AIX 5L Version 5.1

Asynchronous Communications Guide

AIX 5L Version 5.1

Asynchronous Communications Guide

Tenth Edition (April 2001)

Before using the information in this book, read the general information in “Appendix F. Notices” on page 199.

This edition applies to AIX 5L Version 5.1 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9561, 11400 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About This Book

This book provides overview and planning information for asynchronous adapter subsystems. It describes the devices, software, programming concepts, and communications for 8-port, 16-port, 64-port, and 128-port adapters. Information on installing, configuring, and troubleshooting the adapters is also provided.

Who Should Use This Book

This book is intended for system programmers who want to evaluate and design asynchronous devices, program the tty subsystem interface, and configure asynchronous adapters for use with the system.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

- *8 Port Asynchronous ISA Adapter Installation Guide*
- *Adapters, Devices, and Cable Information*
- *AIX 5L Version 5.1 System Management Guide: Communications and Networks*
- *AIX 5L Version 5.1 System Management Guide: Operating System and Devices*

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

- IBM
- Micro Channel
- RS/6000

Intel is a trademark of the Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be the trademarks or service marks of others.

Chapter 1. Asynchronous Communication Planning

This section provides an overview of the available asynchronous adapters as well as an introduction on how to use each of the adapters. Both the hardware and software aspects of using serial ports are also covered.

Asynchronous Adapter Overview

The operating system allows many users to access system resources and applications. Each user must be connected through a terminal session. The connection can be local or remote through a serial port or network connection using Transmission Control Protocol/Internet Protocol (TCP/IP).

Each system unit has at least two standard serial ports available (some systems have three serial ports). These ports can support asynchronous communication and device attachment.

Two standard ports are sufficient for users requiring an additional ASCII terminal or a modem. Many users require additional asynchronous ports.

> **Note:** For the Itanium-based platform, each system has at least one serial port available, which is sufficient for users requiring an additional ASCII terminal or modem.

Asynchronous communications products offer the advantages of low cost, multi-user, medium- to high-performance terminal and device communications. Asynchronous ports allow attachment of asynchronous peripheral devices that meet EIA 232, EIA 422, or RS-423 standards such as:

- Asynchronous modems
- Bar code scanners
- Graphic and character printers
- Keyboard and display terminals
- Personal computers
- Plotters and printers
- Point-of-sale terminals
- Sensors and control devices
- Text scanners
- Time clocks

Evaluating Asynchronous Communications Options

Expanded asynchronous capability can be added to the system unit with direct-attached adapters using Micro Channel (MC), Industry Standard Architecture (ISA), or Peripheral Component Interconnect (PCI) buses; distributed subsystems; or by local area network (LAN) attached communications servers. Several factors will influence the type of asynchronous connectivity you choose. The following table summarizes these products.

Asynchronous Attachment	POWER-based	Itanium-based	Bus Type	Feature Code or Machine Type (Model)	Maximum Data Rate per Port (KBits/sec)	Salient Features
Standard serial port	X	X	System planar	n/a	Selectable based on baud rate generator clock speed of universal asynchronous receiver and transmitter (UART).	Standard feature
8-Port EIA 232	X		Micro Channel	2930	76.8	Pervasive standard
8-Port EIA 422A	X		Micro Channel	2940	76.8	Greater distance
8-port MIL-STD 188	X		Micro Channel	2950	Selectable based on baud rate generator clock speed of UART.	MIL-STD 188-114 for unbalanced voltage digital interface
8-port EIA 232	X		ISA	2931	115.2	Greater efficiency
8-port EIA 232	X		ISA	2932	115.2	Greater efficiency
8-port EIA 422	X	X	PCI	2943	230	Greater efficiency
16-Port EIA 232	X		Micro Channel	2955	76.8	Local connection focus
16-Port EIA 422A	X		Micro Channel	2957	76.8	Greater distance
232 RAN	X	X		8130	57.6	Remote capability
Enhanced 232 RAN	X	X		8137	230	Remote capability
16-Port RAN EIA 422	X	X		8138	230	Remote capability
128-Port Controller	X		Micro Channel	8128	230	Efficiency, higher device counts
	X		ISA	2933		
	X	X	PCI	2944		
Note 1: Rack Mount RAN FC is 8136.						
Note 2: Maximum Data Rate per Port is limited by line bandwidth (1.2 MBits/sec for standard RAN, or 2.4 MBits/sec for Enhanced RAN).						

The first feature in this table represents the standard-attached serial ports that are standard with every system unit. The next five features are the direct-attached adapters. The 128-port asynchronous subsystem includes the remote asynchronous nodes (RANs) that attach to it.

Note: MCA and ISA devices are not available on the Itanium-based platform-based systems, and NTA is not supported.

Standard-Attached Asynchronous Ports

Most system unit models have two integrated (standard) EIA 232 asynchronous serial ports. The Model M20/M2A features a single integrated asynchronous serial connector that can be converted to support two serial devices using an optional fan-out cable. EIA 232 asynchronous serial devices can be attached directly to the standard serial ports using standard serial cables with 25-pin D-shell connectors. Some multiprocessor systems have a third serial port used for communication to the remote service center.

Note: Itanium-based systems have one or two integrated serial ports. Initial workstation models have one port, while the initial server class models have two ports.

Direct-Attached Asynchronous Ports

Each of the direct-attached adapters requires a bus slot and can only be used in systems that support the required bus type. The Micro Channel 8- and 16-port adapters are relatively simple, meaning that very little processing of asynchronous traffic is offloaded from the system processor. The 128-port, ISA 8-port adapters, and PCI 8-port adapters are intelligent adapters that provide significant offload of the main system processor. The 8-port and 16-port adapters use fan-out cables to connect to devices and require no additional power supply.

EIA 232 is the most common communication standard, but the EIA 422A (used when a longer cable distance is needed) is also supported by each direct-attached adapter for the Micro Channel. The EIA 422A implementation does not include device status detection capability. A Micro Channel 8-port adapter complying with the military standard (MIL-STD) 188-114 is also available.

Note: The Itanium-based platform supports only the 8- and 128-port PCI adapters.

Node-Attached Asynchronous Ports

The 128-port adapter, available for the Micro Channel, ISA, or PCI bus, allows attachments of one to eight remote asynchronous nodes (RANs). Each RAN has 16 asynchronous ports for connection to devices and are separately powered units. Up to four RANs can be daisy-chain-connected from each of two connections on the 128-port adapter card. RANs can support 16 EIA 232 devices or 16 EIA 422 devices. The 128-port controller is an intelligent adapter that increases the number of asynchronous sessions possible at a given CPU usage level.

The following are additional characteristics of the 128-port feature:

- RANs may be located up to 300 meters from the system processor using 8-wire shielded cabling while maintaining full performance ratings.
- Distance may be extended to 1200 meters by reducing the data rate between the RANs and the system processor.
- RANs may be remotely located from the system processor using a synchronous EIA 232 and EIA 422 modems. Each four-RAN daisy chain is allowed only one modem pair at some point in the chain.
- System performance is enhanced by offloading tty character processing from the system processor.

Product Selection Considerations

This section will help you determine what asynchronous product you should choose for a particular situation.

Product Selection Aid

The following questions will help you choose which product you need to install.

Expandability

How many asynchronous ports are needed?

How may ports will be needed in the future?

Topology

Will devices be in other buildings or remote locations?

Does an Ethernet LAN exist?

Where will system/network administration be done?

Is there an HACMP cluster?

What type of cabling is required or already there?

Is connection to multiple hosts a requirement?

Performance

Is your application CPU-intensive?

What types of devices will be attached?

What is the relative asynchronous bandwidth demand for the aggregate sum of the devices?

Relative Device Bandwidth Demand		
Low Demand	**Moderate Demand**	**High Demand**
ASCII terminals, Point-of-sale terminals, Asynchronous modems	Printers, Low-speed FAX/modems, Bar code scanners	Serial X-terminals, High-speed FAX/modems, High-speed printers, File transfer applications

Micro Channel and ISA Bus Slots

How many slots are available for asynchronous adapters?

Device Interface Requirement

What asynchronous interface is required, for example, EIA 232, EIA 422A, EIA 423, MIL-STD 188?

Do the devices or applications require the full EIA 232 interface?

Is a remote parallel port required for printing?

Security

Is dedicated calling a requirement?

Is a device name to a physical device mapping required?

Is system assurance kernel (SAK) required?

The following table shows the detailed product characteristics.

Asynchronous Attachment Product Characteristics								
	Native Serial Ports	**8-port**			**16-port**	**128-port with RAN**		
		MC	**ISA**	**PCI**		**MC**	**ISA**	**PCI**
Number of asynchronous ports per adapter	n/a	8	8	8	16	128	128	128

Asynchronous Attachment Product Characteristics								
Maximum number of adapters	n/a	8	7	8	8	7	7	8
Maximum number of asynchronous ports	2 or 3	64	56	64	128	896	896	1024
Number of asynchronous ports per RAN	n/a	n/a	n/a	n/a	n/a	16	16	16
Number of parallel ports	n/a	0	0	0	0	0	0	0
Maximum number of RANs	n/a	n/a	n/a	n/a	n/a	56	56	64
Maximum speed (KBits/sec)	Selectable based on baud rate generator clock speed of UART.	76.8	115.2	230	76.8	230	230	230
Attachment method	standard	direct	direct	direct	direct	node	node	node
Asynchronous electrical interfaces supported	EIA 232	EIA 232 EIA 422A[4]MIL-STD[4]188-114[4]	EIA 232 EIA 422A	EIA 232 EIA 422A	EIA 232 EIA 422A	EIA 232 EIA 422	EIA 232 EIA 422	EIA 232 EIA 422
Standard connector	DB25M/ MODU	DB25M	DB25M	DB25M	DB25M	RJ-45[2]	RJ-45[2]	RJ-45
DB25 cable options	n/a	n/a	n/a	n/a	n/a	RJ-45-DB25	RJ-45-DB25	RJ-45-DB25
Rack mount option	n/a	n/a	n/a	n/a	n/a	yes	yes	yes
Power supply	n/a	n/a	n/a	n/a	n/a	external	external	external
Signals supported (EIA 232)	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS[3]- DTR -DCD -	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI

Notes:

1. Socket accepts 8p RJ-45, 6p RJ-11, or 4p RJ-11 plugs with a reduction in signals supported.
2. RTS is tied high (+12V) in the fanout connector box of the 16-port interface cable EIA 232 (FC 2996).
3. Micro Channel Only.

Adapter Applications

Note: The Itanium-based platform supports only the PCI 8- and 128-port adapters.

Each product offering is characterized by a representative scenario for its strengths. The following are recommendations for each:

8-Port Micro Channel

- A Micro Channel bus slot available for asynchronous I/O.
- Fewer than eight ports with little or no expansion.
- All local terminals located within 61 meters (200 feet) from the system.
- Remote terminals needed (support through OEM multiplexer/modem).
- Device bandwidth demand low to moderate (up to 76.8 Kbps).

8-Port ISA Bus EIA 232 or EIA 232/EIA 422

- ISA slot available.
- Fewer than eight ports required with little or no expansion.
- Requires all EIA 232, all EIA 422, or a mix of EIA 232 and EIA 422 ports.
- Offload character interrupt and terminal I/O processing from the main CPU.
- Asynchronous speeds to 115.2 Kbps.
- Maximum performance for high speed (28.8 Kbps) modems with data compression.

8-Port PCI Bus EIA 232/EIA 422

- PCI slot available.
- Fewer than eight ports required with little or no expansion.
- Requires all EIA 232, all EIA 422, or a mix of EIA 232 and EIA 422 ports.
- Offload character interrupt and terminal I/O processing from the main CPU.
- Asynchronous speeds to 230 Kbps.
- Maximum performance for high speed (33.6 Kbps) modems with data compression.

16-Port Micro Channel

- A Micro Channel bus slot available for asynchronous I/O.
- Eight ports now, fewer than 16 ports with little or no expansion.
- All local terminals located within 61 meters (200 feet) from the system.
- Remote terminals needed (support through OEM multiplexer/modem).
- Devices do not need all EIA 232 signals.
- Device bandwidth demand low to moderate (up to 38.4 Kbps for asynchronous devices).

128-Port Adapter (Micro Channel, ISA, and PCI)

- A Micro Channel, ISA, or PCI bus slot available for asynchronous I/O.
- Sixteen ports now with expansion of up to 128 ports without additional slots.
- Most distant terminal located about 300 meters (1000 feet) from the system at maximum data rate for MC and ISA Adapters, 90 meters (300 feet) from the system at maximum data rate for the PCI adapter.
- Terminals planned: nearby or on premises, distant on premises, and remote.
- Need high asynchronous throughput with low processor demand.
- Need terminal attached printer capability.
- Need to connect to remote premises through fiber-optic or synchronous modems.

Customer Scenarios

The following represents some typical customer scenarios with suggested asynchronous solutions:

Real Estate Office

- Simplicity and cost are high priority.
- Operating system and server.
- Six to ten devices tied into the server accessing the database.
- One slot is available for asynchronous communication.
- Devices are less than 61 meters (200 feet) from the server.

Solution: 8-port or 16-port asynchronous adapter.

Retail Point-of-Sale

- Cost per seat is high priority.
- Operating system and server.
- 20 or more ASCII terminals: for example, cash registers.
- One slot is available for asynchronous communication.
- Future expansion for additional terminals is planned.

Solution: 128-port asynchronous controller with two RANs. Future expansion with additional RANs.

Topology Considerations

The asynchronous family of adapters offers a wide variety of choices where distance topology is concerned.

The maximum cable lengths from the standard- and direct-attach adapters is generally the distance between the port and the asynchronous device, operating at the maximum specified data rate. The 128-port adapter is measured from the adapter card to the daisy-chained RAN attached to it. With the 128-port, unlimited distances can effectively be achieved by using the EIA 422 synchronous modems to attach the RANs to the adapter.

Proper cabling is extremely important and is unique to each environment.

Chapter 2. Serial Communication

This chapter provides information on asynchronous communication standards, hardware, terminology, and concepts used throughout this book.

To understand serial ports, consider:
- Synchronization
- Serial Communication Parameters
- The EIA 232D Standard

Serial ports are used to physically connect asynchronous devices to a computer. They are located on the back of the system unit, using the multiport adapter, such as, the 8-, 16-, 64- and 128-port asynchronous adapters.

To understand the functionality of a serial port, it is necessary to first examine the parallel communications. A standard parallel port uses eight pins, or wires, to simultaneously transmit the data bits, making up a single character. The following illustration shows the parallel transmission of the letter a.

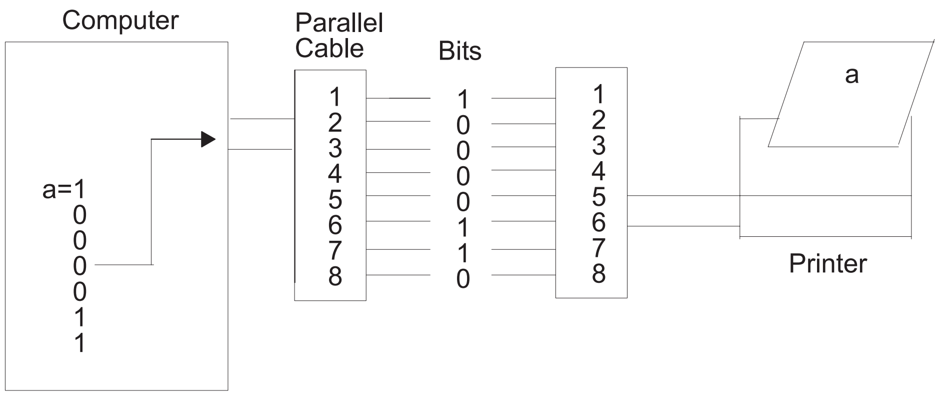


Figure 1. Parallel Communications Port

Serial ports require only a single pin, or wire, to send the same data character to the device. To accomplish this, the data is converted from a parallel form (sent by the computer), to a sequential form, where bits are organized one after the other in a series. The data is then transmitted to the device with the least significant bit (or zero-bit) sent first. Once received by the remote device, the data is converted back into parallel form. The following illustration shows the serial transmission of the letter a.

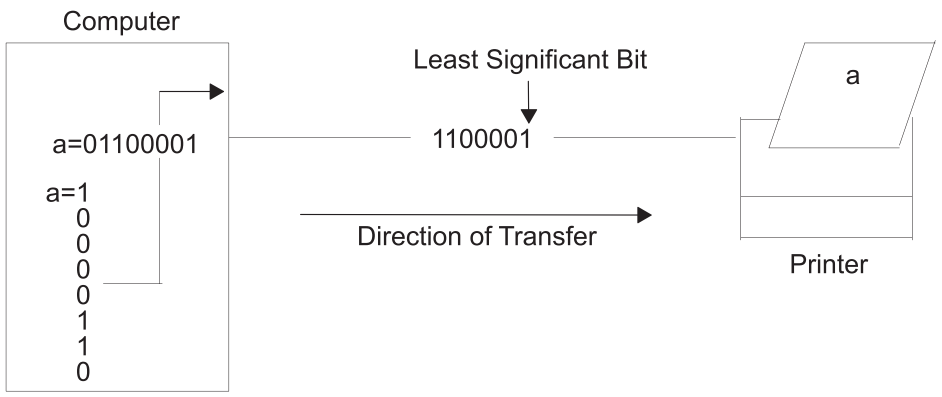


Figure 2. Serial Communications Port

Serial transmissions of a single character are simple and straight forward; however, complications arise when a large number of characters are transmitted in series as shown in the following illustration. The receiving system does not know where one character ends and the other begins. To solve this problem, both ends of the communication link must be synchronized or timed.

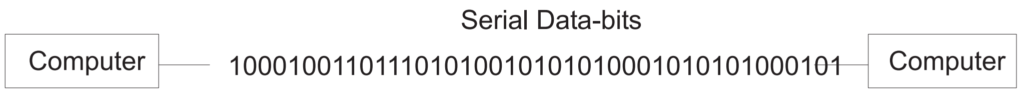


Figure 3. Serial Transmission

Synchronization

Synchronization is the process of timing the serial transmission to properly identify the data being sent. The two most common modes are synchronous and asynchronous.

Synchronous

The term synchronous is used to describe a continuous and consistent timed transfer of data blocks. These types of connections are used when large amounts of data must be transferred very quickly from one location to the other. The speed of the synchronous connection is attained by transferring data in large blocks instead of individual characters.

The data blocks are grouped and spaced in regular intervals and are preceded by special characters called `syn` or synchronous idle characters. See the following illustration.

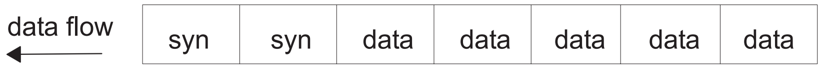


Figure 4. Synchronous Transmission

Once the `syn` characters are received by the remote device, they are decoded and used to synchronize the connection. Once the connection is correctly synchronized, data transmission may begin.

An analogy of this type of connection would be the transmission of a large text document. Before the document is transferred across the synchronous line, it is first broken into blocks of sentences or paragraphs. The blocks are then sent over the communication link to the remote site. With other transmission modes, the text is organized into long strings of letters (or characters) that make up the words within the sentences and paragraphs. These characters are sent over the communication link one at a time and reassembled at the remote location.

The timing needed for synchronous connections is obtained from the devices located on the communication link. All devices on the synchronous link must be set to the same clocking.

The following is a list of characteristics specific to synchronous communication:
- There are no gaps between characters being transmitted.
- Timing is supplied by modems or other devices at each end of the connection.
- Special `syn` characters precede the data being transmitted.
- `Syn` characters are used between blocks of data for timing purposes.

Asynchronous

The term asynchronous is used to describe the process where transmitted data is encoded with start and stop bits, specifying the beginning and end of each character as shown in the following figure.

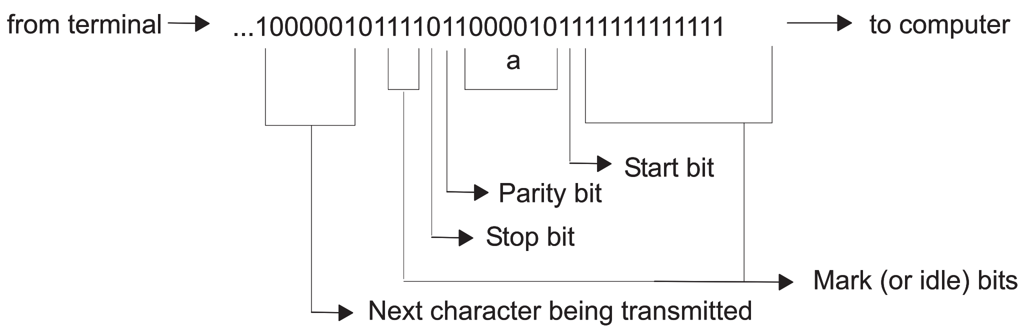
Start-bit									Stop-bit
0	1	2	3	4	5	6	7	8	1

Figure 5. Asynchronous Transmission

These additional bits provide the timing or synchronization for the connection by indicating when a complete character has been sent or received; thus, timing for each character begins with the start bit and ends with the stop bit.

When gaps appear between character transmissions, the asynchronous line is said to be in a mark state. A mark is a binary 1 (or negative voltage) that is sent during periods of inactivity on the line as shown in the following figure.

Figure 6. Mark (idle) Bits in the Data Stream



When the mark state is interrupted by a positive voltage (a binary 0), the receiving system knows that data characters are going to follow. It is for this reason that the start bit, which precedes the data character, is always a space bit (binary 0) and that the stop bit, which signals the end of a character, is always a mark bit (binary 1).

The following is a list of characteristics specific to asynchronous communication:

- Each character is preceded by a start bit and followed by one or more stop bits.
- Gaps or spaces between characters may exist.

Serial Communication Parameters

The following describes the parameters used during serial communication.

Bits-Per-Character

The number of bits-per-character (bpc) indicates the number of bits used to represent a single data character during serial communication. This number does not reflect the total amount of parity, stop, or start bits included with the character. Two possible settings for bpc are 7 and 8.

When using the seven bits-per-character setting, it is possible to only send the first 128 characters (0-127) of the Standard ASCII character set. Each of these characters is represented by seven data bits. The eight bits-per-character setting must be used to send the ASCII Extended character set (128-255). Each of these characters may only be represented using eight data bits. Refer to the following table for a numeric representation.

Table 1. Binary Numbers

Decimal Value	Binary Number
0	0
1	1
3	11
7	111
15	1111
31	11111
63	111111
127	1111111
255	11111111

Bits-Per-Second (bps)

Bits-per-second (bps) is the number of data bits (binary 1s and 0s) that are transmitted per second over the communication line.

Baud Rate

The baud rate is the number of times per second a serial communication signal changes states; a state being either a voltage level, a frequency, or a frequency phase angle. If the signal changes once for each data bit, then one bps is equal to one baud. For example, a 300 baud modem changes its states 300 times a second.

Parity

The parity bit, unlike the start and stop bits, is an optional parameter, used in serial communications to determine if the data character being transmitted is correctly received by the remote device.

Start-bit									Stop-bit
0	1	2	3	4	5	6	7	8 or parity	1

Figure 7. Parity

The parity bit can have one of the following five specifications:

none	Specifies that the local system must not create a parity bit for data characters being transmitted. It also indicates that the local system does not check for a parity bit in data received from a remote host.
even	Specifies that the total number of binary 1s, in a single character, adds up to an even number. If they do not, the parity bit must be a 1 to ensure that the total number of binary 1s is even. For example, if the letter a (binary 1100001) is transmitted under even parity, the sending system adds the number of binary 1s, which in this case is three, and makes the parity bit a 1 to maintain an even number of binary 1s. If the letter A (binary 1000001) is transmitted under the same circumstances, the parity bit would be a 0, thus keeping the total number of binary 1s an even number.
odd	Operates under the same guidelines as even parity except that the total number of binary 1s must be an odd number.
space	Specifies that the parity bit will always be a binary zero. When used for error detection, space will indicate a problem only if the parity bit is not a zero. Another term used for space parity is bit trimming, which is derived from its use as a filler for seven-bit data being transmitted to a device which can only accept eight bit data. Such devices see the space parity bit as an additional data bit for the transmitted character.
mark	Operates under the same guidelines as space parity except that the parity bit is always a binary 1. The mark parity bit acts only as a filler.

Most modern serial communications devices (modems, terminals, etc.) use a 10-bit transmission character:

```
     1  bit  for the start bit
7 or 8  bits for the data character
1 or 0  bits for the parity setting
2 or 1  bits for the stop bits
----------------------------------
    10  bits for total character transmission.
```

If the number of data bits is set to eight:

```
8 bpc + 1 start +1 stop + 0 parity = 10 bits
```

there are no bit positions left for parity; therefore, parity must be set to none.

If the number of data bits is set to seven, parity can be even, odd, or none.

```
7 bpc + 1 start + 1 stop + 1 parity = 10 bits
```

Another alternative is having parity set to none and stop bits set to two (this is rarely used).

```
7 bpc + 1 start + 2 stop + 0 parity = 10 bits.
```

If the parity bit is used for error detection on a communication line, both the sending and receiving systems must be configured identically to avoid possible problems during transmissions.

Start, Stop, and Mark Bits

The start and stop bits are used in asynchronous communication as a means of timing or synchronizing the data characters being transmitted. Without the use of these bits, the sending and receiving systems will not know where one character ends and another begins.

Another bit used to separate data characters during transmission is the mark (or idle)RS bit. This bit, a binary 1, is transmitted when the communication line is idle and no characters are being sent or received. When a start bit (binary 0) is received by the system, it knows that character data bits will follow until a stop bit (binary 1) is received. See the following figure.

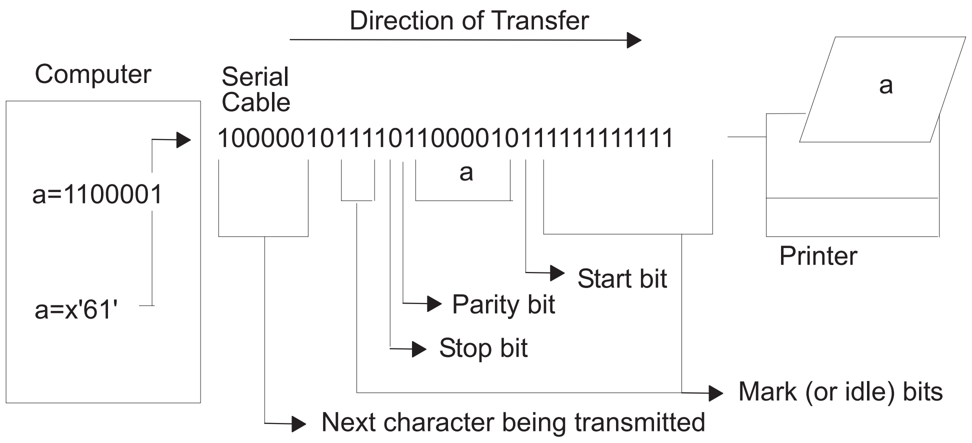


Figure 8. Start, Stop, and Mark Bits

The EIA 232D Standard

The EIA 232D standard was developed in 1969 to specify the connections between a computer and a modem. The term itself is an acronym which can be read as follows:

```
Electronics Industry Association (EIA) accepted standard, ID number 232 revision D.
```

EIA 232D specifies the characteristics of the physical and electrical connections between two devices. Names and abbreviations are assigned to each pin or wire necessary for serial communications, for example:

Signal	**Symbol**	**Pin**
Transmit Data	TxD	2
Receive Data	RxD	3
Request to Send	RTS	4
Clear to Send	CTS	5
Data Set Ready	DSR	6
Signal Ground	SG	7
Carrier Detected	CD	8
Data Terminal Ready	DTR	20
Ring Indicator	RI	22

In EIA 232D, devices using pin 2 (TxD) for output (for example, computers and terminals) are given the name data terminal equipment (DTE). Devices using pin 2 (TxD) for input (for example, modems) are given the name data communication equipment (DCE).

EIA 232D also specifies the connectors. A DTE device normally has male connectors while DCE devices have female connectors. This standard is not always adhered to by manufacturers; therefore users should always review the device documentation before cable connection.

Communication Methods

Simplex, or one-way communication, is the simplest connection form between two devices. This mode of communication allows data to be transmitted in only one direction and requires only two lines be connected, for example, TxD (or RxD) and SG.

There are two forms of two-way communications: half-duplex and full-duplex. A connection in half-duplex mode allows data to be transmitted in two directions but not simultaneously. An analogy of half-duplex would be the use of a CB-radio where two-way communication is possible but only one person can speak at a time.

In full-duplex, or duplex mode, data communication can take place in two directions simultaneously. An analogy for full-duplex is a telephone conversation when both persons are talking at the same time.

Flow Control

Serial devices, such as printers and modems, do not process data as quickly or efficiently as the computers they are connected to. Some type of data flow control is needed by the serial device to limit the amount of data transmitted by the system.

The term *flow control* is used to describe the method in which a serial device controls the amount of data being transmitted to itself. The three types of flow control discussed in this section are:

- RTS/CTS (hardware flow control)
- DTR/DSR (hardware flow control)
- XON/XOFF (software flow control).

RTS/CTS

Ready to send/clear to send (RTS/CTS) is sometimes called pacing or hardware handshaking instead of flow control. The term hardware handshaking comes from the use of cabling and voltages as a method of data transmission control. Unlike XON/XOFF, which sends control characters in the data stream, RTS/CTS uses positive and negative voltages along dedicated pins or wires in the device cabling.

A positive voltage means data transmission is allowed while a negative voltage signifies that data transmission should be suspended.

DTR/DSR

Data terminal ready (DTR), another form of hardware flow control, is normally generated by the devices, such as printers to indicate that they are ready to communicate with the system. This signal is used in conjunction with data set ready (DSR) generated by the system to control data flow.

A positive voltage means data transmission is allowed while a negative voltage signifies that data transmission should be suspended.

XON/XOFF

Transmitter on/transmitter off (XON/XOFF) flow controls involves the sending of data transmission control characters along the data stream (TxD and RxD). For this reason it is referred to as software flow control.

When data is sent to a modem, it is placed in a buffer. Just before that buffer reaches its maximum capacity, the modem will send an XOFF character to the system and the system will stop transmitting the data. When the modem's buffer is almost empty and ready for more data, it will send an XON character back to the system causing more data to be sent.

Configuring a Port for RTS/CTS Hardware Handshaking

Modems attached to the server operating at a speed of 9600 or above are recommended to use RTS/CTS hardware handshaking instead of XON/XOFF flow control. This will avoid buffer overrun in a system with limited resources. RTS is not a default value on any tty port and must be set by the system administrator accordingly.

To enable RTS/CTS for a port, use the following steps:

1. Use the **smit tty** fast path.
2. Select **Change / Show Characteristics of a TTY**.
3. Select the tty on which RTS/CTS is to be enabled.
4. Set the FLOW CONTROL to be used field to **rts**.
5. Select **Do**.
6. Exit SMIT.

Chapter 3. Asynchronous Devices and Software

This section describes asynchronous devices and related necessary software. Asynchronous devices include terminals, printers, plotters, or any device that conforms to the EIA 232C standard.

Topics discussed are:
- IBM 3151 Asynchronous Terminals
- Dynamic Screen Utility
- Transparent Printing
- Setting Terminal Options with stty-cxma

IBM 3151 Asynchronous Terminals

The following provides general information concerning the use of IBM 3151 asynchronous terminals with this operating system's tty subsystem. The following information is specific to the IBM 3151 terminal:
- Overall Characteristics
- Model Identification
- Connectivity Options
- Attributes for the IBM 3151 ASCII Display Station
  - IBM 3151 ASCII Display Station Models 110, 310, 410
  - IBM 3151 ASCII Display Station Models 510, 610
- Problem Determination
  - Problems with the Keyboard
  - Problems with the Display
  - Miscellaneous Problems
  - Common Questions

Overall Characteristics

The IBM 3151 has 5 models: 110 (withdrawn by IBM), 310, 410, 510, and 610. The models can be identified by the part number on the rear of the IBM 3151 terminal. Refer to ″Model Identification″ for further information.

All IBM 3151 ASCII displays support the following capabilities:
- EIA 232 interface.
- For EIA 422, two options exist:
  - Models 310 or 410, upgraded using one of the following cartridges:
    - IBM/DEC
    - Expansion
    - Connectivity.
  - Model 510 or 610 support EIA 422.
- 24x80, 25x80, 24x132, and 25x132 screen sizes. Other screen sizes such as 28x80 and 28x1302 require the expansion cartridge.
- Connectivity to the system unit.
  - The 3151 Models 510 and 610 can attach to this operating system, but are intended for PC attachment in PC mode.
  - Models 310 and 410 are best suited for this operating system.

For a detailed description of the 3151 display and cartridges, refer to the following manuals:

- *IBM 3151 ASCII Display Station*
- *IBM 3151 ASCII Display Station Models 110, 310, 410 Guide to Operations*
- *IBM 3151 ASCII Display Station Models 510, 610 Guide to Operations*
- *User's Guide for the Cartridge for Expansion*
- *User's Guide for the Cartridge to Emulate IBM and DEC Terminals*
- *User's Guide for the Cartridge for Connectivity*
- *User's Guide for the Cartridge for WSYE 50/50+*

Model Identification

- Model 310/410

Display	Green/Amber
Interface	EIA 232
Part Numbers	81x4501, 09F3484/81x4511, 09F3485
Keyboard	102-key ASCII
Part Number	1392595
Machine Modes	3151, 3101, 10 ASCII Emulations

- Model 510/610

Display	Green/Amber
Interface	EIA 232/EIA 422
Part Numbers	38F5101/38F5104
Keyboard	101-key PS/2
Part Number	1395162
Machine Modes	3151 PC, 3151 PC II and 925 PC

Connectivity Options

The 3151 display can connect directly, or through a modem, to the operating system. The connection to the operating system can be made to one of the native serial ports or to an asynchronous adapter. Additionally, a printer can be connected to the 3151 display and is supported by the operating system as terminal-attached printing.

The 3151 display has two ports on the back: the main port and the auxiliary port. In the scenario discussed previously, the operating system is connected to the main port, and the printer is connected to the auxiliary port.

The main port is DTE (DB-25) and does not require an interposer. The printer port already has an interposer (DCE); therefore, when connecting a serial printer to the 3151 display, no interposer is necessary.

Attributes for the IBM 3151 ASCII Display Station

Following are the recommended setup values for the 3151 ASCII Display Station, Models 110, 310, 410, 510, and 610 when used with this operating system.

For a detailed description of the 3151 setup values, refer to ″Setup Procedures,″ in the *IBM 3151 ASCII Display Station Models 110, 310, 410 Guide to Operations* or to *IBM 3151 ASCII Display Station Models 510, 610 Guide to Operations*.

IBM 3151 ASCII Display Station Models 110, 310, 410

- General Setup Menu

Machine Mode	IBM 3151
Screen	Normal
Row and Column	24x80
Scroll	Jump
Auto LF	Off
CRT Saver	On or Off
Line Wrap	On
Forced Insert	Line
Tab	Field

- Communication Setup Menu

Operating Mode	Echo
Line Speed (bps)	9600
Word Length (bits)	8
Parity	NONE
Stop Bit	1
Turnaround Character	CR
Line Control	IPRTS
Break Signal	500
Send Null Suppress	On

- Keyboard/Printer Setup Menu

Enter	Return
Return	New Line
New Line	CR
Send	Page
Insert Character Space	Space
Line Speed (bps)	9600
Word Length (bits)	8
Parity	None
Stop Bit	1
Characters	National

IBM 3151 ASCII Display Station Models 510, 610

- General Setup Menu

Machine Mode	IBM 3151 PC
Screen	Normal

Row and Column	24x80
Scroll	Jump
Auto LF	Off
CRT Saver	On or Off
Line Wrap	On
Message Type	Status Line
Forced Insert	Line
Tab	Field
Term. ID	(varies)
Alarm Volume Level	7
Cursor	Steady-Block

- Communication Setup Menu

Operating Mode	Echo
Line Speed (bps)	9600
Word Length (bits)	8
Parity	None
Stop Bit	1
Turnaround Character	CR
Interface	RS-232C
Line Control	IPRTS
Break Signal (ms)	500
Send Null Suppress	On
Pacing	XON/XOFF

- Keyboard/Printer Setup Menu

Generated Code Set	ASCII
Line Speed (bps)	9600
Word Length (bits)	8
Parity	None
Stop Bit	1
Characters	National
DTR Pacing	Off

Problem Determination

Problems with the Keyboard

Function keys do not work:

Possible Causes:

- The **Turnaround Character** value specified in the Communication SETUP menu is invalid.
- You are using an IBM 3151 Model 510 or 610, and the host application you are using requires redefinable function keys.

- An application redefines the function keys on 3151 Model 310/410 but does not reset the function keys back to default values upon exiting.

Procedures for Recovery:

- Specify the correct **Turnaround Character** value.
  - The IBM 3151's default value for the **Turnaround Character** is set to *ETX*. The operating system requires that the **Turnaround Character** is set to *CR*.
- Use a 3151 Model 310 or 410.
  - The 3151 Models 510 and 610 do *not* support redefinable function keys
- Reset the function keys.
  - The application must transmit the 3151 ″Set All Default Function Keys Command″ (*ESC SP t*).
  - The operator can reset function keys to the default value in **DEF F** (define function key) setup mode by pressing the Clear key.

The insert key does not work:

Procedures for Recovery:

Change the **Insert Character** attribute from *mode* to *space*.

Double Prompt occurs when the Enter or Return key is pressed:

Possible Causes:

The **New Line** attribute is set to CRLF.

Procedures for Recovery:

Change the New Line attribute:

- On the 3151 setup attribute on Keyboard setup menu (Models 310 or 410 only):
  Select **Default**.
- From your application send:
  Set Control 2 Command: `ESC ! 9 ( J`

The IBM 3151 terminal locks up:

Possible Causes:

- Operator pressed Ctrl-S (XOFF).
- To see other possible causes, turn on the **Status Line** either from the keyboard or from setup. This provides additional information. The following items may be displayed on the status line:

`HOST BUSY`	The system told the 3151 terminal to not send data.
`KEYS LOCKED`	The application sent a **Keyboard Lock** command.
`HOLD SCREEN`	Operator pressed Hold or Pause key.

Procedures for Recovery:

- Problem must be corrected at the host system.
- The application must send the **Keyboard Unlock** command.
- Press Hold or Pause key to restore screen.
- Press Ctrl-Q (XON) to remove a possible hold (XOFF) condition.

The *, /, -, Alt, Ctrl, Page Up, Page Down keys are not working:

Possible Causes:

When a Model 510 or 610 is in ASCII mode, the keyboard uses a 310/410 ASCII layout.

Procedures for Recovery:

Refer to *3151 Guide to Operations* for correct keyboard layout.

Cannot redefine function keys:

Possible Causes:

Using a 3151 Model 510 or 610.

Procedures for Recovery:

Redefining the function keys not supported for Models 510 or 610.

The 3151 backspace key is not working:

Possible Causes:

Using DEC emulation.

Procedures for Recovery:

With DEC emulation, use the DEL key to backspace.

Problems with the Display

Double characters are displayed:

Possible Causes:

The **Operating Mode** attribute is not set correctly.

Procedures for Recovery:

Change **Operating Mode** attribute from char to echo.

The IBM 3151 displays backwards question marks:

Possible Causes:
- Frame error.
- Mismatch of the emulation type between the system and the display.

Procedures for Recovery:
- Ensure that the **Parity**, **Word Length**, **Stop Bits**, and **Line Speed** attributes of the terminal match the tty line characteristics of this operating system.
- Check with your administrator to verify the emulation type.

The screen is wavy:

Possible Causes:

The refresh rate of the terminal is incorrect.

Procedures for Recovery:

Vary the refresh rate between between 50 and 60 hertz. Press Ctrl-Shift-Hold. The 3151 displays the keyboard test pattern. Press Ctrl-F10. The 3151 displays A or B (refresh rate). Press F1 to toggle between A or B. Power off and on the 3151.

Miscellaneous Problems

The IBM 3151 terminal does not communicate with the operating system:

Possible Causes:
- The **Line Control** attribute is not set to IPRTS.
- The cable is attached to the auxiliary port rather than the main port.
- The Transmit Data (TxD), Receive Data (RxD) pins do not match the operating system's communications port.
- The system does not have the communications port defined and enabled.

Procedures for Recovery:
- Turn on **Status Line**. If COMM NOT READY 2 is displayed on the status line, change the **Line Control** attribute to IPRTS.
- Move the cable to the main port. The main port is on the right-hand connector, looking from the back.
- Use correct cabling: Asynchronous cable EIA 232/V.24 cable D and Printer/Terminal Interposer EIA 232 cable E or equivalent.
- Use SMIT or the **penable** command to enable the port. Refer to ″Managing TTY Devices″ in *AIX 5L Version 5.1 System Management Guide: Communications and Networks* for further information.

Second session not working when using Connectivity Cartridge in Dual Session by Dual Port:

Possible Causes:

The directions of pins 2 (TxD) and 3 (RxD) may be reversed.

Procedures for Recovery:

Switch pins 2 and 3 on the communications cable attached to the auxiliary port.

The 3151 display is missing or losing characters:

Possible Causes:
- Pacing or data flow problem
- Incorrect **TERM** environment variable

Procedures for Recovery:
- To confirm that a pacing problem exists, reduce line speed to 2400 or 4800. If problems persist, activate pacing at the 3151 and the operating system.
- Check **TERM** environment variable and change it so that it matches the terminal.

