 220-240 VAC, a 2.5 amp fuse is required. Actual power consumption is configuration dependent and will range from approximately 80 to 150 watts.

2.3.3. External Connectors

The CIM has eight positions for DB25S serial communications connectors, located on the back panel of the unit. In most configurations, all but one are data ports that employ a modified RS-422/449 interface for high-speed serial communication using the Synchronous Data Link Control (SDLC) protocol.

The remaining port uses a modified RS-232 interface for low speed asynchronous communication with a dumb terminal. An ordinary 3-wire interface cable can connect a CRT terminal to this port for self-test and fault isolation.

If circumstances require, a CIM can be configured with all ports using a low-speed RS-232 interface or with all ports configured for RS-422. However, this latter configuration sacrifices the diagnostic capabilities.

2.3.4. Compatibility

The CIM is designed to interact with a variety of equipment to meet special site requirements.

A CIM demultiplexer can be hard-wired for high speed data transmission to an SH-3E or a SUN workstation for data acquisition applications. A pair of modem control signals in the interface between the CIM and the workstation can be used to control the flow of data. This allows for convenient workstation system maintenance without loss of data.

2.4. Architecture

The composition of a CIM depends on its function within a given system. A CIM can be configured for:

- Multiplexing
- Multiplexing and buffering
- Demultiplexing and buffering
- Buffering

The actual configuration for a given installation depends on the number and speed of data streams being handled. Each microprocessor board has two input/output (I/O ports). A CIM can be built with 1-, 2-, 3-, or 4-board configurations, which provide two, four, six or eight I/O ports. One, two or three buffer memory boards (4Mbyte each) can be used.

A typical two-processor unit can handle two incoming and one outgoing data streams. For extremely high data rates, it is recommended that one coprocessor board per data stream be used.

Usually, a CIM configured as a multiplexer or demultiplexer, is composed of at least two 68010 CPU boards with two multi-protocol serial I/O channels, and at least one 4MB memory board. On-board baud-rate clock generation is available on each port. To adjust to modems with baud rates of 16000, 32000, 64000, etc., an external crystal oscillator can be added to a multiplexer. This provides a more accurate, stable crystal oscillator timebase for satellite modem applications.

A demultiplexer contains sufficient slave memory to buffer enough data to cover the short periods of maintenance down-time required for the receiving workstation. Therefore, the demultiplexer commonly contains two additional 4MB memory cards for a total of 12MB of buffer memory.

A CIM used only for buffering is composed of one CPU board, two interface cards, and sufficient memory to buffer data during down times.

A basic one-unit CIM configuration that multiplexes and buffers two data streams includes the following components:

- Two microprocessor cards that include:
  - Motorola 68010 CPU
  - Motorola 68450 4 channel DMAC
  - 1MB read-write memory
  - 128K read-only memory
  - A VMEbus interface
  - Zilog Z8530 SCC with 2 serial I/O ports
  - 4 user LEDs under software control, 3 system LEDs under hardware control

  - 8 user-definable switches

- Two interface cards for RS-232 to RS-422/449 conversion
- One memory card that includes:
  - 4MB memory
  - A VMEbus
  - 3 system LEDs under hardware control
  - Starting address selectable on 64K boundaries

A facility using higher data rates and/or more data streams. can use additional microprocessor boards (up to a maximum of four) in each unit to service additional data streams. Figure 2-6 is a block diagram of the CIM architecture.

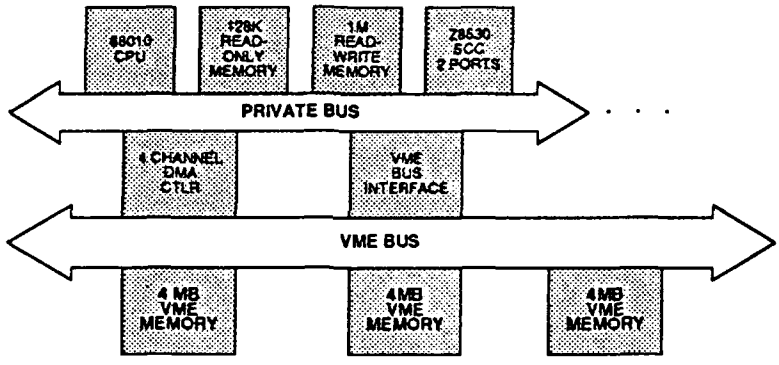


Figure 2-6. CIM architecture block diagram.

2.4.1. Port/Channel Configuration

Places for up to eight DB25S serial communications connectors or *ports* are provided on a CIM. The ports are located on the back panel of the unit, and are referred to as A ports and B ports. Figure 2-7 shows the two column arrangement, with A ports on the left and B ports on the right.

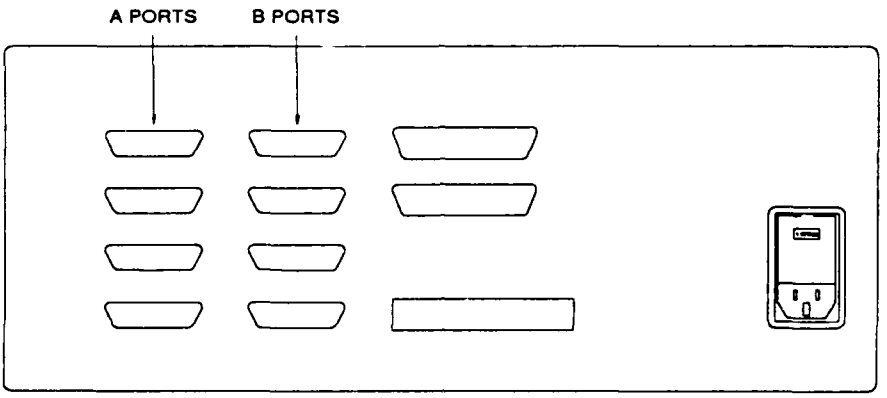


Figure 2-7. A and B ports on back panel.

The number and function of active ports depends on the site-specific use of the CIM. One port can be configured for diagnostics and wired for data communications equipment (DCE), using RS-232C signal levels and a slightly modified connector pin out.

The remaining active ports are configured for modified RS-449 data terminating equipment (DTE), using RS-422 signaling levels and DB25S connectors.

Figure 2-8 shows the CIM rear panel wired for three data ports and a test port. Table 2-1 lists the DCE (test-port) pin assignments, and Table 2-2 lists the DTE (data-port) pin assignments. Figure 2-9 shows how the CIM pinouts correspond to the RS-449 standard. Only those signals listed to the left of the DB25S connector in the figure are provided by the interface; all other pins are left open. Signal directions, as indicated by the arrows, are oriented as though the gray stripes were ribbon cable connecting the on-board ports to the CIM rear panel connectors. Signals received by the CIM are shown by left-pointing arrows, those emanating from it are shown by right-pointing arrows.

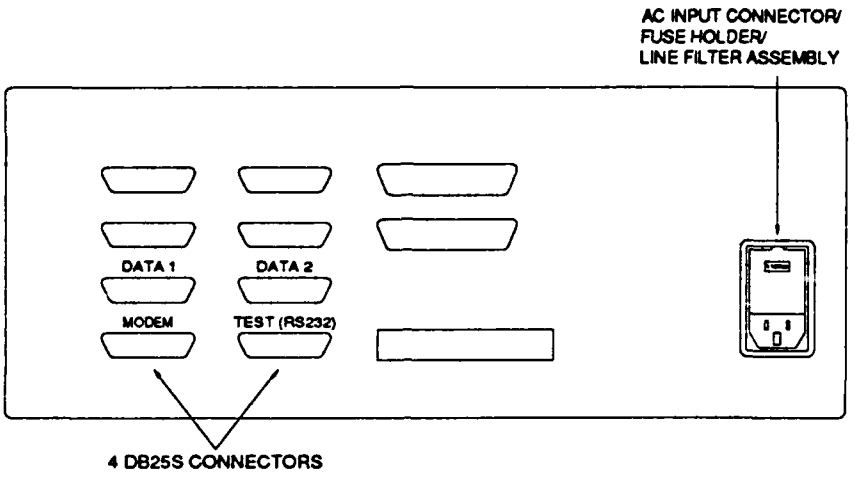


Figure 2-8. Data, modem and test ports.

Table 2-1 DCE with Modified RS-232 Interface

Pin	Name	Direction	Description
2	TxD	DCE < DTE	Transmitted data
3	RxD	DCE > DTE	Received data
4	RTS	DCE < DTE	Request to send
5	CTS	DCE > DTE	Clear to send
6	DSR	DCE > DTE	Data set ready
7	SG	-	Signal Ground
8	DCD	DCE < DTE*	Carrier detect
15	TxC	DCE < DTE*	Transmit clock (DCE)
17	RxC	DCE < DTE*	Receive Clock
20	DTR	DCE < DTE	Data terminal ready
22	RI	DCE < DTE	Ring indicator

* Signal direction reversed from standard.

Table 2-2. DTE with Modified RS-449 Interface

Pin	Name	Direction	Description
1	SG	-	Signal Ground
3	SG	-	Signal Ground
4	SD –	DTE > DCE	Send Data –
5	ST –	DTE > DCE*	Send Timing –
6	RD –	DTE < DCE	Receive Data –
7	RS –	DTE > DCE	Ready to Send –
8	RT –	DTE < DCE	Receive Timing –
9	CS –	DTE < DCE	Clear to Send –
12	TR –	DTE > DCE	Terminal Ready –
13	RR –	DTE < DCE	Receive Ready –
14	SG	-	Signal Ground
15	SG	-	Signal Ground
16	SD +	DTE > DCE	Send Data +
17	ST +	DTE > DCE*	Send Timing +
18	RD +	DTE < DCE	Receive Data +
19	RS +	DTE > DCE	Ready to Send +
20	RT +	DTE < DCE	Receive Timing +
21	CS +	DTE < DCE	Clear to Send +
24	TR +	DTE > DCE	Terminal Ready +
25	RR +	DTE < DCE	Receive Ready +

* Signal direction reversed from standard.

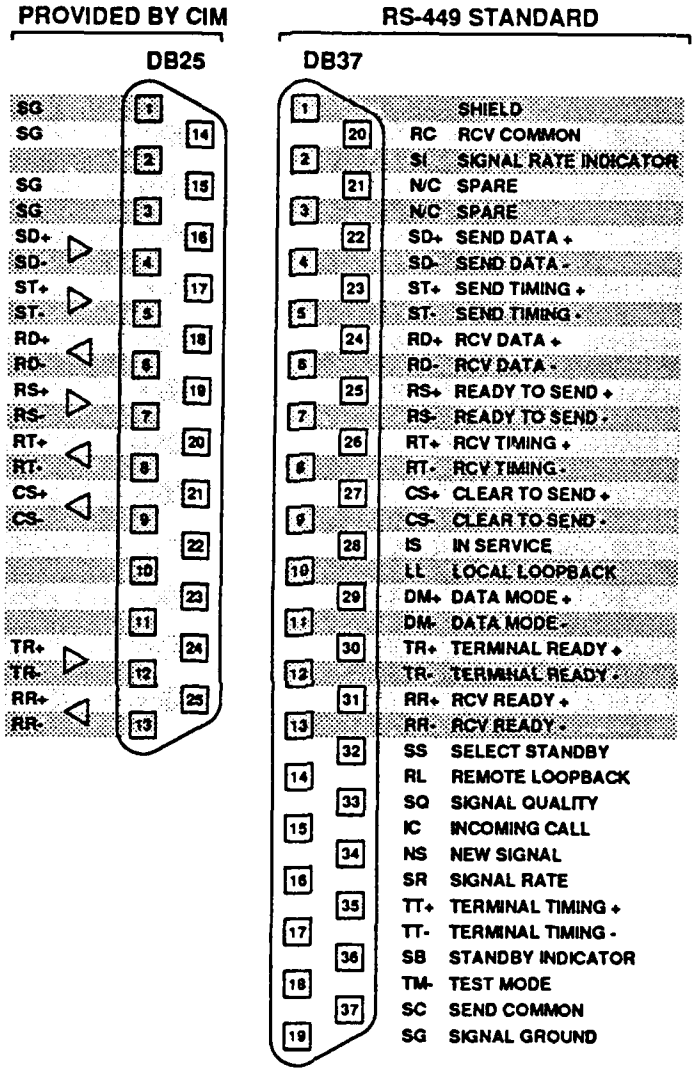


Figure 2-9. Modified RS-442/449

2.4.1.1. RS-232C and RS-449 Interfaces

As previously noted, the microprocessor boards come equipped with an RS-232C interface. The other ports use a modified RS-422/RS-449 interface. However, for equipment requiring low data rates, the CIM can be configured for all RS232C ports.

The serial I/O ports are converted from RS-232C to RS-422 signal levels by replacing the RS-232C interface chips on the CPU board with jumpers that allow raw TTL signals to be sent off board.

These signals are first sent to an RS-422 conversion card mounted inside the CIM chassis, and then relayed to the ports on the back panel.

Figure 2-10 shows the location of the on-board RS-232C interface chips. These chips process I/O signals to ports A and B as follows:

- The MC1488 and MC1489 pair closest to the edge of the board are, respectively, the output and input interfaces for port A (positions U5 and U2).
- The MC1488 and MC1489 pair towards the center of the board are the output and input interfaces for port B (positions U6 and U4).
- The interface chip in position U3, the central MC1489 chip, carries input signals for both ports.

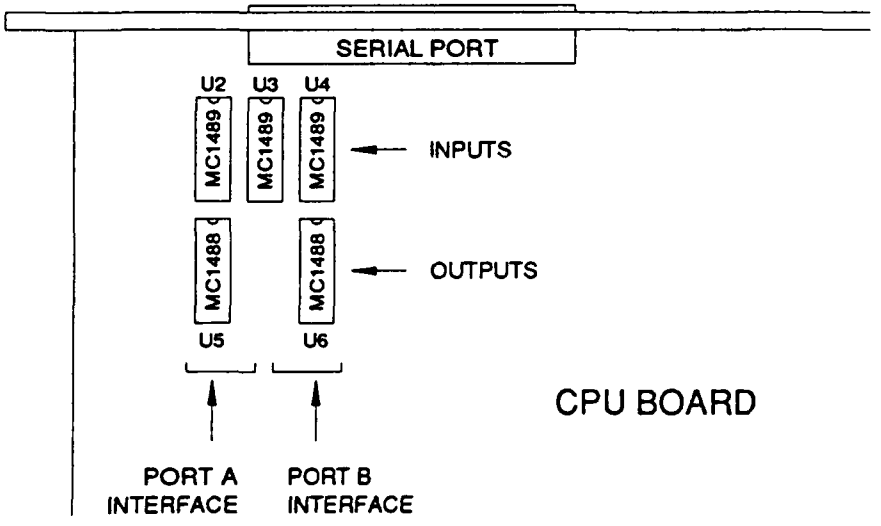


Figure 2-10. Location of RS-232C interface chips.

Table 2-3 shows the conversion jumper configuration.

Table 2-3. RS-232 Interface Socket Jumper Configuration

MC1488 Outputs (U5, U6)			MC1489 Inputs (U2, U4)		
1	to	3	1	to	3
4	to	6	4	to	6
8	to	10	8	to	10
11	to	13	11	to	13

2.4.1.2. Channel Configuration

Each of the microprocessor boards in a CIM unit has two programmable full-duplex serial I/O channels available. These channels are also designated ports A and B, to correspond to the A and B connectors on the back panel of the unit. For a CIM wired for three RS-449 data ports and one RS-232 test port (previous figure Figure 2-8) the channels are configured as follows:

- The three data ports (ports A and B on one board, and port A only on the second board) are configured for high-speed synchronous serial communications using the SDLC low-level frame protocol and RS-422 signal levels.
- The remaining port, port B, is configured for low speed (9600 baud) asynchronous communications suitable for use with a dumb terminal. This port is used for self-test and diagnostic purposes to help in isolating faults.

Figure 2-ll is a schematic illustration of the CPU board with RS-232C signal levels on just one serial port. Figure 2-12 shows the board with all the chips removed and both ports converted to RS-422/449 operation.

