1500016(H)
March 1987

ACP 629
User's Manual

# ACP 629
**User's Manual**

Advanced Computer Communications
720 Santa Barbara Street
Santa Barbara
CA 93101

## ACC CUSTOMER SERVICE

California, Alaska, Hawaii
phone (805) 963-9431 collect

elsewhere in the United States
phone (800) 222-7308 toll free

outside the United States
TWX 910 334-4907
(answer back ACC SNC)

# MANUAL INTRODUCTION

This manual describes how to install and use the ACP 629 system (ACC PN 8600213 for RS-232C applications or PN 8600216 for RS-449/422 applications) manufactured by Advanced Computer Communications, Santa Barbara, California.

**Content.** This manual consists of the chapters and the appendices shown below.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTERS (Continued) Page

TABLE OF CONTENTS

## TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES (Continued)

## ABBREVIATIONS USED IN THIS DOCUMENT

| | |
|---|---|
| <CR> | indicates carriage return |
| ac | alternating current |
| ACC | Advanced Computer Communications |
| CCITT | International Consultative Committee for Telephone and Telegraph |
| COMREGs | Communication Registers |
| COMREGs | Communication Registers |
| CPU | Central Processing Unit |
| CSR | Command Status Register |
| dc | direct current |
| DCE | Data Circuit Equipment |
| DEC | Digital Equipment Corporation |
| DM | Disconnect Mode |
| DMA | Direct Memory Access |
| EPROM | Erasable Programmable Read Only Memory |
| FE | Front End |
| HDLC | High-level Data Link Control |
| I/O | Input/Output |
| IA5 | International Alphabet No.5 |
| IVP | Installation Verification Program |
| LAPB | Link Access Procedure, Balanced Mode |
| LCN | Logical Channel Number |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| NPG | Non-Processor Grant |
| PAD | Packet Assembly/Disassembly |
| PDN | Packet-Switching Data Network |
| PN | Part Number |
| SPC | Small Peripheral Controller |
| TT | Terminal Timing |
| V | Volts |

# Chapter 1

# System Overview

## 1.1 Purpose and Capabilities

The ACP 629 is a microprocessor-based communications Front End (FE) that supports DMA (Direct Memory Access) applications needing X.25 Packet Assembly/Disassembly (PAD) capability. The ACP 629's X.25 packet switching technology reduces telephone expenses and transmission errors. X.25 is the CCITT international protocol for connecting computer equipment to a Packet-switching Data Network (PDN). X.25 defines three layers of the International Standards Organization's open systems architecture.

The lowest protocol layer, level I, defines the electrical and mechanical interface to the network. The ACP 629 is available with an RS-232C (ACC PN 8600213) or RS-449/422 (ACC PN 8600216) electrical interface option. The level II protocol (frame level) assures reliable data communication through one physical communication circuit. The ACP 629 uses the LAPB protocol with HDLC framing. Level III protocol (packet level) establishes and maintains multiple virtual communications circuits on the PDN. The ACP 629 software handles the three X.25 levels. The ACP 629 supports 32 switched virtual circuits. It assembles, disassembles, and verifies packets and frames without interrupting host activities.

Features of the ACP 629 follow:

- Offloads all X.25 protocol processing (HDLC, LAPB, PACKET, PAD).
- Supports 32 X.25 switched virtual circuits over a single X.25 network connection. All circuits can receive or place X.25 calls.
- Supports 18 CCITT X.3 parameters.
- Supports X.28 user interface with additional commands.
- Supports X.29 PAD interface.
- Supervisor command interface enables you to customize operation, gather statistics, and verify installation.
- Supports network link line speeds up to 56k bits/sec.
- Host interface uses DMA output and multiple character input under a single host interrupt.
- Outgoing packet strategy maximizes characters per packet with minimum character delay.

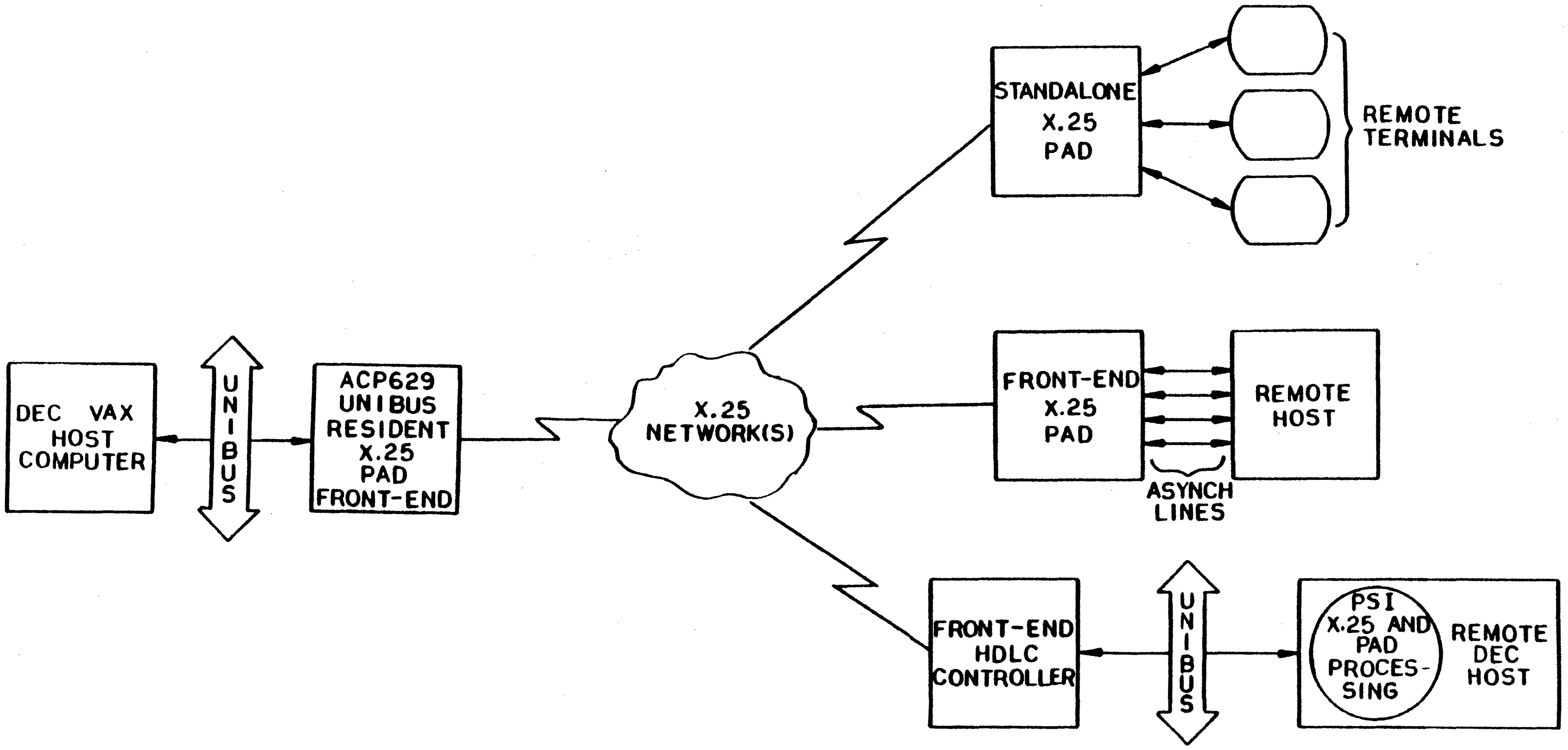


Figure 1-1. Typical X.25 Component PAD Configurations

## 1.2 Equipment Supplied

1. One ACP 629 board (RS-422: PN 8100262, RS-232C: PN 8100257)
2. One distribution panel assembly with cabling to connect to the ACP 629 (RS-422: PN 8000132, RS-232C: PN 8000081)
3. One *ACP 629 User's Manual* (PN 1500016)
4. One software distribution kit
5. One loopback connector (RS-422: PN 8300392, RS-232C: PN 8300404)

## 1.3 Reference Documents

Digital Equipment Corporation (DEC order number in parentheses)

1. *VAX/VMS System Management & Operations Guide* (AA-M574A-TE)
2. *VAX/VMS Guide to Writing a Device Driver* (AA-H499C-TE)
3. *VAX/VMS System Services Reference Manual, Vol. 1* (AA-D018C-TF)
4. *Guide to VAX/VMS Software Installation* (AA-Y514A-TE)
5. *VAX/VMS I/O User's Reference Manual: Part I* (AA-Z600A-TE)

International Consultative Committee for Telephone and Telegraph

1. *CCITT Recommendation X.25*
2. *CCITT Recommendation X.3*
3. *CCITT Recommendation X.28*
4. *CCITT Recommendation X.29*

University of California at Berkeley, Department of Electrical Engineering and Computer Science

1. *UNIX Programmer's Manual Reference Guide*
   4.2 Berkeley Software Distribution

2. *UNIX System Manager's Manual*
   4.2 Berkeley Software Distribution

3. *UNIX System V Release 2.0 User Reference Manual*
   AT&T Order Code 307-109

4. *UNIX System V Release 2.0 Programmer Reference Manual*
   AT&T Order Code 307-113

5. *UNIX System V Release 2.0 Administrator Reference Manual*
   AT&T Order Code 307-111

# Chapter 2

# Hardware Installation

## 2.1 Inspection and Preparation

The ACP 629 arrives in a carton marked FRAGILE. The components are surrounded in foam packing material, with the ACP 629 wrapped in a static-resistant plastic bag and packaged in a separate padded envelope. The carton and packing materials are reusable and must be saved in case reshipment becomes necessary.

**2.1.1 Unpacking**. Open the corrugated carton from the side indicated. Brush away any packing material from the padded envelope before opening. Carefully remove the board from the padded envelope. Leave the board in its plastic bag until you are ready to install it, or whenever it is not in use. Do not stack boards that are unprotected or that are protected only by plastic bags.

**2.1.2 Inspection**. Check the ACP 629 shipping container for signs of having been dropped or severely shocked. Check for severe indentations or abrasions of the container. If the container appears damaged, notify both the carrier and the Customer Service Department at ACC immediately. After removing the ACP 629 from its plastic bag, carefully blow off any dust or packing material. Check for superficial damage (i.e., scratches or dents) on painted surfaces and electrical connectors. Check the board for scratches, particularly on the gold fingers or across any traces.

**2.1.3 Reporting Damaged or Incomplete Equipment**. If the inspection reveals flaws in the equipment, if any parts are missing, or if the container is damaged contact the Customer Service Department at ACC. In California, Alaska, and Hawaii call collect (805) 963-9431. Elsewhere in the United States call toll-free (800) 222-7308. Outside the United States use TWX number 910 334-4907 (answer back ACC SNC).

## 2.2 Installation Considerations

The ACP 629 system needs one SPC (small peripheral controller) slot on the UNIBUS backplane. The ACP 629 may be installed in a PDP-11 or VAX processor box, or an expansion chassis.

**2.2.1 Electrical Requirements.** The maximum electrical requirements for the ACP 629 are shown below:

| | |
|---|---|
| +5V dc @ | 6.0A |
| +15V dc @ | 0.35A |
| -15V dc @ | 0.40A |

**2.2.2 Environmental Requirements.** The system needs an ambient temperature between +5 and +50 degrees Celsius, with relative humidity between 10% and 95%, non-condensing.

## 2.3 Printed Circuit Board Preparation

Check that the ACP 629 jumper and switch settings agree with the settings in figure 2-1. As shown in this figure, the ACP 629 Command Status Register (CSR) address is set to the default address 766740. With this setting, the CSR is located starting at UNIBUS address 766740. The Communication Registers (COMREGs) are located after the CSR. The CSR and COMREGs occupy the 40 octal bytes following 766740. If any of these bytes are already in use, or if more than one ACP 629 is to be installed, adjust the CSR address setting as in table 2-1.

**NOTE**

1. The ACP 629 requires 40 octal bytes of space for its CSR and COMREGs. This space must start on a 40 octal byte boundary.

2. Before choosing an address, check that it is not in use by another device.

Table 2-1. ACP 629 CSR Settings

| Address Bit | Switch U82 | First Board 766740 | Second Board 767000 | Third Board 767040 | Fourth Board 767100 |
|---|---|---|---|---|---|
| A-11 | S-1 | off | off | off | off |
| A-10 | S-2 | off | off | off | off |
| A-9 | S-3 | on | off | off | off |
| A-8 | S-4 | off | on | on | on |
| A-7 | S-5 | off | on | on | on |
| A-6 | S-6 | off | on | on | off |
| A-5 | S-7 | off | on | off | on |
| A-4 | S-8 | on | on | on | on |
| A-3 | S-9 | on | on | on | on |
| A-12 | S-10 | on | on | on | on |
| A-2 | S-11 | on | on | on | on |

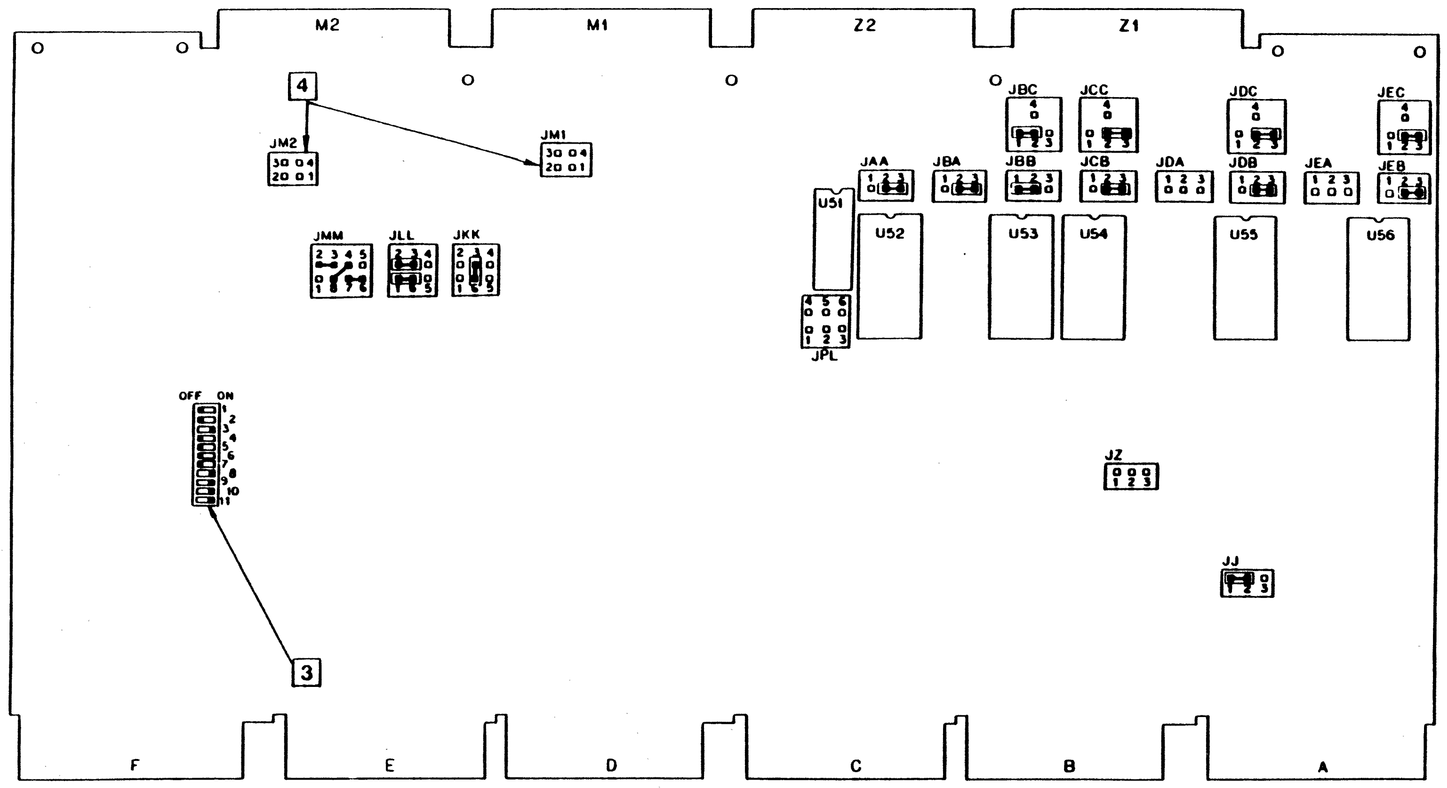
On the ACP 629, there is no hardware setting for the ACP 629 interrupt vector. This vector is set during ACP 629 driver software installation or system initialization.

## 2.4 Preparation of the SPC Slot

**CAUTION**

The following instructions tell you to modify DEC hardware. This might jeopardize the DEC equipment warranty. If in doubt about the warranty or if you don't understand the following instructions, contact DEC field service before starting.

Cut the Non-Processor Grant (NPG) jumper, if it is present and intact, at the SPC slot where the ACP 629 will be installed. The NPG jumper is located on the wiring side of the backplane between coordinates CA1 to CB1. Once this jumper has been cut, inserting a G727 grant continuity module when the slot is unoccupied is *not* sufficient to restore normal bus operation. Insertion of a DMA and Interrupt Grant Card (ACC PN 8100140 or DEC Module number G7273) restores normal bus operation. To ease recognition of such DMA-SPC slots (which are also used by RX211, RL11, and other disk controllers), we strongly recommend that you clearly mark them.



| JUMPER | CONNECTION |
|---|---|
| JAA | 2 TO 3 |
| JBA | 2 TO 3 |
| JBB | 1 TO 2 |
| JBC | 1 TO 2 |
| JCB | 2 TO 3 |
| JCC | 2 TO 3 |
| JDA | |
| JDB | 2 TO 3 |
| JDC | 2 TO 3 |
| JEA | |
| JEB | 2 TO 3 |
| JEC | 2 TO 3 |
| JJ | 1 TO 2 |
| JKK | 3 TO 6 |
| JLL | 1 TO 6, 2 TO 3 |
| JMM | 2 TO 3, 7 TO 6, 8 TO 4 |
| JM1 | |
| JM2 | |
| JZ | |
| JPL | |

$\boxed{4}$ Configured for RS-232C. For RS-422/449, connect pin 3 to 4 and pin 1 to 2.

$\boxed{3}$ Black mark indicates depressed position. Set address for 766740.

2 Numbering does not appear on board.

1 Jumper/EPROM locations exaggerated for clarity.

Notes: unless otherwise specified.

Figure 2-1. Jumpers and Switches, ACP 629

## 2.5 Attachment to the UNIBUS Subsystem

Refer to figure 2-2 while doing the following steps.

1. Switch off power to the entire system before doing the following steps:

2. Install the distribution panel at the back of the cabinet and route cables through the chassis.

3. Remove the NPG jumper in the SPC slot where the ACP 629 is to be inserted. (See 2.4 for instructions.)

4. Insert the ACP 629 into the selected SPC slot.

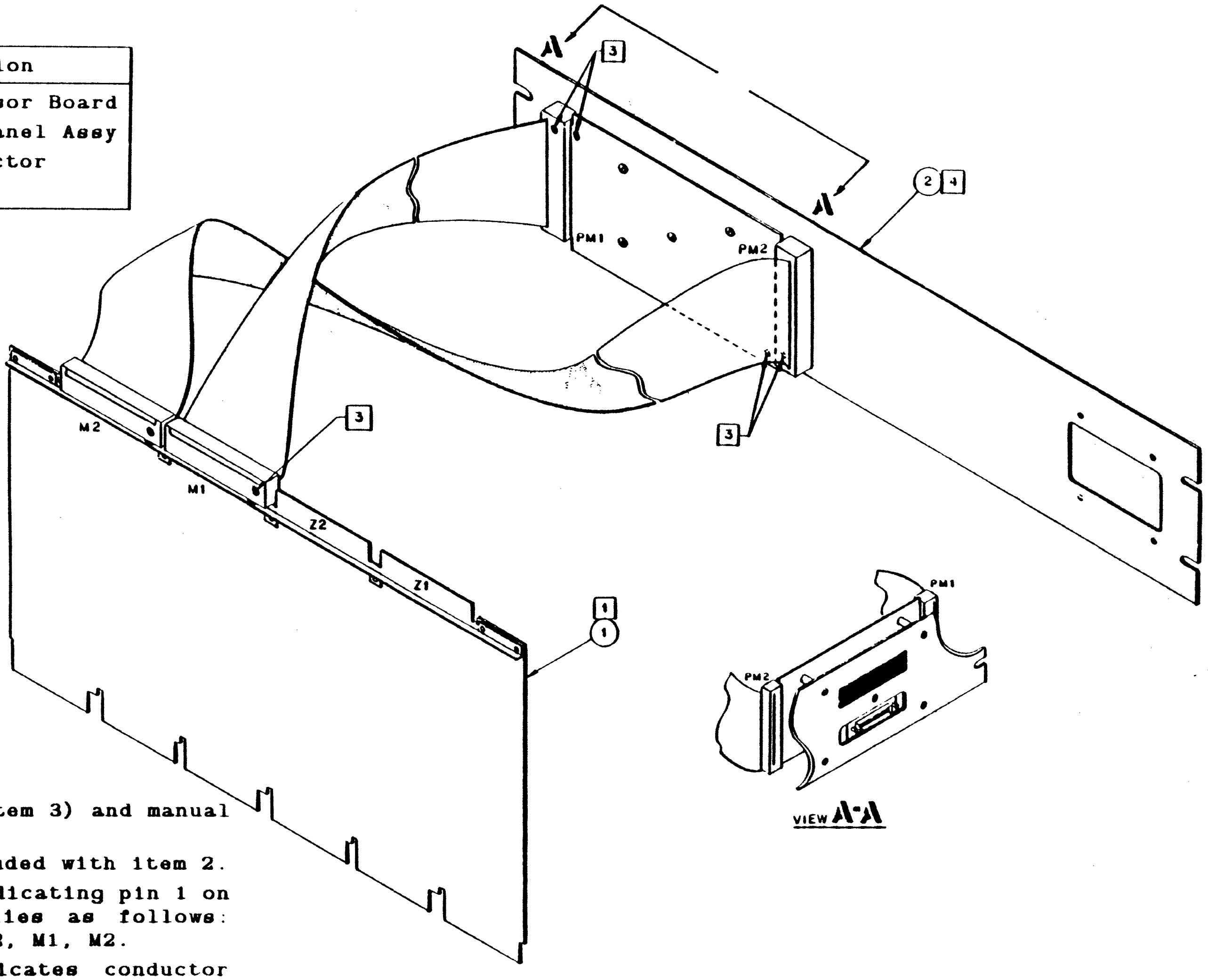
5. Install the cables between the ACP 629 and the distribution panel. Connect M1 on the ACP 629 to PM1 on the distribution panel, and M2 on the ACP 629 to PM2 on the distribution panel.

6. Power up the system and check power. The necessary voltage range for the ACP 629 is between 4.75V dc and 5.25V dc. Installing the ACP 629 board increases the load on the backplane power supply. A low voltage on the +5 volt bus causes erratic operation of the ACP 629. Check the +5 volt power supply and adjust to 5.0 volts if necessary.

7. Close the processor/expansion box.

8. Start up the system.

| Item | Description |
|---|---|
| ① | ACP 629 Processor Board |
| ② | Distribution Panel Assy |
| ③ | Loopback Connector |
| ④ | Manual |



5 Loopback connector (item 3) and manual (item 4) not shown.

$\boxed{4}$ Cable assemblies included with item 2.

$\boxed{3}$ Affix colored dots indicating pin 1 on cable and PC assemblies as follows: Blue dots for PM1, PM2, M1, M2.

2 Stripe on cable indicates conductor no. 1.

$\boxed{1}$ Component side of board shown.

Notes: unless otherwise specified

Figure 2-2. ACP 629 System Configuration

## 2.6 Terminal Timing and Data Reliability

In the RS-449/422 configuration, terminal timing is provided on pins 17/35 (TT). In the RS-232C configuration, terminal timing is provided on pin 24 (TxC). This is necessary for two reasons:

1. The RS-232C and RS-449/422 configurations of the ACP 629 can be required to provide timing on the TT lead.
2. All ACC ACP 629 self-test diagnostics for RS-232C and RS-449/422 require this configuration.

Appendix G gives translation tables for RS-449 and RS-232C signals.

*RS-449/422 Configurations*

When the ACP 629 with RS-449 interface receives external timing, the extra clock signal on TT can hinder function of the serial interface. This problem is caused by crosstalk from the unused clock to other control and data leads in the serial interface. Long, unshielded data cables connecting the ACP 629 to other modems and interfaces compound this problem.

The ACP 629 can be jumpered internally to remove the unused clock on the TT lead. For the "A" side, remove the jumper from pins 2 to 3 on JLL. For the "B" side, remove the jumper from pin 2 to pin 3 on JMM. The ACP 629 diagnostics need a clock source to run correctly. Therefore, these jumpers must be in place or an external clock must be used when running the diagnostics.

Before removing any jumpers, be sure that all the connecting data cables have a separate shield and ground for each signal/signal pair. Use Belden cable type 9768 Multiple Pair Individually Shielded 22 gauge computer cable or its equivalent to assemble connecting cables from the ACC distribution panel to the interface device. Such cables lessen crosstalk and promote data reliability.

## 2.7 Modem Eliminators

Modem eliminators replace modems between two data communications devices. Null modem cables, unlike modem eliminators, are typically used only when two communications processors are less than 25 feet apart and no clock source is required. Cables longer than 25 feet or poor quality cables might cause signal distortions and might interfere with data transmission.

NOTE

Because of the possible distortion of timing signals over null modem cables, we do not recommend their use. Instead use active modem eliminators as a better solution for connection to data communications devices without the use of modems.

ACC does not supply modem eliminators or null modem cables with its products. If these devices are to be used with ACC equipment, we strongly suggest that both the network switch and the host system use a common ground source.

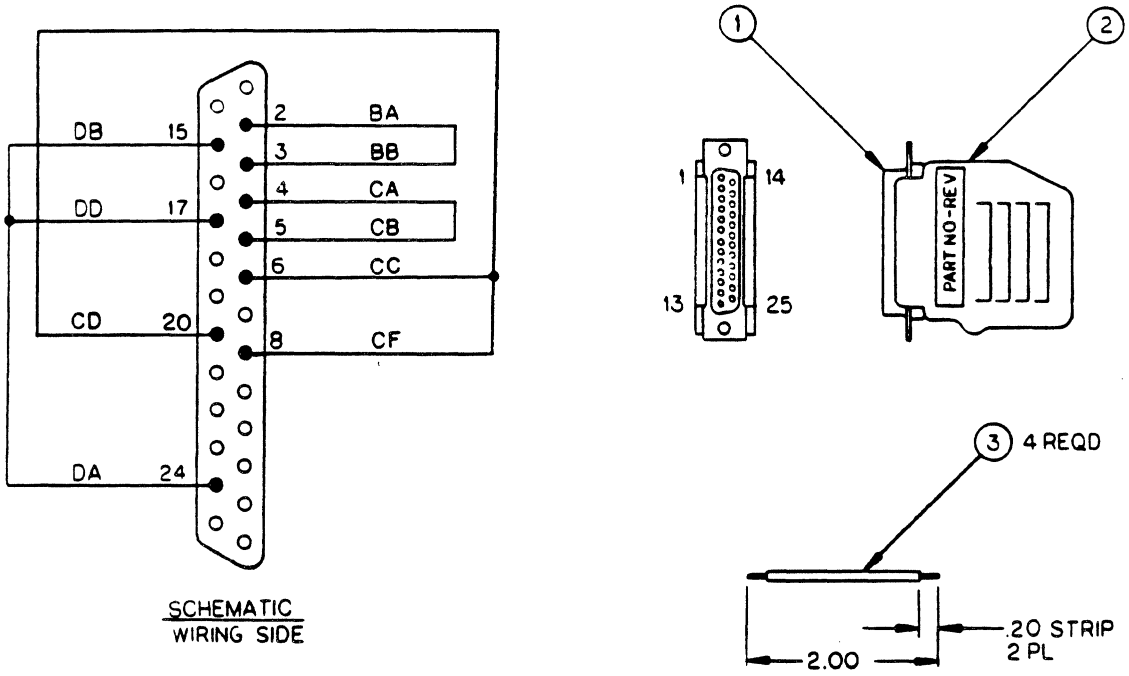
## 2.8 Recommended Modem Eliminator

For an ACP 629 with RS-422 interface, we recommend the stand-alone synchronous modem eliminator model GA-ME101 available from

Black Box Corp., P.O. Box 12800,
Pittsburgh, PA 15241-9980,
telephone (412) 746-5500.