The 3151 terminal is losing its setup values:

Possible Causes:
- Application could be changing setup values using 3151 commands: Set Control, Create Viewport.

- User changes setup options in setup menu, but does not save the changes.

Procedures for Recovery:

- Problem must be corrected at the system.
- Place cursor on the Save option and press the space bar to save new setup.

Cannot place the 3151 terminal in setup mode:

Possible Causes:

- Mismatch of keyboards with the video element, that is, using a 310/410/510/610 video element with a 84-key ASCII (3151 Model 110) keyboard
- DEC VT200/VT100/VT52 emulation is selected, and keyboard does not have DEC Keycap kit.

Procedures for Recovery:

- Impossible to enter setup. Exchange the 84-key ASCII keyboard with the proper keyboard.
- Press the F3 key, the location of the setup key when in DEC emulation.

Cannot print or not is printing correctly:

Possible Causes:

- Attached printer is a parallel type printer, not a serial printer.
- Incorrect cable being used with printer.

Procedures for Recovery:

- Printer must be a serial printer.
- Use correct cable.
- Printer is not configured with pacing.

Line-drawing characters do not display:

Possible Causes:

A character set mismatch between system and terminal.

Procedures for Recovery:

The 3151 offers the following character sets:

- Special Graphics
- PC 00437
- PC 00850
- ISO 8859/1.2

Determine which character set is defined on the 3151, and which character set the system is expecting. The character set used by the system and the terminal must match.

The 3151 has extra blank lines:

Procedures for Recovery:

- From the terminal, set Auto LF = OFF in general setup menu.
- From the operating system, send the Set Control 3 command (ESC ″ 9 = b).

The setup values of one 3151 does not match another 3151:

Possible Causes:

The 3151 offers many models, 8 cartridges with over 20 emulations. Compare the machine mode of each 3151. Possibilities include:

- The two 3151s may be of two different models: 110 versus 310/410 versus 510/610.
- One 3151 could have a cartridge, the other 3151 may not have a cartridge.
- Both 3151s have cartridges, but they may be different types of cartridges, such as IBM/DEC, Expansion, Connectivity, or WYSE 50/50+.
- Both 3151s have the same cartridge, but the Machine Mode is not set the same.

Common Questions About the IBM 3151 Terminal

How do you change the cursor style to Block, Underscore, or Blink?

- Model 310 or 410 with:

No cartridge	Press Ctrl-8 (numeric keypad).
Connectivity cartridge	Use the User setup menus.
Expansion cartridge	Use the User setup menus.
IBM/DEC cartridge	Press Ctrl-8 (numeric keypad).
WYSE cartridge	Press Ctrl-8 (numeric keypad).

- Model 510, 610
  - User setup menus or using the Set Cursor Type command. (ESC F pa)

How do you change the status line (None, Machine Status, Host Message)?

From the system:

1. Write Host Message
2. Display Host Message
3. Display Machine Status.

From the user, by keyboards or user setup.

- Model 310 or 410 with:
  - No cartridge - press Ctrl-9 (numeric keypad).
  - Connectivity cartridge - User setup menus.
  - Expansion cartridge - User setup menus.
  - IBM/DEC cartridge - press Ctrl-8 (numeric keypad).
- Model 510, 610 User setup menus.

Does the auxiliary port support an EIA 422 interface?

No, it only supports EIA 232.

What is the difference between EIA 232 and EIA 422?

- EIA 232 defines 9 pins. Two pins are used for Transmit and Receive Data (TxD, RxD). EIA 422 defines 5 pins. Four pins are used for Transmit and Receive Data (TD+, TD-, RD+, RD-).
- RS-232C can attach 50-75 feet (15.2-22.9 meters), EIA 232D can attach 200 feet (61 meters). EIA 422 can attach 4000 feet (1219 meters).

How do you change the function keys?

- Models 310 or 410
  - From the system: Issue the appropriate command sequence.
  - From the terminal: Use User setup in (DEF F) mode.
- Models 510 or 610 - not available.

How many function keys are available on the 3151?

- Model 310 and 410
  - 12 physical keys, shiftable to 36 (unshifted, Shift, Ctrl-Shift) redefinable from system or user setup (DEF F).
- Model 510 and 610
  - 12 physical keys, shiftable to 36 (unshifted, Shift, Ctrl-Shift) Not redefinable.

How do you reset/clear a function key to default values?

- From User setup: Press Ctrl-Clear key.
- From the application: Send 'Reset All Default Default Function Keys' (ESC SP t) command.

Can an input device be attached to the auxiliary port?

Yes, the auxiliary port is bidirectional (for Models 310, 410, 510 and 610).

Can a selected key (setup, break, ctrl...) be physically disabled?

No, this requires a keystopper under keycap.

Can a selected key be remapped to another key?

No.

Can you use a 3151 Model 310, 410, 510, 610 keyboard on a 3151 Model 110?

No, you cannot mix any combination of 84-, 101-, and 102-key keyboards. The keyboard must match video display.

How do you automatically dial a modem?

Use the connectivity cartridge, or assign an unused function key. Cartridge requires 6 keystrokes; unused function key requires 2 to 3 keystrokes.

How do you activate or deactivate the printer from the keyboard?

Press Ctrl-Trace key.

How do you display CAPS information?

Turn on the status line.

What screen sizes do the various 3151 terminals support?

Model 310, 410: 24x80, 25x80, 24x132 or 25x132.

Model 310, 410 Expansion Cartridge: Models 310, 410 base screen sizes, plus 28x80 or 28x132.

Model 510, 610: 24x80, 25x80, 24x132 or 25x132.

How do you change the screen size?

- From the terminal: Use the setup menu.
- From the application: use the Create Viewport Command:

24 x 80	ESC SP r ! ! SP 8 ″ P
25 x 80	ESC SP r ! ! SP 9 ″ P
28 x 80	ESC SP r ! ! SP < ″ P (Expansion Cartridge)
24 x 132	ESC SP r ! ! SP 8 $ D
25 x 132	ESC SP r ! ! SP 9 $ D
28 x 132	ESC SP r ! ! SP < $ D (Expansion Cartridge)

Are communication cables included?

No.

What is the command to turn the printer on/off from the system?
Begin/End Pass through mode (**DLE DC2** and **DLE DC4**)

Can you switch a 3151 between a 110 and a 220 volt?
No, each country has the appropriate video display with the proper power supply.

How can the 3151 display line-drawing characters?
1. Assign G0/G1 using Select Character Set command.
2. Select either G0 or G1 by:

   8-bit mode, turn on most significant bit.

   7-bit mode, use SO/SI ASCII control characters.

What character sets are available?
- Model 110/310/410: US ASCII and IBM Special Graphics.
- Model 510/610: US ASCII, IBM Special Graphics, PC Code Page 00437, PC Code Page 00850.

What is the gender of the main and auxiliary ports?
Both are 25-pin D-Shell female.

Which port is the main communication port?
Looking at the back of the 3151, the main port is on the right-hand side, closest to the edge. The auxiliary port is closest to the power cord.

How do you get a dual session?
Use the Cartridge for Connectivity. Select Dual Session by Dual Port in **SESSION ASSIGNMENT** setup menu.

Is there a keyboard extension cable available?
Changes between the keyboard and display elements are not offered and are not supported.

Is there support for an IBM 7695 Bar Code Reader placed between keyboard and display elements?
Only the Models 51/61 support the 7695 Bar Code Reader. Other bar code reader products are available that can be placed between the keyboard and display element; however, only the 7695 bar code reader is supported.

Does the 3151 offer a CRT Saver/Screen Saver?
Yes, CRT is selectable between No Save or 15 minutes. The CRT is selectable by the user in the General Setup menu, or by the system using the Set Control 2 command.

Is a tilt/swivel stand included?
Yes.

Does the 3151 have a data entry keyboard?
No.

Dynamic Screen Utility

The dynamic screen (dscreen) utility allows a single physical terminal to have several virtual terminal sessions or screens running at one time. This utility works with all 128-port and ISA/PCI 8-port adapters. It is mainly intended for use with terminals that have two or more pages of screen memory; for example, the IBM 3151 display with the Expansion Cartridge.

With such terminals, switching between virtual screens also switches between physical terminal screen pages allowing each virtual screen's image to be saved and restored. On terminals without multiple pages of screen memory, **dscreen** can still be used to switch among virtual screen sessions although the appearance of the screen will not be maintained.

For full support of **dscreen**, the terminal must be able to switch internal screen pages on command and must remember the cursor position for each page. While **dscreen** will work on both smart and dumb terminals, screen images are not saved during screen changes on dumb terminals.

dscreen Terminal Configuration Information File

The **dscreen** terminal configuration information file (or **dsinfo** file) is used to define a different set of keys to be used with **dscreen**. This might be done, for example, when the originally defined **dscreen** keys conflict with a software application in use on the system.

The **dsinfo** file terminal type assumes a single page of screen memory. Therefore, if a terminal supports additional pages of screen memory, the **dsinfo** file must be customized to use the appropriate sequence for page memory control. Consult the appropriate terminal reference guide for the specific control sequence.

The default **dsinfo** file is the **/usr/lbin/tty/dsinfo** file. Use the **-i** flag to specify a different **dsinfo** file. This remainder of this section will refer to the default **dsinfo** file. However, the same information applies to any customized **dsinfo** file you create.

For more information concerning the **dsinfo** file, refer to ″Dynamic Screen Assignment″.

Key Action Assignments

When **dscreen** is run, it starts a virtual screen. Some of the keys on the terminal keyboard are not passed through to the virtual screen; instead, **dscreen** intercepts these keys and performs certain actions when they are pressed. The actions include:

Select	Selects a specified screen.
Block	Blocks all input and output.
New	Starts a new screen session.
End	Ends **dscreen**.
Quit	Quits **dscreen**.
Previous	Switches to previous screen.
List	Lists the **dscreen** assigned keys and their actions.

The function of each key is dependent upon the terminal and the terminal description in the **/usr/lbin/tty/dsinfo** file.

Select Keys

When a new virtual screen is created, it is assigned a select key. Pressing the select key causes the following actions:

- A switch from the physical terminal to the video page associated with the particular virtual screen.
- Input and output is directed appropriately between the physical terminal and the virtual screen.

Once all of the select keys defined in the **dsinfo** file have virtual screens assigned to them, no more screens may be created. Individual screen sessions end when the original shell process exits. This frees the associated select key for use with another virtual screen. The **dscreen** utility is ended when there are no more active screens.

Block Keys

Block keys are used to stop output in a fashion similar to the Ctrl-S key when using IXON flow control. The purpose of these keys is to allow for transparently setting up terminal sessions on two computers using a terminal that has two serial ports.

New Keys

Pressing a new screen key creates a new logical screen and assigns it to one of the select keys. Each new screen requires:

- A select key as defined in the **dsinfo** file.
- A **dscreen** pseudo terminal device.
- Enough memory for the various structures used in screen tracking.
- A process to run the shell from.

If any of these are not available, the new screen operation will fail with a message indicating the reason for the failure.

End and Quit Keys

Pressing an end key will cause the following to occur:

- Send a **SIGHUP** signal to all the screen sessions
- Clean up
- Exit with a status of 0.

Pressing a quit key will perform the same actions but will exit with status of 1.

Previous Key

Pressing a previous key switches the terminal to the screen that was last displayed.

Notes:

1. Do not switch screens when the current screen is being written to; an escape sequence may be truncated and leave the terminal in an unknown state.
2. Some terminal displays may save the cursor position for individual screens but may not save other states such as insert mode, inverse video, etc. If this is the case, users should avoid these modes while switching screens.

List Key

Pressing a list key will display a list of keys and their actions on the terminal display. Only those keys recognized by **dscreen** will be shown. When a new screen is created using **dscreen**, the message `Press` *`key`* `for help` displays, where *key* is the name of the list key displayed on the terminal. Note that the message is displayed *only* if there is a list key defined.

Dynamic Screen Assignment

The terminal description entry in the **/usr/lbin/tty/dsinfo** file will have the same number of screen selection keys as the terminal has physical screen pages. If more screen selection keys are defined than the number of physical screen pages, **dscreen** will dynamically assign physical screen pages to virtual screens.

When a virtual screen is selected that does not have an associated page of screen memory, **dscreen** assigns the least recently used physical screen to the virtual screen. Depending on the specifications maintained in the **/usr/lbin/tty/dsinfo** description file, an indication that the physical screen is connected to a different virtual screen may be noticeable; for example, the screen is cleared.

dsinfo File

The **dsinfo** file is a database of terminal descriptions used by the **dscreen** multiple screen utility. The file contains the following information:

- **dscreen** key definitions and the functions they perform.
- Number of screen memory pages for the terminal.
- Terminal specific control sequences sent or received to control the above features.

The terminal type entries in the default **dsinfo** file resemble the following IBM 3151 ASCII terminal values.

> **Note:** \r may be used in place of ˆM.

```
# The Cartridge for Expansion (pn 64F9314) needed for this entry
ibm3151|3151|IBM 3151,
dsks=\E!aˆM|Shift-F1|,           # Selects first screen
dsks=\E!bˆM|Shift-F2|,           # Selects second screen
dsks=\E!cˆM|Shift-F3|,           # Selects third
screendsks=\E!dˆM|Shift-F4|,     # Selects fourth screen
dskc=\E!eˆM|Shift-F5|,           # Creates a new screen
dske=\E!fˆM|Shift-F6|\E pA\EH\EJ, # Go to screen 1 and end
dskl=\E!gˆM|Shift-F7|,           # Lists function keys (help)
dskp=\E!hˆM|Shift-F8|,           # Go to previous screen
dskq=\E!iˆM|Shift-F9|\E pA\EH\EJ, # Go to screen 1 and
quitdsp=\E pA|\EH\EJ,            # Terminal sequence for screen 1
dsp=\E pB|\EH\EJ,                # Terminal sequence for screen 2
dsp=\E pC|\EH\EJ,                # Terminal sequence for screen 3
dsp=\E pD|\EH\EJ,                # Terminal sequence for screen 4
dst=10,                          # Allow 1 second timeout buffer
```

Entry Format for dsinfo

Entries in the **dsinfo** file consist of comma-separated fields. The first field is a list of alternative names for the terminal, each name is separated by a pipe (|) character. Any text preceded by a pound (#) character is regarded as a comment and ignored by **dscreen**. The remaining fields are strings describing the capabilities of the terminal to the **dscreen** utility. Within these strings, the following escape codes are recognized:

Escape Sequence	**Description**
\E,\e	escape character
\n,\l	newline (or linefeed) character
\r	carriage return
\t	tab character
\b	backspace character
\f	formfeed character
\s	space character
\nnn	character with octal value *nnn*
ˆx	Ctrl-*x* for any appropriate *x* value

Any other character preceded by a backslash yields the character itself. The strings are entered as *type=string*, where *type* is the type of string as listed below, and *string* is the string value.

It is important that the entry fields in the **dsinfo** file be separated by commas. If a comma is omitted or truncated from the end of a **dsinfo** file entry, the file becomes unreadable by the **dscreen** utility and an error is returned to the display.

String Types

The string types are as follows:

dsk*x* A string type that starts with **dsk** describes a key. The type must be four letters long, and the fourth letter *x* indicates what action is taken when the key is received. The key types are:

Type	**Action**
dsks	Switch Screens
dskb	Block Input and Output
dske	End **dscreen**
dskq	Quit **dscreen** (exit status=1)
dskc	Create New Screen
dskp	Switch to Previous Screen
dskl	List Keys and Actions

Any other key type (that is, a string type **dsk*x*** that doesn't end in s, b, e, q, c, p, or l) causes no internal **dscreen** action, but shows up in the key listing and is recognized and acted upon.

A type of **dskn** (**n** for No Operation) should be used when no internal **dscreen** action is desired.

The value string for each key has three substrings, which are separated by pipe (|) characters.

Note: Use \| to include the | character in one of the substrings.

The first substring is the sequence of characters that the terminal sends when the key is pressed. The second substring is a label for the key that is printed when a list of keys is displayed. The third substring is a sequence of characters that **dscreen** sends to the terminal when this key is pressed before performing the action this key requests.

dsp A string type of **dsp** describes a physical screen in the terminal. One **dsp** string should be present for each physical screen in the terminal. The value string for each physical screen has two substrings, which are separated by a pipe (|) character.

The first substring is the sequence of characters to send to the terminal to display and output to the physical page on the terminal.

The second substring is sent to the terminal when the page is used for something new. This second substring is often set to the clear screen sequence. It is sent under the following two conditions:

1. When a new virtual terminal session is being created.
2. When there are more virtual terminals than there are physical screens. If a virtual terminal is selected which requires **dscreen** to reuse one of the physical screens, it will send this sequence to the screen to indicate that the screen contents do not match the output of the virtual terminal connected.

Note: Running with more virtual terminals than physical screens can be confusing and is not recommended; it can be avoided by defining no more screen selection keys (**dsks=**) than physical screens (**dsp=**) in the **dsinfo** entry.

dst A String with a type of **dst** adjusts **dscreen**'s input timeout. The value of the string is a decimal number. The timeout value is in tenths of seconds and has a maximum value of 255 (default=**1** [or 0.1 seconds]).

When **dscreen** recognizes a prefix of an input key sequence but does not have all the characters of the sequence, it will wait for more characters to be sent until it is recognizable. If the timeout occurs before more characters are received, the characters are sent on to the virtual screen and **dscreen** will not consider these characters as part of an input key sequence.

It may be necessary to raise this value if one or more of the keys **dscreen** is to trigger on is actually a number of keystrokes (that is assigning Ctrl-Z 1, Ctrl-Z 2, Ctrl-Z 3, etc. for screen selection, and Ctrl-Z N for new screen and so on).

Example 1

The following example **/usr/lbin/tty/dsinfo** entry is for a Wyse-60 with three screen sessions:

```
wy60|wyse60|wyse model 60,
dsks=^A'^M|Shift-F1|,
dsks=^Aa^M|Shift-F2|,
dsks=^Ab^M|Shift-F3|,
dskc=\200|Ctrl-F1|,
dske=\201|Ctrl-F2|\Ew0\E+,
dskl=\202|Ctrl-F3|,
dsp=\Ew0|\E+,
dsp=\Ew1|\E+,
dsp=\Ew2|\E+,
```

With this entry:

- Shift-F1 through Shift-F3 are used for selecting screens 1 through 3.
- Ctrl-F1 creates a new screen.
- Ctrl-F2 sends: `Esc w 0 Esc +` to the screen (switching to window 0 and clearing the screen) and then end **dscreen**.
- Ctrl-F3 will list the keys and their functions.

Each time a physical screen is used for a new screen, the sequence `Esc +` will be sent to the terminal, which will clear the screen.

Example 2

This example is for a Wyse-60 with three screen sessions, but one of the screens is on a second computer communicating through the second serial port on the terminal:

```
wy60-1|wyse60-1|wyse model 60 - first serial port
dsks=^A'^M|Shift-F1|,
dsks=^Aa^M|Shift-F2|,
dsks=^Ab^M|Shift-F3|\Ed#^Ab\r^T\Ee9,
dskc=\200|Ctrl-F1|,
dske=\201|Ctrl-F2|\Ed#\201^T\Ew0\E+,
dskl=\202|Ctrl-F3|,
dsp=\Ew0|\E+,dsp=\Ew1|\E+,
wy60-2|wyse60-2|wyse model 60 - second serial port
dsks=^A'^M|Shift-F1|\Ed#^A'\r^T\Ee8,
dsks=^Aa^M|Shift-F2|\Ed#^Aa\r^T\Ee8,
dsks=^Ab^M|Shift-F3|,
dskc=\200|Ctrl-F1|,
dske=\201|Ctrl-F2|\Ed#\201^T\Ew0\E+,
dskl=\202|Ctrl-F3|,
dsp=\Ew2|\E+,
```

dscreen must be run on both computers, with terminal type wy60-1 on the first computer and terminal type wy60-2 on the second computer (using the **-t** option to **dscreen**). The wy60-1 entry will be examined first.

The first two key entries are unchanged from the original wy60 entry. The third key, however, has type **dskb**, which means block both input and output. When this key is pressed, the sequence:

```
Esc d # Ctrl-A b CR Ctrl-T Esc e 9
```

is sent to the terminal; after this, output is blocked and **dscreen** continues scanning input for key sequences but discards all other input.

The sequence `Esc d #` puts the terminal in Transparent Print mode, which echoes all characters up to a Ctrl-T out through the other serial port.

The characters `Ctrl-A b CR` are sent out the other serial port, informing the **dscreen** process on the other computer that it should activate the window associated with the Shift-F3 key.

The Ctrl-T key exits the Transparent Print mode. The `Esc e 9` causes the terminal to switch to the other AUX serial port for data communications.

At this point, the other computer takes over and sends an `Esc w 2` to switch to the third physical screen, and then resumes normal communication.

The wy60-2 entry follows the same general pattern for keys Shift-F1 and Shift-F2:
- Switch to transparent print mode.
- Send function key string to other computer.
- Switch transparent print off.
- Switch to the other serial port.

The end key, Ctrl-F2, works the same for both computers; it sends the end key sequence to the other computer through the transparent print mechanism, switches the terminal to window 0, clears the screen, then exits.

Transparent Printing

Most terminals have an auxiliary port that can be connected to a serial printer. These terminals support two print modes, Auxiliary and Transparent. If both print modes are OFF, data received by the terminal is simply displayed on the screen. With Auxiliary print mode ON, data received by the terminal is displayed on the screen and is also transmitted to the printer. With Transparent Print Mode ON, the terminal transmits data received directly to the printer, without displaying it on the screen.

Transparent printing allows you to use your terminal in a normal manner, while information is also being sent over the same serial connection from the host to the printer connected to the terminal's auxiliary printer port. This is transparent printing. The transparent printing software determines whether packets of data are bound for the screen or for the printer, and precedes data bound for the printer with the Transparent Print Mode ON command, and follows it with the Transparent Print Mode OFF command.

Data for the terminal screen has the highest priority, and data is sent to the printer only when there is a break in information being sent to the screen. If continuous data is being transmitted to the terminal device, nothing gets sent to the printer.

Whenever an auxiliary printer port is used, flow control to the printer becomes an issue. If the printer falls behind and invokes flow control, output to both the printer and the terminal is stopped. The transparent print feature provides three parameters, accessible through SMIT, to limit printer output and avoid this situation.

The SMIT Transparent Print Maximum Characters per Second parameter limits the maximum printer port character-per-second data rate. This number should be set to the minimum character rate the printer can sustain in typical use.

The SMIT Transparent Print Maximum Character Packet Size parameter limits the number of characters queued to the printer ahead of terminal output. Lower numbers increase system overhead, higher numbers result in keystroke echo delays. Specify a value of **50** for 9600 baud.

The SMIT Transparent Print Printer Buffer Size parameter should be set to a value just below the printer's buffer size. After a period of inactivity, the driver will burst up to this many characters to the printer to fill the print buffer before slowing to the maxcps rate.

The printer on/off strings are also set using SMIT. A cable must be connected between the auxiliary port of the terminal and the printer. The baud rate on the terminal auxiliary port and the printer must be the same, and the printer and the auxiliary port of the terminal must use the same handshaking mode. The auxiliary port must also be enabled. If your terminal is not one of those directly supported, you must know the escape sequence of your terminal.

Refer to your terminal and printer manuals for connection information, escape codes, and to see what handshaking modes are supported (for example, busy/ready or RTS/CTS). Printer devices (xtty1, for example) must not be in either the **/etc/inittab** or **/etc/ttys** files and must not be enabled.

For information about activating transparent printing, see: ″Configuring a Terminal-Attached Printer″.

Setting Terminal Options with stty-cxma

stty-cxma is a utility program that sets and displays the terminal options for the Micro Channel 128-port, ISA 8-, and 128-port, PCI 8-, and PCI 128-port adapters and is located in `/usr/lbin/tty` directory. The format is:

```
stty-cxma [-a] [option(s)] [ttyname]
```

With no options, **stty-cxma** displays all special driver settings, modem signals, and all standard parameters displayed by stty(1) for the tty device referenced by standard input. Command options are provided to change flow control settings, set transparent print options, force modem control lines, and display all tty settings. Any unrecognized options are passed to stty(1) for interpretation. The options are:

-a Displays all of the unique adapter option settings, as well as all of the standard tty settings reported by the **stty -a** command.

ttyname
Sets and displays options for the given tty device, instead of standard input. This form can be used with a tty pathname prefixed by `/dev/` or with a simple tty name beginning with tty. This option may be used on a modem control line when no carrier is present.

The following options specify transient actions to be performed immediately:

break Sends a 250 ms break signal out on the tty line.

flush Indicated an immediate flush (discard) of tty input and output.

flushin
Flushes tty input only.

flushout
Flushes tty output only.

The following options specify actions that are reset when the device is closed. The device will use the default values the next time it is opened.

stopout
Stops output exactly as if an XOFF character was received.

startout
Restarts stopped output exactly as if an XON character was received.

stopin Activates flow control to stop input.

startin
Releases the flow control to resume stopped input.

[-]dtr [drop]
Raises the DTR modem control line, unless DTR hardware flow control is selected.

[-]rts [drop]
Raises the RTS modem control line, unless RTS hardware flow control is selected.

The following options remain in effect until the system is rebooted or until the options are changed.

[-]fastcook
Performs cooked output processing on the intelligent card to reduce host CPU usage, and increase raw mode input performance.

[-]fastbaud
Alters the baud rate tables, so 50 baud becomes 57,600 baud, 75 baud becomes 76,800 baud, 110 baud becomes 115,200 baud, and 200 baud becomes 230,000 baud for supported devices.

[-]rtspace
Enables/disables RTS hardware input flow control, so RTS drops to pause remote transmission.

[-]ctspace
Enables/disables CTS hardware output flow control, so local transmission pauses when CTS drops.

[-]dsrpace
Enables/disables DSR hardware output flow control, so local transmission pauses when DSR drops.

[-]dcdpace
Enables/disables DCD hardware output flow control, so local transmission pauses when DCD drops.

[-]dtrpace
Enables/disables DTR hardware input flow control, so DTR drops to pause remote transmission.

[-]forcedcd
Disable [re-enable] carrier sense, so the tty may be opened and used even when carrier is not present.

[-]altpin
Maps the RJ-45 connector pinouts to the default 10-pin connector values or the 8-pin connector values. When this parameter is **enabled**, the location of DSR and DCD is switched so that DCD is available when using an 8-pin RJ-45 connector instead of the 10-pin RJ-45 connector. (Default=**disable**.)

Possible values:

enable (specifies 8-pin connector values)

disable (specifies 10-pin connector values)

startc *c*
Sets the XON flow control character. The character may be given as a decimal, octal, or hexadecimal number. Octal numbers are recognized by the presence of a leading zero, and

hexadecimal numbers are denoted by a leading 0x. For example, the standard XON character, CTRL-Q, can be entered as 17 (decimal), 021 (octal), or 0x11 (hexadecimal).

stopc *c*
Sets the XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

astartc *c*
Sets auxiliary XON flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

astopc *c*
Sets auxiliary XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

[-]aixon
Enables auxiliary flow control, so that two unique characters are used for XON and XOFF. If both XOFF characters are received, transmission will not resume until both XON characters are received.

[-]2200flow
Uses 2200 style flow control on the port. The 2200 terminals support an attached printer and use four flow control characters: terminal XON (0xF8), printer XON (0xF9), terminal XOFF (0xFA) and printer XOFF (0xFB).

[-]2200print
Determines how these flow control characters are interpreted. If 2200print is set, run independent flow control for terminal and transparent print devices. Otherwise, terminal and printer flow control are logically tied together. If either XOFF character is received, all output is paused until the matching XON character is received.

maxcps *n*
Sets the maximum characters per second (cps) rate that characters are output to the transparent print device. The rate chosen should be just below the average print speed. If the number is too low, printer speed will be reduced. If the number is too high, the printer uses flow control, and user entry time is reduced. The default is 100 cps.

maxchar *n*
Sets the maximum number of transparent print characters the driver places in the output queue. Reducing this number increases system overhead; increasing this number delays operator keystroke echo times when the transparent printer is in use. The default is 50 characters.

bufsize *n*
Sets the driver's estimate of the size of the transparent printer's input buffer. After a period of inactivity, the driver bursts this many characters to the transparent printer before reducing to the maxcps rate. The default is 100 characters.

onstr *s*
Sets the terminal escape sequence to turn transparent printing on. The strings can be composed of standard ASCII printing and non-printing characters. Control (non-printing) characters must be entered by their octal values, and must consist of three digits preceded by a back-slash character. For example, the Escape character, 33 octal, should be entered as \033. If transparent printing is turned on by the string <Esc>[5i (ANSI standard), it would be entered as: \033[5i.

offstr *s*
Sets the terminal escape sequence to turn transparent printing off. Refer to **onstr** *s* for the format of the strings.

term *t*
Sets the transparent printer on/off strings to values found in the internal default table. Internal defaults are used for the following terminals: adm31, ansi, dg200, dg210, hz1500, mc5, microterm, multiterm, pcterm, tvi, vp-a2, vp-60, vt52, vt100, vt220, wyse30, wyse50, wyse60, or wyse75. If the

terminal type is not found in the internal default table, ditty reads the terminfo entry for the terminal type and sets transparent print on/off strings to values given by the mc5/mc4 attributes found in the terminfo entries.

Chapter 4. Standard, 8-Port Micro Channel, and 16-Port Asynchronous Adapters

This chapter describes the standard, 8-port, and 16-port asynchronous adapters.

Topics discussed are:
- Standard I/O Ports
- Moving a TTY to Another Port
- Removing a TTY
- 8-Port Micro Channel Asynchronous Adapters
- 16-Port Asynchronous Adapters

Standard I/O Ports

Most system unit models have two integrated (standard) EIA 232 asynchronous serial ports. The model M20/M2A features a single integrated asynchronous serial port that can be converted to support two serial devices using an optional fanout cable. EIA 232 asynchronous serial devices can be attached directly to the standard serial ports using standard serial cables with 9-pin or 25-pin D-shell connectors.

Note: For the Itanium-based platform, EIA 232 asynchronous serial devices can be attached directly to the standard serial ports using standard serial cables with 9-pin D-shell connectors.

The machines capable of multiprocessing have three serial ports.

Configuring an EIA 232 Asynchronous Terminal Device

This procedure allows you to define and configure a tty device connected to a standard serial port, an 8-port, or 16-port asynchronous adapter.

Procedure

1. Use the **smit mktty** fast path to access the **Add a TTY** menu.
2. Select **Add a TTY**.
3. Select **tty rs232 Asynchronous Terminal**.
4. Make a selection from the available standard I/O, 8-port, or 16-port adapters displayed on the screen. If no adapters are displayed or if they are in a defined state, check the configuration, cabling, and setup again.
5. In the displayed dialog fields, you can add or change the tty attributes.
6. When you have finished, select **Do**.

Configuring an EIA 232 Asynchronous Printer/Plotter Device

This procedure allows you to define and configure a printer/plotter device connected to a standard serial port, an 8-port asynchronous adapter, or a 16-port asynchronous adapter.

Procedure

1. To create a printer/plotter device on an asynchronous adapter, use the **smit pdp** fast path to access the **Printer/Plotter Devices** menu.
2. Select **Add a Printer/Plotter**.

3. Make a selection from the list of printer and plotter types shown on the screen, and press Enter. For this example, the following selection was made:

   ```
   osp Other serial printer
   ```
4. Select the **rs232** option.
5. Make a selection from the available 8-port controllers on the screen. If no controllers are displayed or if they are shown in a defined state, check the configuration, cabling, and setup again.
6. In the displayed dialog fields, you can add or change the printer/plotter device attributes.
7. When you have finished, select **Do**.

Moving a TTY to Another Port

This procedure allows you to move a tty device to another port and retain all the characteristics of the device. Use the System Manager Interface Tool (SMIT) to build the **chdev** command or issue the **chdev** command directly from the command line.

Prerequisites

1. A tty device must be defined to the system.
2. The tty must be disabled. Refer to the **pdisable** command.

Move a TTY to Another Port Using SMIT

1. Use the **smit movtty** fast path to access the **TTY** screen.
2. Select the tty device type to be moved.
3. Select the adapter device to which the tty is or will be attached.
4. At the PORT number field, use the **List** option to display a list of available ports. Select the port to which the tty device is to be moved.
5. Select **Do** to move the device.

Move a TTY to Another Port from the Command Line

Enter the **chdev** command at the command line, specifying the following flags:

-l *Name*	Identifies the tty device to be moved.
-p *ParentName*	Identifies the parent device.
-w *ConnectionLocation*	Identifies the port number to which the tty device is to be moved.

The following example moves a defined tty device `tty11` to port `0` on serial adapter `sa5`:

```
chdev -l tty11 -p sa5 -w 0
```

See the **chdev** command for a detailed description.

Removing a TTY

This procedure allows you to make a tty device unavailable and either keep its definition in the customized database or remove it from the database. To make a device unavailable is to unconfigure it. A tty device that is made unavailable but that has not been deleted from the database can be made available again with the same attributes. If the definition of the tty device has been deleted, the device cannot be made available again.

Prerequisites

1. A tty device must be defined to the system.
2. The tty must be disabled. Refer to the **pdisable** command.

Remove TTY Using SMIT

1. Use the **smit rmtty** fast path to access the **TTY** screen.
2. Select the tty device you want to make unavailable or remove.
3. Select **yes** to keep the definition of the tty in the database or **no** to remove the definition.
4. Select **Do** to remove the tty device.

Remove a TTY from the Command Line

Enter the **rmdev** command at the command line as follows:

- To make the logical device `tty0` unavailable but keep its definition in the customized database, enter:
  ```
  rmdev -l tty0
  ```
- To make the logical device `tty0` unavailable and delete its definition from the customized database, enter:
  ```
  rmdev -l tty0 -d
  ```

8-Port Micro Channel Asynchronous Adapters

Note: The following section is not applicable to the Itanium-based platform.

The family of asynchronous adapters is based on a common functional design. The individual adapter characteristics, however, are determined by the supported device interfaces. The family consists of three adapters:

- 8-Port Asynchronous Adapter - EIA 232
- 8-Port Asynchronous Adapter - MIL-STD-188
- 8-Port Asynchronous Adapter - EIA 422A

The family of 8-port adapters is based on the dual universal asynchronous receiver and transmitter (DUART) chip providing two serial communications channels.

The following sections contain detailed information about 8-port adapters:

- 8-Port Asynchronous Adapter - EIA 232 Description
- Installing the 8-Port Asynchronous Adapter
- 8-Port Asynchronous Adapter Hardware Information
- Communications Channel Priority
- 8-Port Asynchronous Adapters Interrupt Logic Description
- 8-Port Asynchronous Adapters MIL-STD 188 Interface Signals
- MIL-STD 188 Signal Voltage Levels
- Standards Compliance
- 8-Port Asynchronous Adapters EIA 422A Interface Signals
- EIA 422A Signal Voltage Levels
- 8-Port Asynchronous Adapters EIA 232 Interface Signals
- EIA 232 Signal Voltage Levels
- 8-Port Asynchronous Adapters Control Logic

8-Port Asynchronous Adapter - EIA 232 Description

The EIA 232 is an 8-Port asynchronous adapter that provides support for attaching a maximum of eight EIA 232D asynchronous serial devices (such as modems, terminals, plotters, and printers) to a system unit. The system must be based on a Micro Channel bus or an ISA bus and support up to eight 8-port adapters.

This adapter is fully programmable and supports asynchronous communications only. It can also add and remove start and stop bits and supports even, odd, or no parity on serial data. A programmable baud rate generator allows operation from 50 to 38,400 bps for the Micro Channel bus and 50 to 115,200 bps for the ISA bus. The adapters support 5-, 6-, 7-, or 8-bit characters with 1, 1.5, or 2 stop bits. A priority interrupt system controls transmit, receive, error, line status, and data set interrupts.

Installing the 8-Port Asynchronous Adapter

The 8-port asynchronous adapter fits into a single Micro Channel slot in the system. To install the adapter use the following steps:

1. Verify that all users are logged off the system and run the following command:
   ```
   shutdown -F
   ```
2. When the **shutdown** command completes, turn the system power switch to the off position.
3. Open the system case and insert the 8-port asynchronous adapter into a free Micro Channel slot.
4. Attach the 78-pin D-shell connector from the 8-port interface cable to the 8-port adapter.
5. Put the cover panels back on the system unit.
6. Push the system power switch to the on position.

   The system will recognize and configure the 8-port adapter during the boot process.
7. After the boot completes, log in using the root user ID boot process.
   ```
   lsdev -Cc adapter | pg
   ```

Only those adapters that are in an available state are ready for use by the system.

If the newly installed adapter is *not* available, then verify:

- The adapter is installed correctly into the Micro Channel slot.
- All necessary cabling is attached and fitted tightly into place.
- Run the command: **errpt -a | pg** and examine the system error report for problems relating to the adapters.
- Run the command: **cfgmgr -v | pg**. This command will attempt to reconfigure the adapter without rebooting. Observe the paged output for errors.

If running **cfgmgr** fails, a reboot will be necessary.

8-Port Asynchronous Adapter Hardware Information

The system interface presents a 3-bit address and 8-bit data as well as control lines to the DUART chip. Data from the system interface is serialized for transmission to an external device. The serial data can include a parity bit at the byte boundary. Conversely, data from an external device is deserialized for transmission to the system interface. This data may also include a parity bit, which can be optionally checked. As an option, the channel can operate in first-in-first-out (FIFO) mode.

In FIFO mode, up to 16 bytes can be buffered in both the transmitter and receiver. The serial interface uses start-stop protocol for both data transmission and reception. That is, each byte (plus the parity bit) is framed by one or more start bits and stop bits, which allows synchronization on an individual character (byte) basis.

The DUART chip uses a 12.288 MHz oscillator to generate its internal timing to drive the transmitter and receiver logic. The channel supports full duplex operation. Four DUART chips are implemented on each 8-port adapter.

Thirteen system-accessible registers are available. Programmable features on each channel include:

- Character length: 5, 6, 7, or 8 bits
- Parity generation/detection: Even, odd, or none
- Number of stop bits: 1, 1.5, or 2
- Enable/disable interrupts. Received data available
- Transmitter holding register empty
- Line status
- Overrun error
- Parity error
- Framing error
- Break.

The following table is a summary of port (device interface) characteristics for the adapters.

Parameter	EIA 232	MIL-STD 188	EIA 422A
Topology	Point to Point	Point to Point	Point to Point
Maximum data rate	138.4Kbps (MC)/115.2 (ISA)	138.4Kbps	138.4Kbps
Transmission media	Multiconductor	Multiconductor	Multiconductor
Number of cable wires	9 including signal ground	9 including signal ground	5 including signal ground
Maximum cable length	61 m (200 feet)	130 m at 38.4Kbps	1200 m < 90Kbps
Device connector	25-pin D	25-pin D	25-pin D
Electrical interface	Unbalanced	Unbalanced	Balanced
Bit encoding	Digital bi-level	Digital bi-level	Digital bi-level

The interrupt arbitration logic sets priority for adapters according to the following scheme:

```
Adapter   Priority
    1     Highest
    2       |
    3       |
    4       |
    5       |
    6       |
    7       |
    8     Lowest
```

Communications Channel Priority

The DUART channels with pending interrupts are serviced according to a fixed-priority scheme. The highest priority is assigned to port 0. Next in priority is port 1, and so forth. The lowest priority is port 7.

8-Port Asynchronous Adapters Interrupt Logic Description

The interrupt logic is divided into two sections:

- Interrupt generation logic
- Interrupt arbitration logic

Both logic sections are implemented on every 8-port adapter. The interrupt generation logic provides the interface to the system. This logic generates the system interrupt requests and contains the interrupt-sharing circuitry.

The function of the interrupt arbitration logic is to identify the 8-port adapter with the highest priority interrupt pending. The logic then places the interrupt information for the highest priority port in the Interrupt Arbitration register. This is accomplished in one read operation.

The interrupt arbitration logic is unique to the 8-port adapter and should not be confused with the Micro Channel arbitration logic.

Interrupt Generation Logic

The adapter implements the following eight system interrupt request lines:

- IRQ 3
- IRQ 5
- IRQ 9
- IRQ 10
- IRQ 11
- IRQ 12
- IRQ 14
- IRQ 15

Only one request line is active during normal operation. All 8-port adapters in one system should use the same interrupt level for optimal system performance. The active line is selected by writing to the appropriate POS register during the setup cycle. The adapter supports interrupt sharing and implements an open collector configuration. In this arrangement, the interrupt line is pulled high by a system pull-up resistor. The adapter pulls the line low to indicate an active interrupt request.

Interrupt Arbitration Logic

Up to eight 8-port adapters can co-reside and concurrently operate in a system. The interrupt arbitration logic determines the priority for software service when two or more 8-port or 16-port adapters generate interrupts. This logic provides the system with adapter and port identification as well as the interrupt type in a single read operation. Once an interrupt request is detected, the system reads the 16-bit interrupt arbitration register, which is located at I/O address 0130.

8-Port Asynchronous Adapters MIL-STD 188 Interface Signals

The following interface signals are implemented on each port of the adapter:

Signal	Definition
Tx Data	Transmit Data
RTS	Request To Send
CTS	Clear To Send
DSR	Data Set Ready
Rx Data	Receive Data
DCD	Data Carrier Detect
DTR	Data Terminal Ready
RI	Ring Indicator
Sig Gnd	Signal Ground

MIL-STD 188 Signal Voltage Levels

Voltage levels for the MIL-STD 188 Adapter are explained in the following sections:

- Normal Mark and Space Polarity
- Mark and Space Polarity Inversion.