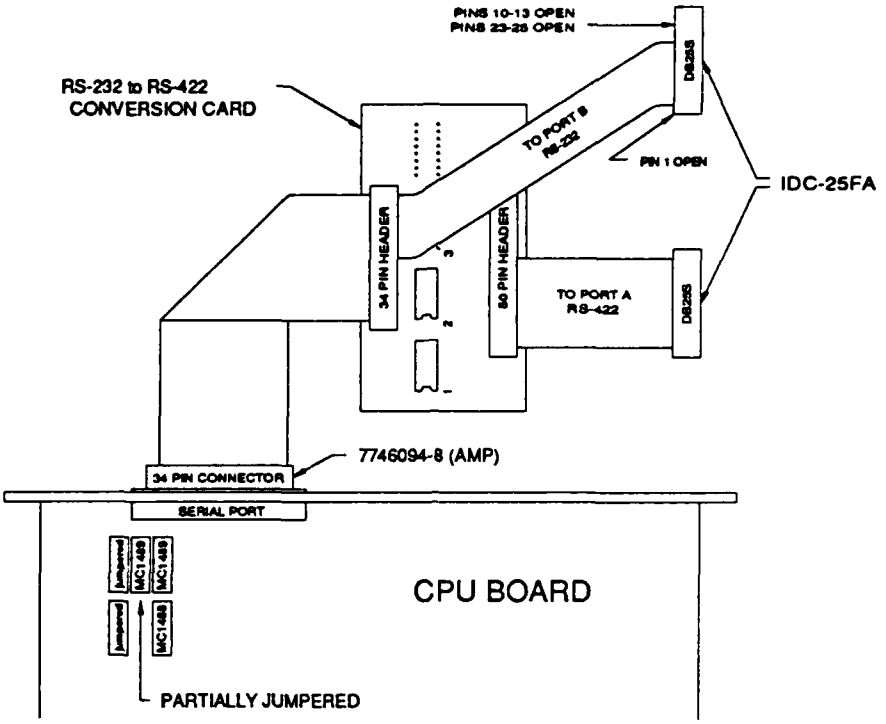


Figure 2-11. One-port conversion schmatic

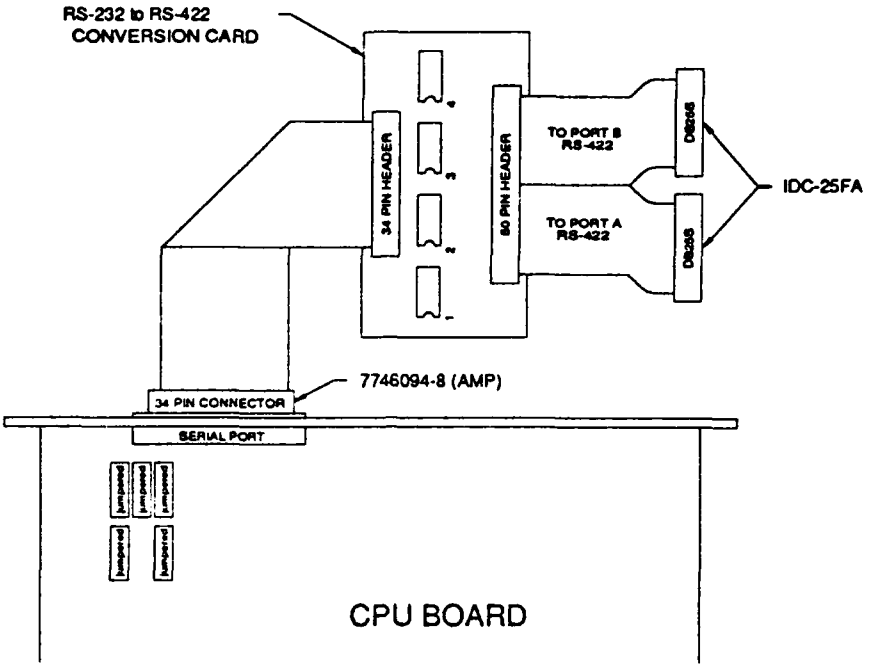


Figure 2-12. Two-port conversion schematic

2.4.2. Crystal Oscillator Circuit

For installations requiring more accurate modem clocking that operate at baud rates of 16Kbaud, 32Kbaud, 64Kbaud, etc., a baud rate generator (BRG) consisting of a crystal oscillator and logic chip strapped to produce a Baud Rate Clock of the required rate, can be included in the unit.

In configurations using both a multiplexing and demultiplexing CIM, one conversion card has four interface chips to convert signals on both ports to RS-422 levels. The other conversion card has only two interface chips to convert just port A to RS-422 levels. The multiplexer takes advantage of this configuration by using the unoccupied space for the crystal oscillator and logic chip that provide the Baud Rate Generator (BRG) for the modem clock. Figure 2-13 shows the oscillator location on the conversion card.

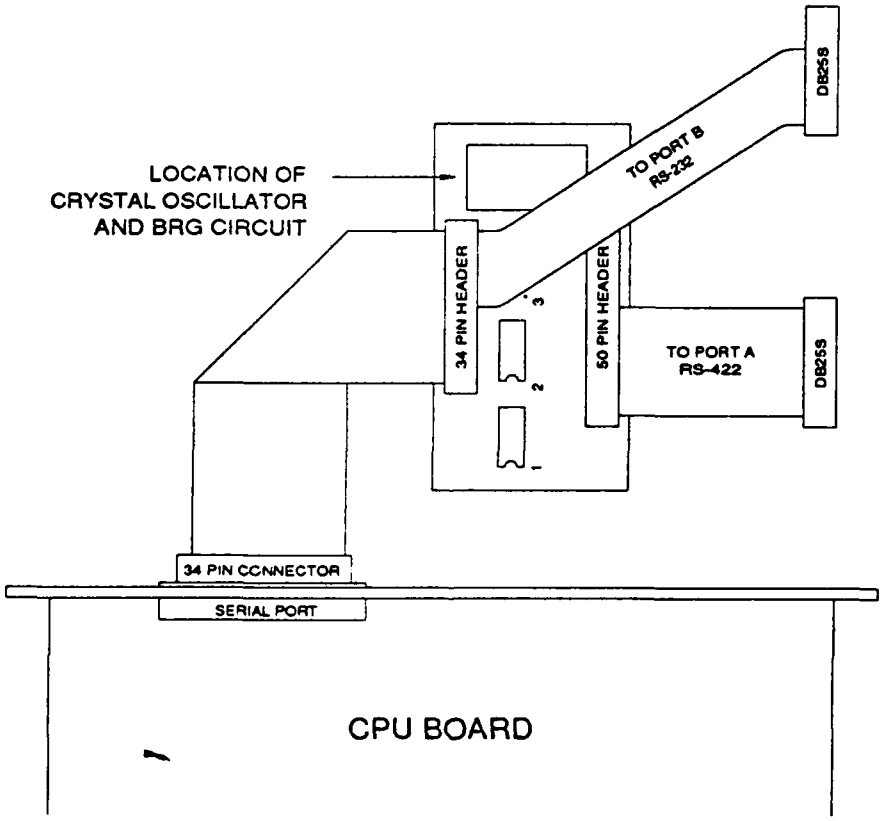


Figure 2-13. Oscillator location.

2.4.3. Dip Switch Settings

Each CPU board contains an eight position dip switch that is used for firmware configuration. The setting of these switches determines the behavior of the board, including the:

- Firmware tasks to execute
- Data stream/port asignments
- Baud rate clock settings
- Board function: master or slave
- Port function: reading or writing
- Clock function: send or receive
- Handshake function: generate or receive
- Diagnostic mode: on or off

The port/channel edge of each factory installed CPU board is labeled with a 7-digit binary number. That number determines the proper setting for the seven least significant dip switches on the board. The least significant digit (LSD) on the label corresponds to the least significant switch (switch 1). The eighth dip switch is a diagnostic toggle that should be in the off (*open*) position during normal operation. Figure 2-14 shows the location of the dip switches on the CPU board, and Figure 2-15 illustrates the digit/dip switch correspondence.

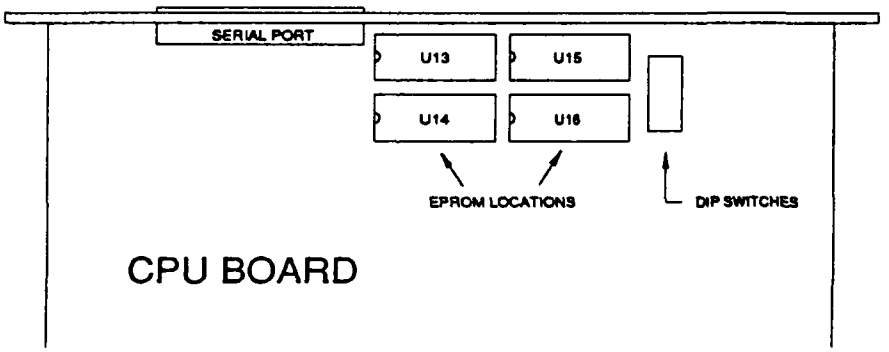


Figure 2-14. Dip switch panel location

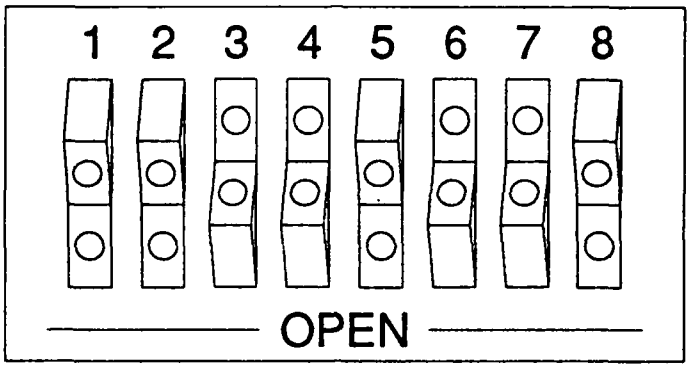


Figure 2-15. DIP switch settings

2.4.4. Master Board Controller Jumpers

Normally in a multi-board system, a VMEbus System Controller card provides the system bus clock and participates in the arbitration logic. However in the CIM, only one level of bus arbitration is used, and a separate system controller card is not required. Instead, the system controller functions are provided by the sysclock and self arbitration jumpers. In this situation, bus arbitration level three is the only level supported. Figure 4-16 shows the location on the CPU board of these jumpers.

It is important to note that these jumpers are installed *only* when the board is the bottom board in the card cage; all other CPU cards should have these jumpers removed. All other jumpers on the CPU cards are pre-set and should not be changed.

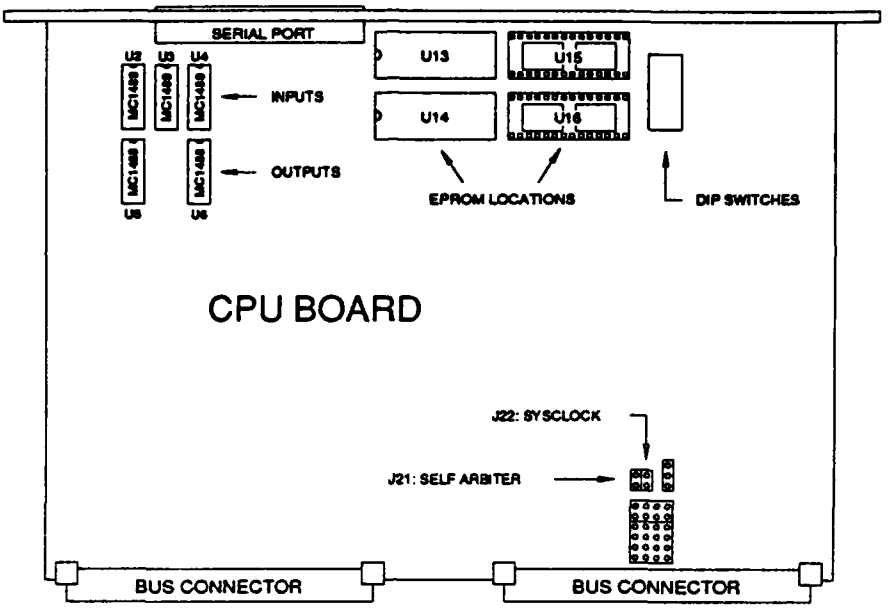


Figure 2-16. Master Board Jumper Location

2.4.5. Memory Board Configuration

Each CIM memory board provides 4MB of Random Access Memory (RAM), which is used to buffer incoming data streams for multiplexing and for output to the workstation. The boards are identical, except for the VMEbus address assignment.

The starting address in memory and the Control and Status Register (CSR) addressing mode are set by jumpers E01 through E23, located as shown in Figure 2-17. Each memory address jumper indicates a certain size of memory from 64KB to 2GB. Used in combination, these jumpers permit addressing the memory on any 64KB boundary in 4MB increments. Notice that on the board, the jumpers are in right to left order.

The CSR is used only by the memory board processor to record runtime errors affecting its assigned memory area. Therefore, even though the CIM does not access this register, it is important not to have several boards with the same address.

Like the rest of the board, the memory and CSR jumpers are all pre-set for installation and should not be modified. However, understanding the address jumper configuration is useful for identifying a board's slot in the CIM card cage and is also necessary to properly configure a replacement board. Table 2-4 lists the memory address associated with each of the memory address jumpers E01 through E16.

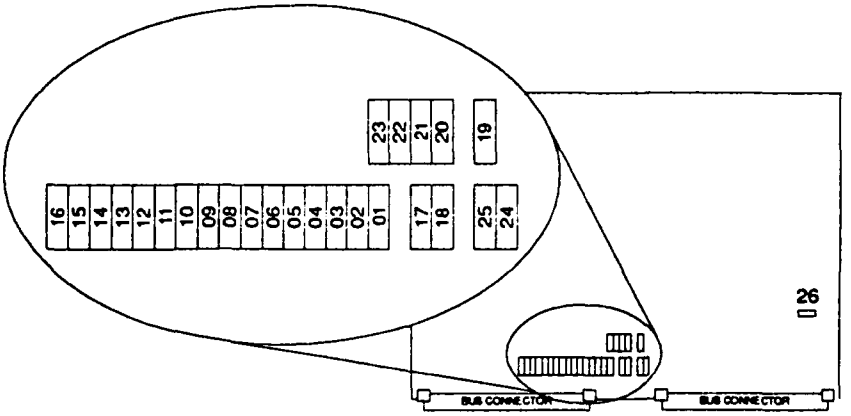


Figure 2-17. VMERAM Jumper Location

Because the jumpers are used in combination, all 16 jumpers must be set. The starting address of the 4MB increments of memory and corresponding jumper up/down settings are listed in Table 2-5. The settings are shown in the same right to left order as the jumpers appear on the board.

The addresses reserved for the CSR are the last 16 even addresses in the memory map. Jumpers E20-E23 are used in combination to configure these 16 CSR addresses. On the first memory board (bottom of the memory board stack), the CSR is set to the lowest of the 16 possible addresses (0xFEE0), the next board is set to 0xFEE2, etc. The CSR addresses for additional boards increase sequentially up to 0xFEFE. The CSR jumpers (like the address jumpers) are also on the board in right to left (E23-E20 order. Table 2-6 shows the 16 CSR address/jumper configurations.

All the other jumpers, E17-E19 and E24-E26 are pre-set according to the settings in Table 2-7. Jumpers E27-E29 are factory set and should not be changed.

Table 2-4. VMERAM Address Jumpers

Jumper No.	Value		
E01	2^{16}	=	64KB
E02	2^{17}	=	128KB
E03	2^{18}	=	256KB
E04	2^{19}	=	512KB
E05	2^{20}	=	1MB
E06	2^{21}	=	2MB
E07	2^{22}	=	4MB
E08	2^{23}	=	8MB
E09	2^{24}	=	16MB
E10	2^{25}	=	32MB
E11	2^{26}	=	64MB
E12	2^{27}	=	128MB
E13	2^{28}	=	256MB
E14	2^{29}	=	512MB
E15	2^{30}	=	1GB
E16	2^{31}	=	2GB

Table 2-5. VMERAM Address/Jumper Configuration

Starting Address	Jumper Settings E16->E01	Slot No.
0x100000 (1MB)	ddddddddddddudddd	5
0x500000 (5MB)	dddddddddududdddd	6
0x900000 (9MB)	ddddddddudduddddd	7

Where:

d = down

u = up

Table 2-6 Memory CSR Address Jumper Configurations

Absolute Address	Short I/O Address	Jumper E23	E22	E21	E20
(FF)FFFEE0	FEE0	down	down	down	down
(FF)FFFEE2	FEE2	up	down	down	down
(FF)FFFEE4	FEE4	down	up	down	down
(FF)FFFEE6	FEE6	up	up	down	down
(FF)FFFEE8	FEE8	down	down	up	down
(FF)FFFEEA	FEEA	up	down	up	down
(FF)FFFEEC	FEEC	down	up	up	down
(FF)FFFEEE	FEEE	up	up	up	down
(FF)FFFEF0	FEFE0	down	down	down	up
(FF)FFFEF2	FEFE2	up	down	down	up
(FF)FFFEF4	FEFE4	down	up	down	up
(FF)FFFEF6	FEFE6	up	up	down	up
(FF)FFFEF8	FEFE8	down	down	up	up
(FF)FFFEFA	FEFEA	up	down	up	up
(FF)FFFEFC	FEFEC	down	up	up	up
(FF)FFFEFE	FEFEE	up	up	up	up

Table 2-7. VMERAM Pre-set Jumpers

Jumper	Setting	Function
E17	down	Supervisor Only Mode off
E18	up	24-bit addressing on
E19	up	CSR short I/O addressing on
E24	up	Battery powered backup off
E25	up	same as E24
E26	left	VMEbus ACFAIL line off

2.4.6. Backplane Configuration

In the event that the technician is unsure from which slot in the card cage a faulty board was removed, the following is a discussion on how to configure the backplane of the card cage.

In the CIM enclosure backplane there are four Bus Grant daisy-chained signal lines, BG3IN/OUT through BG0IN/OUT (P1 pins B4 through B11), and one Interrupt Acknowledge daisy-chained signal line, IACKIN/IACKOUT (P1 pins A21 & A22). Except for IACKIN at slot one, which is connected to IACK (pin A20) of every other slot, the IACKOUT (pin A22) of each slot is connected to the IACKIN (pin A21) of the next higher-numbered slot.

In order for the IACK signal to pass through slots containing no plug-in cards and slots containing cards unable to drive the IACKOUT signal line, jumpers must be installed. It is vital that jumpers *not* be installed across slots containing plug-in cards that drive the IACKOUT signal line. Placing jumpers across these slots disables the interrupt arbitration system and could damage board logic components. In a similar fashion, the Bus Grant jumpers should *not* be installed across slots containing plug-in cards that are potential VMEbus masters.

