CONVEX Multibus Plotter (*dev4410*) Diagnostics Manual

Document No. 760-002430-000

First Edition
May 1991

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX Multibus Plotter (dev4410) Diagnostics Manual
Order No. DHW-236
First Edition

© 1991 CONVEX Computer Corporation
All rights reserved.

This document is copyrighted. All rights reserved. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored or reduced to machine readable form without prior written consent from CONVEX Computer Corporation (CONVEX).

Although the material contained herein has been carefully reviewed, CONVEX does not warrant it to be free of errors or omissions. CONVEX reserves the right to make corrections, updates, revisions, or changes to the information contained herein. CONVEX does not warrant the material described herein to be free of patent infringement.

UNLESS PROVIDED OTHERWISE IN WRITING WITH CONVEX COMPUTER CORPORATION (CONVEX), THE EQUIPMENT DESCRIBED HEREIN IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES. THE ABOVE EXCLUSION MAY NOT BE APPLICABLE TO ALL PURCHASERS BECAUSE WARRANTY RIGHTS CAN VARY FROM STATE TO STATE. IN NO EVENT WILL CONVEX BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OR INABILITY TO USE THIS EQUIPMENT. CONVEX WILL NOT BE LIABLE EVEN IF IT HAS BEEN NOTIFIED OF THE POSSIBILITY OF SUCH DAMAGE BY THE PURCHASER OR ANY THIRD PARTY.

CONVEX and the CONVEX logo ("C") are registered trademarks of CONVEX Computer Corporation
C1, C120, C201, C202, C210, C220, C230 and C240 are trademarks of CONVEX Computer Corporation
C100 Series and C200 Series are trademarks of CONVEX Computer Corporation
UNIX is a registered trademark of AT&T Bell Laboratories
ConvexOS is a registered trademark of CONVEX Computer Corporation

Printed in the United States of America

Revision Sheet

CONVEX Multibus Plotter
(dev4410) Diagnostics Manual

Edition	Document No.	Date	Description
First	760-002430-000	May 1991	First release. Contains the *dev4410* diagnostic test information from the *CONVEX PBUS I/O Systems Diagnostics Manual.*

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

Appendixes

A Reporting Problems

List of Tables

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Purpose and Intended Audience

This manual explains how to run the *dev4410* diagnostic, which checks the IKON Versatec plotter controller, model 10085, and any attached Versatec plotter. This document is not a tutorial, but rather a reference for the users of the *dev4410* diagnostics, including field service and manufacturing test personnel, as well as the diagnostics sustaining staff. In addition, CONVEX customers can use this manual to execute the *dev4410* diagnostic.

Scope

This manual applies to all CONVEX computers.

Organization

This document consists of the following:

- **Chapter 1. Diagnostics Environment**—Introduces the theories and concepts that underlie I/O diagnostics on CONVEX machines as well as the basic overview, philosophy, and structure of I/O diagnostics.
- **Chapter 2. EGOS Overview**—Provides a brief overview of the Event Governed Operating System (EGOS) and how it relates to device and peripheral diagnostics testing.
- **Chapter 3. Dshell Overview**—Provides a brief overview of and a general introduction to the *dshell* utility.
- **Chapter 4. Multibus Plotter Test (*dev4410*)**—Describes how to operate the diagnostic, including prerequisites, test invocation, hardware initialization sequence, and class descriptions. It also describes error messages produced by the diagnostic.
- **Appendix A. Reporting Problems**—Provides an example of the CONVEX ***contact*** utility for reporting minor software and hardware problems.

Notational Conventions

The notational conventions used in this text are listed below:

- Bit numbering is left to right, `N-1` through `0`. The most significant numerical bit is `N-1`, the least significant `0`. The bit numbering represents the binary weight of a position.
- Bit fields are specified using the following convention: *name*<*x..y*> where the bit field is *name* from bits *x* through *y*.
- Individual bit positions within a register are denoted by specific positions separated by commas. For example, REG<15,4,0> denotes bits 15, 4, and 0 of REG.
- Byte numbering is from left to right
- A *bit* is a single binary value or entity
- A *nibble* is 4 bits
- A *byte* is 8 bits
- A *halfword* is 16 bits
- A *word* is 32 bits
- A *longword* is 64 bits
- *Single precision* is a 32-bit floating point word
- *Double precision* is a 64-bit floating point longword
- An *instruction* is a multihalfword operand
- A bit is *set* when it contains a binary value of `1`.
- A bit is *clear* when it contains a binary value of `0`.
- All memory and I/O addresses are written in hexadecimal notation unless explicitly stated otherwise.
- All register contents are written in hexadecimal notation unless explicitly stated otherwise.
- A *register* is a programmer-visible hardware storage element internal to the processor
- *Physical memory* is the physical storage installed in the processor
- *Virtual memory* is the perceived amount of physical memory as seen by the application programmer
- The symbol *K* is an abbreviation for *kilo* or 1,024
- The symbol *M* is an abbreviation for *mega* or 1,048,576
- The symbol *G* is an abbreviation for *giga* or 1,073,741,824
- A *stack* is a linked-list group of words useful for dynamic allocation and deallocation of memory
- A *return block* is a collection of registers that is pushed or popped from a context stack in response to an instruction or other event
- *Reserved* or *undefined* convey what to expect, if anything, from unused fields in registers, reserved memory, or reserved I/O space. Algorithm implementation based on the use of undefined or reserved fields is not recommended.

Warnings

The following are examples of warnings, cautions, and notes and their typical content as used in CONVEX documents:

WARNING

Warnings highlight procedures or information necessary to avoid injury to personnel. A warning immediately precedes the critical information and includes a description of the hazard.

CAUTION

Cautions highlight procedures or information necessary to avoid damage to equipment, loss of data, or invalid test results. A caution immediately precedes the critical information and includes a description of the possible damage.

NOTE

Notes highlight useful information that is supplemental in nature. A note may immediately precede or follow the information that is being highlighted.

Associated Documents

The following is a partial list of other manuals or books that may provide more detailed information on the topics presented in this manual:

- *CONVEX Processor Diagnostics Manual (C1, C120)*, Order No. DHW-071
- *CONVEX Processor Diagnostics Manual (C200 Series)*, Order No. DHW-081
- *CONVEX Architecture Reference*, Order No. DHW-005
- *CONVEX SPU UNIX Utilities Manual*, Order No. DHW-021
- *CONVEX Processor Operation Guide (C100 Series, C200 Series)*, Order No. DHW-015
- *CONVEX Diagnostic Utilities Manual (C1, C120)*, Order No. DHW-072
- *CONVEX Diagnostic Utilities Manual (C200 Series)*, Order No. DHW-082
- *CONVEX UNIX Tutorial Papers*, Order No. DSW-002
- *The C Programming Language*, Kernighan & Ritchie, Order No. DSW-046

Ordering Documentation

To order the most current version of this or any other CONVEX document, use the product number. If the product number is not known, order by the exact title. In some situations, the most current version may not be desired. To receive a specific version of a manual, order the manual by its document number, or part number, which can be obtained by contacting the local CONVEX office or by calling the Technical Assistance Center.

The product number for this manual is DHW-236.
The document number for this manual is 760-002430-000.