Normal Mark and Space Polarity

The signal is in the mark state when the voltage on the interchange circuit, measured at the interface point, is less than -4 V dc with respect to the signal ground. The signal is in the space state when the voltage is greater than +4 V dc with respect to the signal ground. The region between +4 V dc and -4 V dc is defined as the transition region and is not a valid level. The voltage that is less than -6 V dc or greater than +6 V dc is also not a valid level.

During the transmission of data, the mark state denotes binary 1 and the space state denotes binary 0.

For interface control circuits, the function is ″on″ when the voltage is greater than +4 V dc with respect to the signal ground and is ″off″ when the voltage is less than -4 V dc with respect to the signal ground. MIL-STD 188 signal levels are shown in the following table:

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
+ Voltage	0	Space	On
- Voltage	1	Mark	Off

Mark and Space Polarity Inversion

Military standard MIL-STD 188 requires that adapters provide the capability to optionally invert the polarities of the mark and space states of the transmit and receive lines. The capability is provided independently on each port.

The DUART modem control register bit 3 (Out 2) is used for this purpose. When bit 3 is set to a value of 1, the polarities for the mark and space states are set to the normal state. When bit 3 is set to a value of 0, the polarities for the mark and space states are inverted.

The signal is in the *space state* when the voltage is less than -4 V dc with respect to the signal ground. The signal is in the mark state when the voltage is greater than +4 V dc with respect to the signal ground.

The region between +4 V dc and -4 V dc is defined as the *transition region* and is not a valid level. The voltage that is less than -6 V dc or greater than +6 V dc is also not a valid level.

Standards Compliance

The electrical characteristics of the 8-Port asynchronous MIL-STD 188 adapter ports conform to those sections of MIL-STD 188-114 that address an unbalanced voltage interface. The standard is dated March 24, 1976.

The adapter ports meet the functional requirements for asynchronous operation (start-stop protocol) as described in the EIA Standard 232C dated October 1969 and in the EIA Standard 232D dated January 1987.

8-Port Asynchronous Adapters EIA 422A Interface Signals

The following EIA 422A interface signals are implemented on each port of the adapter:

Signal	Definition
TxA	Transmit Data

Signal	Definition
TxB	Transmit Data
RxA	Receive Data
RxB	Receive Data
Sig Gnd	Signal Ground

EIA 422A Signal Voltage Levels

The line driver produces a differential voltage in the range of 2 to 6 volts (measured at the generator interface point). The magnitude of the differential voltage at the receiver must be in the range of 200 millivolts to 6 volts (measured at the load interface point).

Measurements are taken at terminal A (positive lead) with respect to terminal B (negative lead). The following table describes the signal states with respect to voltage levels:

Interchange Voltage	Binary State	Signal Condition
+ Voltage	0	Space
- Voltage	1	Mark

Surge Protection Circuitry

The 8-Port asynchronous EIA 422A adapter supports indoor cabling up to 1200 m (4000 ft) in length. Cables of such lengths are susceptible to sudden voltage surges due to induced voltages such as indirect lightning strikes. Secondary surge protection circuitry is implemented on the EIA 422A adapter to protect it from these voltage surges. The surge protection circuitry is implemented on the adapter interface data lines.

Fail-Safe Circuitry

Fail-safe circuitry has been added to the input leads of each EIA 422A receiver to prevent fault conditions when the receiver is not connected to a driver (open cable). The fail-safe circuitry sets the receiver to the mark state (binary 1) whenever the receiver is not connected to a driver.

Standards Compliance

The electrical characteristics of the 8-Port asynchronous EIA 422A adapter ports comply with the EIA Standard 422A dated December 1978.

8-Port Asynchronous Adapters EIA 232 Interface Signals

The following interface signals are implemented on each port of the adapter:

Signal	Definition
TxD	Transmit Data
RTS	Request To Send
CTS	Clear To Send
DSR	Data Set Ready
RxD	Receive Data
DCD	Data Carrier Detect
DTR	Data Terminal Ready
RI	Ring Indicator
Sig Gnd	Signal Ground

EIA 232 Signal Voltage Levels

The signal is in the mark state when the voltage on the interchange circuit, measured at the interface point, is less than -3 V dc with respect to the signal ground. The signal is in the space state when the voltage is greater than +3 V dc with respect to the signal ground. The region between +3 V dc and -3 V dc is defined as the *transition region* and is not a valid level. Voltage less than -15 V dc or greater than +15 V dc is also not a valid level.

During the transmission of data, the mark state denotes binary state 1 and the space state denotes binary state 0.

For interface control circuits, the function is on when the voltage is greater than +3 V dc with respect to the signal ground and is off when the voltage is less than -3 V dc with respect to the signal ground. See the following table for EIA 232 signal levels:

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
+ Voltage	0	Space	On
- Voltage	1	Mark	Off

Standards Compliance

The electrical characteristics of the 8-Port asynchronous EIA 232 adapter ports conform to the EIA Standard 232C dated October 1969 and to the EIA Standard 232D dated January 1987.

The adapter ports meet the functional requirements for asynchronous operation (start-stop protocol) as described in the EIA Standard 232C dated October 1969 and in the EIA Standard 232D dated January 1987.

8-Port Asynchronous Adapters Control Logic

The PAL-based control logic section coordinates the activities of all major adapter functions and is clocked with a 40 MHz square-wave generator. It interfaces with the Micro Channel, and its functions include decoding addresses, checking address parity, responding with the proper I/O control signals, and driving the selected interrupt request (IRQ) line (one of eight IRQ lines).

The control logic interfaces with the other adapter logic blocks and in this capacity provides the control lines to the communication channels (DUART) and the interrupt arbitration logic. The control logic also interfaces with the data bus driver logic and provides control for the direction of data flow and for the selection data bytes, which are placed onto the local bus. It controls the data parity generator, parity checker, and latches.

16-Port Asynchronous Adapters

Note: The following section is not applicable to the Itanium-based platform.

The family of adapters is based on a common functional design. The individual adapter characteristics, however, are determined by the supported device interfaces. The family consists of two adapters, detailed in the following sections:

- 16-Port Asynchronous Adapter - EIA 422A
- 16-Port Asynchronous Adapter - EIA 232.

The family of 16-port adapters is based on the dual universal asynchronous receiver and transmitter (DUART) chip, which provides two serial communications channels. More information on the DUART chip and its functionality can be found in the 16-Port Asynchronous Adapter Hardware Information.

The following sections contain detailed information about 16-port adapters:

- 16-Port Asynchronous Adapter - EIA 4224A Description
- Installing the 16-Port Asynchronous Adapter
- 16-Port Asynchronous Adapter Hardware Information
- 16-Port Asynchronous Adapters Board Priority
- 16-Port Asynchronous Adapters Interrupt Logic Description
- 16-Port Asynchronous Adapters EIA 232 Interface Signals
- Standards Compliance
- 16-Port Asynchronous Adapters EIA 422A Interface Signals

16-Port Asynchronous Adapter - EIA 422A Description

The 16-Port asynchronous adapter - EIA 232 provides support for attaching a maximum of 16 EIA 232 asynchronous serial devices (printers and terminals) to a system unit. Up to eight adapters (any combination within the family) can be used in a single system unit.

This adapter is fully programmable and supports asynchronous communications only. It adds and removes start bits and stop bits. The adapters support even, odd, or no parity on serial data. A programmable baud-rate generator allows operation from 50 to 38400 bps. The adapters support 5-, 6-, 7-, or 8-bit characters with 1, 1.5, or 2 stop bits. A priority interrupt system controls transmit, receive, error, line status, and data set interrupts. The 16 connectors for device attachment are provided in the EIA 422A 16-port cable assembly.

Characteristics

- Standard Micro Channel form factor card.
- Data rates up to 38.4K bps per port.
- 16 byte buffering on transmit and receive.
- Single 78-pin output connector (Multiport interface cable attaches to this connector).
- Surge protection circuitry.
- Supports cabling up to 1200 m (4000 ft).
- Supports the TxD and RxD interface signals.
- 8-bit/16-bit Micro Channel slave interface.

Installing the 16-Port Asynchronous Adapter

The 16-port asynchronous adapter fits into a single Micro Channel slot in the server. To install the adapter, use the following steps:

1. Verify that all users are logged off the system and run the following command:
   ```
   shutdown -F
   ```
2. When the **shutdown** command completes, turn the system power switch to the ″off″ position.
3. Open the server case and insert the 16-port asynchronous adapter into a free Micro Channel slot.
4. Attach the 78-pin D-shell connector from the 16-port interface cable to the 16-port adapter.
5. Put cover panels back on the system unit
6. Push the system power switch to the ″on″ position.

   The system will recognize and configure the 16-port adapter during the boot process.

After the boot completes, login using the root user ID and issue the following command to check the adapter availability:

```
lsdev -Cc adapter | pg
```

Only those adapters in the available state are ready for use by the system.

If the newly installed adapter is NOT available, then verify the following:

1. The adapter is installed correctly into the Micro Channel slot.
2. All necessary cabling is attached and fitted tightly into place.
3. Run the command: **errpt -a | pg** and examine the system error report for problems relating to the adapters.
4. Run the command: **cfgmgr -v | pg**. This command will attempt to reconfigure the adapter without rebooting. Watch the paged output for errors.
5. If running **cfgmgr** fails, a reboot will be necessary.

16-Port Asynchronous Adapter Hardware Information

The system interface presents a 3-bit address and 8-bit data as well as control lines to the chip. Data from the system interface is serialized for transmission to an external device. The serial data may include a parity bit at the byte boundary. Conversely, data from an external device is deserialized for transmission to the system interface. This data may also include a parity bit, which can be optionally checked. As an option, the channel can operate in first-in-first-out (FIFO) mode.

In FIFO mode, up to 16 bytes can be buffered in both the transmitter and receiver. The serial interface uses start-stop protocol for both data transmission and reception. That is, each byte (plus parity bit) is framed by a start bit and stop bit, which allows synchronization on an individual character (byte) basis.

The DUART chip uses a 12.288 MHz oscillator to generate its internal timing to drive the transmitter and receiver logic. The channel supports full duplex operation. Eight DUART chips are implemented on each 16-port adapter.

Thirteen system-accessible registers are available. Programmable features on each channel include:

- Character length: 5,6, 7, or 8 bits
- Parity generation/detection: even, odd, or none
- Number of stop bits: 1, 1.5, or 2
- Enable/disable interrupts. Received data available
- Transmitter holding register empty
- Line status
- Overrun error
- Parity error
- Framing error
- Break.

The following table is a summary of port (device interface) characteristics for the adapters.

Parameter	**EIA 232**	**EIA 422A**
Topology	Point to Point	Point to Point
Maximum data rate (standard)	20Kbps	2Mbps
Maximum data rate (board)	38.4Kbps	38.4Kbps
Transmission media	Multiconductor	Multiconductor
Number of cable wires	5 including signal ground	5 including signal ground
Maximum cable length	61 m (200 ft)	1200 m < 90Kbps
Device connector	25-pin D	25-pin D

Electrical interface	Unbalanced	Balanced
Bit encoding	Digital bi-level	Digital bi-level

16-Port Asynchronous Adapters Adapter Board Priority

The interrupt arbitration logic sets priority for adapters according to the following scheme:

```
Adapter   Priority
    0     Highest
    1        |
    2        |
    3        |
    4        |
    5        |
    6        |
    7        |
    8        |
    9        |
    10       |
    11       |
    12       |
    13       |
    14       |
    15    Lowest
```

Communications Channel Priority

The DUART channels with pending interrupts are serviced according to a fixed-priority scheme. The highest priority is assigned to port 0. Next in priority is port 1, and so forth. The lowest priority is port 15.

16-Port Asynchronous Adapters Interrupt Logic Description

The interrupt logic is divided into two sections:

- Interrupt generation logic
- Interrupt arbitration logic.

Both logic sections are implemented on every 16-port adapter. The interrupt generation logic provides the interface to the system. This logic generates the system interrupt requests and contains the interrupt-sharing circuitry.

The function of the interrupt arbitration logic is to identify the 16-port adapter with the highest priority interrupt pending. The logic then places the interrupt information of the highest priority port in the interrupt arbitration register. This is accomplished in one read operation.

The interrupt arbitration logic is unique to the 16-port adapters and should not be confused with the Micro Channel arbitration logic.

Interrupt Generation Logic

The adapter implements the following eight system interrupt request (IRQ) lines:

- IRQ 3
- IRQ 5
- IRQ 9
- IRQ 10
- IRQ 11
- IRQ 12
- IRQ 14

- IRQ 15

Only one request line is active during normal operation. All 16-port adapters in one system should use the same interrupt level for optimal system performance. The active line is selected by writing to the appropriate POS register during the setup cycle. The adapter supports interrupt sharing and implements an open collector configuration as defined in the Micro Channel architecture. In this arrangement, the interrupt line is pulled high by a system pull-up resistor. The adapter pulls the line low to indicate an active interrupt request.

Interrupt Arbitration Logic

Up to eight 8-port or 16-port adapters can co-reside and concurrently operate in a system. The interrupt arbitration logic determines the priority for software service when two or more 8-port or 16-port adapters generate interrupts. This logic provides the system with adapter and port identification as well as the interrupt type in a single read operation. Upon detection of an interrupt request, the system reads the 16-bit interrupt arbitration register located at I/O address 0130.

16-Port Asynchronous Adapters EIA 232 Interface Signals

The following interface signals are implemented on each port of the adapter:

Signal	**Definition**
TxD	Transmit data
DCD	Data carrier detect
DTR	Data terminal ready
RxD	Receive data
Sig Gnd	Signal ground

EIA 232 Signal Voltage Levels

The signal is in the mark state when the voltage on the interchange circuit, measured at the interface point, is less than -3 V dc with respect to the signal ground. The signal is in the space state when the voltage is greater than +3 V dc with respect to the signal ground. The region between +3 V dc and -3 V dc is defined as the transition region and is not a valid level. Voltage less than -15 V dc or greater than +15 V dc is also not a valid level.

During the transmission of data, the mark state denotes binary state 1 and the space state denotes binary state 0.

For interface control circuits, the function is on when the voltage is greater than +3 V dc with respect to the signal ground and is off when the voltage is less than -3 V dc with respect to the signal ground. See the following table for EIA 232 signal levels.

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
+ Voltage	0	Space	On
- Voltage	1	Mark	Off

Standards Compliance

The electrical characteristics of the 16-Port asynchronous EIA 232 adapter ports conform to the EIA Standard 232C dated October 1969 and to the EIA Standard 232D dated January 1987.

The adapter ports meet the functional requirements for asynchronous operation (start-stop protocol) as described in the EIA Standard 232C dated October 1969 and in the EIA Standard 232D dated January 1987.

16-Port Asynchronous Adapters EIA 422A Interface Signals

The following EIA 422A interface signals are implemented on each port of the adapter:

Signal	**Definition**
TxA	Transmit Data
TxB	Transmit Data
RxA	Receive Data
RxB	Receive Data
Sig Gnd	Signal Ground

EIA 422A Signal Voltage Levels

The line driver produces a differential voltage in the range of 2 to 6 volts (measured at the generator interface point). The magnitude of the differential voltage at the receiver must be in the range of 200 millivolts to 6 volts (measured at the load interface point).

Measurements are taken at terminal A (positive lead) with respect to terminal B (negative lead). The following table describes the signal states with respect to voltage levels:

Interchange Voltage	Binary State	Signal Condition
+ Voltage	0	Space
- Voltage	1	Mark

Surge Protection Circuitry

The 16-Port Asynchronous EIA 422A adapter supports indoor cabling up to 1200 m (4000 ft) in length. Cables of such lengths are susceptible to sudden voltage surges due to induced voltages such as indirect lightning strikes. Secondary surge protection circuitry is implemented on the EIA 422A adapter to protect it from these voltage surges. The surge protection circuitry is implemented on the adapter interface data lines.

Fail-Safe Circuitry

Fail-safe circuitry has been added to the input leads of each EIA 422A receiver to prevent fault conditions when the receiver is not connected to a driver (open cable). The fail-safe circuitry sets the receiver to the mark state (binary 1) whenever the receiver is not connected to a driver.

Standards Compliance

The electrical characteristics of the 16-Port asynchronous EIA 422A adapter ports comply with the EIA Standard 422A dated December 1978.

Chapter 5. 64-Port Asynchronous Adapters

Note: The following chapter is not applicable to the Itanium-based platform.

The following sections contain information on planning, configuring, and managing the 64-port asynchronous adapter. The 64-port adapter is only supported on Micro Channel uniprocessor machines.

- 64-Port Asynchronous Adapter Subsystem
- Transient Voltage Line Protector
- 64-Port Problem Determination and Troubleshooting

64-Port Asynchronous Adapter Subsystem

The 64-Port asynchronous controller subsystem provides attachment for 64 asynchronous communication ports from a single I/O slot in the system unit. The subsystem has connectors to attach up to four external 16-port asynchronous concentrators thus giving a total of 64 asynchronous ports.

A single concentrator provides 16 connections for EIA 232 asynchronous devices. Although application dependent, data rates up to 38.4Kbps per port are supported.

A 7.6 m (25 ft) controller attachment cable is included with each concentrator. Cabling distances of up to 762 m (2500 ft) between controller and concentrator are supported but must be supplied by the customer. EIA 232 cabling of up to 61 m (200 ft) between concentrator and device is also supported.

The 16-Port asynchronous concentrator uses RJ-45 connectors for device attachment. The RJ-45 to DB25 converter cable plugs in to the RJ-45 connector and provides a 25-pin D-shell connector for device attachment.

The 64-port asynchronous adapter hardware is no longer available as a supported product. However, if you purchased one of the models (with a 64-port adapter) specified in the following illustration before November 30, 1993, the adapter is still supported.

Supported Models					
7011-220	7011-22W				
7012-320	7012-32E	7012-32H	7012-340	7012-350	7012-360 7012-370
7013-520 7013-55E 7013-58F	7013-52H 7013-55S	7013-530 7013-560	7013-53E 7013-56F	7013-53H 7013-570	7013-540 7013-550 7013-57F 7013-580
7015-930 7015-98E	7015-950 7015-98F	7015-95E	7015-970	7015-97E	7015-97F 7015-980

If you upgraded from the models previously specified to one of the models in the following illustration, your 64-port adapter is still supported.

Suported Models (as Upgrades)			
7011-25E	7011-25F		
7012-380	7012-390	7012-39H	
7013-58H 7015-99E	7013-590 7015-99F	7013-59H 7015-99J	7015-99K

Characteristics

- Standard Micro Channel form factor card.
- Maximum of 38.4Kbps per port (dependent on application and use).
- Up to 2500 feet between controller and concentrator.
- Up to 200 feet from concentrator to device.
- Surge protection circuitry.
- Supports the following interface signals on each port:

  `TxD, RxD, DTR, CTS, RTS, and DCD`
- 8-bit/16-bit Micro Channel slave interface.

Topology

The following figure illustrates the 64-port asynchronous controller with one of four concentrator boxes that can be attached to any one of the controller connectors. The position of the adapter determines the controller connector numbering:

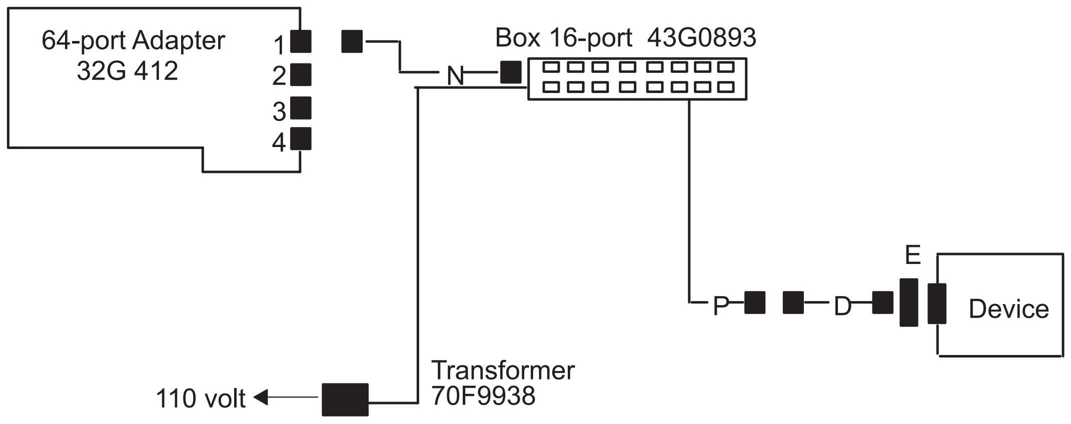


Figure 9. 64–Port Asynchronous Controller

Cables	PN	FC	Length	
			m	ft
D	6323741	#2936	3	10
E	58f2861	#2937	N/A	N/A
N	00G1109	N/A	7.6	25
P	53F3432	#6402	.45	1.5

- Top to bottom 1 through 4, when the card is in the vertical position
- Right to left 1 through 4, when the card is in the horizontal position.

A choice of cables can be attached to any of 16 concentrator box ports. The ports on the concentrator box are labeled 0 through 15 and accept RJ-45 connectors. In order to make the necessary connections to this adapter, the system administrator needs to know the devices and persons assigned to each port.

Listing All Defined 64-Port Asynchronous Adapters

This procedure provides a list of the 64-port asynchronous adapters that are defined in the customized database. A defined device can be either available (configured) or unavailable to the operating system. To obtain a list of defined 64-port adapters, use the System Management Interface Tool (SMIT) to build the **lsdev** command, or issue the **lsdev** command directly from the command line. The list obtained contains the name of the defined 64-port asynchronous adapters, their status (available or defined), location code (which identifies the 64-port asynchronous adapter physical device), and a device description.

> **Note:** The status of a defined and configured device is *available*. The status of a device that is defined but not configured is *defined*.

List 64-Port Asynchronous Adapters Using SMIT

1. To list all defined 64-port asynchronous adapters regardless of whether they are available or not, use the **smit lsddev** fast path.
2. A scrollable list of all defined 64-port asynchronous adapters is displayed.
3. Exit the SMIT interface.

List 64-Port Asynchronous Adapters from the Command Line

Enter the **lsdev** command at the command line as follows:

- To list all 64-port asynchronous adapters, regardless of whether they are available, enter: `lsdev -Cc adapter -t 64p232 -H`
- To list all 64-port asynchronous adapters that are available to the operating system, enter: `lsdev -Cc adapter -t 64p232 -H -S a`
- To list all 64-port asynchronous adapters that are *not* available to the operating system, enter: `lsdev -Cc adapter -t 64p232 -H -S d`

See the **lsdev** command for a detailed description and optional flags.

Transient Voltage Line Protector

The Transient Voltage Line Protector kit for the 16-port concentrator of the 64-port asynchronous adapter, clips off dataline voltage spikes that come from induced noise in a cabling environment. It is possible, in some circumstances, for voltage spikes to cause unrecoverable data interruptions to the port. In some severe cases, spikes may cause permanent damage to the 64-port hardware.

The line protectors are effective in protecting the 16-port concentrators from voltage spikes due to thunderstorm activity and other sources of damaging electromagnetic or electrostatic energy.

> **Attention:** Line protectors must not be used in place of outdoor surge protectors when going out of doors or between buildings.

Transient Voltage Line Protectors Terms and Conditions

The following are the terms and conditions necessary for transient voltage line protectors:

- If the concentrators are damaged during thunderstorms, replace the damaged hardware and install line protectors for all concentrators on the system.
- If ports on the concentrators frequently hang and cannot be reset, replace the bad or damaged hardware, if any, and install line protectors for all concentrators on the system.
- If ports on the concentrators hang and can be reset, it is not necessary to replace the concentrator. The addition of a line protector may help prevent further occurrences of the problem if the hang condition was not caused by bad cable connections.

- If the system is located in a hazardous storm area with medium to long (100 ft - 200 ft) cabling distances between the 16-port concentrators and devices.
- If the system is installed in a highly industrial area with long (150 ft - 200 ft) cabling distances between the 16-port concentrators and devices.

Follow the installation instructions and verify that the concentrator power cord ground is connected to a solid AC ground. Contact your hardware service representative for more information concerning transient voltage line protectors.

64-Port Problem Determination and Troubleshooting

Problems with the 64-port system generally fall into three categories:

- Hung ports which can be reset
- Hardware failure with the controller, concentrator, or devices
- Data integrity.

This procedure provides a list of actions and suggestions for debugging 64-port Asynchronous Adapter problems on the system unit. Use them to record data or correct the problem before placing a service call.

1. Before restarting the system with the **shutdown** command or reset key, check for obvious problems such as a disconnected cable or unplugged device.
2. Do not work on the system without proper electro-static discharge (ESD) protection.
3. Do not replace a 16-port concentrator because of a dead port without first having tried to reset the port through a hardware power cycle (shutdown) or software commands. In many cases, replacing the concentrator corrects the problem only because it performs a power cycle which, in this instance, may be the only action necessary.
4. Do not reconnect cabling without first dissipating possible built-up electrical charges.
5. Keep a repair log for the system unit. Look for repeat problems or patterns to problems which may help isolate the original source of trouble. This log will help isolate problems with a particular run, concentrator, or device and is invaluable in determining the root cause in a problem environment.

Known Problems with Asynchronous Adapters

Non-Shielded Wire

Any wiring scheme which runs several hundred feet is subject to atmospheric noise through magnetic induction. Shielding of cable grounds this noise and prevents its induction to signal wires. Non-shielded wiring subjects the system to port hangs, equipment damage, and erroneous data.

Corrective Action:

Minimum	Transient voltage surge suppression (TVSS) devices should be used to minimize noise and over current situations. A TVSS device should be used at both the device and concentrator end of any port having a higher than normal failure rate. Note that it may become necessary to use TVSS devices on all ports of a concentrator if system environmental problems are serious enough. In most cases, a good contact with ″earth ground″ is ample enough to dissipate high voltage spikes.
Maximum	Replace the wiring to meet specification.

Outdoor Wiring Run and/or Frequent Lightning Damage

Any wiring run that is located outdoors is *very* subject to atmospheric noise or lightning surges through magnetic induction. A direct lightning hit is not necessary to cause system damage.

Corrective Action:

Minimum	Transient voltage surge suppression (TVSS) devices should be used to minimize noise and over current situations. A TVSS device should be used at both the device and concentrator end of any port having an out-door run.
Maximum	All outdoor wiring runs should be inside metal conduit, and the conduit should be grounded.

Concentrator to Device Exceeds 200 ft or 2500 Pico-Farads

Any wiring system that exceeds the rated capacitance is subject to data loss, port hangs, or both.

Corrective Action:

Minimum	By moving the concentrator closer (within 200 ft) to its devices, you can take advantage of the maximum length of the controller-to-concentrator cable (2500 ft) and reduce the concentrator-to-device cable length.
Maximum	If the concentrator is already 2500 ft from the controller, then the 200 ft run will need a booster device.

Thunderstorms and ESD

Lightning strikes pose the greatest hazard to communication equipment circuitry. A lightning strike several miles away can induce voltages in the communication lines that are sufficient to destroy driver/receiver circuits.

Corrective Action:

Cables should be adequately shielded while keeping the cable length to a minimum. Transient voltage surge suppression (TVSS) devices must be used if connecting cables out-of-doors between buildings. This should be done even if cabling is buried in the ground or in conduit.

Chapter 6. 128-Port Asynchronous Subsystem Overview

The 128-port asynchronous adapter subsystem meets the multiuser requirements for workstations in the open system environment. The subsystem can support up to 128 devices and offers higher speeds to the end user. It is an *intelligent* adapter with the ability to offload line protocol processing from the operating system.

The 128-port product consists of an adapter card, up to eight remote asynchronous node (RAN) units, each supporting 16 devices, for a total of 128 devices per adapter. The 128-port asynchronous adapter card resides in the system unit, and is connected by either EIA 422, direct cabling, EIA 422/EIA 232 synchronous modems, or Data Service Unit/Channel Service Units (DSU/CSUs) to the RAN. The number of 128-port asynchronous adapters that can be installed is dependent on the number of slots available in the system with a hardware limit of seven adapters per bus (an optional I/O bus can provide eight additional slots).

To understand the 128-port asynchronous adapter in more detail, consider:
- 128-Port Asynchronous Adapter
- 16-Port Remote Asynchronous Node
- Topology
- RAN Removal
- RAN Termination
- RAN Node Numbers
- The Number of 128-Port Asynchronous Adapters Per System

128-Port Asynchronous Adapter

The 128-port asynchronous adapter is an intelligent dual-channel EIA 422 synchronous board for the operating system. The 128-port asynchronous adapter features the following main components:
- On card microprocessor
- Up to 1 Megabyte of memory
- Two high-speed EIA 422 synchronous lines (or channels) used to communicate with 128-port RAN at data rates of up to 2.458 Mbps. The synchronous lines are also capable of communicating with EIA 422 and EIA 232 synchronous modems.
- Available for Micro Channel, ISA, or PCI bus.
- The Itanium-based platform supports only PCI.

16-Port Remote Asynchronous Node (EIA 232)

The 128-port remote asynchronous node (RAN) is a complete subsystem and contains the following main components:
- 25 MHz 80C186 microprocessor
- 128K of RAM and 16K of EPROM
- Sixteen 16C550 universal asynchronous receiver/transceivers (UARTs) for the 16 EIA 232 asynchronous serial ports
- High-speed synchronous EIA 232 port for communication with the adapter and other RANs.

The RAN receives data packets from the adapter at data rates of up to 1.2 Mbps, and then distributes the data, as appropriate, to the 16 EIA 232 ports. Data received by the EIA 232 ports is similarly divided into packets and sent back to the adapter over the high-speed synchronous line. The EIA 232 ports operate at data rates of up to 57,600 bps.

For local applications, where the work groups are not located more than 300 m (1000 ft) from the system, RANs can be connected to a 128-port asynchronous adapter through 4- or 8-conductor twisted-pair cable (the 8-conductor cable is recommended). With the 8-conductor cable, data is transferred between RANs and the adapter at rates of up to 1.2 Mbps. With 4-wire cable, the maximum synchronous data rate is 460 Kbps.

For more information concerning cable types, lengths, and baud rates, refer to ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″.

16-Port Enhanced Remote Asynchronous Node (EIA 232)

The 128-port remote asynchronous node (RAN) is a complete subsystem and contains the following main components:

- 25 MHz 80C186 microprocessor
- 128K of RAM and 16K of EPROM
- Sixteen 16C550 universal asynchronous receiver/transceivers (UARTs) for the 16 EIA 232 asynchronous serial ports
- High-speed synchronous port for communication with the adapter and other RANs.

The RAN receives data packets from the adapter at data rates of up to 2.458 Mbps, and then distributes the data, as appropriate, to the 16 EIA 232 ports. Data received by the EIA 232 ports is similarly divided into packets and sent back to the adapter over the high-speed synchronous line. The EIA 232 ports operate at data rates of up to 230 Kbps.

Adapter configuration software will autodetect and configure RANs of multiple types on a single line. Line speed will be limited to 1.2 Mbps when connected on the same line as the original 128-port asyncronous node EIA 232.

For local applications, where the work groups are not located more than 300 m (1000 ft) from the system, RANs can be connected to a 128-port asynchronous adapter through 4- or 8-conductor twisted-pair cable (the 8-conductor cable is recommended). With the 8-conductor cable, data is transferred between RANs and the adapter at rates of up to 1.2 Mbps. With 4-wire cable, the maximum synchronous data rate is 460 Kbps.

For more information concerning cable types, lengths, and baud rates, refer to ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″.

16-Port Enhanced Remote Asynchronous Node (EIA 422)

The 128-port remote asynchronous node (RAN) is a complete subsystem and contains the following main components:

- 16 MHz 80C186 microprocessor
- 128K of RAM and 16K of EPROM
- Sixteen 16C550 universal asynchronous receiver/transceivers (UARTs) for the 16 EIA 422 asynchronous serial ports
- High-speed synchronous EIA 232 port for communication with the adapter and other RANs.

The RAN receives data packets from the adapter at data rates of up to 2.458 Mbps, and then distributes the data, as appropriate, to the 16 EIA 422 ports. Data received by the EIA 232 ports is similarly divided into packets and sent back to the adapter over the high-speed synchronous line. The EIA 422 ports operate at data rates of up to 230 Kbps.

Adapter configuration software will autodetect and configure RANs of multiple types on a single line. Line speed will be limited to 1.2 Mbps when connected on the same line as the original 128-port asyncronous node EIA 232.

For local applications, where the work groups are not located more than 300 m (1000 ft) from the system, RANs can be connected to a 128-port asynchronous adapter through 4- or 8-conductor twisted-pair cable (the 8-conductor cable is recommended). With the 8-conductor cable, data is transferred between RANs and the adapter at rates of up to 2.458 Mbps. With 4-wire cable, the maximum synchronous data rate is 460 Kbps.

For more information concerning cable types, lengths, and baud rates, refer to ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″.

Topology

Each 128-port asynchronous adapter has two synchronous lines. Up to four RANs can be connected to a single synchronous line in daisy-chain fashion. That is, the first RAN is connected to one of the adapter's lines, the second RAN is connected to the first RAN, and so on (up to four RANs).

The RANs can be attached to the 128-port adapter with a direct local connection, a synchronous modem connection, or a combination of the two. The following figure shows some of the possible connection types.

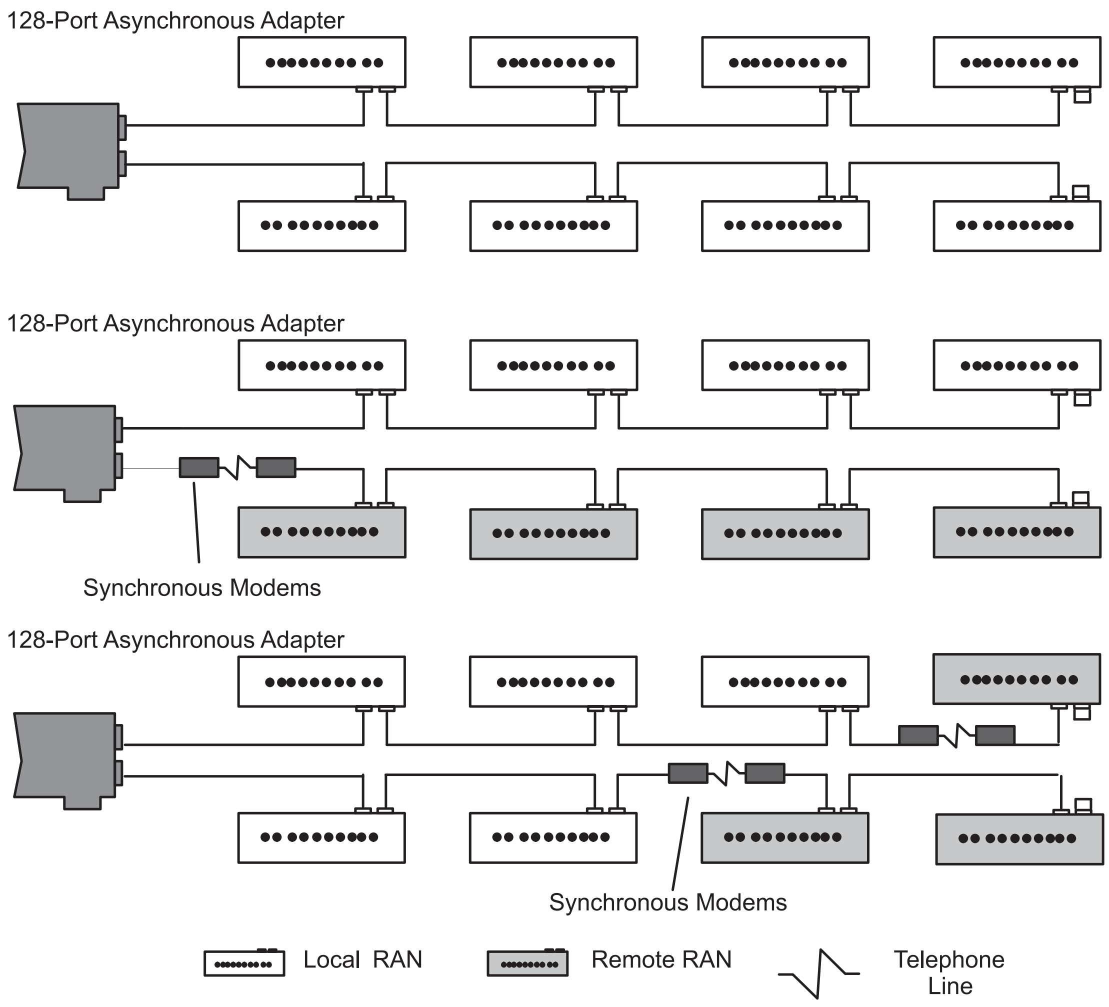

128-Port Asynchronous Adapter
128-Port Asynchronous Adapter
Synchronous Modems
128-Port Asynchronous Adapter
Synchronous Modems
Local RAN
Remote RAN
Telephone Line

Figure 10. Local and Remote Configuration Scenarios

Note: Only a single modem connection is allowed per adapter line.

See the following figure for an example of a scenario in a distributed environment using the 128-port asynchronous adapter.

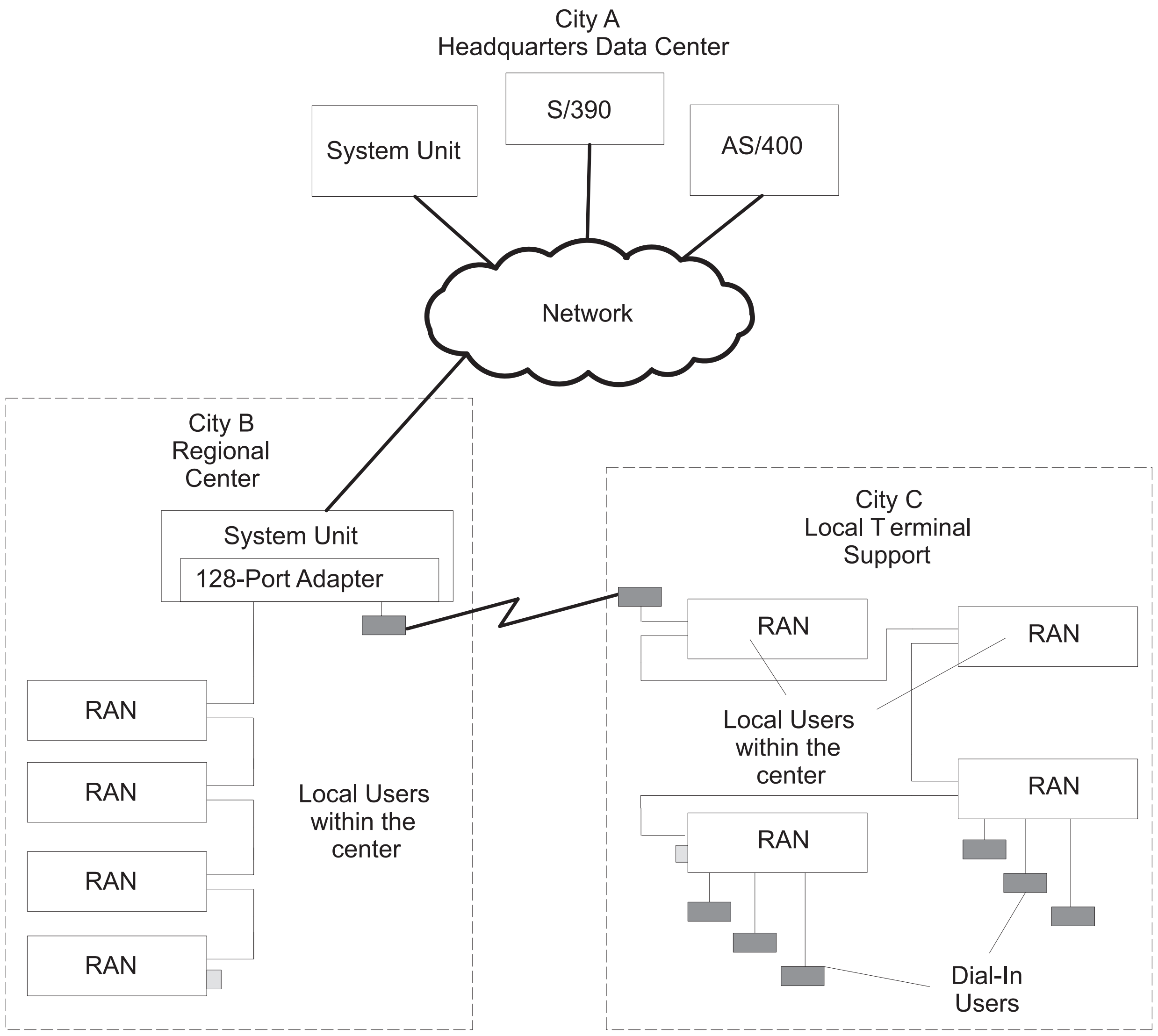


Figure 11. Distributed Environment Solution

- The system unit in City B can support up to 64 terminal/printers for local users within the center.
- City C is connected through synchronous modems to the system unit in City B.
- The RANs in City C provide support for both local and dial-in users.
- The system unit in the regional center at City B is connected by the backbone network using a networking protocol to the Headquarters Data Center in City A.
- All users in Cities B and C can have access to the Headquarters Data Center.

RAN Removal

RANs are assigned physical node numbers (set by the operator during installation, see Setting a RAN Node Number). The node number is used by the adapter to route data to and from a specific RAN. If a particular RAN is turned off or removed from the daisy chain, the 16 devices on that RAN become unavailable to the system, but the rest of the system remains unaffected. Since the RAN's IN and OUT/T ports are of opposite gender, a direct or local RAN can be removed from the middle of a daisy chain by plugging the cables together so that the chain remains unbroken. To remove the last RAN, simply plug the terminator plug into the end of the daisy-chain cable.