The backplane of the enclosure is configured with jumper blocks at each of the first six card-cage slots to facilitate jumpering of the IACK and Bus Grant signals. All six of the jumper blocks, labeled A01 IACK through A06 IACK and A01 BUS GRANT through A06 BUS GRANT are depicted in Figure 2-18. In the figure, jumpered slots are depicted by small rectangles representing the jumpers, while unjumpered slots are depicted by pairs of dots representing pins. The illustration shows a configuration in which cards that are potential VMEbus masters are installed only in slots A01 and A02, and therefore, only the jumpers labeled A01 BUS GRANT, A01 IACK and A02 BUS GRANT and A02 IACK are removed. However, if processor cards reside in slots 1 and 3 for example, jumpers for 1 and 3 should be removed.

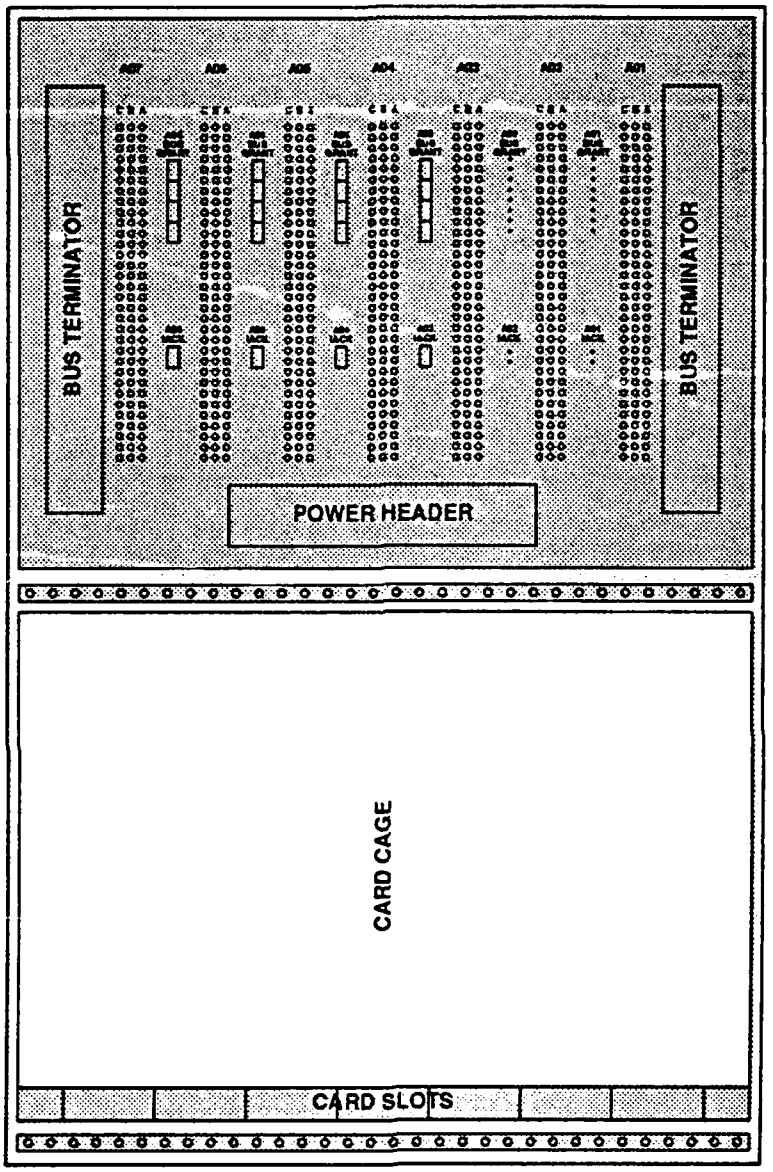


Figure 2-18. Backplane with Jumpers

2.5. Condition Reporting

Each microprocessor and memory board is equipped with a group or groups of light emitting diodes (LEDs) that monitor and signal the state of the module, the system, the software and the individual board. Coded illumination patterns of the LEDs provide information for diagnosing system faults. Specific conditions are indicated by the illumination of specific displays:

On memory boards:

- A green LED means initialization is complete and operation is normal.
- A yellow LED means a single bit error has occurred since last system reset. These errors are self-correcting and do not cause a system interrupt.
- A red LED means a double bit or multiple bit error has occurred since the last system reset. These errors generate bus errors and system interrupt.

On processor boards:

- There are two groups of LEDs on the back edge labeled SMF and 1234.
  - The SMF LEDS (Slave, Master and Fail) indicate the manner in which the board is interacting with the VMEbus (slave mode, master mode or neither), and monitors the system.
  - The 1234 set is referred to as user LEDs and are used to monitor the health of the CIM module.
- The S LED, if lighted, indicates a serious error in the CIM because the microprocessor boards never operate in slave mode. If the condition is not cleared by a reset, it is likely that a system component on another CPU board has failed.
- The M LED glows only on a board that is currently the VMEbus master. This happens any time a board accesses VMEbus memory.
- The F LED is a system fail indicator that monitors the state of the SYSFAIL bus signal. It will extinguish after system initialization unless some on-board component has failed.
- The 1234 LEDs can show three different illumination modes.
  - Walking mode, where each LED in a group blinks on and off in turn, means operating conditions are normal.
  - Fixed mode, where the group shows a steady pattern, is caused by system related errors. The specific pattern is a code that indicates the type of error.
  - Flashing mode, where the group shows a flashing pattern, is caused by software related errors. The specific pattern is a code that indicates the type of error.

2.5.1. System Errors

When the 1234 LEDs glow in a steady fixed pattern, they are signalling a special condition exception. On occurrence, processor activity is suspended, and the pattern remains on the 1234 LED display until the system is reset. Table 2-8 lists the error codes and corresponding LED pattern and meaning. Errors 0x5, 0x6, and 0x7 are firmware errors that are not likely to ever occur in the field.

Table 2-8. System Errors - Fixed Display

Error Code	Steady Pattern	Error
0x1	● ○ ○ ○	bus error
0x2	○ ● ○ ○	privilege violation
0x3	● ● ○ ○	illegal instruction
0x4	○ ○ ● ○	illegal address
0x5	● ○ ● ○	divide by zero
0x6	○ ● ● ○	bounds check
0x7	● ● ● ○	two's complement overflow
0x8	○ ○ ○ ●	parity error
0xf	● ● ● ●	purge in progress

● = on ○ = off

2.5.1.1. Bus Error

Often a temporary condition that will clear after a system reset. Other causes:

- Master board does not exist – nonresponse to an attempt to access VME bus memory, or double bit bus errors.
- Memory board is misaddressed.
- Replacemment board put in wrong slot in card cage.

2.5.1.2. Privilege Violation

Unlikely error, but occurrence should clear with a system reset. If not, contact Science Horizons for analysis. Possible cause:

- Attempt by code not running in supervisor mode to execute a privileged instruction.

2.5.1.3. Illegal Instruction

Similar to a privilege error and should clear with a reset. Possible causes:

- Processor *wild jump*
- On board memory failure (usually produces a memory parity error instead).

2.5.1.4. Illegal Address

Signals an attempt to perform a word access on an odd address. Possible causes are similar to those for privilege violation and illegal instruction exceptions.

2.5.1.5. Divide by Zero

CIM performs very few divide operations, so error is not likely.

2.5.1.6. Bounds Check

Exception is taken when special bounds checking code is used. No such code is incorporated in the CIM design.

2.5.1.7. Two's Complement Overflow

Initiated by the trapv instruction, which is not used in the CIM firmware.

2.5.1.8. Parity Error

Indication of byte parity error during a memory read cycle. If occurrence is frequent, the CPU on-board memory or associated circuitry has probably failed.

2.5.1.9. Purge in Progress

This pattern is an exception to the previously stated rule. All four LEDs come on briefly to indicate a slave memory data purge is in progress.

A purge occurs if data has been allowed to back up in the allotted buffers, causing a situation in which there is no space for incoming data. During the multiplexing and demultiplexing process, data frames are stored in slave memory on linked lists, each list associated with a given data stream. When the need arises, data is purged from the list that occupies the most memory. During the purge, some number of the oldest data frames are popped off the offending list, and the deallocated buffers are made available for incoming data. The number of frames that are freed is large enough to prevent the need for frequent purges and may be different for different installations.

2.5.2. Software Errors

When the 1234 LEDs present a flashing pattern, there is an error in the CIM firmware configuration. On occurrence, processing is suspended and the pattern is left on the LED display until the system is reset (except in the case of read completion errors). Table 2-9 lists the software error codes and corresponding LED pattern and meaning. The only two of these that are likely to occur in the field are 0x1 and 0x8.

Table 2-9. Software Errors - Flashing Display

Error Code	Flashing Pattern				Error
0x1	●	○	○	○	illegal configuration
0x3	●	●	○	○	read error
0x4	○	○	●	○	error opening port
0x5	●	○	●	○	error creating task
0x6	○	●	●	○	error completing write
0x7	●	●	●	○	write error
0x8	○	○	○	●	slave memory bounds error

● = flashing ○ = off

2.5.2.1. Illegal Configuration

Refers to the CPU board and will occur during system start up after a reset. Causes:

- Incorrect setting of dip switch indicating a configuration lookup table entry that does not exist.
- Memory board placed in wrong slot.
- Memory board starting address is incorrect.

2.5.2.2. Read Error

Either the specified channel is not open for reading or an illegal channel (*i.e.*, neither Port A or Port B) was requested. This error can be caused only by a firmware malfunction.

2.5.2.3. Error Opening Port

Either an illegal channel was requested, or the requested channel is already in use. This error can be caused only by a firmware malfunction.

2.5.2.4. Error Creating Task

Unlikely error because CIM firmware is designed to prevent it. Possible causes:

- There are no task control blocks available for the task create function.
- Task id number has already been assigned.

2.5.2.5. Error Completing Write

Frame ended prematurely or was otherwise prevented from completing.

2.5.2.6. Write Error

Either a specified channel is not open for writing or an illegal channel was requested. This error can be caused only by a firmware malfunction.

2.5.2.7. Slave Memory Bounds Error

Slave memory is not properly sized and no record exists of its upper bound. Cause:

- Improperly set dip switches.
- Master board cannot find slave memory
- Memory board missing or misaddressed.

2.5.3. Error Completing Read

This error *does not* suspend processing; therefore, the LED pattern for it is temporary. These LEDs are a diagnostic tool that is lit only when frames containing bit errors are entering the CIM. When the frame check log in the acquisition software shows an inordinate amount of dropped data frames, pull the back of the CIM and check for one of these patterns. If present, you have found the problem.

Affected Port	Temporary Pattern				Error
A	●	●	○	○	error completing read
B	○	○	●	●	error completing read

2.6. Communications

Communications processing is performed primarily by the hardware in the Communications Interface Module and by a communications microprocessor board in the workstation.

2.6.1. Protocols

The software to perform communications tasks is implemented to follow approximately the ISO-DP7498 layered model shown below in Figure 2-19.

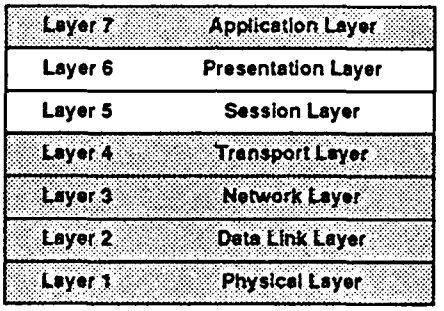


Figure 2-19. The ISO Communications Protocol Model

The shaded layers in the figure are those for which corresponding modules or processes (tasks) may exist in the Science Horizons implementation. The communications processor board includes a mini-operating system in read-only memory. Many of the layers are actually implemented as tasks that communicate with one another through a queue management system. Installations using simplex communications do not implement layers 3 and 4. The following sections outline the functions that are performed in the various layers of the protocol mechanism.

2.6.1.1. Layer 1 – The Physical Layer

The physical layer provides services to connect, maintain and disconnect the physical circuits that form the communications mechanism. It is at this layer that the traditional interface between data terminal equipment (DTE) and data communications equipment (DCE) is made. Physical layer protocols define the electrical and mechanical standards and the functional and procedural requirements for interconnecting the components that form the physical medium.

The communications board and associated cables and connectors form the physical layer in the Workstation and the input ports for the physical layer in the CIM. The devices involved have been configured for operation under the EIA RS-232C or EIA RS-422/RS-449 standards.

2.6.1.2. Layer 2 – The Data Link Layer

The data link layer provides services related to the interchange of data across a link established by the physical layer. Link layer protocols manage the establishment, maintenance and release of data link connections. Data link control protocols such as BSC, SDLC, HDLC and some portions of ADCCP reside in this layer.

Two tasks in the CIM occupy this layer. One handles raw packet transfers and the other establishes and terminates a raw connection between stations. CIM applications are typically *hardwired* to the communications equipment and therefore no facility is provided for establishing connections.

2.6.1.3. Layer 3 – The Network Layer

The network layer deals primarily with the movement of data through a network. It is utilized only by installations having full duplex communications. The services provided at this layer comprise routing, switching, data link concatenation, sequencing, flow control and some error recovery functions.

The CIM software occupying this layer does not conform to the above definition. It deals with managing data frame buffers so that the time critical buffer requirements of the data link layer are met. In particular, the data link layer software requires a new frame buffer address to be available whenever a read operation completes in order to ensure that no data is lost.

2.6.1.4. Layer 4 – The Transport Layer

This layer is a very important one for full duplex communications, whether the network is a simple point-to-point connection or a multidrop arrangement of stations. It provides a transparent data transfer mechanism for the layers above. Reliable transfers over imperfect networks are implemented here. Thus sequence and control field interpretation, message queuing and acknowledgement, error detection and frame retransmission, duplicate packet suppression and multiplexing are services provided in layer 4.

The software at this layer manages all frame checking and retransmission mechanisms used to form a reliable connection between CIM units across a communications the network. In particular, this layer maintains the queue of

transmitted packets and matches acknowledgements it receives from the remote station with messages that have arrived there and frame-checked successfully. It responds to rejection messages and retransmits frames as needed. Most CIM applications do not implement this layer.

2.6.1.5. Layer 5 – The Session Layer

The session layer binds cooperating processes into a temporary relationship for purposes of exchanging data messages. The services provided are generally related to establishing and terminating data streams between stations.

No session layer software exists in the current implementation.

2.6.1.6. Layer 6 – The Presentation Layer

This layer, when present, is used to adapt the information handling characteristics of one applications process to those of another. Services provided include data transformation, formatting, syntax, encryption and decryption.

No presentation layer software exists in the current implementation.

2.6.1.7. Layer 7 – The Application Layer

This layer is actually an interface between applications and the services provided by the other layers making up the protocol handling package.

At the receiving end of the system, a host interface module occupies this layer. This module performs interface functions to allow the workstation to use the layers below.

2.6.2. Task Management

Communication between elements in the CIM is accomplished through the manipulation of double linked lists of memory buffers. On each processor board there are three such lists:

- One list associated with Port A
- One list associated with Port B
- One free list for use by the memory allocation scheme

These on-board lists are referred to as local lists. In slave memory, there may be as many as nine such lists:

- One free list to support slave memory allocation
- One list for each of the eight possible data streams.

These lists are manipulated by four types of tasks:

- Tasks that receive data
- Tasks that copy data out of a local list into slave memory (copyout)
- Tasks that copy data out of slave memory into a local list (copyin)

- Tasks that transmit data

The names of the tasks reflect which ports and linked lists are minipulated. This allows the flow of data through the CIM to be specified and tracked. The names of the two receive and two transmit tasks are based on which port is used. Table 2-10 lists the task names, the port and the local list that each uses.

Table 2-10. Data Processing Tasks

Task Name	Port Used	Local List Used
receive0	port 0 (Port A)	1
receive1	port 1 (Port B)	2
transmit0	port 0 (Port A)	1
transmit1	port 1 (Port B)	2

Note: Local list 0 is always the free list.

The conventions used for the names of the copy tasks are based on which linked lists the tasks manipulate. For example, the task *copyout21* copies data from the local list 2 to slave list 1, *copyout1s* copies data from local list 1 to one of several lists in slave memory, and *copyin42* copies data from slave list 4 to local list 2.

Copyout tasks in the multiplexer prepend an additional address and control field to each frame that indicates the port on which the frame was received. The copyout tasks in the demultiplexer use this information to copy data to the unique list in slave memory associated with that stream, stripping the identifying numbers in the process.

These identifying numbers are an integral part of the board configurations, as are selecting which tasks will run, the baud rate for each port, various parameters associated with opening each port, etc. Therefore, it is important to select the appropriate configuration for the specific task and the specific data stream and hardware configuration in use. For example, certain board configurations are designed to work with specific other configurations to process specific data streams. Figure 2-20 describes these logical groups for an installation with two CIM units and three microprocessor boards per unit.

Multiplexer Configuration Examples

Configuration 000000:

Datastream1 (DS1) data to Port A on board 3.	Port A opened for reading at 32000 baud; data copied to slave list 1 with an id of 0x11.
	Port B unused.

Configuration 000001:

Datastream1 (DS2) data to Port A on board 2.	Port A opened for reading at 19200 baud; data copied to slave list 1 with id of 0x12.
	Port B unused.

Configuration 000010:

Slave list 1 data to Local list 1 transmitted from Port A on board 1.	Port A opened for writing at 48000 baud. Port B unused.

Demultiplexer Configuration Examples

Configuration 000011

DS1 and DS2 data (ids 0x11 and 0x12) to Port A on board 1.	Port A opened for reading at 48000 baud; 0x11 data copied to slave list 1 and 0x12 data copied to slave list 2.
	Port B unused.

Configuration 000100:

Slave list 1 data (array) to Local list 1 transmitted from Port A on board 2.	Port A opened for writing to DS1 analysis workstation at 32000 baud (handshake accepted).
	Port B unused.