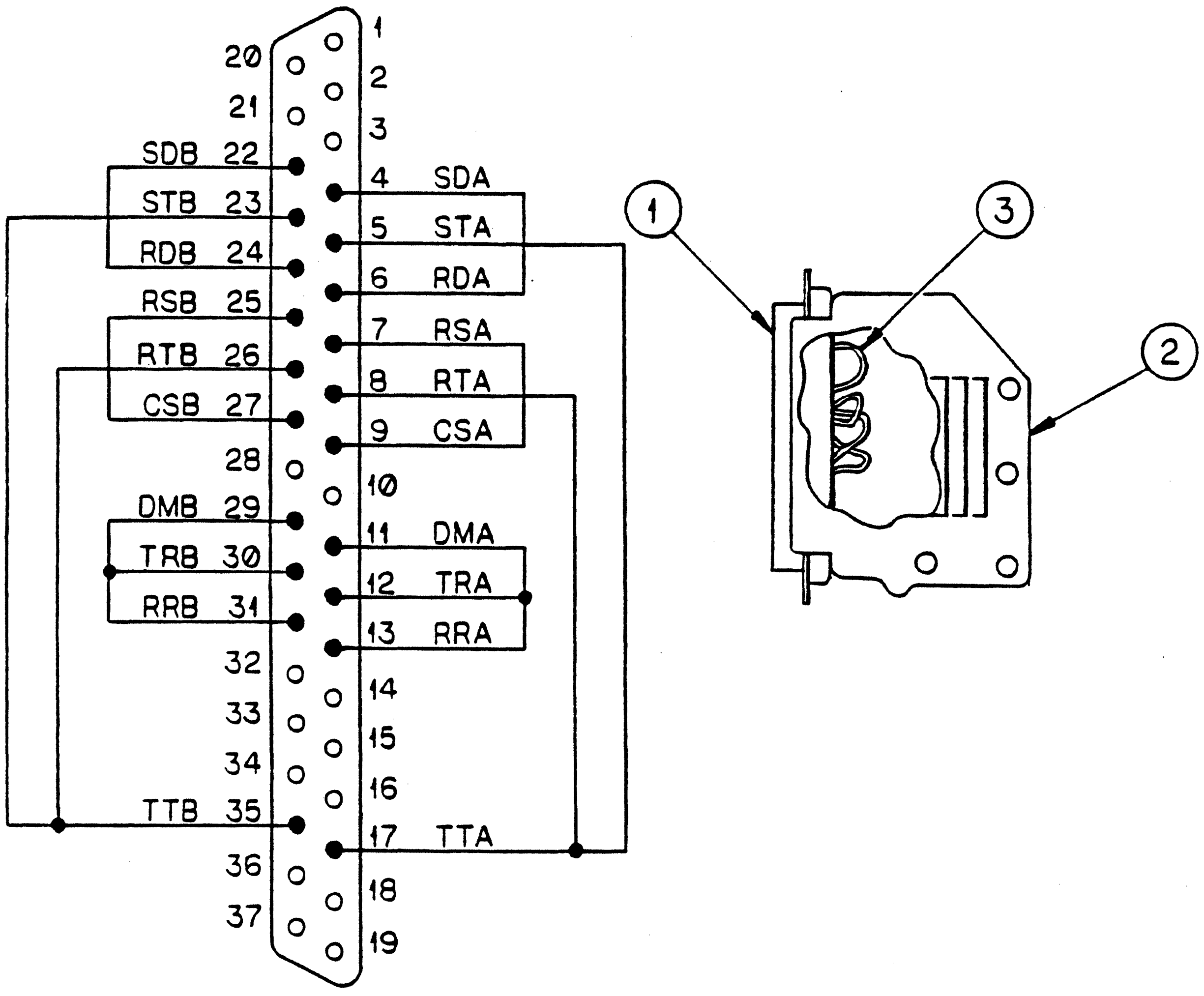
## 2.9 Loopback Connectors

Figures 2-3 and 2-4 detail external loopback connectors that can be used during system testing.



(1) 25-pin connector (2) 2-piece connector shell (3) 24 awg wire

Figure 2-5. RS-232C Loopback Connector



(1) D-subminini 37-pin connector (2) 2-piece connector shell (3) 24 awg wire

Figure 2-6. RS-449/422 Loopback Connector

# Chapter 3

# VMS Driver Installation

## 3.1 Introduction

The ACP 629 VMS support package is shipped as a save set suitable for installation using the VMSINSTAL command procedure in the *Guide to VAX/VMS Software Installation* in the VMS documentation kit. Table 3-1 lists the files contained in or created by the save set A629Vxxx.A (where xxx is the revision level, e.g., 020 for revision 2) and their normal disposition at the end of the installation procedure.

Table 3-1. ACP 629 VMS Save Set Files

| FILENAME | DESCRIPTION | DISPOSITION |
|---|---|---|
| KITINSTAL.COM | Automatic installation procedure | (deleted) |
| TPDRIVER.MAR | Class driver source file | (deleted) |
| ACCRESET.EXE | Program to reset front-end | SYS$SYSTEM |
| ACP629T1.EXE | Installation verification program (IVP) | SYS$TEST |
| ACP629T1.C | Source for IVP | SYS$EXAMPLES |
| | - Installation makes the following files: | |
| TPDRIVER.EXE | User class driver | SYS$SYSTEM |
| TPCONN.COM | Startup configuration file | SYS$MANAGER |
| ACP629T1.COM | Installation verification procedure | SYS$TEST |

## 3.2 Before the Installation

Before beginning installation, do the following:

1. Ensure that the system receiving the ACP 629 has been recently backed up.

2. Locate a CSR and vector address for each ACP 629 to be installed. Each ACP 629 requires 32 bytes of unused UNIBUS address space beginning on a 40 octal byte boundary, and 8 contiguous bytes of interrupt vector space (enough for two vectors) beginning on a 10 octal byte boundary. You can get this information using the VMS program SYSGEN, combined with the SYSGEN commands SHOW/UNIBUS and SHOW/CONFIGURATION.

3. Decide on naming conventions for the following elements:
   a. The executable image of TPDRIVER
   b. The ACP 629 reset program ACCRESET
   c. The device prefix for ACP 629 ports

## 3.3 Installation Procedure

The following sections describe installation of the TPDRIVER on a VAX/VMS system. This installation requires that the privileges PHY_IO, CMKRNL, OPER, and SYSPRV be enabled. The example that follows is in section 3.4 in its original form, showing what a typical installation session looks like.

1. Login under the system usercode/password:

```
Username: SYSTEM
Password:
```

   VMS responds with a system greeting similar to the following:

```
     Welcome to VAX/VMS version V4.1 on node ACCTST
    Last interactive login on Monday, 19-MAY-1986 07:18
    Last non-interactive login on Sunday, 18-MAY-1986 15:57
     19-MAY-1986 07:19:13   _RTA1:

Hello, user!
```

2. Enter the following command:

```
$ @sys$update:vmsinstal a629v msa0:
```

   This initiates the VAX/VMS installation procedure. The command line has the following elements:

| Element | Explanation |
|---|---|
| @sys$update:vmsinstal | Command procedure that controls installation of software products on VAX/VMS systems. |
| a629v | Name of the product to be installed. |
| msa0: | Device where saveset containing the ACP 629 software package is located. This could also be a directory if the G option of VMSINSTAL has been used to transfer the saveset from tape to disk prior to initiation of the installation session. |

If the VMSINSTAL command procedure has been correctly initiated, it displays

```
    VAX/VMS Software Product Installation Procedure V4.0

It is 19-MAY-1986 at 07:20.
Enter a question mark (?) at any time for help.
```

3. Answer the following question:

```
* Are you satisfied with the backup of your system disk [YES]?
```

The default response to this question is YES.

We recommend that you make a full system backup before installing any software product in case of installation failure. If a system backup has not been done, enter NO to abort the installation session. A YES answer continues the session.

4. Mount the tape containing the ACP 629 saveset:

```
Please mount the first volume of the set on MSA0:.
```

Respond YES when the tape is ready:

```
* Are you ready? y
```

VAX/VMS responds with the following messages if the tape was loaded correctly:

```
%MOUNT-I-MOUNTED, A629V0 mounted on _MSA0:

The following products will be processed:

  A629V V2.0

    Beginning installation of A629V V2.0 at 07:20

%VMSINSTAL-I-RESTORE, Restoring product saveset A...
```

If VAX/VMS doesn't find the tape or the saveset, check the tape to ensure that it is loaded correctly on the tape drive, and check that the drive is online. VMSINSTAL automatically requests the tape again if the first attempt fails.

5. Answer the following question:

```
* Do you want to run the IVP after the installation [YES]?
```

The default response for this question is YES.

The Installation Verification Program (IVP) automatically tests the TPDRIVER and ACP 629 hardware installation. A YES answer to this question causes VMSINSTAL to initiate the IVP command procedure for all ACP 629s installed during this session. A NO answer causes the installation procedure to skip execution of the IVP when installation is complete.

The advantage of running the IVP after completing installation is that IVP ensures that all pieces of the ACP 629 system have been installed and are responding correctly.

However, it might be undesirable (or impossible) to run the IVP after software installation. If the software package was installed before the hardware, or if more ACP 629s were installed than are physically available, the IVP should not be run automatically after installation. The command procedure SYS$TEST:ACP629T1.COM can be run later to check when the entire ACP 629 system is installed.

6. Answer the following question:

```
* Do you want to purge files replaced by this installation [YES]?
```

The default response to this question is YES.

A YES answer purges old copies of the files that are installed by this process. A NO answer leaves the old copies of the files installed by this process on disk.

If the system disk where the ACP 629 software is to be installed lacks space, you might need to purge all files being replaced by this installation. If the installation fails, you might need to restore the purged files from backup before the system can be put back into service.

If the files are not purged, copies of the driver and command files can accumulate on the system disk after each installation. Leaving the old files on disk eases restoring the system to service if the installation fails.

7. Choose a name for the ACP 629 driver:

```
* Driver Image Filename to use [TPDRIVER]:
```

The default response is TPDRIVER.

This prompt enables you to rename the ACP 629 driver (TPDRIVER) to avoid a filename already in use.

8. Choose a name for the ACP 629 reset program:

```
* Reset Image Filename to use [ACCRESET]:
```

The default response is ACCRESET.

This prompt enables you to rename the ACP 629 reset program (ACCRESET) to avoid a name already in use.

After you answer this prompt, the installation procedure assembles and links the driver:

```
  Assembling Driver
  Linking Driver (ignore transfer address warning)
%LINK-W-USRTFR, image VMI$ROOT:[SYSUPD.A629V020]TPDRIVER.EXE;1
                                        has no user transfer address
End TPDRIVER build
```

9. Choose a prefix for the ACP 629 port names:

```
* Device prefix to use with ACP629 [TP]:
```

The default response is TP.

The installation procedure uses this prefix to build a name for each ACP 629 port. Each name consists of this prefix, followed by a letter denoting which board the port belongs to, followed by a number identifying the specific port. For example if the prefix is TP, the identifier for the fourth port on the second board would be TPB3 and the identifier for the fifteenth port on the first board would be TPA14. You might need to change this prefix if it is in use by another device.

After you enter the prefix, the installation procedure responds with the following messages:

```
Configure hardware ... build SYS$MANAGER:TPCONN.COM
Describe each ACP629 end with ^Z
```

**NOTE**

After you answer the series of prompts in steps 10 through 13 as follows, the installation procedure repeats this series of prompts to allow you to describe each additional ACP 629 installed in your system. You must answer this series of prompts for each ACP 629 installed. After you have described all ACP 629s, press <CNTRL Z> in response to the prompt in step 10 to continue with step 14.

10. The following prompt enables you to enter the CSR address for the ACP 629:

```
* Address of CSR for ACP629 TPA [766740]:
```

The default answer for the first ACP 629 is 766740.

Each ACP 629 requires 40 bytes of UNIBUS address space located on a 40 octal byte boundary. The default answer (766740) is automatically increased by octal 40 each time you repeat this step. If you enter an address different than the default answer, that answer is automatically increased by 40 octal bytes each time you repeat this step. Section 2.3 describes how to set the CSR address on ACP 629 hardware.

11. Enter the interrupt vector address for the ACP 629:

```
* Address of Vector for this ACP629 [410]:
```

The default answer for the first ACP 629 is 410.

Each ACP 629 requires eight contiguous bytes located on a 10 octal byte boundary for its two four-byte interrupt vectors. The default value is automatically increased by 10 octal each time this step is repeated. If you enter an address other than 410, the default answer is automatically increased by 10 octal each time you repeat this step.

12. Enter the ID of the UNIBUS adapter the ACP 629 is plugged into:

```
* ACP629 TPA is used with which Unibus Adapter [UB0]:
```

The default answer is UB0.

13. Choose the number of ports to be configured for outgoing calls:

```
* Number of ports to be configured for outgoing calls [1]: 16
```

The range for this response is 0 to 32. The default response is 1.

The series of prompts in steps 10 through 13 will be repeated to enable you to describe additional ACP 629s being installed.

14. When all ACP 629s have been described, answer the request for a CSR address with <CNTRL>Z:

```
* Address of CSR for ACP629 TPB [767000]: ^Z
```

At this point, the installation procedure builds the installation command file for the ACP 629s described in steps 10 through 13. This file is run automatically when VMSINSTAL is finished. If this configuration is to be loaded each time the system is booted, follow the instructions in the displayed message:

```
You will need to edit SYS$MANAGER:SYSTARTUP to include the line:
      @SYS$MANAGER:TPCONN.COM
```

At this point the installation procedure moves all of the files generated by this session to their final directories (see table 3-1):

```
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
                                                       directories...
```

15. After all the files have been moved, the installation procedure initiates the IVP (if requested).

The command procedure (ACP629T1.COM) that will be used to initiate the installation verification program is listed:

```
Verifying installation of ACP629 using following command file:
$! -----------------------------------
$ IVP := $Sys$test:ACP629T1
$ IVP TPAO
$! -----------------------------------
```

This file contains two parts. The first is the command definition to be used to invoke the installation verification program (ACP629T1.EXE):

```
$ IVP := $Sys$test:ACP629T1
```

The second part invokes the verification program for the first port of each board:

```
$ IVP TPAO
```

The installation verification program requires one argument, the identifier of the port to be tested. The program requires the privileges CMK, PHY_IO, OPER, and SYSPRV to run. The verification program leaves the tested port in PAD mode (ready for an outgoing call).

The verification program assumes the following:

a. At least one port other than the one being tested is ready and waiting for an incoming call (enabled).

b. The ACP 629 being tested is installed with the loopback connector in place. The program does not work if the ACP 629 is connected to a network or if the loopback connector is not installed.

A typical IVP run (with comments) follows.

*Typical IVP Run:*

Beginning of test indication with name of port being tested:

```
Begin Test of Device: tpa0
```

Status request/response to ensure that the ACP 629 is responding and that the physical link is down:

```
status
X.25 Link Down
FREE
```

Disable the ACP 629 and configure it for internal loopback:

```
@1 96 0 0 4 0 0 66 3
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629:

```
@1 96 0 0 1 1
```

Within sixty seconds, the ACP 629 responds

```
@
X.25 Link Down

X.25 Link Up
```

Initiate a call:

```
c
```

Call response:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

Disable the ACP 629 and configure it for external loopback (an external loopback connector must be installed for external connector tests to pass):

```
1 96 0 0 4 0 0 66 1
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629:

```
@1 96 0 0 1 1
```

Within sixty seconds the ACP 629 responds

```
@
X.25 Link Down

X.25 Link Up
```

Initiate a call:

```
c
```

Call response:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

Disable the ACP 629 and turn off loopback mode:

```
1 96 0 0 4 0 0 66 0
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629:

```
@1 96 0 0 1 1
```

End of test with name of port tested:

```
End Test of Device: tpa0
```

If any of these steps fails, a message is printed describing what happened or what was expected and the program aborts. If all verifications complete successfully, VMSINSTAL displays the following messages:

```
Installation of A629V V2.0 completed at 07:28

VMSINSTAL procedure done at 07:28
```

At this point the installation procedure is finished, the software is installed, and the ACP 629 can be connected to the network.

## 3.4 Installation Example

```
Username: SYSTEM
Password:
     Welcome to VAX/VMS version V4.1 on node ACCTST
    Last interactive login on Monday, 19-MAY-1986 07:18
    Last non-interactive login on Sunday, 18-MAY-1986 15:57
     19-MAY-1986 07:19:13    _RTA1:

Hello, user!

$ @sys$update:vmsinstal a629v msa0:

     VAX/VMS Software Product Installation Procedure V4.0

It is 19-MAY-1986 at 07:20.
Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]?

Please mount the first volume of the set on MSAO:.
* Are you ready? y
%MOUNT-I-MOUNTED, A629VO mounted on _MSAO:
The following products will be processed:

  A629V V2.0

     Beginning installation of A629V V2.0 at 07:20

%VMSINSTAL-I-RESTORE, Restoring product saveset A...
* Do you want to run the IVP after the installation [YES]?
* Do you want to purge files replaced by this installation [YES]?
* Driver Image Filename to use [TPDRIVER]:
* Reset Image Filename to use [ACCRESET]:

  Assembling Driver
  Linking Driver (ignore transfer address warning)
%LINK-W-USRTFR, image VMI$ROOT:[SYSUPD.A629V020]TPDRIVER.EXE;1
                                         has no user transfer address
End TPDRIVER build

* Device prefix to use with ACP629 [TP]:
Configure hardware ... build SYS$MANAGER:TPCONN.COM
Describe each ACP629 end with ^Z

* Address of CSR for ACP629 TPA [766740]:
* Address of Vector for this ACP629 [410]:
* ACP629 TPA is used with which Unibus Adapter [UBO]:
* Number of ports to be configured for outgoing calls [1]: 16
* Address of CSR for ACP629 TPB [767000]: ^Z
You will need to edit SYS$MANAGER:SYSTARTUP to include the line:
      @SYS$MANAGER:TPCONN.COM
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
                                                    directories...
```

```
Verifying installation of ACP629 using following command file:
$! ---------------------------------
$ IVP := $Sys$test:ACP629T1
$ IVP TPAO
$! ---------------------------------
Begin Test of Device: tpa0

status
X.25 Link Down
FREE

@1 96 0 0 4 0 0 66 3

@
X.25 Link Down
1 96 0 0 1 1

@
X.25 Link Down

X.25 Link Up
c

PAD Received Answer Message While In CWA State.
0 CONNECTED
1 96 0 0 4 0 0 66 1

@
X.25 Link Down
1 96 0 0 1 1

@
X.25 Link Down

X.25 Link Up
c

PAD Received Answer Message While In CWA State.
0 CONNECTED
1 96 0 0 4 0 0 66 0

@
X.25 Link Down
1 96 0 0 1 1

End Test of Device: tpa0

     Installation of A629V V2.0 completed at 07:28

     VMSINSTAL procedure done at 07:28
$
```

## 3.5 TPCONN.COM File

The TPCONN.COM file is created during driver installation. It contains the commands and definitions needed to bring the ACP 629 online. The file performs the following functions after being created by the installation process:

1. Connects and configures TPDRIVER

2. Defines the VMS-specific commands

3. Configures the host ports

You can modify the TPCONN.COM file to include site-specific commands and operations (section 3.7) after the initial installation of TPDRIVER.

**3.5.1 Driver Connection.** The driver connection section of TPCONN.COM connects TPDRIVER and all of its terminals to the ACP 629. This section requires the CMKRNL privilege. The following example was created by the installation procedure in section 3.4.

```
$ MCR SYSGEN
  Connect TPA0/NUMVEC=2/DRIVER=TPDRIVER/ADAPTER=UB0/CSR=%O766740/VECTOR=%O410
  Connect TPA1/ADAPTER=UB0
  Connect TPA2/ADAPTER=UB0
  Connect TPA3/ADAPTER=UB0
  Connect TPA4/ADAPTER=UB0
  Connect TPA5/ADAPTER=UB0
  Connect TPA6/ADAPTER=UB0
  Connect TPA7/ADAPTER=UB0
  Connect TPA8/ADAPTER=UB0
  Connect TPA9/ADAPTER=UB0
  Connect TPA10/ADAPTER=UB0
  Connect TPA11/ADAPTER=UB0
  Connect TPA12/ADAPTER=UB0
  Connect TPA13/ADAPTER=UB0
  Connect TPA14/ADAPTER=UB0
  Connect TPA15/ADAPTER=UB0
  Connect TPA16/ADAPTER=UB0
  Connect TPA17/ADAPTER=UB0
  Connect TPA18/ADAPTER=UB0
  Connect TPA19/ADAPTER=UB0
  Connect TPA20/ADAPTER=UB0
  Connect TPA21/ADAPTER=UB0
  Connect TPA22/ADAPTER=UB0
  Connect TPA23/ADAPTER=UB0
  Connect TPA24/ADAPTER=UB0
  Connect TPA25/ADAPTER=UB0
  Connect TPA26/ADAPTER=UB0
  Connect TPA27/ADAPTER=UB0
  Connect TPA28/ADAPTER=UB0
  Connect TPA29/ADAPTER=UB0
  Connect TPA30/ADAPTER=UB0
  Connect TPA31/ADAPTER=UB0
EXIT
```

**3.5.2 Command Definitions.** The second part of the default TPCONN.COM command file defines commands that manipulate the states/actions of the ACP 629:

```
$ x25break :== set terminal/speed=75/permanent
$ x25pad :== set terminal/speed=110/nobroad/nomodem/pasthru/permanent
$ x25close :== set terminal/speed=134/permanent
$ x25enable :== set terminal/speed=150/disc/modem/hangup/permanent
$ x25super :== set terminal/speed=1200/nobroad/nomodem/pasthru/permanent
```

SET TERMINAL commands are used to pass specific command information and terminal characteristics to TPDRIVER. The SPEED parameter, which doesn't apply to ACP 629 ports, is used to pass commands to ACP 629 ports. The TPDRIVER interprets the SET TERMINAL/SPEED request as an ACP 629 command and transfers it directly to the ACP 629. Note that /PASTHRU has been combined with the X25PAD and X25SUPER (for outgoing ports) commands, and /DISC/MODEM/HANGUP has been added to the X25ENABLE command (for incoming ports). These are needed for correct operation of terminal emulation programs (such as SET HOST/DTE TPAx:) that use the ACP 629 and for automatic logout or disconnect operations.

These commands require the /PERMANENT qualifier and therefore the PHY_IO privilege to execute them. The following sections detail each of these commands. Normally these commands are issued by the system administrator. Section 3.8 summarizes VMS commands.

**3.5.2.1 X25BREAK.** This command sends an X.25 break packet. The result of this packet is determined by X.3 parameter number 7 (appendix E).

**3.5.2.2 X25PAD.** This command puts a port into PAD mode (section 7.3.3). PAD mode ports allow calls to be made from the local system to another system through the X.25 network.

**3.5.2.3 X25CLOSE.** This commands puts a port into disabled mode (section 7.3.1). In disabled mode, ports cannot initiate or accept calls.

**3.5.2.4 X25ENABLE.** This command puts a port into enabled mode (section 7.3.2). In enabled mode, ports accept incoming calls from the X.25 network.

**3.5.2.5 X25SUPER.** This command puts a port into supervisor mode. Supervisor mode is similar to PAD mode (section 7.3.3) in that it allows outgoing calls to be initiated. Supervisor mode provides an extended command set for administrative functions that PAD mode does not provide.

**3.5.3 Port Configuration.** The last section of the default TPCONN.COM file configures each of the host ports using the commands defined in section 3.5.2.

```
$ x25pad TPA0
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA0
$ x25pad TPA1
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA1
$ x25pad TPA2
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA2
$ x25pad TPA3
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA3
$ x25pad TPA4
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA4
$ x25pad TPA5
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA5
$ x25pad TPA6
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA6
$ x25pad TPA7
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA7
$ x25pad TPA8
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA8
$ x25pad TPA9
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA9
$ x25pad TPA10
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA10
$ x25pad TPA11
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA11
$ x25pad TPA12
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA12
$ x25pad TPA13
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA13
$ x25pad TPA14
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA14
$ x25pad TPA15
$ Set Protection=Owner:RW/Device/Owner_uic=[0,0] TPA15
$ x25enable TPA16
$ x25enable TPA17
$ x25enable TPA18
$ x25enable TPA19
$ x25enable TPA20
$ x25enable TPA21
$ x25enable TPA22
$ x25enable TPA23
$ x25enable TPA24
$ x25enable TPA25
$ x25enable TPA26
$ x25enable TPA27
$ x25enable TPA28
$ x25enable TPA29
$ x25enable TPA30
$ x25enable TPA31
```

This section configures the ACP 629 with the 16 ports as PADs (for outgoing calls) and the remaining 16 ports as host ports (for incoming calls).

Before PAD ports can be accessed by users other than SYSTEM, the protection/ownership of each port must be modified. KITINSTAL does this automatically by adding the following line after each PAD port definition:

```
SET PROTECTION=OWNER:RW/DEVICE/OWNER_UIC=[0,0] TPA<n>
```

This causes the device TPA<n> (<n> = port number) to have no specific owner until it is assigned. Once assigned, the user has read/write access.

## 3.6 ACCRESET

ACCRESET.EXE is used to reset the ACP 629 if it gets hung. The commands that invoke ACCRESET.EXE follow:

```
MCR ACCRESET TPA0
MCR ACCRESET TPA0/RESET
```

Either command resets the ACP 629. The RESET option of ACCRESET is the only option that works on the ACP 629. All other options for ACCRESET (described in other ACC manuals) are not allowed and do not work on the ACP 629.

**NOTE**

VMS does not allow a port with the /MODEM option set to be reset.

## 3.7 Changing TPCONN.COM

To change device or port configuration parameters to fit local needs (and to keep the changes through system reboots), you must modify the file TPCONN.COM. For example, you can modify TPCONN.COM to enable the VMS DISCONNECT feature. This feature allows processes to be disconnected rather than terminated when an X.25 call is cleared. Enabling the VMS DISCONNECT feature prevents loss of work caused by network failures. To enable VMS DISCONNECT, add the following line to the TPCONN.COM file after the line `MCR SYSGEN`.