RAN Termination

RANs always receive input data via their IN ports and always transmit output data from their OUT/T ports. When multiple RANs are daisy-chained together, data travels in a circular fashion. Thus, if there are four RANs connected to an adapter's synchronous port, data from the adapter to RAN 4 must pass through RAN 1, 2, and 3 before reaching RAN 4. At the same time, data from RAN 1 to the adapter must travel the full circle through RAN 2, 3, and 4 before being returned to the adapter. Refer to the following figure for an example of the data flow.

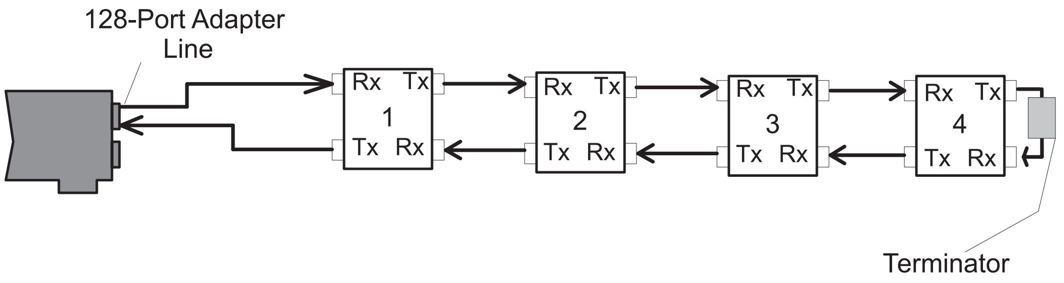


Figure 12. Logical Data Flow in a Daisy-Chain Configuration

To make the loop complete, a terminator plug must be installed on the OUT/T port of the last RAN in the daisy chain. This plug ties all of the OUT/T port's output signals back to their corresponding input signals (TxD to RxD, TxC to RxC, and so on). The RAN's OUT/T port input signals are hard-wired to its IN port output signals, so once any RAN's output data reaches the terminator plug on the last RAN, it is passed back through all of the RANs until it is ultimately received by the adapter. Note that if only one RAN is installed, it is by default the last one and needs to have a terminator plug (IBM PN 43G0926) installed.

RAN Node Numbers

The adapter identifies RANs by their node numbers. Each RAN in a daisy chain must have a unique node number (*1n* through *4n*), which must be set during installation (see ″ Setting a RAN Node Number″). The node numbers must be assigned in ascending order with the lowest number assigned to the RAN closest to the adapter. You can skip node numbers (to facilitate insertion of additional RANs at a later date), as long as the ascending sequence is maintained. The following figure summarizes the node number assignment.

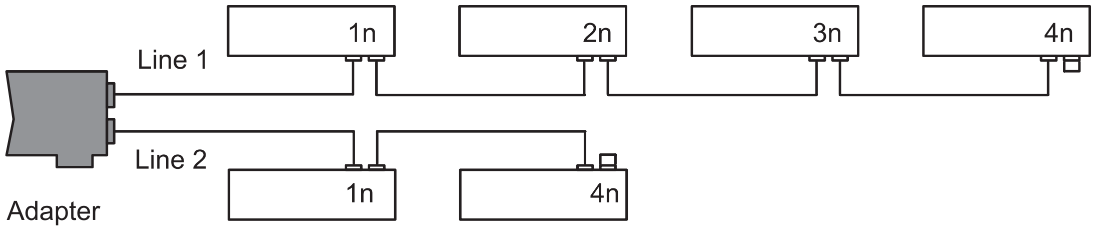


Figure 13. RAN Node Numbers

Number of 128-Port Asynchronous Adapters per System

The number of adapters that can be installed in a system is dependent on the number of available slots. See Product Selection Considerations for more information.

Planning for Your 128-Port Asynchronous Adapter

Successfully installing and configuring a 128-port asynchronous adapter and its associated remote asynchronous node (RAN) requires careful planning. The following discusses planning issues for the 128-port asynchronous adapter subsystem.

A planning worksheet is provided to assist you in gathering information needed for SMIT before beginning the configuration process.

The major areas of configuration for the 128-port asynchronous adapter are:

- RAN configuration
  - Local or direct RAN attachment
  - Remote RAN attachment
- RAN attached device configuration
  - EIA 232 asynchronous terminal configuration
  - EIA 232 asynchronous printer/plotter for a RAN
  - EIA 422 asynchronous terminal configuration printer/plotter

Planning Steps

1. Determine topology for each line based on user requirements and physical locations:
   - How many devices will be supported?
   - What kinds of devices?
   - Are the devices local, remote, or a combination of both?
   - What cables are needed, and can existing cables be used?

   The type and number of devices, their geographical location, and the expected data load associated with the devices will determine the types of cables to be used.

   Refer to ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″ for information on cabling requirements.

   Refer to ″Connecting a RAN to a 128-Port Asynchronous Adapter″ .
2. Select RAN node IDs. Refer to ″Setting a RAN Node Number″ .
3. Determine the values for the SMIT 128-port asynchronous adapter configuration parameters.
   a. Use the ″Line Speed and Line Cable Type Determination Flowchart″ to select the *line speed* and *line cable type*.
   b. To select the *node comm mode*, follow the instructions in ″Determining a RAN's Connection Type and SMIT Node Comm Mode Values″ .

128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios

There are three basic wiring modes for connecting remote asynchronous nodes (RAN) to an adapter or to each other:

- 8-wire direct
- 4-wire direct
- 8-wire synchronous modem

Additionally, there are four types of modular plugs that can be used to connect your asynchronous device to the RAN's RJ-45 10-pin jack:

- 4-pin RJ-11 plugs (EIA 232 only)
- 6-pin RJ-11 plugs
- 8-pin RJ-45 plugs
- 10-pin RJ-45 plugs.

This article discusses the following cables and jacks including their pin-outs and uses.

- Attachment Methods
- Cable Planning
- 128-Port Asynchronous Adapter EIA 422 Connectors
- RAN EIA 422 and Power Connectors
- Local 8-Wire and 4-Wire Direct Cabling Scenario
- Synchronous Modems Cabling Scenario
- Device Cabling Scenario
- 8-Wire Direct Daisy-Chain Cable (NC and NB) Specifications
- 4-Wire Direct Daisy-Chain Cable (ND) Specifications
- 8-Wire Synchronous Modem (NE, NF, NG ad NH) Cable Specifications
- RAN 16-Port Device Cables
- RAN Diagnostics Loopback Plug

Attachment Methods

The following table illustrates three general methods for attaching remote asynchronous nodes (RANs) to the 128-port asynchronous adapter.

Attachment Methods and Their Benefits		
Attachment Method	**Recommended Environment**	**Benefit**
8-wire direct	Moderate to heavy data loads	Maximum performance
4-wire direct	Light data loads*	Reduced cabling cost
Synchronous modems	Light data loads*	Remote location

Note: *Contact your sales representative for information on the 4-wire direct and synchronous modem cables.

Cable Planning

In order to make the necessary connections to the 128-port asynchronous adapter, your setup person needs to know the type of connection and which cables to use. The following cable planning chart shows what cables are used.

Cable Planning	
Cable Letter	**Name/Description**
NC	8-Wire Daisy-Chain cable (9 in.)
NB	8-Wire Daisy-Chain cable (15 ft)
ND	4-Wire Daisy-Chain cable
NE and NF	8-Wire EIA 232 Synchronous Modem cables
NG and NH	8-Wire EIA 422 Synchronous Modem cables
NL	RAN-to-Device EIA 232 cable

Cable Planning	
NM	RAN-to-Modem EIA 232 cable
NK[2]	RJ-45 to DB-25 Converter cable (4 cables in a kit)
D[2]	EIA 232 Asynchronous cable
E[2]	EIA 232 Printer/Terminal Interposer

Notes:

1. Contact your sales representative for information on cable availability.
2. For additional information on cable planning, see ″128-Port Async Controller to Remote Async Node Cables″ in *Adapters, Devices, and Cable Information*.

128-Port Asynchronous Adapter Connectors

The 128-port asynchronous adapter connectors are two 15-pin female connectors that can be found on the back of your machine. The following figure illustrates the EIA 422 channels or lines of the 128-port asynchronous adapter.

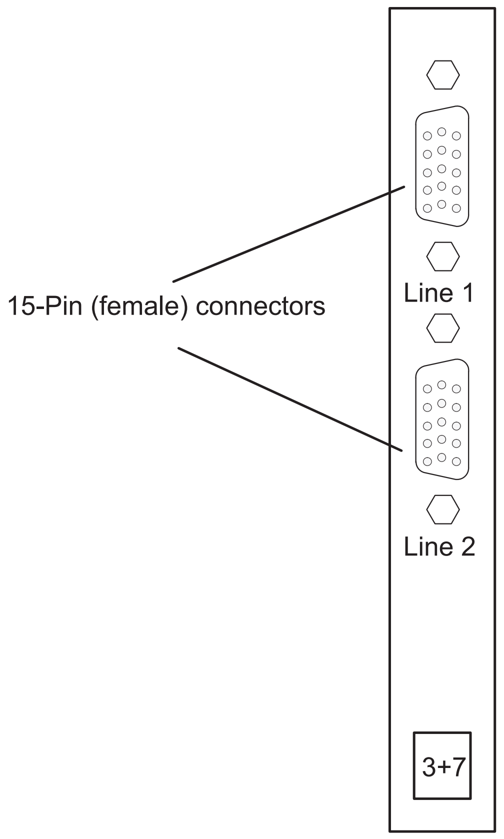


Figure 14. 128-Port Asynchronous Adapter Lines

RAN EIA 422 and Power Connectors

The following figure illustrates the RAN connectors, power connectors, and On/Off switch.

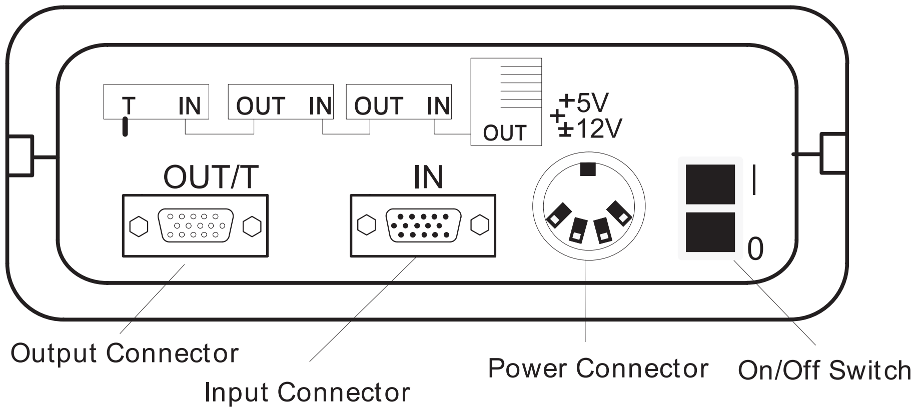


Figure 15. RAN and Power Connectors

These connectors consist of the following:

- Two RAN connectors: one for output and one for input.
- A power connector
- An on/off switch

Local 8-Wire and 4-Wire Direct Cabling Scenario

The following figure shows a typical configuration in which eight RANs are attached to the 128-port asynchronous adapter using both 4-wire and 8-wire cabling.

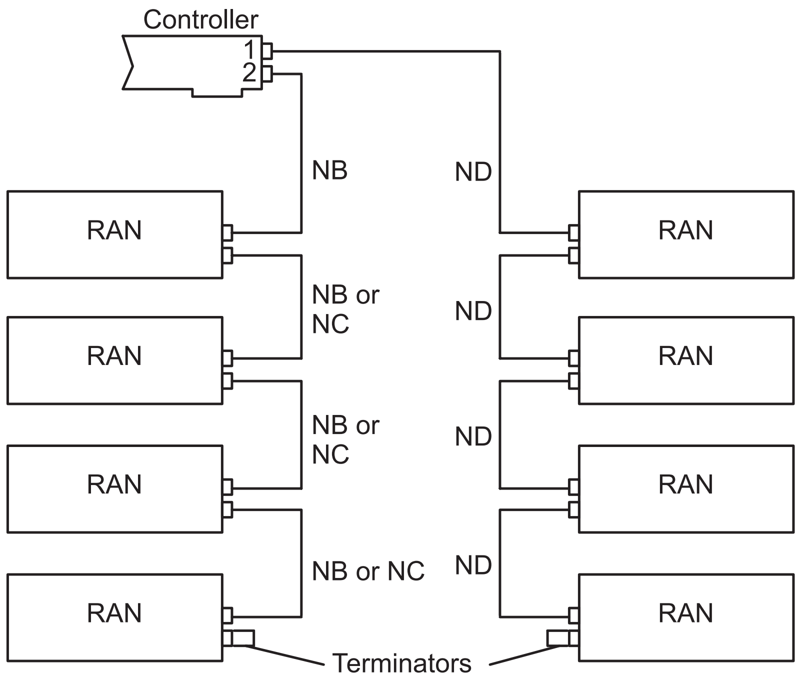


Figure 16. Typical 8-Wire and 4-Wire Topology

Note: 8-wire cabling cannot be used in combination with 4-wire cabling on the same adapter line.

Synchronous Modems Cabling Scenario

The following figure illustrates the use of EIA 232 or EIA 422 synchronous modems in typical 128-port asynchronous adapter configurations. Note that each configuration requires a unique set of customer-supplied cables for modem attachment.

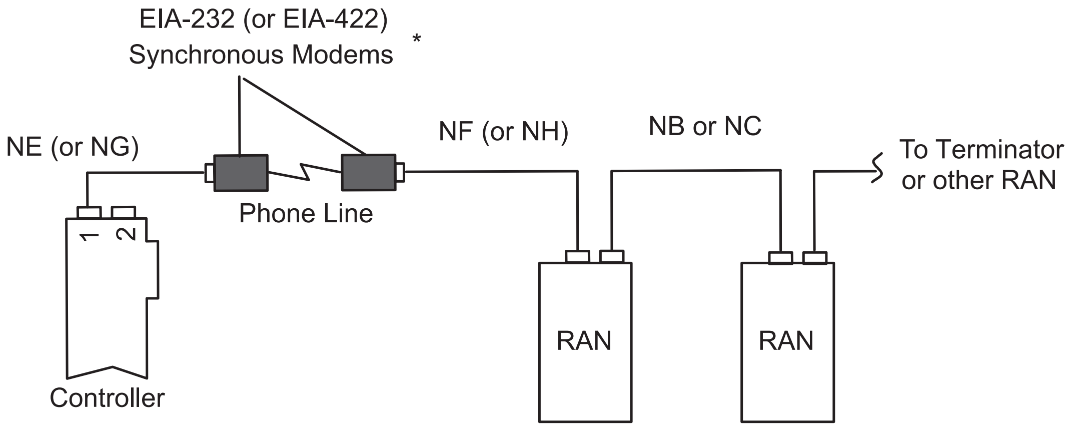


Figure 17. Synchronous Modems Cabling Scenario. A controller is connected to two RANs by a single synchronous line in daisy-chain fashion. The connection between the controller and the first RAN is through a synchronous modem connection, while the connection between the two RANs is a direct local connection. Any subsequent connections, either to a terminator or another RAN, will also be a direct local connection.

Note: * Cables NE, NG, NF, and NH are not available from IBM.

Any combination of 8-wire cabling and synchronous modems can be used to attach remote asynchronous nodes. However, 4-wire cabling cannot be used in combination with synchronous modems on the same adapter line.

Note: Only one pair of synchronous modems can be used per adapter line.

Device Cabling Scenario

A choice of cables can be attached to any of the 16 RAN ports. These ports are labeled 0 through 15 and accept 4-, 6-, 8-, and 10-pin RJ type connectors. The device can be a display, a printer, or a plotter.

8-Wire Direct Daisy-Chain Cable (NC and NB) Specifications

8-wire direct is the standard method for connecting RANs to the 128-port adapter. It provides transmit and receive data signals plus discrete clock signals for transmit and receive data. This permits synchronous data rates of up to 2.458 Mbps, which results in the maximum data throughput under moderate to heavy loads. The following illustration and table show the wire specifications for the 8-wire direct cable.

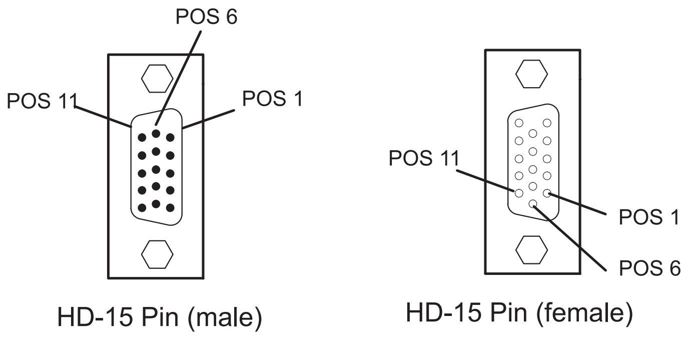


Figure 18. 8–Wire Direct Wiring (Cable NC or NB). Male and female ends of HD-15 pin connectors. Each 15-pin connector has three rows of five pins. Pin positions 1, 6, and 11 on the male connector mirror the same positions on the female connector.

128-Port Adapter or RAN OUT/T Port	Pin	RAN IN Port
RxD –	1	– TxD
RxD +	2	+ TxD
RxC –	4	– TxC
RxC +	5	+ TxC
TxD –	6	– RxD
TxD +	7	+ TxD
TxC –	9	– RxC
TxC +	10	+ RxC
Gnd	Shell	Gnd

The maximum length of a daisy chain depends upon the synchronous data rate used for the 128-port line to which it is connected. The following table lists the maximum ″cumulative″ daisy-chain cable lengths for various line speeds or baud rates.

Baud Rate	Maximum Cable Length (24 AWG Twisted Pair, 12 pF/ft)
up to 460K	2000 ft
up to 1.2M	1000 ft
up to 2.4M	300 ft

Note: The information in this table represents the maximum ″recommended″ supported configurations and are intended for general guidelines only. Configurations above these recommendations can be used, but be aware that, depending on your particular operating environment, loss of data integrity and possible hardware failures may occur.

For example, to run a synchronous line at 1.2 Mbps, the total length of all daisy-chain cables for that synchronous line cannot exceed 1000 ft. Therefore, a single RAN could be placed 1000 ft from the host adapter, or four RANs could be spaced at 250-ft intervals and still operate at 1.2 Mbps.

4-Wire Direct Daisy-Chain Cable (ND) Specifications

4-wire direct wiring can be used to connect RANs where longer synchronous cable runs are necessary. While not as fast as 8-wire connections (the maximum data transfer rate is 460 Kbps), this wiring method is more economical and is sufficient in all but the most demanding high-performance applications (terminal users should see no degradation in performance). In the 4-wire direct-wiring mode, the clock signals are encoded with the receive and transmit data signals (non-return-to-zero inverted encoding at 230 Kbps and FM0 encoding at 460 Kbps), so only two twisted pairs are required.

The cable has four conductors, two twisted-pair, and is shielded on the outside. If built to a length of 300 m (1000 ft) or less, conductors should be 28 AWG (stranded wire) with a capacitance rating of 52 pF/m (16 pF/ft) or less (Belden type 9804 or equivalent). For lengths greater than 300 m (1000 ft), conductors should be 24 AWG (stranded wire) with a capacitance rating of 52 pF/m (16 pF/ft) or less (Belden type 9829 or equivalent).

The following table and figure show the wire specifications for the 4-wire direct cable.

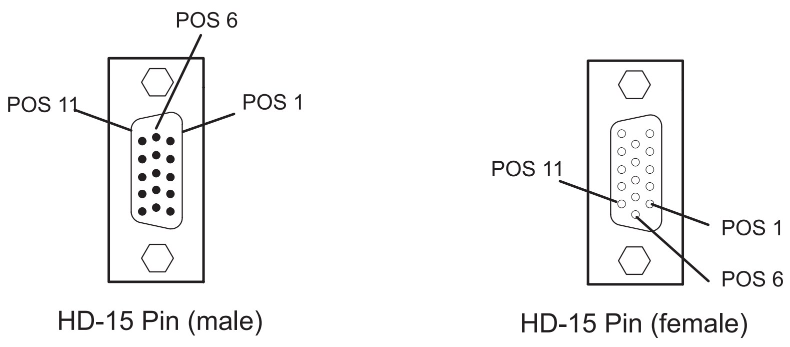


Figure 19. 4–Wire Direct Wiring (Cable NC or NB). Male and female ends of HD-15 pin connectors. Each 15-pin connector has three rows of five pins. Pin positions 1, 6, and 11 on the male connector mirror the same positions on the female connector.

128-Port Adapter or RAN OUT/T Port	Pin	RAN IN Port
RxD –	1	– TxD
RxD +	2	+ TxD
TxD –	6	– RxD
TxD +	7	+ TxD
Gnd	Shell	Gnd

Note: The 4-wire direct daisy-chain cable is a customer-supplied cable.

The 128-port asynchronous adapter supports two adapter line-baud rates in 4-wire direct-attach mode. The following table shows the maximum allowable adapter line length for each supported baud rate. The adapter line length is the actual cable length from the adapter to the last remote asynchronous node in the adapter line.

Baud Rate	Total Adapter Cable Length	
bps	m	ft
230000	400	1350
460000	300	1000

Note: This table assumes no intermediate connectors between remote asynchronous nodes. Each additional connection will decrease the maximum allowable adapter line length by approximately two percent due to increased line capacitance.

8-Wire Synchronous Modem (NE, NF, NG and NH) Cable Specifications

8-wire synchronous modem wiring allows RANs to be installed in remote locations and connected to the adapter using EIA 232 or EIA 422 synchronous modems. The 128-port asynchronous adapter and the RAN are designed so that the synchronous ports can support either EIA 422 or EIA 232 line levels.

EIA 232 Synchronous Modem Cables NE and NF

The following figure and table show the wire specifications for a 128-port asynchronous controller EIA 232 modem cable and system using a DB-25 male connector on the modem end and an HD-15 male connector on the system end.

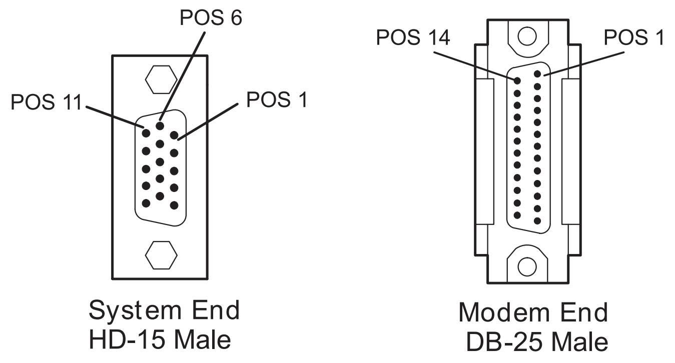


Figure 20. Cable NE

System End HD-15 Male		Modem End DB-25 Male	
Signal Characteristic	Pin	Pin	Signal Characteristic
RxD	1	3	RxD
RxC	4 5*	17	RxC
TxD	6	2	TxD
TxC	9 10*	15	TxC
SG	12	7	SG
FGND	Shell	Shell, 1	FGND
		4	RTS
		6	DSR
		20	DTR

The system end HD-15 connector is made up of three rows of five pins. Pins 4 and 5 are wired together with a 2200pf capacitor from pin 5 to pin 4. Pins 9 and 10 are also wired together with a 2200 pf capacitor from pin 10 to pin 9. In addition, pin 12 is also connected to 9 and 10.

The modem end DB-25 connector is made up of two rows: one with 12 pins, the other with 13 pins. Pins 15 and 7 are connected. Pins 4, 6, and 20 are wired to one another but do not connect to the system end HD-15 connector.

The following figure and table show the wire specifications for a 128-port asynchronous controller EIA 232 modem cable and device using a DB-25 male connector on the modem side and an HD-15 female connector on the device side.

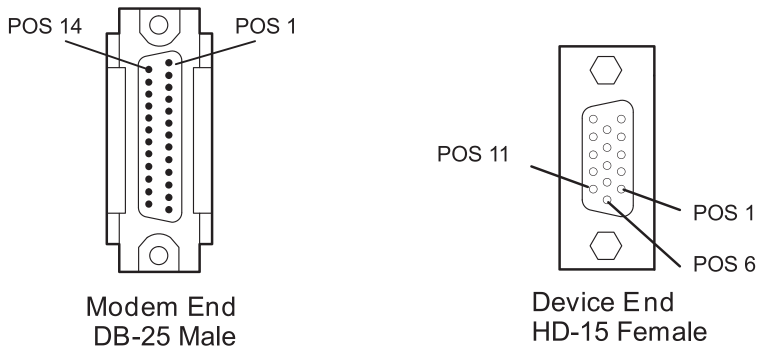


Figure 21. Cable NF

Modem End DB-25 Male		System End HD-15 Male	
Signal Characteristic	**Pin**	**Pin**	**Signal Characteristic**
RxD	3	6	RxD
RxC	17	9	RxC
		10*	
TxD	2	1	TxD
TxC	15	4	TxC
		5	
SG	7	12	SG
FGND	Shell, 1	Shell	FGND
RTS	4		
DSR	6		
DTR	20		

The modem end DB-25 connector is made up of two rows: one with 12 pins, the other with 13 pins. Pins 15 and 7 are connected. Pins 4, 6, and 20 are wired to one another but do not connect to the system end HD-15 connector.

The device end HD-15 connector is made up of three rows of five pins. Pins 9 and 10 are wired together with a 2200pf capacitor from pin 9 to pin 10. Pins 4 and 5 are also wired together with a 2200 pf capacitor from pin 4 to pin 5. In addition, pin 12 is also connected to 4 and 5.

The cable has eight twisted-pair conductors and is shielded on the outside. Cable length can be from 1.8 m (6 ft) to 3.7 m (12 ft). Conductors should be 24 AWG (stranded wire) with a capacitance rating of 41 pF/m (12.5 pF/ft) or less.

The 128-port asynchronous controller supports multiple controller line baud rates in EIA 232 synchronous-modem-attach mode. However, to ensure data integrity, controller line baud rates of 57.6 Kbps or less are recommended.

EIA 422 Synchronous Modem Cables NG and NH

The cable has eight conductors, four twisted-pair, and is shielded on the outside. If built to a length of 300 m (1000 ft) or less, conductors should be 28 AWG (stranded wire) with a capacitance rating of 52 pF/m (16 pF/ft) or less (Belden type 9806 or equivalent). For lengths greater than 300 m (1000 ft), conductors should be 24 AWG (stranded wire) with a capacitance rating of 52 pF/m (16 pF/ft) or less (Belden type 9831 or equivalent).

The following figure and table show the 128-port asynchronous adapter supporting multiple adapter line-baud rates in EIA 422 synchronous modem-attach mode. See the Adapter Line Baud Rate table.

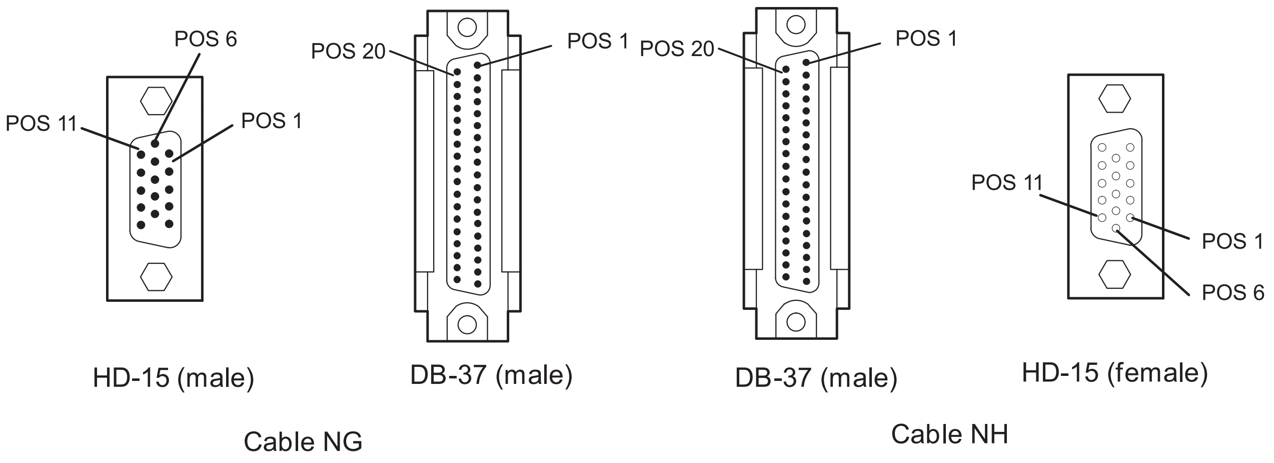


Figure 22. EIA 422 Synchronous Modem Cable (NG and NH)

128-Port Adapter or RAN OUT/T Port		**EIA 422 Synchronous Modem**		**EIA 422 Synchronous Modem**		**RAN IN Port**	
Signal Characteristic	**Pin**	**Pin**	**Signal Characteristic**	**Signal Characteristic**	**Pin**	**Pin**	**Signal Characteristic**
RxD	± 1	6 ±	RD	RD	±6	6±	RxD
	+2	24+			+24	7+	
RxC	±4	±	RT	RT	±8	9±	RxC
	+5	26+			+26	10+	
TxD	±6	4±	SD	SD	±4	1±	TxD
	+7	22+			+22	2+	
TxC	±9	5±	ST	ST	±5	4±	TxC
	+10	23+			+23	5+	
Gnd	12	19	Gnd	Gnd	19	12	Gnd

RAN 16-Port Device Cables

There are four types of modular plugs that can be used with the RAN RJ-45 10-pin jack:

- 4-pin RJ-11 plugs
- 6-pin RJ-11 plugs
- 8-pin RJ-45 plugs
- 10-pin RJ-45 plugs

RAN-to-Device EIA 232 Cable (NL and NM)

The following figure, Cable NL, shows a customer-supplied cable for connecting remote asynchronous node 16-Port EIA 232 to a printer or terminal device.

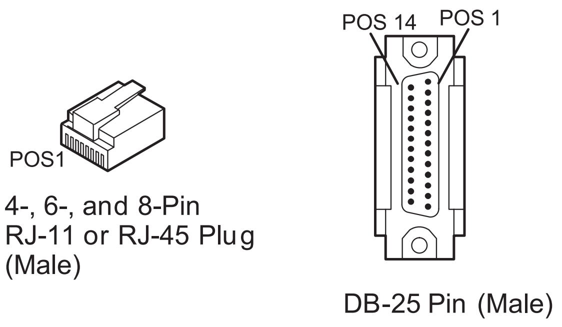


Figure 23. RAN to Printer/Terminal Cable (NL) for 4- and 6-Pin RJ-11 and 8-Pin RJ-45 Plugs

	4-Pin RJ-11	6-Pin RJ-11	8-Pin RJ-45	Terminal/Printer DTE DB-25	
RI					
DSR*			1	20	DTR
RTS		1	2	5	CTS
FGND	Shell 1	2	3	Shell	FGND
TxD	2	3	4	3	RxD
RxD	3	4	5	2	TxD
SG	4	5	6	7	SG
CTS		6	7	4	RTS
DTR			8	6	DSR
DCD*				8**	DCD

*The physical location of DCD and DSR may be interchanged through software control if desired.

**Wired to pin 6.

Cable length can be up to 61 m (200 ft). For 115Kbps the maximum cable length is 80 ft. For 230Kbps the maximum cable length is 40 ft. Use overall foil/braid shielded multiconductor cable with a capacitance rating of 41 pF/m (12.5 pF/ft) or less. Conductors should be 28 AWG (stranded wire). For lengths less than 61 m (200 ft), higher capacitance cable can be used, as long as the total capacitance (including intermediate connectors and cables) does not exceed 2500 pF.

The following table shows cable NL using a 10-pin RJ-45 plug.

Table 2. RAN to Printer/Terminal Cable (NL) for 10-Pin RJ-45 Plug

10-Pin RJ-45		Terminal/Printer DTE DB-25	
RI	1	22	RI
DSR	2	20	DTR
RTS	3	5	CTS
FGND	4	Shell	FGND
TxD	5	3	RxD
RxD	6	2	TxD
SG	7	7	SG
CTS	8	4	RTS
DTR	9	6	DSR

Table 2. RAN to Printer/Terminal Cable (NL) for 10-Pin RJ-45 Plug (continued)

DCD	10*	8**	DCD

* Wired to pin 2.

** Wired to pin 6.

Attention: The receivers and drivers used in most asynchronous communications devices are sensitive to electrostatic discharge (ESD). To reduce the possibility of exposure to ESD, observe the following cabling practices when building or using device cables for attachment to the RAN 16-Port EIA 232:

1. Do not build a cable that has exposed conductors, leads, or pins that could be touched by someone not protected against ESD. Avoid the use of punchdown blocks and patch panels which have exposed terminator/pins. If you use intermediate connectors or cables, be sure to discharge them to ground before plugging them into equipment.
2. Do not run any cables outdoors without having proper transient voltage suppression devices installed.
3. Do not route cables near or around items such as power transformers, high-power switching devices, and refrigeration units.
4. Use shielded cables. All wires should be terminated, not floating. The shield should be connected to shield ground at the remote asynchronous node.

Cable NM is used to connect modems to the RAN's RJ-45 connectors. See the following table for an illustration of the cable NM and EIA 232 using 4- and 6-pin RJ-11 and 8- and 10-pin RJ-45 plugs.

Table 3. RAN to Modem Cable (NM) for 4- and 6–Pin RJ-11, and 8- and 10-Pin RJ-45 Plugs

		4-Pin RJ-11	6-Pin RJ-11	8-Pin RJ-45	10-Pin RJ-45	Cable NM Modem CDE DB-25	
RI	1				1	22	RI
DSR	2			1	2	6	DSR
RTS	3		1	2	3	4	RTS
GND	4	1	2	3	4	Shell	GND
TxD	5	2	3	4	5	2	TxD
RxD	6	3	4	5	6	3	RxD
SG	7	4	5	6	7	7	SG
CTS	8		6	7	8	5	CTS
DTR	9			8	9	20	DTR
DCD	10				10	8	DCD

The following table shows cable NM and EIA 422 using 6-pin RJ-11 and 8- and 10-pin RJ-45 plugs (4-pin connectors are not supported).

Table 4. RAN to Modem Cable (NM) for EIA 422 Plugs

		6-Pin RJ-11	8-Pin RJ-45	10-Pin RJ-45	Cable NM Modem CDE DB-25	
	1			1	22	
	2		1	2	6	
TxA	3	1	2	3	4	TxA
GND	4	2	3	4	Shell	GND

Table 4. RAN to Modem Cable (NM) for EIA 422 Plugs (continued)

TxB	5	3	4	5	2	TxB
RxB	6	4	5	6	3	RxB
SG	7	5	6	7	7	SG
RxA	8	6	7	8	5	RxA
	9		8	9	20	
	10			10	8	

Notes:

1. These cable assemblies are shielded.
2. These cable assemblies and the 64-port RJ-45 to DB-25 converter cable (FC 6402) are not interchangeable.

Cable length can be up to 61 m (200 ft). Use overall foil/braid shielded multiconductor cable with a capacitance rating of 41 pF/m (12.5 pF/ft) or less. Conductors should be 28 AWG (stranded wire). For lengths less than 61 m (200 ft), higher capacitance cable can be used, as long as the total capacitance (including intermediate connectors and cables) does not exceed 2500 pF.

4-, 6-, 8- and 10-Pin Plug Considerations (EIA 232)

The 8- and 10-pin RJ-45 connections are suitable for both terminal and printer attachment. The 4- and 6-pin RJ-11 connections are not recommended for printer attachment.

10-Pin RJ-45 Plugs

The 10-pin RJ-45 plug carries all eight of the EIA 232 signals supported by RANs, plus the two ground lines, Signal Ground (SG) and Chassis Ground (GND). The 10-pin configuration includes the modem control lines Ring Indicator (RI) and Data Carrier Detect (DCD).

8-Pin RJ-45 Plugs

The 8-pin RJ-45 plug supports all of the EIA 232 signals except the modem control lines RI and DCD. It is ideal for use with terminals and printers that require full hardware handshaking, both Data Set Ready (DSR) and Clear To Send (CTS) must be satisfied for data transmission to occur.

Note: The physical location of DCD and DSR may be swapped. The operating system permits software ″rewiring″ of the RJ-45 connectors so that DCD is available in 8-pin configurations. When connecting to Data Terminal Equipment (DTE) devices (such as terminals and printers), you can either use the standard configuration shown above, or the ALTPIN configuration (see ″RJ-45 Connection to Modem Considerations″).

6-Pin RJ-11 Plugs

6-pin RJ-11 plugs can be used in hardware handshaking situations that require only RTS and CTS to be available. This cable is not recommended for printer attachment.

4-Pin RJ-11 Plugs

The 4-pin RJ-11 plug can be used in situations requiring software flow control (XON/XOFF) but no hardware handshaking. No hardware handshake lines are available with this configuration. This is useful for the popular ″three-wire″ connection (TxD, RxD, and SG) used for terminals that support XON/XOFF handshaking. The 4-pin plug is not supported for printers that must have CTS connected to work properly.

When using the 4-pin plug with terminals, DCD sensing must be disabled in the software. This can be done with SMIT, or with the command:

```
chdev -l ttyXX -aforcedcd=enable
```

RJ-45 Connection to Modem Considerations (EIA 232)

Modems are generally equipped with DB-25 connectors, and the NE cable is suitable for this use. Since the modem control lines RI and DCD are on pins 1 and 10, respectively, of the RAN RJ-45 jack, the 10-pin RJ-45 plug is ideal for modems.

Due to the wide availability of RJ-45 8-pin cable, the operating system software incorporates an optional feature called ALTPIN, which swaps the logical functions of DSR with DCD. When ALTPIN is enabled, DCD becomes available on pin 1 of an 8-pin RJ-45 connector (equivalent to pin 2 of a 10-pin connector).

The operating system does not require DSR in modem-control applications, and since almost all of today's modems have auto-answering capability, the RI signal is generally unnecessary.

Cable length can be up to 61 m (200 ft). Use overall foil/braid shielded multiconductor cable with a capacitance rating of 41 pF/m (12.5 pF/ft) or less. Conductors should be 28 AWG (stranded wire). For lengths less than 61 m (200 ft), higher capacitance cable can be used, as long as the total capacitance (including intermediate connectors and cables) does not exceed 2500 pF.

6-, 8-, and 10- Pin Considerations (EIA 422)

The *forcedcd* option must be set to enable. The 422 RAN defaults to *forcedcd=enable*. The 4-pin RJ-11 connection is not supported. Hardware flow control is not supported for EIA 422 connections. XON/XOFF flow control should be used.

RAN Diagnostics Loopback Plug (EIA 232 only)

The loopback plug is used during the RAN diagnostic asynchronous external test to enable an asynchronous port to transmit and receive data. This plug is not supported for EIA 422.

The loopback plug consists of a *single* 10-pin RJ-45 plug wired as follows:

- Pin 3 connected to pins 1 and 8 (RTS to RI and CTS)
- Pin 5 connected to pin 6 (TxD to RxD)
- Pin 9 connected to pins 2 and 10 (DTR to DSR and DCD).

Configuring a Defined 128-Port Asynchronous Adapter

This procedure allows you to make a defined 128-port asynchronous controller available for use by the operating system. Use the System Management Interface Tool (SMIT) to build the **mkdev** command, or issue the **mkdev** command directly from the command line.

Prerequisites

- A 128-port asynchronous controller must be defined to the system.
- You must have root user authority.

Configuring a Defined 128-Port Asynchronous Adapter Using SMIT

1. Use the **smit ttyadapters** fast path to access the **128-Port Async Adapters** screen.
2. Select the adapter type you want to make available. The system scans for the device and configures it.
3. Select **Done** or **Cancel**.

Configure a Defined 128-Port Asynchronous Adapter from the Command Line

Enter the **mkdev** command at the command line, specifying the logical name of the already defined 128-port asynchronous adapter in the **-l** *Name* flag. For example, to make a defined MCA 128-port asynchronous adapter `cxma#` available, enter:

```
mkdev -l cxma#
```

To make a defined 128-port ISA adapter `cxia#` available, enter:

```
mkdev -l cxia#
```

To make a defined 128-port PCI adapter `cxpa#` available, enter:

```
mkdev -l cxpa#
```

See the **mkdev** command for a detailed description and optional flags.

Configuring the 128-Port Asynchronous ISA Adapter

Note: ISA adapters are not supported on the Itanium-based platform.

This procedure describes how to use SMIT to configure a 128-port asynchronous EIA 232 ISA adapter.

Prerequisites

- The **devices.isa.cxia128** software package is installed for the ISA 128-port adapter.
- All RAN connections are local to the system. There is no remote RAN attached using a modem.

Procedure

1. Use the **smit mkdev_isa** fast path.
2. Select **cxia** from the list of adapters displayed on the screen.
3. Select the appropriate bus and press Enter.
4. Select **Bus I/O Address** and set the address to the address of the adapter (set by DIP switches on the adapter). Refer to the *128 Port Asynchronous ISA Adapter Installation Guide* for more information about DIP switches. The remainder of the adapter configuration is done automatically when the system displays **cxia Available**.
5. Exit the SMIT interface.