CONVEX documents can be ordered by mail by sending a request to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

Technical Assistance

Hardware and software support can be obtained through the CONVEX Technical Assistance Center (TAC):

- From all locations in the continental United States, call 1(800)952-0379.
- From locations in Alaska, Hawaii, and Canada, call 1(214)497-4379.
- From all other locations, contact the nearest CONVEX office.

Reader's Forum

If you wish to mail your comments to us, please use the form at the end of this manual and list the document page number with your questions and comments. Thank you.

Chapter 1
Diagnostics Environment

1.1 Overview

CONVEX system diagnostics consist of a suite of test programs designed (except where noted) to execute under the Service Processor operating system, SPU UNIX. These programs utilize the capabilities of the Service Processor to test the operation of one or more of the functions of the system and report any errors detected. All of the diagnostics in this manual are intended to be executed "off-line"; that is, while CONVEX UNIX is not being executed by any of the Central Processing Units (CPUs) in the system.

The Service Processor, together with SPU UNIX, various diagnostic utilities, and the test programs, themselves, comprise the CONVEX diagnostic environment. This chapter describes the hardware and software components of this environment and is intended to provide the background necessary to fully utilize the capabilities of the CONVEX processor diagnostics.

For more information about the diagnostic environment refer to the Diagnostic Environment chapter in the *CONVEX Processor Diagnostics Manual (C200 Series)* or the *CONVEX Processor Diagnostics Manual (C1, C120)* depending on the architecture of the machine under test.

1.2 Test Program Naming Conventions

Test program names are in the form *cattypedevnn.suffix* where:

- *cat* is the subsystem being tested
- *type* is the type of test being performed, e.g., standalone, self-test, or offline functional test
- *dev* is the device being tested, e.g., disk, tape, or printer. This segment of the test program name is used *only* if the category is a device.
- *nn* is a CONVEX code used for distinguishing between test programs
- *suffix* is one of three program identifiers:
  - *.t* are programs that execute on SP2
  - *.x00* and *.rnn* are object files for different target processors other than the SP2. The target processor depends on the subject of the test. The test program name must have the test program category (*cat*) at the beginning of the name to determine the target processor.

1.2.1 Test Program Categories

Test program categories include those tests for the CPU, peripheral devices, I/O system, memory system, SP2, and entire system. For example, *cpu4041* is a CPU vector instruction test while *mem4000* is a memory system functional test. The following table lists test program categories:

Table 1–1, Test Program Categories

TEST PROGRAM CATEGORIES	
Test Category (*cat*)	**Description**
cpu	CPU subsystem related test
dev	Peripheral device test
io, idc, tli	I/O subsystem related test
mem	Memory subsystem related test
spu	SP2 subsystem related test

1.2.2 Test Program Types

A test program type describes whether a test is a standalone test, self-test, kernel hardware test, or an offline or online functional test. See the following table for the numbering system and description of test program types:

Table 1–2, Test Program Types

TEST PROGRAM TYPES	
Number (*type*)	**Description**
0	Standalone test
1	Self-test
2	Kernel hardware test
4, 5	Offline functional test

1.2.3 Test Program Device Types

Test programs will test disks, tapes, terminals, printers, and networks. See the following table for the numbering scheme and a description of the test program device types:

Table 1–3, Test Program Device Types

TEST PROGRAM DEVICE TYPES	
Number (*dev*)	**Description**
1	Disk
2	Tape
3	Terminal
4	Printer
5	Network

1.2.4 Examples of Test Program Names

The following table presents some examples using the naming conventions outlined above:

> NOTE
>
> In the following table, SOFF stands for Standard Object File Format.

Table 1–4, Example Test Program Names

EXAMPLE TEST PROGRAM NAMES	
Test Program Name	**Description**
cpu4041.t	SP2 object code in *b.out* format for *cpu4041*
cpu4041.rnn	C210 or C220 machine object code in SOFF format (relocatable)
cpu4041.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
mem4000.t	SP2 object code in *b.out* format for *mem4000*
mem4000.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
dev4100.t	SP2 object code in *b.out* format for *dev4100*
dev4100.x00	IOP object code in *b.out* format

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2
EGOS Overview

2.1 Overview

This chapter provides an overview of the Event Governed Operating System (EGOS) and how it relates to device and peripheral diagnostics testing. There are three basic types of EGOS systems, one for each type of CCU. There is one for the Multibus interface, one for the VME interface, and one for the HIA interface. This chapter will explain the three types of EGOS systems and how EGOS is positioned within the overall operating system environment.

2.2 Purpose of EGOS for Diagnostic Testing

EGOS is basically a simple operating system that the device tests use to handle interrupts, schedule processes, and generally allocate and control IOP/VIOP resources. The diagnostics code uses both EGOS and the Message Based System (MBS) to manipulate test program control over to the CCU side of the test program. MBS is not a part of EGOS but rather a system that allows a common section of memory to be used as a message area between multiple processors. For more information on MBS, refer to the *CONVEX Guide to Writing Device Drivers.*

EGOS initially sets up interrupt tables, determines how many chassis there are, and initializes its windows and resource allocation tables.

2.3 EGOS for the Multibus Interface

EGOS for the Multibus interface supports event driven device drivers. The Multibus version of EGOS takes interrupts that are local to a CCU and channels those errors to the proper piece of code to handle the error. It basically supplies the error interrupt handlers for the CCU error interrupts. It also contains support routines to control allocation of the various CCU-related resources.

2.4 EGOS for HSP Interface, HSP EGOS

EGOS for the HSP interface supports event driven device drivers. The HSP version of EGOS is like the Multibus version. It takes interrupts that are local to a CCU and channels those errors to the proper piece of code to handle the error. It basically supplies the error interrupt handlers for the CCU error interrupts. It also contains support routines to control allocation of the various CCU-related resources.