Configuration 000101

Slave list 2 data (DS2) to Local list 1 transmitted from Port A on board 3.	Port A opened for writing to DS2 analysis workstation at 19200 baud (handshake accepted).
	Port B unused.

Figure 2-20. Unit Configuration Example

2.6.3. Communications Processing

The Communications Interface Module is a point-to-point subsystem, where one multiplexing unit controls the communications lines to one or more demultiplexing units. The SDLC-based implementation does not assume that a reverse path, which would permit communication between receiving and transmitting sites, is provided (this would be the case, for instance, if a simplex satellite link were used as the communications medium). Therefore, the address and control fields of the SDLC frames are used to facilitate the multiplexing and demultiplexing of the data frames for different data streams (see the frame format information at the end of this section).

The multiplexing process is simple. Data frames from each source are received by the CIM on individual input ports serviced by asynchronous controlling tasks. At completion of a frame, each task adds a data source identifier by constructing new address and control bytes at the beginning of the frame. The new address byte is a unique 8-bit number assigned to the receiving port at system configuration time; the new control field is a dummy byte of all zeroes. The tasks then add the frames, including the additional identifying information, to a common list of frames available for transmission.

Meanwhile, another independently running task uses a third I/O port to transmit the frames as they become available. The resulting composite stream consists of the original frames, each with the added two-byte identification information, interleaved in the order they are received.

A demultiplexing unit uses the identifying information in the address and control fields to sort the data frames for the different sources of data, as follows:

1. A receiving task acquires the composite stream from the communications link and appends each new frame to a small input pool.
2. As frames become available, another task sorts and splits this pool into two or more (up to 8) separate lists according to the identification information provided in the address and control fields.
   - In order to make the end-to-end multiplexing/demultiplexing mechanism transparent, the additional address and control fields are stripped from the frame during this phase of the process.
3. Two or more transmitting tasks relay the frames from these lists to appropriate receivers (workstations).

2.6.3.1. Data Buffering

A demultiplexing unit or single multiplexing unit has functionality beyond the simple sorting just described. Because the data is ultimately acquired by UNIX-based workstation equipment, provisions to allow system maintenance without loss of data are required. Typical system maintenance may involve taking the workstation system off-line for some period of time and possibly rebooting it (on a large system this can take as much as 30 minutes).

To facilitate this type of operation, the CIM must buffer data while system changeover is accomplished. When the workstation again comes on-line

and is prepared to receive data, any backlog accumulated in the CIM must be dispensed rapidly to the workstation to allow room for incoming data. A buffering CIM performs these tasks as follows:

1. Allows transmission bandwidth on these channels sufficient to dispose of accumulated backlogs; the hard-wired interface between the CIM and the workstation equipment uses a high bit-rate clock, typically 76800 baud, more than double the required bit rate.
2. Enables the workstation to notify the CIM that it must temporarily stop or suspend the flow of data; a *handshake* mechanism uses a pair of modem control signals on the serial interface. The signals go false any time a system failure is detected in the workstation.
3. Resumes the flow of data automatically when a system becomes available and the acquisition software in the workstation signals its readiness to begin.

2.6.4. SDLC Frame Format

There are four fields in a basic SDLC data packet or frame, as follows:

- Address field - 8-bits
- Control field - 8-bits
- Information field - variable length
- Frame check field - 16-bits

The address, control and frame check fields, collectively, are sometimes called the *envelope*. These are fixed length fields, as indicated above. In some implementations the control field may be 16-bits long, but in the CIM subsystem, it is always 8-bits. The information field may be whatever length the data requires. Figure 2-21 shows the structure of a frame, including the delimiting flag characters.

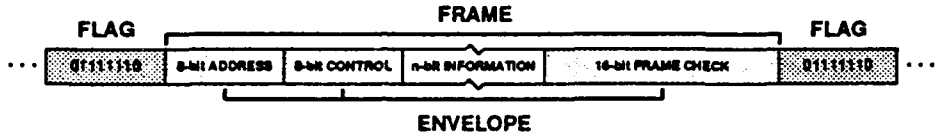


Figure 2-21. SDLC frame format

An 8-bit sequence known as a flag character delimits a frame and occupies idle time between frames. A flag character consists of an initial zero, six ones, and a final zero (01111110). In SDLC protocol, at least one interframe flag character must appear between any two frames. Therefore, even though a flag may mark the beginning of a frame, there is no such thing as a *beginning flag*, because the trailing flag character of one frame may be coincident with the leading flag of the following frame.

In order for the receiving interface to distinguish flag characters unambiguously from flag-like bit sequences within the frame, the flag-like portions of the frame are modified by a mechanism called *zero insertion*. During transmission, all frame bits, from the beginning of the address field through the end of the frame check field, are shifted out through zero insertion logic. This logic detects any occurrence of five contiguous ones (11111) in the stream and automatically inserts a zero (0) bit after each such sequence. The zero insertion logic is disabled during the transmission of an actual flag character.

At the receiving interface, this process is reversed. As bits are shifted into the receive shift register, similar detection logic is employed to notice sequences of five ones (11111). If the sixth bit is a zero (0), it is automatically removed from the stream. If the sixth bit is a one (1), the sequence is recognized as a frame delimiting flag or as an abort sequence and the frame is ended.

If an abort sequence is received, it ends a transmitted frame prematurely. This sequence consists of seven or more ones (1111111) in a row, and is not preceded by a frame check sequence. The transmitting interface is programmed to completely and automatically retransmit any frame it aborts, and the receiving interface is programmed to discard any frame terminated by an abort sequence.

2.7. Installation and Maintenance

In general, CIM units are installed at the site of operation. However, specific information on system take-down and start-up procedures, board replacement and configuration, and instructions for using the Science Horizons CIM Spare Parts Kit are contained in the MAINTENANCE module of this manual.

CHAPTER 2

APPENDIX A

2A.1. NORSAR Cabling

Figures 2A-1, 2A-2, and 2A-3 show the cable wiring from the CIM to the converter, from the CIM to the DCP board in the workstation, and from the converter to the modem, respectively.

CABLE DIAGRAM: CIM TO V.35 CONVERTER

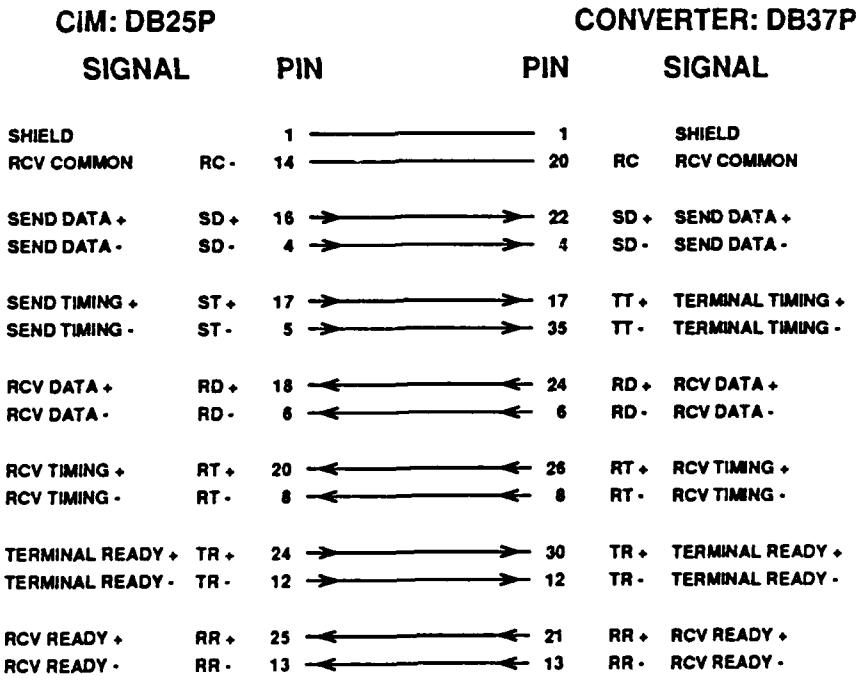


Figure 2A-1. Cable wiring from CIM to converter

CABLE DIAGRAM: CIM TO DCP

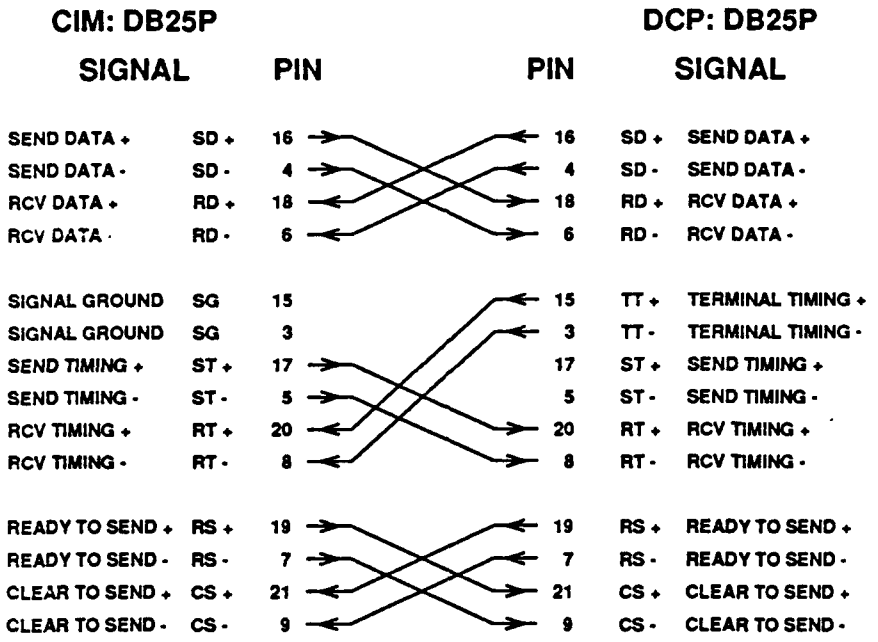


Figure 2A-2. Cable wiring from CIM to DCP

CABLE DIAGRAM: MODEM TO CONVERTER

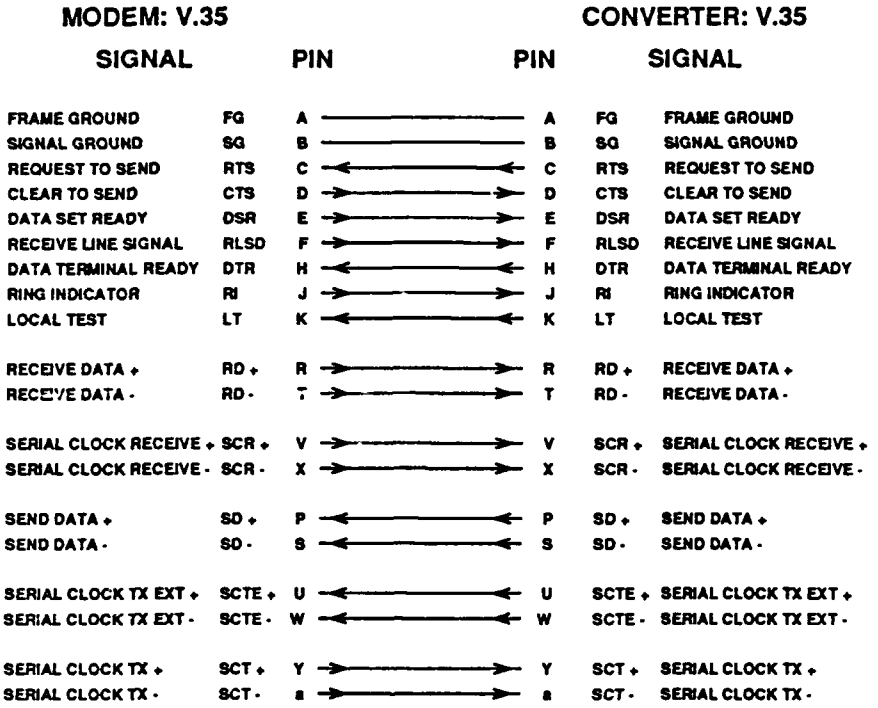


Figure 2A-3. Cable wiring from converter to modem

CHAPTER 3

SEISMIC ACQUISITION SOFTWARE

October 1988

TABLE OF

CONTENTS

FIGURES

TABLE OF CONTENTS

CHAPTER 3

NORSAR SEISMIC ACQUISITION SOFTWARE

3.1. Introduction

At the NORSAR installation, a multiplexed data stream is received and demultiplexed by a CIM into array (hub) and high frequency (HF) data. These two raw data streams are received by different ports on a special microprocessor (DCP) board in a SUN Workstation. A handshake mechanism between the CIM and the workstation signals the CIM to buffer incoming data whenever the SUN is not operational or a disk loop needs to be purged.

From the DCP, each stream of raw data undergoes a rudimentary quality control check to be certain its frames are in order and the data is valid. Then the frames are written to a specially created circular buffer disk loop. When the circular buffer is full, new data overwrites the old in an endless stream. Once on the disk loop, selected portions of the data can be formatted and written to a Center data base for analysis. Figure 3-1 illustrates the flow of data from the receiving satellite to the workstation, and Figure 3-2 illustrates the data acquisition processes in the workstation.

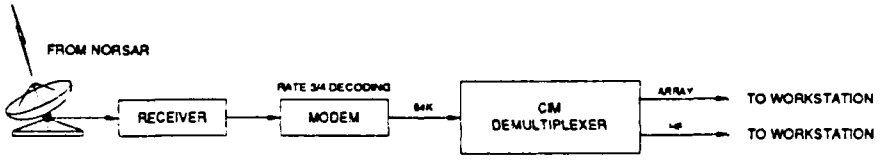


Figure 3-1. NORSAR acquisition data flow

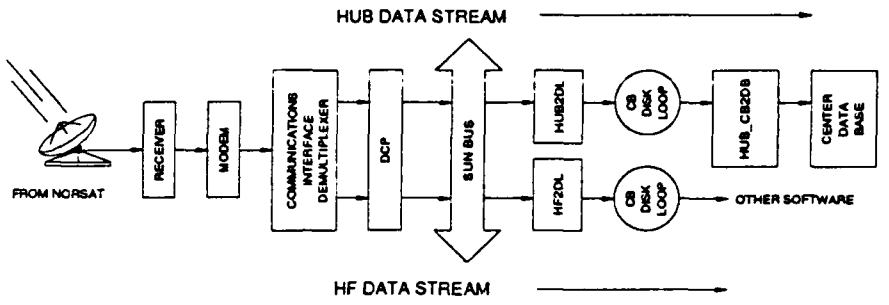


Figure 3-2. NORSAR acquisition processes

3.1.1. Disk Loop

A disk loop is a circular data store whose size is limited only by the total network-wide disk capacity and whose records must all be the same size. A disk loop appears to be a single *virtual* mega-file whose size can be much larger than any single one of the UNIX file systems that are mounted as part of the Network File System (NFS) mechanism. A dl is created by running the disk loop creation utility, **dltouch**. This utility:

- Reads an operator-supplied disk loop configuration file
- Creates all of the files associated with the disk loop,
- Fills these files with appropriate NULL values,
- Creates and writes a dl descriptor file for referencing the disk loop.

All the procedures that record the record numbers and byte offsets and that open and close the dl data files are transparent. The disk loop has three major functional characteristics:

Data Buffering

Allows for manipulation of a single very large data store in the same manner as a single file is manipulated using the low level UNIX I/O procedures. This provides a facility for buffering large amounts of data in a common network-wide data pool that can be accessed by a number of different processes.

Circular Nature

Has no *end-of-information* boundary and instead is a circular data store. Although there is a disk loop start point, it is only needed as a reference point for specifying record positions within the disk loop. An application that performs continuous disk loop writing operations does not extend the size of the loop or encounter an *end-of-information* condition. Once the dl is completely filled, new disk loop write operations overwrite old data from a previous pass. The disk loop anchor file tracks the position of the disk loop write pointer, and this information prevents reading processes from attempting to read past the write pointer.

Similarly, reading applications never encounter *end-of-file* conditions, and a reading request that reaches the linear end of the disk loop automatically wraps back to the beginning.

Network Access

The disk loop works across a local area network if the file systems on which the disk loop is stored physically are accessed via NFS, so that the various processes that access a disk loop need not all run on the same machine.

3.1.2. Circular Buffers

A circular buffer disk loop is a circular data store for buffering a data stream between the data acquisition processing that writes into the buffer, and the processes that read from the buffer. Circular buffers can be used for real-time or near-real-time applications because they provide *elasticity* in the data flow between the potentially rigid real-time requirements of data acquisition and the more flexible requirements of other processing.

Write/Read Coordination

Circular buffers provide automated coordination between the writing process and the various reading processes. Because of the endless nature of a disk loop, new data written into a cb normally overwrites the oldest data. Therefore the circular buffers are designed to prevent reading processes from trying to read through the moving write pointer. This is implemented with a blocking-read mechanism that waits to read future time-tagged records.

Allowable Read Zone

In order to properly coordinate simultaneous writing and reading processes, circular buffers define a portion of the disk loop as being *safe* for reading. This safe portion is called the Allowable Read Zone (ARZ). The ARZ is maintained by a disk loop write procedure in its in-core disk loop anchor buffer. This information is communicated to the reading processes via the disk loop anchor file.

Time Tagging

Circular buffers make it possible to place absolute, user-defined *tags* on each record. This allows the data to be accessed on time-based increments instead of record numbers. Time tag requirements specify that each tag must be unique and must monotonically increase with time. Therefore, record time tags are absolute time values expressed as double precision floating point numbers. Use of these tags allows a read process to request data in the future and block until that data is available.

Anchor Buffer

The anchor buffer contains a list of disk loop record number/tag value pairs. It always contains at least two entries, the head and tail. The head has the largest (newest) tag value, and the tail has the smallest.