```
CONNECT VTA0/NOADAPTER/DRIVER=TTDRIVER
```

See the topic "Disconnected Terminals" in the *VAX/VMS I/O User's Reference Manual: Part I* for more information on the VMS DISCONNECT feature.

You can also change the TPCONN.COM file to automatically set port or PAD parameters at initialization. An example change to TPCONN.COM to allow automatic setup of an X.3 parameter suite and several other PAD parameters follows:

```
$       X25SUPER TPAO
! set supervisor mode on TPAO:
!
$       SET TERM/NOTYPEAHEAD/NOWRAP/PERM TPAO
! setup the port so than echoed characters and responses are
! discarded (/NOTYPEAHEAD) and so that all command strings are
! sent without being broken up (/NOWRAP) if they are longer than
! 80 characters.
!
! perform commands by copying them to tpa0:
!
$       COPY SYS$INPUT TPAO:
X 1:1 2:0 3:127 4:0 5:0 6:5 7:8 8:0 9:0 10:0 12:0 13:0 14:0 15:0
F N
L 96 0 0 4 0 76 187 1
$       SET TERM/TYPEAHEAD/WRAP/PERM TPAO
! reset terminal attributes.
!
$       X25CLOSE TPAO
! disable supervisor mode on TPa0:
!
$       X25PAD TPAO
! enable TPa0 for outgoing calls
```

In the above example, the line

```
X 1:1 2:0 3:127 4:0 5:0 6:5 7:8 8:0 9:0 10:0 12:0 13:0 14:0 15:0
```

causes the X.3 parameters (appendix E) sent by the ACP 629 upon receipt of an incoming call to be set.

The line

```
F N
```

turns off facilities checking.

The line

```
L 96 0 0 4 0 76 187 1
```

sets the bit rate to 56 kbits/sec.

Note that the X command can be up to 127 characters but must not contain an embedded <CR>. An embedded <CR> is interpreted as the end of the command. See chapter 7 or 8 for information on other commands that can be put into the TPCONN.COM file.

The attributes /NOWRAP and /NOTYPEAHEAD serve a special purpose in the above example. The attribute /WRAP specifies that VMS is to insert a <CR> at the end of a line sent to a terminal device. (Line length is set by the /WIDTH attribute). This means that a command (for example, X) that is longer than /WIDTH characters (typically 80 for the ACP 629) is truncated to 80 characters by the insertion of a <CR> when written to the ACP 629. The rest of the command is also sent to the ACP 629 but will most likely be treated as an error.

The attribute /NOWRAP ensures that all commands are sent to the ACP 629 without interference from VMS. Note that you can also do this by setting the attribute /WIDTH to greater than the longest command in the command file. However, the attribute /NOWRAP is not effected by changes in the length of command lines.

The attribute /NOTYPEAHEAD disables the type-ahead buffer. This is necessary because the ACP 629 echos all characters sent to it and some commands cause responses. When the type-ahead buffer is enabled, VMS places each character echoed by the ACP 629 into the type-ahead buffer. When the type-ahead buffer becomes full, the ACP 629 becomes flow-controlled. Because there is no read outstanding against the port (copy issues writes only aginst the destination device) the ACP 629 remains hung until another user or process issues a read against the suspended port. When this happens, all the remaining data is sent to the process issuing the read. With the attribute /NOTYPEAHEAD set, all echoed data and generated responses are discarded.

## 3.8 VMS Quick Reference

| Function | Command |
|---|---|
| Connect to the ACP 629 in order to place an outgoing call. | `SET HOST/DTE TP<n>` |
| Return to VMS from the ACP 629. | `<CNTRL>\` |
| Reset the ACP 629 if it is hung. | `MCR ACCRESET TP<n>` |
| Disable a port. | `X25CLOSE TP<n>` |
| Transmit an X.25 break packet. | `X25BREAK TP<n>` |
| Enable a port for outgoing calls. | `X25PAD TP<n>` |
| Enable a port for incoming calls. | `X25ENABLE TP<n>` |
| Enable a host port for supervisor commands. | `X25SUPER TP<n>` |

TP<n> = Terminal identifier (e.g., TPA14). The prefix TP might be different because of site naming conventions.

Command definitions are in section 3.5.2.

# Chapter 4

# UNIX System III Driver Installation

## 4.1 Introduction

This chapter describes how to install the ACP 629 driver on a DEC VAX or DEC PDP-11/70 that runs the UNIX System III Operating System.

## 4.2 Software Components

The ACP 629 driver distribution kit for UNIX System III contains the the following files:

| | |
|---|---|
| tj.c | - UNIX System III driver |
| tj.7 | - unformatted manual page for the driver |
| tj7.txt | - formatted manual page for the driver. This supplements section 7 in the *UNIX Administrator Reference Manual.* |
| setparm.c | - utility program that uses a new IOCTL system call named TCSPARM. TCSPARM enables you to request output of a string of byte pairs over an open ACP 629 connection as a Q-bit data packet for setting X.3 parameters. |
| setparm.1 | - unformatted manual page for the setparm program. (The installation procedure creates the setparm program.) |
| setparm1.txt | - formatted manual page for the setparm program. It supplements section 1 in the *UNIX User Reference Manual.* |
| setx29 | - Bourne shell script that gives simplified syntax to invoke the setparm program. |

## 4.3 Installation Procedure

**CAUTION**

Before doing the following procedure, ensure that an up-to-date backup of the kernel build area exists. Create one if needed. When adding files to the system, rename existing files that have the same name.

In the following examples the device used is /dev/rmt0 and the distribution medium is a magnetic tape. If the name in the /dev directory differs on your system or if you received the driver on floppy diskette, adjust the command lines. To extract the files from the distribution medium, do the following steps.

1. Login as root.

2. cd to a directory that will temporarily be the location for the tape contents. We recommend using a directory in "/tmp". For example, create a directory called "/tmp/acp629".

   ```
   tar xvbf 20 /dev/rmt0
   ```

   cd to the driver's target directory /usr/src/uts/io

   ```
   mv /tmp/acp629/driver/tj.c .
   ```

3. Choose the target directory for the setparm program (suggestion: /etc).

   cd to the setparm program's target directory.

   ```
   mv /tmp/acp629/util/setparm.c .
   ```

4. If you wish to format the manual pages (formatted versions are contained in the distribution), issue the following commands:

   cd to the directory that contains formatted section 1 manual pages.

   ```
   nroff -man /tmp/acp629/doc/setparm.1 > setparm.1
   ```

   cd to the directory that contains formatted section 7 manual pages.

   ```
   nroff -man /tmp/acp629/doc/tj.7 > tj.7
   ```

   cd to the directory that contains unformatted UNIX section 1 manual pages.

   ```
   mv /tmp/acp629/doc/setparm.1 .
   ```

   cd to the directory that contains unformatted UNIX section 7 manual pages.

   ```
   mv /tmp/acp629/doc/tj.7 .
   ```

5. If you want to copy the formatted manual pages directly from the distribution medium to the appropriate directories, do as follows:

   cd to the directory that contains formatted section 1 manual pages.

```
mv /tmp/acp629/doc/setparm1.txt setparm.1
```

cd to the directory that contains formatted section 7 manual pages.

```
mv /tmp/acp629/doc/tj7.txt tj.7
```

6. Choose the target directory for the setx29 shell script (suggestion: /etc). cd to the target directory.

```
mv /tmp/acp629/util/setx29 .
chmod 555 setx29
```

7. Clean up as follows:

```
cd /tmp/acp629
rm -r *
cd ..
rmdir acp629
```

Do the following steps to complete the installation procedure:

8. Arrange comment delimiters and #defines in the driver source file, tj.c, for the appropriate CPU type as follows:

```
UNIX System III on PDP 11/70:
     #define TJPDP1170
     #define TJSYSIII  3.0
     /*
     #define TJV2 5.2
     #define TJDEBUG   1
     */

UNIX System III on a VAX:
     #define TJSYSIII  3.0
     /*
     #define TJPDP1170
     #define TJV2 5.2
     #define  TJDEBUG  1
     */
```

If more than one ACP 629 is installed, change the #define for NTJS to the number of ACP 629 boards installed.

Change the #define for ZVECTOR to the interrupt vector for the first ACP 629, as specified in the dfile described in step 11.

9. The makefile is /usr/src/uts/io/io_p2.mk. Modify the makefile to add tj.o to FILES as follows:

```
FILES =\
     $(LIBNAME)(sys.o)\
     $(LIBNAME)(tj.o)\
     $(LIBNAME)(tm.o)\
```

10. Add the dependencies of the tj driver exactly as follows:

```
$(LIBNAME)(tj.o):\
     $(INCRT)/sys/param.h\
     $(INCRT)/sys/dir.h\
     $(INCRT)/sys/user.h\
     $(INCRT)/sys/errno.h\
     $(INCRT)/sys/file.h\
     $(INCRT)/sys/tty.h\
     $(INCRT)/sys/conf.h\
     $(INCRT)/sys/proc.h\
     $(INCRT)/sys/uba.h\
     $(FRC)
```

11. Change the dfile in /usr/src/uts/cf to include the following line once for each ACP 629 installed. (Separate fields by tabs.)

```
tj11      320       766740  5
```

Change the field containing 320 to an unused octal interrupt vector with eight bytes available. If more than one ACP 629 is present, 8 * number_of_ACP629s bytes must be available and subsequent ACP 629s' interrupt vectors must follow the first at 8-byte intervals. For example, a second ACP 629 would have interrupt vector 330 in the case shown. To find available interrupt vectors, examine the same column of all other dfile device entries.

The field containing 766740 contains the unique UNIBUS octal address for each ACP 629. The UNIBUS address must be on a 40 octal byte boundary and must have 20 octal bytes of space available. Before installation, make sure this location is not in use by another device. The field containing 5 is the bus request level.

12. Change the file /etc/master to add the following line. (Separate fields with tabs.)

```
tj11   8    37    6    tj    8    0    12    32    5
```

Change the field containing 12 to an unused major device number (different from the numbers in the same column of all other master file entries). See appendix C for more information on major device numbers. See master(4) in the *UNIX Programmer's Manual* for the format of the /etc/master file.

13. Ensure that your current directory is /usr/src/uts/cf and run

```
config -t <dfile_name>
```

`<dfile_name>` is the name of the modified dfile. Confirm the tj11's major device number with the table displayed by config.

For VAX systems, edit the file univec.c (made by running config above) and change all tjxint references to tjrint (deleting the redundant extern definition this creates). Specifically, change

```
extern tjrint(), tjxint();
```

to

```
extern tjrint();
```

And change

```
(int *)((int)tjrint+DO),
(int *)((int)tjxint+DO),
```

to

```
(int *)((int)tjrint+DO),
(int *)((int)tjrint+DO),
```

For PDP-11/70 systems, edit the file low.s (made by running config above) and change all _tjxint references to _tjrint (deleting the redundant global definition this creates). Specifically, change

```
.globl   _tjrint, _tjxint
tjin:
         jsr      r0,call; _tjrint
tjou:
         jsr      r0,call; _tjxint
```

to

```
.globl   _tjrint
tjin:
         jsr      r0,call; _tjrint
tjou:
         jsr      r0,call; _tjrint
```

**NOTE**

Every time config is run as described above, the file univec.c (for VAX) or low.s (for PDP-11) must be edited as described above.

14. Ensure that your current directory is /usr/src/uts/cf. To build a new UNIX kernel named /usr/src/uts/<SYS><VER>, edit the file /usr/src/uts/cf/Makefile (which is linked to the file cf.mk) and specify your preferred kernel name. For example, specify SYS as unix and VER as acc to name the kernel unixacc.

    On a PDP-11/70 only, edit lines in Makefile as follows:

    ```
    set "TYPE = id"
    set "CPU = 70"
    ```

    Do the following for either CPU:

    ```
    set "NODE = unix"
    ```
    (or some other reasonable node name).

    After the Makefile is altered, run **make**. Examine the output of the make to confirm that the specified options were recognized.

15. Regardless of CPU type, save the previous /unix under another name (or pick another name for the new UNIX), then copy the new UNIX (/usr/src/uts/<SYS><VER>) to /unix. Or copy your preferred name to the root directory. To avoid problems when you shut down multiuser UNIX, replace the file /unix only when running in single-user mode.

16. Create devices in /dev with the appropriate major/minor device numbers. Use major device numbers as you specified them when changing the master file. (These numbers are echoed in the table displayed by config.) Specify eight-bit minor device numbers as follows:

    PBBLLLLL

    where:

    ```
    P     = 0, inbound (HOST) port
          = 1, outbound (PAD) port
    BB    = board number
    LLLLL = line number on the board.
    ```

    See appendix C for more information on major and minor device numbers.

17. Choose names for HOST ports that begin with the prefix tty and that don't conflict with any existing terminal device names. For example, use tty60 through tty91.

The following examples, using mknod(1M), use major device number 12 and board number 0. HOST (incoming) ports have device names prefixed with tty. PAD (outgoing) ports have device names prefixed with pad:

```
/etc/mknod /dev/tty60 c 12 0
/etc/mknod /dev/tty61 c 12 1
      .
      .
/etc/mknod /dev/tty90 c 12 30
/etc/mknod /dev/tty91 c 12 31

/etc/mknod /dev/pad00 c 12 128
/etc/mknod /dev/pad01 c 12 129
      .
      .
/etc/mknod /dev/pad30 c 12 158
/etc/mknod /dev/pad31 c 12 159
```

If a port will only be used in a single mode, the /dev entry for the unused mode need not be created.

18. Edit /etc/inittab to bring up gettys as needed, adding entries such as the following. (See inittab(4) in the *UNIX Programmer's Manual.)*

```
2:33:c:/etc/getty tty61 5
```

**NOTE**

When ports are enabled in /etc/inittab, init opens the device for incoming calls and prevents use of that port for PAD mode.

19. Ports that are to be used as PAD (outgoing) ports can be accessed by the cu command. After a connection is made, sending <CR> gives the PAD prompt (@). Refer to chapter 7 for the commands available in PAD mode. Use `<CR>~.<CR>` to leave PAD mode.

If appropriate entries such as

```
DIR pad23 0 9600
```

are added to the file /usr/lib/uucp/L-devices, the command

```
cu -lpad23 dir
```

can be used to access the port /dev/pad23.

**NOTE**

> The ports specified in /etc/inittab must not be specified in L-devices because cu will not be able to open them.

20. When the driver has been installed and the new devices created, boot the new UNIX kernel.

## 4.4 Setparm Compilation

The ACP 629 driver supports a custom ioctl, TCSPARM, which can be used to select a string of byte pairs to be output as a data packet with the Q-bit set. Typically, this output can be used to set selected X.3 PAD parameters. See appendix E for a description of PAD Parameters. The setparm command issues a TCSPARM ioctl to the ACP 629 driver. The syntax of the setparm command is given in the UNIX manual page, setparm(1), included in this distribution. ACC provides a shell script, setx29, that invokes the setparm command with simplified syntax. Steps 3 and 6 of section 4.3 described how to install the setparm program and the setx29 shell script.

Compile the setparm program as follows:

1. `cd` to the target directory for the setparm program. (Refer to step 3 of section 4.3 for the location of the target directory.)

2. `cc setparm.c -o /etc/setparm`

## 4.5 Installation Verification Procedure

After installation, do the following steps to check ACP 629 installation.

1. Install the provided loopback connector in the distribution panel slot (see 2.9).

2. Login to root. (This enables access to the ACP 629 in supervisor mode.) Connect to an ACP 629 PAD port using the command

   ```
   cu -lpad<nn> dir
   ```

3. Press `<CR>`. This causes the ACP 629 to display the command prompt `@`. If it does not, review the driver installation procedures and ensure the device being accessed is configured as a PAD port.

4. Enter the following:

```
STATUS
```

The ACP 629 responds

```
X.25 link down
FREE
```

5. Disable the ACP 629 and set it for internal loopback as follows:

```
L 96 0 0 4 0 0 66 3
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629 as follows:

```
L 96 0 0 1 1
```

Within sixty seconds, the ACP 629 responds

```
@
X.25 Link Down

X.25 Link Up
```

If these messages are not displayed within 60 seconds, check installation of the distribution cables.

Initiate a call as follows:

```
c
```

The call response follows:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

6. Disable the ACP 629 and configure it for external loopback as follows. (An external loopback connector must be installed for external connector tests to pass):

```
L 96 0 0 4 0 0 66 1
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629 as follows:

```
L 96 0 0 1 1
```

Within sixty seconds the ACP 629 responds

```
@
X.25 Link Down

X.25 Link Up
```

If these messages aren't displayed, check installation of the distribution cables and loopback connector. If all expected responses were received, the ACP 629, cables, and device driver are correctly installed.

Initiate a call as follows:

```
c
```

The call response follows:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

7. If steps 1-6 succeed, test the setparm program by modifying an X.3 PAD parameter. (The loopback connector must remain connected during this phase.) To modify an X.3 parameter, do the following:

   a. At the @ prompt, enter

      ```
      C<CR>
      ```

      This makes a loopback connection to an incoming ACP 629 Host port.

   b. Login as any user.

   c. Escape back to the PAD by pressing `<CNTRL>P`

   d. Press `<CR>`. This causes the ACP 629 to display @.

   e. Enter `par?`

      A list of the current value of each the 18 X.3 PAD parameters is displayed. The value of each parameter is separated from its parameter number by a colon. Appendix E describes each parameter.

   f. Reconnect to the Host port by entering `C<CR>` at the @ prompt.

   g. Press `<CR>` at the `RECONNECTED` message.

h. Use the setx29 shell script to modify the X.3 PAD parameters. For example, if PAD parameter 14 equals 0, change it to 1 by doing the following:

   `cd` to the directory that contains the setx29 shell script. Then enter

   ```
   ./setx29 14:1
   ```

   If PAD parameter 14 equalled 1 before issuing this command, change parameter 14 to a value other than 1 (by entering a value other than 1 in the setx29 command).

i. A message is displayed stating that the ioctl completed. Escape to the PAD again to check the value of the X.3 PAD parameter you just changed:

   ```
   <CNTRL>P
   ```

j. Enter <CR>. This causes the ACP 629 to display @.

k. To make sure that the parameter has been changed correctly, enter the following at the @ prompt:

   ```
   par?
   ```

   Examine the list displayed by this command for the appropriate parameter value (to make sure the value was changed correctly).

8. Remove the loopback connector and install the X.25 trunk line connector. This trunk line can be attached to one of many X.25 devices such as a network modem, a stand-alone X.25 switch or PAD, or another ACP 629.

9. Disable the X.25 link and configure the ACP 629 for network operations by entering

   ```
   L 96 0 0 3 0 66 0
   ```

10. Then re-enable the link layer by entering

    ```
    L 96 0 0 1 1
    ```

    Within 60 seconds the ACP 629 responds

    ```
    X.25 link up
    ```

    If not, review the problems described in appendix A and try again.

## 4.6 Usage Notes

1. The constant NTJLNS (defined in the driver) is the number of lines defined per ACP 629. Minor devices 0 to (NTJLNS-1) map to the first ACP 629, minor devices NTJLNS to (2*NTJLNS-1) map to the second and so on.

2. The UNIX driver uses a kernel variable array to determine the addresses of the ACP 629 boards. Although the addresses for the ACP 629 need not be contiguous, take care that each device has a full 32-byte area that does not overlap the area of any other device.

3. If the COMREGs are not available during the first open on a tj unit (board) or during an attempt to write to the tj unit the following message is displayed on the UNIX console:

   ```
   acp629: unit <n> not responding to <request>.
   ```

   Further attempts to access any port on that tj unit returns an immediate error, with errno = ENXIO.

   This condition can be cleared by opening the tj minor device number (252 + unit_number), that is, minor 252, 253, 254, or 255 for units 0, 1, 2, or 3. This sends a SIGHUP signal to all process groups associated with open ports on that unit, and immediately returns an error to all processes awaiting completion of an open on a port on that ACP 629 unit. Note that this convention makes the last four PAD ports on the fourth ACP 629 unavailable; they can still be used as HOST ports.

   The next open of any port on the affected ACP 629 unit causes an attempt to reset the unit and resume normal operation. If this attempt fails, the unit will again be marked as offline, continually returning errno ENXIO as above.

   One way to do the clear operation follows. Create the necessary device(s) using mknod(1M):

   ```
   mknod /dev/tjclr0 c 9 252
   mknod /dev/tjclr1 c 9 253
                .
                .
   ```

For example, run

```
cat < /dev/tjclr0
```

to do the ACP 629 clear operation on unit 0. It exits with an open error and EPIPE errno (returned, by convention, by the device driver to any process with an open pending on an ACP 629 that is cleared).

Change the /dev/tjclr<n> devices' modes to allow access only by root to prevent unauthorized users from clearing the ACP 629.

4. A diagnostic tool is available for printing port statistics for board 0, including the state of each active process. This tool can be used when up to three ACP629 boards are in a system (boards 0, 1, and 2).

A port must be configured in the /dev directory. For example,

```
/etc/mknod /dev/tjshow c 12 255
```

could be used for major device number 12.

You can get port status for board 0 by issuing an open on minor device 255 (/dev/tjshow):

```
cat < /dev/tjshow
```

The following message results:

```
acp629: state of unit ?
line state    iflag    oflag    cflag    lflag
```

where ? represents the ACP 629 board number.

The state of each active port is listed below this message.

The message

```
remaining lines free
```

indicates that all ports that were not included in the previous message are inactive.

## 4.7 UNIX System III Quick Reference

| Function | Command or Procedure |
|---|---|
| Connect to the ACP 629 to place an outgoing call. | `cu -lpad<nn> dir` |
| Return to UNIX System III from the ACP 629. | `<CR>~.<CR>` |
| Reset the ACP 629 if it is hung. | The ACP 629 is automatically reset upon the first open after a reboot. You can reset the ACP 629 by opening one of the configured clear devices. |
| Disable a port. | A port is automatically disabled upon the final close. |
| Transmit an X.25 break packet. | Use the ioctl command TCSBRK. See tty(4). |
| Enable a port for outgoing calls. | Create the appropriate PAD-port device name/number (with the most significant bit of the minor device number set) using mknod(1). Ensure the device name is not in /etc/inittab. |
| Enable a port for incoming calls. | Create the appropriate HOST-port device name/number (with the most significant bit of the minor device number clear) using mknod(1). Ensure that the corresponding HOST-port device name is included in /etc/inittab. |
| Enable a port for supervisor commands. | Automatically granted to root. |

# Chapter 5

# UNIX 4.2 BSD, UNIX 4.3 BSD, and ULTRIX-32 Driver Installation

## 5.1 Introduction

This chapter describes how to install the ACP 629 device drivers for UNIX 4.2 BSD, UNIX 4.3 BSD, and ULTRIX-32 V1.2 systems.

## 5.2 Software Components

The ACP 629 terminal driver distribution kit for UNIX 4.2 and 4.3 BSD and ULTRIX-32 V1.2 systems contains the following files:

| | |
|---|---|
| xb.c | - driver source code |
| xbreg.h | - driver source code |
| readme.txt | - installation instructions |
| xb.4 | - source file (UNIX manual page) |
| xb4.txt | - formatted manual page for visual reference |

**NOTE**

Before doing the following procedure, ensure that an up-to-date backup of the kernel build area exists. Create one if needed. When adding files to the system, rename existing files that have the same name.

Extract the files from the distribution medium as follows:

`cd` to a temporary storage directory (e.g., /usr/tmp).

```
tar xvbf 20 /dev/rmt0 .
```

To format the xb.4 file to add it to the *UNIX Programmer's Manual,* issue the following command:

```
nroff -man xb.4 > xb4.txt
```

The xb4.txt file must be copied to the directory containing the section 4 formatted manual page. If the system supports troff processing, use `troff` instead of `nroff` in the preceding command.

## 5.3 Preparation

The following installation instructions are for UNIX 4.2 BSD, UNIX 4.3 BSD, and ULTRIX-32 V1.2 source or binary distributions.

1. Login under the root account.

2. Copy the driver source files to the system source directories as follows. If previous versions already exist, rename them with .old as the suffix.

   ```
   cp xb.c /sys/vaxuba/xb.c
   cp xbreg.h /sys/vaxuba/xbreg.h
   ```

   For 4.3 BSD installations, edit the file xb.c to activate the following line by removing its comment delimiter:

   ```
   #define XB_43BSD
   ```

## 5.4 Hardware Configuration

Change the ACP 629 CSR address as needed. The standard ACP 629 address is 0766740, which is translated to 0166740 for a 16-bit rather than 18-bit representation. If multiple ACP 629s are used, they must have their addresses reassigned. The selected CSR addresses must be unique and aligned on a 40 octal-byte boundary. The three CSR addresses defined in xb.c are 0166740, 0167000, and 0167040 for three ACP 629s.

## 5.5 Installation Procedure

Refer to the document *Building 4.2* (or 4.3) *BSD UNIX Systems with Config* in the *UNIX System Manager's Manual* as you perform the following steps:

1. The following files must be modified to define the configuration of a new system that incorporates the ACP 629 driver. If an xb driver has already been configured in the system, add comment delimiters to existing lines or delete them, or rename the files. Modification of the following files is described later.

   /sys/conf/files.vax
   /sys/conf/ACPTST (or the name of the system configuration file)
   /sys/vax/conf.c

2. Add the following line to /sys/conf/files.vax:

   ```
   vaxuba/xb.c      optional xb device-driver
   ```

3. Change /sys/vax/conf.c as follows:

   a. Add the following lines:

```
#include "xb.h"
#if NXB > 0
int xbopen(),xbclose(),xbread(),xbwrite();
int xbioctl(),xbreset(),xbstop(), xbselect();
struct tty xb_tty[];
#else
#define xbopen nodev
#define xbclose nodev
#define xbread nodev
#define xbwrite nodev
#define xbioctl nodev
#define xbreset nulldev
#define xbstop nodev
#define xbselect nodev
#define xb_tty 0
#endif
```

**NOTE**

Previous versions of the ACP 629 driver were named xf instead of xb. If an xf version is being changed to an xb version, the line `#if NXB > 0` must be present in the xb version.

   b. Make an entry in cdevsw for an unused major device number. The following example uses major device number 33. The position in the cdevsw table is the corresponding major device number. Refer to appendix C for more information on major and minor device numbers.

```
struct cdevsw cdevsw[] =
{
      .
      .
      .
/*25-29 reserved to local sites */
      .
      .
      .
xbopen,     xbclose,    xbread,    xbwrite,    /* 33 */
xbioctl,    xbstop,     xbreset,   xb_tty,
xbselect,   nodev
      .
      .
      .
};
```

4. Modify the configuration file in /sys/conf to include the ACP 629 driver. Add a line as follows:

```
device xb0  at uba<n>  csr  0166740  vector  xbint  xbint
```

Note that xbint occurs twice. The field `uba<n>` is the number of the UNIBUS adapter in which the board is installed. Adjust the CSR address 0166740 to match the CSR address set on the board that this entry describes.

5. Do a sysgen as described in the *UNIX 4.2* (or 4.3) *BSD System Manager's Manual.* The following commands are an example (not for a specific system).

```
cd /sys/conf
mkdir ../ACPTST
config ACPTST
cd ../ACPTST
make depend
cp /vmunix /vmunix.old
make vmunix
cp vmunix /vmunix
```

6. Create devices in /dev with the appropriate major and minor device numbers. The major device number is defined when an entry is made in the cdevsw table in /sys/vax/conf.c (major device number 33 was shown in the previous example). Minor device numbers have the following eight-bit format:

```
PBBLLLLL
```

where:

```
    P     = 0, inbound (HOST) port
    P     = 1, outbound (PAD) port
    BB    = board number
    LLLLL = line number on the board.
```

The following examples, using mknod(8), assume major device number 33 and board number 0.