2.5 EGOS for VME Interface, VIOP EGOS

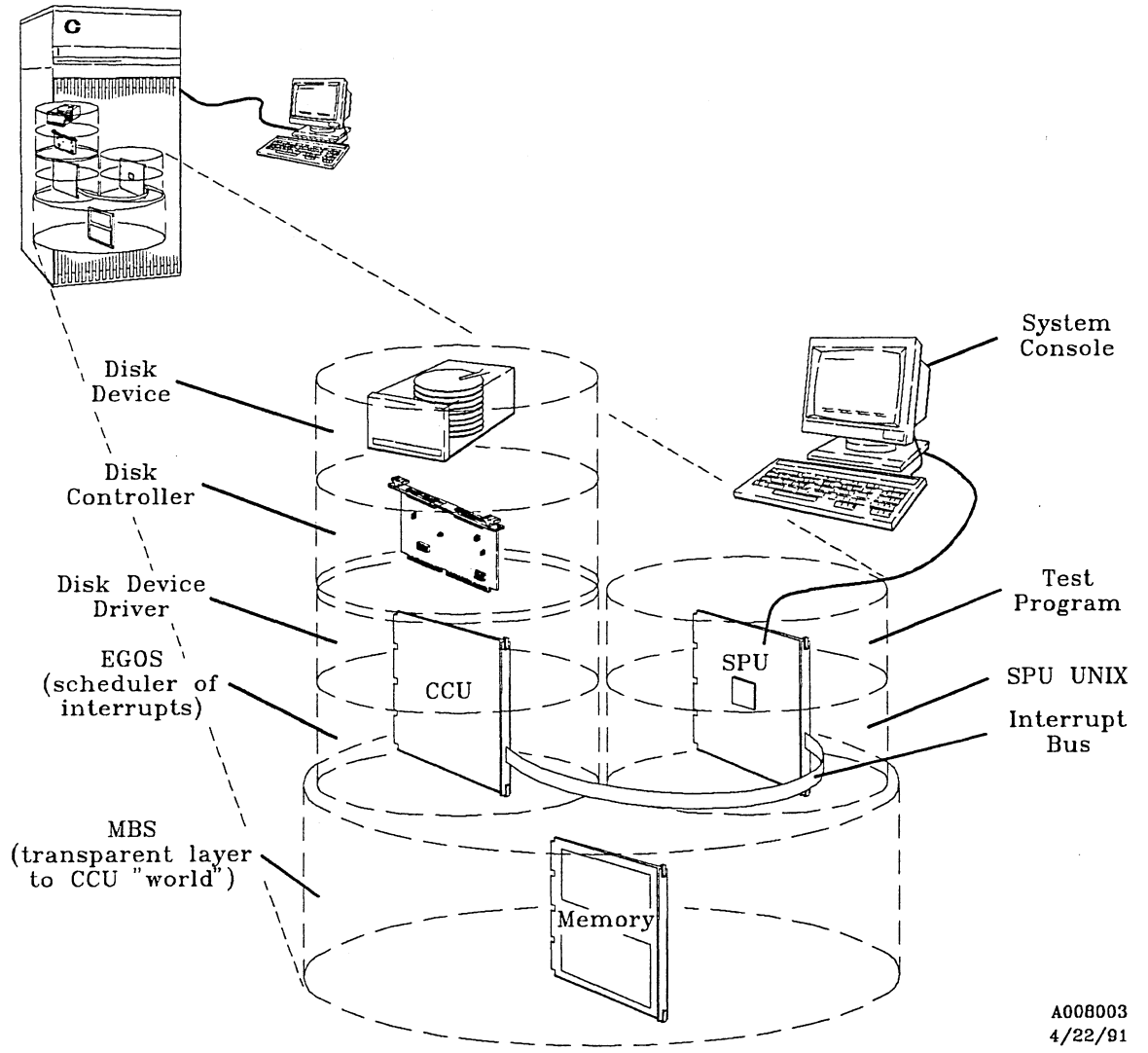
The VME interface version of EGOS is designed with a scheduler for the VIOP and is called VIOP EGOS. VIOP EGOS supports event driven device drivers as well as process type device drivers. VIOP EGOS utilizes a *sleep/wakeup* type of process control that improves efficiency of the device driver and makes it less complicated to create user written device drivers. Each process device driver has a priority level that can be defined relative to other processes. The scheduler supports 32 process priorities and is preemptive for higher priority processes. The VIOP hardware supports 14 device events for event driven device drivers. The 14 levels actually share 2 68020 interrupt levels. Therefore, two is the maximum number of processes at any given time.

2.6 EGOS Position in the Environment

EGOS is positioned in the operating environment between the actual device driver and MBS. MBS is a transparent layer that bridges the CCU and its resources to SPU UNIX. SPU UNIX handles many of the message manipulations that occur during testing. Many error messages that occur during diagnostics testing come from the device driver. When the device driver detects an error from the controller, it calls a routine in EGOS that places a message in the MBS system. This causes SPU UNIX to be interrupted and it retrieves the message from MBS. SPU UNIX then passes a signal to the test program. The test program then prints an error message to the console based on the code that it received.

The following figure illustrates the position of EGOS in the operating system environment.

Figure 2–1, EGOS' Position in the Environment



THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3
Dshell Overview

3.1 Overview

This chapter provides a brief overview of the *dshell* utility. Included in this overview is an overall explanation of the utility and a list of the utility's commands. For a complete description of this utility, refer to the Dshell chapter of the *CONVEX Diagnostic Utilities Manual (C200 Series)* or the *CONVEX Diagnostic Utilities Manual (C1, C120)* depending on the architecture of the machine under test.

3.2 Diagnostic Shell (*dshell*) Overview

The Diagnostic Shell (*dshell*) is a command interface program that runs on the Service Processor. Most of the diagnostics available for the CONVEX machines are interfaced through the *dshell*. Certain peripheral diagnostics are run as standalone tests. To determine whether a test can be run under the *dshell*, consult the appropriate chapter in this manual.

The *dshell* has two basic functions:

- Selecting diagnostics for execution
- Selecting test options
  - Pause on a failure or at the beginning or end of any specific subtest
  - Loop on a specific type of subtest or on a given set of subtests
  - Select subtest execution order
  - Direct test output to a file or to the screen (or both) to monitor the test as it runs or to analyze test results later
  - Select long or short error messages, or turn messages off
  - Execute either user-created or predefined command scripts

The following table list the various *dshell* commands and their functions.

Table 3–1, *dshell* Commands

COMMAND	FUNCTION
! [command]	This command is used to access, or *fork* a UNIX shell to execute the command that follows *!*.
exit	The *exit* command causes immediate termination of the *dshell* process and any test processes that may have been forked.
quit	The *quit* command causes immediate termination of the *dshell* process and any test processes that may have been forked.
^C	Returns user to the *dshell* command level if no subtest is running.
^B	Immediately terminate the *dshell* and any associated active processes. Core is dumped.
help	The *help* command causes a standard *help* menu to be displayed. The menu describes the correct command syntax for each *dshell* command and gives a terse description of what each command does.
status	The *status* command generates a report on the current state of the *dshell* command options. This report gives the name of each flag, its current value, and an explanation of its current effect.
log [options]	The *log* command provides a mechanism for specifying the number of failures that will be allowed to occur before a test or subtest terminates execution.
loop [options]	The *loop* command causes the *dshell* to repeat the execution of a test or subtest.
msgs [options]	The *msgs* command enables or disables different levels of test, class, and subtest result messages.
pause [options]	The *pause* command returns program control to the *dshell* to the beginning, end, or failure of all or specific subtests.
test [options]	The *test* executes specific tests, and displays test, class, and subtest menus.

3.3 Syntax Help for *dshell* Commands

The syntax for each *dshell* command can be obtained by typing the command with no options and pressing <**CR**>. For example, by entering **loop** and pressing <**CR**>, the syntax help in the following figure will be displayed on the screen:

Figure 3–1, Syntax Help for the *loop* Command

```
: loop
Proper syntax is:

loop off (-s) (-t)                          :disables loop modes
loop -s nnn                                 :loop on subtest nnn
loop -t                                     :loop on test
```

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4
Multibus Plotter Test (*dev4410*)

4.1 Overview

The *dev4410* test checks the IKON Versatec plotter controller, model 10085, and any attached Versatec plotter.

4.2 Prerequisites and Required Equipment

The following table lists the required hardware depending on the type of machine under test.

Table 4–1, Hardware Requirements

C1, C120	C200 Series
MCU	Memory System[1]
MAU	MAU
SPU	SP2
IOP	IOP
MBCU	MBCU
	PIA

[1] Memory System consists of a minimum of one pair of memory boards (one odd and one even).