3.1.3. Disk Loop Creation

All disk loops are created by the same program, **dltouch**. The one requirement that all disk loops have in common is that *all records on any individual disk loop must be the same size.* The way the **dltouch** program responds to the differences is through specifications in a *dl.input* file that can reside in the directory housing the associated files for the particular kind of disk loop being created, or the specifications can be entered from the terminal.

If the input information is incorporated into a *dl_input* file, the command line can redirect the pathname to the *dl_input* file as standard input when **dltouch** is invoked to create a disk loop:

dltouch *arg*<*dl_inputpathnm*

A *dl_input* file specifies the size of the disk loop, which should be based on the size and speed of the incoming data frames. When **dl_touch** completes, it creates a disk loop descriptor file that all subsequent programs use to manipulate that particular disk loop.

Specific format for the input information is described in the dltouch(1) man page in the next chapter. The NORESS installation requires the creation of one circular buffer disk loop for hub data and a different circular buffer disk loop for the HF data. Therefore, two different dl_input files or sets of information must exist for the two different disk loops.

3.1.4. Disk Loop Writing

As soon as a circular buffer disk loop has been created, the program **hub2dl** must be run to begin the acquisition of the hub data, and **hf2dl** must be run to begin acquisition of the HF data. The man page documentation for **hub2dl** is included in the next chapter.

Once the circular buffer disk loops have been created, the same disk loops can be used for repetitive runs of the hub and HF writing programs as long as the time stamps laid out on the respective cb disk loops do not appear to go back in time.

If there is reason to believe the time stamps on new data precede those of data already on the loop, the loop must be purged before the new data is written, or the apparent previous time frames will be thrown away.

3.1.5. Purging Disk Loops

Under normal circumstances, it is not necessary to recreate a disk loop unless it has to be resized. Whenever a disk loop is purged, the data on it is lost. However, if the writing programs (**hub2dl** and **hf2dl**) are suspended, the incoming data stream will be buffered in the CIM. This allows the data already on the disk loops to be stored before the loop is purged by running the program **dlrm**.

3.1.6. Circular Buffer to Data Base

Once the data is being written to the circular buffers, it must be read to be used. The program **hub_cb2db** reads the hub data in its circular buffer, reformats it, and saves it in a data base on disk. Once the data resides in a data base, it can be archived to tape (**arctwo**) or accessed and analyzed.

CHAPTER 5

USER COMMANDS

CHAPTER 5

NORSAR USER COMMANDS

5.1. Introduction

The following pages contain the detailed documentation for the Seismic Monitoring System section (1) USER COMMANDS that are relevent to the NORSAR system.

NAME

intro – introduction to user commands

SYNOPSIS

The *commands* section describes Science Horizons user commands that are relevent to the software documented in this manual. The following conventions are used:

Boldface words are considered literals to be entered exactly as shown.

Square brackets [] surround an optional argument. When an argument is giver as name, it always refers to file name.

Ellipses (...) show that the previous argument prototype can be repeated.

Braces {} surround a list of arguments or parameters where one must be selected.

Arguments may appear in any order.

Any command line argument that begins with a minus sign (–) is by definition a flag. Therefore negative numbers and file names that start with a hyphen should not be used as parameters.

In arguments, file and prefix names that start with a slash / are assumed to be a complete pathname and those that begin with a dot (.) are pathnames relative to the current working directory; otherwise, the name is taken to be a relative pathname from the current working directory or other specified directory.

DESCRIPTION

All command pages are based on a common format, but not all subsections always appear. The *synopsis* subsection summarizes the command-line syntax for the program being described, and the *description* subsection explains the program in detail.

ARGUMENTS

This subsection lists and briefly describes all mandatory arguments of a command.

OPTIONS

This subsection lists and briefly describes all optional arguments of a command.

FILES

The *files* subsection gives the names of files that are built into the program.

NOTES

This subsection is for special explanations or requirements for running the program.

SEE ALSO

This subsection references other relevent programs and documents.

DIAGNOSTICS

The *diagnostics* subsection explains diagnostic messages that may be generated (self-explanatory messages are not discussed).

BUGS

This subsection asknowledges known bugs and/or deficiencies. Sometimes fixes are also described.

NAME

arctwo – write waveforms on archive tapes

SYNOPSIS

arctwo [**–vAW**] **–o** outdev **–t** tape **–d** date **–s** time **–l** length [**–f** file] [**–c** chanlist] [**–w** trimlist]

DESCRIPTION

Program **arctwo** reads from a **wfdisc** file and writes the waveform data to an archive tape. The tape volume name must be supplied so the program can check the tape label. Tapes must be initialized with ANSI standard labels prior to use (quicklab(1)). The *date, time* and *length* determine what data is written. Short- and medium-period segments having samples inside the archive window are written in their entirety. Long-period continuous waveforms are trimmed to the window boundaries prior to being written. By default, **arctwo** writes binary waveform tape files.

ARGUMENTS

–o outdev	Specifies the identifier for the archive tape device
–t tape	Specifies the archive tape volume name
–d date	Specifies the date of the data to be archived.
–s time	Specifies the time of the data to be archived.
–l length	Specifies the length of the data to be archived.

OPTIONS

–v	Produces very verbose output about the archive run-in progress. This output includes details of tape file opening and closing and tape usage statistics such as number of tape marks, record gaps and bytes written and the number of feet used to write them.
–A	Causes data to be converted to ASCII for writing export tapes. A fixed field width of 12 characters is used regardless of whether integer or floating point data is written.
–W	Produces a **.wftape** file with the same path name and prefix and which conforms to the external file format as the **.wfdisc** file. Therefore, the tape file can easily be loaded into an ingres data base.
–f file	Is the relative pathname (including prefix) of the **.wfdisc** file to be read. If not specified, then **arctwo** reads **.wfdisc** records from standard input, and writes **.wftape** records to standard output.
–c chanlist	Restricts the archive to the specified chanels (bands). Recognized letters are **s, m, i, l**. If used, the band specifications must be runs together (*i.e.*, **–c smil** or **–c sl**).
–w trimlist	Similar to **–c**, except it restricts the channels to be trimmed to the archive window.

FILES

file.**wfdisc**

SEE ALSO

quicklab(1), dmparc(1)

DIAGNOSTICS

It is an error to attempt to archive on a tape that has not been initialized with volume labels. Supplying the wrong volume name will also result in failure. Arctwo will not overwrite previously written data, but only appends. The tape can be positoned anywhere between BOT and EOI to begin with, but if it is

positioned after the EOI, it will run away and probably fail on a read error.

BUGS

arctwo currently pays no attention to the WFPATH environment variable. It only writes waveforms and takes them in the order found in the .wfdisc records it reads. It always builds tape files based on date and station names.

NAME

cbdump – dump the contents of a circular buffer

SYNOPSIS

cbdump dldesc [startrec [nrec]]

DESCRIPTION

Program **cbdump** dumps current circular buffer disk loop record values to standard output. The disk loop record number, the record tag value, and a hex dump of the remaining bytes in the record are printed for each disk loop record to be dumped.

ARGUMENT

dldesc	Name the disk loop descriptor file.

OPTIONS

startrec	Number of the record in the disk loop that starts the dump.
nrec	Number that specifies the number of records to dump.

SEE ALSO

cbdumptag(1)

NAME

cbdumpanchor – dump the contents of a circular buffer anchor file

SYNOPSIS

cbdumpanchor dldesc

DESCRIPTION

Program **cbdumpanchor** dumps the current circular buffer anchor file to standard output.

ARGUMENT

dldesc Name the disk loop descriptor file.

NAME

cbdumptags – dump the record tag values for the contents of a circular buffer

SYNOPSIS

cbdumptags dldesc

DESCRIPTION

Program **cbdumptags** dumps current circular buffer disk loop record tag values to standard output. The record tag value, and the first two words of the data frame (generally, some header information) are listed for all of the disk loop records. The flags normallly are double precision epoch times for each data frame.

ARGUMENT

dldesc	Name the disk loop descriptor file

SEE ALSO

cbdump(1)

NAME

cbgethead – return head time of the current circular buffer (cb) anchor buffer

SYNOPSIS

gethead cbdld

DESCRIPTION

Shellscript **cbgethead** invokes **cbdumpanchor** to output the head time of the current cb disk loop anchor buffer to standard output.

ARGUMENT

cbdld Name the cb disk loop descriptor file.

NOTE

To be run only when invoked by runnrtdb(1).

SEE ALSO

cbdumpanchor(1)

NAME

cbshowgaps – show data gaps in a circular buffer

SYNOPSIS

cbshowgaps dldesc

DESCRIPTION

Program **cbshowgaps** reads the contents of the circular buffer and prints to standard output beginning and ending record numbers, along with their respective tag numbers for consecutive data records. If a data gap is encountered, the beginning and ending record numbers of the gap are also listed.

ARGUMENT

dldesc Name the disk loop descriptor file.

BUGS

Works best on a static disk loop. Must be exited with a ^C (control-c).

NAME

cbtrack – track a circular buffer in real time

SYNOPSIS

cbtrack dldesc

DESCRIPTION

Program **cbtrack** tracks a cb implemented disk loop in real time. Reading begins at the disk loop head. The record number and tag value for each cb record is printed to standard output.

ARGUMENT

dldesc Name the disk loop descriptor file.

SEE ALSO

cbdump(1)

BUGS

Must be exited with ^C (control-C).

NAME

dlcp – copy a disk loop

SYNOPSIS

dlcp dldesc

DESCRIPTION

Program **dlcp** copies a disk loop. The operator enters the name of disk loop descriptor file to be copied, the name of the copy file, and the specification values for the new disk loop to be read from standard input by **dltouch**(1). The values for *nfiles*, *nrecords*, and *nbytes* must be the same for both disk loops or an error is generated.

ARGUMENT

dldesc	Name the disk loop descriptor file.

SEE ALSO

dltouch(1)

NAME

dlrm – remove a disk loop

SYNOPSIS

dlrm dldesc

DESCRIPTION

Program **dlrm** must be run in order to delete a disk loop. All of the files associated with the disk loop are removed by this program.

ARGUMENT

dldesc	Name the disk loop description file.

SEE ALSO

dltouch(1)

NAME

dltouch – create a new disk loop

SYNOPSIS

dltouch dldesc

DESCRIPTION

Program **dltouch** creates a disk loop and all of its associated files and nulls out their content. The program also creates an additional file (the disk loop descriptor file) that describes the disk loop and is used to specify it. Program **Dltouch** reads the disk loop specification values from standard input, which consist of the following:

nfiles	= The number of files making up the disk loop.
nrecords	= The number of disk loop records per file.
nbytes	= The number of bytes per disk loop record.
dlfile[0]	= The first disk loop file name.
dlfile[1]	= The second disk loop file name.
.	.
.	.
.	.
dlfile[nfiles-1]	= The nfiles'th disk loop file name.
anchorfile	= The disk loop anchor file name.

These values must appear in the proper order, but otherwise the input is free format. *Nfiles*, *nrecords*, and *nbytes* are specified as integer numbers and all of the file names are specified as normal UNIX file names (with directory path prefixes if appropriate).

If an error occurs while creating or nulling out any of the disk loop files, then all files that were successfully created are removed and an error message is generated.

ARGUMENT

dldesc	Name the output disk loop descriptor file.

NAME

hub2dl – coalesces raw hub data frames into disk loop data frames

SYNOPSIS

hub2dl **–i** dldesc [**–n** writesiz] [**–a** arzsiz] [**–t** tailincr] [**–u** anchorincr] [**–D** rawdumpfl] [**–d** baddumpfl] [{ **–s** | **-S**}]

DESCRIPTION

Program **hub2dl** continuously acquires raw hub data frames, performs rudimentary quality control checks on the frames, coalesces them into disk loop data records, and writes out the disk loop records. This program should be the sole writing program, at any given time, for the disk loop associated with a particular array data stream.

hub2dl acquires its raw hub frames from a DCP port, and it assumes that each data frame is a 100-byte SDLC frame with control/address values of **f325** hex (after byte swapping). The particular DCP port associated with a hub data stream is defined at compile time through the definitions **DCP_DEV** and **DCP_PORT** in the include file *hub2dl_defines.h* (see SOFTWARE MAINTENANCE).

Each disk loop data record consists of one nominal second worth of data and is laid out as follows. The first eight bytes of a disk loop record consist of the double floating epoch time associated with that second's worth of data. Following the epoch time is the 100-byte hub master frame and following the master frame are the 100-byte hub data frames for each data channel in the array. The list of acceptable hub channel identifiers (cid) and the order in which the data frames appear in the disk loop records are defined at compile time by the integer array **cid_table**, which is defined in the include file **hub2dl_inits.XXXXXX**.

For example, **cid_table** is initialized to the following in **hub2dl_inits.NORESS**.

```
int cid_table = {0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xa, 0xb, 0xc, 0xd,
                 0xe, 0xf, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16,
                 0x17, 0x18, 0x21, 0x22, 0x23, 0x24, 0xa1, 0xa2, 0xa3,
                 0x24, 0x61, 0x62, 0x63, 0x64, 0xc1};
```

With this cid table, **hub2dl** will expect 34 hub data frames per second with the above cids, and each disk loop record will be constructed by ordering the hub data frames within the disk loop record, with cid 0x4 as the first and cid 0xc1 as the last.

NOTE:

Run **hub2dl** in the background (by putting an & at the end of the command line) or else control of your terminal will be lost; the UNIX command **bg** is ignored by **hub2dl**. (See TERMINATION).

ARGUMENT

–i dldesc — Name the disk loop descriptor file. This file is created by running **dltouch**.

OPTIONS

–n writesiz — Specify the number of disk loop records to be written to the disk loop per write operation. Each record is equivalent to one second of time. Default = 10. In addition to specifying the disk loop writing frequency, *writesiz* also determines the frequency at which the *hub2dl.stat* file is updated (see FILES).

–a arzsiz — Specify the length (records/seconds) of the allowable read zone. Specification must not exceed the size of the disk loop. Default _ 80% of total disk loop size.

–t tailincr — Increment (records/seconds) by which the ARZ tail is updated. Each time a write to the disk loop occurs, the length of the ARZ is increased by *writesiz* number of records. Once the ARZ length exceeds *arzsiz*, the ARZ tail is advanced by *tailincr*

number of records (or seconds), effectively shrinking the size of the ARZ until enough writes occur to increase it to *arzsiz* again. Default = 10% of total disk loop size.

-u anchorinvl Specify the maximum allowable time interval (seconds) between anchor file interior entries. If an update to the anchor file results in an interval between entries greater than this value, new interior entries are created as needed. Default = 10% of total disk loop size.

-D rawdumpfil Name the dump file where *all* raw frames received from the DCP port are to be written. This is a binary file with each SDLC frame resulting in a four byte integer frame byte count followed by the frame itself (after byte swapping). The program **bin2asc** can convert this binary file into an ASCII dump of the SDLC frames. This option causes *all* SDLC frames to be dumped immediately after they are received from the DCP, so the resulting dump file grows rapidly. The only way to disable this option, once **hub2dl** is running, is to terminate execution of **hub2dl** (see TERMINATION). If not specified, then the raw SDLC frames are not dumped.

-d baddumpfil Name the dump file where only discarded SDLC frames are to be written. (Discarded frames are those that failed one or more of the quality control checks.) This is a binary file with each SDLC frame resulting in a four byte integer frame byte count followed by the frame itself (after byte swapping). The program **bin2asc** can convert this binary file into an ASCII dump of the SDLC frames. The only way to disable this option, once **hub2dl** is running, is to terminate execution of **hub2dl** (see TERMINATION). If not specified, then the bad SDLC frames are not dumped.

-s Silencing flag - causes all quality control errors except DF02 errors (bad data frame 0xfa character) to be reported in the *hub2dl.log* file. Default = all quality control errors are reported to the *hub2dl2cb.log* file.

-S Alternate silencing flag – causes all quality control errors except DF01 (unexpected data frame), DF02 (bad data frame 0xfa character), and DF03 (wrong data frame mf count) errors to be reported in the *hub2dl.log* file. Default = all quality control errors are reported.

QUALITY CONTROL

One of the major functions of **hub2dl** is to make rudimentary quality control checks of each SDLC frame before that frame is deposited into the disk loop. The intent of this quality control checking is to ensure that the disk loop contains only *good* data so that reading processes need not be concerned with the validity of the data as it appears in the disk loop. Whenever a frame fails one of the QC checks it is discarded. This results in a disk loop data gap. No attempt is made to patch up a frame that fails a QC check.

The quality control checks are based upon assumed characteristics of the hub SDLC frames and the order in which they appear in the data stream. These assumptions are listed below.

1. Each hub SDLC frame is 100 bytes in length and has SDLC control/address bytes of **f325** hex (after byte swapping).
2. For each second worth of array data there are nominally 35 SDLC frames consisting of one master frame and 34 data frames each data frame corresponding to one of the cid's given in Bcid_table.
3. The master frame always immediately precedes the 34 data frames for a given second worth of

array data.

4. The 34 data frames always immediately come after the master frame for a given second worth of data. There may be less than 34 data frames in situations where one or more sensors were out when the hub startup occured, however there are never more than 34 data frames, and each data frame has a unique cid that corresponds to one of the cid's in **cid_table**.

5. The GM time as determined from the master frame is never greater than the real GM time when that frame was received (within some clock uncertainty). In other words, a master frame will not contain a future time stamp.

6. The GM time as determined from the master frame is always monotonically increasing with real time. In other words, a master frame will not contain a time stamp that is older than the previous master frame's time stamp.