Listing All Defined 128-Port Asynchronous Adapters

This procedure provides a list of the 128-port asynchronous adapters that are defined in the customized database. A defined device can be either available (defined and configured) or unavailable (defined only) to the operating system. To obtain a list of defined 128-port asynchronous adapters, use the System Manager Interface Tool (SMIT) to build the **lsdev** command, or issue the **lsdev** command directly from the command line. The list obtained contains the name of each defined 128-port asynchronous adapter, its status (*available* or *defined*), its location code (which identifies the 128-port asynchronous adapter physical device), and its device description.

Note: The status of a defined and configured device is *available*. The status of a device that is defined but not configured is *defined*.

List Defined Micro Channel 128-Port Asynchronous Adapters Using SMIT

To list all defined 128-port asynchronous adapters regardless of whether they are available, proceed as follows:

1. Use the **smit lsd128psync** fast path. The system scans for the information and displays it.
2. Exit the SMIT interface.

List Defined 128-Port Asynchronous Adapters from the Command Line

Enter the **lsdev** command at the command line as follows:

- To list all defined 128-port asynchronous adapters regardless of whether they are available, enter:
  ```
  lsdev -C -c adapter -t 128psync -H
  ```
- To list all defined 128-port asynchronous adapters regardless of whether they are available, enter:
  ```
  lsdev -C -c adapter -t cxia128
  ```
- To list all defined 128-port asynchronous adapters that are available to the operating system, enter:
  ```
  lsdev -C -c adapter -t 128psync -H -S a
  ```
- To list all defined 128-port asynchronous adapters that are available to the operating system, enter:
  ```
  lsdev -C -c adapter -t 128psync
  ```
- To list all defined MCA 128-port asynchronous adapters that are *not* available to the operating system, enter:
  ```
  lsdev -C -c adapter -t 128psync -H -S d
  ```
- To list all defined PCI 128-port asynchronous adapters that are *not* available to the operating system, enter:
  ```
  lsdev -C -c adapter -t 4f111b00
  ```

See the **lsdev** command for a detailed description and optional flags.

Changing/Showing Characteristics of a 128-Port Asynchronous Adapter

This article describes the general process of changing or showing the characteristics of a 128-port asynchronous adapter. For a detailed discussion of configuring the 128-port asynchronous adapter and RAN, refer to ″Configuring a RAN with SMIT″ .

Prerequisites

- A 128-port asynchronous adapter must be defined and available to the system.
- You must have root user authority.

Change or Show the Characteristics of a 128-Port Asynchronous Adapter Using SMIT

1. Use the **smit ttyadapters** fast path to access the **128-Port Async Adapters** screen.
2. Select the appropriate adapter (**cxma** for Micro Channel, **cxia** for ISA, or **cxpa** for PCI) to change or see the characteristics.
3. In the displayed dialog fields, supply the values or accept the default values.
4. When you have finished, select **Do**.

   Note: To make the remote asynchronous nodes detectable by the adapter, you must power-cycle the nodes affected by the configuration changes and select the **Configure Devices Added After IPL** option from the SMIT Devices menu.

Show the Characteristics of a 128-Port Asynchronous Adapter from the Command Line

To display the characteristics of a defined and available 128-port asynchronous adapter, enter the **lsattr** command at the command line, specifying the **-l** *Name* and **-E** flags. For example, to display the characteristics of a 128-port asynchronous adapter, enter:

```
lsdev -Cc adapter
```

Find the name of the defined adapter (for example, `cxma0`) in the displayed output from the **lsdev** command, and enter:

```
lsattr -El cxia0
```

```
lsattr -El cxma0
```

```
lsattr -El cxpa0
```

See the **lsattr** and **lsdev** commands for a detailed description and optional flags.

Change the Characteristics of a 128-Port Asynchronous Adapter from the Command Line

To change the characteristics of a defined and available128-port asynchronous adapter, enter the **chdev** command at the command line, specifying the **-l** *Name* flag. In addition, specify (multiple instances of) the **-a** *Attribute=Value* flag to change attribute values, or specify the **-f** *File* flag to retrieve attribute values from a file, or specify a combination of both.

For example, to change a defined 128-port Micro Channel asynchronous adapter named `cxma0` to use an 8-wire cable on adapter line 1, enter:

```
chdev -l cxma0 -a line1_cabletype=8
```

To change a 128-port ISA adapter named `cxia1` to use an 8-wire cable on adapter line 1, enter:

```
chdev -l cxia1 -a line1_cabletype=8
```

For an 128-port PCI adapter named `cxpa0` to use an 8-wire cable on adapter line 1, enter:

```
chdev -l cxpa0 -a line1_cabletype=8
```

Specify the **-P** flag if the configuration changes should be applied only to the database. This allows the database to be changed for a 128-port asynchronous adapter that is in use, so that changes take effect the next time the system is started.

See the **chdev** command for a detailed description and other optional flags.

Consult the following list to find out the *Attribute* name for attributes that can be changed in the **-a** *Attribute=Value* flag for the characteristic to change. For example, specify `-a commode_21c2=direct` to change the communication mode of node 2 for the line 2 cable to direct communication transmission. For further information, see ″Configuring a RAN with SMIT″ .

Characteristic	**Attribute Name**
Line 1 speed	line1_speed
Line 1 cable type	line1_cabletype
Line 1 cable type node 1 comm mode	commode_11c1
Line 1 cable type node 2 comm mode	commode_11c2
Line 1 cable type node 3 comm mode	commode_11c3
Line 1 cable type node 4 comm mode	commode_11c4
Line 2 speed	line2_speed
Line 2 cable type	line2_cabletype
Line 2 cable type node 1 comm mode	commode_21c1

Characteristic	Attribute Name
Line 2 cable type node 2 comm mode	commode_21c2
Line 2 cable type node 3 comm mode	commode_21c3
Line 2 cable type node 4 comm mode	commode_21c4

Removing the 128-Port Asynchronous Adapter from the Command Line

1. To get information specific to the 128-port asynchronous adapter, enter:

   ```
   lsdev -C | grep cx
   ```

   The output of this command may be similar to the following:

   ```
   cxia0    Available   01-02    IBM 128-Port Async, EIA-232 (ISA)
   ```

   This output indicates that adapter `cxia0` is in slot 2.
2. To remove a 128-port asynchronous adapter and its associated RAN, printer devices, and TTYs, enter:

   ```
   rmdev -l cxia0 -R
   ```

   Running this command maintains the adapter definition in the configuration database.

   To remove a 128-port asynchronous adapter and delete the device definition from the Configuration database, enter:

   ```
   rmdev -l cxia -dR
   ```

   This command also removes the associated printer devices, RANs, and TTYs.

Connecting a RAN to a 128-Port Asynchronous Adapter

This procedure allows you to connect a remote asynchronous node (RAN) to a 128-port asynchronous adapter. The connection between the 128-port adapter and the RAN can be as follows:

- Direct attachment
- Remote attachment using synchronous modems (EIA 232 or EIA 422)

Prerequisites

1. A 128-port asynchronous adapter must be defined and available to the system.
2. The power switch on the RAN must be in the Off (O) position.

Connect First Local RAN Directly to 128-Port Asynchronous Adapter

1. Connect the male end of an IBM PN 43G0937, or identical cable to line 1 of the 128-port adapter.

   See ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″ for cable wiring information.
2. Connect the female end of the cable to the RAN port marked IN.
3. Install the terminator plug on the OUT/T port of the RAN.
4. Plug the AC connector of the power supply (US IBM PN 40H3611, or PN 93H7091 for the enhanced RANs) into a standard grounded wall outlet. Plug the DIN connector into the receptacle labeled +5V/+-12V on the RAN.
5. Turn the RAN's power switch to the On (|) position. The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence, and the seven-segment LED display will display `P1` to indicate that the POST sequence has completed successfully.

The display will show AC to indicate that it is online or active and has received instructions from the adapter.

6. Set the node number. Refer to ″Setting a RAN Node Number″ for instructions.
7. Reconfigure the adapter by running *cfgmgr -l cxia0*, *cxma0*, or *cxpa0*.

Connect Remote RAN to 128-Port Asynchronous Adapter via Modems

1. Connect the local EIA 232 or EIA 422 synchronous modem to one of the adapter synchronous lines using an EIA 232 or EIA 422 synchronous modem cable.
2. Connect remote EIA 232 or EIA 422 synchronous modem to the RAN port marked IN using the EIA 232 or EIA 422 synchronous modem cable.

   See ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″ for cable wiring information.
3. Install the terminator plug on the OUT/T port of the RAN.
4. Plug the AC connector of the power supply (IBM PN 40H3611, or PN 93H7091 for the enhanced RANs) into a standard grounded wall outlet. Plug the DIN connector into the receptacle labeled +5V/+-12V on the RAN.
5. Turn the RAN's power switch to the On (|) position. The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence, and the seven-segment LED display will eventually display P1 to indicate that the POST sequence passed.
6. Set the node number. Refer to ″Setting a RAN Node Number″ for instructions.
7. Reconfigure the adapter by running *cfgmgr -l cxia0*, *cxma0*, or *cxpa0*.

Add Subsequent RAN to a System (Daisy Chain)

Refer to ″128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios″ for further information.

Note: Each 128-port asynchronous adapter line is limited to no more than four daisy-chained RANs.

1. Using a daisy-chain cable, connect the male end of the cable to the port marked OUT/T on an existing RAN.
2. Connect the female end of the cable to the IN port on the new RAN.
3. If the new RAN is being placed in the middle of a daisy chain, connect the male end of another daisy-chain cable to the OUT/T of new RAN.
4. Connect the female end of the cable to the IN port to the next RAN.
5. Ensure that a terminator plug is on the OUT/T port of the last RAN in the chain.
6. Plug the AC connector of the power supply (IBM PN 40H3611, or PN 93H7091 for the enhanced RANs) into a standard grounded wall outlet. Plug the DIN connector into the receptacle labeled +5V/+-12V on the RAN.
7. Turn the RAN's power switch to the On (|) position. The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence, and the seven-segment LED display will display P1 to indicate that the POST sequence passed.
8. Set the node number. Refer to ″Setting a RAN Node Number″ for instructions.
9. Reconfigure the adapter by running *cfgmgr -l cxia0*, *cxma0*, or *cxpa0*.

   Note: The new RAN can be added in the middle or at the end of the daisy chain. However, the RAN must be placed so that it can be assigned a free node number. Each RAN must have a unique node number (*1n*-*4n*) assigned in ascending order with the lowest number assigned closest to the adapter.

Configuring a RAN with SMIT

The 128-port asynchronous adapter and RAN are automatically configured during the boot process. With locally or directly attached RANs using IBM cabling, it is not necessary to change any of the default parameters supplied by the operating system for the 128-port asynchronous adapter or RAN.

Because the 128-port subsystem is configured during the boot process, administrators should verify that all RANs are correctly set up and cabled before turning on the power. RAN node numbers should be set in an ascending order with the lowest node number being nearest the 128-port asynchronous adapter card.

This section discusses the configuration of the 128-port asynchronous controller for:
- Local or direct RAN Attachment
- Remote RAN Attachment Using Modems or DSU/CSU
- Determining Line Speed and Line Cable Type Values for a RAN
- Determining a RAN's Connection Type and SMIT Node Comm Mode Values
- Example Node Comm Mode Parameter Determination
- Sample 128-port Asynchronous Adapter Configuration

Prerequisites
1. A 128-port asynchronous adapter must be installed, defined, and available.
2. At least one RAN must be connected. See ″Connecting a RAN to a 128-Port Asynchronous Adapter″.
3. Set the RAN's node ID. Refer to ″Setting a RAN Node Number.
4. Fill out the ″128-Port Asynchronous Adapter Planning Worksheets″.

Configure a Local or Direct RAN Attachment
1. Use the **smit ttyadapters** fast path to access the **128-Port Async Adapter** screen.
2. Select the appropriate Micro Channel 128-port adapter (for example, *cxma0*, *cxma1*, *cxma2*), ISA 128-port adapter (for example, *cxia0*, *cxia1*, *cxia2*), or the PCI port adapter *cxpa0* to make the change.
3. In the displayed dialog fields, supply the values or accept the default values.
4. Select **Do** to implement the new configuration.

Configure a Remote RAN Attachment Using Modem or DSU/CSU
1. Use the **smit ttyadapters** fast path.
2. Select the appropriate 128-port asynchronous adapter from the list displayed on the screen.
3. To view a list of possible values for each field, move the cursor inside the field and select the **List** option.
4. Change the Comm Mode fields to the appropriate values based on the interface type of your modem or DSU/CSU (data service unit/channel service unit). Refer to ″Determining a RAN's Connection Type and SMIT Node Comm Mode Values″.

   Note: Only one pair of modems or DSU/CSU is allowed per adapter line.
5. Select **Do** to implement the new configuration.

Determining Line Speed and Line Cable Type Values for a RAN
A line from the adapter is considered a remote line whenever there is a modem or DSU/CSU used to connect any pair of RANs within that line. The following flowchart can help you determine the values for the SMIT line speed and cable type parameters.

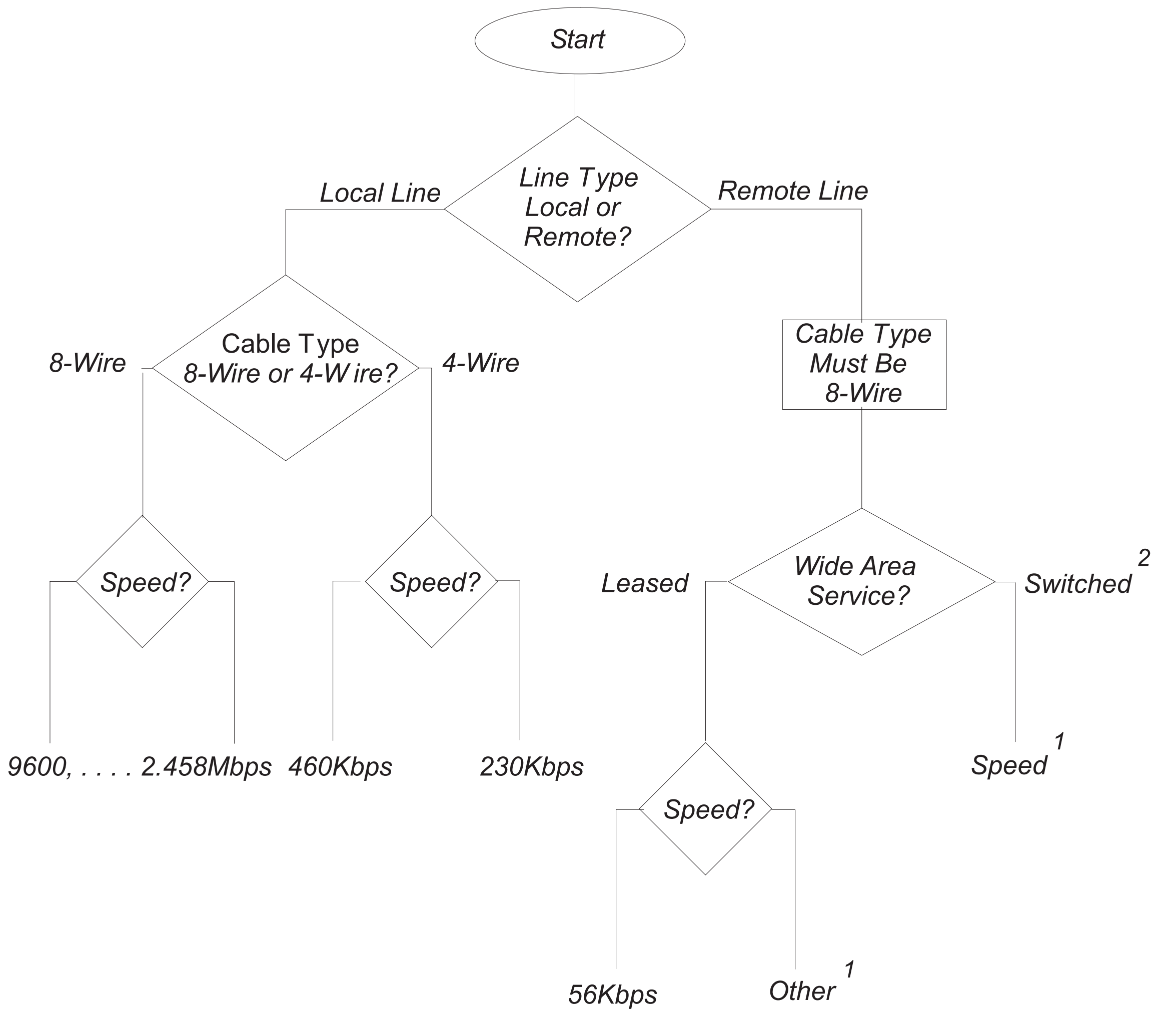


Figure 24. Line Speed and Cable Type Determination Flowchart. If the line type is local, either 8-wire or 4-wire cable can be used. If 8-wire cable is used, only 9600 Kbps and 2.458 Mbps speeds are supported. If a 4-wire cable is used, only 460 Kbps and 230 Kbps speeds are supported. However, if the line type is remote, only 8-wire cable can be used. The speeds available will depend upon your wide area service. If the wide area service is leased, you may use a line speed of 56 Kbps or one that is provided by the carrier service. If the wide area service is switched[1], then the speed will be determined by the carrier service.

Notes:

1. Set the SMIT line speed parameter based on the line speed provided by the carrier service.
2. For initial configuration, the switched line must be manually established and a connection made between the two modems or DSU/CSU in order to bring up the remote RAN.

Determining a RAN’s Connection Type and SMIT Node Comm Mode Values

For a line that consists of 4 RANs, the value for the Node Comm Mode field is defined as shown in the following table.

Node Comm Mode Parameter Selection	
Node Comm Mode	**Description**
`direct`	Selected when no modem is used.
`none`	Selected when the designated node is not used or does not exist.

`232_modem`	Selected when an EIA 232 modem is used to communicate to the RAN at a remote location.
`422_modem`	Selected when an EIA 422 modem is used to communicate to the RAN at a remote location.

Example Node Comm Mode Parameter Determination

The following two examples summarize the Node Comm Mode field for the virtual nodes.

See the following figure for an example of three RANs. The first is directly connected to the asynchronous adapter line 1; the second is connected to the first with an EIA 422 modem, and the third is connected to the asynchronous adapter line 2 with an EIA 232 modem.

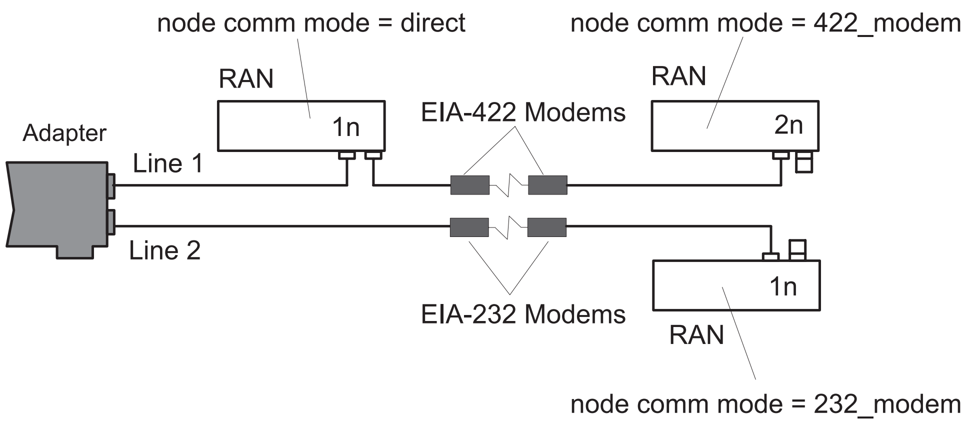


Figure 25. Example Topology for RAN Node Comm Mode Determination

The values of the Node Comm Mode field for this example are:

Example Node Comm Mode Parameter Determination			
Line	**Node**	**Node Comm Version 3.2**	**Mode Value Version 4**
1	1n	`direct`	`direct`
	2n	`422_modem`	`422_modem`
	3n	`422_modem*`	`none`
	4n	`direct*`	`none`
2	1n	`232_modem`	`232_modem`
	2n	`232_modem*`	`none`
	3n	`direct*`	`none`
	4n	`direct*`	`none`
* - Denotes virtual node definition			

Sample 128-Port Asynchronous Adapter Configurations

This section discusses three sample 128-port asynchronous adapter configurations and their corresponding SMIT configurations.

Example A:

Line 1::

- Has four local- or direct-attached RANs with node numbers 1n through 4n.

- Direct attachment at 1.2 Mbps.

Line 2::

- Has no local RANs and two remote RANs with node numbers 1n and 4n.
- Communication is over a leased-line (speed 14.4 Kbps) using EIA 232 synchronous modems.

The following illustration is a diagram of example A.

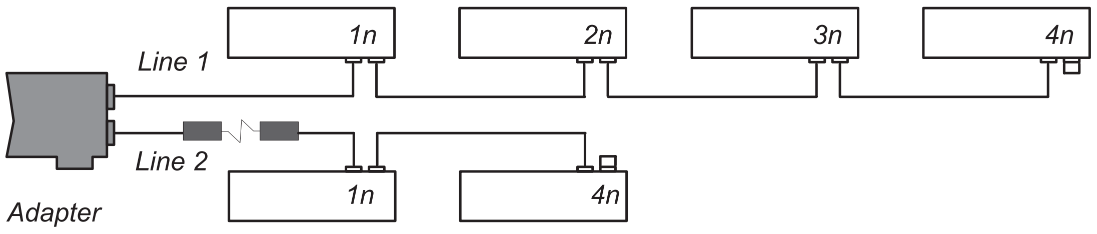


Figure 26. Example A: 128-Port Asynchronous Adapter Configuration

The SMIT fields for the above scenario are shown in the following table.

```
Line 1 Speed                                                  [1.2M]
Line 1 Cable Type                                             [8]
                                                            Version 3.2   Version 4
    Node 1 Comm Mode                                        [direct]      [direct]
    Node 2 Comm Mode                                        [direct]      [direct]
    Node 3 Comm Mode                                        [direct]      [direct]
    Node 4 Comm Mode                                        [direct]      [direct]
Line 2 Speed                                                  [14.4K]
Line 2 Cable Type                                             [8]
                                                            Version 3.2   Version 4
    Node 1 Comm Mode                                        [232_modem]   [232_modem]
    Node 2 Comm Mode                                        [232_modem]   [none]
    Node 3 Comm Mode                                        [direct]      [none]
    Node 4 Comm Mode                                        [direct]      [direct]
```

Example B:

Line 1::

- Has two local and two remote RANs
  - Local RAN node numbers are 1n and 2n.
  - Remote RAN node numbers are 3n and 4n.
- Communication is over a leased-line (speed 56 Kbps) using EIA 232 synchronous modems.

Line 2::

- Has no local RANs and one remote RAN with a node number of 4n.
- Communication is over a leased-line (speed 56 Kbps) using EIA 232 synchronous modems.

The following figure is a diagram of example B.

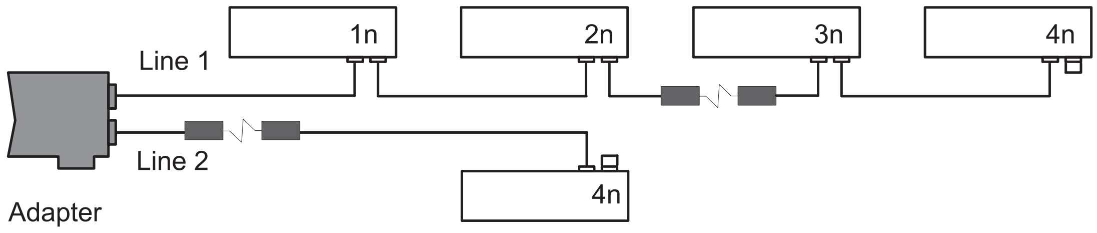


Figure 27. Example B: 128-Port Asynchronous Adapter Configuration

The SMIT fields for the above scenario are shown in the following table.

```
Line 1 Speed                                                      [57.6K]
Line 1 Cable Type                                                 [8]
                                                                Version 3.2   Version 4
    Node 1 Comm Mode                                            [direc]       [direct]
    Node 2 Comm Mode                                            [direct]      [direct]
    Node 3 Comm Mode                                            [232_modem]   [232_modem]
    Node 4 Comm Mode                                            [direct]      [direct]
Line 2 Speed                                                      [57.6K]
Line 2 Cable Type                                                 [8]
                                                                Version 3.2   Version 4
    Node 1 Comm Mode                                            [232_modem]   [none]
    Node 2 Comm Mode                                            [direct]      [none]
    Node 3 Comm Mode                                            [direct]      [none]
    Node 4 Comm Mode                                            [232_modem]   [232_modem]
```

Example C:

Line 1::

- Has no local RANs and one remote RAN with a node number of 1n.
- Communication is over a leased-line (speed 56 Kbps) using EIA 232 synchronous modems.

Line 2::

- Has one local RAN and one remote RAN.
  - Local RAN node number is 1n.
  - Remote RAN node number is 4n.
- Communication is over a leased-line (speed 56 Kbps) using EIA 232 synchronous modems.

The following figure is a diagram of example C.

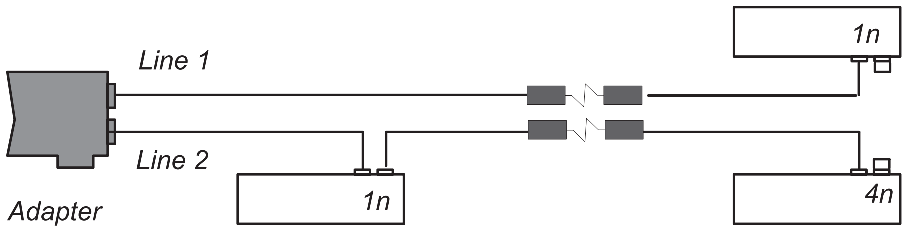


Figure 28. Example C: 128-Port Asynchronous Adapter Configuration

The SMIT fields for the above scenario are shown in the following table.

```
Line 1 Speed                                                [57.6K]
Line 1 Cable Type                                           [8]
                                                          Version 3.2   Version 4
    Node 1 Comm Mode                                      [232_modem]   [232_modem]
    Node 2 Comm Mode                                      [232_modem]   [none]
    Node 3 Comm Mode                                      [direct]      [none]
    Node 4 Comm Mode                                      [direct]      [none]
Line 2 Speed                                                [57.6K]
Line 2 Cable Type                                           [8]
                                                          Version 3.2   Version 4
    Node 1 Comm Mode                                      [direct]      [direct]
    Node 2 Comm Mode                                      [232_modem]   [none]
    Node 3 Comm Mode                                      [direct]      [none]
    Node 4 Comm Mode                                      [232_modem]   [232_modem]
```

Connect Remote RAN to 128-Port Asynchronous Adapter Using Modems

1. Connect the local EIA 232 or EIA 422 synchronous modem to one of the adapter synchronous lines using an EIA 232 or EIA 422 synchronous modem cable.
2. Connect the remote EIA 232 or EIA 422 synchronous modem to the RAN port marked IN using the EIA 232 or EIA 422 synchronous modem cable. See 128-Port Asynchronous Adapter Cable Specifications and Cabling Scenarios for cable wiring information.
3. Install the terminator plug on the OUT/T port of the RAN.
4. Plug the AC connector of the power supply (IBM PN 40H3611 for the standard RAN or PN 93H7091 for the enhanced RAN) into a standard grounded wall outlet. Plug the DIN connector into the receptacle labeled +5V/+-12V on the RAN.
5. Turn the RAN's power switch to the on position (I). The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence. The seven-segment LED display should eventually display `P1` to indicate that the POST sequence passed.
6. Set the node number. Refer to Setting a RAN Node Number for instructions.
7. Reconfigure the adapter by running *cfgmgr -l cxia0, cxma0,* or *cxpa0*.

Setting a RAN Node Number

The adapter identifies RANs by their node numbers. Each RAN in a daisy chain must have a unique node number (*1n* through *4n*), which must be set during installation. The node numbers must be assigned in ascending order with the lowest number assigned to the RAN closest to the 128-port asynchronous adapter. For direct connect configuration, it is permissible to skip node numbers (to facilitate insertion of additional RANs at a later date), as long as the ascending sequence is maintained.

Set the RAN Node Number

1. Turn the RAN on and wait for the power-on self-test (POST) to complete.
2. When `P1` is displayed on the front panel seven-segment LED display, press the Left Arrow button once. The current node number will be displayed, for example, `1n` for node 1.

   Note: Refer to RAN Front Panel and Display Modes for an illustration of the RAN front panel.
3. Press the Right Arrow button to advance the node number through the eight possible settings (*1n* through *8n*). *Only 1n* through *4n* are supported, *5n* through *8n* can be displayed but are *not* supported node numbers.
4. When the desired node number is displayed, press the Left Arrow button again to select the number. The display should now read `Pn` (indicating a pass condition). If there was an error, the display will read `En`.

In the case of duplicate node numbers, the RAN farthest from the host adapter will display `En`, instead of `AC`, when the system is started.

RAN Front Panel and Display Modes

The remote asynchronous node (RAN) is equipped with:

- Ten LED indicators
- Two-digit, seven-segment LED display
- Two push buttons

The LED indicators can be used to reflect the activity of each of the EIA 232 lines and flow control status for a given line. They can also be set to act as a bar graph to show CPU utilization and the activity level of the EIA 422 synchronous line.

The RAN front panel display has several different display modes as indicated by the two-digit, seven-segment display. Pushing the Right or Left arrow push buttons will cycle the display sequentially through the modes.

The following table describes the RAN display modes.

RAN Display Modes

Mode	Mode Name	Description
P1	POST Complete	`P1` appears on the seven-segment display. Power-on self-test is complete, relays are open waiting for connection.
P2	Ping Packet Receive	`P2` appears on the seven-segment display. Indicates that the operating system successfully transmitted a ping packet to RAN. The ping packet contains configuration information used by the RAN (for example baud rate, type of interface).
P3	Transmit Configuration Packet	`P3` is *not* displayed on the seven-segment display. The RAN transmits a packet that contains information about the RAN's physical characteristics. The operating system uses this information to determine which download image to send to the RAN. The RAN does not receive confirmation that the operating system has received the packet.
P4	Image Receive	`P4` appears on the seven-segment display. Download image is being received from the host. The RAN will normally stay at `P4` for a length of time, depending on the synchronous baud rate being used.
AC	Activity	`AC` appears on the seven-segment display. The 10 LEDs turn on sequentially from left to right. The speed of this ″chase light″ display increases with the overall activity level of the RAN.
00-15	Line Monitor	`00-15` appears on the seven-segment display. Modes `00` through `15` correspond to channels 0 through 15. Press the right or left push buttons until the desired channel number appears in the seven-segment display. The LEDs act as line monitor for the selected channel. The first eight LED indicators show the activity of each of the eight EIA 232 signals (TD, RD, RTS, CTS, DSR, DCD, DTR, and RI). The last two LED indicators show when output flow control (OFC) and input flow control (IFC) are active.
En	Error Node	`En` appears on the seven-segment display. Indicates that a valid ping packet was received but the node number in EEPROM is incorrect.
PC	Packet Count	`PC` appears on the seven-segment display. The 10 LEDs show a binary representation of the total number of packets transmitted or received. Pressing both push buttons simultaneously resets the count to 0.

EC	Error Count	`EC` appears on the seven-segment display. The 10 LEDs show a binary representation of the total number of errors counted in the data. Pressing both push buttons simultaneously resets the count to 0.
		When the EC counter is rapidly increasing, this is often an indicator that: 1. The line quality between RANs and/or the adapter is poor. 2. The line between RANs and/or the adapter is open.
PU	Processor Utilization	`PU` appears on the seven-segment display. The 10 LEDs become a bar graph indicating the percentage (0-100%) of the time the RAN microprocessor is being used.
LU	Line Utilization	`LU` appears on the seven-segment display. The 10 LEDs become a bar graph indicating the percentage (0-100%) of the time that the synchronous communications line is being used.
1n, 2n,...,8n *	Node Number	The seven-segment display shows the node number of the RAN.
Ed	POST Error Diagnostics	RAN Error, RAN is Defective. RAN will stay isolated from host.

Note: * Only node numbers *1n* through *4n* are valid. Node numbers *5n* through *8n* are not supported.

Removing or Replacing a RAN

This procedure allows you to remove or replace a remote asynchronous node (RAN) from your system.

If a particular RAN is turned off or removed from the daisy chain, the 16 ports on that RAN become unavailable to the system, leaving the rest of the system unaffected.

Because the RANs IN and OUT/T ports are of opposite gender, a RAN can be removed from the middle of a daisy chain by plugging the cables together so that the chain remains unbroken.

Prerequisites

1. Make sure the power switch on all RANs is in the Off (O) position.
2. Unplug the AC connector of the power supply from the wall outlet. Unplug the DIN connector from the receptacle on the RAN.

Remove a RAN

1. Disconnect the cables from the RAN.
2. Reconnect other RANs (if any) to complete the daisy chain.
3. Ensure that the terminator plugs on the OUT/T port of the last RAN.
4. Reconfigure the adapter using *cfgmgr -l* (*adapter name*).

Replace a RAN

1. Disconnect the cables from the RAN to be removed.
2. Connect the new RAN. Refer to ″Connecting a RAN to a 128-Port Asynchronous Adapter″ .
3. Ensure that the terminator plug is on the OUT/T port of the last RAN.
4. Turn the RAN’s power switch to the On (|) position. The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence, and the seven-segment LED display will display `P1` to indicate that the POST sequence has completed successfully.
5. Set the node number of the new RAN. Refer to ″Setting a RAN Node Number″ .
6. Reconfigure the adapter using *cfgmgr -l* (*adapter name*).

Configuring an Asynchronous Terminal Connected to a RAN

This procedure allows you to define and configure a tty device connected to a 128-port asynchronous adapter RAN.

Prerequisites

1. You must have root user authority.
2. A 128-port asynchronous adapter must be installed, defined, and available.
3. At least one RAN must be connected. See ″Connecting a RAN to a 128-Port Asynchronous Adapter″ .
4. Set the RAN node ID. Refer to ″Setting a RAN Node Number″ .

   Note: Correct cabling must be used so the asynchronous terminal operates correctly on the remote asynchronous node. RJ-45 10-pin cable is recommended to support the full EIA 232 specifications.

Procedure

1. Use the **smit tty** fast path to access the **TTY** menu.
2. Select **Add a TTY**.
3. Select **tty rs232 Asynchronous Terminal** or EIA 422 as appropriate for the RAN type.
4. Make a selection from the available RANs displayed on the screen. If no RANs are displayed or if they are shown in a **defined** state, check the RAN configuration, cabling, and setup again.

   For this example, the following selection was made:

   ```
   sa4 Available 00-03-21 16-Port RAN EIA 232 for 128-Port Adapter
   ```
5. In the displayed dialog fields, you can add or change the field values as needed.
6. When you have finished, select **Do**.

Configuring a Printer or Plotter Connected to a RAN

This procedure allows you to define and configure a printer or plotter attached to a 128-port asynchronous adapter RAN.

Prerequisites

1. You must have root user authority.
2. A 128-port asynchronous adapter must be installed, defined, and available.
3. At least one RAN must be connected. See ″Connecting a RAN to a 128-Port Asynchronous Adapter″ .
4. Set the RAN's node ID. Refer to ″Setting a RAN Node Number″ .

   Note: To work properly with the Print Spooler Subsystem, serial printers and plotters should be attached to the RAN using 8- or 10-pin RJ-45 cabling. See ″RAN 16-Port Device Cables″ for proper cabling information.

Procedure

1. To add a printer/plotter device to a 128-port RAN, use the **smit pdp** fast path to access the **Printer/Plotter Devices** menu.
2. Select **Add a Printer/Plotter**.
3. Select the appropriate printer device from the list of printer and plotter types shown on the screen and press the Enter key. For this example, the following selection was made:

   ```
   osp     Other serial printer
   ```

4. Select **rs232** or RS-422, as appropriate for the RAN type.
5. Make a selection from the available RANs displayed on the screen. If no RANs are displayed or if they are shown in a **defined** state, check the RAN configuration, cabling, and setup again. For this example, the following selection was made:

   ```
   sa4 Available 00-03-21 16-Port RAN EIA-232 for 128-Port Adapter
   ```
6. In the displayed dialog fields, you can add or change the desired printer/plotter device attributes.
7. When you have finished, select **Do**.

The 128-Port Configuration String

The configuration string of the 128-port asynchronous adapter is a multi-byte, hexadecimal character field that represents the communication modes used between the adapter and the RAN. The values shown in this string should be examined whenever the connection modes are in question.

In the following example, the **lsattr** command is used to view the current 128-port adapter settings.

#lsattr -Elcxma0			
bus_io_addr	0x108	Bus memory address	False
bus_mem_addr	0x380000	Bus I/O address	False
config	0008102910081008102900e100e100e100e100eff	N/A	True
line1_speed	57.6K	Line Speed	True
line1_cabletype	8	Cable Type	True
commode_l1c1	direct	Comm Mode	True
commode_l1c2	232_modem	Comm Mode	True
commode_l1c3	direct	Comm Mode	True
commode_l1c4	direct	Comm Mode	True
line2_speed	1.2M	Line Speed	True
line2_cabletype	8	Cable Type	True
commode_l2c1	direct	Comm Mode	True
commode_l2c2	direct	Comm Mode	True
commode_l2c3	direct	Comm Mode	True
commode_l2c4	direct	Comm Mode	True

In the following illustration, the first byte [A] of the config string starts with 00. The next byte [B] 08H shows the communication mode from LINE 1 of the adapter to the first 16 (10H) ports of RAN node 1 as being configured with a line speed of 57.6K internal clocking.

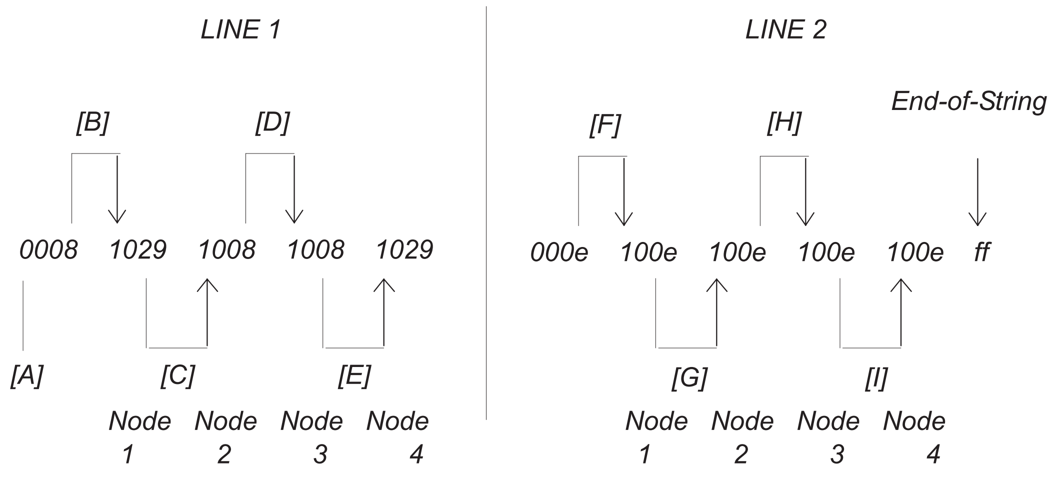


Figure 29. Communication Mode

The communication mode [C] from RAN node 1 to RAN node 2 is 29H (57.6K, external EIA 232 clocking). The communication mode [D] from RAN node 2 to RAN node 3 is 08H (57.6K, internal clocking). The communication mode [E] from RAN node 3 to RAN node 4 is 08H (57.6K, internal clocking). See the following figure, Configuration String Data.

Configuration String Data		
Mode	**Bit Rate**	**Clocking**
00 (00H)	115K	8-wire, internal clock
01 (01H)	230K	4-wire, self clocked
02 (02H)	460K	4-wire, self clocked
03 (03H)	2400	8-wire, internal clock
04 (04H)	4800	8-wire, internal clock
05 (05H)	9600	8-wire, internal clock
06 (06H)	29.2K	8-wire, internal clock
07 (07H)	38.4K	8-wire, internal clock
08 (08H)	57.6K	8-wire, internal clock
09 (09H)	76.8K	8-wire, internal clock
10 (0AH)	115K	8-wire, internal clock
11 (0BH)	230K	8-wire, internal clock
12 (0CH)	460K	8-wire, internal clock
13 (0DH)	920K	8-wire, internal clock
14 (0EH)	1.2M	8-wire, internal clock
15 (0FH)	2400	8-wire, internal clock (EIA 422)
16 (10H)	4800	8-wire, internal clock (EIA 422)
17 (11H)	9600	8-wire, internal clock (EIA 422)
18 (12H)	19.2K	8-wire, internal clock (EIA 422)
19 (13H)	38.4K	8-wire, internal clock (EIA 422)

Configuration String Data		
20 (14H)	57.6K	8-wire, internal clock (EIA 422)
21 (15H)	76.8K	8-wire, internal clock (EIA 422)
22 (16H)	115K	8-wire, internal clock (EIA 422)
23 (17H)	230K	8-wire, internal clock (EIA 422)
24 (18H)	460K	8-wire, internal clock (EIA 422)
25 (19H)	920K	8-wire, internal clock (EIA 422)
26 (1AH)	1.2M	8-wire, internal clock (EIA 422)
33 (21H)	14000	8-wire, internal clock (EIA 422)
35 (23H)	2400	8-wire, internal clock (EIA 232)
36 (24H)	4800	8-wire, internal clock (EIA 232)
37 (25H)	9600	8-wire, internal clock (EIA 232)
38 (26H)	14000	8-wire, internal clock (EIA 232)
39 (27H)	19.2K	8-wire, internal clock (EIA 232)
40 (28H)	38.4K	8-wire, internal clock (EIA 232)
41 (29H)	57.6K	8-wire, internal clock (EIA 232)
42 (2AH)	64000	8-wire, internal clock (EIA 232)
43 (2BH)	76.8K	8-wire, internal clock (EIA 232)

LINE 2 entries of the config string also start with 00 from the adapter. The communication mode shown as [F] from the adapter to the first 16 (10H) ports of RAN node 1 is 0EH (1.2M, internal clocking). [G], [H], and [I] entries contain the same information as [F]. A final entry of **ff** indicates the end of this string.