```
/etc/mknod /dev/tty0 c 33 0
/etc/mknod /dev/tty1 c 33 1
    .
    .
/etc/mknod /dev/tty30 c 33 30
/etc/mknod /dev/tty31 c 33 31

/etc/mknod /dev/pad0 c 33 128
/etc/mknod /dev/pad1 c 33 129
    .
    .
/etc/mknod /dev/pad30 c 33 158
/etc/mknod /dev/pad31 c 33 159
```

If a port will only be used in a single mode, the /dev entry for the unused mode need not be created.

The shell script, /dev/MAKEDEV, can be modified to do the previous steps. For further explanation of device numbers, see appendix C.

7. Make entries for terminal initialization in /etc/ttys for ports that are to accept incoming calls. (ttys(5) defines the format for these entries.)

   For 4.2 BSD/ULTRIX, add the following lines:

```
12ttyx0
12ttyx1
    .
    .
```

   For 4.3 BSD, add the following lines:

```
ttyx0     "/etc/getty std.9600"     unknown on"
ttyx1     "/etc/getty std.9600"     unknown on"
```

   **NOTE**

   When ports are enabled in /etc/ttys, "init" opens the device for incoming calls and prevents use of that port for PAD mode.

8. When the system is rebooted from the new kernel, auto-configuration reports the available xb devices.

9. To make a full-duplex connection to a remote system (with seemingly a direct connection), use tip (described in tip(1C) of the *UNIX Programmer's Manual).*

   Before using tip with the xb driver, modify the tip source file (tip.c) as described below. This change causes tip to restore terminal characteristics (a) after returning to the local system upon disconnection from a remote PAD or (b) upon an xb driver clear operation. (Clear resets the ACP 629 from the driver.) Without this change, the terminal must be reset manually with a reset command or tset command to restore the original terminal characteristics. For example, enter **reset** followed by a line feed (not a carriage return).

   Find the directory that contains the tip sources. The 4.2 and 4.3 BSD UNIX distribution standard directory is /usr/src/usr.bin/tip.r. This directory contains a Makefile and the tip.c source module. First, rename the binary file tip in /usr/bin to tip.old. Then rename the source file tip.c to tip.c.old. Add the following line to tip.c:

```
extern int finish();
```

   One SIGHUP signal call in tip.c should be changed to do a finish instead of a cleanup. Change the line in the main() routine under the notnumber label as follows:

Change the line

```
signal(SIGHUP, cleanup)
```

to

```
signal(SIGHUP, finish)
```

For ULTRIX only, the file tipout.c must be altered similarly. Add the line

```
extern int finish();
```

Then one SIGHUP signal call in tipout.c that must be changed to do a finish instead of intTERM. Change one line in the tipoutinit() routine as follows:

Change

```
signal(SIGHUP, intTERM)            /* for dial-ups*/
```

to

```
signal(SIGHUP, finish)             /* for dial-ups*/
```

Finally, for BSD UNIX and ULTRIX, do the following. Ensure that /usr/src/usr.bin/tip/tip (or its equivalent) is the current directory. Enter **make** to rebuild the tip executable file. When the makefile has been successfully run, copy the new tip file to /usr/bin:

```
cp /usr/src/usr.bin/tip/tip /usr/bin/tip
```

Alternatively, if you are unable to recompile the tip command, you can spawn another shell to run tip, causing the terminal state to be restored after disconnecting from a remote PAD connection. For example, the following shell procedure can be created and named x25 (under the C shell):

```
#
tip PAD
```

10. Before tip can be used, the file /etc/remote must be modified (see remote(5) of the *UNIX Programmer's Manual)* by inserting lines with the following format.

```
PAD0:dv=/dev/pad20
PAD1:dv=/dev/pad21
PAD2:dv=/dev/pad22
    .
    .
    .
PAD30:dv=/dev/pad31
```

Also, inserting a line such as

```
PAD:dv=/dev/pad20,/dev/pad21,/dev/pad22...
```

enables the TIP command to automatically select the next unused PAD port from those specified. Make sure that the ports named by the entries in /etc/ttys and /etc/remote are mutually exclusive: if a port is enabled in /etc/ttys, init opens the device for incoming calls and prevents use of that port for PAD mode.

11. As noted in the UNIX manual for tip(1), the usr/spool/uucp directory contains any lock files that might be needed to avoid conflicts with uucp (1C). Check that the usr/spool/uucp directory on the system is available and readable (i.e., mode 755). You might need to execute CHMOD(1) to make it accessible. Check that the /dev/pad<nn> devices are owned by uucp. If these devices are not owned by uucp, use chown (e.g., chown uucp /dev/pad0).

12. Shut down the system and boot the new /vmunix. Login under the root account.

13. Enter the following command:

    ```
    tip PAD<CR>
    ```

    When `connected` is displayed, press <CR> again and the `@` prompt displays. At this point, the terminal is connected directly to the ACP 629 and ready to initiate an outbound call (the port is in PAD mode). See chapter 7 for the commands available in this state. To exit from tip, enter the following:

    ```
    <CR>~.
    ```

    or

    ```
    <CR>~<CNTRL>D
    ```

14. If the COMREGs are not available during the first open on an xb unit (ACP 629) or during an attempt to write to the xb unit an error message displays on the UNIX console. (The diagnostic messages for the ACP 629 are described on the xb(4) manual page shipped on the xb driver distribution medium.) Further attempts to access any port on that xb unit return the error code ENXIO.

This condition can be cleared by doing an open of xb minor device number (252 + unit_number), that is, minor device number 252, 253, 254, or 255 for units 0, 1, 2, or 3. The open operation causes the sending of a SIGHUP signal to all process groups associated with open ports on that unit. The open operation also causes an immediate error return to all processes awaiting completion of an open on a port on that ACP 629 unit. This procedure makes the last four PAD ports on the fourth ACP 629 unavailable, but they can still be used as HOST ports.

The next open of any port on the affected ACP 629 causes an attempt to reset the unit and resume normal operation. If this attempt fails, the unit is again marked as offline, continually returning error code ENXIO.

One way to do the clear operation follows. Create the necessary devices(s) using mknod:

```
mknod /dev/xbclr0 c 33 252
mknod /dev/xbclr1 c 33 253
     .
     .
```

Run `cat < /dev/xbclr0`, for example, to perform the ACP 629 clear operation on unit 0. The clear operation exits with an EPIPE open error (returned by convention by the device driver).

15. Change the modes of the /dev/xbclr<n> devices to allow access only by root (to prevent others from resetting the ACP 629).

## 5.6 Installation Verification Procedure

Do the following steps to check installation of the ACP 629.

1. Install the provided loopback connector in the distribution panel slot (see 2.9).

2. Login to the root account (to access the ACP 629 in supervisor mode) and connect to an ACP 629 PAD port using the command

   ```
   tip PAD
   ```

3. Press <CR>. This causes the ACP 629 to display the command prompt @. If this prompt is not displayed, review the driver installation procedures and check that the device being accessed is configured as a PAD port.

4. Enter the following:

   ```
   STATUS
   ```

   This causes the ACP 629 to display

   ```
   X.25 link down
   FREE
   ```

5. Disable the ACP 629 and set it for internal loopback as follows:

   ```
   L 96 0 0 4 0 0 66 3
   ```

   The ACP 629 responds

   ```
   @
   X.25 Link Down
   ```

   Enable the ACP 629 as follows:

   ```
   L 96 0 0 1 1
   ```

   Within sixty seconds, the ACP 629 responds

   ```
   @
   X.25 Link Down

   X.25 Link Up
   ```

   If these messages are not displayed within 60 seconds, check installation of the distribution cables.

   Initiate a call as follows:

   ```
   c
   ```

   The call response follows:

   ```
   PAD Received Answer Message While In CWA State.
   0 CONNECTED
   ```

6. Disable the ACP 629 and configure it for external loopback as follows. (An external loopback connector must be installed for external connector tests to pass):

   ```
   L 96 0 0 4 0 0 66 1
   ```

   The ACP 629 responds

   ```
   @
   X.25 Link Down
   ```

   Enable the ACP 629 as follows:

   ```
   L 96 0 0 1 1
   ```

Within sixty seconds the ACP 629 responds

```
Q
X.25 Link Down

X.25 Link Up
```

If these messages aren't displayed, check installation of the distribution cables and loopback connector. If all expected responses were received, the ACP 629, cables and device driver are correctly installed.

Initiate a call as follows:

```
c
```

The call response follows:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

7. Remove the loopback connector and install the X.25 trunk line connector. The trunk line can be attached to one of many X.25 devices such as a network modem, stand-alone X.25 switch or PAD, or another ACP 629.

8. Disable the X.25 link and configure the ACP 629 for network operations by entering

```
L 96 0 0 3 0 66 0
```

9. Then re-enable the link layer by entering

```
L 96 0 0 1 1
```

Within 60 seconds the ACP 629 displays

```
X.25 link up
```

If not, review the problems described in appendix A.

## 5.7 Usage Notes

1. The constant NXBLINES (defined in the driver) is the number of ACP 629 lines defined for your system. Minor devices 0 to (NXBLINES-1) map to the first ACP 629, minor devices NXBLINES to (2*NXBLINES-1) map to the second and so on.

2. The UNIX driver uses a kernel variable array to determine the addresses of the ACP 629 boards. Although the addresses for the ACP 629 need not be contiguous, take care that each device has a full 32-byte area that does not overlap the area of any other device.

3. If the COMREGs are not available during the first open on a xb unit (board) or during an attempt to write to the xb unit the following message is displayed on the UNIX console:

   ```
   xb: xbopen, ACP629 unit <n> not responding to reset
   ```

   Further attempts to access any port on that xb unit returns an immediate error, with errno = ENXIO.

   This condition can be cleared by opening the xb minor device number (252 + unit_number), that is, minor 252, 253, 254, or 255 for units 0, 1, 2, or 3. This sends a SIGHUP signal to all process groups associated with open ports on that unit, and immediately returns an error to all processes awaiting completion of an open on a port on that ACP 629 unit. Note that this convention makes the last four PAD ports on the fourth ACP 629 unavailable; they can still be used as HOST ports.

   The next open of any port on the affected ACP 629 unit causes an attempt to reset the unit and resume normal operation. If this attempt fails, the unit will again be marked as offline, continually returning errno ENXIO as above.

   One way to do the clear operation follows. Create the necessary device(s) using mknod(1M):

   ```
   mknod /dev/xbclr0 c 9 252
   mknod /dev/xbclr1 c 9 253
               .
               .
   ```

For example, run

```
cat < /dev/xbclr0
```

to do the ACP 629 clear operation on unit 0. It exits with an open error and EPIPE errno (returned, by convention, by the device driver to any process with an open pending on an ACP 629 that is cleared).

Change the /dev/xbclr<n> devices' modes to allow access only by root to prevent unauthorized users from clearing the ACP 629.

4. A diagnostic tool is available for printing port statistics for board 0, including the state of each active process. This tool can be used when up to three ACP629 boards are in a system (boards 0, 1, and 2).

   A port must be configured in the /dev directory. For example,

   ```
   /etc/mknod /dev/xbshow c 12 255
   ```

   could be used for major device number 12.

   You can get port status for board 0 by issuing an open on minor device 255 (/dev/xbshow):

   ```
   cat < /dev/xbshow
   ```

   The following message results:

   ```
   acp629: state of unit ?
   line state    iflag    oflag    cflag    lflag
   ```

   where ? represents the ACP 629 board number.

   The state of each active port is listed below this message.

   The message

   ```
   remaining lines free
   ```

   indicates that all ports that were not included in the previous message are inactive.

## 5.8 BSD UNIX and ULTRIX Quick Reference

| Function | Command or Procedure |
|---|---|
| Connect to the ACP 629 to place an outgoing call. | `tip PAD` |
| Return to BSD UNIX or ULTRIX from the ACP 629. | `<CR>~.` or `<CR>~<CNTRL>D` |
| Reset the ACP 629 if it is hung. | The ACP 629 is automatically reset upon the first open after a reboot. You can reset the ACP 629 by opening one of the configured clear devices. |
| Disable a port. | A port is automatically disabled upon the final close. |
| Transmit an X.25 break packet. | Use the ioctl command TIOCSBRK. See tty(4). |
| Enable a port for outgoing calls. | Create the appropriate PAD-port device name/number (with the most significant bit of the minor device number set) using mknod(1). Ensure the device name is not in /etc/ttys. |
| Enable a port for incoming calls. | Create the appropriate HOST-port device name/number (with the most significant bit of the minor device number clear) using mknod(1). Ensure that the corresponding HOST-port device name is included in /etc/ttys. |
| Enable a port for supervisor commands. | Root automatically has supervisor privileges. |

# Chapter 6

# UNIX System V Driver Installation

## 6.1 Introduction

This chapter describes how to install the ACP 629 driver on a DEC VAX or PDP-11/70 that runs UNIX System V, release 0, 1, or 2.0.

## 6.2 Software Components

The ACP 629 driver distribution kit for UNIX System V contains the the following files:

| | |
|---|---|
| tj.c | system V driver |
| tj.7 | unformatted manual page for the driver |
| tj7.txt | formatted manual page for the driver. This supplements section 7 in the *UNIX Administrator Reference Manual.* |
| setparm.c | utility program that uses a new IOCTL system call named TCSPARM. TCSPARM enables you to request output of a string of byte pairs over an open ACP 629 connection as a Q-bit data packet for setting X.3 parameters. |
| setparm.1 | unformatted manual page for the setparm program |
| setparm1.txt | formatted manual page for the setparm program. It supplements section 1 in the *UNIX User Reference Manual.* |
| setx29 | Bourne shell script that gives simplified syntax to invoke the setparm program. (The installation procedure creates the setparm program.) |

## 6.3 Installation Procedure

**CAUTION**

Before doing the following procedure, ensure that an up-to-date backup of the kernel build area exists. Create one if needed. When adding files to the system, rename existing files that have the same name.

In the following instructions the device used is /dev/rmt0 and the distribution medium is a magnetic tape. If the name in the /dev directory differs on your system or if you received the driver on floppy diskette, adjust the command lines.

1. Login as root.

2. cd to a directory that will temporarily be the location for the files to be extracted from the distribution tape contents. We recommend using a directory in "/tmp". For example, create a directory called "/tmp/acp629".

   ```
   tar xvbf 20 /dev/rmt0
   ```

   The driver's target directory for System V on a VAX is /usr/src/uts/vax/io. The driver's target directory for System V on a PDP-11 is /usr/src/uts/pdp11/io.

   cd to the driver's target directory.

   ```
   mv /tmp/acp629/driver/tj.c .
   ```

3. Choose the target directory for the setparm program (suggestion: /etc).

   cd to the setparm program's target directory.

   ```
   mv /tmp/acp629/util/setparm.c .
   ```

4. If you wish to format the manual pages (formatted versions are contained in the distribution), issue the following commands:

   cd to the directory that contains formatted section 1 manual pages.

   ```
   nroff -man /tmp/acp629/doc/setparm.1 > setparm.1
   ```

   cd to the directory that contains formatted section 7 manual pages.

   ```
   nroff -man /tmp/acp629/doc/tj.7 > tj.7
   ```

   cd to the directory that contains unformatted UNIX section 1 manual pages.

   ```
   mv /tmp/acp629/doc/setparm.1 .
   ```

   cd to the directory that contains unformatted UNIX section 7 manual pages.

   ```
   mv /tmp/acp629/doc/tj.7 .
   ```

5. If you want to copy the formatted manual pages directly from the distribution medium to the appropriate directories, do as follows:

   `cd` to the directory that contains formatted section 1 manual pages.

   ```
   mv /tmp/acp629/doc/setparm1.txt setparm.1
   ```

   `cd` to the directory that contains formatted section 7 manual pages.

   ```
   mv /tmp/acp629/doc/tj7.txt tj.7
   ```

6. Choose the target directory for the setx29 shell script (suggestion: /etc).

   `cd` to the target directory.

   ```
   mv /tmp/acp629/util/setx29 .
   ```

   ```
   chmod 555 setx29
   ```

7. Clean up as follows:

   ```
   cd /tmp/acp629
   rm -r *
   cd ..
   rmdir acp629
   ```

Do the following steps to complete the installation procedure:

8. Arrange comment delimiters and #defines in the driver source file, tj.c, for the appropriate CPU type and System V release number as follows:

   ```
   UNIX System V release 2.0 on a PDP 11/70:
       #define TJPDP 1170
       #define TJV2  5.2
       /*
       #define TJSYSIII  3.0
       #define TJDEBUG   1
       */

   UNIX System V release 2.0 on a VAX:
       #define TJV2  5.2
       /*
       #define TJPDP 1170
       #define TJSYSIII  3.0
       #define  TJDEBUG  1
       */

   UNIX System V release 0 or release 1 on a PDP 11/70:
       #define  TJPDP    1170
       /*
       #define  TJSYSIII 3.0
   ```

```
     #define TJV2 5.2
     #define TJDEBUG   1
         */
```

```
UNIX System V release 0 or release 1 on a VAX:
     /*
     #define  TJPDP    1170
     #define  TJSYSIII 3.0
     #define TJV2 5.2
     #define TJDEBUG   1
         */
```

If more than one ACP 629 is installed, change the #define for NTJS to the number of ACP 629 boards installed.

Change the #define for ZVECTOR to the interrupt vector for the first ACP 629, as specified in the dfile described in step 11.

9. The makefile on a VAX is /usr/src/uts/vax/io/io.mk. The makefile on a PDP-11 is /usr/src/uts/pdp11/io/io.mk. Modify the appropriate makefile to add tj.o to FILES as follows:

```
FILES =\
     $(LIBNAME)(dz.o)\
     $(LIBNAME)(dzb.o)\
     $(LIBNAME)(tj.o)\
```

Add tj to IFILES as follows:

```
IFILES =\
     dz|\
     dzb|\
     tj|\
```

10. Add the dependencies of the tj driver exactly as follows:

```
$(LIBNAME)(tj.o):\
     $(INCRT)/sys/param.h\
     $(INCRT)/sys/types.h\
     $(INCRT)/sys/dir.h\
     $(INCRT)/sys/signal.h\
     $(INCRT)/sys/user.h\
     $(INCRT)/sys/errno.h\
     $(INCRT)/sys/file.h\
     $(INCRT)/sys/sysmacros.h\
     $(INCRT)/sys/tty.h\
     $(INCRT)/sys/termio.h\
     $(INCRT)/sys/conf.h\
     $(INCRT)/sys/proc.h\
     $(INCRT)/sys/uba.h\
     $(INCRT)/sys/mtpr.h\
     $(FRC)
```

11. The dfile on a VAX is in /usr/src/uts/vax/cf. The dfile on a PDP-11 is in /usr/src/uts/pdp11/cf.

    Change the appropriate dfile to include the following line once for each ACP 629 installed. (Separate fields by tabs.)

    ```
    tj11     0     320     766740     5
    ```

    Change the field containing 320 to an unused octal interrupt vector with eight bytes available. If more than one ACP 629 is present, 8 * number_of_ACP629s bytes must be available and subsequent ACP 629s' interrupt vector must follow the first at 8-byte intervals. For example, a second ACP 629 would have interrupt vector 330 in the case shown. To find available interrupt vectors, examine the same column of all other dfile device entries. The field containing 766740 is the unique UNIBUS octal address for each ACP 629. This UNIBUS address must be on a 40 octal byte boundary and must have 20 octal bytes of space available. Note that the second field of the above dfile entry (0) is the UNIBUS adapter number that must be present *only* when using the DEC UDA50 device driver.

    The field containing 766740 contains the unique UNIBUS octal address for each ACP 629. The UNIBUS address must be on a 40 octal byte boundary and must have 20 octal bytes of space available. Before installation, make sure this location is not in use by another device. The field containing 5 is the bus request level.

12. Add the following line to the master file /etc/master. (Separate fields by tabs.)

    ```
    tj11   8   37   6   tj   8   0   12   32   5
    ```

    Change the field containing 12 to an unused major device number (different from the numbers in the same column of all other master file entries). See appendix C for more information on major device numbers. See master(4) for the format of /etc/master.

13. On a VAX, ensure that your current directory is /usr/src/uts/vax/cf. On a PDP-11, ensure that your current directory is /usr/src/uts/pdp11/cf. If using the DEC UDA50 device driver, run

    ```
    scsgen -i <dfile_name>
    ```

    In any case, run

    ```
    config -t <dfile_name>
    ```

`<dfile_name>` is the name of the modified dfile created previously. Confirm the tj11's major device number with the table displayed by config.

For VAX systems, edit the file univec.c (made by running config) and change all tjxint references to tjrint (deleting the redundant extern definition this creates). Specifically, change

```
extern tjrint(), tjxint();
```

to

```
extern tjrint();
```

And change

```
(int *)((int)tjrint+DO),
(int *)((int)tjxint+DO),
```

to

```
(int *)((int)tjrint+DO),
(int *)((int)tjrint+DO),
```

Ensure that your current directory is /usr/src/uts/vax/cf. To build a new UNIX kernel named /usr/src/uts/vax/unix<VER>, edit the file /usr/src/uts/vax/cf/Makefile (which is linked to the file cf.mk) and specify your preferred character string through "VER =", then execute make. The output of the ld step of the make confirms the name.

For PDP-11 systems, edit the file low.s (made by running config) and change all _tjxint references to _tjrint (deleting the redundant global definition this creates). Specifically, change

```
.globl  _tjrint, _tjxint
tjin:
     jsr     r0,call; _tjrint
tjou:
     jsr     r0,call; _tjxint
```

to

```
.globl  _tjrint
tjin:
     jsr     r0,call; _tjrint
tjou:
     jsr     r0,call; _tjrint
```

**NOTE**

> Every time config is run as described, the file univec.c or low.s must be edited as described.

Ensure your current directory is /usr/src/uts/pdp11/cf. To build a new UNIX kernel named /usr/src/uts/pdp11/unix<VER>, edit the file /usr/src/uts/pdp11/cf/Makefile (which is linked to the file cf.mk) and specify your preferred character string through "VER =", then execute make. The output of the ld step of the make confirms the specified name.

Regardless of the CPU type, save the previous /unix under another name (or pick another name for the new UNIX).

On a VAX, copy the new UNIX (/usr/src/uts/vax/unix<VER>) to /unix (or your preferred name) in the root directory.

On a PDP-11, copy the new UNIX (/usr/src/uts/pdp11/unix<VER>) to /unix (or your preferred name) in the root directory.

To avoid problems when you shut down multiuser UNIX, replace the file /unix only when running in single-user mode.

14. Create devices in /dev with the appropriate major/minor device numbers. Use major device numbers as you specified them when changing the master file. (These numbers are echoed in the table displayed by config.) Specify eight-bit minor device numbers as follows:

```
PBBLLLLL
```

where:

```
P     = 0, inbound (HOST) port
      = 1, outbound (PAD) port
BB    = board number
LLLLL = line number on the board.
```

See appendix C for more information on major and minor device numbers.

15. Choose names for HOST ports that begin with the prefix tty and that don't conflict with any existing terminal device names. For example, use tty60 through tty91.

The following examples, using mknod(1M), use major device number 12 and board number 0. HOST (incoming) ports have device names prefixed with tty. PAD (outgoing) ports have device names prefixed with pad:

```
/etc/mknod /dev/tty60 c 12 0
/etc/mknod /dev/tty61 c 12 1
      .
      .
/etc/mknod /dev/tty90 c 12 30
/etc/mknod /dev/tty91 c 12 31

/etc/mknod /dev/pad00 c 12 128
/etc/mknod /dev/pad01 c 12 129
      .
      .
/etc/mknod /dev/pad30 c 12 158
/etc/mknod /dev/pad31 c 12 159
```

If a port will only be used in a single mode, the /dev entry for the unused mode need not be created.

16. Edit /etc/inittab to bring up gettys as needed, adding entries as follows. (See inittab(4) of the *UNIX Programmer's Manual.)*

```
00:2:respawn:/etc/getty tty61 9600
```

**NOTE**

When ports are enabled in /etc/inittab, "init" opens the device for incoming calls and prevents use of that port for PAD mode.

17. Ports that are to be used as PAD (outgoing) ports can be accessed by the cu command. (After a connection is made, issuing <CR> gives the PAD prompt (@). Refer to the chapter 7 for the commands available in PAD mode. Use the command `<CR>~.<CR>` to leave PAD mode.

If appropriate entries such as

```
DIR pad23 0 9600
```

are added to the file /usr/lib/uucp/L-devices, the command

```
cu -lpad23 dir
```

can be used to access the port /dev/pad23.

18. If you add entries such as

```
PAD Any pad23 9600 pad23
```

to the file /usr/lib/uucp/L.sys, the command `cu PAD` can be used to access the next available PAD device listed in the files /usr/lib/uucp/L-devices and /usr/lib/uucp/L.sys. Note that because of a bug in System V, this works only for the first two PAD ports.

**NOTE**

The ports specified in /etc/inittab must not be specified in L-devices or L.sys because cu will not be able to open them.

19. When the driver has been installed and the new devices created, boot the new UNIX kernel.

## 6.4 Setparm Compilation

The ACP 629 driver supports a custom ioctl, TCSPARM, which can be used to select a string of byte pairs to be output as a data packet with the Q-bit set. Typically, this output can be used to set selected X.3 PAD parameters. (See appendix E for a description of PAD parameters.) The setparm command issues a TCSPARM ioctl to the ACP 629 driver. The syntax of the setparm command is given in the UNIX manual page, setparm(1), included in this distribution. ACC provides a shell script, setx29, that invokes the setparm command with simplified syntax. Steps 3 and 6 of section 6.3 describe how to install the setparm program and the setx29 shell script.

Compile the setparm program as follows:

1. `cd` to the target directory for the setparm program. (Refer to step 3 of section 6.3 for the location of the target directory)

2. `cc setparm.c -o /etc/setparm`

## 6.5 Installation Verification Procedure

Do the following steps to check ACP 629 installation.

1. Install the provided loopback connector in the distribution panel slot (see 2.9).

2. Login to root. (This enables access to the ACP 629 in supervisor mode.) Connect to an ACP 629 PAD port using the command

```
cu PAD
```

3. Press <CR>. This causes the ACP 629 to display the command prompt @. If it does not, review the driver installation procedures and ensure the device being accessed is configured as a PAD port.

4. Enter the following:

```
STATUS
```

The ACP 629 responds

```
X.25 link down
FREE
```

5. Disable the ACP 629 and set it for internal loopback as follows:

```
    L 96 0 0 4 0 0 66 3
```

The ACP 629 responds

```
    @
    X.25 Link Down
```

Enable the ACP 629 as follows:

```
    L 96 0 0 1 1
```

Within sixty seconds, the ACP 629 responds

```
    @
    X.25 Link Down

    X.25 Link Up
```

If these messages are not displayed within 60 seconds, check installation of the distribution cables.

Initiate a call as follows:

```
    c
```

The call response follows:

```
    PAD Received Answer Message While In CWA State.
    0 CONNECTED
```

6. Disable the ACP 629 and configure it for external loopback as follows. (An external loopback connector must be installed for external connector tests to pass):

```
    L 96 0 0 4 0 0 66 1
```

The ACP 629 responds

```
@
X.25 Link Down
```

Enable the ACP 629 as follows:

```
L 96 0 0 1 1
```

Within sixty seconds the ACP 629 responds

```
@
X.25 Link Down
X.25 Link Up
```

If these messages aren't displayed, check installation of the distribution cables and loopback connector. If all expected responses were received, the ACP 629, cables and device driver are correctly installed.

Initiate a call as follows:

```
c
```

The call response follows:

```
PAD Received Answer Message While In CWA State.
0 CONNECTED
```

7. If steps 1-6 succeed, test the setparm program by modifying an X.3 PAD parameter. (The loopback connector must remain connected during this phase.) To modify an X.3 parameter, do the following:

   a. At the @ prompt, enter

      ```
      C<CR>
      ```

      This makes a loopback connection to an incoming ACP 629 Host port.

   b. Login as any user.

   c. Escape back to the PAD by pressing `<CNTRL>P`

   d. Press <CR>. This causes the ACP 629 to display @.

   e. Enter the following:

      ```
      par?
      ```

      A list of the current value of the 18 X.3 PAD parameters is displayed. The value of each parameter is separated from its parameter number by a colon. Appendix E describes each parameter.

f. Reconnect to the Host port by entering C<CR> at the @ prompt.

g. Press <CR> at the RECONNECTED message.

h. Use the setx29 shell script to modify the X.3 PAD parameters. For example, if PAD parameter 14 equals 0, change it to 1 by doing the following:

   cd to the directory that contains the setx29 shell script. Then enter