If a plotter is not available, Class 1 subtests can be run as a check on the IKON controller only. If **y** is entered to the `Test Parameter Instructions` prompt in the test parameter menu, this procedure is described.

4.3 Test Invocation

The *dev4410* test executes under the Diagnostic Shell (*dshell*) and supports all the features of the *dshell*. The *dshell* permits tests to be initiated in any order.

To invoke the *dev4410* test, use the procedure shown in the following figure. All responses in **boldface** are entered by the user. The prompts and responses appear sequentially on the screen, one line at a time. All prompts and responses are shown in one figure for convenience.

Figure 4–1, Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> sysreset RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
: test dev4410 [-c [class number(s)]] [-s [subtest number(s)]] [+> filename] RETURN
```

NOTE

After entering **dshell**, specific *dshell* parameters may be changed. Refer to the "Dshell Overview" chapter of this manual for more information.

Entering only **test dev4410** executes all *dev4410* subtests sequentially. Execute a specific class(es) of subtest(s) or one or more individual subtests by using the **-c** or **-s** options, respectively. Detailed information for using these options can be found in the "Dshell Overview" chapter of this manual. The [+>*filename*] option allows the test results to be appended to *filename*.

The following alternate test invocation procedure may be required in some cases.

CAUTION

The user response, **initall**, is typically required if the *initall* utility has not been run since the last power up. However, if any problems have occurred subsequent to the last time *initall* was run, (i.e., system crash, hard error, or failure of previous diagnostic), it should be run again. In this case, failure to run *initall* could result in invalid test results.

NOTE

The *initall* utility requires a significant amount of time (2 to 3 minutes depending on whether the control stores have been previously loaded) to execute. If no system abnormalities have occurred subsequent to the last time the system was booted or *initall* was executed, it is not necessary to run *initall*.

Figure 4–2, Alternate Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> initall RETURN
(spu)> dshell RETURN
: test dev4410 [-c [class number(s)]] [-s [subtest number(s)]] [+> filename] RETURN
```

4.3.1 Test Parameter Menu

Once the test is invoked, a test menu prompt is presented allowing selection of default switches. The following figure shows all prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )). The prompts and responses in the following figure appear sequentially on the screen, one line at a time. All the prompts and responses are shown in one figure for convenience.

The **Test Parameter Menu** illustrates *all* questions that can be displayed during test parameter input. However, some questions may be omitted, depending on answers to previous questions. In all cases, questions are numbered sequentially. However, the numbers displayed on the screen during testing may not correspond to those shown in the example **Test Parameter Menu**, as the questions illustrated are examples only.

Figure 4–3, Test Parameter Menu

```
                    ENTER TEST PARAMETERS

    []          Encloses allowed input ranges or values
    ()          Encloses the default value
    ^           Returns to the previous prompt
    :nn         Returns to the prompt # nn
    :           Returns to the first unsatisfied prompt
    :?          Reviews previous entries

                    PERIPHERAL CONFIGURATION DATA
     CCU        Chassis Type     CSR      Int      Unit     Type
    -------     ------- -------  -----    ---      ----     --------
 1)  iop 3        0     PRC-002 0x2c0     1

Device 0 = user defined configuration

 1: Device Selection [0,1]                                    (0) ->
 2: IOP [3-7][1]                                              (5) ->
 3: Multibus Chassis [0-3]                                    (0) ->
 4: Controller Offset in Multibus [0x0-0xfff]              (0x2c0) ->
 5: Interrupt Number [0-7]                                    (1) ->
 6: Test Parameter Instructions [y,n]                         (n) ->
 7: Is IKON set for TTL interface [y,n]                       (n) ->
 8: Is IKON set for forced byte mode [y,n]                    (n) ->
 9: Suppress manual intervention [y,n]                        (n) ->

     Plotter models

     0: USER DEFINED                        12: 2000,2030   >64 char
     1:   80                                13: 2160
     2: 200A                 64 char        14: 8122
     3: 200A                >64 char        15: 8222
     4:  800                 64 char        16: 8124
     5:  800                >64 char        17: 8224
     6:  900                                18: 8136
     7: 1100                 64 char        19: 8236
     8: 1100                >64 char        20: 8124
     9: 1200                                21: 8242
    10: 1600                                22: 8172
    11: 2000,2030            64 char

10: Plotter model [0, 1-22]                                   (1) ->
11: Print characters per line [1-1000]                        (0) ->
12: SPP Print chars per line [1-1000]                         (0) ->
13: SPP scan lines for character [1-1000]                     (0) ->
14: Plot bytes per line [1-1000]                              (0) ->
15: Can this plotter print [y,n]                              (y) ->
16: Test ASCII print controls [y,n]                           (y) ->
17: Can this plotter plot [y,n]                               (n) ->
18: Does this plotter have shared print/plot mode
    [y,n]                                                     (y) ->
19: Shorten programmed output plot patterns [y,n]             (n) ->
20: Shorten DMA output plot patterns [y,n]                    (y) ->
21: Self test instructions [y,n]                              (n) ->
22: Enter OK, or :NN to return to question NN [OK]
                                                           (OK) ->
```

[1] The possible selections for this prompt will change depending on machine architecture.

At any time during the test parameter sequence, several options are available as denoted at the top of the Test Parameter Menu. The following list summarizes the available options:

:nn — Returns to an earlier prompt (n is the prompt number)

: — Advances to the next unanswered prompt

:? — Displays (reviews) all responses up to the current prompt

? — Requests help for the current prompt (if available)

^ — Returns to the previous prompt

4.3.2 Prompt Explanations

A description of the meaning of each prompt follows:

```
Device Selection [0,1]                                    (0) ->
```

This prompt selects a device for testing from the system configuration. If the **0** (default) is entered, then user-defined prompts are displayed to determine the configuration to be tested; otherwise, if a device is selected, the prompts are not displayed.

```
IOP [3-7]                                                 (5) ->
```

Enter the CCU slot number for the desired IOP. Then additional prompts are displayed requesting hardware configuration information.

```
Multibus Chassis [0-3]                                    (0) ->
```

Enter the number of the chassis to be tested. (The drive is attached to a controller which is in a Multibus chassis.)

```
Controller Offset in Multibus [0x0-0xfff]                 (0x2c0) ->
```

Enter the low-order 12 bits of the controller's address within the Multibus (this address is selected with switches on the controller).

```
Interrupt Number [0-7]                                    (1) ->
```

Enter the interrupt level of the controller within the Multibus (this is selected with switches on the controller).