7. Each data frame contains a byte near the end of the frame that is equal to **fa** hex.

8. Each data frame contains a master frame count that is equal to the master frame count that was contained in the previous master frame.

Immediately after receiving an SDLC frame, the first two QC checks are made. If one of the checks fails, then the frame is discarded and one of the following error codes is generated.

FR01 Wrong SDLC frame byte count (not equal to 100).

FR02 Wrong SDLC frame control/address bytes (not equal to **f325** hex).

The checks are made in the order given above. After the frame has passed the first two checks, it is identified is either a master frame or a data frame. If it is a master frame, then one of the following three errors is generated if the corresponding check fails.

MF01 The master frame GM time words are undecipherable.

MF02 The master frame GM time is greater than the real GM time when the frame was received.

MF03 The master frame GM time is less than the previous master frame GM time.

For data frames five QC checks are made and one of the following five errors is generated if the corresponding check fails.

DF01 Unexpected data frame. This happens whenever there were 34 previous data frames with the correct master frame counts and cid's. Once 34 good data frames have been recieved, then it is assumed that the next frame will be a master frame.

DF02 This data frame does not contain the **fa** character in its proper place.

DF03 The master frame count for this data frame is not equal to the master frame count for the previous master frame.

DF04 The cid for this data frame is not in the list of cid's in *cid_table*.

DF05 This data frame has a cid that has already been received (repeated cid).

Once again the checks are made in the order given above and any frame that fails one of the checks is discarded.

TERMINATION

Program **hub2dl** is meant to run constantly in normal data aquisition tasks; therefore, all keyboard interrupts and all but the most severe kill(2) requests have been disabled. There is, however, a mechanism which allows temporary (up to 30 minutes) suspension of operation to allow changes to be made without loss of data:

kill -HUP process id

This UNIX command option gracefully terminates execution of **hub2dl**, including the **DCP** program, with all buffers properly flushed, and all data in the disk loop preserved. The *white box* will buffer data for up to 30 minutes; therefore, **hub2dl** can be restarted using the same disk loop descriptor file (*dldesc*) without incurring the loss of any data.
Note: this is the *only* command that should be used to bring down **hub2dl** for any reason.

FILES

Running **hub2dl** causes two files to automatically be written: *hub2dl.log* and *hub2dl.stat*. The actual names for these two files are specified in **hub2dl_defines.h**. *hub2dl.log* records the initial startup parameters, and maintains a list of any frames that fail the QC checks. (The actual error conditions that are reported depends upon which, if any, silencing option is chosen). If a file by this name already exists, it is appended to.

File *hub2dl.stat* records the current status of **hub2dl** processing. *hub2dl.stat* is re-written whenever disk loop records are written to the disk loop, as specified by *writesiz*. This file is accessed by the disk loop utility, **showstats** (no arguments), which continuously displays *hub2dl.stat* onto a VT100 type terminal screen.

Specifying either the **-D** or **-d** options causes an additional dump file to be written as well.

SOFTWARE MAINTANANCE

Some options in **hub2dl** have been hard-wired into include files that a user may need to change. These include the cid table and DCP port specifications, and the names of the log and stat files. All but the cid table are specified in **hub2dl_defines.h**. Cid tables are specified in both **hub2dl_inits.FINESS** and **hub2dl_inits.NORESS**. If a user makes changes to an include file, the file *hub2dl.o* must be removed, and the source re-compiled with the UNIX **make** command, in order to incorporate the changes.

SEE ALSO

dltouch(1), dlrm(1)

NAME

hub_cb2db – converts cb records into a Version 2.8 Center data base

SYNOPSIS

hub_cb2db –i cbname –o dbname –r recipflnm –ts stime {–tg timgate | –te etime}]
[–d wavformdir] [–f datype] [–b timeout] [–V]

DESCRIPTION

Program **hub_cb2db** reformats circular buffer data records into a Center data base (design meets CSS Version 2.8 standards). The processing for **hub_cb2db** is controlled both by its command-line arguments and its recipe file. The recipe file specifies which channels in the cb records are to be converted, and the command-line arguments define the time window to be converted and the pathnames to the circular buffer and the output data base.

When invoked, **hub_cb2db** reads the recipe file and creates a **.wfdisc** record and a .w file for each seismic channel specified in the recipe file. The seismic channel is specified by the channel and circuit identifiers (chid and cbid fields) in the recipe file and similarly in the **.wfdisc** file. The matching identifier in the cb record is the circuit identifier (cid). It is the function of the recipe file to map channel identifiers to circuit identifiers. If a frame identifier specified in the recipe file cannot be found in the cb record, then a warning message is written to standard error and this channel is omitted from the data base.

Then **hub_cb2db** checks the circular buffer for the first data frame containing an epochal time stamp greater than or equal to *stime*. It processes data forward from this cb data frame until it reaches a frame with a time stamp greater than or equal to *etime*. A new **.wfdisc** record is created each time a channel experiences a data drop. Currently, a single .w file is produced for each channel specified in the recipe file. These files contain all relevant data segments indexed by the different **.wfdisc** records,

Upon completion, a Center data base resides in the output data base directory. It consists of one **.wfdisc** file, one **counter** file, and one .w file per channel, and contains all the data found for this channel in the requested time window.

Any errors that occur during processing are reported to standard error. Whenever possible, processing continues.

ARGUMENTS

–i cbname — Name the cb disk loop descriptor file, which contains information on the structure of the circular buffer and is created by **dltouch(1)**.

–o dbname — Prefix (or pathname with prefix) of the output database where the **.wfdisc** and **.counter** files are to be written. If *dname* does not begin with a slash(/) or a dot (.) and if the WFPATH environment variable is set, then it is assumed to be a path relative to WFPATH. Otherwise, it is a path relative to the current directory.

–r recipflnm — Prefix (or pathname with prefix) of the recipe file that drives **hub_cb2db** processing.

–ts stime — Time of the first (oldest) record to begin processing. Time may be specified in any one of the following formats:

544920051.123	- epoch time
1987:may:03:8:45:56.34	- yr:month:day:hr:min:sec
1986:253:14:55:02.5	- yr:doy:hr:min:sec
1985043:23:10:13	- juliandate:hr:min:sec

Reading begins at the first record in the circular buffer with an epochal time stamp

greater than or equal to *stime*. If no such cb record exists, then no output database is produced.

-te time — Epochal time at which to stop processing records. Must be greater than *stime*.

-tg timgate — An alternative to specifying *etime* is specifying *timgate* in which case, *etime* is derived by adding *timgate* to *stime*.

OPTIONS

-d wavformdir — Prefix (or pathname with prefix) of the waveform directory where the raw waveform data is to be written. If omitted, the .w files are written according to the *dbname* specification.

-f datatyp — Format (**s4** or **t4**) of the raw waveform (.w) data files.

0	= s4	= 4 byte IEEE integer
1	= t4	= 4 byte IEEE float

Default is **0**.

-b timeout — Timeout value (in seconds) for cb record read blocking. If a record is requested in the future, **hub_cb2db** will block for *timeout* number of seconds to try and fulfill the request. Cb records are processed in 10 second increments; therefore, a *timeout* of 20 should be sufficient for concurrent processing. Default = 0 (no read blocking).

-V — Verbose flag causes display to standard output of all of the command line argument values. Default values are shown for all options not specified on the command line.

FILES

dname.**wfdisc**
dbname.**counter**
dbname.**w**
hub_cb2db_recipe

SEE ALSO

intro(1)
hub_cb2db_recipe(5)
dltouch(1)

BUGS

If the recipe file is not of the proper form, random errors can occur. This follows the cid assignment convention explained in the "HUB SDLC PORT AND HUB SATELLITE PORTS FORMAT" document (Sandia National Laboratories). If this convention is violated, the output database will probably be corrupted.

NAME

hub_cb2db_recipe – input file that drives the **hub_cb2db** process

DESCRIPTION

File **hub_cb2db_recipe** specifies each seismic channel that is to be extracted from the cb records and converted into Version 2.8 Center data base format.

FORMAT

The **hub_cb2db_recipe** file consists of a series of records. Each record specifies a seismic channel contained in the circular buffer by a unique frame identifier (cbid), its corresponding channel identifier (chid) in the Center Database, as well as certain other fields of the wfdisc relation. Each record has the following format:

station channel cbid chid calib calper instyp

In the current implementation, the meanings attached to these fields are strictly those defined in Version 2.8 of the Center Database definitions.

FIELDS

station – A 6-character channel station identification code.

channel – A 2-character channel name. The first character identifies the period length, such as **s** for short, **m** medium, **l** for long, or **b** for broad band, and the second character identifies the direction, such as **z** for vertical, **n** for north, and **e** for east.

cbid – A hex frame identifier. Must correspond to an actual cid in the circular buffer. See hub_2dl(1) for the full list of cids in the data stream.

chid – A unique trace channel identifier. An integer in the range 0 to 99999999 that determines the names of the waveform (.w) files for each channel.

calib – The calibration factor.

calper – The calibration period in seconds.

EXAMPLE

Recipe file for hub_cb2db on NORESS data

# sta	ch	cbid	chid	calib	calper	instyp
D1Z	sz	04	01	0.006837	1.0	GS-13
D2Z	sz	05	02	0.006837	1.0	GS-13
D3Z	sz	06	03	0.006837	1.0	GS-13
D4Z	sz	07	04	0.006837	1.0	GS-13
D5Z	sz	08	05	0.006837	1.0	GS-13
D6Z	sz	09	06	0.006837	1.0	GS-13
A0Z	sz	21	07	0.006837	1.0	GS-13
A0N	sn	61	08	0.006837	1.0	GS-13
A0E	se	a1	09	0.006837	1.0	GS-13
E0Z	mz	c1	10	0.077300	1.0	KS36K4
E0N	mn	c1	11	0.077300	1.0	KS36K4
E0E	me	c1	12	0.077300	1.0	KS36K4
E0LPZ	lz	c1	13	0.001447	1.0	KS36K4
E0LPN	ln	c1	14	0.001447	1.0	KS36K4
E0LPE	le	c1	15	0.001447	1.0	KS36K4

KS36K4 is an abbreeviation for KS-36000-04

In the example above of a hub_cb2db recipe file, 15 channels will be extracted from the cb records and converted into Version 2.8 Center Database format. The first six are short period vertical channels with frame identifiers 04 through 09 hex, and channel identifiers 1 through 6 decimal in the Center data base format. The next three channels come from a 3-axis instrument. They have frame identifiers 21, 61, and a1 hex, and will have channel identifiers 7 through 9 decimal in the Center data base format.

The remaining records pertain to a frame containing broad-band data, with frame identifier c1. Each broad-band frame in the circular buffer contains data mapped to six seismic channels, three medium-period components and three long-period components. In the example, the 3-component medium-period channels are mapped into channel identifiers 10 through 12 in the Center data base format, and the 3-component long-period channels are mapped into channel identifiers 13 through 15 in the Center data base format.

Lines beginning with a number sign (#) are comments and can be inserted anywhere within the recipe file.

SEE ALSO

hub_cb2db(1)

CHAPTER 6

FILE FORMATS

October 1988

CHAPTER 6

NORSAR FILE FORMATS

6.1. Introduction

The following pages contain the detailed documentation for the Seismic Monitoring System section (5) FILE FORMATS that are relevent to the NORSAR system.

NAME

intro – introduction to input file formats

DESCRIPTION

The *files* section specifies the format for files used to input information to Science Horizons applications programs. All file format pages are based on a common format, but not all sections always appear. There is always a description section that describes the content of the file.

FORMAT

This section names and provides information about the file records.

FIELDS

This section identifies and explains the record fields.

EXAMPLE

This section contains an example of the file.

NAME

hub_cb2db_recipe – input file that drives the **hub_cb2db** process

DESCRIPTION

File **hub_cb2db_recipe** specifies each seismic channel that is to be extracted from the cb records and converted into Version 2.8 Center data base format.

FORMAT

The **hub_cb2db_recipe** file consists of a series of records. Each record specifies a seismic channel contained in the circular buffer by a unique frame identifier (cbid), its corresponding channel identifier (chid) in the Center Database, as well as certain other fields of the **wfdisc** relation. Each record has the following format:

station channel cbid chid calib calper instyp

In the current implementation, the meanings attached to these fields are strictly those defined in Version 2.8 of the Center Database definitions.

FIELDS

station – A 6-character channel station identification code.

channel – A 2-character channel name. The first character identifies the period length, such as **s** for short, **m** medium, **l** for long, or **b** for broad band, and the second character identifies the direction, such as **z** for vertical, **n** for north, and **e** for east.

cbid – A hex frame identifier. Must correspond to an actual cid in the circular buffer. See hub_2dl(1) for the full list of cids in the data stream.

chid – A unique trace channel identifier. An integer in the range 0 to 99999999 that determines the names of the waveform (.w) files for each channel.

calib – The calibration factor.

calper – The calibration period in seconds.

EXAMPLE

Recipe file for hub_cb2db on NORESS data

# sta	ch	cbid	chid	calib	calper	instyp
D1Z	sz	04	01	0.006837	1.0	GS-13
D2Z	sz	05	02	0.006837	1.0	GS-13
D3Z	sz	06	03	0.006837	1.0	GS-13
D4Z	sz	07	04	0.006837	1.0	GS-13
D5Z	sz	08	05	0.006837	1.0	GS-13
D6Z	sz	09	06	0.006837	1.0	GS-13
A0Z	sz	21	07	0.006837	1.0	GS-13
A0N	sn	61	08	0.006837	1.0	GS-13
A0E	se	a1	09	0.006837	1.0	GS-13
E0Z	mz	c1	10	0.077300	1.0	KS36K4
E0N	mn	c1	11	0.077300	1.0	KS36K4
E0E	me	c1	12	0.077300	1.0	KS36K4
E0LPZ	lz	c1	13	0.001447	1.0	KS36K4
E0LPN	ln	c1	14	0.001447	1.0	KS36K4
E0LPE	le	c1	15	0.001447	1.0	KS36K4

KS36K4 is an abbreeviation for KS-36000-04

In the example above of a hub_cb2db recipe file, 15 channels will be extracted from the cb records and converted into Version 2.8 Center Database format. The first six are short period vertical channels with frame identifiers 04 through 09 hex, and channel identifiers 1 through 6 decimal in the Center data base format. The next three channels come from a 3-axis instrument. They have frame identifiers 21, 61, and a1 hex, and will have channel identifiers 7 through 9 decimal in the Center data base format.

The remaining records pertain to a frame containing broad-band data, with frame identifier c1. Each broad-band frame in the circular buffer contains data mapped to six seismic channels, three medium-period components and three long-period components. In the example, the 3-component medium-period channels are mapped into channel identifiers 10 through 12 in the Center data base format, and the 3-component long-period channels are mapped into channel identifiers 13 through 15 in the Center data base format.

Lines beginning with a number sign (#) are comments and can be inserted anywhere within the recipe file.

SEE ALSO

hub_cb2db(1)

CHAPTER 7

CIM MAINTENANCE

October 1988

TABLE OF

CONTENTS

CHAPTER 7 – CIM MAINTENANCE

FIGURES

TABLES

CHAPTER 7

CIM MAINTENANCE

7.1. Introduction

The Communications Interface Module (CIM) is an automated, low maintenance microprocessor-based unit designed as a stand-alone communications processor to capture and retransmit near-real-time data. The actual function and internal configuration of each CIM is dependent on the requirements of a specific installation.

Each CIM is configured and installed by Science Horizons and is expected to operate uneventfully. However, because the function of any CIM is to process a continuous stream of data, it is important to ensure that it operates without interruption. Therefore, as an option to the CIM installation, Science Horizons offers a spare parts kit that enables users to replace a malfunctioning part without the delay of waiting for a replacement. The Spare Parts Kit includes the following components:

- 1 68010 CPU board complete with EPROM firmware
- 1 4MB memory board,
- 1 full RS-422 interface card,

This chapter provides procedures and criteria to help you diagnose an exception and, if necessary, configure and replace a malfunctioning board. For those without the spare parts, your correct problem analysis will clarify communications and expedite replacement agreements with Science Horizons. The following procedures are described:

- System take down and start up
- Fault isolation analysis
- Removal and replacement of components
- Board configuration

Useful diagrams, pinouts and jumper configuration tables are supplied. For additional explanation and background, see the CIM chapter of this manual.

7.2. Components

Each CIM unit contains the same basic components:

- Lighted power and reset switches on the front panel
- Two banks of serial communications ports and the power connector on the back panel
- A 325W multiple output switching power supply
- UL/VDE/CSA approved line filter
- 115 to 230 VAC input power recepticle
- Dual 78 cfm cooling fans
- A 7-slot card cage
- Two interface cards for RS-232 to RS-422/449 conversion

7.3. Card Cage Configuration

The card cage configuration of each CIM varies with its function and site-specific requirements.

- A CIM used only for buffering contains at least one microprocessor board, one interface card, and at least three 4MB memory boards.
- A multiplexing CIM contains a minimum of two CPU boards, two interface cards and one memory board.
- A demultiplexing CIM contains the same number of CPU boards and interface cards as its companion multiplexer (in a single CIM configuration, a demultiplexing unit contains at least 2 CPU boards and two interface cards) and at least three memory boards.

In addition, the master CPU board in a multiplexer unit may contain a crystal oscillator for generating the modem clock. Figures 9-1 and 9-2 show the order of the basic boards in a multiplexer and a demultiplexer card cage.

The only absolute rules are:

1. The the master cpu board *must* occupy the bottom slot.
2. In a two-CIM configuration, the multiplexer and demultiplexer *must* have the same number of microprocessor boards.