```
./setx29 14:1
```

   If PAD parameter 14 equalled 1 before issuing this command, change parameter 14 to a value other than 1 (by entering a value other than 1 in the setx29 command).

i. A message is displayed stating that the ioctl completed. Escape to the PAD again to check the value of the X.3 PAD parameter you just changed:

```
<CNTRL>P
```

j. Enter <CR>. This causes the ACP 629 to display @.

k. To make sure that the parameter has been changed correctly, enter the following at the @ prompt:

```
par?
```

   Examine the list displayed by this command for the appropriate parameter value (to check that the value was changed correctly).

8. Remove the loopback connector and install the X.25 trunk line connector. This trunk line can be attached to one of many X.25 devices such as a network modem, a stand-alone X.25 switch or PAD, or another ACP 629.

9. Disable the X.25 link and configure the ACP 629 for network operations by entering

```
L 96 0 0 3 0 66 0
```

10. Then re-enable the link layer by entering

    ```
    L 96 0 0 1 1
    ```

    Within 60 seconds the ACP 629 responds

    ```
    X.25 link up
    ```

    If not, review the problems described in appendix A and try again.

## 6.6 Usage Notes

1. The constant NTJLNS (defined in the driver) is the number of lines defined per ACP 629. Minor devices 0 to (NTJLNS-1) map to the first ACP 629, minor devices NTJLNS to (2*NTJLNS-1) map to the second and so on.

2. The UNIX driver uses a kernel variable array to determine the addresses of the ACP 629 boards. Although the addresses for the ACP 629 need not be contiguous, take care that each device has a full 32-byte area that does not overlap the area of any other device.

3. If the COMREGs are not available during the first open on a tj unit (board) or during an attempt to write to the tj unit the following message is displayed on the UNIX console:

    ```
    acp629: unit <n> not responding to <request>.
    ```

    Further attempts to access any port on that tj unit returns an immediate error, with errno = ENXIO.

    This condition can be cleared by opening the tj minor device number (252 + unit_number), that is, minor 252, 253, 254, or 255 for units 0, 1, 2, or 3. This sends a SIGHUP signal to all process groups associated with open ports on that unit, and immediately returns an error to all processes awaiting completion of an open on a port on that ACP 629 unit. Note that this convention makes the last four PAD ports on the fourth ACP 629 unavailable; they can still be used as HOST ports.

    The next open of any port on the affected ACP 629 unit causes an attempt to reset the unit and resume normal operation. If this attempt fails, the unit will again be marked as offline, continually returning errno ENXIO as above.

One way to do the clear operation follows. Create the necessary device(s) using mknod(1M):

```
mknod /dev/tjclr0 c 9 252
mknod /dev/tjclr1 c 9 253
          .
          .
```

For example, run

```
cat < /dev/tjclr0
```

to do the ACP 629 clear operation on unit 0. It exits with an open error and EPIPE errno (returned, by convention, by the device driver to any process with an open pending on an ACP 629 that is cleared).

Change the /dev/tjclr<n> devices' modes to allow access only by root to prevent unauthorized users from clearing the ACP 629.

4. A diagnostic tool is available for printing port statistics for board 0, including the state of each active process. This tool can be used when up to three ACP629 boards are in a system (boards 0, 1, and 2).

   A port must be configured in the /dev directory. For example,

```
/etc/mknod /dev/tjshow c 12 255
```

   could be used for major device number 12.

   You can get port status for board 0 by issuing an open on minor device 255 (/dev/tjshow):

```
cat < /dev/tjshow
```

   The following message results:

```
acp629: state of unit ?
line state    iflag    oflag    cflag    lflag
```

   where ? represents the ACP 629 board number.

   The state of each active port is listed below this message.

   The message `remaining lines free` indicates that all ports that were not included in the previous message are inactive.

## 6.7 UNIX System V Quick Reference

| Function | Command or Procedure |
|---|---|
| Connect to the ACP 629 to place an outgoing call. | `cu PAD` |
| Return to UNIX System V from the ACP 629. | `<CR>~.<CR>` |
| Reset the ACP 629 if it is hung. | The ACP 629 is automatically reset upon the first open after a reboot. You can reset the ACP 629 by opening one of the configured clear devices. |
| Disable a port. | A port is automatically disabled upon the final close. |
| Transmit an X.25 break packet. | Use the ioctl command TCSBRK. See termio(4). |
| Enable a port for outgoing calls. | Create the appropriate PAD-port device name/number (with the most significant bit of the minor device number set) using mknod(1). Ensure the device name is not in /etc/inittab. |
| Enable a port for incoming calls. | Create the appropriate HOST-port device name/number (with the most significant bit of the minor device number clear) using mknod(1). Ensure that the corresponding HOST-port device name is included in /etc/inittab. |
| Enable a port for supervisor commands. | Automatically granted to root. |

# Chapter 7

# Operation

## 7.1 Introduction

This chapter describes port allocation, port modes and states, and the ACP 629 user interface.

## 7.2 Port Allocation

The ACP 629 gives the host 32 virtual connections. These connections resemble auto-answer dial-out modems that give the host 32 full-duplex communications channels across one physical connection. The ACP 629 numbers the 32 host connections, called host ports, 0 to 31. The ACP 629 gives the network 32 possible connections: network ports 1 through 32. Host port numbers are one less than corresponding network port numbers (e.g., host port 0 is network port 1).

Each port is in DIS mode (disabled: see 7.3.1) upon initialization. A port must be in ENA mode (enabled: see 7.3.2) for an incoming call, or PAD mode (see 7.3.3) for an outgoing call, to establish communication.

Some networks assign a block of addresses to each site. Other networks assign only one address per site. The ACP 629 uses the two low-order bytes of this address field on incoming call packets to allocate ports. The ACP 629 tries to assign an incoming call packet to the network port number that matches the two low-order bytes of the packet's address field as follows.

If the number is 00 or greater than 32, the ACP 629 searches (in numerically ascending order) for a port in ENA mode starting with network port 1. The ACP 629 establishes the call at the first port it finds in ENA mode. If port 1 is in ENA mode, the ACP 629 establishes the call and the port changes to XFR (incoming call) condition. If port 1 is not in ENA mode, the ACP 629 checks port 2. If port 2 is not in ENA mode, the ACP 629 checks port 3, and so forth. The ACP 629 clears the call (sends a clear request) if it finds none of the ports 1 through 32 in ENA mode.

If the number is 1 through 32, the ACP 629 follows a slightly different search procedure as follows. The ACP 629 searches (in numerically ascending order) for a port in ENA mode, starting with the corresponding network port number. If that port is in ENA mode, the ACP 629 establishes the call at that port and the port changes to XFR (incoming call) condition. If that port is in XFR (incoming call) condition, the ACP 629 tries the next higher numbered port. If the next port would be 33, the ACP 629 tries port 1. When the ACP 629 finds a port in any condition other than ENA or XFR (incoming call), it clears the call (sends a clear request). The ACP 629 checks the ports in numerical order until

1. it finds a port in ENA mode and establishes the call,
2. it finds a port not in ENA or XFR (incoming call) condition and clears the call, or
3. returns to its beginning port number (without making a connection) and clears the call.

## 7.3 Port Modes and States

ACP 629 ports as viewed by the host are in one the following modes (parentheses enclose the state identifier returned in the response to the supervisor command Q):

1. Disabled Mode (DIS)
2. Enabled Mode (ENA)
3. PAD Mode (PAD)

Refer to the appropriate driver installation chapter for information on how to set port modes.

**7.3.1 Disabled Mode.** Upon initialization, all ACP 629 ports are in disabled mode. While in this mode, the port does not accept X.25 calls and discards outgoing host data. Ports can be returned to disabled mode by the close function specific to the operating system used with the ACP 629.

**7.3.2 Enabled Mode.** Ports in enabled mode accept incoming X.25 calls. Enabled ports cannot place outgoing X.25 calls. After the ACP 629 completes an incoming call, it passes a connected status to the device driver which causes the initiation of a login function on that port. If an enabled port has not indicated a connected status, the host data sent to the port is discarded.

**7.3.3 PAD Mode.** Ports in PAD mode can place outgoing X.25 calls. PAD mode ports do not answer incoming X.25 calls. In this mode the ACP 629 interprets all host data as PAD commands until an X.25 call is established. After a call has been established, all data except for the PAD recall character (X.3 parameter number 1 in appendix E) is passed uninterpreted from the local PAD to the remote host.

**7.3.4 PAD States.** PAD mode ports are in one of the following four states (parentheses enclose the state identifier returned in response to the supervisor command Q):

1. Command state (PAD)
2. Call Wait state (CWA)
3. Transfer state (XFR)
4. Disconnect Wait state (DWA)

Transitions between these states are caused by user commands and key stroke sequences. Access to the local PAD is possible only in the command state. Data transfer to the remote host is possible only in the transfer state. These two states are mutually exclusive; their functions do not overlap.

**7.3.4.1 Command State.** The PAD command prompt (@) is displayed while in this state, enabling you to enter the PAD commands. From this state, the terminal can advance to the Call Wait state or Transfer state.

**7.3.4.2 Call Wait State.** This state is entered from the Command state when a call command is entered. A call command attempts a connection on the X.25 network. While in this state, all terminal input is ignored. From this state, the terminal either advances to the Transfer state (upon call acceptance) or returns to the Command state (upon disconnect received or after the call time-out period expires).

**7.3.4.3 Transfer State.** This state is entered from the Call Wait state (receipt of an answer to a call request), from the Command state (reconnect request to an already established circuit), or from the Enabled state (receipt of an incoming call request). While in the Transfer state, the port is connected to the called device and responds according to the remote device's disposition and the setting of the PAD parameters.

After the Transfer state has been entered, most data is passed uninterpreted to the remote host. The Command state can be reentered from the transfer state without breaking the established virtual circuit by entering the PAD recall character. The key stroke sequence needed to do this transfer depends on the setting of PAD parameter number 1 (appendix E) and on the terminal's characteristics. The default is <CNTRL> P (control key and P key pressed simultaneously).

**7.3.4.4 Disconnect Wait State.** This state is entered when an attempt is made to disconnect a network connection. The state advances to the Command state after receiving a disconnect confirmation from the network or after the time-out period expires.

NOTE

Logout or final messages are sometimes lost or truncated. This is due to data packets (which have not been delivered to the host) being purged upon receiving a clear indication.

**7.3.5 PAD Commands.** PAD commands can be issued from local terminals or programs. The two types of PAD commands are user commands and supervisor commands. User commands can be issued by anyone accessing a PAD mode port. Supervisor commands can be issued to ports in supervisor mode. (See the appropriate software installation chapter for information on how to set a port for supervisor mode.) Both command types can be issued to any port in supervisor mode. Note that a command can be continued on a second line by entering a backslash (\) as the last character on the first line. This also enters a space into the command string.

PAD commands can be at most 127 characters. The ACP 629 truncates commands longer than 127 characters, adds a <CR> and null character to the end of the truncated command, and processes the truncated command. The ACP 629 then outputs the error message `End of Buffer`. The ACP 629 buffers the rest of the command as the start of a subsequent command. When this part of the command is processed, another error message might happen (unless this part of the command is a valid command).

**7.3.5.1 User Commands.** The ACP 629 PAD mode command state provides terminal users with the commands listed in table 7-1. These commands connect and disconnect calls, display the status of the network connection, reset a connection, interrupt data flow, and modify or query an X.3 parameter list (see appendix E) that describes the physical terminal to the network.

The X.3 parameter list accessed by the commands in table 7-1 are unique to each host port. This means that each PAD port can be configured independently of all other PAD ports. The configuration of PAD ports is kept between outbound calls; however, ports must be reconfigured after each powerup or UNIBUS INIT.

Table 7-1. ACP 629 PAD Mode User Commands

| Command | Description |
|---|---|
| C [n][/R] | Request outgoing connection |
| C | Return to outgoing connection |
| CLR | Clear connection |
| D | Disconnect connection |
| INT | Interrupt transmission |
| INTD | Interrupt transmission and discard data |
| PAR? | Display X.3 (PAD) parameters |
| PROF [i] | Select X.3 parameter profile |
| RESET | Reset virtual circuit |
| SET id:val[,id:val...] | Modify X.3 parameters |
| SET? id:val[,id:val...] | Modify/query X.3 parameters |
| STAT | Query port status |

### 7.3.5.1.1 Call.

| | | |
|---|---|---|
| Syntax: | | C [n][/R] |
| Elements: | C | Call request command. |
| | n | Optional 1 to 15 digit ASCII network address. This address is right justified and defaults to "0". |
| | /R | Optional request for reverse call charging. |
| Example: | | C 41105550010221 |

The call command establishes a connection through the X.25 network. The interpretation of the resulting 15 digit address depends on the particular network and the remote PAD.

### 7.3.5.1.2 Continue.

| | |
|---|---|
| Syntax: | C |

The continue command reconnects a terminal to an already established virtual circuit after the PAD recall character has been entered.

#### 7.3.5.1.3 Clearing a Call.

| | | |
|---|---|---|
| Syntax: | | CLR |
| | | D |
| Elements: | CLR | Clear call command |
| | D | Disconnect call command |
| Example: | | CLR |
| Result: | | Attempting disconnect. |

The clear (or disconnect) command discontinues or breaks a virtual circuit. The terminal must reenter the Command state by using the PAD recall character before this command can be used. If the ACP 629 receives a disconnect (clear) request from the network while in PAD mode, a message is displayed on the local terminal that includes the reason code (appendix F).

#### 7.3.5.1.4 Interrupt.

| | | |
|---|---|---|
| Syntax: | | INT |
| | | INTD |
| Elements: | INT | Interrupt data transmission. |
| | INTD | Interrupt data transmission and discard data. |

Both forms of the interrupt command cause an interrupt packet to be transmitted. Also, INTD discards all pending output. Both interrupt commands need an established virtual circuit. If either command is entered without an active connection established, the message `No call` is displayed. A second interrupt packet will not be transferred until the first is acknowledged.

#### 7.3.5.1.5 X.3 Parameter Display.

| | | |
|---|---|---|
| Syntax: | | PAR? |
| Elements: | PAR? | X.3 parameter query command |
| Example: | | PAR? |
| Result: | | Port Number N Parameters<br>1:0 2:0 3:127 4:0 5:0 6:0 7:8 8:0 9:0 10:0<br>11:14 12:0 13:0 14:0 15:0 16:127 17:24 18:18 |

The X.3 parameter display command lists the 18 X.3 parameters for the port the local terminal is attached to. Each parameter is displayed in the following format: [Parameter ID]:[Parameter value]

### 7.3.5.1.6 Set Profile.

| | | |
|---|---|---|
| Syntax: | | PROF [i] |
| Elements: | PROF | Set X.3 parameter profile command |
| | i | Optional profile number. Profile numbers range from 0 to 2 (table 7-2). If no profile number is entered profile 0 is used. |
| Example: | | PROF 2 |

The set profile command enables you to set the 18 X.3 (PAD) parameters by selecting one of three predetermined PAD parameter profiles. If an out of range profile number is entered, the message `Bad profile` is displayed.

Table 7-2. PAD Profile Values

| X.3 Parameter Name | Parameter ID | Parameter Value for Profile 0 | Profile 1 | Profile 2 |
|---|---|---|---|---|
| PAD recall | 1 | 1 | 0 | 16 |
| Echo | 2 | 0 | 0 | 1 |
| Data forwarding character set | 3 | 127 | 127 | 126 |
| Data forwarding timeout | 4 | 20 | 0 | 20 |
| PAD to DTE flow control | 5 | 1 | 0 | 1 |
| Control of PAD service messages | 6 | 5 | 0 | 5 |
| PAD action on BREAK | 7 | 8 | 8 | 8 |
| Discard output | 8 | 0 | 0 | 0 |
| <CR> padding | 9 | 0 | 0 | 0 |
| Line folding | 10 | 0 | 0 | 80 |
| Bit rate | 11 | 14 | 14 | 14 |
| DTE to PAD flow control | 12 | 1 | 0 | 0 |
| Line feed after <CR> | 13 | 0 | 0 | 7 |
| Line feed padding | 14 | 0 | 0 | 0 |
| Editing | 15 | 0 | 0 | 1 |
| Character delete | 16 | 127 | 127 | 127 |
| Line delete | 17 | 24 | 24 | 24 |
| Line display | 18 | 18 | 18 | 18 |

Table 7-3. PAD Profile Purposes

| Profile | Purpose |
|---|---|
| 0 | Standard terminal characteristics. |
| 1 | Transparent data transfer. |
| 2 | Special terminal characteristics. |

#### 7.3.5.1.7 Reset Circuit.

| | | |
|---|---|---|
| Syntax: | | RESET |

The reset command sends a reset packet to the X.25 network. This command is used to reinitialize a virtual circuit. When the reset command is entered, all outstanding data and interrupt packets (for both directions) are purged. The reset procedure only applies to a circuit in Data Transfer state.

#### 7.3.5.1.8 Set Terminal X.3 Parameters.

| | | |
|---|---|---|
| Syntax: | | SET id:val[,id:val]... |
| | | SET? id:val[,id:val]... |
| Elements: | SET | Set PAD parameter command |
| | SET? | Set and query PAD parameter command |
| | id | PAD parameter ID |
| | val | PAD parameter value |
| Example: | | SET? 3:127,5:1,2:0 |
| Result: | | 3:127<br>5:1<br>2:0 |

The set command enables you to change one or more of the 18 X.3 (PAD) parameters. The set command enables you to tailor the behavior and logical appearance of the terminal to the physical terminal. One or more parameters can be modified in a single command and can appear in any order. If an illegal or out of range parameter ID or value is detected, an error message is displayed and the command aborts at that point.

The SET? command functions the same as the SET command but SET? also confirms the change by displaying the new PAD parameter values.

#### 7.3.5.1.9 Port Status.

Syntax: STATUS

The status command queries the status of a PAD port. The response shows the state of the X.25 trunk connection and, if a virtual circuit is established, the address of the connection.

### 7.3.5.2 PAD Supervisor Commands.

Table 7-4. ACP 629 PAD Mode Supervisor Commands

| Command | Description |
|---|---|
| B | Summary of X.25/PAD buffer states |
| F | Check facilities |
| R | Accept/reject reverse charge calls |
| L | Low level interface. See chapter 8 |
| Q | Query terminal/port status |
| X? | Query X.3 parameters |
| X [id:val[ id:val]...] | Modify X.3 parameter suite |

PAD supervisor commands modify and query the operating status of the ACP 629. These commands are intended for system administrators. For details on activating PAD supervisor commands, refer to the "Quick Reference" section at the end of the appropriate driver installation chapter.

### 7.3.5.2.1 Buffer Status Query.

| | | |
|---|---|---|
| Syntax: | | B |
| Elements: | B | Buffer status query command |
| Example: | | B |
| Result: | | |

```
         Free     Total
Large    NNN      NNN
Small    NNN      NNN

Buffer Low: XXXXX          Count: NNN
Lower Limit: NNN           Upper Limit: NNN
Lowest Free Count: NNN
```

The buffer status query command lists the internal X.25/PAD buffer states. ACP 629 buffers consist of two pools, one containing large transmission buffers and one containing small transmission buffers. This command shows the number of buffers available (`Free`) and the total number of buffers in the system for each buffer pool.

If `Buffer Low` is FALSE, the ACP 629 can allocate buffers from its Large buffer pool. If `Buffer Low` is TRUE, the ACP 629 temporarily cannot allocate buffers from its Large buffer pool. `Count` is the number of times that the ACP 629 has entered a Buffer Low condition since the last ACP 629 reset.

`Lower Limit` displays the threshold where the ACP 629 will enter a `Buffer Low` status. `Upper Limit` displays the threshold where the ACP 629 will leave a `Buffer Low` status. See chapter 8 for information on the parameters that control the Upper and Lower buffer thresholds.

`Lowest Free Count` is the lowest number of free Large buffers that the ACP 629 has had since the ACP 629 was reset.

The above information shows system activity levels. System problems are indicated only in the following cases:

1. The number of free buffers steadily increases (e.g., because buffers are released more than once).

2. The number of free buffers decreases without recovering (e.g., because buffers are lost)

3. The ACP 629 is in a Buffer Low condition for a long time (e.g., because the ACP 629 cannot release buffers).

### 7.3.5.2.2 Facilities Checking.

| | | |
|---|---|---|
| Syntax: | | F [N/Y] |
| Elements: | F | Facilities checking command |
| | [Y/N] | Y = Yes, check facilities (default).<br>N = No, do not check facilities.<br><br>If this indicator is omitted the current status is displayed. |
| Example: | | F N |

The facilities checking command sets whether the ACP 629 checks the facilities field in call request packets. If this option is set to N, the facilities field is ignored in all call request packets. If set to Y, the facilities field in all call requests is scanned for the attributes Reverse Charge Call, Packet Window Size, and Packet Size.

If the attribute Packet Window Size is found in a call request, the call accepted packet will contain an entry for Packet Window Size that negotiates the attribute back to the value set by the parameter Packet Window (described in chapter 8).

If the attribute Packet Size is found in a call request, the call accepted packet will contain an entry for Packet Size that negotiates the attribute back to 128 bytes. (This is the only packet size that the ACP 629 can handle.)

If the attribute Reverse Charge Call is found in a call request, the call request is processed in accord with the Reverse Charge Call Acceptance command.

The facilities checking command is a global option and affects all circuits.

### 7.3.5.2.3 Reverse Charge Call Acceptance.

| | | |
|---|---|---|
| Syntax: | | R [N/Y] |
| Elements: | R | Reverse charge call acceptance command. |
| | [Y/N] | Y = Yes, accept reverse charge calls.<br>N = No, do not accept reverse charge calls (default).<br><br>If this indicator is omitted the status is displayed. |
| Example: | | R Y |

The reverse charge call acceptance command enables the ACP 629 to accept or reject incoming calls with the reverse charge option set in the facilities field of call request packets. The reverse charge call acceptance command is a global option and affects all circuits. Note that the facilities checking command must be set to Y before the reverse charge call acceptance command affects incoming call requests.

### 7.3.5.2.4 Query Terminal/Port Status.

Syntax: Q

The query command displays the status of all 32 ports. Port number is paired with a three character abbreviation of its mode as follows:

| | |
|---|---|
| DIS | disabled |
| ENA | enabled. The port can receive CALLS |
| PAD | PAD mode. The port can be used to place CALLS and do PAD commands |
| XFR | transfer mode. The port has an active CALL in progress. |
| CWA | Call Wait |
| DWA | Disconnect Wait |

### 7.3.5.2.5 View or Modify X.3 Host Parameters.

| | | |
|---|---|---|
| Syntax: | | X? |
| | | X [id:val[ id:val]...] |
| Elements: | X? | Query X.3 parameter suite command |
| | X | Modify X.3 parameter suite command |
| | id | X.3 parameter identifier |
| | val | X.3 parameter value |

The X and X? commands allow access to the suite of X.3 parameters (appendix E) that the ACP 629 keeps to configure remote PADs when an ACP 629 port is acting as a host port. The X command replaces the list of X.3 parameters, and the X? command displays the current parameter list.

These commands have a different function than the SET and SET? commands described in section 7.3.5.1.8. SET and SET? describe individual terminals for outbound calls where the ACP 629 is acting as a PAD. The X command builds an X.3 parameter suite that is used to modify a remote PAD's terminal descriptions. The suite of X.3 parameters built by the X command is sent to a remote PAD after an incoming call has been accepted. The X command is used by the PAD to modify the suite of X.3 parameters that describe the terminal that initiated the call. When the remote PAD is another ACP 629, the PAD terminal description is built using the SET command and modified after a call is accepted by the suit of parameters created with the X command on the host.

After the ACP 629 has been reset at powerup or by an external process, the ACP 629 reloads the following default values for its X.3 parameter suite:

```
X 1:1 2:0 3:127 4:0 5:0 6:5 7:8 8:0 9:0 10:0 12:0 13:0 14:0 15:0 0:33 1:0 10:0 18:0
```

Because these values are loaded after ACP 629 reset, you must reenter site-specific changes to the X.3 parameter suite. You can do this manually or as part of an automated startup procedure that copies a file (using COPY or CAT) containing the X command (or other configuration commands) to an ACP 629 port that is in supervisor mode.

To modify an existing suite of parameters or to create a new suite, *all* parameters must be entered. Any parameter that is not in the list following the X command will not be included in the parameter suite. Also, the parameters are sent to the remote PAD in the order that they are entered in the input string. This means that unused or unimportant parameters can be omitted.

The input order of the parameters also allows you to set network specific parameters. The parameter ID 0 denotes the start of a network specific parameter list. The value following a parameter value of 0 identifies the network that the parameters belong to. For example, the sequence 0:33 (as in the default suite above) means that *all* of the parameters that follow are TELENET specific (33 is the identifier for TELENET). These parameters (0:33 1:0 10:0 18:0) were implemented for users running through TELENET PADs that do not support X.3 parameter number 3 set to 127 (data forwarding = all characters). Instead, the default suite of parameters resets several timers so that data is forwarded at a reasonable rate. Any other network that sees the combination 0:33 ignores the rest of the message.

# Chapter 8

# ACP 629 Low-Level Command Interface

## 8.1 Introduction

The ACP 629 has a low-level command and status interface. This interface enables system administrators to get information on ACP 629 operation and customize ACP 629 operation to meet X.25 network needs.

The ACP 629 low-level command interface is accessible from ports in supervisor mode. Refer to the appropriate driver installation chapter for information on how to enter supervisor mode. (A terminal displays @ when it enters supervisor mode.) The syntax for entering commands to this interface follows:

```
L <decimal command ID> <decimal command parameter list>

                                 OR

LO <octal command  ID> <octal command  parameter  list>
```

Both the decimal and octal parameter lists represent byte values and are separated by either spaces or tabs. Information on specific commands and parameters is in section 8.3.

ACP 629 responses are printed in the following format:

```
0       1       2       3       4       5       6       7
<byte>  <byte>  <byte>  <byte>  <byte>  <byte>  <byte>  <byte>
<byte> .......
```

The header numbers 0 to 7 (separated by tab characters) are displayed to help you decode the response. The header is followed by a list of octal byte values that comprise the message or response. Multi-byte values are returned as 16-bit or 32-bit integers. 16-bit integers are returned low-order byte first; 32-bit integers are returned as two 16-bit integers, with the high-order integer first. A query and response example is in section 8.2.

Responses to queries are returned to the originating terminal. Unsolicited status and error messages from the ACP 629 are returned to the first available terminal that is in supervisory mode. If none of the available terminals are in supervisory mode, unsolicited responses from the ACP 629 are discarded.

## 8.2 Low-Level Query and Response Example

This section gives an example of a query to and a response from the ACP 629 using the low-level command interface. The command used for this example, the virtual circuit query command, is explained further in section 8.3.4.

Command Line:

```
@L 129 0 5 0
```

NOTE

The offsets used in this section to describe command formats are offsets from the initial command designator L (see table 8-1). The offsets used to describe the response formats are zero relative byte counts from the start of the byte string.

Table 8-1. Explanation of Example Virtual Circuit Query Command

| Offset | Command | Meaning |
|---|---|---|
| | L | Command to access low-level command interface. |
| 0 | 129 | Decimal value for query virtual circuit. |
| 1 | 0 | Reserved (must be 0) |
| 2 | 5 | Virtual circuit number |
| 3 | 0 | Number of bytes to follow (must be 0) |

Response to terminal:

```
@0    1     2     3     4     5     6     7
200   000   005   035   001   000   000   000
002   000   000   000   002   000   000   000
146   376   000   000   000   000   000   000
000   000   000   000   000   000   000   000
000
```

Table 8-2. Explanation of Example
Virtual Circuit Query Response

| Offset | Response | Meaning |
|---|---|---|
| 0 | 0200 | Response type (128) |
| 1 | 0000 | Reserved |
| 2 | 0005 | Virtual circuit number requested. |
| 3 | 0035 | Response length |
| 4 | 0001 | Reserved |
| 5 | 0000 | Virtual circuit number |
| 6 | 0000 | P(S) for receive side |
| 7 | 0000 | P(R) for receive side |
| 8 | 0002 | Receive window |
| 9 | 0000 | Receive flags |
| 10 | 0000 | P(S) for transmit side |
| 11 | 0000 | P(R) for transmit side |
| 12 | 0002 | Transmit window |
| 13 | 0000 | Transmit flags |
| 14-15 | 0000 0000 | Address of first packet on queue to frame level. |
| 16-17 | 0146 0376 | Address of last packet on queue to frame level. |
| 18 | 0000 | X.25 logical channel state (table 8-15) |
| 19 | 0000 | Virtual circuit state (table 8-16) |
| 20-21 | 0000 0000 | Reserved |
| 22 | 0000 | Host port number times 2 |
| 23 | 0000 | Flags for virtual circuit |
| 24-25 | 0000 0000 | Address of first buffer on queue to host. |
| 26-27 | 0000 0000 | Address of last buffer on queue to host. |
| 28-29 | 0000 0000 | Address of current input buffer to host. |
| 30-31 | 0000 0000 | Address of current output buffer from host. |
| 32 | 0000 | Trailer |

## 8.3 Command and Response Formats

Table 8-3. Command and Response Codes

| Command | | Response | | Meaning |
|---|---|---|---|---|
| Decimal | Octal | Decimal | Octal | |
| – | – | 2 | 2 | Clear Virtual Circuit |
| 96 | 140 | – | – | System Control |
| – | – | 97 | 141 | Line Status |
| 129 | 201 | 128 | 200 | Virtual Circuit Query |
| 131 | 203 | 130 | 202 | Logical Channel Query |
| 135 | 207 | 134 | 206 | System Error Query |
| 137 | 211 | 136 | 210 | System History Query |
| 139 | 213 | 138 | 212 | Frame Query |
| 141 | 215 | 140 | 214 | Memory Query |
| 143 | 217 | 142 | 216 | Statistics Query |

The ACP 629 low-level command interface enables you to get information about ACP 629 operation from any port in supervisory mode. Commands can be used to adjust the characteristics of the ACP 629 to maximize throughput.

The responses to the query commands display how the ACP 629 configuration is performing. The identifiers for the query responses are one less than the identifiers for the command. For example, the virtual circuit query has identifier 129 and its response identifier is 128.

The system control command alters the characteristics of the ACP 629. This enables you to use the information gotten from the query commands to improve the performance of the ACP 629. The system control command has no response.

The line status and clear virtual circuit messages are asynchronous messages generated by the ACP 629. They are delivered to the first available port in supervisory mode. If none of the available ports are in supervisory mode, the message is discarded.

### 8.3.1 Clear Virtual Circuit.

Table 8-4. Clear Virtual Circuit Format

| Byte | Meaning |
|---|---|
| 0 | Command code (2) |
| 1 | VCN cleared |
| 2 | Cause code (table 8-5) |
| 3 | Number of bytes to follow |
| 4 | Diagnostic code (table 8-6) |

The clear virtual circuit message is an asynchronous message generated by the ACP 629 when acting as a DCE. The message is transmitted to the first available supervisory port whenever the the ACP 629 clears a virtual call. If no supervisory port is available, the message is discarded. Tables 8-5 and 8-6 describe the codes returned by the clear virtual circuit message.

Table 8-5. Clearing Cause Codes

| Value | Meaning |
|---|---|
| 0 | DTE originated |
| 1 | Number busy |
| 3 | Invalid facility request |
| 5 | Network congestion |
| 9 | Out of order |
| 11 | Access barred |
| 13 | Not obtainable |
| 17 | Remote procedure error |
| 19 | Local procedure error |
| 33 | Incompatible destination |

Table 8-6. Clearing Diagnostic Codes

| Byte | Meaning |
|---|---|
| 0-15 | No additional information |
| 16-31 | Packet type invalid |
| 32-47 | Packet not allowed |
| 48-63 | Timer expired |
| 64 | Call setup problem |
| 65 | Facility code not allowed |
| 66 | Facility parameter not allowed |
| 67 | Invalid called address |
| 68 | Invalid calling address |
| 69-79 | Call setup problem |
| 80-127 | Not assigned |
| 200-377 | Network specific diagnostic |

### 8.3.2 System Control Command.

Table 8-7. System Control Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (96) |
| 1-2 | Reserved (must be 0) |
| 3 | Count of number of bytes in command list (N). |
| 4-(3+N) | Command list (table 8-8) |

The system control command gives access to low-level timers and thresholds on the ACP 629. It is used mainly to fine tune the ACP 629's transmission characteristics. Table 8-8 summarizes the timers and thresholds accessible through the system control command.

Table 8-8. System Control Command IDs

| Command | | Section | Meaning | Activated |
|---|---|---|---|---|
| Decimal | Octal | | | |
| 0 | 0 | 8.3.2.2 | Disable X.25 trunk line | Immediately |
| 1 | 1 | 8.3.2.1 | Enables X.25 trunk line | Immediately |
| 66 | 102 | 8.3.2.3 | Set loop back mode | Link enable |
| 67 | 103 | 8.3.2.4 | Set DTE/DCE mode | Link enable |
| 68 | 104 | 8.3.2.5 | Set DTE address | Link enable |
| 69 | 105 | 8.3.2.6 | Set DCE address | Link enable |
| 70 | 106 | 8.3.2.7 | Set I-Frame timeout | Link enable |
| 71 | 107 | 8.3.2.8 | Set poll timeout | Link enable |
| 72 | 110 | 8.3.2.9 | Set disconnect timeout | Link enable |
| 74 | 112 | 8.3.2.10 | Set retry limit | Link enable |
| 75 | 113 | 8.3.2.11 | Set watchdog timeout | Link enable |
| 76 | 114 | 8.3.2.12 | Set bit rate | Immediately |
| 77 | 115 | 8.3.2.13 | Set idle poll flag | Link enable |
| 78 | 116 | 8.3.2.14 | Set frame level window size | Link enable |
| 79 | 117 | 8.3.2.15 | Set packet level window size | Packet Level Restart |
| 81 | 121 | 8.3.2.16 | Set high buffer threshold | |
| 82 | 122 | 8.3.2.17 | Set low buffer threshold | |
| 83 | 123 | 8.3.2.18 | Set maximum queued buffers | |
| 84 | 124 | 8.3.2.19 | Set maximum queued I-frames | |
| 86 | 126 | 8.3.2.20 | Set logical channel group number | Packet Level Restart |
| 87 | 127 | 8.3.2.21 | Set number of switched VCs | Packet Level Restart |

### 8.3.2.1 Link Level Enable Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 1 | 1 | – | No parameters. |

This parameter controls whether the link level protocol is enabled to attempt to enter data transfer state. This parameter does not have an associated value.

### 8.3.2.2 Link Level Disable Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 0 | 1 | – | No parameters. |

This parameter disables the physical link. When link level is disabled, no packet-level traffic occurs.

### 8.3.2.3 Loopback Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 66 | 2 | 0 | No loopback (default) |
| | | 1 | External loopback |
| | | 3 | Internal loopback |

The loopback parameter enables the ACP 629 to operate with the communications circuit by looping the data back into the ACP 629. During external loopback, it is assumed that the data stream is being looped back to the ACP 629 through the external communications circuit such as a modem set to loopback or through a loopback connector. Internal loopback causes the ACP 629 to loop back the data stream within itself. The loopback parameter takes effect when a link level enable parameter is processed.

### 8.3.2.4 DTE/DCE Mode Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 67 | 2 | 0 | Front end operates as DTE (default) |
| | | 1 | Front end operates as DCE |

If two ACP 629s are to be used together without an X.25 network between them, one of the ACP 629s must act as a DCE. This parameter controls whether the ACP 629 performs the X.25 protocol as a DTE or as a DCE. Note that changing from DTE to DCE mode automatically swaps the DTE/DCE addresses. The DTE/DCE parameter takes effect when link level is enabled.

### 8.3.2.5 DTE Address Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 68 | 2 | 0-255 | DTE station address (default 3) |

This parameter controls the address used by the link level protocol to reference the DTE station. Initially, the DTE address equals 03, and should not be altered except for unique network requirements. This parameter becomes active when link level is enabled.

### 8.3.2.6 DCE Address Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 69 | 2 | 0-255 | DCE station address (default 1) |

This parameter controls the address used by the link level protocol to reference the DCE station. Initially, the DCE address equals 01, and should not be altered except for unique network requirements. This parameter becomes active when link level is enabled.

### 8.3.2.7 I-Frame Timeout Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 70 | 2 | 2-255 | I-Frame timeout in seconds (default 3) |

This parameter controls the link level timeout for the acknowledgement of transmitted I-Frames. This parameter becomes effective when link level is enabled.

### 8.3.2.8 Poll Timeout Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 71 | 2 | 2-255 | Poll timeout value in seconds (default 3) |

This parameter controls the timeout of acknowledgments to link level commands sent with the poll bit turned ON. This parameter becomes active when link level is enabled.

### 8.3.2.9 ADM Timeout Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 72 | 2 | 2-255 | ADM timeout value in seconds (default 3) |

This parameter controls the length of time that the link level will wait before reattempting link setup following a link setup failure. This parameter becomes active when link level is enabled.

#### 8.3.2.10 Retry Limit Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 74 | 2 | 0-255 | Maximum number of unsuccessful retransmissions (default 20) |

This parameter controls the number of unsuccessful retransmissions that the link level will attempt before invoking error recovery. This parameter becomes effective when link level is enabled.

#### 8.3.2.11 Watchdog Timeout Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 75 | 2 | 2-255 | Watchdog timeout value in seconds (default 3) |

This parameter controls the length of time allowed for the transmission of a frame on the communications circuit. If a frame has not completed transmission in this interval, the communications circuit is assumed to have failed and the frame is aborted. The watchdog timeout value should be less than or equal to all other timeout values. This parameter becomes effective at link level enable.

#### 8.3.2.12 Bit Rate Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 76 | 2 | 0-255 | Bit rate indicator (table 8-9) (default 9600 bits/sec) |

This parameter controls the speed of the internal clock. Line speed is affected only if this clock source is connected (see 2.6). The highest recommended bit rate is 57.6k bits/sec. This parameter takes effect immediately.

Table 8-9. Bit Rate

| Nominal Bit Rate | Bit Rate Indicator (Decimal) |
|---|---|
| 800 | 00 |
| 1200 | 17 |
| 1760 | 34 |
| 2150 | 51 |
| 2400 | 68 |
| 4800 | 85 |
| 9600 | 102 |
| 19200 | 119 |
| 28800 | 136 |
| 32100 | 153 |
| 38400 | 170 |
| 57600 | 187 |
| 76800 | 204 |
| 115200 | 221 |
| 153600 | 238 |
| 316000 | 255 |

### 8.3.2.13 Idle Poll Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 77 | 2 | 0 | No idle polling |
| | | 1 | Enable idle polling (default) |

This parameter controls whether link level will send periodic supervisory commands on the communication line when no data is being transmitted. The rate is determined by the I-frame timeout value described in 8.3.2.7. This parameter becomes active when link level is enabled.

### 8.3.2.14 Frame Window Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 78 | 2 | 1-7 | Frame window size (default 7) |

This parameter limits the number of I-frames the ACP 629 can have outstanding (unacknowledged) at any time. This parameter does not affect the remote frame window size (number of frames that can be sent to the ACP 629 before waiting for acknowledgement), which can be any value up to 7. The value is not normally altered. This parameter becomes effective when the link is enabled.

### 8.3.2.15 Packet Window Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 79 | 2 | 1-7 | Packet window size (default 2) |

This parameter controls the size of the packet level window for all virtual circuits. This count is the number of unacknowledged X.25 packets that the ACP 629 can have outstanding. This parameter becomes effective when packet level is restarted.

### 8.3.2.16 High Buffer Threshold Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 81 | 2 | – | Contact ACC |

This parameter controls when the ACP 629 can once again allocate large onboard buffers for data transmission on the network. It is used for fine tuning on high speed networks and must not be altered without contacting ACC.

### 8.3.2.17 Low Buffer Threshold Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 82 | 2 | – | Contact ACC |

This parameter controls the point at which the ACP 629 must stop allocating large on board buffers for data transmission purposes. It is used for fine tuning on high speed networks and must not be altered without contacting ACC.

### 8.3.2.18 Maximum Number of Queued Buffers.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 83 | 2 | – | Contact ACC |

This parameter controls the maximum number of buffers that can be queued for transmission. This is a critical parameter and must not be altered without contacting ACC.

### 8.3.2.19 Maximum Number of Queued I-Frames.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 84 | 2 | – | Contact ACC |

This parameter controls the maximum number of I-frames that can be queued for transmission. This is a critical parameter and must not be altered without contacting ACC.

### 8.3.2.20 Logical Channel Group Number Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 86 | 2 | 0-15 | Logical channel group number (default 0) |

This parameter supports networks on which virtual calls must use a logical channel group number (the high-order four bits of the 12-bit circuit identifier in the X.25 packet header) other than zero. This value is not normally altered. This parameter becomes effective when packet level is restarted.

### 8.3.2.21 Switched Virtual Circuit Limit Parameter.

| Command Identifier | Command Length | Parameter Values | Parameter Meanings |
|---|---|---|---|
| 87 | 2 | 1-32 | Switch virtual circuit limit (default 32) |

This parameter is for situations or networks in which fewer than 32 virtual circuits are desired. It limits the number of virtual circuits that can be active at any time. This value is not normally altered. This parameter becomes effective when packet level is restarted.

**8.3.2.22 System Parameter Message Example.** An example system control message to change the ACP 629 from a DTE to a DCE and to change the bit rate from 9600 to 57.6k bits/sec is explained in table 8-10.

Input Line:

```
@L 96 0 0 6 0 67 1 76 187 1
```

Table 8-10. System Control Message
Example Explanation

| Byte | Byte Value | Remarks |
|---|---|---|
| | L | Low-level command identifier |
| 0 | 96 | Set system parameter command code |
| 1 | 0 | Reserved |
| 2 | 0 | Reserved |
| 3 | 6 | Number of bytes to follow |
| 4 | 0 | Link disable parameter |
| 5 | 67 | DTE/DCE mode parameter |
| 6 | 1 | Designate it as DCE |
| 7 | 76 | Bit rate parameter |
| 8 | 187 | Set bit rate to 57.6k bits/sec |
| 9 | 1 | Link enable parameter |

### 8.3.3 Line Status.

Table 8-11. Line Status Format

| Byte | Meaning |
|---|---|
| 0 | Command code (97) |
| 1 | Reserved |
| 2 | Status code (table 8-12) |
| 3 | 0 |

The line status message is an asynchronous message generated by the ACP 629. The message is transmitted to the first available supervisory port whenever the status of the physical link changes. If no supervisory port is available, the message is discarded. Table 8-12 describes the status codes returned by the Line Status message.

Table 8-12. Line Status Codes

| Code | Meaning | Explanation |
|---|---|---|
| 0 | Line down | HDLC link is down because the remote host is not responding, the remote host has sent a DISC command, or the remote host responded with a DM to a SABM sent by the ACP 629. |
| 1 | Line up | HDLC link is up. Either the remote host has reset the link (sent a SABM), the ACP 629 has successfully brought the link up after receiving a Line Enable system control command, or the ACP 629 has received a Line Enable system control command while the link was up. |
| 2 | Line disabled | HDLC link has been disabled in response to a Line Disable system control command. |

## 8.3.4 Virtual Circuit Query.

Table 8-13. Virtual Circuit Query Format

| Byte | Meaning |
|---|---|
| 0 | Command code (129) |
| 1 | Reserved (must be 0) |
| 2 | Virtual circuit number |
| 3 | Number of bytes to follow (must be 0) |

The virtual circuit query command is used to access information about a virtual circuit. The information returned gives a static picture of the status/state of the virtual circuit.

The logical channel states in table 8-15 are described in detail in *CCITT Recommendation X.25*. The virtual circuit states described in table 8-16 describe the local state of the virtual circuit. The response to the virtual circuit query is the same as the response to the logical channel query.

Table 8-14. Virtual Circuit Query Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (128) |
| 1 | Reserved |
| 2 | Virtual circuit number. |
| 3 | Response length |
| 4 | Reserved |
| 5 | Virtual circuit number or 377 if inactive |
| 6 | P(S) for receive side |
| 7 | P(R) for receive side |
| 8 | Receive window |
| 9 | Receive flags |
| 10 | P(S) for transmit side |
| 11 | P(R) for transmit side |
| 12 | Transmit window |
| 13 | Transmit flags |
| 14-15 | Address of first packet on queue to frame level. |
| 16-17 | Address of last packet on queue to frame level. |
| 18 | X.25 logical channel state (table 8-15) |
| 19 | Virtual circuit state (table 8-16) |
| 20-21 | Reserved |
| 22 | Host port number times 2 |
| 23 | Flags for virtual circuit |
| 24-25 | Address of first buffer on queue to host. |
| 26-27 | Address of last buffer on queue to host. |
| 28-29 | Address of current input buffer to host. |
| 30-31 | Address of current output buffer from host. |

Table 8-15. X.25 Logical Channel States

| Code | Meaning |
|---|---|
| 1 | P1_READY |
| 2 | P2_DTE_WAITING |
| 3 | P3_DCE_WAITING |
| 4 | P4_ (processed as G1,G2,D2,D3) |
| 5 | P5_CALL_COLLISION |
| 6 | P6_DTE_CLR_REQ |
| 7 | P7_DCE_CLR_IND |
| 8 | G1_FLOW_CNTRL_RDY |
| 9 | D2_DTE_RESET_REQ |
| 10 | D3_DCE_RESET_IND |
| 11 | G2_FLOW_CNTRL_BLOCKED |

Table 8-16. Virtual Circuit States

| Code | Meaning | Explanation |
|---|---|---|
| 0 | READY | IDLE |
| 1 | Awaiting answer from PAD | The X.25 layer has received a call from the network and is waiting for the PAD layer code to accept or reject the call. |
| 2 | Awaiting call accepted from network | The host has initiated a call and the ACP 629 is waiting for a response. |
| 3 | Open & ready to send data | |
| 4 | In reset sequence | The X.25 layer has received a reset from the network and is waiting for an acknowledgement from the PAD layer. |
| 5 | Awaiting clear confirmation from PAD. | The X.25 layer has received a clear request from the network and is waiting for an acknowledgement from the PAD layer. |
| 6 | Awaiting clear confirmation from network. | The PAD layer has cleared a call and the X.25 layer is waiting for a response from the network. |
| 7 | Waiting reset response | X.25 has requested a reset and is awaiting confirmation from the network. |

### 8.3.5 Logical Channel Query.

Table 8-17. Logical Channel Query Format

| Byte | Meaning |
|---|---|
| 0 | Command code (131) |
| 1 | LCN * 2 |
| 2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The logical channel query command is used to access information about a specific logical channel. For the ACP 629, the Logical Channel Number (LCN) for a given port is derived by the formula: LCN = Port Number + 1. The information returned gives a static picture of the status/state of the logical channel.

The logical channel states in table 8-15 are described in detail in *CCITT Recommendation X.25*. The virtual circuit states described in table 8-16 describe the local state of the virtual circuit. The response to the logical channel query is the same as the response to the virtual circuit query.

Table 8-18. Logical Channel Query Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (130) |
| 1 | Logical channel number |
| 2 | Reserved |
| 3 | Response length |
| 4 | Reserved |
| 5 | Virtual circuit number or 377 if inactive |
| 6 | P(S) for receive side |
| 7 | P(R) for receive side |
| 8 | Receive window |
| 9 | Receive flags |
| 10 | P(S) for transmit side |
| 11 | P(R) for transmit side |
| 12 | Transmit window |
| 13 | Transmit flags |
| 14-15 | Address of first packet on queue to frame level |
| 16-17 | Address of last packet on queue to frame level |
| 18 | X.25 logical channel state (table 8-15) |
| 19 | Virtual circuit state (table 8-16) |
| 20-21 | Reserved |
| 22 | Host port number times 2 |
| 23 | Flags for virtual circuit |
| 24-25 | Address of first buffer on queue to host |
| 26-27 | Address of last buffer on queue to host |
| 27-29 | Address of current input buffer to host |
| 30-31 | Address of current output buffer from host |

### 8.3.6 System Error Query Command.

Table 8-19. System Error Query Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (135) |
| 1-2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The system error query command accesses the ACP 629 system error table. The ACP 629 keeps a table containing all the errors that have occurred since the last system reset (powerup). The response to the system error query contains all of the current entries in the system error table with pointers to the next entry to be used in the table and the index of the last entry accessed.

Table 8-20. System Error Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (134) |
| 1-2 | 0 |
| 3 | Response length |
| 4 | Index of next entry in error table |
| 5 | Index of last used entry in error table |
| 6 | Count of errors encountered (N) |
| 7-38 | Error table (table 8-21). Each entry is 4 bytes long. |

Table 8-21. Error Table Entry Format

| Byte | Meaning |
|---|---|
| 0 | Error code. The error codes are in the lower six bits of this byte. The high order bits are used internally and should be masked off to yield the error codes listed in table 8-22. |
| 1 | Reserved |
| 2-3 | ACP 629 internal program counter where error occurred. |

Table 8-22. System Error Response Error Codes

| Status Code | Meaning |
|---|---|
| 4 | Frame level attempted to retransmit a non-existent frame. |
| 5 | CMDR received by frame level. |
| 6 | Frame level found itself in an undefined state. |
| 7 | Attempt to send call-accepted packet from the wrong state. |
| 8 | Attempt to send call-request packet on an active logical channel. |
| 9 | Attempt to send clear-confirm packet from the wrong state. |
| 10 | Attempt to send data packet on virtual circuit zero. |
| 11 | Attempt to send data packet from wrong state. |
| 12 | Attempt to send interrupt packet from wrong state. |
| 13 | Attempt to send interrupt-confirm packet from wrong state. |
| 14 | Attempt to send reset-confirm packet from wrong state. |
| 15 | Attempt to send RR or RNR packet from wrong state. |
| 16 | Packet level was requested to transmit a poorly-formed packet. |
| 17 | Packet level was given a packet for an invalid logical channel number. |
| 18 | A packet from the DCE had an invalid p(s). |
| 19 | Invalid packet received while in state p1. |
| 21 | Invalid packet received while in state p2. |
| 22 | Invalid packet received while in state p3. |
| 23 | Invalid packet received while in state p5. |
| 24 | Invalid packet received while in state p6. |
| 25 | Invalid packet received while in state p7. |
| 26 | Invalid packet received while in state d1. |
| 27 | Invalid packet received while in state d2. |
| 28 | Invalid packet received while in state d3. |
| 29 | Illegible packet received from DCE. |
| 30 | A virtual circuit was found to be in an undefined state. |
| 33 | A packet from the DCE contained an invalid p(r). |
| 35 | Free list exhausted. |
| 36 | The buffer monitor was unable to account for all buffers. |
| 37 | A transfer to or from the host failed. |
| 38 | A write completed on a logical unit assigned for reading. |
| 39 | The output side of a logical channel was in an undefined state. |
| 40 | A read completed on a logical unit assigned for writing. |
| 41 | The input side of a logical channel was in an undefined state. |

| Status Code | Meaning |
|---|---|
| 42 | I/O was attempted to an inactive logical unit. |
| 43 | Invalid supervisory command received while in data transfer state. |
| 44 | Invalid supervisory command received while in answer-wait state. |
| 45 | Invalid supervisory command received while in call-wait state. |
| 46 | Invalid supervisory command received while in idle state. |
| 47 | A virtual circuit was found to be in an undefined state. |
| 48 | Attempt to reassign an active logical unit number. |
| 49 | NCP received an undefined command. |
| 50 | A supervisory command specified an invalid virtual circuit number. |

### 8.3.7 System History Query Command.

Table 8-23. System History Query Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (137) |
| 1-2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The system history query command gives a history of packet types and counts. The history table has 16 entries. Each entry is 8 bytes long and indicates who made the entry and contains a place for a time stamp and 4 additional bytes of information. The additional information is specific to the type of history entry. The table entries wrap around, showing the last 16 events in the system. You can find the most recent entry in the table by subtracting the offset field value from 17.

Table 8-24. System History Response Format

| Byte | Meaning |
|---|---|
| 0 | Response code (136) |
| 1-2 | Reserved |
| 3 | Response length |
| 4 | Index of most recent entry |
| 5-133 | History entries (table 8-25) |

Table 8-25. History Entry Format

| Byte | Meaning |
|---|---|
| 0 | Entry type (table 8-26) |
| 1-2 | Time stamp in seconds |
| 3 | Time stamp extension in hundredths of a second |
| 4 | X0 (table 8-26) |
| 5 | X1 |
| 6 | X2 |
| 7 | X3 |

Table 8-26. X-Field Meanings

| Type | Meaning | X0 | X1 | X2 | X3 |
|---|---|---|---|---|---|
| 1 | Receive frame | Count | Frame Address | Frame Type | 0 |
| 2 | Send frame | Count | Frame Address | Frame Type | 0 |
| 3 | Receive packet | Count | Packet Format | Packet Logical Channel | Packet Type |
| 4 | Send packet | Count | Packet Format | Packet Logical Channel | Packet Type |
| 5 | NCP receive command | -- NCP Command Header -- | | | |
| 6 | NCP send command | -- NCP Command Header -- | | | |
| 7 | PDP receive packet | Count | Packet Format | Packet Logical Channel | Packet Type |
| 8 | PDP send packet | Count | Packet Format | Packet Logical Channel | Packet Type |

**NOTE**

When X0 contains a Count, 255 means the length was greater than 254.

### 8.3.8 Frame Query Command.

Table 8-27. Frame Query Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (139) |
| 1-2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The frame query command queries most of the values setable using the system control command (section 8.3.2). It also lists the status of the physical line, network, front end, frame level, and restart state.

Table 8-28. Frame Query Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (138) |
| 1-2 | 0 |
| 3 | Response length |
| 4 | Non-zero if line is enabled |
| 5 | Non-zero if network is ready |
| 6 | Non-zero if FEP is ready |
| 7 | Non-zero if transmitter is active |
| 8 | Frame level state (table 8-29) |
| 9 | Initiator substate |
| 10 | Responder substate |
| 11 | Restart state (table 8-30) |
| 12 | T1 timer |
| 13 | N2 counter |
| 14 | Watchdog timer |
| 15 | Reserved |
| 16 | Loopback flag |
| 17 | Local frame address |
| 18 | Remote frame address |
| 19 | Idle poll flag |
| 20 | I-frame timeout value |
| 21 | Poll timeout value |
| 22 | Disconnect timeout value |
| 23 | Retry (N2) limit |
| 24 | Watchdog timeout |
| 25 | VA |
| 26 | RA |
| 27 | VS |
| 28 | VR |
| 29-36 | Reserved |
| 37 | Retransmission flag |
| 38 | Poll flag |

Table 8-29. Frame Level States

| Code | Meaning | |
|---|---|---|
| 0 | Disconnected | The frame level is down. |
| 1 | UA/SABM wait | The frame level has sent a SABM and is waiting for a UA. |
| 2 | UA/DISC wait | The frame level has sent a DISC and is waiting for a UA. |
| 3 | Ready | The frame level is up. |

Table 8-30. Restart States

| State | Meaning |
|---|---|
| 0 | No restart pending (system up). |
| 1 | Restart confirmation required from both host and network. |
| 2,4 | Restart confirmation required from network. |
| 3,5 | Restart confirmation required from host. |

### 8.3.9 Memory Query Command.

Table 8-31. Memory Query Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (141) |
| 1-2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The memory query command lists memory usage on the ACP 629. This information can be used to check how the ACP 629 is handling inbound and outbound traffic.

Table 8-32. Memory Query Command Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (140) |
| 1-2 | 0 |
| 3 | Response length |
| 4 | Count of received I-frames queued for packet level processing |
| 5 | Count of buffers waiting for outbound frame level processing, or waiting for frame level acknowledge |
| 6 | Count of buffers waiting for outbound packet level processing |
| 7 | Count of buffers waiting to be transmitted to host |
| 8 | Count of buffers waiting to be filled by host |
| 9 | Buffers low flag |
| 10 | Count of times buffer-low state was entered |
| 11 | 0 |
| 12 | Number of free large buffers |
| 13-16 | Reserved |
| 17 | Total number of large buffers |
| 18 | Number of free small buffers |
| 19-22 | Reserved |
| 23 | Total number of small buffers |
| 24-26 | Reserved |

### 8.3.10 Statistics Query Command.

Table 8-33. Statistics Query Command Format

| Byte | Meaning |
|---|---|
| 0 | Command code (143) |
| 1-2 | Reserved (must be 0) |
| 3 | Number of bytes to follow (must be 0) |

The statistics query command is useful for systems that need to tune the ACP 629 based on loading. The response contains counts of all of the frame types transmitted and received and elapsed time since the last query. All counters are reset to zero after a system reset or a statistics query command.

Table 8-34. Statistics Response Format

| Byte | Meaning |
|---|---|
| 0 | Response type (142) |
| 1-2 | 0 |
| 3 | Response length |
| 4-7 | Uptime in seconds since FEP was reset |
| 8-11 | Elapsed time in seconds |
| 12-15 | Frames transmitted |
| 16-19 | Bytes transmitted |
| 20-23 | Frames received |
| 24-27 | Bytes received |
| 28-29 | SABMs sent |
| 30-31 | SABMs received |
| 32-33 | DISCs sent |
| 34-35 | DISCs received |
| 36-37 | FRMRs sent |
| 38-39 | FRMRs received |
| 40-41 | RRs sent |
| 42-43 | RRs received |
| 44-45 | REJs sent |
| 46-47 | REJs received |
| 48-49 | RNRs sent |
| 50-51 | RNRs received |
| 52-53 | CRC errors |
| 54-55 | Reserved |
| 56-57 | I-frames retransmitted |
| 58-59 | UAs sent |
| 60-61 | UAs received |
| 62-63 | DMs sent |
| 64-65 | DMs received |
| 66-67 | I-frames sent |
| 68-69 | I-frames received |

# Chapter 9

# UNIBUS Interface Programming Specification

## 9.1 Introduction

This chapter provides information for driver writers. If an ACC-supplied driver is being used, skip this chapter.

## 9.2 Functional Description

The ACP 629 interface enables the ACP 629 and host to transfer data and commands across the UNIBUS. The ACP 629 uses four mailboxes in the communication registers (COMREGs) to do the following functions:

1. Host to ACP 629 command/data buffer transfer.
2. Host to ACP 629 data description.
3. ACP 629 to host command/data character transfer.
4. ACP 629 transmission completion indication.

This interface controls information exchange and interrupt mechanisms that indicate information availability. Incoming data is forwarded and completion notifications are returned to the host. Data transfers between the host and ACP 629 are controlled by the following notifications:

1. The host must notify the ACP 629 when an output request or command from the host application code is queued on ports 0 through 31. (Note that only one outstanding output request per port is legal).
2. The ACP 629 must notify the host when each host output request has completed.
3. The ACP 629 must notify the host of incoming commands and character data from the network.

## 9.3 UNIBUS Communication Registers

Control information is exchanged between the host and the ACP 629 through the COMREGs, which occupy sixteen contiguous words in UNIBUS address space. The COMREGs include one hardware-defined word and fifteen firmware-defined words. The following sections describe the information placed in these COMREGs by the host and the ACP 629.

**9.3.1 Communication Registers' UNIBUS Addresses.** The UNIBUS base address of this block is set by the U82 DIP switch on the ACP 629, which sets UNIBUS address bits 5 through 12. This DIP switch can specify a UNIBUS base address from 760000 to 777740 (octal).