The following prompts display one line at a time after the configuration to be tested has been selected:

```
Test Parameter Instructions [y,n]                         (n) ->
```

This prompt allows for the display of a short help file containing information about test strategy and the test parameters.

```
Is IKON set for TTL interface [y,n]                    (n) ->
```

Enter **y** to for TTL (Transistor-Transistor Logic) mode. Most Class 1 subtests (if selected) are skipped unless TTL mode is selected.

```
Is IKON set for forced byte mode [y,n]                 (n) ->
```

Enter either forced-byte mode or word-mode configuration for the IKON. Generally, word mode is selected.

```
Suppress manual intervention [y,n]                     (n) ->
```

If **y** is entered, Class 1 subtests, which require the plotter to be disconnected, are skipped. In addition, Class 3 subtests (plotter offline and out-of-paper subtests) are skipped. If **n** is entered, instructions such as:

```
Disconnect or reconnect plotter
Put plotter offline or online
Remove or restore plotter paper
```

are displayed when manual functions are performed.

```
Plotter model [0, 1-22]                                (1) ->
```

If **0** (user-defined plotters) is entered, prompts are displayed for specifying the characteristics of the plotter being used. Entering a value in the range [1-22] causes the user-defined questions to be skipped.

The following four prompts are displayed for user-defined plotters.

```
Print characters per line [1-1000]                     (0) ->
```

Enter the number of characters to print per line.

```
SPP Print chars per line [1-1000]                      (0) ->
```

Enter the number of characters that the plotter can print in a single line when in shared print-plot mode (SPP).

```
SPP scan lines for character [1-1000]                  (0) ->
```

Enter the number of plot scans required to get the entire character block on the output medium.

```
Plot bytes per line [1-1000]                           (0) ->
```

Enter the number of bytes that can be plotted in a single-plot scan on the plotter under test.

```
Can this plotter print [y,n]                           (y) ->
```

Enter whether the plotter being tested has print capability.

```
Test ASCII print controls [y,n]                        (y) ->
```

This prompt is only displayed if **y** is entered to the previous prompt. If the printer has the capability to handle ASCII control characters, enter **y**.

```
Can this plotter plot [y,n]                                    (n) ->
```

If the plotter being tested has plot capability, enter **y**; otherwise, press CTRL for the default.

```
Does this plotter have shared print/plot mode
[y,n]                                                          (y) ->
```

This prompt specifies whether the plotter has shared print-plot mode (print and plot can be overlaid).

```
Shorten programmed output plot patterns [y,n]                  (n) ->
Shorten DMA output plot patterns [y,n]                         (y) ->
```

These prompts allow for shortening the plot-pattern generation. Normally, the full pattern is plotted in programmed output mode, and a short set is plotted in Direct Memory Access (DMA) mode.

```
Self test instructions [y,n]                                   (n) ->
```

Enter **y** to display a help file giving instructions for operating the IKON self-test.

```
Enter OK, or :NN to return to question NN [OK]                 (OK) ->
```

If **OK** or RETURN is entered, the test parameter menu terminates and all inputs are no longer changeable.

After entering all the parameters, the test parameter selections are echoed to the screen. If standard output is directed to a disk file, the test parameter summary is also directed to the disk file.

4.4 Hardware Initialization Sequence

After the last prompt is entered, and before test code execution, the following events occur:

- A sysreset is performed
- Main memory is allocated for the test
- SPU windows to main memory are initialized
- SPU local test variables are initialized
- The IOP is booted and loaded
- A driver on the IOP is started
- IOP local test variables are initialized

After all the above events have occurred, the test code is started.

4.5 Class Descriptions

The *dev4410* test contains the following three classes of subtests as shown in the following table.

Table 4–2, *dev4410* Test Classes

CLASS	DESCRIPTION
1	IKON 10085 controller and loopback tests
2	IKON/Versatec plotter status/interrupt/pattern tests
3	IKON/Versatec plotter offline/no-paper tests

> **NOTE**
>
> The times shown are for the full subtest only (not the shortened version).

4.5.1 Class 1 Subtests

Class 1 subtests verify some of the basic functionality of the controller. Class 1 subtests include board reset and loopback subtests.

Class 1 subtests are listed in the following table:

Table 4–3, Class 1 Subtests

SUBTEST	DESCRIPTION	TIME (min:sec)
100	IKON Reset Capability	1:40
101	IKON Command Loopback	1:40
102	IKON Plotter Exception Loopback	:26
103	IKON Port Selection	1:40
104	IKON Plotter Mode Selection	1:40
105	IKON Programmed Output Loopback	:11
106	IKON DMA Output Loopback	2:00
107	IKON Data Output Interrupt Capability	:01

Standard ASCII printables are used for print patterns. Plotters may have special characters available for use that are normally considered ASCII control characters. Since some of these characters execute control functions that vary according to user specification, all tests avoid control characters, except for carriage return, newline, form feed, and end-of-transmission. If possible, use a plotter self-test to see all printable characters.

The following describes those items not formally tested:

- Versatec 8xxx series remote toner-on and toner-off commands
- Versatec 8xxx series status signals for low toner supply, toner-on/off status, and data transfer errors
- Versatec 8xxx series ASCII control characters (in print mode) which may be used to turn the toner subsystem on and off
- Serial interface plotters
- ASCII DC1 capability to switch to plot mode
- Option (Centronics) port on the IKON controller

4.5.1.1 Subtest 100, IKON Reset Capability

Subtest 100 assures that the controller resets properly. This subtest checks the diagnostic command register and status register for the proper values. The controller must generate no interrupt. A plotter does need to be attached for this subtest.

4.5.1.2 Subtest 101, IKON Command Loopback

Subtest 101 tests the controller command repertoire (remote buffer clear, end-of-line, form feed, end-of-transmission) using an internal loopback mechanism.

4.5.1.3 Subtest 102, IKON Plotter Exception Loopback

Subtest 102 executes only when the associated test parameters specify this controller as being set for TTL mode and manual intervention is not suppressed. This subtest tests the offline and no-paper exceptions by forcing them through a loopback test mechanism provided for the TTL interface.

4.5.1.4 Subtest 103, IKON Port Selection

Subtest 103 verifies Versatec versus Option (Centronics compatible) port selection by checking a status bit. The Option port is not tested. A plotter does not need to be attached for this subtest.

4.5.1.5 Subtest 104, IKON Plotter Mode Selection

Subtest 104 checks that plotter mode selection is properly echoed in the status register. A plotter does not need to be attached for this subtest.

4.5.1.6 Subtest 105, IKON Programmed Output Loopback

Subtest 105 executes only when the associated test parameters specify this controller as being set for TTL mode and manual intervention is not suppressed.

This subtest performs programmed output of byte patterns, which are checked via the loopback mechanism in the TTL interface. All possible byte patterns are tested by sequentially passing the 256 possible byte values. A random pattern of `0x10001` bytes is tested, using *rand()* and *srand()* with a seed value of 137.

4.5.1.7 Subtest 106, IKON DMA Output Loopback

Subtest 106 executes only when the associated test parameters specify this controller as being set for TTL mode and manual intervention is not suppressed.

This subtest performs Direct Memory Access (DMA) output of byte patterns, which are checked via the loopback mechanism in the TTL interface. All possible byte patterns are tested by sequentially passing the 256 possible byte values. Many random patterns are tested by using *rand()* and *srand()* with a seed value of 137 plus the buffer size. This subtest tests the following DMA buffer sizes (hex values):

Table 4–4, DMA Buffer Sizes

INITIAL SIZE	SIZE INCREMENT	SIZE LIMIT
1	1	10
11	11	100
101	111	ED0
ED1	1	EF0
EF1	101	1000
1001	1001	10000
FFFD	1	10000

Precautions are taken to detect any improper byte swapping by a controller operating in auto-optimizing word mode.