By convention, the CPU boards occupy the bottom slots, the memory boards occupy the top slots.

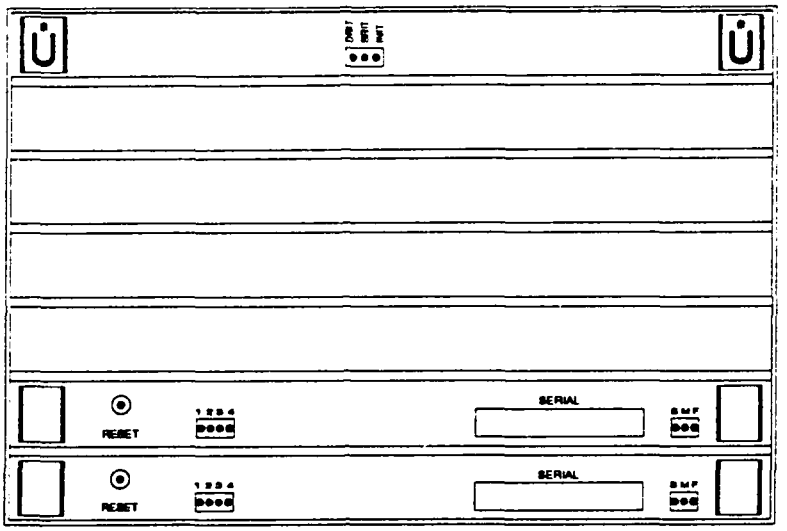


Figure 7-1. Basic multiplexer card cage stack order

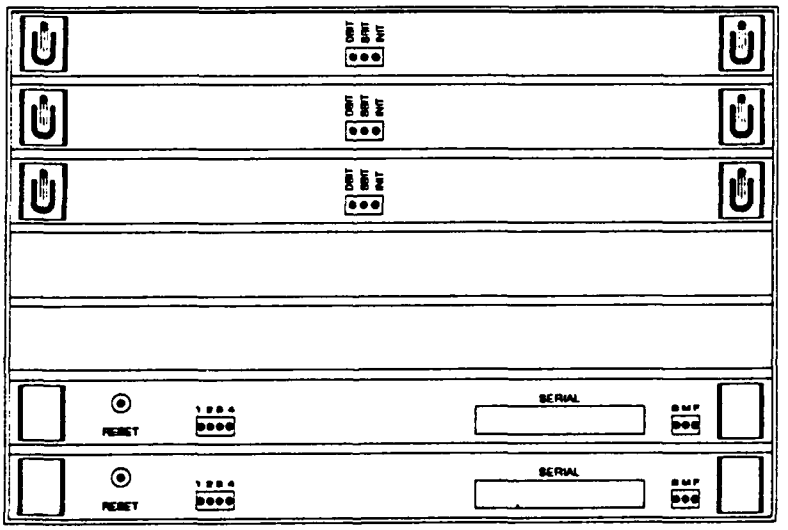


Figure 7-2. Basic demultiplexer card cage stack order

7.4. Power Switch and Reset

Figure 7-3 shows the power switch and reset button on the CIM front panel.

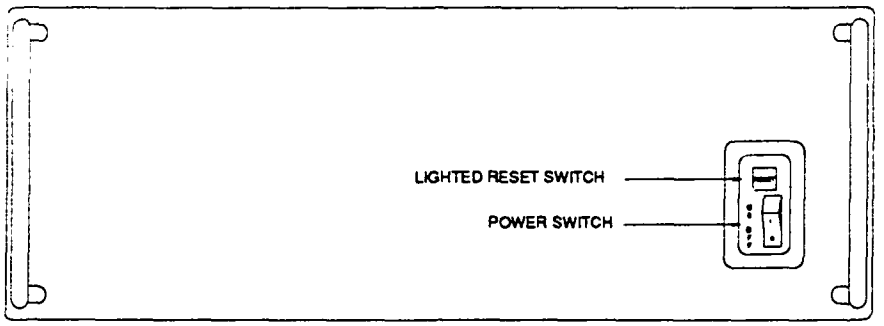


Figure 7-3. Front panel of CIM cabinet

Power Switch:

- Turn the switch to *off* to power down the CIM in preparation for board replacement.
- Turn the switch to *on* to connect power and restart the CIM.

  NOTE: Disconnect the power cord before sticking your hands inside the box.

Reset Button:

- Press the button to restart the CIM in the event the CIM stops operating when there is no power loss. To learn the cause of the suspension, follow instructions in the paragraph on Fault Isolation.

Initialization

- Occurs immediately after power *on* or reset.
- Performs diagnostic check of SYSFAIL bus signal and components. Reports failure due to:
  - Bus error
  - Illegal configuration
  - Slave memory bounds error

7.5. Exception Reporting

If the CIM stops functioning and is not restarted by pressing the reset button, the next step is to examine the various light emitting diode (LED) displays on the back edge of the boards in the card cage. Most of the error conditions that can occur in the CIM are reported by illuminating one of the LED displays with a specific pattern and then suspending processor activity until a system reset occurs.

Figure 7-4 shows the location of the LEDs and reset button on the rear edge of a processor board, and Figure 7-5 shows the location of the LED display on the rear edge of a memory board.

Figure 7-4. Back view of microprocessor board

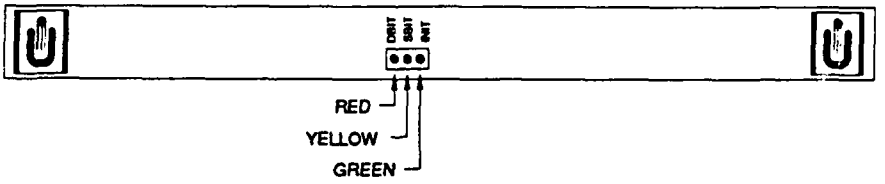


Figure 7-5. Back view of memory board

7.5.1. LED Displays

Specific conditions are indicated by the illumination of specific displays:

On memory boards:

- A green LED means initialization is complete and operation is normal.
- A yellow LED means a single bit error has occurred since last system reset. These errors are self-correcting and do not cause a system interrupt.
- A red LED means a double bit or multiple bit error has occurred since the last system reset. These errors generate bus errors and can cause a system interrupt.

On processor boards:

- There are two groups of LEDs on the back edge labeled SMF and 1234.
  - The SMF LEDS (Slave, Master and Fail) indicate the manner in which the board is interacting with the VMEbus (slave mode, master mode or neither), and monitors the system.
  - The 1234 set is referred to as user LEDs and are used to monitor the health of the CIM module.
- The S LED glows only if the on-board memory is being addressed by some other board, which does not occur in a CIM except as an error. If the condition is not cleared by a reset, it is likely that a system component on another CPU board has failed.
- The M LED glows only on a master board (the board that controls the bus).
- The F LED is a system fail indicator that monitors the state of the SYSFAIL bus signal. It will extinguish after system initialization unless some on-board component has failed.
- The 1234 LEDs can show three different illumination modes.
  - Walking mode, where each LED blinks on and off in turn, means operating conditions are normal.
  - Fixed mode, where the group shows a steady pattern, is caused by system related errors. The specific pattern is a code that indicates the type of error.
  - Flashing mode, where the group shows a flashing pattern, is caused by software related errors. The specific pattern is a code that indicates the type of error.

7.6. Fault Isolation

There can be two basic reasons for CIM failure:

- Component failure
- Faulty configuration

If the CIM fails after successful operation and is not restarted by pressing the system reset on the front panel, chances are that a component CPU, memory, or interface card has failed and needs to be replaced.

If failure occurs during initialization after a replacement board has been installed, there are three probable cases in which a system reset will not eliminate the error condition:

- The user DIP switches have been improperly set
- The starting address of slave memory (or the CSR register) is improperly configured
- A board has been replaced in the wrong slot in the card cage so that the backplane is no longer properly strapped relative to the boards in the card cage.

Unfortunately, depending on the circumstances, these configuration errors can lead to the posting of a bus error, an illegal configuration error, or a slave memory bounds error.

To determine the exact cause, it is necessary to remove the CIM rear panel to examine the LED displays.

7.6.1. Analysis Procedure

1. Unscrew the CIM back panel.
2. Carefully pull the panel away from the chassis without disturbing the power cord and connectors.
3. Check the user 1234 LEDs on the back edge of each microprocessor board in the card cage. If the 1234 LEDs are blinking on and off in turn, skip to step 4. If the 1234 LEDs are flashing a pattern or glowing in a steady pattern:
   - Compare the displayed patterns with the patterns in Table 7-1.
   - Press the reset button on the front panel.
   - If reset does not correct the problem and restart the CIM, follow the corrective action in the table for the indicated error.

   NOTE: Occasionally, power cycling the CIM clears error conditions when a reset does not.
4. Check the SMF LEDs on the processor boards, if only the M LED glows occasionally, skip to step 5.

- The S LED glows only if the board is operating in slave mode, which indicates an error condition in the CIM. If the system has been running for awhile, a glowing S LED may mean that a system component has failed, causing the processor to take a *wild jump* to another CPU's on-board memory. If the condition is not cleared by a reset, it probably means a component on another CPU board has failed and the other board should be replaced.
- The F LED is a system fail indicator that monitors the state of the SYSFAIL bus signal during system initialization. If the LED continues to glow after initialization, it indicates that some system component (likely on-board memory) has failed, and the board should be replaced.
- The master M LED will light intermittantly to indicate the board has the VMEbus and is addressing some off board address between 100000_{16} and $FE0000_{16}$.

5. Check the color of the LED on the memory boards:
   - A red glow signifies a double bit error on a memory board. This, along with a bus error signal on a processor board, means the memory board is at fault. If a reset does not clear the condition, replaced the memory board.
   - A yellow glow means a single bit error has occurred since the last reset. This error is self-correcting and does not cause suspension of operation.
   - A green glow means that initialization is complete and operation is normal.

6. If the CIM has restarted, replace the back panel, If not, follow the board removal and replacement procedures in the next section.

Table 7-1. CPU Trouble Guide

Steady Pattern				System Error	Action
●	○	○	○	bus error	reset / check memory address jumpers check CSR address jumpers check DIP switch settings check card cage locations
○	●	○	○	privilege violation	reset / replace component
●	●	○	○	illegal instruction	reset / replace component
○	○	●	○	illegal address	reset / replace component
●	○	●	○	divide by zero	reset / notify SHI
○	●	●	○	bounds check	reset / notify SHI
●	●	●	○	two's complement overflow	reset / notify SHI
○	○	○	●	parity error	reset / replace component
●	●	●	●	purge in progress	no action

● = on ○ = off

Flashing Pattern				Software Error	Action
●	○	○	○	illegal configuration	reset / check DIP switch settings check memory address settings
●	●	○	○	read error	reset / notify SHI
○	○	●	○	error opening port	reset / notify SHI
●	○	●	○	error creating task	reset / notify SHI
○	●	●	○	error completing write	reset / notify SHI
●	●	●	○	write error	reset / notify SHI
○	○	○	●	slave memory bounds error	reset / check DIP switch settings

● = flashing ○ = off

7.7. Removing/Replacing Components

Whether or not an existing board needs to be replaced or just reconfigured, the CIM must be powered down and one or more boards removed, examined and reinstalled. The only reason for removing an interface card is to replace it with one from the Spare Parts Kit.

7.7.1. Board Removal Procedure

To remove a board or interface card:

1. Power down the module by turning the power switch on the front panel of the CIM to the *off* position.
2. Disconnect the power cord from back panel. Notice which connectors are associated with which ports before proceeding.
3. Unscrew the DB25 connectors on the back panel of the module, and disconnect. Retain screws.
4. Unscrew back panel and remove. Retain screws.
5. If removing a CPU board, disconnect the cable to the serial port by means of the ejector wings on either side of the connector. This is done by placing the thumbs on the inner side of the ejector wings and pressing outwards, parallel to the edge of the CPU board, until the connector is ejected.

   If removing a CPU or memory board, continue at Step 6. If removing a faulty interface card, skip to Step 9.
6. Unscrew the faulty board (screws are captive).
7. Using the ejector wings on either end of the faulty board, press outward with the thumbs, parallel with the edge of the board, until the board is ejected.
8. Remove the faulty board from card cage, noticing the slot from which the card is removed.
9. If you are removing an interface card, unclip the card from the back panel of the CIM, and remove the DB25S connectors from the back panel.

   NOTE: On some newer units, these boards are mounted on standoffs in the floor of the CIM chassis. To remove the board, unscrew the two #4-40 machine screws that hold it in place.

7.7.2. Board Replacement Procedure

Replacement can mean reinstalling a removed CPU or memory board after examination and correction, or it can mean replacing an existing board with a new one from the spare parts kit.

To replace a board in the card cage:

1. Take the appropriately configured CPU or memory board and insert it into the card cage slot left vacant by removal of a faulty board.
2. Push firmly on the edge of the board until the ejector wings snap back into place and the board is firmly seated against the backplane of the card cage.
3. Screw the board to the cardcage with the screws at either end. This is important as the board may not be properly seated in the card cage or may not remain properly seated over time unless the board is securely screwed into place.
4. If replacing a CPU board, next reconnect the cable to the serial I/O port.
5. It is very important that the appropriate cable and conversion card be connected to the serial I/O port of the processor card.
   - If the faulty board has all interface chips removed and replaced by jumpers, then the board is generating TTL levels on both ports. Therefore, connect it to an RS-422 interface card that converts both ports to RS-422.
   - If only a portion of the interface chips have been replaced by jumpers, then the card is generating TTL levels on port A and RS-232C levels on port B. Therefore, connect the board to an RS-422 interface card that converts only port A to RS-422.
6. In the proper relative orientation, both the components on the processor card and the stripes on the 34-wire ribbon cable are up, and pin 1 of this cable is on the right. In addition, a small rectangular protrusion on one of the long edges of the connector fits into a small slot in the connector body on the CPU board, ensuring that the connector cannot be inserted upside down.
   - With the cable connector in the proper orientation, push it firmly against the board connector until the ejector wings on the board snap back into place.
7. Replace the back panel and, using screws retained when back panel was removed, screw back panel into place.
8. Reconnect the DB25 connectors to the appropriate ports on the back panel of the module, and using the appropriate screws (these are captive), screw connectors into place.
9. Reconnect the power cord to the back panel.
10. Power up the module by turning the power switch on the front panel of the CIM to the *on* position.

7.7.2.1. Interface Card Replacement Procedure

To replace an interface card:

1. Select the appropriate interface card from the Science Horizons Spare Parts Kit:
   - A full interface card if both ports A and B use RS-422 signal levels
   - A card that converts only port A.

   A half interface card with the baud rate generator for the modem is not included in the Spare Parts Kit. If this card in a CIM multiplexer is faulty, contact Science Horizons for a replacement.
2. Connect the 34-pin connector at the end of the interface card cabling to the serial I/O port on the appropriate processor card.
   - In the proper relative orientation, both the components on the processor card and the stripes on the 34-wire ribbon cable are up.
   - With the cable connector in the proper orientation, push it firmly against the board connector until the ejector wings on the board snap back into place.
3. Clip the interface card to the back panel of the CIM with the component side facing the interior of the module.
4. Place the DB25 connectors in their appropriate punch-outs on the back panel, being sure that the cables carrying the signals from ports A and B are not reversed.
7. Replace the back panel and, using screws retained when it was removed, screw the back panel into place.
8. Reconnect the DB25 connectors to the appropriate ports on the back panel of the module, and using the appropriate screws, screw connectors into place.
9. Reconnect the power cord to the back panel.
10. Power up the module by turning the power switch on the front panel of the CIM to the *on* position.

7.8. Board Configuration

When replacing a faulty board with a new one, it is important to duplicate the slot position and configuration of the original board. The following procedures provide the criteria and steps, but replication of the old board is the required result.

7.8.1. CPU Board Configuration

There are three items on a new microprocessor board that require configuration before it can successfully be placed in the CIM:

- Interface chips
- DIP switches
- Self arbiter and sysclock jumpers on the master board

Figure 7-6 is a CPU configuration map that shows the location of these items.

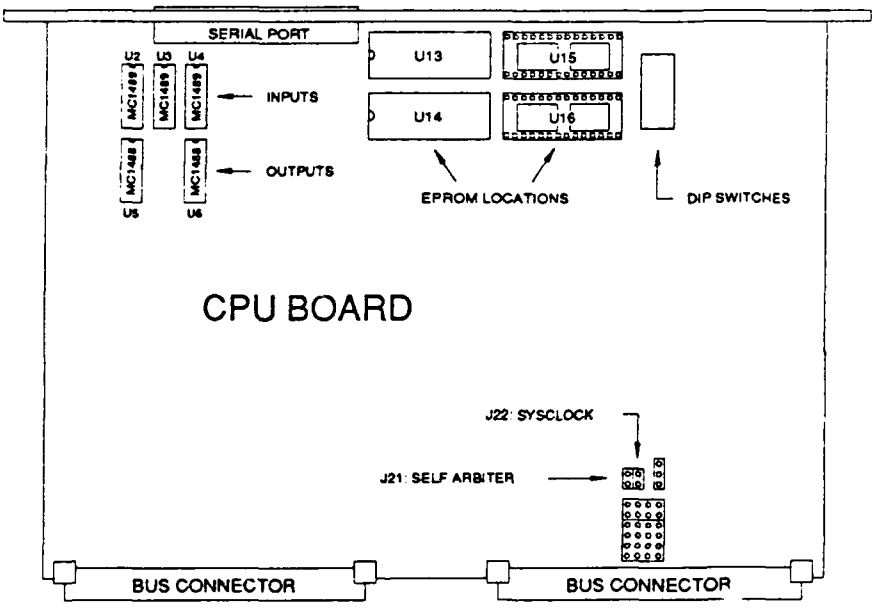


Figure 7-6. CPU Configuration Map

7.8.1.1. Interface Chip Configuration

In the CIM there are two possible interface configurations. Either both ports are converted to RS-422 signal levels, or only Port A is converted, while Port B is left configured with RS-232C signal levels. Figure 7-7 is a blow-up of the interface chip location.