Running the Remote Asynchronous Node Diagnostics

This procedure allows you to run diagnostics on your remote asynchronous node (RAN) in one of two modes:

Front Panel Mode	Diagnostics are run from the front panel of the RAN.
Video Mode	Diagnostics are run from a terminal attached to the RAN.

The diagnostics routines can test the following aspects of the RAN:

- Front panel
- Memory
- Memory/direct memory access (DMA)
- Asynchronous internal
- Asynchronous external
- Sync EIA 422/DMA
- Sync EIA 232/DMA
- EEPROM
- Watchdog timer

Run Diagnostics from the Front Panel

1. Disconnect the RAN from the adapter and other RANs before running diagnostic tests. Make sure the RAN is terminated.

   Note: Refer to ″Removing or Replacing a RAN″
2. Turn the power switch to the Off (O) position.
3. Press and hold the Right Arrow button while turning the power switch to the On (|) position.
4. The front panel display window shows `P0`. Then, starting with the leftmost LED and proceeding to the right, all LEDs must turn on one at a time until all LEDs are on. All the LEDs then turn off. At this point, release the Right Arrow button. The number 1 with a blinking ″.″ in the lower-right corner of the window will be displayed. The `1` refers to TEST 1, the blinking ″.″ in the lower-right corner of the window indicates that TEST 1 is currently selected.
5. Use the right arrow button to cycle through the possible test numbers 1-9.
6. When the desired test number is displayed in the window, press the Left Arrow button to start the test. A solid ″.″ will display in the lower-left corner of the window to indicate the test is currently running.
7. To stop a test, press the Left Arrow button once. The status of the test will display in the window. A `Pn` indicates the test passed successfully. An `Fn` indicates a failure of test *n*.
8. To exit diagnostic test mode and return to the initialized state (P1), run TEST 9, the Watchdog timer test. This test ends by resetting the RAN and running the POST sequence.

Run Diagnostics from a Terminal

1. Disconnect the RAN from the host adapter and other RANs before running diagnostic tests. Make sure the RAN is terminated properly using wrap plugs.
2. Turn the RAN off by switching the power switch to the Off (O) position.
3. Connect a serial terminal to either port 0 or port 15. Set the terminal's communications parameters to VT100 emulation, 9600, 8 data bits, no parity, and 1 stop bit.
4. Turn the RAN's power switch to the On (|) position. The lights on the front panel should flash as the RAN executes its power-on self-test (POST) sequence, and the seven-segment LED display will display `P1` to indicate that the POST sequence passed.
5. While `P1` is displayed on the front panel, press the letter V (either uppercase or lowercase) on your terminal.

   Note: Some terminals (including the Wyse 60) will not transmit data if their clear to send (CTS) line is LOW. They will only transmit if CTS is HIGH or floating. When using a fully configured null modem cable, the terminal's CTS line is connected to the RAN's request to send (RTS) line, which is held LOW by the RAN until `AC` is displayed. This can prevent the terminal from communicating with the RAN when attempting to run diagnostics.

   If you are using such a terminal, use a cable that does not have CTS connected at the terminal's end.
6. Enter test selection or command.

Diagnostics Menu Options Descriptions

1-9	Runs individual tests as shown on the menu. Refer to ″Diagnostic Test Descriptions″ for further information.
A	Runs all tests except test 9 (Watchdog timer test). The screen will display the number of passes run for each test. The test will halt if an error is detected if `Stop on 1st` is set to Y.
N	Runs the test continuously, keeping count of the errors detected. This must be selected prior to running the test.
Esc key	Cancels test execution at any time during execution. The current pass of the test will be completed and control will be returned to the terminal.
B	Toggles the bell option. When set to `ON`, the terminal beeps each time an error is encountered.

D	Allows you to dump 256 bytes of the RAN's memory to the screen beginning with a specified address (you will be prompted for the starting segment address).
S	Allows you to view or change the RAN's node number. The node number is written in three different EEPROM locations to ensure correctness of the number.
T	Toggles the loopback mode for the EIA 422 and EIA 232 tests (tests 6 and 7) between INTERNAL and EXTERNAL. When set to INTERNAL, the signals are looped back internally by the relays in the RAN. When set to EXTERNAL, a standard 8-wire daisy-chain cable must be installed between the RAN's IN port and its OUT/T port.

Diagnostic Test Descriptions

TEST 1: Front Panel Test	This test alternately turns on and off at one-second intervals all 10 LED indicators and all segments (plus the decimal point) of both seven-segment displays to verify proper operation of the front panel indicators. Since the indicators are write-only, the operator must visually verify success or failure of the test.
TEST 2: Memory Test	Each pass of this test performs a pattern test and an address tag test of DRAM. The byte pattern is incremented for each pass and is displayed on LEDs 0-7. The pattern is written to 32KB, beginning at 08000H. The pattern is written again to 32KB beginning at 10000H. The two 32KB blocks are compared to determine the pass or fail status of the test. The address tag test writes 32KB beginning at address 08000H. Address 08000H will be written with 0H, address 08001h will be written with 1h, etc. This same tag pattern will be written beginning at address 10000H. The two 32KB blocks are compared to determine pass or fail status.
TEST 3: Memory/DMA Test	The DMA tests DMA transfers from one memory location to another. This test may be run without being connected to the RAN. The byte pattern is incremented for each pass and is displayed on LEDs 0-7. The pattern is written to 32KB beginning at 08000H. DMA0 is used to move the 32 KB from 08000H to 10000H. When the move is complete, DMA0 interrupts the CPU and the two 32KB blocks are compared to determine pass or fail status. If the DMA transfer is not completed within 2 seconds, a time-out error causes the test to fail.

TEST 4: Asynchronous Internal Test

This test checks the asynchronous ports hardware. The test puts the port in local loopback mode. The four output signals (DTR, RTS, OUT1, and OUT2) are looped back within the UART chip to the four input signal lines (CTS, DSR, DCD, and RI). The following steps are completed:

- The output and input signals are checked for high and low conditions.
- The ports are initialized to 9600 baud, 8 data bits, 1 stop bit, and no parity.
- Data is transmitted and received at the same port.
- Received data is compared to the transmitted data.
- An interrupt is generated and checked for each port.

This test functions differently depending on whether it is run in video mode or front panel mode.

- **Front Panel Mode**

  In front panel mode, all 16 asynchronous ports of the RAN are checked in each pass. To begin the test, press the Left Arrow button while the number `4` is displayed in the seven-segment LED display. The left-hand digit of the seven-segment display shows a pass count. The first five LEDs (TD, RD, RTS, CTS, and DSR) give a binary representation of the number of the port under test (TD=LSB; DSR=MSB). The test will be repeated until a failure occurs, or until the Left Arrow button is pressed. At this point either `P4` (pass) or `F4` (fail) will be displayed. Press the Left Arrow button now to restart the test or the Right Arrow to advance to TEST 5.
- **Video Mode**

  In video mode, 15 asynchronous ports (1-15 if the terminal is connected to Port 0, or 0-14 if the terminal is connected to Port 15) are tested in each pass. In addition to the number of passes and errors, the display shows the number of each port under test, and its pass/fail status. A failure will also result in a beep from the terminal.

TEST 5: Asynchronous External Test

This test is used to check the asynchronous ports and their associated driver and receiver circuits. Loopback plugs are required. See ″RAN Diagnostics Loopback Plug″ for more information. These plugs enable one asynchronous port to transmit and receive data. The following steps are completed:

- The two control signals (DTR and RTS) are used to test the four input signals (DSR, DCD, CTS, and RI) on the same port.
- The port is tested for transmit and receive data:
  - DTR is looped back to DCD and DSR.
  - RTS is looped back to CTS and RI.
  - The port is initialized to 9600 baud, 8 data bits, 1 stop bit, and no parity.
  - Data is transmitted out of the port and received by the same port.
  - Received data is compared to the transmitted data.

This test functions differently depending on whether it is run in video mode or front panel mode.

- **Front Panel Mode**

  This mode checks one port at a time. Only one loopback plug is needed. Pressing the Left Arrow button once will display the current port number to be tested. Pressing the Left Arrow button repeatedly causes the port number to be incremented through all 16 ports. The test will be repeated until a failure occurs or until the Left Arrow button is pressed again. At this point, either `P5` (pass) or `F5` (fail) will be displayed. Once the test is halted, pressing the Left Arrow button again increments the port number as before. To advance to TEST 6, press the Right Arrow button while `P5` or `F5` is shown.
- **Video Mode**

  In video mode, 15 asynchronous ports are tested in each pass. For repeated testing, 15 loopback plugs are necessary. In addition to the number of passes and errors, the display shows the number of each port under test, and its pass/fail status (a failure will beep the terminal).

  To test a single port in video mode, plug the loopback plug in the port to be tested and set `Stop on first error` to `YES`. If the port passes, 14 errors will be counted (for 14 ports with no loopback plugs). If the port fails, 15 errors will be reported.

For repeated testing of a single port, front panel mode is recommended.

TEST 6: Sync EIA 422/DMA Test

This test uses DMA0, DMA1, and the SCC 8530 sync port to transmit, receive, and verify data using the EIA-422 circuitry. The byte pattern is incremented for each pass and is displayed on LEDs 0-7. The pattern is written to a block of memory beginning at address 08000H.

DMA0 (Rx) and the sync port are used to move the block of data from 08000H to 10000H. The test checks three modes of synchronous data transmission: 8-wire, 1.2 Mbps (32K-byte blocks); 4-wire NRZI, 230 Kbps (8K-byte blocks); and 4-wire FMO, 460 Kbps (16-byte blocks).

When the move is complete, the sync port generates an interrupt and the two blocks are compared to determine pass or fail status. If the DMA transfer using the sync port is not completed within 2 seconds, a time-out error causes the test to fail.

- **Front Panel Mode**

  While `6` is shown in the seven-segment LED display, press the Left Arrow button to begin the test. The left-hand digit will now show the number of sync port receive interrupts and the right-hand digit displays the number of transmit interrupts. This test always runs in internal mode (*no* loopback plug is required) and checks the port in three modes of synchronous transmission. The test halts when an error is detected (`F6` will be displayed) or when the Left Arrow button is pressed again (`P6` will be displayed). At this point pressing the Left Arrow button again restarts the test; pressing the Right Arrow button advances to TEST 7.
- **Video Mode**

  This test can be run with internal or external loopback mode. The current mode is indicated at the end of the `<T>oggle` menu option line. To select between internal and external mode, press T. In internal mode, the port is tested in three modes of synchronous transmission. In external mode, only one transmission method is checked (8-wire, 1.2 Mbps, 32 Kbyte blocks). External mode requires a standard 8-wire daisy-chain cable to be installed between the concentrator's IN port and its OUT/T port.

TEST 7: Sync EIA 232/DMA Test

This test uses DMA0, DMA1, and the SCC 8530 sync port to transmit, receive, and verify data using the EIA-232 circuitry. The byte pattern is incremented for each pass and is displayed on LEDs 0-7. The pattern is written to 512 bytes beginning at address 08000H. DMA1 (Tx), DMA0 (Rx), and the sync port are used to move the 512 bytes of data from 08000H to 10000H at an SDLC rate of 19200 baud. When the move is complete, the sync port generates an interrupt, and the two 512-byte blocks are compared to determine pass or fail status.

If the DMA transfer using the sync port is not completed within 2 seconds, a time-out error causes the test to fail. An audible relay click will be heard upon entering and exiting this test. In video mode, this test can be run with either internal or external loopback by pressing T on the terminal. The external mode requires a standard 8-wire daisy-chain cable to be installed between the RAN's IN port and OUT/T port.

TEST 8: EEPROM Test

This test generates a checksum of the EPROM contents and compares it with the checksum stored in the EEPROM. If the checksums match, the test passes. This does not write to the EEPROM (EEPROM write operations only occur when a new node number is set).

TEST 9: Watchdog Timer Test

This test checks out the watchdog timer. This is a hardware feature that is used to ensure system reliability. When the watchdog timer is loaded and enabled, the timer begins counting down. It is up to the system to keep reloading the timer to prevent it from expiring. If the timer expires, the hardware forces the CPU into a reset state and the POST sequence is started (exactly as if the concentrator is turned off and then on again). Normal test execution allows the timer to expire and force POST execution. The test fails after 1 second if the timer has not expired.

Monitoring the 128-Port Asynchronous Subsystem

The mon-cxma program provides the following capabilities:

- Displays adapter line speeds and wire schemes.
- Displays RAN status, node numbers, and attachment topology.
- Realtime monitoring of synchronous packets transmitted to and received from the RAN.
- Shows whether synchronous packets are being received from the RAN.
- Displays port EIA 232 signal activity through a simulated RAN front panel.
- Displays port input, output, and control modes.
- Displays port and RAN input and output flow control activity.
- Displays RAN status condition as active (AC) or down (DN).
- Allows loopback testing of individual ports on attached RAN.
- Displays 128-port adapter VPD (Vital Product Data).
- Allows screen contents to be dumped to a log file.

Prerequisite Information

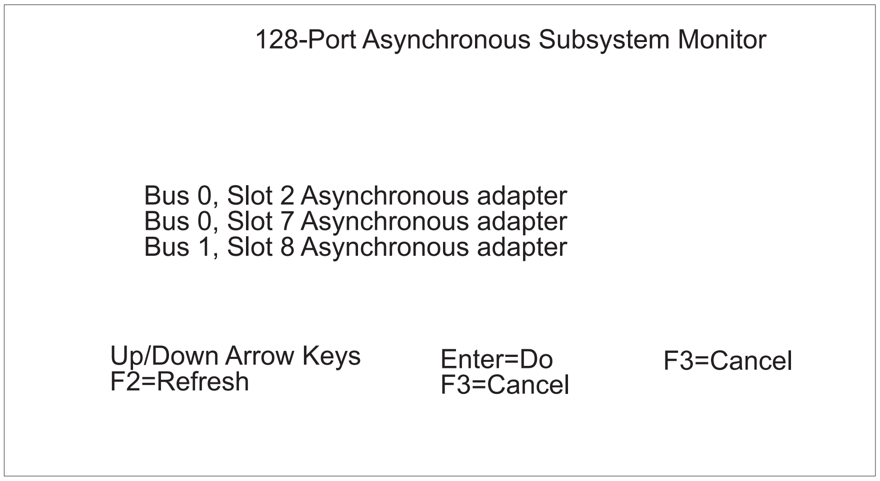
- Root user authority is required.
- 128-port asynchronous adapter software must be installed on the system.
- At least one128-port asynchronous adapter must be installed, configured, and available on the system.

Running the Monitor from SMIT

Use the following procedure to execute the mon-cxma program from SMIT.

Procedure

1. Use the **smit ttyadapters** fast path to access the **128-Port Asynchronous Adapter** menu.
2. Select **Monitor Async Adapters.**
3. Select the appropriate 128-port asynchronous adapter from the SMIT panel. In the following example, three asynchronous adapters are installed in the system; one of which is located on a different bus.



```
                    128-Port Asynchronous Subsystem Monitor



          Bus 0, Slot 2 Asynchronous adapter
          Bus 0, Slot 7 Asynchronous adapter
          Bus 1, Slot 8 Asynchronous adapter


        Up/Down Arrow Keys        Enter=Do          F3=Cancel
        F2=Refresh                F3=Cancel
```

Figure 30. SMIT Panel Example

4. For this example, select **Bus 0, Slot 2 Asynchronous adapter**. This selection takes you to the **ACTIVE CONFIGURATION** screen.

5. The **ACTIVE CONFIGURATION** screen is used as the main panel for the monitor program and displays the following 128-port subsystem information:
   - Bus and slot numbers.
   - Line topology.
   - RAN node numbers.
   - Current RAN status.

   You can also view Vital Product Data (VPD) and adapter status from this panel by pressing the appropriate function key (F6 or F7). A screen image can be saved to the monitor log file, **/tmp/mon-cxma.log**, by pressing the F8 (Image) key.

   Vital Product Data (VPD) is a collection of device information such as part numbers, serial numbers, and engineering change levels. This information is used by service personnel to determine compatibility of hardware components. The data is collected from the 128-port asynchronous subsystem and automatically added to the system configuration. Service representatives may find it necessary to view this information before making upgrade or replacement decisions concerning the 128-port subsystem.

   The following illustration displays the main screen for the monitor program.

   On this screen, RANs are represented by [AC] if active, or (-) if inactive (turned off, defective). In this

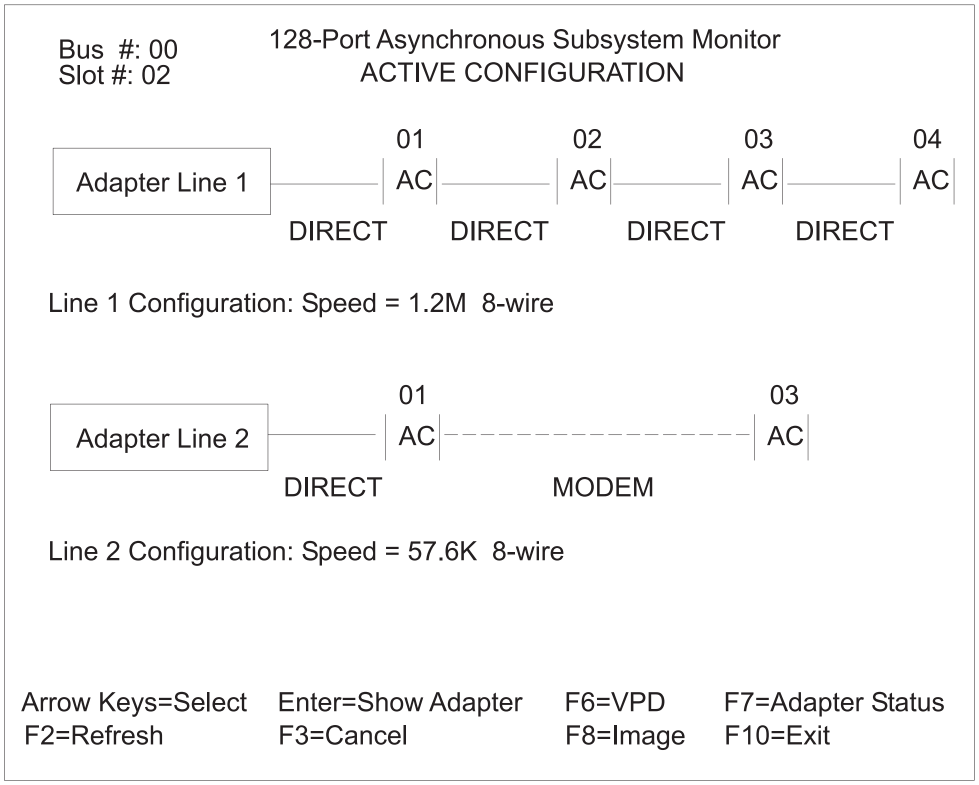


Figure 31. Asynchronous Subsystem Monitor Main Screen

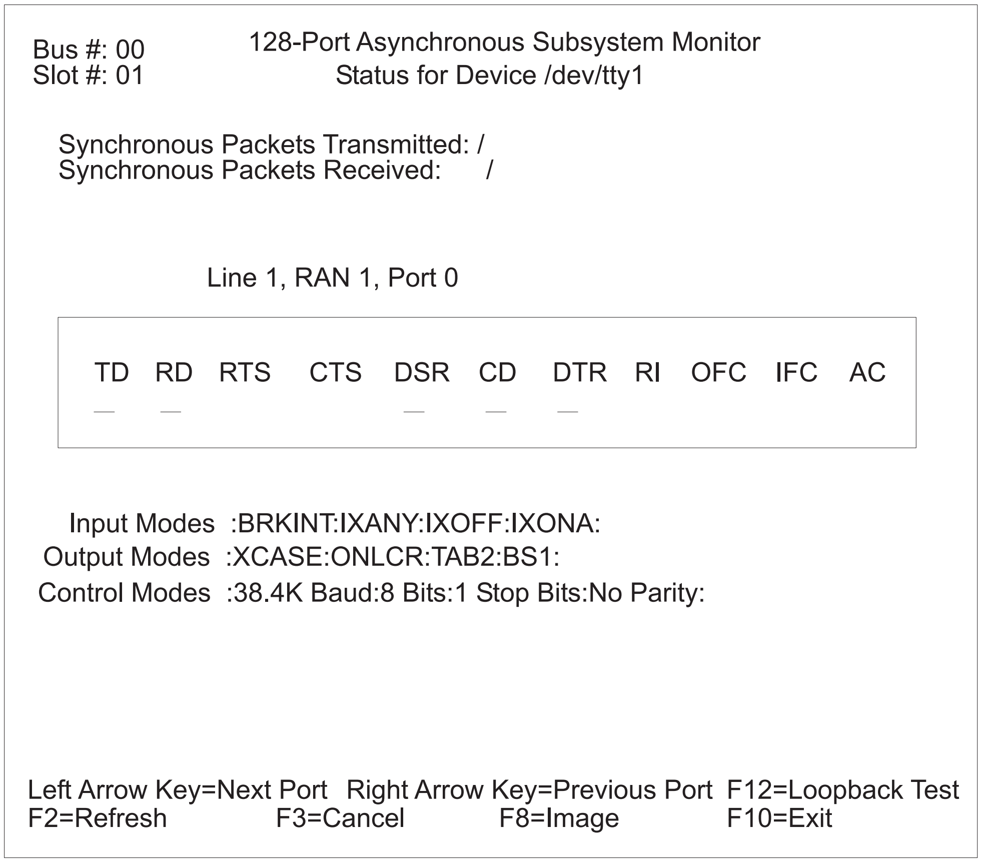
   example, `Adapter Line 1` has four directly attached RANs which are daisy-chained directly off the 128-port adapter. They are all active and assigned node numbers sequentially from `01` to `04`.

   `Adapter Line 2` has two RANs attached. The first RAN, `node 01`, is directly attached to the 128-port adapter while the second, `node 03`, is connected to the first using EIA 232 modems.

You can move between the Adapter Lines or RANs by using the up and down arrow keys. If you press Enter on an Adapter Line, the **ADAPTER CONFIGURATION** screen for that adapter is displayed.

The **ADAPTER CONFIGURATION** screen shows the adapter's line configuration as well as the configuration string stored in memory during system startup. This string is used during initialization to determine line speeds and connection types between the 128-port adapter and RAN. It is useful for diagnosing configuration problems.

6. To view a port's activity, use the arrow keys to select the desired, active RAN (represented by [AC]) and press the Enter key. The software displays a graphic representation of the RAN front panel as shown in the illustration below:



```
Bus #: 00          128-Port Asynchronous Subsystem Monitor
Slot #: 01                 Status for Device /dev/tty1

  Synchronous Packets Transmitted: /
  Synchronous Packets Received:      /


                  Line 1, RAN 1, Port 0

    TD   RD   RTS   CTS   DSR   CD   DTR   RI   OFC   IFC   AC
    —    —                —     —    —

   Input Modes  :BRKINT:IXANY:IXOFF:IXONA:
  Output Modes  :XCASE:ONLCR:TAB2:BS1:
 Control Modes  :38.4K Baud:8 Bits:1 Stop Bits:No Parity:


Left Arrow Key=Next Port  Right Arrow Key=Previous Port  F12=Loopback Test
F2=Refresh            F3=Cancel          F8=Image             F10=Exit
```

Figure 32. Status for Device Screen

This screen is updated at half second intervals and shows the current status of the selected port.

The box in the center of the screen shows the state of the EIA 232 signals TD, RD, CTS, DSR, CD, DTR, and RI, plus input and output flow control status (IFC and OFC) for the selected port.

An inactive signal is indicated by a dash (-) under its name. An active signal is indicated by a block under its name. If data is currently being transmitted or received for the selected port, a blinking ″X″ will appear under the signal name (TD or RD, respectively).

The status of the RAN is represented at the right-hand end of the box with [AC] if the connection to the host adapter is good.

Beneath the line status box are the input, output and control modes for that port. These are terminal attributes with descriptions.

INPUT MODES	
IGNBRK	Ignore Break.
BRKINT	Interrupt on Break.
IGNPAR	Ignore parity errors.
PARMRK	Mark parity errors.
INPCK	Input parity check.
ISTRIP	Strip input characters.
ITOSS	Toss IXANY characters.
IXON	Enable start/stop output.
IXANY	Restart output on any character.
IXOFF	Enable start/stop input.
IXONA	Enable start/stop output Auxiliary.
OUTPUT MODES	
XCASE	Canonical upper/lower display.
OLCUC	Map lower case to upper.
ONCLR	Map NL to CR/NL.
OCRNL	Map CR to NL.
ONOCR	No CR output at column 0.
ONLRET	NL performs CR function.
OFILL	Use fill characters for delay.
OFDEL	Fill character id DEL; otherwise NUL.
NLDLY	NL delay is selected.
CR1	Carriage Return delay type 1.
CR2	Carriage Return delay type 2.
CR3	Carriage Return delay type 3.
TAB1	Tab delay type 1.
TAB2	Tab delay type 2.
TAB3	Tab delay type 3.
BS1	Backspace delay.
VT1	Vertical tab delay.
FF1	Form feed delay.
CONTROL MODES	
Baud Rate	50,75,110,134,150,200,300,600,1200,1800,2400,4800,9600,9200,38400. If fastbaud is enabled, baud rates of 50, 75, 110, and 200 are translated to 57600, 75600, 115200, and 230000, respectively.
Char Bits	5, 6, 7, 8 or none.
Stop Bits	2, 1 or none.
Parity	Enabled, odd or none.

fastbaud	Fast baud rates.
	
	To examine the fastbaud setting, use the command:
	`/usr/lbin/tty/stty-cxma -a tty1`
	
	Fastbaud is disabled if it is preceded by a minus sign. To enable fastbaud, use the command:
	`/usr/lbin/tty/stty-cxma fastbaud tty1`
	
	The baud rate of the tty device must be 50, 75, 110, or 200 for fastbaud to take effect.

Up-arrow	Increment RAN number. If the currently selected RAN is the last one on the synchronous line, then the number wraps around to first RAN.
Down-arrow	Decrement RAN number. If the currently selected RAN is the first one on the synchronous line, then the number wraps around to the last RAN on the line.
Right-arrow,l	Increment port number. If the currently selected port is the last one on the RAN, then the port number wraps around to the first port.

Left-arrow,h	Decrement port number. If the currently selected port is #1, then the port number wraps around to the last port on the RAN.
0-9, a-f	Fastpath port selection. Press 0-9 for ports 0-9, respectively, and a-f for ports 10-15, respectively.
F12	Run loop back test on the selected port. The loop back test transmits 128 bytes and attempts to read them back through the same port. A loop back plug is required.
F8	Dumps the current screen contents to the log file. The default log filename is **/tmp/dpalog**. A different file may be specified by using the **-l** *LogFile* command line option. Refer to the **mon-cxma** command for a description. If the Print key does not map correctly, use ˆP (Ctrl-P) instead.
F3	Return to the previous screen.
F10	Quit the monitor program.

7. To run a loop back diagnostic test from the **Status for Device** screen, press the F12 key. A loop back plug (IBM part number 43G0928) must be installed in the port to be tested. The loop back test consists of five phases:

PHASE 1
Verify that the port is not currently in use. All signals must be low. If a port is busy, a message similar to the following is displayed:

```
Port is Busy  :DTR:CD:DSR:RTS
```

PHASE 2
128 bytes of test data (the letter A is used) are put into the transmit buffer. If the write operation fails, a message similar to the following is displayed:

```
Loop Back Test Failure #1
```

After the write operation is complete, the FEP/OS is notified that data is available for transmission. At this time, the 128 bytes of data is transmitted. If the transmission fails, the following message is displayed:

```
Loop Back Test Failure #2
```

PHASE 3
At this point, the subsystem should have received 128 bytes of data. If not, the following message is displayed:

```
Loop Back Test Failure #3
Data Not Transferred to RX.
```

PHASE 4
128 bytes of data are read from the receive buffer. If the read operation fails, the following message is displayed:

```
Loop Back Test Failure #4
```

PHASE 5
The data that was transmitted is compared with the data that has been read from the receive buffer. If the comparison fails, the following message is displayed:

```
Loop Back Test Failure #5
```

If all test phases pass, the following message is displayed:

```
Loop Back Test Passed.
```

8. Press the F3 key to return to the **Status for Device** screen after the test is completed.
9. You can then press the F3 key to return to the **ACTIVE CONFIGURATION** screen and continue in the monitoring program or exit the SMIT interface.

Configuring Terminal-Attached Printers

Many of the ASCII terminals used today have an auxiliary serial or parallel port on which to connect a printer device. Connections of this type offer administrators a means of sharing valuable computer resources and increasing user productivity and efficiency by moving the printers as close as possible to the users. This section describes the steps necessary to configure this type of terminal-attached printer in this operating system environment.

Hardware Requirements

- IBM 8- or 128-port adapter
- ASCII display (an IBM 3151 display is used for this hardware)
- Serial printer
- EIA 232 serial cabling and gender changers

Prerequisites

1. The 128-port adapter is properly configured and operational.
2. The 3151 display is attached to the 128-port adapter and is operational.
3. The ASCII display is attached to the 128-port adapter and is operational.
4. Printer software for terminal-attached printing is installed.

Setting Up the Hardware

Before adding the printer to your system, perform the following steps:

1. Connect the serial printer to the auxiliary port of your terminal using an EIA 232 modem cable (IBM cable D). A null-modem cable is *not* needed on the AUX port of an IBM 3151 terminal.
2. Make note of the following settings for your printer: line speed, word length (or bits-per-character), parity (odd, even, no, space, mark), and the number of stop bits.

Configuring the Auxiliary Port

On the IBM 3151 terminal, perform the following steps:

1. Turn on your terminal simultaneously pressing the Ctrl and Setup keys. The SETUP menu is then displayed on the 3151 screen.
2. Use the Send key to move between menu screens until the **KEYBOARD/PRINTER** option is displayed. Fill in the **PRINTER** options with the printer settings you noted previously.
3. Press the Send key until the FUNCTION screen is displayed. Select the **Save** option and press the Space Bar to save the configuration.

Adding the Print Queue

At this point, you have physically attached the printer to the terminal and configured its auxiliary port with the correct printer settings. The following procedures describe the creation of a local print queue on the host that will access the terminal-attached printer device.

1. Log on as root or as a member of the printq admin group.
2. Use the **smit mkpq** fast path to access the **Add a Print Queue** menu.

   Note: The screen content may vary depending on the printer software installed on the host.
3. Select the **ascii** option, which takes you to the **Printer Type** screen.
4. Select the appropriate printer manufacturer or Other. For this example, we selected `IBM`. An additional **Printer Type** screen is displayed.
5. Select the appropriate printer model. For this example, we selected `ibm2380-2`. The **TTY Name** screen is displayed.
6. Select the tty for the terminal that has your printer attached. For this example, we selected `tty0`. The **Add a Print Queue** screen is then displayed.
7. Enter a descriptive name for the terminal-attached print queue (for example, `tty0asc`) in the **Name of new PRINT QUEUE to add** field, and press the Return key. The print queue will NOT be created.

Testing the Printer

To verify that the printer is functioning properly, issue an **lpstat** command to list all available print queues on the system. The following output is displayed:

```
# lpstat
Queue    Dev    Status   Job Files   User   PP %   Blks   Cp  Rnk
-----    ---    ------   ---------   ----   ----   ----   --  ---
4019gl   lp0    READY
4019ps   lp1    READY
tty0asc  tty0   READY
```

Use the following **enq** command to send an ASCII file to the printer:

```
enq -P tty0asc /etc/qconfig
```

The file should be printed without altering the display or functionality of the terminal.

Chapter 7. 8-Port Asynchronous ISA/PCI Adapter

The 8-port asynchronous ISA and PCI adapters are multi-channel, intelligent, serial communications features available for POWER-based computers.

> **Note:** And, for the Itanium-based platform, the 8-port asynchronous PCI adapter makes available the multi-channel, intelligent, serial communications features.

The ISA adapters contain 128K of dual-ported high-speed Random Access Memory (RAM) used for program code and data buffering. The asynchronous ports are run by a 32-bit 16 MHz IDT 3041 processor that supports throughput speeds of 115Kbps.

The 3041 processor and dual-ported RAM help to offload much of the character processing from the system. Large blocks of data are transferred directly to the adapter, and then sent out on serial ports one character at a time.

The dual-ported RAM is accessible for read and write operations by both the adapter and the computer. The computer sees the dual ported RAM as its own memory and accesses it using the same high-speed memory referencing commands it uses for internal memory.

The 8-port EIA 232 ISA adapter supports EIA 232 devices only. This adapter requires the device package `devices.isa.cxia` to be installed on the system.

The 8-port EIA 232/422 ISA adapter supports EIA 232 and EIA 422 devices. Both device types may be configured in any combination on a per-port basis. This adapter requires the device package `devices.isa.pc8s` to be installed on the system.

The 8-port EIA 232/422 PCI adapter supports EIA 232 and EIA 422 devices. Both device types may be configured in any combination on a per port basis. This adapter requires the device package `devices.pci.4f111100` to be installed on the system.

All of the above packages require the `devices.common.IBM.cx` package.

Installing 8-Port Adapters

ISA adapters cannot be autodetected by the operating system and must be manually installed. PCI adapters are autodetected and installed by the operating system, but can also be added later.

The following sections contain detailed information about the 8-Port asynchronous ISA/PCI adapters:
- Configuring the Adapter
- Performance Tuning
- Setting Terminal Options with stty-cxma

Configuring the Adapter

ISA

1. To configure the IBM 8-Port asynchronous EIA 232/EIA 422 ISA adapters, use the **smit mkdev_isa** fast path to access the **Add an ISA Adapter** screen.
2. Select **pcxr** (for the 8-port EIA 232 adapter) or **pc8s** (for the 8-port EIA 232/EIA 422 adapter) and press Enter.
3. Select the appropriate bus and press Enter.

4. In the Bus I/O Address field, set the address to the address of the adapter (set by DIP switches on the adapter). For additional information on DIP switches, refer to the *8 Port Asynchronous ISA Adapter Installation Guide*. The remainder of the adapter configuration is done automatically when the system displays **saX Available**.
5. When you have finished, select **Do**.

PCI

Run **cfgmgr** to configure the PCI 8-port adapter. There are no configurable attributes for this adapter.

Performance Tuning

The device driver software is configured to give the best performance under the widest variety of conditions. Performance under certain conditions can be improved through the use of tunable parameters. As with most tunable parameters, increasing performance in one area decreases performance in other areas. The software supports a number of tunable parameters that may be useful under special conditions. These parameters are tunable on a per-port basis and can be set with **stty-cxma**, **chdev**, or SMIT.

EDELAY is a tunable parameter used to determine the number of milliseconds of delay between the time the first character arrives after a period of no characters and notification of its arrival to the host. This is also referred to as the wakeup rate between the host software Front End Process Operating System (FEPOS) and the host device driver. This has the advantage of reducing host overhead by allowing the host to process larger blocks of incoming data. Larger EDELAY values result in more characters being sent in a given time period. This will reduce host processor utilization and character response time and increase overall system throughput. Smaller EDELAY values result in fewer characters being sent in a given time period. This increases character response time and increases host processor utilization. The default value for EDELAY is 100.

This is a good value for normal tty activity including typing and UUCP. For applications receiving continuous input at high speeds, increasing EDELAY results in lower host overhead and an increase in overall system throughput. A value of 250 is reasonable.

Note: For more terminal options, see ″Setting Terminal Options with stty-cxma″ .

Related Information

For additional information about error messages and error logging, see:

The errdemon daemon.

The **errclear** command, **errdead** command, **errlogger** command, **errpt** command, **errstop** command.

Error Logging Overview in *AIX 5L Version 5.1 General Programming Concepts: Writing and Debugging Programs*.

Chapter 8. Modems Overview

Modems provide serial communications across ordinary telephone lines. This chapter discusses modem standards, general modem setup, and specific configuration recommendations when attaching a modem to the server family of asynchronous adapters.

A modem is a device that can connect one computer to another across ordinary telephone lines. The current telephone system is incapable of carrying the voltage changes required for a direct digital connection. A modem overcomes this limitation by modulating digital information into audio tones for transmission across the phone line and by demodulating those tones back into digital information upon reception. It is from these actions that the name MODEM is derived (MOdulate, DEModulate).

Modern modems generally communicate at speeds ranging to 57,600 baud. Some modems use proprietary means to communicate at even faster rates. The disadvantage to these proprietary, faster modems is that in order to achieve faster speeds, identical modems (or more precisely, modems implementing the same proprietary protocol) must be used at both ends. Older modems communicate at speeds ranging from 300 baud to 1200 baud.

Often, the term baud is used to refer to a modem's speed instead of bps. Baud is actually a measurement of the modulation rate. In older modems, there was only 1 bit encoded in each signal change, so a modem's baud rate was equal to the modem's speed. A modem's bps rate is calculated by multiplying the number of data bits per signal with the baud (for example, 2400 baud x 6 bits per signal change = 14,400 bps). Most modern modems can communicate at a variety of speeds (for example: 14400, 9600, 7800, 4800, and 2400 baud). In general, the terms bps and baud are interchangeable.

The following sections contain detailed information about modems:
- Telecommunications Standard
- Modem Considerations for AIX
- Considerations for the TTY Device
- Modem Cabling
- Modem Attachment and Configuration
- Troubleshooting Modem Problems

Telecommunications Standards

The older speeds of 300, 1200, and 2400 baud were well-defined. However, as modem manufacturers began to devise methods for gaining higher speeds, each modem manufacturer started to use a proprietary method or algorithm, incompatible with modems from other manufacturers.

Section topics included are:
- Full and Half Duplex Transmission
- CCITT Communication Standard
- Microcom Networking Protocol
- MNP Communication Standards

Full and Half Duplex Transmissions

When studying telecommunications standards, it is important to understand the differences between half duplex and full duplex transmissions. In a *half duplex* (HDX) transmission, a data packet is sent by one system and received by the other. Another data packet CANNOT be sent until the receiving system sends an acknowledgement back to the sender.

In a *full duplex* (FDX) transmission, both the sending and receiving systems communicate with each other simultaneously; in other words, both modems can send and receive data at the same time. This means a modem can be receiving a data packet while acknowledging the receipt of another.

CCITT Communication Standard

Today, the United Nations Consultative Committee for International Telephony and Telegraphy (CCITT) defines standards for most high-speed communications.

Even modern high-speed modems are much slower than other methods of computer communication. A high-speed modem typically operates at 28,800 baud, while an Ethernet LAN connection operates at 10,000,000 bps. In order to boost data throughput, high-speed modems typically offer one or more data compression algorithms. These algorithms can boost the throughput of a high-speed modem to speeds above 57,600 bps.

Note that these algorithms are sensitive to the data being transmitted. If the data has already been compressed (for example, with the **compress** command), the data compression methods of high-speed modems will offer little or no benefit, and might even reduce data throughput.

When using a modem with data compression technology, the speed of the DTE/DCE connection between the computer and the modem should be greater than the nominal data rate of the connection between modems. For example, with a V.32*bis* modem with V.42*bis* data compression, the data rate of the modem (the speed at which the modem communicates across telephone lines) is 14,400 baud. When the V.42*bis* compression is active, actual data throughput can reach 57,600 bps. To accommodate the greater throughput offered by data compression, the speed of the DTE/DCE between the computer and the modem should be set to 57,600 bps (38.4Kbps).

Some modems implementing data compressions and modem modulation schemes may yield a higher data throughput than some systems and asynchronous adapters can accommodate.

Today, the CCITT defines standards for high-speed communications including data compression algorithms. CCITT standards are usually named V.*nn*, where *nn* is a number.

Following is a list of some common communications standards defined by the CCITT:

V.29	CCITT standard for half-duplex 9600 baud communications.
V.32	CCITT standard for full-duplex 9600 baud communications.
V.32*bis*	CCITT standard for 14,400 communications. V.32*bis* is a revision to the V.32 standard.
V.fast	Proposed CCITT standard for 28,800 baud communications. Note that this standard will achieve 28,800 bps data rates through multiple bit encoding, instead of the data compression scheme used by MNP Class 9.
V.42	CCITT data compression standard.
V.42*bis*	Revised CCITT data compression standard.

Microcom Networking Protocol

Another de-facto standard is the Microcom Networking Protocol (MNP) which was originally developed by Microcom, Inc. Available in versions (called classes) 1-9, MNP is a high-performance, high-speed protocol that was available before the advent of the CCITT standards. With MNP, errors in the transmitted data packets are detected by the remote modem causing it to request a retransmission of the data packet in error. The capability to recognize and quickly correct data errors makes MNP one of today's most common protocols.