4.5.1.8 Subtest 107, IKON Data Output Interrupt Capability

Subtest 107 executes only if the associated test parameters specify this controller as being set for TTL mode and manual intervention is not suppressed. This subtest tests the IKON capability to generate an interrupt when output data circuits go idle.

4.5.2 Class 2 Subtests

Class 2 subtests verify correct operation of the controller and plotter combination. Patterns are plotted to verify correct operation. Class 2 subtests are listed in the following table:

Table 4–5, Class 2 Subtests

SUBTEST	DESCRIPTION	TIME (min:sec)
200	IKON/Versatec Status Reporting	:19
201	IKON/Versatec Interrupt Reporting	:01
202	IKON/Versatec FF/EOT Busy Check	2:10
210	IKON/Versatec Programmed Output Print	4:40
211	IKON/Versatec DMA Output Print	4:40
220	IKON/Versatec Programmed Output Plot	4:30
221	IKON/Versatec DMA Output Plot	7:35
230	IKON/Versatec Programmed Output Shared	9:00
231	IKON/Versatec DMA Output Shared	12:00

4.5.2.1 Subtest 200, IKON/Versatec Status Reporting

Subtest 200 verifies the IKON/Versatec capability to generate proper status for conditions "device ready," "device/interface ready," "data output complete." The subtest may plot some unlabeled data; it is to be ignored.

4.5.2.2 Subtest 201, IKON/Versatec Interrupt Reporting

Subtest 201 verifies the IKON/Versatec capability to generate proper interrupt for conditions "device ready," "device/interface ready," "data output complete." This subtest also tests the capability to force "device ready" internally to the controller. The subtest may plot some unlabeled data; it is to be ignored.

4.5.2.3 Subtest 202, IKON/Versatec FF/EOT Busy Check

Subtest 202 tests the capability of form feed and end-of-transmission to keep the plotter busy until all previously transmitted data is plotted. Visually verify the plotted output for correctness. Instructions for verification are plotted on the patterns.

4.5.2.4 Subtest 210, IKON/Versatec Programmed Output Print

Subtest 210 executes only if the associated test parameter specifies that the plotter has print capability. All plotter controls are tested.

The subtest prints test patterns, including interpretation instructions, using programmed output. Visually check all patterns for errors. The test generates the following patterns:

- Rotating pattern of ASCII printable characters
- Sail pattern with X's, using remote line terminate function
- Truncated sail pattern, terminating lines with ASCII newline
- Truncated sail pattern, terminating lines with ASCII carriage return
- Truncated sail pattern, terminating lines with ASCII newline plus carriage return
- Single-line pattern testing remote buffer clear function
- Empty pattern testing remote reset function
- Simple pattern testing remote form-feed function
- Simple pattern testing ASCII form-feed capability
- Simple pattern testing remote end-of-transmission function
- Simple pattern testing ASCII end-of-transmission capability

Only the rotating pattern prints if the shortened programmed output test mode is selected with the associated test parameter. The ASCII control capabilities in the truncated sail patterns, the simple pattern testing ASCII form-feed capability, and the simple pattern end-of-transmission capability are not tested unless requested by the associated test parameter.

4.5.2.5 Subtest 211, IKON/Versatec DMA Output Print

Subtest 211 executes only if the associated test parameter specifies that the plotter has print capability. The subtest checks all plotter controls.

The test patterns, described in Subtest 210, are printed, using DMA output with interrupt on "device/interface ready." Only the rotating pattern prints the shortened DMA test mode is selected with the associated test parameter.

4.5.2.6 Subtest 220, IKON/Versatec Programmed Output Plot

Subtest 220 executes only if the associated test parameter specifies that the plotter has plot capability. The subtest checks all plotter controls.

This subtest plots test patterns, including instructions for interpretation, using programmed output. Visually check all patterns for errors. The subtest generates the following patterns:

- Nib-grid pattern (checks each plot nib so that failing nibs may be visually isolated)
- Sail pattern of small bars
- Simple pattern testing remote buffer clear function
- Simple pattern testing remote reset function
- Simple pattern testing remote form-feed function
- Simple pattern testing remote end-of-transmission function

The subtest generates only the nib-grid pattern if the shortened programmed output test mode is selected with the associated test parameter.

4.5.2.7 Subtest 221, IKON/Versatec DMA Output Plot

Subtest 221 executes only if the associated test parameter specifies that the plotter has plot capability. The subtest checks all plotter controls.

This subtest plots the test patterns, described in Subtest 220, using DMA output with interrupt on "device/interface ready." Only the nib-grid pattern is generated if the shortened DMA test mode is selected with the associated test parameter.

4.5.2.8 Subtest 230, IKON/Versatec Programmed Output Shared

Subtest 230 executes only if the associated test parameter specifies that the plotter uses shared print/plot mode. The subtest checks all applicable plotter controls.

The subtest patterns are printed/plotted using shared print/plot mode and programmed output. Visually check all patterns for errors; each pattern provides instructions for interpretation. The subtest generates the following patterns in shared plot mode (print buffer empty):

- Nib-grid pattern (checks each plot nib so that failing nibs may be visually isolated)
- Sail pattern of small bars
- Simple pattern testing remote buffer clear
- Simple pattern testing remote reset

The subtest generates the following patterns in shared print mode (using zero-filled plot buffers to scan out the print):

- Rotating pattern of printables
- Sail pattern of X
- Truncated sail pattern of X, using ASCII newline to terminate print buffers
- Truncated sail pattern of X, using ASCII carriage return to terminate print buffers
- Truncated sail pattern of X, using ASCII carriage return plus newline sequence to terminate print buffers
- Empty pattern to test remote buffer clear function
- Empty pattern to test remote reset function

Also, the subtest generates a "close shades on X" pattern, using actual data in both print and plot buffers.

The "truncated sail patterns" (ASCII) are not generated unless enabled by the associated test parameter. Also, only the "close shades on X" pattern is generated when the shortened programmed output test mode is selected with the associated test parameter.

4.5.2.9 Subtest 231, IKON/Versatec DMA Output Shared

Subtest 231 executes only if the associated test parameter specifies that the plotter uses shared print/plot mode. The test patterns, described in Subtest 230, are printed/plotted using shared print/plot mode and DMA output with interrupt on "device/interface ready." The subtest only generates "close shades of X" pattern when the shortened DMA test mode is selected.

4.5.3 Class 3 Subtests

Class 3 subtests verify correct operation of the controller and plotter when offline and out-of-paper conditions exist. Class 3 subtests are listed in the following table:

Table 4–6, Class 3 Subtests

SUBTEST	DESCRIPTION	TIME (min:sec)
300	IKON/Versatec Offline Status/Interrupt	N/A[1]
301	IKON/Versatec Out-of-Paper Status/Interrupt	N/A

[1] Manual intervention time

4.5.3.1 Subtest 300, IKON/Versatec Offline Status/Interrupt

Subtest 300 tests the capability to properly reflect plotter online status and the associated interrupt capability. This subtest does not execute if manual intervention is suppressed with the associated test parameter.