- When both ports are converted to RS422 signal levels, all five interface chips are removed and replaced with jumper chips as shown in Figure 7-8.
- When only Port A is converted, only three RS-233C interface chips are replaced as follows (see Figure 7-9):
  1. Remove chips U2 and U5 and replace them with jumper chips.
  2. Remove chip U3 and replace it with the MC1489 chip on a special jumper block. On this chip, half the pins are removed and the corresponding signals are jumpered. The other half are soldered to a socket, which makes the chip sit somewhat higher than normal.

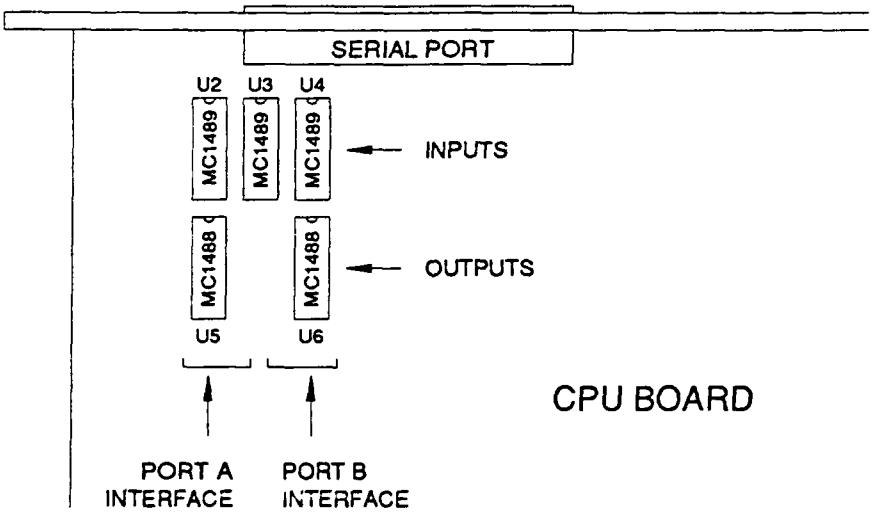


Figure 7-7. Location of RS-232C interface chips.

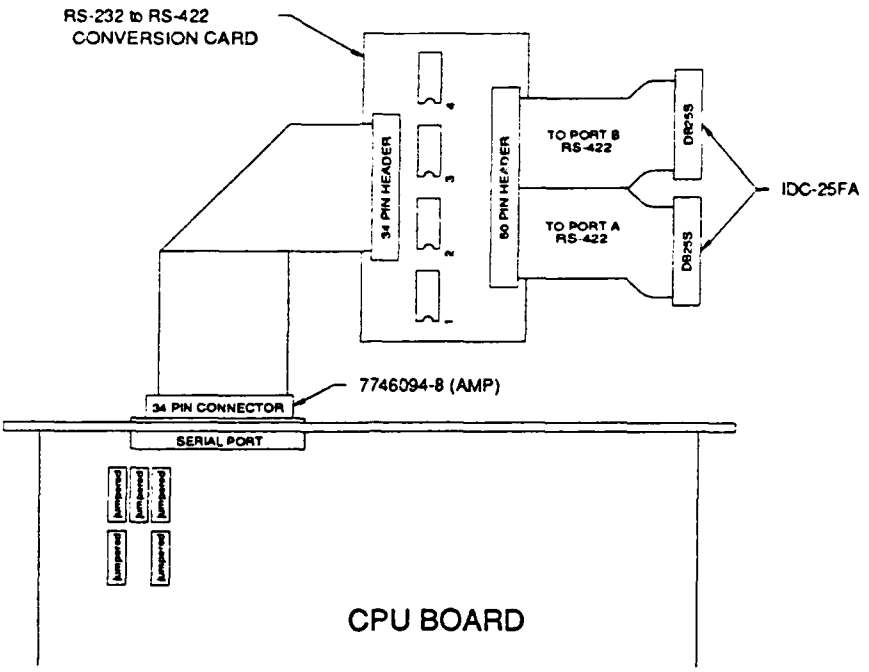


Figure 7-8. Ports A and B converted to RS-422/449 interface

7.8.1.1.1. Removing/Replacing Chips

To remove the interface chips:

1. Take a knife blade or flat-bladed screw driver and carefully lift up on one end of the chip and then the other, rocking gently until the chip is free of its socket.
   - Notice that both the chip and the socket have a semicircular-shaped notch at one end.
   - Always make sure the notch on the chip is positioned above the notch on the socket before replacing a chip.

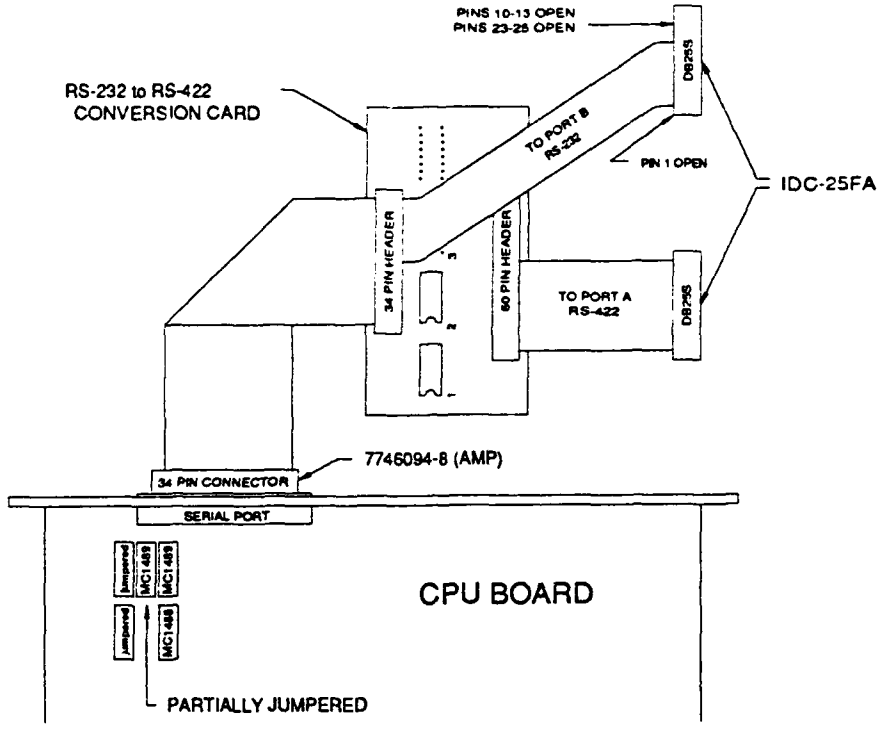


Figure 7-9. Port A only converted to RS-422/449 interface

To insert the jumper chips:

1. Place them in the proper orientation above the socket.
2. Align the pins on one side of the chip with the appropriate holes on the socket, and then place thumbs along that side of chip.
3. Using forefingers, gently pinch the pins on the other side, easing them into their proper positions in the socket.
4. Once the chip is in place, press gently but firmly on the chip to make sure it is properly seated.

7.8.1.2. Setting DIP Switches

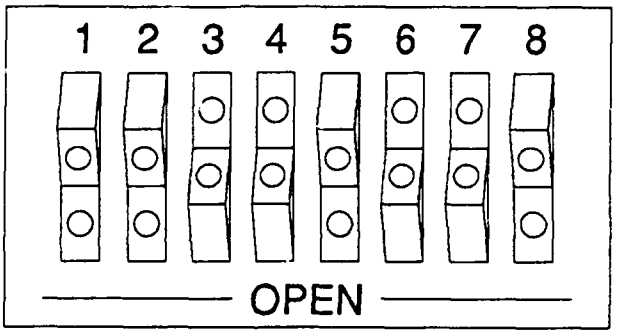
DIP switches on a replacement board should be set exactly like those on the original board. To double check the setting:

- Check the 7-digit binary number on the back edge of the original board.

  NOTE: The left or most significant digit (MSD) on the label corresponds to switch 7, while the right or least significant digit (LSD) corresponds to switch 1.

- Set DIP switches 1 through 7 on the new board to correspond to the digits on the label, where a **1** is closed and a **0** is open.
- Set switch number 8 to open for normal operation.

Figure 7-10 shows the proper switch settings for a label that reads **1101100**.



BOARD LABEL: 1101100
LSD maps to DIP SWITCH 1
MSD maps to DIP SWITCH 7
0 = OPEN 1 = CLOSED

Figure 7-10. DIP switch settings

7.8.1.3. Master Board Jumpers

The master CPU board is the board in slot one (bottom) of the CIM card cage. The system controller functions that handle the bus clock and arbitration logic are provided by the on-board SELF ARBITER and SYSCLOCK jumpers, J21 and J22 respectively. These jumpers are installed *only* on the CIM master board. If these positions are not jumpered on the board being replaced, leave them open on the replacement board.

7.8.2. Memory Board Configuration

The only necessary configuration of a replacement memory board is that of the starting address jumpers and Control and Status Register (CSR) address jumpers. All other jumpers are pre-set and should not be modified. The location of the address jumpers, numbered E1-E16, is illustrated in Figure 7-16.

As with the CPU boards, a replacement memory board must be configured exactly like the original board and put in the same card-cage slot where the original was housed. As a check on the slot order and board configuration, Table 7-2 lists the memory address associated with each of the memory address jumpers E01 through E16. All 16 jumpers must be set because they are used in combination. The starting address of the 4MB increments of memory and corresponding jumper up/down settings are listed in Table 7-3. The settings are shown in the same right to left order as the jumpers appear on the board.

Jumpers E20-E23 are used in combination to configure the CSR address. On the first memory board (bottom of the memory board stack), the CSR is set to the lowest of the 16 possible addresses (0xFEE0), the next board is set to 0xFEE2, etc. The CSR addresses for additional boards increase sequentially up to 0xFEFE. Table 7-4 shows the 16 CSR address/jumper configurations.

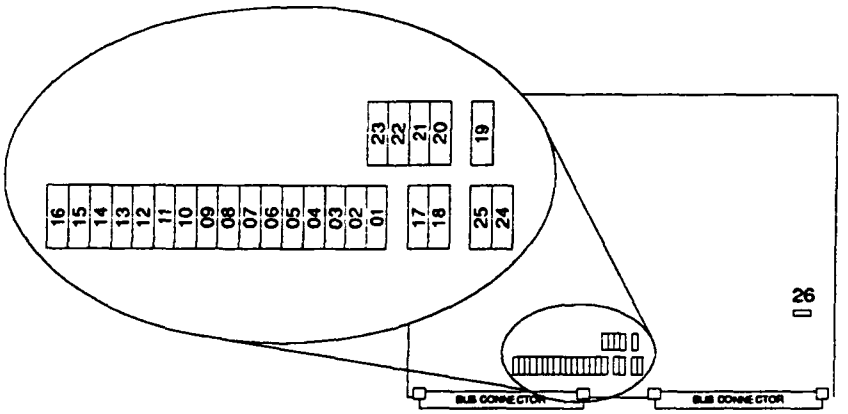


Figure 7-16. VMERAM Jumper Location

Table 7-2. VMERAM Address Jumpers

Jumper No.	Value		
E01	2^{16}	=	64KB
E02	2^{17}	=	128KB
E03	2^{18}	=	256KB
E04	2^{19}	=	512KB
E05	2^{20}	=	1MB
E06	2^{21}	=	2MB
E07	2^{22}	=	4MB
E08	2^{23}	=	8MB
E09	2^{24}	=	16MB
E10	2^{25}	=	32MB
E11	2^{26}	=	64MB
E12	2^{27}	=	128MB
E13	2^{28}	=	256MB
E14	2^{29}	=	512MB
E15	2^{30}	=	1GB
E16	2^{31}	=	2GB

Table 7-3. VMERAM Address/Jumper Configuration

Start Address	Jumper Settings E16->E01	Slot No.
0x100000 (MB)	ddddddddddudddd	5
0x500000 (5MB)	ddddddddduddddd	6
0x900000 (9MB)	ddddddddudddddd	7

Where:

d = down

u = up

Table 7-4. Memory CSR Address Jumper Configurations

Absolute Address	Short I/O Address	Jumper E23	E22	E21	E20
(FF)FFFEE0	FEE0	down	down	down	down
(FF)FFFEE2	FEE2	up	down	down	down
(FF)FFFEE4	FEE4	down	up	down	down
(FF)FFFEE6	FEE6	up	up	down	down
(FF)FFFEE8	FEE8	down	down	up	down
(FF)FFFEEA	FEEA	up	down	up	down
(FF)FFFEEC	FEEC	down	up	up	down
(FF)FFFEEE	FEEE	up	up	up	down
(FF)FFFEF0	FEFE0	down	down	down	up
(FF)FFFEF2	FEFE2	up	down	down	up
(FF)FFFEF4	FEFE4	down	up	down	up
(FF)FFFEF6	FEFE6	up	up	down	up
(FF)FFFEF8	FEFE8	down	down	up	up
(FF)FFFEFA	FEFEA	up	down	up	up
(FF)FFFEFC	FEFEC	down	up	up	up
(FF)FFFEFE	FEFEE	up	up	up	up

7.9. Backplane Configuration

The backplane of the CIM enclosure is configured with jumper blocks at each slot of the card cage to facilitate jumpering of the IACK and Bus Grant signals. A CIM hat has been functioning properly should not require changes to its backplane jumpers.

Figure 7-12 illustrates all six of the jumper blocks, labeled A01 IACK through A06 IACK and A01 BUS GRANT through A06 BUS GRANT. In the figure, jumpered slots are depicted by small rectangles representing the jumpers, while unjumpered slots are depicted by pairs of dots representing pins. The illustration shows a configuration in which cards that are potential VMEbus masters are installed only in slots A01 and A02. Therefore, only the jumpers labeled A01 BUS GRANT, A01 IACK, and A02 BUS GRANT and A01 IACK are removed.

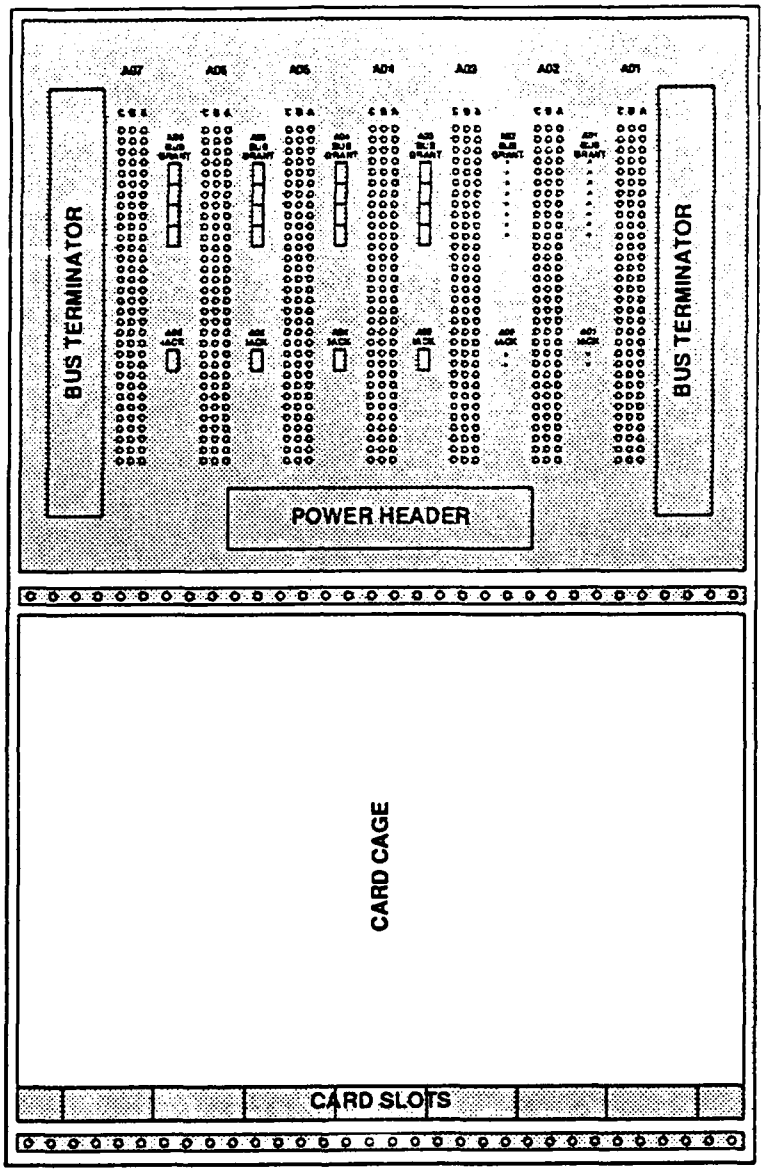


Figure 7-12. Backplane with Jumpers

7.10. Component Return

Facilities Outside the United States:

If you need to return any faulty component to Science Horizons for repair, be sure to prepare a packing slip with the proper information, as follows:

- List all items that are being returned by name and serial number (*i.e.*, microprocessor board, serial #_____).
- Include the following declaration:

  "These items are of United States origin and were previously exported from the United States on the validated export license no. _____ on (date)_____ via (shipper)_____."

To be sure of the exact information and wording, check with Science Horizons when you arrange for the repairs.

INDEX