MNP Communication Standards

MNP Class 1	An asynchronous, half-duplex, byte-oriented method of transferring data realizing about 70% efficiency. This standard is uncommon in modern modems.
MNP Class 2	A full-duplex counterpart to MNP Class 1 which is also uncommon in modern modems.
MNP Class 3	A synchronous, bit-oriented full-duplex method of transferring data realizing about 108% efficiency. Efficiency greater than 100% is realized because the start/stop bits required for an asynchronous connection are eliminated. The DTE/DCE between the modem and the system are still asynchronous.
MNP Class 4	An enhancement to MNP Class 3 including a mechanism for varying the packet size (adaptive packet assembly) and a means of eliminating redundant administrative overhead (data phase optimization). An MNP Class 4 modem offers approximately 120% efficiency.
MNP Class 5	Includes data compression along with Class 4 features. An MNP Class 5 modem offers 200% efficiency.
MNP Class 6	Allows incorporation of multiple, incompatible modulation techniques into one modem (universal link negotiation). This allows MNP Class 6 modems to begin communication at a slower speed and negotiate a transition to a higher speed. Class 6 also includes a statistical duplexing scheme which dynamically allocates utilization of half-duplex modulation to simulate full-duplex service. All features of MNP Class 5 are supported.
MNP Class 7	Incorporates enhanced data compression. Combined with Class 4, efficiencies of 300% can be realized.
MNP Class 8	Not Applicable.
MNP Class 9	Combine enhanced data compression with V.32 technology to allow data rates up to 28,800 bps.

Modem Considerations

The configuration of a modem attached to this operating system is different than that of a personal computer (PC) or workstation. This section discusses modem interface requirements for the general user.

Section topics include the following:
- Supported Modems
- Data Carrier Detect Handling
- Data Terminal Equipment or Data Communication Equipment Speeds
- Modem Control Signals

Supported Modems

Any modem that is EIA 232 compliant, capable of returning results in response to a command, and capable of communication at one of the following baud rates can be attached to this operating system:

```
 50 134  300 1800  9600
 75 150  600 2400 19200
110 200 1200 4800 38400
```

Data Carrier Detect Handling

The server uses the Data Carrier Detect (DCD) signal to monitor the true state of a modem. If the DCD signal on the modem's port is ″high,″ the server believes the modem to be in use. It is therefore important to know which circumstances cause this signal to be forced into a ″high″ state. The DCD signal can be raised high for the following reasons:

- The use of **clocal** in the stty attributes for runtime field on the SMIT **TTY Configuration** panel.
- Having the Ignore Carrier Detect field set to **enable** on the SMIT **TTY Configuration** panel for ttys connected to a 128-port adapter.
- The modem forces DCD high with either AT commands or switches.
- The tty port is already in use by an application.

Note: When modems make a connection with another modem, the modem raises the CD. Most modem defaults settings set this signal ″high″ at all times even when the modem is idle. CD should not be forced ″high.″

Data Terminating Equipment or Data Circuit-Terminating Equipment Speeds

Data Terminating Equipment (DTE) and Data Communication Equipment (DCE) are used to describe two different hardware groups. The term DTE is used primarily for those devices that display user information. It also includes any devices that store or generate data for the user. The system units, terminals, and printers all fall into the DTE category.

DCE includes any device which can be used to gain access to a system over telecommunication lines. The most common forms of DCEs are modems and multiplexers.

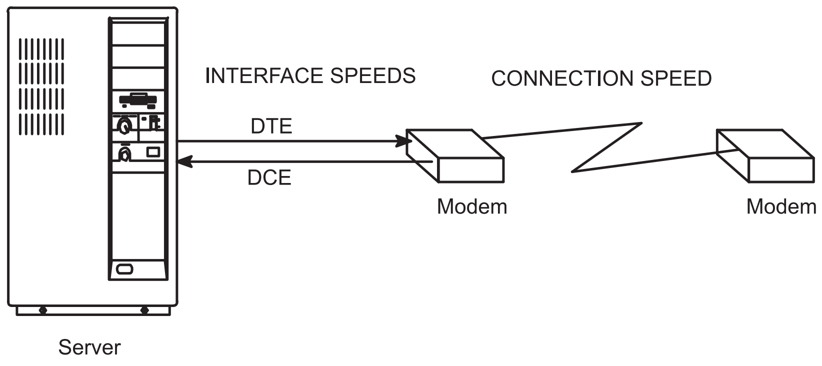


Figure 33. Modem Speed Considerations

With serial communication on this operating system involving modems, as pictured in the above illustration, there are three major considerations:

- DTE interface speed (server to modem). This is the speed the server communicates to the modem.
- DCE interface speed (modem to server) sometimes called the ″serial port interface speed.″ This is the speed at which the modem communicates to the server.
- Connection speed (modem to modem). This is the speed at which a modem communicates (or talks) to another modem.

Most modern, high-speed modems allow the DCE interface speed to be different than the connection speed. This allows the DTE speed to be locked at a single baud rate while allowing the connection speed to fluctuate, up or down as needed, for proper communication between modems.

Modern high-speed modems hold the data to be transmitted to the server in a buffer and send it when the system can accept it. They can also hold data to be transmitted to the other modem in a buffer and send it as the remote is able to accept it. This kind of data transmission requires the modem and the server to engage in *flow control*.

Modem Control Signals

Modems are often used to initiate and receive calls. It is therefore important to program the modem to negotiate a connection at the highest possible speed and to reset itself to a known state after a connection is stopped. The server will toggle the Data Terminal Ready (DTR) signal from on to off to instruct the modem to terminate the connection. Most modems can be configured to reset themselves when this on-to-off DTR transition occurs.

Note: The tty can be configured to not drop DTR by unsetting the `hupcl` flag in the stty run-time attributes.

For the connection between the server and the modem to be fully functional, the cabling must have the following qualifications:

- It must meet specifications.
- It should be properly shielded.
- The following signals should be provided: RxD, TxD, RTS, CTS, SG, DCD, and DTR.

  Note: The 16-port asynchronous adapter does not provide support for the RTS and CTS signals. It is therefore impossible to use RTS/CTS hardware flow control with this adapter.

If binary data is to be transferred using a modem on this adapter, a file transfer protocol that detects incorrect data and resends the missing data (for example, Xmodem, zmodem, Kermit, and UUCP) should be used.

The following describes the signals used by the server:

Signal	Description
FG	Frame Ground. Pin 1 of the EIA 232D specification that provides for a cable shield. Properly used, the signal is attached at pin 1 on one side of the cable only and is connected to a metal sheath around the cable.
TxD	Transmit Data. Pin 2 of the EIA 232D specification. Data is transmitted on this signal. Controlled by the server.
RxD	Receive Data. Pin 3 of the EIA 232D specification. Data is received on this signal, controlled by the modem, which is sent by the modem.
RTS	Request To Send. Pin 4 of the EIA 232D specification. Used when RTS/CTS flow control is enabled. This signal is brought high when the system is ready to send data and dropped when the system wants the modem to stop sending data.
CTS	Clear To Send. Pin 5 of the EIA 232D specification. Used when RTS/CTS flow control is enabled. This signal will be brought high when the modem is ready to send or receive data. It will be dropped when the modem wishes the server to stop sending data. Controlled by the modem.
DSR	Data Set Ready. Pin 6 of the EIA 232D specification. Signals the server that the modem is in a state where it is ready for use. Controlled by the modem.
SG	Signal Ground. Pin 7 of the EIA 232D specification. This signal provides a reference voltage for the other signals.
DCD	Data Carrier Detect. Pin 8 of the EIA 232D specification. This provides a signal to the server that the modem is connected with another modem. When this signal is brought high, programs running on the server will be able to open the port. Controlled by the modem.
DTR	Data Terminal Ready. Pin 20 of the EIA 232D specification. This provides a signal to the modem that the server is on and ready to accept a connection. This signal is dropped when the server wishes the modem to drop connection to another modem. It is brought high when the port is being opened. Controlled by the server.
RI	Ring Indicate. Pin 22 of the EIA 232D specification. This provides a signal to the server that the modem is receiving a call. It is seldom used and is not needed for common operations. Controlled by the modem.

Considerations for the TTY Device

Each tty port must be set up before attaching any input/output device. Before a tty device is created, several factors concerning its functionality should be determined. This section discusses those factors and their importance. Topics covered in this section are the:

- Physical port
- Baud rate
- Bits-per-character, parity, and stop bits

- Port startup and control
- Terminal emulation

Physical Port

Requirements for the physical port to which a modem is connected are dependent on the server and its serial adapters. The system and adapter must support (at least) the following modem control signals to be fully functional on all asynchronous software applications: RxD, TxD, RTS, CTS, SG, DCD, and DTR.

A server typically has two built-in serial ports. Additional multiport adapter cards or terminal servers can be installed on the server to allow the configuration of more serial devices. The following optional adapter cards and base units are supported on the server:

- 8-port adapter
- 16-port adapter
- 64-port adapter
- 128-port adapter

The above adapters and terminal server support the required modem control signals except the 16-port adapter. The 16-port asynchronous adapter does not provide support for the RTS and CTS signals. It is therefore impossible to use hardware flow control with this adapter.

If a binary file is to be transferred using a modem on the 16-port adapter, a transfer protocol that detects incorrect data and resends the missing data (for example, Xmodem, Zmodem, Kermit, and UUCP) should be used.

Review the associated chapter for each of the previously mentioned serial adapters for more information.

Baud Rate

There are several important factors to take into consideration when planning the port baud rate:

- Will the remote system be forced to connect at the local system's communication speed?

  If the remote system is going to be forced to connect at a single speed, that speed should be specified in the BAUD rate field of the SMIT **TTY Setup** menu of the local system. The modem must be set to connect at that same speed.
- Will the local system adapt to the remote system's communication speed?

  The MultiTech MT932 MultiModem II has programmable variables controlling the two separate communication speeds:

  ```
  $SB=9600 $MB=38400
  ```

  The `$SB` value is the serial baud rate between the modem and the server. The `$MB` value is the modem baud rate between the two modems themselves. With the above settings, the modem would communicate to the server at 9600 bps while at the same time communicating with another modem at 38400 bps.

  Note: To avoid incompatibility problems between modems due to nonstandardized implementation of high-speed protocols, it is advisable to use the same make and model of modems at both ends of the connection.

Bits-Per-Character, Parity, and Stop-Bits

The parity, bits-per-character, and stop bits for a serial device are interrelated and require a basic knowledge of serial communications. Most modern serial communications devices (modems, terminals, etc.) use a 10-bit transmission character:

- 1 bit for the start bit
- 1 or 2 bits for the stop bits

- 0 or 1 bits for the parity setting
- 7 or 8 bits for the data character.

The following table shows all the valid combinations.

Bits-per-character	Start	Stop	Parity
8	1	1	none
7	1	1	1
7	1	2	0

Startup and Control

Local/Remote Attachment

When a port is used for bidirectional communication (both dial-in and dial-out or connect-in and connect-out), it should be set up to enable a login on the port while still allowing other users to dial-out. This bidirectional capability can be accomplished by using the System Management Interface Tool (SMIT) to change the tty's Enable LOGIN field to either `share` or `delay`.

Connection Over a Modem (Enable LOGIN=share):

With the tty's Enable LOGIN field set to `share`, the **/etc/getty** command, which controls the login process, will start up on the port and wait for the modem to turn on a carrier signal.

When the modem turns the carrier signal on, the **getty** command attempts to lock the port so that no other processes can use it. If another program (for example, **cu** or **ate**) has already locked the port for dial out, the **getty** command waits until the port is freed by that process before attempting to open it again. If the port is not locked, the **getty** command locks it and sends out a login herald, thus allowing a login to take place on the port.

Direct Connect (Enable LOGIN=delay):

With the tty's Enable LOGIN field set to `delay`, **/etc/getty** waits until the remote system sends a character to the local tty before locking the port and sending a login herald.

Note: The `share` and `delay` settings lock the port as soon as the **getty** process starts. This is because the carrier signal is already up on a direct tty connection.

When LOGIN is set to `delay`, the **getty** command monitors the input buffer of the terminal device for a character. When a character is found, it attempts to lock the port, exactly like the shared port.

Terminal Emulation

A variety of applications and system functions are tailored to specific terminal types such as the DEC VT100, IBM 3151, or Wyse 50. Each terminal type requires its own special emulation to function properly under the operating system. Without proper emulation settings, terminals may display incorrect characters or change the functions of specific keys on the terminal keyboard.

Terminal emulation can be set from the operating system command line by using the TERM= environment variable (for example, TERM=ibm3151 <enter>) or in SMIT by altering the TERMINAL type variable for a tty.

Note: For most modems used for both dial-in and dial-out connections, a TERMINAL type of `dumb` is sufficient. This value allows users dialing in to set their own terminal type (using TERM=) after logging in. If logins occur more frequently from a single, known terminal type, users can setup the TERMINAL type field when defining the tty from SMIT.

The supported terminal types are located in the **/usr/lib/terminfo** directory and subdirectories.

Modem Cabling

The following tables displays a summary of the cable information needed to properly attach a modem to any of the IBM serial controllers.

Adapter/Controller	IBM Part Number(s)
Native Serial (S1 or S2)	00G0943*, 6326741
8-port Controller	6323741
64-port Controller	53F3367, 6323741
128-port Controler	43G0935, 6323741
7318 Terminal Server	60-1166-01

IBM Part Number	Description	Length in Feet
00G0943*	Serial Port Jumpter (pigtail)	.33
6323741	Asynchronous	10
53F3367	Cable-EIA-232/V.24	.33
43G0935	RJ-45 to DB25 Converter Cable	2

*This part number is not required for some machine types.

Modem Attachment and Configuration

The following section contains information necessary to attach and configure a modem to the native ports, 8-, 64-, and 128-port adapters on this operating system.

Topics covered in this section include:
- Creating a TTY device on the server
- Modem connection
- Programming the modem
- Testing the modem

Prerequisites

- The user must have root authority.
- The serial adapter card and port must be available for use by the system.
- Proper modem cabling for serial connection must be used.

Creating a TTY Device on the Operating System for Modem Attachment

Use the System Management Interface Tool (SMIT) to define a tty port for the device attachment. Most fields are for the general device type. The only field that can affect the modem is the Enable LOGIN field with the following values:

DISABLE	No getty process is run on the port. Use this setting for dial-out only modem ports.
ENABLE	A getty process is run on the port. Use this setting for dial-in modems only.

SHARE	A getty process is run on the port, but the getty process allows programs to dial in and out of this port without manually changing to disable or enable. Use this setting for bidirectional port usage.
DELAY	A getty is run on the port in bi-directional mode, but no herald is sent until the getty process receives a keystroke from the user.

Fields specific to the 128-port asynchronous adapter:

Force Carrier or Ignore Carrier Detect	disable
Perform Cooked Processing in Adapter	disable

Note: * This setting is set to disabled if the 10 pin RJ-45 connector is used. This setting should be enabled if the 8 pin RJ-45 connector is used.

Attaching the Modem to the Port

The next step is to physically attach the modem to the serial adapter port using appropriate cabling. If possible, avoid placing the modem and cabling near any high line-noise sources such as florescent lighting, light switches, or uninterruptable power supply (UPS) devices. Sudden, or continuous power fluctuations can cause problems with tty ports (see troubleshooting section for more information).

Programming the Modem

Most modems require customization of their default settings in order to function properly on the system unit. The commands and settings discussed in this section configure a Hayes compatible modem with the basic parameters needed for operation on the server's serial ports.

There are two methods of configuration commonly used. They are:

- Using the **cu** command.
- Using a C program.

Configuring the Modem with the cu Command

The **cu** command is the interactive and convenient way to program a modem on the server. This command is in the **/usr/bin** directory and is a part of **bosext1.uucp.obj** licensed program product.

Prerequisites:

- The Basic Networking Utilities (BNU) should be installed on the system. These utilities include the **cu** command required for configuration.
- The modem must be attached to the serial port with the correct cabling.
- The modem must be switched on.
- Some knowledge of modem configuration is assumed.
- Modem reference manual must be available for problem determination.

Procedure

1. Add the following line to the **/usr/lib/uucp/Devices** file verifying that the letter D (Direct) is in the leftmost position in the file and that the # in **tty#** is replaced with the tty device number of the modem:

   ```
   Direct tty# - Any direct
   Direct tty43 - Any direct
   ```

   Note: Any lines in this file that start with a # (pound sign) in the leftmost position are comments only.
2. Verify that the tty is disabled. Enter: `pdisable tty#`.
3. Enter: `cu -ml tty#`.
4. The message `Connected` is displayed.

5. Verify that you have the attention of the modem. Enter:

   ```
   AT
   ```

   The modem should respond with OK. If it does not respond to your command, refer to troubleshooting.

   - Some modems are case sensitive. Use uppercase letters for the **AT** commands.
   - In normal operation, it is preferable for the modem to reset when the DTR is dropped (&D3 setting). When the modem is being set up for the first time, however, it is advisable not to have the modem reset if DTR is dropped (&D2 setting). If the modem resets itself, all programmed settings which are not saved into the modem's memory will be lost.

     Not having the modem reset also protects changes when &C1 is set, because changing the Carrier Detect status may cause the carrier detect line to toggle on some modems, causing the **cu** command to drop the line. You may wish to set the modem to &D3 after the final setup is made.
   - Although these commands are standard for most Hayes-compatible modems, there is no guarantee that they are standard for your modem. Compare the commands with your modem manual before proceeding.

6. Enter the following commands. The modem should respond with OK after each line is entered.

   ```
   AT&F    <enter>
   ATE1    <enter>
   AT&D2   <enter>
   AT&C1   <enter>
   ATS0=1  <enter>
   ATS9=12 <enter>
   AT&W    <enter>
   ~.      <enter> (terminate connection)
   ```

   AT Command Descriptions

AT&F	Recall the factory configuration as the active configuration.
ATE1	In command state, echo characters from the keyboard to the screen. (Make sure carrier is not ON on the port or modem.)
AT&D2	Monitor the DTR signal. When an on-to-off transition of the DTR signal occurs, the modem hangs up and enters the command state.
AT&D3	Monitor the DTR signal. When an on-to-off transition of the DTR signal occurs, the modem hangs up and resets.
AT&W	Write the storable parameters of the current configuration to memory.
AT&C1	Track the status of the carrier detect signal.
ATS0=1	Autoanswer.
ATS9=12	Carrier-detect response time. Default is 6. Possible values are 1 to 255, in tenths of seconds.

7. Enter *one* of the following commands on the operating system command line, substituting the tty device number for *n*.

   ```
   penable  ttyn
   pshare   ttyn
   pdelay   ttyn
   pdisable ttyn
   ```

   The modem is now configured with the basic commands needed for most of the operating system's serial communication needs.

Configuring the Modem with a C Program

Prerequisites:

- A C Language compiler must be installed on the system.
- An ASCII text editor must be installed on the system (for example, **/bin/vi**).
- The modem must be attached to serial port with the correct cabling.
- The modem must be switched on.

- Some knowledge of modem configuration is assumed.
- Some knowledge of C programming is assumed.
- Modem reference manual must be available for problem determination.

Procedure

1. Using a text editor such as **/bin/vi**, enter the following C code: for example, **vi motalk.c**.

```
/*************************************************************/
/* Motalk - A "C" program for modem setup.                   */
/*                                                           */
/* NOTE: This program is supplied as an example only and     */
/* is not officially supported by IBM.                       */
/*                                                           */
/*                                                           */
/* To create: vi motalk.c <enter>                            */
/* Usage: motalk /dev/tty? [speed]                           */
/*                                                           */
*/************************************************************/
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <termio.h>
FILE *fdr, *fdw;
int fd;
struct termio term_save, stdin_save;
void Exit(int sig)
{
   if (fdr) fclose(fdr);
   if (fdw) fclose(fdw);
   ioctl(fd, TCSETA, &term_save);
   close(fd);
   ioctl(fileno(stdin), TCSETA, &stdin_save);
   exit(sig);
}
main(int argc, char *argv[])
{
   char *b, buffer[80];
   int baud = 0, num;
   struct termio term, tstdin;
   if (argc < 2 || !strcmp(argv[1], "-?"))
   {
      fprintf(stderr, "Usage: motalk /dev/tty? [speed]\n");
      exit(1);
   }
   if ((fd = open(argv[1], O_RDWR|O_NDELAY)) < 0 )
   {
    perror(argv[1]);
    exit(errno);
  }
    if (argc > 2)
  {
       switch(atoi(argv[2]))
       {
         case   300: baud = B300;
                     break;
         case  1200: baud = B1200;
                     break;
         case  2400: baud = B2400;
                     break;
         case  4800: baud = B4800;
                     break;
         case  9600: baud = B9600;
                     break;
         case 19200: baud = B19200;
                     break;
```

```
            case 38400: baud = B38400;
                        break;
            default:    baud = 0;
                        fprintf(stderr, "%s: %s is an unsupported baud\n", argv[0], argv[2]);
                        exit(1);
            }
         }
      /* Save stdin and tty state and trap some signals */
      ioctl(fd, TCGETA, &term_save);
      ioctl(fileno(stdin), TCGETA, &stdin_save);
      signal(SIGHUP, Exit);
      signal(SIGINT, Exit);
      signal(SIGQUIT, Exit);
      signal(SIGTERM, Exit);
      /* Set stdin to raw mode, no echo */
      ioctl(fileno(stdin), TCGETA, &tstdin);
      tstdin.c_iflag = 0;
      tstdin.c_lflag &= ~(ICANNON | ECHO);
      tstdin.c_cc[VMIN] = 0;
      tstdin.c_cc[VTIME] = 0;
      ioctl(fileno(stdin), TCSETA, &tstdin);
      /* Set tty state */
      ioctl(fd, TCGETA, &term);
      term.c_cflag |= CLOCAL|HUPCL;
      if (baud > 0)
      {
         term.c_cflag &= ~CBAUD;
         term.c_cflag |= baud;
      }
      term.c_lflag &= ~ECHO;
      term.c_cc[VMIN] = 0;
      term.c_cc[VTIME] = 10;
      ioctl(fd, TCSETA, &term);
      fcntl(fd, F_SETFL, fcntl(fd, F_GETFL, 0) & ~O_NDELAY);
      /* Open tty for read and write */
      if ((fdr = fopen(argv[1], "r")) == NULL )
      {
         perror(argv[1]);
         exit(errno);
      }
      if ((fdw = fopen(argv[1], "w")) == NULL )
      {
         perror(argv[1]);
         exit(errno);
      }
      /* Talk to the modem */
      puts("Ready... Press <ctrl> C to exit");
      while (1)
      {
         if ((num = read(fileno(stdin), buffer, 80)) > 0)
                       write(fileno(fdw), buffer, num);
         if ((num = read(fileno(fdr), buffer, 80)) > 0)
                       write(fileno(stdout), buffer, num);
      }
      Exit(0);
      }
      /* ********************************************************* */
      /* ******************* END OF PROGRAM ******************** */
      /* ********************************************************* */
```

2. Compile the above program using the system's C compiler.

```
cc -o motalk motalk.c
```

3. Run the program using the following syntax:

```
motalk /dev/ttyn #
```

Where *n* is the number of the tty port and *#* is the speed of the tty port. For example, to set port 43 for 9600 bps, the syntax would be:

```
motalk /dev/tty43 9600
```

4. The program should run and display `Ready...press <ctrl> C to exit`. Use step 6 from Method 1 to program the modem. Press Ctrl-C to stop the program when finished.

Automated Modem Configuration

To program a modem automatically with a shell script, use the UUCP **cu** command. Create and run the following script:

```
#!/bin/ksh
tty=$1
shift
speed=$1
shift
{
       while [ -n "$1" ];do
       echo "$1/r"
       sleep 2
       shift
       done
       echo '˜.'
{| cu -ml $tty -s $speed
```

Troubleshooting Modem Problems

A convenient way to debug any modem problem is to remove the modem and attach an ASCII terminal (with an interposer or null modem) to the same port and cabling as the modem. Set up the terminal with the same line speed, bits per character, and parity as the modem. A login herald should be displayed on the screen. If the herald is displayed on the terminal screen, then the problem is quickly isolated to the modem configuration.

The following tips will help you isolate problems associated with modem connections:

Problem	**Resolution**
Respawning too rapidly	The **getty** program is respawned by **init**.
Messages on console or errpt	If **init** sees that it has to respawn any program more that five times in 225 seconds, it will display the message on the console and not respawn it for a period of time. The solution is to find out why **getty** is dying. There may be several causes: • Incorrect modem settings, usually as a result of having CD strapped high on the modem or cabling and also having either ″echo″ or ″command response″ turned on. (CD can also be assumed high by adding clocal in the runmodes and/or logmodes in the port configuration or also forced on the 128 port.) • Toggling of the CD signal. The **getty** process will die every time CD is toggled from an on to off state. (This action could be caused by a number of reasons. Be sure to verify that the cable is properly shielded. Logging in and out several times in rapid succession can cause this.)
No login prompt displayed after connection to modem	Make sure **getty** is running on the port. If it is, verify that the carrier detect connection to modem signal is being raised after the remote side has connected to the modem. If CD is being properly asserted, then verify the modem is connected to the right port. If you still do not see login, then attach a terminal with interposer to the cable in place of the modem and verify that a login prompt does appear. If you still don't see a prompt, try to echo characters to the terminal screen to verify the cable and hardware are functioning properly.
When a remote modem connects, it immediately disconnects	Verify that the modem is talking to the server at the same speed at which the server is listening to the modem. Try different baud rates for the tty, or program the modem to lock DTE speed to match the speed of the tty port. Verify that the modem or port is not keeping the carrier detect signal high or that the port is already being used by another process.

Problem	Resolution
Getting garbage characters instead of a login prompt	This is due to a difference in protocols. Be sure to verify that the modem and the tty port agree on the same parity, baud rate, flow control, and character size.
Sometimes, after a successful session, no one is able to login	It may be that the modem does not reset after disconnect. See the modem manual to see how the modem can be set to reset after an on to off DTR transition.
Receiver buffer overruns in errpt	The UART chip buffer is being overrun. Lower the value of the receive trigger in SMIT for the tty. This solution is only valid for the native, 8- or 16-port asynchronous adapters. Verify that the modem and tty port are using the same flow control.
ttyhog errors in errpt	Modem and tty either do not agree on flow control, or no flow control is taking place.

Where to Get Additional Assistance

- Your local area representative can assist in the modem configuration.
- Another source of assistance for customers, There are many different support options available to customers in the Support Services offered, including on-site assistance or over-the-phone support. Contact your nearest service representative for assistance.
- Perhaps an often overlooked source of help is the modem manufacturer themselves. Most manufacturers have some type of online assistance for their products.

/usr/lib/uucp/Dialers Sample File Entries

The examples in this section are supplied without any warranty and will work as is for the models mentioned, but may not meet your specific needs. Some modifications will be required to meet your individual needs. Consult your modem manual for a more detailed explanation of the settings.

To use the settings to program the modem, you need an entry in the **/usr/lib/uucp/Systems** file such as:

```
hayes Nvr HayesPRGM Any
```

The **/usr/lib/uucp/Devices** file should have an entry such as:

```
HayesPRGM tty0 - 2400 HayesProgrm2400
```

With the above two entries made, use the following **cu** command to program the modem:

```
cu -d hayes
```

```
# @(#)28 1.3 Dialers.samples, cmduucp, bos325 1/23/94 11:23:14
#
# COMPONENT_NAME: cmduucp
#
#
# (C) COPYRIGHT International Business Machines Corp. 1994
# All Rights Reserved
# Licensed Materials - Property of IBM
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM
# Corp.
#################################################################
# Motorola UDS Modem
#
# Use udsmodemPROGRAM to program the modem.
# Port needs to have rts/cts set.
# Use uds or hayes dialer.
#
# The "udsmodemPROGRAM" line should be a single, continuous line
#
#################################################################
udsmodemPROGRAM =,-, "" \dAT&FQ2\r\c OK
ATE0Y0&C1&D2&S1%B5%E0*LC\r\c OKAT&K3&W\r\c OK

uds =,-, "" \dAT\r\c OK\r ATDT\T\d\r\c CONNECT
```

```
#################################################################
#
# IBM 7855 Model 10
# Use IBMProgrm to program the modem.
# This sets rts/cts flow control, turns
# off xon/xoff, and sets the DTE speed at 19,200 bps.
# The modem will connect at the appropriate speed and
# flow control with the server.
# Port needs to have rts/cts set.
#
# The "IBMProgrm" line should be a single, continuous line
#
################################################################
IBMProgrm =,-, "" \dATQ0\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OK AT&C1\R2\Q2\M14\r\c OK AT&B8N1L0E0\A0\r\c OK
ATS0=1\r\c OK ATQ1&W0&Y0\r\c ""

#######################################################
# The following are used for Dialing out on a 7855
# regular ACU device. We have to turn on result
# codes (Q0) because they are turned off when we
# programmed it. (Keeps all upper case login from
# happening on dial in attempts.)
# We have to have an extra "\" before "\N" because
# the BNU programs strips it if it's before an "N".
########################################################
ibm =,-, "" \dATQ0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 ECL (No Compression)
ibmecl =,-, "" \dAT\\N3%C0Q0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 ECLC (Compression)
ibmeclc =,-, "" \dAT\\N3%C1Q0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 ECLC Compression with 256 byte block size
ibmeclc256 =,-, "" \dAT\\N3%C1Q0\A3\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 1200bps
ibm_ne12 =,-, "" \dATQ0\\N0&A2%C0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 2400bps
ibm_ne24 =,-, "" \dATQ0\\N0&A3%C0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 9600bps
ibm_ne96 =,-, "" \dATQ0\\N0&A6%C0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 19200bps
ibm_ne192 =,-, "" \dATQ0\\N0%C0\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 12000bps
ibm_ne120 =,-, "" \dATQ0\\N3%C0&AL8\r\c OK ATDT\T\d\r\c CONNECT

# IBM 7855 No Compression 1200bps (Dial Quietly)
ibmq12 =,-, "" \dATQ0\r\c OK AT&A2M0DT\T\d\r\c CONNECT

# IBM 7855 No Compression 2400bps (Dial Quietly)
ibmq24 =,-, "" \dATQ0\r\c OK AT&A3M0DT\T\d\r\c CONNECT

# IBM 7855 No Compression 9600bps (Dial Quietly)
ibmq96 =,-, "" \dATQ0\r\c OK AT&A6M0DT\T\d\r\c CONNECT

# IBM 7855 No Compression 19200bps (Dial Quietly)
ibmq192 =,-, "" \dATQ0\r\c OK ATM0DT\T\d\r\c CONNECT

################################################################
#
```

```
# Intel 9600EX Modem
# Use IntelProgram to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff.
# Port needs to have rts/cts set. (Use hayes dialer)
#
# The "IntelProgram" line should be a single, continuous line
#
################################################################
#IntelProgram =,-, "" \d\dAT\r\c OK AT&F\r\c OK AT&S1M1\r\c OK
AT&D3\r\c OKAT&C1\r\c OK ATL0E0Y0&Y0\X1\r\c OK ATS0=1\r\c OK
AT&W\r\c OK

################################################################
# Practical Peripherals 1440FXMT Modem
# Use PracPerProgram144 to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff. (Use hayes dialer)
# DTE speed will be locked at connect speed when
# the modem is programmed. (Suggestion: 38400 baud)
#
# The "PracPerProgram144" line should be a single, continuous
# line
################################################################
PracPerProgram144 =,-, "" \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OKAT&C1&K3\r\c OK ATQ2E1&Q9\r\c OK ATS0=1S9=20\r\c OK
AT&W\r\c OK

################################################################
# Practical Peripherals 9600 bps Modem
# Use PracPerProgram9600 to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff. (Use hayes dialer)
#
# The "PracPerProgram144" line should be a single, continuous
# line
################################################################
PracPerProgram9600 =,-, "" \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OKAT&C1&K3\r\c OK ATL0E0\r\c OK ATS0=1S9=20\r\c OK
AT&W\r\c OK

################################################################
# Practical Peripherals 2400 bps Modem
# Use PracPerProgram to program the modem
#
# The "PracPerProgram2400" line should be a single, continuous
# line
################################################################
PracPerProgram2400 =,-, "" \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OKAT&C1\r\c OK ATL0E0\r\c OK ATS0=1S9=20\r\c OK AT&W\r\c OK

################################################################
# Hayes 2400 bps Modem
# Use HayesProgrm2400 to program the modem.
# (Use hayes dialer to dial)
#
# The "HayesProgrm2400" line should be a single, continuous line
#
################################################################
HayesProgrm2400 =,-, "" \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
AT&D3\r\c OKAT&C1\r\c OK ATL0E0\r\c OK AT S0=1\r\c OK AT&W\r\c OK

#################################################################
# Telebit t2000 Trailblazer Plus
# Use TelebitProgrm to program the modem
# This sets rts/cts flow control, and turns
# off xon/xoff and sets the Default DTE speed at
# 19,200 bps.
# Port needs to have rts/cts set.
# This sets modem to send PEP tones last as they can
# can confuse some other modems.
#
# The "TelebitProgram" line should be a single, continuous line
#
#################################################################
TelebitProgram =,-, "" \dAT&F\r\c OK
ats2=255s7=60s11=50s41=2s45=255s51=254s52=2s54=3s58=2s64=1s66=1\r\c OK
ATs69=1s92=1s96=0s105=0s110=1s111=30s130=3s131=1F1M0Q6TV1W0X3Y0\r\c OK
ATE0&W\r\c OK
# Telebit T2000 dialers Entries:
# Forces a PEP connection:
tbfast =,-, "" \dATs50=255s7=60\r\c OK\r ATDT\T\r\c
CONNECT-\d\c-CONNECT

# 2400bps connection:

#tb2400 =,-, "" \dATs50=3\r\c OK\r ATDT\T\r\c CONNECT

# 2400 MNP:
tb24mnp =,-, "" \dAT\r\c OK ATS0=0S95=2S50=3S41=0\r\c OK
ATDT\T\r\c CONNECT

# 1200bps connection:#tb1200 =,-, "" \dATs50=2\r\c OK\r
ATDT\T\r\c CONNECT

# 1200 MNP:
tb12mnp =,-, "" \dAT\r\c OK ATS0=0S95=2S50=2S41=0\r\c OK
ATDT\T\r\c CONNECT

#################################################################
# Telebit WorldBlazer
# WORLDBLAZERProgram sets the DTE speed at 38400, but
# you could set it higher if the DTE connection can
# handle it. We answer with PEP tones last so as not
# to confuse other modems. This turns off xon/xoff
# and turns on RTS/CTS flow control. The port should
# be locked to 38400 with these settings, and needs
# to have RTS/CTS turned on.
#
# The "WORLDBLAZERProgram" line should be a single, continuous
# line
#################################################################
WORLDBLAZERProgram =,-, "" \dAT\r\c AT AT&F3M0\r\c AT
ATs51=253s92=1\r\c ATAT&W\r\c AT

#######################################################
# ACU Dialers for various BAUD rates for the
# WorldBlazer - each sets the modem to attempt to
# connect at a specific speed and lower. The
# WBlazer will accept whatever the remote modem can
# do. You will want to use PEP for other Telebits,
# so use WBlazer38400 or WBlazer19200 for those
#######################################################
# WBlazer =,-, "" \dAT\r\c OK ATDT\T\d\r\c CONNECT
WBlazer38400 =,-, "" \dATs50=255\r\c OK ATDT\T\d\r\c CONNECT
WBlazer19200 =,-, "" \dATs50=255\r\c OK ATDT\T\d\r\c CONNECT
# WBlazer14400 attempts to negotiate a V.42bis connection.
WBlazer14400 =,-, "" \dATs50=7\r\c OK ATDT\T\d\r\c CONNECT

# For a V.32 connection:
WBlazer9600 =,-, "" \dATs50=6\r\c OK ATDT\T\d\r\c CONNECT

# For a V.22 connection:
WBlazer2400 =,-, "" \dATs50=3\r\c OK ATDT\T\d\r\c CONNECT
```

```
# For a 1200 bps connection:
WBlazer1200 =,-, "" \dATs50=2\r\c OK ATDT\T\d\r\c CONNECT
```

128-Port Modem Cabling Considerations

This operating system does not require DSR in modem-control applications, and since almost all of today's modems have autoanswering capability, the Ring Indicator signal is generally unnecessary.

ALTPIN Modem Wiring for RJ-45 Cabling:

The 10-pin RJ-45 plugs is not the predominant cabling subsystem and may be difficult to obtain in the retail market. The TTY subsystem of this operating system provides an optional feature called ALTPIN, which swaps the logical functions of DSR (Data Set Ready) with DCD (Data Carrier Detect) for a port. When ALTPIN is enabled, DCD becomes available on pin 1 of an 8-pin RJ-45 connector (equivalent to pin 2 of a 10-pin connector).

If you wish to build an 8-wire modem cable for the 128-port RAN, use the 8-pin RJ-45 plug wired as in the following table:

SYSTEM END CONNECTOR 8-pin RJ-45	**DEVICE END**	
	RI	22
1	DSR*	6
2	RTS	4
3 (Chassis)	GND	SHELL
4	TxD	2
5	TxD	3
6 (Signal)	GND	7
7	CTS	5
8	DTR	20
	CD*	8

*The physical location of DSR and CD may be swapped with the ALTPIN parameter when enabled using the stty-cmxa command.

The following figure is a candid look at the asynchronous signal communication between the system unit and an attached modem. Here, data is being sent from the system unit to a remote system.

DEVICE	**SIGNAL**	**ON/OFF**	**MEANING**
Computer	DTR	+	Hey, modem, are you ready to connect to another system?
Modem	DSR	+	Yes, I am ready. Go ahead and dial.
Modem	DCD	+	I've got the other system on the phone.
Computer	RTS	+	OK, can I send data now?
Modem	CTS	+	Sure, go ahead.
Computer	TxD		Sending data out to modem.
Modem	RxD		I've received the data.
Modem	CTS	–	Don't send me any more data, I'm sending it out...
Modem	CTS	+	OK, I'm ready for more data, let me have it!
Transmit data steps may be repeated until...			
Computer	DTR	–	FINISHED! Go ahead and hang up.

Modem	DCD	–	OK.

Here is the signal communication between a RISC and a modem about to receive an incoming call from another system.

Computer	DTR	+	I’m ready and have ″enabled″ the port for dial-in.
Modem	DSR	+	I’m ready also but I’m just waiting around for a call.

Someone calls in!

Modem	DCD	+	Somebody called in and I’ve got them on the line.
Modem	CTS	+	I’ve got data from another box, can I send you data now?
Computer	RTS	+	I’m ready to receive. Go ahead and send.
Modem	RxD		Here it comes!

Modem continues to send data until...

Computer	RTS	–	WAIT! My buffer is full, don’t send any more data.
Computer	RTS	+	I’m OK now. Send me more data.
Modem	DCD	–	Call has ended.
Computer	DTR	–	OK, please hang up.

Chapter 9. Asynchronous Communication Applications

This section describes the communication for asynchronous applications.

Topics discussed are:
- Serial Line Internet Protocol
- SLIP Questionnaire
- Asynchronous Terminal Emulation
- Basic Network Utilities

Serial Line Internet Protocol

Serial Line Internet Protocol (SLIP) is the protocol which TCP/IP uses when operating through a serial connection. It is commonly used on dedicated serial links and dial-up connections that operate at speeds between 1200bps and 19.2Kbps or higher.

> **Note:** To use baud rates higher than 38400, specify a baud rate of 50 in the **/etc/uucp/Devices** file for the desired tty, then change the SMIT configuration for that tty to reflect the actual baud rate desired.
>
> For example. to run the **cu** command on tty0 with a baud rate of 115200, use the following procedure:
> 1. Ensure the hardware supports the baud rate.
> 2. Edit **/etc/uucp/Devices** to include the following line:
>    ```
>    Direct tty0 - 50 direct
>    ```
> 3. Enter the **smit chtty** fast path.
> 4. Select tty0.
> 5. Change the baud rate to 115200.
> 6. Exit SMIT.

See the following for further discussions of SLIP:
- SLIP Configuration Steps
- Modem Considerations
- Manual Modem Programming Using cu
- Automated Modem Configuration
- Configuring SLIP over a Modem
- Temporarily Deactivating a SLIP Connection
- Activating a SLIP Connection
- Removing a SLIP Interface
- Debugging SLIP Problems
- Common Problems and Error Messages

SLIP Configuration Steps

There are two recommended steps to follow during SLIP configuration. Using this two-step approach separates the hardware and machine-dependent configuration requirements from the SLIP software and command syntax problems.

Step 1

Use ATE or the **cu** utility to accomplish a successful login at the remote system. This proves the usability and correctness of the physical link.

It is important to verify the operability of any modems involved in a SLIP link as they are the most frequent cause of problems during the setup phase. For this reason, topics on modem considerations and modem programming are presented before any SLIP configuration procedures in this chapter.

Step 2

After establishing an error-free login to the remote system using ATE or the **cu** command, the user can begin the SLIP configuration.

Modem Considerations

When configuring modems for SLIP, it is important that the following changes be made on both ends of the communication link. Both the local and remote modems must be configured exactly the same.

1. The modem must acknowledge the presence of DTR.

   Referencing the local modem, if DTR is assumed or ignored, the modem can never perform a hang-up. It can only close the line or hangup when it recognizes the loss of carrier from the other end. This means that disconnects can occur only when instigated by the other end. The AT commands &D2 or &D3 are proper settings for most Hayes-compatible modems.
2. The modem must never force, assume, or ignore data carrier detect (DCD).

   DCD must follow or track the real condition. This means that carrier will exist after a bona fide connection to the other end (modem) across the switched telephone line. This also applies to a dedicated line. &C1 is the suggested setting for most Hayes-compatible modems.
3. The modem must never force, assume, or ignore a clear to send (CTS) signal.

   CTS must track or follow request To send (RTS). If CTS is forced true, the port open will fail whenever a **getty** is put upon the port or when RTS flow control protocol is added to the port.
4. Modems should be configured to turn off automatic repeat request (ARQ) codes if problems arise during **slattach** dial attempts.