4.5.3.2 Subtest 301, IKON/Versatec Out-of-paper Status/Interrupt

Subtest 301 tests the capability to properly reflect plotter out-of-paper status. The subtest checks online status, instead, for differential plotters. For TTL plotters, it tests the associated out-of-paper interrupt. This subtest does not execute if manual intervention is suppressed with the associated test parameter.

4.6 Error Messages

8259A status: poll | IRR | IMR | ISR *0xnnnnnnnn*
The complete status of the 8259A interrupt controller is presented. The first byte is the poll status; the next, the interrupt request register; the third, the interrupt mask register; and the fourth, the in-service register.

Datum: [**expected** *0xnn*] [**actual** *0xnn*] [**offset** *0xnn*]
This logs a data byte, including expected value, in case of error. The offset, if given, specifies an offset in a data buffer.

Erroneous interrupt
IKON expected interrupt code *0xnn*
{Interpretation}
IKON actual interrupt code *0xnn*
{Interpretation}
An interrupt of one type occurred but an interrupt of another type was expected. Refer to "Missing interrupt" for interpretation detail.

Exception for iop d from recv_iop
{error message detail}
An error has occurred using the SPU/IOP interface to wait for a signal from the IOP that the command is finished. Refer to "Exception from setup_iop" for message detail.

Exception from load_iop
{error message detail}
An error has occurred while bootstrapping the IOP. Make sure that the IOP program image file *dev4410.x00* is available and readable. It must reside either in the current directory or */mnt/test*. Refer to "Exception from setup_iop" for message detail.

Exception from send_iop (n)
{error message detail}
An error has occurred using the SPU/IOP interface to signal the IOP that a command is ready. Refer to "Exception from setup_iop" for message detail.

Exception from setup_iop
{error message detail}
An error occurred when preparing to access the IOP. When this error occurs, the message detail may be:

Table 4–7, IOP Access Error Messages

TEXT	MEANING
HARD ERROR	Hardware error
IOP BUS ERROR	Controller address error
IOP CACHE ERROR	IOP hardware error
IOP PBUS ERROR	IOP hardware error
MMIO ERROR	Main memory error
MULTIBUS ERROR	Hardware error
TIMEOUT	Possible hardware error

Exception from start_iop
{error message detail}
An error has occurred while starting the IOP. Refer to "Exception from setup_iop" for message detail.

Exception in mmalloc_init. Is main memory initialized?
There is a problem accessing main memory. Assure that main memory is present; initialize, if necessary, using *mminit*. If problems persist, consult the Technical Assistance Center.

Function status: message
{UNIX error message if applicable}
There is a problem in IOP processing. The following messages may appear:

Table 4–8, IOP Processing Error Messages

TEXT	MEANING
UNIX ERROR	Error specified in message
SUBSYSTEM WON'T GO READY	Hardware problem

IKON diag dcmd 0xnn **means:**
{interpretation of byte}
This logs a diagnostic command loopback byte. (The interpretation appears in the next entry.)

IKON diag dcmd: expected 0xnn, **actual** 0xnn, **differences:**
{interpretation of difference between expected and actual}
The interpretation may be one or more of the following, where the brackets enclose an optional *NOT* indicator:

Figure 4–4, Interpreted Error Messages

```
 UN:  UNUSED BITS SET
_VR:  [NOT] VERSATEC RESET
_VC:  [NOT] VERSATEC CLEAR
_VF:  [NOT] VERSATEC FORM FEED
_VT:  [NOT] VERSATEC END OF XMIT
_VL:  [NOT] VERSATEC END OF LINE
```

IKON status 0xnn **means:**
{interpretation}
This is the IKON status byte. The interpretation may be:

Figure 4–5, Interpreted Status Byte Error Messages

```
_OP:  VERSATEC PORT
_OP:  OPTION PORT

 DR:  DEV READY (diag)
 DR:  DEV NOT READY (diag)

 IR:  DEV RDY & OUTPUT IDLE
 IR:  DEV BSY | OUTPUT ACTV

 DP:  OUTPUT IDLE
 DP:  OUTPUT ACTIVE

 PL:  PRINT MODE
 PL:  PLOT MODE

 SP:  NOT SHR'D PRT/PLOT
 SP:  SHR'D PRT/PLOT MODE

 OL:  DEV ON LINE
 OL:  DEV OFF LINE

 NP:  DEV NO PAPER
 NP:  DEV HAS PAPER
```

IKON status: expected 0xnn, **actual** 0xnn, **differences:**
{interpretation}

An unexpected status code was obtained. The differences between the actual code and what was desired are interpreted (refer to previous interpretations).

Main memory allocation error

There is a problem allocating main memory. Be sure that main memory is present and initialized. If problems persist, consult the Technical Assistance Center.

Missing interrupt
IKON expected interrupt code *0xnn*
{Interpretation}

An interrupt, which should have occurred, did not occur. The interpretation may be:

Figure 4–6, Interpreted Interrupt Error Messages

```
IR0:  DEVICE/INTERFACE READY
IR1:  OUTPUT IDLE
IR2:  DEVICE READY
IR3:  DEVICE OUT OF PAPER
IR4:  DEVICE OFF LINE
```

Spurious interrupt error: *dd* **expected,** *dd* **actual**

The count of spurious interrupts was not as expected. Spurious interrupts are those which are actually 'spurious' (IKON 8259A spurious interrupt) or are extra, but otherwise valid interrupts. (An IKON 8259A spurious interrupt is a condition where the original interrupt cause vanishes before processor interrupt service performs the interrupt acknowledge.)

Unexpected interrupt

IKON actual interrupt code *0xnn*

{interpretation}

An interrupt that was not expected occurred. Refer to "Missing interrupt" for interpretation detail.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A
Reporting Problems

A.1 Overview

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the *contact* utility. The *contact* utility is an online system for reporting problems to the TAC. To learn *contact* by using it, enter **contact** at the system prompt and then answer the questions as they appear on the screen. To find out more about using *contact*, read through this appendix. It describes prerequisites and tips for using *contact* and the step-by-step process *contact* takes you through.

A.2 Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address the diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation problem, contact the TAC. This group stands ready to solve such problems.

A.3 The *contact* Utility

The TAC recommends using the *contact* utility to report a hardware, software, or documentation problem. The *contact* utility is an interactive utility that helps the TAC track reports and route them to the the CONVEX personnel most qualified to fix them.

After invoking *contact*, it prompts for information about the problem. When you finish your report, *contact* electronically mails it to the TAC. You are notified within 48 hours that the TAC has received your report.

A.4 Prerequisites

To use *contact* requires

- a UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- the full path name of the program or utility in question
- the version number of the program or utility in question

A.4.1 UUCP Connection

Before using *contact*, check with your system administrator to be sure there is a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX system to another. The *uucp* (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

A.4.2 Finding the Program Path Name

To determine the full path name of the program or utility in question, use the *which* command. The following screen illustrates using the *which* command to find the full path name of the loader (*ld*) utility:

```
>which ld
/bin/ld
>
```

In this example, the full path name of the loader is */bin/ld*.

For more information on the *which* command, refer to the *which*(1) man page. You can also use the *info* online information system. Enter **info which** at the system prompt. If you use the C shell (*csh*), you can also use the *whence* command to find the program path name. The *whence* command works like *which*, only faster.