```
                         UNIBUS Address
                          (octal word)
        Registers
     |--------------|
     |     UCSR     |   7wxy00 or 7wxy40
     |--------------|
     | Firm CSR #1  |     UCSR + 002
     |--------------|
     | Firm CSR #2  |     UCSR + 004
     |--------------|
     | Firm CSR #3  |     UCSR + 006
     |--------------|
     /              /
     /              /
     |--------------|
     | Firm CSR #15|     UCSR + 036
     |--------------|

  (w = 6 or 7     x and y = 0 through 7)
```

Figure 9-1. Communication Registers' UNIBUS Addresses

**9.3.2 UNIBUS Control and Status Register (UCSR).** The UCSR comprises one word with hardware-defined bits through which the ACP 629 and the host exchange control information. Table 9-1 outlines which bits can be set by the host. The host can reset the ACP 629 (bit 1), enable/disable the ACP 629 to do DMA on the UNIBUS (bits 3 and 4), enable/disable the ACP 629 to interrupt the UNIBUS (bit 2), and request an ACP 629 interrupt (bit 0).

Table 9-1. UNIBUS Control and Status Register Bit Definitions

| Bit No. | UNIBUS R:W | ACP R:W | Bit Name | Definition |
|---|---|---|---|---|
| 0 | 0/W | R0 | NMI | Non-maskable interrupt<br>write 1: Wake up ACP 629<br>write 0: Has no effect |
| 1 | 0/W | R0 | RST | Reset<br>write 1: Reset ACP 629<br>write 0: Has no effect |
| 2 | R/W | R | IEN | UNIBUS Interrupt enable<br>1 = ACP can request interrupt<br>0 = ACP cannot request interrupt |
| 3 | R/W | R | WRT | DMA change enable<br>1 = ACP can change host memory<br>0 = ACP cannot change host memory |
| 4 | R/W | R | DMA | DMA access enable<br>1 = ACP can access host memory<br>0 = ACP cannot access host memory |
| 5-15 | | | | Reserved |

0/W = always reads 0; set bit by writing 1  R0 = always reads 0
R = read only  R/W = read/write

**9.3.2.1 Interrupt Mechanism.** The ACP 629 gives the host one interrupt mechanism. The host requests an ACP 629 interrupt by writing a 1 to UCSR bit 0.

In the onboard application code, the ACP 629 provides two vectored interrupts to the UNIBUS. The ACP 629 interrupts the host by generating an interrupt to either of these vectors. Because of the nature of the interrupt queue on the front end, interrupts might be overwritten and lost. We therefore suggest that both interrupt vectors point to the same interrupt routine. This interrupt routine should check all ACP 629 mailboxes every time it is called so that events are not lost or delayed.

**9.3.2.2 Sample Initialization Scenario.** When the host brings the ACP 629 online, the host should do as follows:

1. Clear the COMREGs.
2. Set the HST flag in both the Front End to Host Data Transfer (displacement 18 hex) and Front End Complete (displacement 1B hex) mailboxes in the COMREGs (See figure 9-2).
3. Divide the UNIBUS interrupt vector by four and load the results into the COMREGs (VECTOR at displacement 1F hex). This address is the UNIBUS location of the vector used for interrupts by the ACP 629. The address of the second vector used for interrupts from the ACP 629 is offset 4 bytes from this the address of the first vector.
4. Reset the ACP 629 by writing 1 to the RST field in the CSR.
5. Mark the status of the ACP 629 as coming-on-line.

After the ACP 629 initializes itself and is ready to accept data traffic, it sets the ACP flag in the Host to Front End Data Ready Mailbox (displacement 1A hex). The host should then do the following:

1. Mark the status of the ACP 629 as online
2. Enable ACP 629 interrupts by setting the IEN and DMA flags in the CSR.

After a reset, the only indication the host has that the ACP 629 is online is the setting of the ACP flag in the Host to Front End Data Ready Mailbox. The host software can assume that if the ACP flag in the Host to Front End Data Ready Mailbox has not been set in 3 to 5 seconds, the ACP 629 will not be coming online (possibly a hardware error).

## 9.4 Firmware-Defined Communication Registers

The ACP 629 uses the last four words of the COMREGs, defined by the DIP switch at U82, as its mailbox interface to the host. These four words are divided into four separate mailboxes that the ACP 629 and the host use to pass information across the UNIBUS. Figure 9-2 shows the physical layout of these mailboxes.

```
                | MS (odd) byte | LS (even) byte|      COMREG
          bits  |15............8|7............0 |      offset
----------      |-------------------------------|
                |                     D W I R N |        00
   CSR          |                     M R E S M |
                |                     A T N T I |
----------      |###############################|
                |/ / / / / / / / / / / / / / / /|        02
 Reserved       | / / / / / / / / / / / / / / / |
                |/ / / / / / / / / / / / / / / /|
----------      |###############################|
 Firmware                       |               |        04
 Version                        |    Version    |
                                |               |
----------      |###############################|
                |/ / / / / / / /|
                | / / / / / / / |
                |/ / / / / / / /|
 Reserved       | / / / / / / / ################|
                |/ / / / / / / / / / / / / / / /|
                | / / / / / / / / / / / / / / / |
                |/ / / / / / / / / / / / / / / /|
----------      |###############################|
 Front End      |   CHARACTER   |A H C |        |        18
  to Host       |      OR       |C S M |  PORT  |
   Data         |    COMMAND    |P T D |        |
----------      |###############################|
 Host to                        |A H C |        |        1A
Front  End                      |C S M |  PORT  |
Data Ready                      |P T D |        |
----------      |###############################|
 Front End      |A H C |        |
 Complete       |C S M |  PORT  |
                |P T D |        |
----------      |###############################|
                |               |   COMMAND     |        1C
                |               ################|
                |          OUTPUT DATA          |
 Host to        |            ADDRESS            |
Front End       |                               |
  Data          |###############################|
Description     |   A E |/ / / |               |        1E
                |   D X | / / /|    OUTPUT     |
                |   R T |/ / / |  BYTE  COUNT  |
                |###############|               |
                | INTR. VECTOR  |               |
----------      +-------------------------------+
```

Least significant (LS) and most significant (MS) bytes are defined from host viewpoint.

Figure 9-2. Detailed COMREG Layout

**9.4.1 Front End to Host Data Transfer.** The ACP 629 uses the word at displacement 18 hex in the COMREGs as a mailbox to transfer data from the network into the host a byte at a time. Table 9-2 shows the layout of this mailbox and figure 9-2 shows its location within the COMREGs.

When the ACP 629 has a data byte or command to transfer to the host, the ACP 629 uses the following algorithm:

1. The interrupt pending flag is tested. If it is set, the ACP 629 attempts the transfer later.

2. The HST flag is tested to see if the host has received the last information byte. If the HST flag is 0, the ACP 629 attempts the transfer later.

3. The port that the current byte came from is loaded into bits 0-4.

4. The byte to be transferred is loaded into bits 8-15.

5. If the byte to be transferred is a command (see table 9-3) the CMD flag (bit 5) is set to 1.

6. The HST flag (bit 6) is set to 0 and the ACP flag (bit 7) is set to one.

7. An input interrupt is generated to the host.

When the host receives the interrupt, it checks the ACP flag, for this mailbox, to see if there is incoming data. If there is incoming data, the host uses the following algorithm:

1. Check the CMD flag to find out how to interpret the byte.

2. If the CMD flag is set, the byte is processed as a command for the port. Otherwise, the byte is transferred to the data destination for the port (usually a readahead buffer in the terminal device driver).

3. After reading the current byte, the host sets the ACP flag to 0, the CMD flag to 0 and the HST flag to 1.

4. When character processing completes, the host should reexamine the ACP flag because the ACP 629 will try to send additional characters/commands under a single host interrupt. Tests done by ACC show that over 100 characters can be processed under a single host interrupt.

The flags ACP and HST must never contain the same value when the host exits its incoming data interrupt routine. If these flags contain the same values after the host has exited its interrupt processing routine, the ACP 629 will not transfer data correctly.

Table 9-2. Front End to Host Data Transfer Mailbox

| Field | Displacement | Bits | Value | Meaning |
|---|---|---|---|---|
| Port | 0x18 | 0-4 | 0-31 | Data path number for data transfer. |
| CMD | | 5 | 1 | Command from front end. |
| | | | 0 | Data from front end (initial value). |
| HST | | 6 | 1 | Host ready for data (initial value). |
| | | | 0 | Host not ready for data. |
| ACP | | 7 | 1 | ACP data Present. |
| | | | 0 | No ACP data present (initial value). |
| Character | | 8-15 | 0-255 | Character data or command (see table 9-3) from front end. |

Table 9-3. Commands From Front End to Host

| Value | Meaning |
|---|---|
| 1 | Break indication |
| 2 | X.25 call established |
| 3 | X.25 call cleared |
| 5 | Resumed data flow |
| 6 | Suspended data flow |

**9.4.2 Host to Front End Data Transfer.** Tables 9-4 and 9-6 describe the mailboxes that the host uses to transfer data to the ACP 629. The following sections describe how the host uses the Host to Front End Data Description Mailbox (displacement 1C hex) and the Host to Front End Data Ready Mailbox (displacement 1A hex) to transfer data to the ACP 629.

**9.4.2.1 Host to Front End Data Description.** The Host to Front End Data Description Mailbox (table 9-4) has three formats depending on the situation or the data being transferred.

The first format is used during initialization. The only field in use is the VECTOR field (bits 8-15 of the word at displacement 1E hex). The rest of the mailbox is unused. See section 9.3.2.2 for more information on the initialization of the ACP 629.

The second format is used when a data buffer is to be transferred to the ACP 629. This format uses the ADDRESS field (bits 0-15 of the word at displacement 1C) plus the two address extension bits (labeled as ADR EXT in figure 9-2 and table 9-4) to locate the data buffer in memory. The address extension bits (bits 12-13 of the word at displacement 1E hex) are used to store the most significant bits of the 18 bit memory address. The data transfer format uses the BYTE COUNT field (bits 0-7 of the word at displacement 1E hex) to transfer the length in bytes of the data buffer.

The third format is used when a command is to be sent to the ACP 629 (see table 9-5). This format consists of the COMMAND field (bits 0-7 of the word at displacement 1C hex). The rest of the mailbox is unused.

Table 9-4. Host to Front End Data Description Mailbox

| Field | Displacement | Bits | Value | Meaning |
|---|---|---|---|---|
| Command | 0x1C | 0-7 | 0-255 | Command from host to front end (see table 9-5). |
| Address |  | 0-15 |  | Address of data from host to front end. |
| Byte Count | 0x1E | 0-7 | 0-255 | Length of data buffer being passed to front end. |
| Vector |  | 8-15 | 0-255 | Interrupt vector address. |
| ADR EXT |  | 12-13 | 0-3 | Address extension bits for address at displacement 0x1C. |

Table 9-5. Commands From Host to Front End

| Value | Meaning |
|---|---|
| 1 | Break indication |
| 2 | Enter PAD command mode |
| 3 | Clear call |
| 4 | Enable incoming calls |
| 5 | Resume flow |
| 6 | Suspend flow |
| 7 | Enter supervisor mode |

**9.4.2.2 Host to Front End Data Ready**. During data transfer to the ACP 629, the host should use the following algorithm:

1. Check the ACP flag (bit 7 of the byte at displacement 1A hex) in the Host to Front End Data Ready Mailbox (see table 9-6).
2. If the ACP flag is 0, wait until it becomes 1.
3. If the port with data ready to transmit has not finished its previous transmission (see section 9.4.3), queue data transmission for that port until the previous transmission is complete. (Note that commands do not receive completion notifications).
4. Load the data description or the command (table 9-5) into the Host to Front End Data Description Mailbox (section 9.4.2.1).
5. Load the port number into the PORT field (bits 0-5 of the byte at displacement 1A hex).
6. Set the CMD flag (bit 5 of the byte at displacement 1A hex) if the data to be transferred is a command.
7. Set the HST flag (bit 6 of the byte at displacement 1A hex) to 1 and the ACP flag (bit 7 of the byte at displacement 1A) to 0.
8. Interrupt the ACP 629 by writing 1 to the NMI flag (bit 0 of the CSR at displacement 0).
9. Set the port state to transmitting data if the current transfer is not a command.

The flags ACP and HST must never contain the same value when the host exits its data transmission routine. If these flags contain the same values after the host has exited its data transmission routine, the data buffer will not be transmitted correctly.

Table 9-6. Host to Front End Data Ready Mailbox

| Field | Displacement | Bits | Value | Meaning |
|---|---|---|---|---|
| Port | 0x1A | 0-4 | 0-31 | Data path number for data transfer. |
| CMD | | 5 | 1 | Command from host. |
| | | | 0 | Data from host (initial value). |
| HST | | 6 | 1 | Host data ready for transfer. |
| | | | 0 | Host data not ready transfer (initial value). |
| ACP | | 7 | 1 | ACP ready to accept data (initial value). |
| | | | 0 | ACP not ready to accept data. |

**9.4.2.3 Host to Front End Q-Bit Buffer Ready.** The ACP 629 allows the HOST to pass buffers marked as commands. These buffers have the following format:

```
        | MS (odd) byte | LS (even) byte|
bits    |15...........8|7...........0 |       offset
        |-------------------------------|
        |                               |         00
        |            HEADER             |
        |                               |
        |###############################|
        |                               |       02-END
        |         COMMAND DATA          |
        |                               |
        |/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

The 16 bit header describes the contents of the command buffer. The only valid value is 1. All other values are ignored. The value 1 specifies that the command data consists of byte pairs with the following format:

```
| MS (odd) byte | LS (even) byte|
|15............8|7............0 |
|-------------------------------|
|               |               |
| X.3 Parameter | X.3 Parameter |
|     Value     |      ID       |
|               |               |
|-------------------------------|
```

The ACP 629 sends these byte pairs as is (without checking the format or semantics) to the remote PAD in a packet with the Q-bit set (set X.3 parameters). This command enables users to dynamically access X.3 parameters on remote PADs. This feature is valid only on HOST ports with calls connected. Ports in any other state ignore the message and return an I/O completion.

During Q-Bit buffer transfer to the ACP 629, the host should use the following algorithm:

1. Check the ACP flag (bit 7 of the byte at displacement 1A hex) in the Host to Front End Data Ready Mailbox (see table 9-6).

2. If the ACP flag is 0, wait until it becomes 1.

3. If the port that has a Q-Bit buffer ready has not completed its previous transmission (see section 9.4.3), queue the Q-Bit buffer for that port until the previous transmission is complete. (Note that Q-Bit buffers receive completion notifications and must be treated as data buffers).

4. Load the buffer description into the Host to Front End Data Description Mailbox (section 9.4.2.1).

5. Load the port number into the PORT field (bits 0-5 of the byte at displacement 1A hex).

6. Set the CMD flag (bit 5 of the byte at displacement 1A hex). Because the CMD flag is set and a buffer length is set in the Host to Front End Data Description Mailbox, the ACP 629 processes the buffer being transferred as a command buffer.

7. Set the HST flag (bit 6 of the byte at displacement 1A hex) to 1 and the ACP flag (bit 7 of the byte at displacement 1A) to 0.

8. Interrupt the ACP 629 by writing a 1 to the NMI flag (bit 0 of the CSR at displacement 0).

9. Set the port state to I/O outstanding.

The flags ACP and HST must never contain the same value when the host exits its Q-Bit Buffer routine or the Q-Bit buffer will not be transmitted correctly.

**9.4.3 Front End Complete**. After the ACP 629 has accepted outbound data for a port, the ACP 629 cannot accept any more outbound data for that port until the current data buffer has been transmitted. When a data buffer for a port has been transmitted, the ACP 629 uses the Front End Complete Mailbox (see table 9-7) to notify the host that it is ready to accept another data buffer for that port. The ACP 629 uses the following algorithm to notify the host that a data transfer or command has completed:

1. Check the HST flag (bit 6 of the byte at displacement 1B hex) to see if the host is ready to accept another completion notification.

2. If the HST flag is 0, wait until it is 1 (ready) before continuing.

3. Load the port number that completed into the PORT field (bits 0-4 of the byte at displacement 1B).

4. Set the HST flag to 0 and the ACP flag (bit 7 of the byte at displacement 1B hex) to 1.

5. Generate an interrupt to the host.

When the host checks the Front End Completion Mailbox, it should use the following algorithm:

1. Check the ACP flag (bit 7 of the byte at displacement 1b hex) to see if a data buffer has completed.

2. If the ACP flag is 0, exit the completion interrupt routine.

3. Get the number of the port that completed from the PORT field (bits 0-5 in the byte at displacement 1B hex).

4. Mark the status of the port as ready for data transmission

5. Set the HST flag to 1 and the ACP flag to 0.

The flags ACP and HST must never contain the same value when the host exits its data transmission completion routine. If these flags contain the same values after the host has exited its data transmission completion routine, completion notifications will not be transmitted to the host correctly.

Table 9-7. Front End Complete Mailbox

| Field | Displacement | Bits | Value | Meaning |
|---|---|---|---|---|
| Port | 0x1B | 0-4 | 0-31 | Data path number for data transfer. |
| CMD | | 5 | 0 | No meaning. |
| HST | | 6 | 1 | Host ready for transfer complete (initial value). |
| | | | 0 | Host not ready for transfer complete |
| ACP | | 7 | 1 | ACP transfer complete. |
| | | | 0 | ACP transfer not complete (initial value). |

# APPENDIX A

# Installation Verification Problems

## A.1 Introduction

This appendix describes problems that might occur while verifying the installation of the ACP 629.

## A.2 ACP 629 X.25 Line Connection

This attributes of the ACP 629 serial line signals and some problems with typical configurations follow.

**A.2.1 ACP 629 Serial Line Signals**. Appendix G lists the pin numbers and signals names for RS-232C and RS-449/422 interfaces. The only signals required for ACP 629 operation are receive and transmit clocks and data. At initialization, the ACP 629 raises Data Terminal Ready (DTR) and Request to Send (RTS) and begins transmitting a 9600 bits/sec clock signal on terminal timing. The ACP 629 does not require Clear to Send or Data Set Ready, but if these signals are allowed to float, performance might be degraded by induced signal noise.

**A.2.2 Clock Sourcing**. The ACP 629 terminal timing signal can be used to source transmit and receive clock signals. However, doing this might not work unless RS-232C or RS-449/442 specifications for cable quality, line speed, line length and signal termination are followed.

**NOTE**

> The ACP 629 does not source transmit clocks (TT, TXC, SMTE) and must be connected to a modem or modem eliminator that sources transmit and receive clocks.

If the ACP 629 terminal timing is not required as a clock source, we recommend the pins in the connecting cable not be connected.

**A.2.3 DTE/DCE Configuration**. The ACP 629 normally operates as a DTE. Therefore the network side of the X.25 link should operate as a DCE. If you need to operate the ACP 629 as a DCE, use the set system parameters command. Note that ACP 629 DCE operation applies to link and packet level, but does not apply to physical level.

# APPENDIX B

# UNIX Manual Pages

## B.1 UNIX 4.2 BSD / ULTRIX Manual Page

NAME
xb – ACC ACP629 driver

SYNOPSIS
**optional xb device-driver**
**device xb0 at uba? csr 0166740 vector xbint xbint**

DESCRIPTION
The *xb* driver enables network connections on ACP629 to communicate with UNIX. ACC's ACP629 is a microprocessor front end which provides access to the capability of an X.25 packet-switched network (thus enabling remote computer connections). The product name "ACP629" is derived from ACP600, the front end device, plus firmware support which includes X.29. As specified by CCITT, X.29 defines procedures for communicating control information and user data between a host PAD (Packet Assembler/Disasssambler) and another PAD. The xb driver supports up to four xb devices: xb0, xb1, xb2, xb3. If during the probe of an xb device, or when trying to write to an xb line, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time, a message will be printed on the UNIX console indicating that the ACP629 device is "offline". This condition may be "cleared" by doing an open of xb minor device number (252 + unit_number), that is, minor 252, 253, 254, or 255 for units 0, 1, 2, or 3, respectively. The clear is further described in the installation instructions. The messages are described below.

**xb: xbprobe, ACP629 not responding to reset.** During the probe of the xb device, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time.

**xb: xbopen, ACP629 unit%d not responding to reset.** During an attempt to open a line, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time.

**xb: xbstart, ACP629 unit%d offline.** During the start of the xb device, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time.

**xb: xbcntl, ACP629 unit%d offline.** During the execution of a control function, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time.

**xb: xbreset, ACP629 unit%d.** This message indicates the state is reset and transmitters are restarted if UBA reset was necessary.

**xb: xbreset, ACP629 unit%d not responding to reset.** During an attempt to perform a reset, the bit in the xb CSR which indicates that the communication registers are available does not come on in a given amount of time.

**xb: xbclear, clear connections to ACP629 unit%d.** This message indicates that a "clear" operation is being performed. For the specified unit, a SIGHUP signal is sent to all process groups associated with open ports, and an immediate error return is given to all processes awaiting completion of an open on a port.

## B.2 UNIX System V / System III tj Manual Page

NAME

tj – ACC ACP629 driver

DESCRIPTION

The driver enables network connections on an ACP 629 to communicate with UNIX. ACC's ACP 629 is a microprocessor front-end which provides access to the capability of an X.25 packet-switched network (thus enabling remote computer connections). The product name "ACP 629" is derived from ACP 600, the front end device, plus firmware support which includes X.29. As specified by CCITT, X.29 defines procedures for communicating control information and user data between a host PAD (Packet Assembler/Disasssambler) and another PAD. The tj driver supports up to four tj devices: tj0, tj1, tj2, tj3.

All standard ioctl(2) system calls are supported by this driver as well as open(2), close(2), read(2), and write(2) system calls for tj devices.

ACP 629 ports may be used as login ports, as described in the ACP 629 USER'S MANUAL. The procedure consists primarily of creating the proper devices (via MKNOD(1M)) and making appropriate entries in the table /etc/inittab (see INITTAB(4)). Such ports are called "host" ports.

The "cu" command (see CU(1)) may be used to access outgoing ACP 629 ports, called "PAD" ports, after appropriate devices and uucp file entries have been made (in /usr/lib/uucp/L-devices and /usr/lib/uucp/L.sys). Refer to the ACP 629 USER'S MANUAL and "UUCP Administration", Chapter 10 in the System V Administrator Guide.

The tj device driver supports a new ioctl command, TCSPARM, which may be used to request that a specified string of pairs of bytes be passed to the ACP 629 front-end as a set parameter request. Typically, the front-end will use the data to specify values for X.3 PAD parameters in a data packet with the Q-bit set. The ioctl synopsis is as follows:

```
#include <termio.h>
#define TCSPARM (TIOC | 8)

int ioctl (fildes, request, arg)
int fildes, request;
char *arg;
```

Refer to IOCTL(2) and TERMIO(7) for general details of ioctl use. Here, fildes must be the file descriptor for an open ACP 629 host port. Request must be TCSPARM. Arg must point to a string of no more than 65 bytes, of which the first byte must contain a count of the following data bytes; the second and third bytes contain a function code and must currently have the (binary) values 1 and 0 respectively; there must be an even number of remaining bytes, which will be output over the connection as the parameter fields in a set-PAD-parameter Q-bit packet. See SETPARM(1) for details of a command which provides a user interface to this ioctl facility.

If the ioctl succeeds, it will return zero. Otherwise, it will return -1, with one of the following errno values:

| | |
|---|---|
| [EBUSY] | The ACP 629 port is not in OPOST output control mode (required by the driver for this ioctl to succeed (see TERMIO(7)). |
| [EFAULT] | The driver is unable to copy the byte string pointed to by 'arg'. Some part of the buffer may be outside the process's allocated space. |
| [EINVAL] | The specified parameter string is too long. I.e., the value of the first byte pointed to by 'arg' is greater than 64. |

## FILES

/dev/tty*, /dev/pad*

## SEE ALSO

cu(1), stty(1), setparm(1), ioctl(2), open(2), close(2), read(2), write(2), termio(7), ACP 629 USER'S MANUAL.

## DIAGNOSTICS

The driver will print diagnostic messages to the system console if the following conditions arise:

If during the first open a tj port, the bit in the tj CSR which indicates that the communication registers are available does not come on in a given amount of time the following diagnostic message is given:

**acp629: unit ? not responding to reset.**
where "?" represents the ACP 629 board number.

If during the start of the tj device, the bit in the tj CSR which indicates communication registers availability does not come on in a given amount of time, the following diagnostic message will be printed on the console:

**acp629: unit ? not responding to tjstart.**
where "?" represents the ACP 629 board number.

If during the execution of a control function, the bit in the tj CSR which indicates that the communication registers are available does not come on in a given amount of time the following message will be printed on the console:

**acp629: unit ? not responding to tjcntl.**
where "?" represents the ACP 629 board number.

If any of the timeouts described above occur, an attempt at clearing the condition may be made by issuing an open of tj minor device number (252 + unit_number), that is, minor device 252, 253, 254, or 255 for units 0, 1, 2, or 3, respectively. when an open is issued on a "clear" port, the following diagnostic message is printed on the console:

**acp629: clearing connections to unit ?** br where "?" represents the ACP 629 board number.

This message indicates that a "clear" operation is being performed. For the specified unit, a SIGHUP signal is sent to all process groups associated with open ports, and an immediate error return is given to all processes awaiting completion of an open on a port.

Additionally, a diagnostic tool is available for printing out port statistics for board 0. This tool may be used when there are up to three ACP629 boards in a system (boards 0, 1, and 2). It may not be used when a fourth board is present. Port status for board zero is obtained by issuing an open on minor device 255. The resulting message is as follows:

**acp629: state of unit ?**
**line state iflag oflag cflag lflag**
where "?" represents the ACP 629 board number.

The state of each active port is listed below this message. The message:

**remaining lines free**
indicates that all ports that were not included in the previous message are inactive.

## B.3 UNIX System V / System III SETPARM Manual Page

NAME

setparm – invoke ACP 629 TCSPARM ioctl system call

SYNOPSIS

setparm dev param_byte ...

DESCRIPTION

Setparm is used in conjunction with an ACC ACP 629 microprocessor front-end. The ACP 629 driver supports a custom ioctl, TCSPARM, which may be used to request that a specified string of pairs of bytes be sent to the ACP 629 front-end as a parameter command. Currently, the only supported application of this capability is for the output of the string of byte pairs as the parameter fields in a data packet, with the Q-bit set, to set selected X.3 parameters.

The dev argument is the pathname of an ACP 629 host port. Setparm will open this port and execute a TCSPARM ioctl system call with the specified parameter byte string (described next) as its argument.

There is an even number from two to 64 of additional arguments, each of which is a param_byte with a value from 0 to 255 decimal. The first and second param_bytes contain a function code. The only currently supported function code value consists of 1 followed by 0, which requests that the remaining even number of param_bytes be output over the connection as the parameter fields in a set-PAD-parameter Q-bit data packet. Thus, in accordance with CCITT X.29 section 4.4, these parameter byte pairs each consist of a PAD parameter number followed by a requested value for that parameter. See Appendix E of the ACP 629 User's Manual for a complete list of PAD parameters.

For example, the command "setparm /dev/tty 1 0 2 1 14 0" would request that a Q-bit data packet be sent on the virtual circuit assigned to the host port being used for the current login session, requesting PAD echo (PAD parameter 2 set to value 1) and no linefeed padding (PAD parameter 14 set to 0).

Simpler syntax for requesting setparm to send a Q-bit packet out over the currently-logged-in host port is provided by the following Bourne Shell script:

```
IFS=:
setparm /dev/tty 1 0 $*
```

If this script is named setx29, then typing "setx29 2:1 14:0", for example, will invoke the setparm command example given above.

Internally, setparm functions as follows: it determines the count of param_byte arguments; inserts the byte count at the beginning of a buffer; appends all param_byte arguments to the buffer; and invokes the TCSPARM ioctl system call with a pointer to the buffer.

SEE ALSO

open(2), ioctl(2), termio(7), tj(7), ACP 629 USER'S MANUAL.

DIAGNOSTICS

If the setparm command is issued with less than three arguments, the following diagnostic message will be given:

**Usage: setparm device param_byte...**

If the parameter string exceeds 64 bytes, the ioctl system call will not be issued and the following message will be given:

**setparm: parameter string ? bytes long; length must be < 65.**

where ? represents the size of the parameter string in bytes.

If the parameter string length is not even, a message will be printed as follows:

**setparm: parameter string ? bytes long; length must be even.**
where ? represents the size of the parameter string in bytes.

If an acceptable string is entered, a message appears as follows:

**setparm: parameter string (? bytes) 'arg list'**
where ? is equal to the parameter byte count and arg list contains the 8 bit hexadecimal value of each param_byte argument.

If the open system call issued on the device given by the dev argument returns an error, the following error message is given:

**setparm: can't open 'dev', errno = ?**
where ? represents the error code returned by open(2).

If the TCSPARM ioctl system call fails, an error message is printed as follows:

**setparm: ioctl failed, errno = ?**
where ? is the error code returned by the ioctl call (See TJ(7)).

If the TCSPARM ioctl system call is successful, a message is given as follows:

**setparm: ioctl succeeded**

# APPENDIX C

# Device Numbers For UNIX Drivers

## C.1 Introduction

The installation instructions for UNIX drivers (chapters 4, 5, and 6) refer to major and minor device numbers associated with the the ACP 629 device. This section further describes the major and minor device numbers.

**NOTE**

The examples used in this appendix refer to the UNIX 4.2 BSD xb driver. This appendix also applies to the System III and System V tj drivers. For the tj drivers, replace instances of `xb` with `tj`.

**C.1.1 Major Device Number.** One major device number is associated with the xb driver. In UNIX 4.2 BSD installation instructions, the number is 33. The major device number is determined by the position of the driver entries in the cdevsw table.

**C.1.2 Minor Device Numbers.** There are 256 minor device numbers (0 to 255) associated with the xb driver. The minor device numbers are assigned by the system manager when configuring inbound and outbound ports for the ACP 629. The minor device numbers are also used for clear devices (CLEAR ports) to do a clear operation as described in the installation instructions (see chapters 4, 5, and 6).

The inbound ports are typically /dev/tty<nn> devices (HOST ports). The outbound ports are typically /dev/pad<nn> devices (PAD ports).

Because of the eight-bit PBBLLLLL format defined for minor device numbers, the distribution of PAD and HOST ports has constraints. Minor device numbers for HOST ports range from 0 to 127, and minor device numbers for PAD ports range from 128 to 255. As shown in the following figure, four minor device numbers for outbound ports (252 to 255) are reserved for CLEAR ports which are used for software resetting the ACP 629.

The distribution of HOST and PAD ports is equally balanced unless four ACP 629s are configured in the system. If the system is configured for one, two, or three ACP 629s, half of the minor device numbers designate HOST ports, and half PAD ports. If four ACP 629s are installed (as in figure C-1), the number of PAD ports is decreased by 4, but there are still 32 HOST ports for the xb3 device. In the following figure, the xb3 unit has PAD port numbers 224 to 251 (28 ports). The ports 252 to 255 are reserved for the CLEAR ports.

```
+-----------------------------------------------------------------------+
| UNIT        HOST                            PAD                 CLEAR |
|-----------------------------------------------------------------------|
|    |                           |                                |     |
|    |                    -------|                        --------|     |
|xb3 |                    96  127|                        224  251| 255 |
|    |             -------       |                --------        |     |
|xb2 |             64   95       |                192  223        | 254 |
|    |      -------              |        --------                |     |
|xb1 |      32   63              |        160  191                | 253 |
|    |------                     |--------                        |     |
|xb0 |0   31                     |128  159                        | 252 |
|    |                           |                                |     |
+-----------------------------------------------------------------------+
```

| | |
|---|---|
| xb0 - xb3 | = ACP 629 numbers 1 through 4 |
| HOST | = inbound port |
| PAD | = outbound port |
| CLEAR | = xb driver "clear" |

Figure C-1. Minor Device Numbers for ACP 629

For ACP 629 number 1 (xb0), the HOST ports range from 0 to 31, and the PAD ports range from 128 to 159. The device reserved for the xb driver clear function has minor device number 252.

# APPENDIX D

# ACP 629 VMS Technical Notes

## D.1 Missing PAD Prompt

When a user connects to an ACP 629 PAD port, for example, using KERMIT or RTPAD (SET HOST/DTE command), the ACP 629 displays the PAD prompt (@) after the user or transfer process (e.g., RTPAD) enters one carriage return. If the PAD prompt does not appear after the initial <CR>, the port selected by the user is in one of the following states:

1. In disabled (DIS) mode
2. In enabled (ENA) mode
3. In transfer mode and not flow controlled
4. In transfer mode and flow controlled

Cases 1 and 2 are caused by a user selecting a port that is either disabled (DIS) or an inactive incoming port (ENA). In this situation, the ACP 629 tells the user that the selected port is in an invalid state. The user can escape to VMS by typing the appropriate escape sequence (e.g., <CNTRL>\ for RTPAD).

Cases 3 and 4 are caused by configuring ACP 629 ports using the /NOHANGUP terminal attribute. For more information on the /HANGUP terminal attribute and its effect on the ACP 629, see section D.3. Because of the problems caused by configuring ACP 629 ports with the /NOHANGUP terminal attribute, ACC strongly recommends that all ACP 629 ports be configured with /HANGUP.

Case 3 might allow a user to continue where a previous session left off. This might cause a security problem.

Case 4 might cause the user's process to become hung and be unable to escape back to DCL.

If a terminal becomes hung without a PAD prompt and the user cannot escape back to VMS, the system administrator must issue a STOP PROCESS/ID= command against the users process (this logs the user off), then reconfigure the port (X25CLOSE followed by either an X25PAD or an X25ENABLE as necessary).

## D.2 File Transfer and Spooled Terminals

When transferring bulk data, the throughput rate of the ACP 629 is influenced by line speed, data record size, and the availability of buffer space on the destination system (type-ahead buffer size). These factors are modified by the VMS terminal attributes ALTYPAHD and HOSTSYNC and the SYSGEN parameters TTY_ALTYPAHD and TTY_ALTALARM.

Inbound data from the ACP 629 and the network is stored in the port's type-ahead buffer. If the port has the terminal attribute /HOSTSYNC set, data is stored in the type-ahead buffer until the type-ahead buffer is filled to within 20 characters of its end. At this point, inbound data for the port is suspended until a read empties the type-ahead buffer. The suspension and resumption of the data flow does not affect the data stream (<CNTRL>S and <CNTRL>Q characters are not added to the data stream).

If /NOHOSTSYNC is set, the input fills the type-ahead buffer and VMS generates a `SYSTEM-W-DATAOVERUN` error to the destination terminal or process if the buffer is filled beyond its capacity.

To avoid buffer overrun (/NOHOSTSYNC) or suspended data flow (/HOSTSYNC), ACC recommends using the alternate type-ahead buffer (terminal attribute /ALTYPAHD) on all ports doing bulk data transfers. The size of the alternate type-ahead buffer is set by the SYSGEN parameter TTY_ALTYPAHD. The end-of-buffer alarm is set by the SYSGEN parameter TTY_ALTALARM.

ACC recommends that, for line speeds up to 9600 bits/sec, the SYSGEN parameter TTY_ALTYPAHD be set to at least 2.5 times the average record size being transferred, and that the SYSGEN parameter TTY_ALTALARM be set to 20 characters from the end of the alternate type-ahead buffer. These settings will minimize problems of excess memory allocated for buffers and excess time spent waiting for data flow to be resumed.

For line speeds higher than 9600 bits/sec, the alternate type-ahead buffer size should be increased to accommodate the increased data flow.

## D.3 HANGUP Terminal Attribute

The terminal attribute /NOHANGUP causes the ACP 629 terminal driver to maintain an X.25 connection upon process termination. When an ACP 629 port is configured with the /NOHANGUP terminal attribute, the ACP 629 is not notified when a process that owns the port is terminated. This can lead to hung or blocked ports and breaches of system security.

Because of the undesirable side effects of configuring ACP 629 ports with the /NOHANGUP terminal attribute, ACC recommends that all HOST ports (ENAbled ports) be configured with /HANGUP. This causes the ACP 629 to disconnect any X.25 session as soon as a process owning an ACP 629 port has been terminated.

As an added precaution, system administrators might consider configuring all PAD ports with /HANGUP. This causes PAD ports to disconnect X.25 sessions as soon as a user (or process) disconnects from any ACP 629 port. Configuring PAD ports with the /HANGUP terminal attribute ensures that all X25 sessions are disconnected as soon as any ACP 629 port is released.

## D.4 PASTHRU Terminal Attribute

If /PASTHRU or /NOINTERACTIVE is set on a HOST port, control characters such as <DEL>, <CNTRL>B, <CNTRL>O do not work correctly. ACC recommends that all HOST ports be configured with /NOPASTHRU and /INTERACTIVE unless there is a process-specific need for control characters to be passed through.

## D.5 EIGHTBIT and PARITY Terminal Attributes

The ACP 629 transmits all of its data as eight bit data. The ACP 629 ignores all parity terminal attributes. The SET TERM/PARITY and SET TERM/EIGHTBIT commands have no effect on the ACP629.

## D.6 X25CLOSE Command

A user with the SHARE and PHY_IO privileges can issue an X25CLOSE command for a port that is owned by another process. X25CLOSE does not delete or disconnect the process that owns the port. The owner process is therefore still attached to the port when the port is reenabled. If the port is reenabled as a HOST port, the next incoming call is connected to the process attached to the port, thus creating a breach of system security.

To prevent this breach of security, ACC recommends that all processes attached to a port to be reconfigured be deleted by issuing a STOP PROCESS/ID=xx command. This security hole is impossible in VMS versions 4.3 and earlier because sharing of TP devices is not allowed.

# APPENDIX E

X.3 PAD Parameters

## E.1 PAD Parameters

The PAD does many functions and has operational characteristics. Some functions allow the local terminal and the remote PAD to configure the PAD so that its operation is adapted to the local terminal's characteristics and to the application. The operation of the PAD depends on the values of the setting of internal variables called PAD parameters. This set of parameters exists for each terminal independently. The value of each PAD parameter sets the operation of its related function.

The functions of the 18 PAD parameters follow. All PAD parameters except those listed below can be modified:

11 - bit rate
16 - character delete
17 - line delete
18 - line display

The acceptable values for all parameters are in table E-1. The PAD parameters that follow are in numerical order.

| Parameter | Description |
|---|---|
| 1 | PAD recall using a character<br><br>This function allows the start-stop mode DTE to start an escape from the Data Transfer state of a virtual call in order to send PAD command signals. |
| 2 | Echo<br><br>This function causes all characters received from the start-stop mode DTE to be sent back to the start-stop mode DTE and be interpreted by the PAD. |
| 3 | Selection of data forwarding signals<br><br>This function allows the selection of defined sets of character(s) received from the start-stop mode DTE to be recognized by the PAD as an indication to complete the assembly and forward a packet as defined in *CCITT Recommendation X.25.* |

| Parameter | Description |
|---|---|
| 4 | Selection of idle timer delay<br><br>This function allows the selection of a timeout interval between successive characters received from the start-stop mode DTE. When the selected interval is exceeded, it causes the PAD to terminate the assembly of a packet and to forward it as defined in *CCITT Recommendation X.25.* |
| 5 | Ancillary device control<br><br>This function allows for flow control between the PAD and the start-stop mode DTE. The PAD indicates whether it is ready to accept characters from the start-stop mode DTE by transmitting special characters. These characters are those which are used in International Alphabet No.5 (IA5) to switch an ancillary transmitting device on and off. |
| 6 | Control of PAD service signals<br><br>This function gives the start-stop mode DTE the ability to decide whether PAD service signals are transmitted. |
| 7 | Selection of operation of the PAD on receipt of the break signal<br><br>This function allows selection of the PAD operation after receiving a break signal from the start-stop mode DTE. |
| 8 | Discard Output<br><br>This function allows a PAD to discard the contents of the input packets rather than disassembling and transmitting them to the start-stop mode DTE. |
| 9 | Padding after carriage return<br><br>This function provides automatic insertion (by the PAD) of padding characters in the character string, which is transmitted to the start-stop mode DTE after the occurrence of a carriage return character. This allows the printing mechanism of the start-stop mode DTE to do the carriage return function correctly. |
| 10 | Line folding<br><br>This function allows the PAD to automatically insert the appropriate formatters in the character string transmitted to the start-stop mode DTE. The preset highest number of graphic characters per line can be set. |

| Parameter | Description |
|---|---|
| 11 | DTE Bit Rate<br><br>This function is a read-only parameter and indicates the DTE's operating bit rate. |
| 12 | Flow control of the PAD by the start-stop mode DTE<br><br>This function allows for flow control between the start-stop mode DTE and the PAD. The start-stop mode DTE indicates whether it is ready to accept characters from the PAD by transmitting special characters. In IA5, these characters are are used to switch an ancillary transmitting device on and off. |
| 13 | Linefeed insertion after carriage return<br><br>This function supports automatic insertion by the PAD of a linefeed character after any carriage return character transmitted or echoed to the start-stop DTE. This function works only in the Data Transfer state. |
| 14 | Padding after linefeed<br><br>This function provides for the automatic insertion by the PAD of padding characters in the character stream transmitted to the start-stop DTE after the occurrence of a linefeed character. This allows for the printing mechanism of the start-stop DTE to do the linefeed operation correctly. This function works only in the Data Transfer state. |
| 15 | Editing<br><br>This function provides for character delete, line delete and line display editing capabilities in the PAD Command Signal state and Data Transfer state for the start-stop mode DTE. During the PAD Command state, the editing function is always available. |
| 16 | Character Delete<br><br>This parameter describes the keystroke value to be used to generate a character delete. |
| 17 | Line Delete<br><br>This parameter describes the keystroke value to be used to generate a line delete. |

| Parameter | Description |
|---|---|
| 18 | Line Display<br>This parameter describes the keystroke value to be used to generate a line display. |

Table E-1. PAD Parameters

| Parameter ID | Description | Supported Values | Effect |
|---|---|---|---|
| 1 | PAD Command state recall | 0 | return to Command state not possible |
| | | 1–32 & 127 | return to Command state by entering this character |
| 2 | echo | 0 | no echo |
| | | 1 | PAD echo |
| 3 | data forwarding character set | 0 | no data forwarding |
| | | 2 | <CR> |
| | | 126 | all chars < 32 & 127 |
| | | 127 | all chars |
| 4 | data forwarding timeout is 20th of a second | 0 | no time out |
| | | 1–255 | value is rounded up to next multiple of 20 |
| 5 | PAD to DTE flow control | 0 | no use of XON XOFF |
| | | 1 | PAD can use XON XOFF |
| 6 | control of PAD service messages | 0 | no PAD service msgs are sent to DTE |
| | | 1 | PAD service msgs other than CMD prompt are sent |
| | | 5 | PAD service msgs and CMD prompt are sent |

Table E-1. PAD Parameters (continued)

| Parameter ID | Description | Supported Values | Effect |
|---|---|---|---|
| 7 | PAD action on receipt of break from DTE | 0 | do nothing |
| | | 1 | interrupt sent |
| | | 2 | reset sent |
| | | 8 | escape from Transfer state |
| | | 21 | discard output interrupt sent and indicate break |
| 8 | discard output | 0 | normal data deliver to DTE |
| | | 1 | discard output destined for DTE |
| 9 | padding after <CR> | 0 | no padding |
| | | 1–7 | number of nulls inserted after <CR> sent to DTE |
| 10 | line folding | 0 | no line folding |
| | | 1–255 | number of graphic chars after which a <CR> or <LF> is inserted in DTE data stream |
| 11 | bit rate | 14 | 9600 bits/sec |
| 12 | DTE to PAD flow control | 0 | XON XOFF is permitted in data stream |
| | | 1 | XON XOFF not transmitted in data stream. NOTE: in either case DTE has local XON XOFF control |

Table E-1. PAD Parameters (continued)

| Parameter ID | Description | Supported Values | Effect |
|---|---|---|---|
| 13 | linefeed insertion after <CR>. (Transfer state only) | 0 | no <LF> insertion |
| | | 1 | insert <LF> after each <CR> in data stream to DTE |
| | | 2 | insert <LF> after each <CR> in data stream from DTE |
| | | 3 | 1 and 2 |
| | | 4 | insert <LF> after each <CR> echoed to the DTE |
| | | 5 | 1 and 4 |
| | | 6 | 2 and 4 |
| | | 7 | 1 and 2 and 4 |
| 14 | linefeed padding. (Transfer state only) | 0 | no padding after <LF> |
| | | 1–7 | number of nulls inserted after <LF> transmitted to DTE |
| 15 | editing | 0 | no use of editing while in Transfer state |
| | | 1 | editing permitted while in Transfer state (editing consists of recognition of parameters 16, 17, & 18 while in Transfer state) |
| 16 | character delete | 127 | delete last char in data buffer by entering <DEL> |
| 17 | line delete | 24 | deletes current line by entering <CAN> |
| 18 | line display | 18 | displays current line by entering <DC2> |

# APPENDIX F

## PAD Clear Indication Service Signals

Table F-1. Clear Indication PAD Service Signals

| Clear Indication Meaning | Displayed Mnemonics | Explanation |
|---|---|---|
| Number busy | OCC | The called DTE is detected by the DCE as engaged in other calls and so is not able to accept the incoming call. |
| Network congestion | NC | 1) Temporary network congestion<br>2) Temporary fault condition within the network |
| Invalid facility request | INV | Invalid facility requested by the calling DTE |
| Access barred | NA | The calling DTE is not allowed to connect to the called DTE. (e.g., because of an incompatible closed user group) |
| Local procedure error | ERR | A procedure error caused by the DTE is detected by the PAD (e.g., because of incorrect format). |
| Remote procedure error | RPE | A procedure error caused by the DTE is detected by the DCE at the remote DTE/DCE interface. |
| Not obtainable | NP | The called DTE address is out of the number plan or is not assigned to any DTE. |
| Out of order | DER | The called number is out of order. |
| PAD clearing | PAD | The call has been cleared by the local PAD as an answer to an invitation to clear from the remote DTE. |
| DTE clearing | DTE | The remote DTE has cleared the call. |

# APPENDIX G

Electrical Interface Pinouts and Conversion Chart
Table G-1. RS-232C Interface

| PIN | MNEM | SIGNAL NAME | EIA | CCITT | DIRECTION |
|---|---|---|---|---|---|
| 1 | FG | frame ground | AA | 101 | N/A |
| 2 | TX | transmit data | BA | 103 | to modem |
| 3 | RX | receive data | BB | 104 | from modem |
| 4 | RTS | request to send | CA | 105 | to modem |
| 5 | CTS | clear to send | CB | 106 | from modem |
| 6 | DSR | data set ready | CC | 107 | from modem |
| 7 | SG | signal ground | AB | 108 | N/A |
| 8 | CO | carrier on | CF | 109 | from modem |
| 9 | | reserved for data set testing | | | |
| 10 | | reserved for data set testing | | | |
| 11 | | unassigned | | | |
| 12 | (S)CO | secondary carrier on | SCF | 122 | from modem |
| 13 | (S)CTS | secondary clear to send | SCB | 121 | from modem |
| 14 | (S)TX | secondary transmit data | SBA | 118 | to modem |
| 15 | SCT | serial clock transmit | DB | 114 | from modem |
| 16 | (S)RX | secondary receive data | SBB | 119 | from modem |
| 17 | SCR | serial clock receive | DD | 115 | from modem |
| 18 | | unassigned | | | |
| 19 | (S)RTS | secondary request to send | SCA | 120 | to modem |
| 20 | DTR | data terminal ready | CD | 108.2 | to modem |
| 21 | SQ | signal quality | CG | 110 | from modem |
| 22 | RI | ring indicator | CE | 125 | from modem |
| 23 | RS | rate select | CH/CI | 111/112 | to/from modem |
| 24 | SCTE | serial clock transmit external | DA | 113 | to modem |
| 25 | | unassigned | | | |

Table G-2. RS-449 (RS-422) Interface

| PIN | EIA | SIGNAL NAME | CCITT | DIRECTION |
|---|---|---|---|---|
| 1 | shield | shield | | N/A |
| 2 | SI | signal rate indicator | 112 | from modem |
| 3 | | spare | | |
| 4/22 | SD | send data | 103 | to modem |
| 5/23 | ST | send timing | 114 | from modem |
| 6/24 | RD | receive data | 104 | from modem |
| 7/25 | RS | request to send | 105 | to modem |
| 8/26 | RT | receive timing | 115 | from modem |
| 9/27 | CS | clear to send | 106 | from modem |
| 10 | LL | local loopback | 141 | to modem |
| 11/29 | DM | data mode | 107 | from modem |
| 12/30 | TR | terminal ready | 108.2 | to modem |
| 13/31 | RR | receiver ready | 109 | from modem |
| 14 | RL | remote loopback | 140 | to modem |
| 15 | IC | incoming call | 125 | from modem |
| 16 | SF | select frequency | 126 | to modem |
| " | SR | signaling rate selector | 111 | " " |
| 17/35 | TT | terminal timing | 113 | to modem |
| 18 | TM | test mode | 142 | from modem |
| 19 | SG | signal ground | 102 | N/A |
| 20 | RC | receive common | 102.6 | N/A |
| 21 | | spare | | |
| 28 | IS | terminal in service | | to modem |
| 32 | SS | select standby | 116 | to modem |
| 33 | SQ | signal quality | 110 | from modem |
| 34 | NS | new signal | | to modem |
| 36 | SB | standby indicator | 117 | from modem |
| 37 | SC | send common | 102a | N/A |

Table G-3. Interface Conversion Chart

| RS-232C | | RS-449/422 | |
|---|---|---|---|
| SIGNAL | PIN | SIGNAL | PIN |
| TX | 2/7 | SD | 4/22 |
| RX | 3/7 | RD | 6/24 |
| RTS | 4/7 | RS | 7/25 |
| CTS | 5/7 | CS | 9/27 |
| CO | 8/7 | RR | 13/31 |
| DSR | 6/7 | DM | 11/29 |
| DTR | 20/7 | TR | 12/30 |
| RI | 22/7 | IC | 15/20 |
| SCT | 15/7 | ST | 5/23 |
| SCR | 17/1 | RT | 8/26 |
| SCTE | 24/7 | TT | 17/35 |
| SG | 7 | SG | 19 |
| FG | 1 | SHIELD | 1 |