A.4.3 Finding the Program Version Number

To determine the version number of the program or utility in question, use the *vers* command. The following screen illustrates using the *vers* command (enter **vers**, then the path name of the program or utility) to find the version number of the loader (*ld*) utility.

```
>vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader utility version number is 7.0.

For more information on the *vers* command, refer to the *vers*(1) man page. You can also use the *info* online information system. To do so, enter **info vers** at the system prompt.

A.5 Tips on Using the *contact* Utility

The *contact* utility is interactive and easy to use. This section lists tips to help use it efficiently. In particular, this section tells how to

- use a *.contact* file
- abort a contact session
- resubmit an aborted report
- suspend a contact session
- move from one prompt to another
- use tilde-escape sequences in the *contact* utility

A.5.1 Using a *.contact* File

When invoked, *contact* prompts for information regarding the problem. The first prompt is for your name, title, phone number, and company name. You can, however, create a *.contact* file to skip this first prompt. Follow these steps:

1. Create a *.contact* file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke *contact*, it automatically includes the *.contact* file as input for the first prompt and proceeds to the next prompt.

A.5.2 Aborting the Report

To abort a contact report, either enter the interrupt key (usually CTRL-C) or choose the abort option when prompted by the *contact* utility. Using CTRL-C to abort does not save the contents of the report. Using the abort option saves the contents of the report in a file named *dead.report* in your home directory.

A.5.3 Submitting the *dead.report* File

When aborting a contact session, the *contact* utility saves the report in a file named *dead.report* in your home directory. Using the *contact* command with the *-r* option automatically merges the contents of the *dead.report* file into the new contact session. Enter

contact -r

and *contact* finds the *dead.report* file in your home directory and merges it into the contact report. You can then edit the report. When you end the editing session, *contact* returns to the final prompt, which asks you to review, edit, submit, or abort the report.

A.5.4 Suspending a Report

Sometimes it is necessary to stop in the middle of a contact report and return to the shell (for instance, to suspend the contact session to find the program path name or version number). To suspend the contact session, press CTRL-Z. To return to the contact session, enter **fg**. Using CTRL-Z and the *fg* (foreground) command lets you switch back and forth between the *contact* utility and the shell. You cannot, however, use CTRL-Z and *fg* to switch back and forth if you are using a Bourne shell (*sh*).

A.5.5 Ending a Response

The *contact* utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press RETURN. Other prompts require more than a one-line response; to move to the next prompt, press CTRL-D.

A.5.6 Tilde-Escape Sequences

The *contact* utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. The following tilde sequences are recognized by *contact*:

~e	Start the text editor (defined in your EDITOR environment variable).
~h	Display a list of available tilde-escape sequences.
~p	Print the contact report to the terminal screen.
~r *filename*	Read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence only works for prompts that allow more a than one-line response.
~~	Insert a single tilde as the first character in the line.

A.6 Using the *contact* Utility

The *contact* utility prompts for the following information:

- your name, title, phone number, and corporate name
- the name and version of the product involved
- a one-line summary of the problem
- a detailed description of the problem
- the priority of the problem
- instructions on how to reproduce the problem
- comments about the problem
- comments about the documentation supporting the problem
- files to include in the contact report

The following is a step-by-step discussion of these prompts:

1a. To invoke the *contact* utility, enter **contact** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. The following screen illustrates the *contact* command and the system response:

```
>contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

1b. If there is a *.contact* file in your home directory, *contact* skips the first prompt. The following screen illustrates the *contact* command and the system response when a *.contact* file is in your home directory:

```
>contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

2. The *contact* utility prompts for the version number of the product. If you do not know the version number, use CTRL-Z to suspend the session. Use the *which* (or *whence* if using *csh*) and *vers* commands to find the version number of the product. Use the *fg* command to return to the session and enter the version number in the form X.X or X.X.X.X.
3. The *contact* utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Make this summary as descriptive as possible in one line.
4. The *contact* utility prompts for a detailed description of the problem. Make this description as complete as possible. Include source code and a stack backtrace whenever possible. (Refer to the *adb*(1) or *csd*(1) man page for information on obtaining a stack backtrace.) The more information provided, the quicker the TAC can isolate and solve the problem.
5. The *contact* utility prompts for the priority of the problem. The following screen illustrates this prompt and the priority levels from which to choose; you must enter a priority number.

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
>
```

6. The *contact* utility prompts for an explanation of how to reproduce the problem. Include the command syntax and options you used and anything else you did to make your program run.
7. The *contact* utility prompts for any other pertinent comments. Include any relevant information.
8. The *contact* utility prompts for suggestions regarding the documentation supporting the product. Indicate if the documentation could be revised to address the question.
9. The *contact* utility asks for the names of files necessary to reproduce the problem. The following screen illustrates the *contact* prompt and sample user response:

```
Are there any files that should be included in this report (yes | no)?
>yes
Please enter the names of the files, one to a line (^D to terminate)
>test.f
>~/subroutines/sub.f
>
```

> **NOTE**
>
> Tilde-escape sequences are not recognized in responses to this prompt. Instead, *contact* treats a tilde in this section to mean your home directory. This convention is based on use of the tilde for expanding file names in *csh*.

If the files specified are small text files, they are automatically included in the contact report. If the files are too big to be included in this report, *contact* gives further instructions on how to submit these files.

To specify a directory, combine the directory files into a single file using the *tar* command (refer to the *tar*(1) man page for further information) or enter each file name in the directory on a single line in the contact report.

10. The *contact* utility prompts you to review, edit, submit, or abort the contact report. The following screen illustrates this prompt:

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review — Review the text of your contact report. You are then prompted again to select an option.

Edit — Edit the text of the contact report. If you choose to edit the report, *contact* puts you in your default text editor.

Submit — Send the report to the CONVEX TAC. You are notified within 48 hours that the TAC has received the report. This option exits the *contact* utility and returns you to the shell environment.

Abort — Save the text of your report in a file named *dead.report* in your home directory. This option exits the *contact* utility and returns you to the shell environment.

Index

R

S

T

U

V

W

CONVEX Multibus Plotter (*dev4410*) Diagnostics Manual
Document No. 760-002430-000
First Edition

Reader's Forum

Please use this form to submit comments or questions concerning the clarity and service of this manual. Constructive critical comments are most welcome and help us continue in our efforts to generate quality customer documentation. Please list the page number for questions or comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
From all locations in continental U.S.	1(800)952-0379
From locations in Alaska & Hawaii	1(214)497-4379
From locations in Canada	1(800)345-2384
From all other locations	Contact nearest CONVEX office

Direct mail orders to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851

(Fold Here Second)

(Tape or Staple)

CONVEX Multibus Plotter (*dev4410*) Diagnostics Manual
Document No. 760-002430-000
First Edition

Reader's Forum

Please use this form to submit comments or questions concerning the clarity and service of this manual. Constructive critical comments are most welcome and help us continue in our efforts to generate quality customer documentation. Please list the page number for questions or comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
From all locations in continental U.S.	1(800)952-0379
From locations in Alaska & Hawaii	1(214)497-4379
From locations in Canada	1(800)345-2384
From all other locations	Contact nearest CONVEX office

Direct mail orders to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851

(Fold Here Second)

(Tape or Staple)