   If, the modems repeatedly fail to make a connection during **slattach** dial-in attempts, the user should check the modem configurations and turn off the ARQ codes if they are currently on. In most Hayes-compatible modems, this is the &A0 setting.

   Disabling ARQ result codes does not affect error-controlled connections nor does it keep the modem from returning standard CONNECT messages (if result codes are enabled) as needed for the **slattach** dial string.
5. ECL (Error Checking on the Link) is critical.

   Either BOTH modems or NEITHER modem can use it. Normally, both modems must agree on its usage during the connect session. If ECL is chosen, the physical telephone line must be good enough to allow a recovery from a data error before the TCP/IP timers expire while awaiting an acknowledge packet for the last data sent across the SLIP link.
6. Data Compression across the link.

   It is acceptable to use data compression across the link as long as it is totally handled by the modems. SLIP does not perform any type of compression. If data compression is invoked, it is much better to have two modems of the exact same type; this ensures that each will perform the compression in the same manner and same time frame.

Manual Modem Programming Using cu

Use the following procedure to manually program modems attached to the system unit.

Prerequisites

- The UNIX-to-UNIX Copy Program (UUCP) must be installed on the system. Use the **lslpp -f | grep bos.net.UUCP** command to verify installation.
- A modem must be attached to the system and powered on.
- Root user authority is needed to change the appropriate files.

Procedure

1. Add the following line to the **/etc/uucp/Devices** file if it does not already exist (replace # with the number for your port).

   ```
   Direct tty# - Any direct
   ```

   Note: Any line in the Devices file which begins with a # sign in the leftmost column is a comment.
2. Save and exit the file.
3. Enter the following command on the command line:

   ```
   cu -ml tty#
   ```
4. A connected message should appear on the screen indicating that the modem is connected and ready to be programmed.
5. Type AT and press Enter. The modem will respond with OK. If there is no response from the modem or if characters typed do not appear on the screen, check the following:
   - Verify modem cabling connections.
   - Verify that modem is powered on.
   - Observe the modem front panel lights when you press Enter. If the Receive Data (RD) and Send Data (SD) lights flash, then the modem is communicating with the system and the problem may lie with the current modem settings. If the lights do not flash, then the problem is with the modem connection.
   - Type the following and see if the condition changes:

     ```
     ATE1 <enter>
     ATQ0 <enter>
     ```

     ATE1 turns the echo mode on which displays any typed characters to the screen. ATQ0 enables the displaying of result codes.
6. Program the modem using the settings shown in the previous section, ″Modem Considerations.″ The following example demonstrates how to program and save basic settings for a Hayes-compatible modem. Enter:

   ```
   AT&F    <enter>
   AT&D2   <enter>
   ATS0=1  <enter>
   ATS9=12 <enter>
   AT&C1   <enter>
   AT&W    <enter>
   ~.      <enter>
   ```

   Where &F is used to reset the modem to factory defaults, &D2 sets DTR, S0 and S9 set register values, &C1 set carrier, and &W writes the settings to the modem. The tilde-period ends the connection.

Automated Modem Configuration

Users can customize their modems manually or use the **cu** utility with its associated files to create an automated modem configuration script.

Prerequisites

- UUCP must be installed on the system. Use the **lslpp -f | grep bos.net.UUCP** command to verify installation.
- A modem must be attached to the system and powered on.
- The modem AT command string must already exist (for example, `at&f&c1&d3`). Users should not attempt automated modem configuration until the command string has first been tried manually using the **cu** command.
- Root user authority is needed to change the appropriate files.

Procedure

The following example shows how to automatically configure a Telebit T3000 modem attached to tty0.

1. Edit the **/etc/uucp/Systems** file.
2. Add the following line at the end of the file. The entry should begin in the leftmost column of the file.
   ```
   telebit Nvr TELEPROG 19200
   ```
3. Save and exit the file.
4. Edit the **/etc/uucp/Devices** file.
5. Add the following line at the end of the file. The entry should begin in the leftmost column of the file.
   ```
   TELEPROG tty0 - 19200 TelebitProgram
   ```
6. Save and exit the file.
7. Edit the **/etc/uucp/Dialers** file.
8. Add the following lines at the end of the file. The entries should begin in the leftmost column of the file.

   Note: The following four lines should be made into one long line:
   ```
   TelebitProgram  =,-,    "" \dAT&F\r\c OK
   ats0=1s2=255s7=60s11=50s41=2s45=255s51=252s63=1s58=2s64=1\r\c OK
   ATs69=2s105=0s111=30s255=0M0&C1Q2&D3&Q0&R3&S1&T5\r\c OK
   ATE0X12&W\r\c OK
   ```
9. Save and exit the file.
10. To begin the automated configuration, enter the following command:
    ```
    cu -d telebit
    ```

    The command will fail because you are not connecting to a system. Watch the debug output of the command to see that `ATE0X12&W` is sent to the modem and that an OK is received. If so, then the modem has been successfully programmed.

Problems may arise because of incorrect values placed in the **Dialers** file or because of the modem's existing configuration. If this occurs, try programming the modem manually and enter the dialers strings (in step 8) one by one.

Configuring SLIP over a Modem

The following procedure lists the steps necessary to configure a SLIP line between two system units communicating over 9600 baud modems. For clarity, these instructions use the names `systemA` and `systemB` for the two hosts.

Prerequisites

1. You must have root user authority.
2. The modems must be physically connected to `systemA` and `systemB`.

Procedure

1. To create a tty on `systemA` through SMIT, use the **smit tty** fast path.
2. Select **Add a TTY.**
3. Select `rs232` as the type of tty you want to create.
4. Select an available serial port, for example `sa0`.
5. Select a port number for this tty. Position the cursor on the PORT number field; list and select an available port number from the list.
6. Set the BAUD rate to the baud rate of the modem.
7. Set Enable LOGIN to **disable**.
8. Set FLOW CONTROL to be used to **RTS** or **none**.
9. Select **Do**.
10. Exit the SMIT interface.
11. To create a tty on `systemB`, repeat steps 1-10, except set Enable LOGIN to **enable**.

    Note: The remaining instructions assume that the tty number on both systems is `tty1`.
12. Test the physical modem connection with the **cu** command.
    a. On `systemA`, enter:
    ```
    cu -m1 tty1
    ```
    b. Once connected to the modem, type:
       - ATDT ###-####
       - Where ###-#### is the phone number of the remote machine.

    c. At this point, a login prompt from `systemB` should appear. If not, return to the section entitled ″Configuring Modems for SLIP″ and verify modem setup for both systems. *Do not continue* unless a successful login to `systemB` is achieved.
    d. Log in to `systemB`.
    e. Now type **exit** to logoff of `system B` and type **˜.** to exit **cu**.
13. Since the tty configuration for use with **cu** is slightly different from the configuration for use with SLIP, enter **smit chgtty** on `systemA`.
14. Select `tty1`.
15. Select **Do**.
16. On systemB, enter the smit chgtty fast path.
17. Select tty1.
18. Set Enable LOGIN to **disable**.
19. Select **Do**.
20. Exit the SMIT interface.
21. Add the following line to the **/etc/uucp/Devices** file on both `systemA` and `systemB`:
    ```
    Direct tty1 - 9600 direct
    ```
    This entry must precede any other entry for tty1 in the Devices file and should always begin in the leftmost column.
22. To create a SLIP network interface on `systemA`, use the **smit mkinet** fast path.
23. Select **Add a Serial Line INTERNET Network Interface**.
24. Select `tty1`.
25. Specify the INTERNET ADDRESS of `systemA`. For example:
    ```
    [130.130.130.2]
    ```
26. Specify the DESTINATION Address of `systemB`. For example:
    ```
    [130.130.130.1]
    ```

DO NOT make entries in the BAUD RATE or DIAL STRING fields at this time. These entries can be added later after the correct operation of SLIP is verified through command line options.

27. Select **Do**.
28. To create a SLIP network interface on `systemB`, repeat steps 26-31. Change the network addresses as follows if following the examples provided:

    ```
    INTERNET ADDRESS     [130.130.130.2]
    DESTINATION Address  [130.130.130.1]
    ```
29. Add the following two entries to the **/etc/hosts** file on both `systemA` and `systemB`:

    ```
    130.130.130.1    systemA
    130.130.130.2    systemB
    ```

    The name assigned should be unique. In other words, if the Token-Ring interface on `systemA` is already assigned the name `systemA`, assign the SLIP interface a name such as systemA_slip.
30. Start SLIP on `systemB` by entering:

    ```
    slattach tty1 9600
    ```
31. Start SLIP on `systemA` by entering:

    ```
    slattach tty1 9600 ' "" AT OK ATDT555-1234 CONNECT "" '
    ```

    The command can be read as: Use `tty1` at 9600 baud. Send AT to the modem. The modem should respond with OK. Dial the phone number 555-1234. The modem should respond with CONNECT.

    Users may add the number nine to the end of the **slattach** dial string in order to obtain debug information during the dial attempt. For example:

    ```
    slattach tty1 9600 ' "" AT OK ATDT555-12349 CONNECT "" ' 9
    ```

    This debug output is similar to information displayed when using the **cu -d** command on a tty device.

    Note: The debug option is not a supported feature of **slattach** and is supplied on an as-is basis. Non-supported features may be removed at any time without notice to the users.
32. Test the SLIP connection using the **ping** command.
    a. On `systemA` enter: **ping systemB**.
    b. On `systemB` enter: **ping systemA**.

    If both tests succeed, the SLIP connection is ready for use. If not, return to step 25 and verify that the configuration on both systems is correct.

Temporarily Deactivating a SLIP Connection

To temporarily deactivate a SLIP connection, do the following on both the local and remote systems:

1. Enter:

   ```
   ifconfig sl# down
   ```
2. List the currently running **slattach** processes using the command:

   ```
   ps -ef | grep slat
   ```

   The output may be similar to the following:

   ```
   root  1269  1    0   Jun 25  ... slattach
   ```
3. Kill the **slattach** process using its process ID. For example, to kill the **slattach** process shown above enter:

   ```
   kill 1269
   ```

   where 1269 is the **slattach** process ID. Do NOT remove the **slattach** process using the -9 flag of the **kill** command.

The SLIP connection is now disabled.

Activating a SLIP Connection

Use the following instructions to activate a SLIP connection that is temporarily disabled using the above instructions. Run these commands on both the local and remote systems.

1. Enter:
   ```
   ifconfig sl# up
   ```
2. Re-issue the **slattach** command used initially. Review the instructions on pages 19 and 20 for additional help with this step.

Removing a SLIP Interface

Use the following instructions to completely remove a SLIP interface. Once these instructions are executed both the sl# interface and its associated **slattach** process are removed. Any entries made to the **/etc/hosts** file will remain and must be removed manually.

1. To remove the SLIP interface and its associated **slattach** process, use the **smit rminet** fast path to access the **Available Network Interfaces** screen.
2. Select the appropriate entry from the **Available Network Interfaces** screen and select **Do**.

Note: Any entries made to the **/etc/hosts** file will remain and must be removed manually.

Debugging SLIP Problems

The following describes the commands needed to debug SLIP problems and supplies you with examples.

netstat Command

The **netstat** command works in conjunction with the **ifconfig** command to provide a status condition of the TCP/IP network interface. The command **netstat -in** for example uses the **-i** flag to present information on the network interfaces while the **-n** flag prints the IP addresses instead of the host names. Use this command to verify SLIP interfaces, addresses, and hostnames. The following section describes **netstat -in** output.

Program the modem using the settings shown in the previous section, ″Modem Considerations.″ The following example demonstrates how to program and save basic settings for a Hayes-compatible modem. Enter:

```
Name  Mtu   Network    Address            Ipkts     Ierrs Opkts Oerrs  Col
lo0   1536  <Link>                        2462       0    2462   0     0
lo0   1536  127        localhost.austi    2462       0    2462   0     0
tr0   1492  <Link>                        1914560    0   21000   0     0
tr0   1492  129.35.16  glad.austin.ibm    1914560    0   21000   0     0
sl0   552   1.1.1.0    1.1.1.1            48035      0   54963   0     0
sl1*  552   140.252.1  140.252.1.5        48035      0   54963   0     0
```

netstat -in Command Output:
Notice the * next to the sl1 interface. This shows that the network interface is down or unavailable for use. The user can correct this by issuing the **ifconfig sl1 up** command if it is a valid SLIP interface.

netstat provides statistics concerning input and output packet counts as well as input and output errors that are helpful when troubleshooting SLIP connections.

Example:
The user enters a **ping** to a remote host across a SLIP link and the **ping** command appears to hang. They quickly run a **netstat -in** command from another command shell and notice that the Opkts are

increasing but that there are no Ipkts from the remote host. This indicates that the remote system is not returning (or not receiving) the information. They must run the same **netstat** command on the remote system to verify the receipt of the **ping** packets or rise in the error count.

The translation of hostnames versus Internet numbers is relative to name resolution and thus critical to proper operation of a SLIP line. To debug hostname, aliases, and routing problems, use the **netstat -rn** command. The basename of the host or hostname is the only name that should return from the **/etc/hosts** file. If the machine is being serviced by a nameserver (ie. **/etc/resolv.conf** exists), then the name-server will return the fully qualified-domain name in this command.

ifconfig Command

The **ifconfig** command is the network interface configuration tool that allows the network interface STRUCTURE to be dynamically created or deleted from the kernel memory. This command accepts data from the command line, then builds a memory structure that conforms to the parameters. For debugging purposes, the **ifconfig** command is used to examine the status of a communications interface.

Example:
To examine the current status of the sl1 interface:

1. Enter the **netstat -i** command and examine the output selecting the appropriate sl# interface. For example, sl0, sl1, sl2, etc.
2. Enter the **ifconfig sl#** command and examine the ifconfig output for the following key fields:

POINTTOPOINT flag	This flag should always be present on an operational SLIP link. If not, the link could be in a down or disconnected state. Try issuing the **ifconfig sl# up** and the **ifconfig sl#** commands again to see if its condition changes.
UP flag	Indicates that the network sl# interface is activated and should be operational.
RUNNING flag	Indicates that the **slattach** command was successful. In actuality, the link is accessed, a dial is completed, the other end has answered, and the remote end has returned CARRIER DETECT status. When the CD status occurs the flags are updated with the running bit.

pdisable and lsdev Commands

Any tty port that is used for SLIP connections must be in a disabled or unavailable state. To verify that the port for tty1 is disabled, obtain root user authority and enter one of the following commands:

- **lsattr -El tty1 -a login**

  This command displays the permanent state of the tty port as recorded in the system's Object Database Manager (ODM). If the output is anything other than `login disable`, use SMIT to change the enable LOGIN field to **disable**.
- **pdisable | grep tty1**

  This command, when used without parameters, displays all tty ports that are in a disabled state. In this example, **pdisable** is piped to the **grep** command to eliminate unnecessary output. If tty1 is not displayed after running this command, the port is not disabled.

ps Command

The **ps** command displays information about active processes to standard output. Use this command to verify the existence (or nonexistence) of **slattach** processes that are used to assign a tty line to network interfaces.

If **netstat -in** shows that the interface is down, the user should run the **ps -ef | grep slat** command to see if an **slattach** process is currently running on the associated tty port. Note that for a directly connected SLIP interface, broken connections are retried automatically without manual intervention. For a SLIP interface connected by modem, broken connections must be manually redialed. If a user supplies a dial string in the **slattach** command line, the user must reenter the command and dial string to restore a broken connection.

ping Command and Modem Lights

The **ping** command and modem lights are used to debug SLIP communication problems. A ping is an echo request packet, sent out of the machine, and an echo response packet is returned. This sequence of events is useful if the administrator can see the modem lights.

Example:
The local system constructs the echo request packet and sends it to the remote system. The Send Data (SD) light on the local modem illuminates. This means that the local TCP/IP, **slattach**, and tty were able to group information and send it out of the modem to the remote system.

The remote modem receives the packet and the receive data light flashes but its SD light does not. This means that the remote system was not able to send (or return) the local system's ping request. As a result, the user on the local system may see the **ping** command hang, requiring a **Ctrl c** to exit the condition.

The most common cause of this problem is the use of XON/XOFF flow control in one or both modems, however, the user should not overlook the possibility of routing or address conflicts on the systems.

Common Problems and Error Messages

Message: 0821-296 Cannot set line discipline for **/dev/tty#** to slip.ioctl(TXSETLD). A system call received a parameter that is not valid.

Possible Causes: This type of error normally occurs when starting the **slattach** process and is attributable to incorrect configuration of SLIP. The problem is most likely caused by a mismatch between the tty device number and the sl interface number. This also explains why the system reported that ifconfig had not been run before **slattach**.

This problem may also occur when **slattach** processes are dropped or killed incorrectly or when the user attempts to move a SLIP connection to another tty port and forgets to reconfigure the sl# interface to match the tty. Check for running **slattach** processes that may still be running (for example, **ps -ef | grep slat**).

Action: The tty device for SLIP is **/dev/tty24** and user has created an sl0 interface. This is incorrect. The user should create an sl24 interface which matches the tty number (tty24 and sl24). If the problem continues, the user should bring down the sl interface (see ″Bringing Down an SLIP Interface″) and reconfigure the connection using the following commands:

```
lsdev  -Cc if  -s SL
lsattr -El sl0
```

Message:

network is not currently available

route to remote host not available

Possible Cause: These errors occur most often when a user attempts to ping a host over the SLIP link and the link has been improperly established. The most likely problem is that one or both tty ports associated with the sl# interface are in an enabled state. It is also possible that there is an address or route conflict between the host systems.

Actions:

- Remove the sl# interface using the **smit rminet** fast path This must be done on both the local and remote SLIP hosts.
- Do the following for each SLIP host:
  1. Enter **pdisable | grep tty#**.
  2. If the tty device is NOT listed in the output of the previous command, the tty is not disabled. Disable the tty either through SMIT or the command line. With tty ports disabled, use SMIT to recreate the SLIP interfaces on both systems. If problem persists, verify network addresses and routes (if any). Use the **netstat -ir** command to quickly view address, routing, and interface information.

Problem: When the remote site dials in to the local host, the modem on the local host connects but does not complete the login process.

Possible Causes: If the two modems connect and begin to handshake or exchange connection information but then disconnect, the problem may be due to modem result codes. This problem can also be caused by an improper **slattach** dial string. If the two modems ring but never begin the handshake process, the problem may be that the modem is not set for auto-answer.

Actions:

1. Test the modem connection first with the **cu** command. The modem on the remote host should allow the user to login to the system. There should not be any garbage on the screen during the login attempt; if so, it may indicate a noisy phone line which may be part of the problem. During the login, multiple login heralds should *not* scroll across the screen. If they are present, this could again indicate a problem phone line or incorrect modem settings.
2. Check the modem configurations and try turning off the ARQ codes if they are currently on. In most Hayes-compatible modems this is the &A0 setting. Disabling ARQ result codes does not affect error-controlled connections nor does it keep the modem from returning standard CONNECT messages (if result codes are enabled) as needed for the **slattach** dial string.

Problem: The user is unable to **ping** across a modem SLIP connection. The **ping** command may hang or return error messages.

Possible Causes:

1. The modems and/or tty ports may be configured to use XON/XOFF flow control.
2. The **slattach** process may have been terminated on the remote host or the modem connection dropped.
3. The addresses assigned to the SLIP hosts may be incorrect.

Actions:

1. Examine both the local and remote modem configurations. They should be set to use RTS/CTS (hardware) flow control or no flow control at all. The user should attempt to ping from each system. Ping systemA to systemB.
2. Verify that the **slattach** process is still running on both local and remote systems. Use the command: **ps -ef |grep slat**. Verify that the sl# interface is in a running state. Use the command: **ifconfig sl#**.
3. Verify that there is not a conflict between the SLIP addresses and those associated with other network interface (if any). Use the command: **netstat -ir**. If the address or address class is in question, reconfigure SLIP using a simpler address scheme such as 1.1.1.1 for the local host and 1.1.1.2 for the remote host.

SLIP Questionnaire

Use this questionnaire to record data on SLIP configurations. Information collected on these sheets can be faxed to a service representative when additional assistance with SLIP configuration is required.

1. Was this SLIP configuration working previously? (Y/N) ____
2. What are the machine types? (example: UNIX/PC, DOS/PC, etc.)

   Local System: ________________ Remote System: _________________

   If the host is not an IBM UNIX system, please name the type of software being used to establish the SLIP connection.

   ____
3. What versions of the IBM UNIX operating system are on each of the system units? Issue the **/bin/oslevel** command. If this command is not recognized use the following method:

   ```
   lslpp -h bos.Rte
    look for the
   active commit
    line release level.
   ```

   Local System: ________________ Remote System: _________________
4. List all interfaces available on both systems (for example, sl0, sl1). To do this, use the command: **lsdev -Cc if**

   Local System: ______________ Remote System: _____________

   ______________ _____________

   ______________ _____________

   ______________ _____________

   The SLIP interface number should match the tty device number. For example, **/dev/tty53** should be used with sl53.
5. Is SLIP being configured through SMIT or with commands? _____________________

   SLIP configurations using commands are not permanent and are not present after a system reboot.
6. Is SLIP being configured over modems or a direct serial line?

   ____
7. If modems are being used, list the manufacturer and modem type for both the local and remote systems.

   ```
           TYPE      BAUD RATE  IBM CABLING If not IBM cable,
                                (Yes/No)    what type?
   Local:  ________  _________    ___       _____________________
   Remote: ________  __________   ___       _____________________
   ```
8. If modems are in use, what is the phone carrier type? (leased-line or normal switched)

   ____
9. What hardware is the SLIP line being used on?

   128-Port Adapter (with 16-port RAN(s): ___

   64-Port Adapter (with 16-port concentrator(s)): ___

   16-Port Fanout Box (DB25 pinouts): ___

   8-Port Fanout Box (DB25 pinouts): ___

   Native, S1 or S2 serial ports: ___
10. Is it possible to ping from the local system to the remote system?

    ```
    (Y/N) _____    (on the local system enter: ping <remote address> )
    ```
11. Is it possible to ping from the remote system to the local system?

    ```
    (Y/N) _____    (on the remote system enter: ping <local address> )
    ```
12. Are the tty ports disabled on both the local and remote systems?

`(Y/N)` _____

Use the command: **pdisable | grep tty#**. Only disabled tty numbers are displayed as output of this command.

13. Are any error messages being displayed? If yes, please list them below:

    ____
    ____
    ____
    ____
    ____
    ____
    ____
    ____
    ____
    ____

Asynchronous Terminal Emulation

The Asynchronous Terminal Emulation (ATE) program enables terminals on the operating system to emulate a terminal, thus allowing users a means of connecting to most other systems that support asynchronous terminals. ATE accomplishes this by making the remote system see a terminal either as a system's display or as a DEC VT100 terminal. The VT100 option allows the user to log in to systems that do not support their terminal, but do support VT100 terminals.

ATE uses both direct (cabled) and modem connections to communicate between the user's system and a remote system as shown in the following illustration.

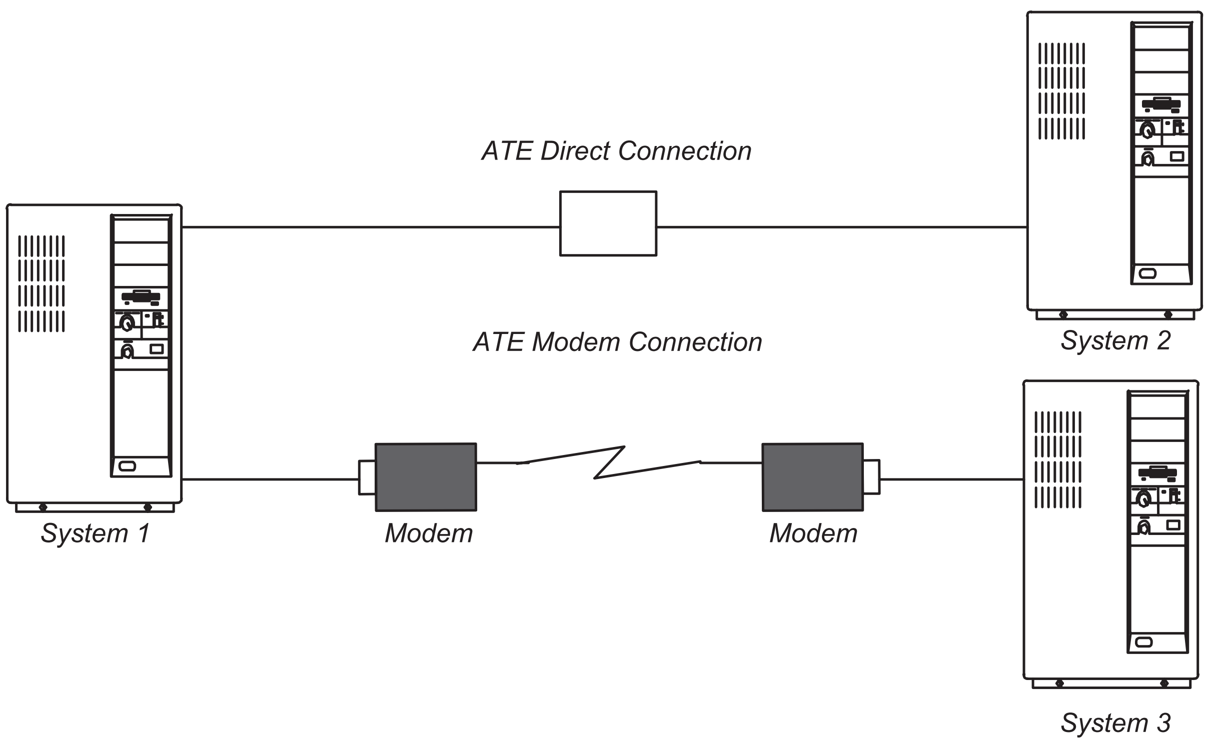


Figure 34. ATE Connection Types

Depending upon the connection type used, the user can configure ATE to connect either to a system in the next room or to a system across the country. For a direct connection, the user must know the port to use on their system. For a modem connection, users must know the port to use on his system and the telephone number of the remote system. Users must also have a login ID and password on the remote system.

This section includes the following information about ATE:
- Setting Up ATE
- Customizing the ATE Program
- ATE Dialing Directory File
- Dialing-out with ATE
- Transferring a File Using ATE
- Receiving a File Using ATE
- Troubleshooting Common ATE Problems

Setting Up ATE

Before running ATE, the system administrator must install the proper software (if needed) and configure the tty ports and connections. ATE uses both direct (cabled) connections and modem connections. Local RS-232C connections allow a maximum distance of 15 meters (50 feet) between machines, and RS-422A connections allow up to 1200 meters (4000 feet) between machines.

Before using ATE to call a remote system, verify that the remote system's tty device is ready to accept a call.

Prerequisites

- ATE is an optional program product. All files necessary for operation of ATE are contained in the **bos.net.ate** program product available on the install media. Use the following commands to verify that ATE is available on your system:

  ```
  lslpp -h | more   <return>
  /bos.net.ate      <return>
  ```

  If ATE is not available on your system, install the **bos.net.ate** image from the installation media (tape, diskette, or network server).
- If ATE is installed on the system, a list of files associated with this program can be displayed using the following commands:

  ```
  lslpp -f | more   <return>
  /bos.net.ate      <return>
  ```
- The user must have root user authority to set up the port for the communications device.

Procedure

To prepare ATE to run on the system, perform the following steps:

1. Install an asynchronous adapter card in an appropriate slot in the system unit, unless the system has a built-in serial port.
2. Plug the RS-232C or RS-422A cable into the adapter card or the built-in serial port.
3. Add a tty device for the communications port using the **smit mkdev** fast path.

   Select the terminal type to emulate with ATE and make the necessary adjustments for the environment. The most common changes are line speed, parity settings, number of bits per character, and whether the line is to be driven as a remote or local line. Use bpc 8 and no parity if National Language Support (NLS) is required.
4. Set up the port for the device. To set up a port to call out with ATE, use the **pdisable** command. For example, to set up port tty1, enter:

   ```
   pdisable tty1
   ```

   To set up a port so that others can call in, use the **penable** command. For example, to let other systems call in to the tty2 port, enter:

   ```
   penable tty2
   ```
5. Ensure the device has previously been defined to the remote system. Once the device is defined, the ATE program must be customized to reflect the device settings on the remote system. Customize the default settings with the alter and modify subcommands or by editing the **ate.def** default file. To change the default settings for a telephone connection, use a dialing directory file entry.

Customizing the ATE Program

ATE creates the **ate.def** default file in the current directory the first time the user runs ATE. Edit the **ate.def** file to customize various aspects of ATE. For example, the user can change the name of the dialing directory file, the type of transfer protocols used to send and receive files from the remote system, and the baud rate ATE expects the modem to use. Refer to ″Changing the Default Files″ for more information on the **ate.def** file.

Users can also make temporary changes to certain aspects of ATE with the **modify** and **alter** subcommands. These subcommands can change all of the ATE default values except the control key sequences (which can only be changed by editing the default file) and the name of the dialing directory (which can be changed with the **directory** subcommand or by editing the default file). Any changes made with the **modify**, **alter**, or **directory** subcommands are effective only for that session of ATE. The next time the user runs ATE, the settings used are those defined in the default file.

When using a modem with ATE, the user can create a dialing directory of up to 20 phone numbers. The **directory** subcommand displays the telephone numbers in menu form and allows the user to select the desired system to call. Refer to ″The ATE Dialing Directory File″ section for more information.

By using a dialing directory, the user avoids having to look up the telephone number when calling a particular system. The user can also specify certain data transmission characteristics in the dialing directory file. This is useful if some connections use characteristics that differ from the ATE defaults.

ate.def Configuration File

The **ate.def** file sets the defaults for use in asynchronous connections and file transfers. This file is created in the current directory during the first run of ATE. The **ate.def** file contains the default values in the ATE program uses for the following:

- Data transmission characteristics
- Local system features
- Dialing directory file
- Control keys.

The first time the ATE program is run from a particular directory, it creates an **ate.def** file in that directory.

```
LENGTH         8
STOP           1
PARITY         0
RATE           1200
DEVICE         tty0
INITIAL        ATDT
FINAL
WAIT           0
ATTEMPTS       0
TRANSFER       p
CHARACTER      0
NAME           kapture
LINEFEEDS      0
ECHO           0
VT100          0
WRITE          0
XON/XOFF       1
DIRECTORY      /usr/lib/dir
CAPTURE_KEY    002
MAINMENU_KEY   026
PREVIOUS_KEY   022
```

Edit the **ate.def** file with any ASCII text editor to permanently change the values of these characteristics. Temporarily change the values of these characteristics with the ATE **alter** and **modify** subcommands, accessible from the ATE Main Menu.

Parameters in the ate.def File

Type parameter names in uppercase letters in the **ate.def** file. Spell the parameters exactly as they appear in the original default file. Define only one parameter per line. An incorrectly defined value for a parameter causes ATE to return a system message. However, the program continues to run using the default value. These are the **ate.def** file parameters:

LENGTH
Specifies the number of bits in a data character. This length must match the length expected by the remote system.

```
Options:  7 or 8
Default:  8
```

STOP Specifies the number of stop bits appended to a character to signal that character’s end during data transmission. This number must match the number of stop bits used by the remote system.

```
Options:  1 or 2
Default:  1
```

PARITY
Checks whether a character is successfully transmitted to or from a remote system. Must match the parity of the remote system.

For example, if the user selects even parity, when the number of 1 bits in the character is odd, the parity bit is turned on to make an even number of 1 bits.

```
Options: 0 (none), 1 (odd), or 2 (even)
Default: 0.
```

RATE Determines the baud rate, or the number of bits transmitted per second (bps). The speed must match the speed of the modem and that of the remote system.

```
Options: 50,75,110,134,150,300,600,1200,1800,2400,4800,9600,19200
Default: 1200
```

DEVICE
Specifies the name of the asynchronous port used to make a connection to a remote system.

```
Options: Locally created port names.
Default: tty0.
```

INITIAL
Defines the dial prefix, a string that must precede the telephone number when the user autodials with a modem. For the proper dial commands, consult the modem documentation.

```
Options: ATDT, ATDP, or others, depending on the type of modem.
Default: ATDT.
```

FINAL Defines the dial suffix, a string that must follow the telephone number when the user autodials with a modem. For the proper dial commands, consult the modem documentation.

```
Options: Blank (none) or a valid modem suffix.
Default: No default.
```

WAIT Specifies the time to wait between redialing attempts. The wait period does not begin until the connection attempt times out or until it is interrupted. If the ATTEMPTS parameter is set to 0, no redial attempt occurs.

```
Options: 0 (none) or a positive integer designating the number of seconds to wait.
Default: 0
```

ATTEMPTS
Specifies the maximum number of times the ATE program tries redial to make a connection. If the ATTEMPTS parameter is set to 0, no redial attempt occurs.

```
Options: 0 (none) or a positive integer designating the number of attempts.
Default: 0
```

TRANSFER
Defines the type of asynchronous protocol that transfers files during a connection.

p (pacing)
File transfer protocol controls the data transmission rate by waiting for a specified character or for a certain number of seconds between line transmissions. This helps prevent loss of data when the transmission blocks are either too large or sent too quickly for the system to process.

x (xmodem)
An 8-bit file transfer protocol to detect data transmission errors and retransmit the data.

```
Options: p (pacing), x (xmodem)
Default: p.
```

CHARACTER
Specifies the type of pacing protocol to be used. Signal to transmit a line. Select one character.

When the **send** subcommand encounters a line-feed character while transmitting data, the subcommand waits to receive the pacing character before sending the next line.

When the **receive** subcommand is ready to receive data, it sends the pacing character, then waits 30 seconds to receive data. The **receive** subcommand sends a pacing character again whenever it finds a carriage return character in the data. The **receive** subcommand ends when it receives no data for 30 seconds.

```
Options: any character
Default: 0
```

Interval
Number of seconds the system waits between each line it transmits. The value of the Interval variable must be an integer. The default value is 0, indicating a pacing delay of 0 seconds.

```
Default: 0.
```

NAME File name for incoming data (capture file).

```
Options: A valid file name less than 40 characters long.
Default: kapture
```

LINEFEEDS
Adds a line-feed character after every carriage-return character in the incoming data stream.

```
Options: 1 (on) or 0 (off).
Default: 0.
```

ECHO Displays the user's typed input. For a remote computer that supports echoing, each character sent returns and displays on the screen. When the ECHO parameter is on, each character is displayed twice: first when it is entered, and again when it returns over a connection. When the ECHO parameter is off, each character displays only when it returns over the connection.

```
Options: 1 (on) or 0 (off).
Default: 0.
```

VT100 The local console emulates a DEC VT100 terminal so DEC VT100 code can be used with the remote system. With the VT100 parameter off, the local console functions like a workstation.

```
Options: 1 (on) or 0 (off).
Default: 0.
```

WRITE
Captures incoming data and routes it to the file specified in the NAME parameter as well as to the display. Carriage-return or line-feed combinations are converted to line-feed characters before they are written to the capture file. In an existing file, data is appended to the end of the file.

The CAPTURE_KEY (usually the Ctrl-B key sequence) can be used to toggle capture mode on or off during a connection.

```
Options: 1 (on) or 0 (off).
Default: 0.
```

XON/XOFF
Controls data transmission at a port as follows:

- When an XOFF signal is received, transmission stops.
- When an XON signal is received, transmission resumes.
- An XOFF signal is sent when the receive buffer is nearly full.
- An XON signal is sent when the buffer is no longer full.

  ```
  Options: 1 (On), or 0 (Off).
  Default: 1.
  ```

DIRECTORY
Names the file that contains the user's dialing directory.

```
Default: the /usr/lib/dir file.
```

CAPTURE_KEY

Defines the control key sequence that toggles capture mode. When pressed, the CAPTURE_KEY (usually the Ctrl-B key sequence) starts or stops capturing (saving) the data that is displayed on the screen during an active connection.

```
Options: Any ASCII control character.
Default: ASCII octal 002 (STX).
```

MAINMENU_KEY

Defines the control key sequence that returns the Connected Main Menu so the user can issue a command during an active connection. The MAINMENU_KEY (usually the Ctrl-V key sequence) functions only from the connected state.

```
Options: Any ASCII control character.
Default: ASCII octal 026 (SYN).
```

PREVIOUS_KEY

Defines the control key sequence that displays the previous screen anytime during the program. The screen displayed varies, depending on the screen in use when the user presses PREVIOUS_KEY (usually the Ctrl-R key sequence).

```
Options: Any ASCII control character.
Default: ASCII octal 022 (DC2).  The ASCII control character is mapped to the interrupt signal.
```

ATE Dialing Directory File

The ATE dialing directory file lists phone numbers that the ATE program uses to establish remote connections by modem. Users name the dialing directory file with any valid file name and place it in any directory where read and write access is owned. Edit the dialing directory file with any ASCII text editor. The default dialing directory information for the ATE program is contained in the **/usr/lib/dir** file. (See the following figure.)

```
# @(#) 69     1.4     com/cmd/ate/dir,  bos, bos320 4/18/91  11:00:27
#
# COMPONENT_NAME:  BOS dir
#
# FUNCTIONS:
#
# ORIGINS:  27
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1989
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Control with IBM
# Corp.
#
#
# dir - sample dialing directory
#
#
Micom   9,555-9400 1200 7 1 2 0 0
R20     9,555-9491 1200 7 1 2 0 0
QT      9,555-8455 1200 7 1 2 0 0
Dallas1 9,555-7151 1200 8 1 0 0 0
```

Users can access the dialing directory information from within ATE by using the **directory** subcommand available in the ″UNCONNECTED MAIN MENU.″ The screen will show the directory information as it would appear from within the ATE program.

Users can have more than one dialing directory. To change the dialing directory file the ATE program uses, the user must modify the **ate.def** file in the current directory.

Note: The dialing directory file can contain up to 20 lines (one entry per line). ATE ignores subsequent lines.

Format of Dialing Directory File Entries

The dialing directory file is similar to a page in a telephone book that contains entries for the remote systems called with the ATE program. The format of a dialing directory entry is:

```
Name Phone Rate Length StopBit Parity Echo Linefeed
```

The fields must be separated by at least one space. More spaces can be used to make each entry easier to read. The fields are:

Name
Identifies a telephone number. The name can be any combination of 20 or fewer characters. Use the _ (underscore) instead of a blank between words in a name, for example, data_bank.

Phone
The telephone number to be dialed. The number can be up to 40 characters. Consult the modem documentation for a list of acceptable digits and characters. For example, if a 9 must be dialed to access an outside line, include a 9, (the numeral 9 and a comma) before the telephone number as follows: 9,1112222.

Although the telephone number can be up to 40 characters long, the directory subcommand displays only the first 26 characters.

Rate
Transmission or baud rate in bits per second (bps). Determines the number of characters transmitted per second. Select a baud rate that is compatible with the communication line being used. The following are acceptable rates:

50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200.

For non-POSIX baud rates, setting the rate at 50 causes the ATE to use the configured baud rate set through SMIT for that device.

Length
Number of bits that make up a character. The entry for the Length field can be 7 or 8.

StopBit
Stop bits signal the end of a character. The entry for the StopBit field can be 1 or 2.

Parity
Checks whether a character was successfully transmitted to or from a remote system. The entry for the Parity field can be 0 (none), 1 (odd), or 2 (even).

Echo
Determines whether typed characters display locally. The entry for the Echo field can be 0 (off) or 1 (on).

Linefeed
Adds a line-feed character at the end of each line of data coming in from a remote system. The line-feed character is similar in function to the carriage-return and new-line characters. The entry for the Linefeed field can be 0 (off) or 1 (on).

Notes:

1. Changing or remapping may be necessary if control keys conflict across applications. For example, if the control keys mapped for the ATE program conflict with those in a text editor, remap the ATE control keys.
2. The ASCII control character selected may be in octal, decimal, or hexadecimal format, as follows:

   octal
   000 through 037. The leading zero is required.

   decimal
   0 through 31.

   hexadecimal
   0x00 through 0x1F. The leading 0x is required. The x may be uppercase or lowercase.

Example

Create an **ate.def** file that defines those characteristics to change characteristics of ATE emulation. For example, to change the RATE to 300 bps, the DEVICE to tty3, the TRANSFER mode to x (xmodem protocol), and the DIRECTORY to my.dir, create an ate.def with the following entries, in the directory running the ATE program:

```
RATE          300
DEVICE        tty3
TRANSFER      x
DIRECTORY     my.dir
```

The program uses the defined values from time the ATE program starts from that directory.

Dialing-Out with ATE

Use the following procedure to dial out of a system using ATE and a customized **/usr/lib/dir** dialing directory file.

Prerequisites

The user should verify that all of the following prerequisites and conditions are met before attempting to dial out.

- The ATE is installed on the system.
- A modem is attached, configured, and ready for use.
- The user is a member of the UUCP group (see Setting Up ATE for more information).
- The **/usr/lib/dir** dialing directory file is already customized with the correct information.
- User's present working directory (**pwd**) contains an **ate.def** file that is properly updated.
- The **/dev/tty** port must have its ENABLE login field in SMIT set to disable, share, or delay.

Procedure

1. Enter:
   ```
   ate
   ```
2. At the main menu, type `d` and press Enter.
3. Type the file name of the directory you want to display and press Enter. To use the current directory, just press Enter.
4. Enter the appropriate directory entry number under the **#** column to dial the corresponding phone number.

Transferring a File Using ATE

Use the following procedure to transfer a file from a local host to the remote system.

Prerequisites

- A connection must already be established using the **ATE** program.
- The Xmodem file transfer protocol must already exist on both the local and remote systems. On the operating system, Xmodem is located in the **/usr/bin** directory.

Procedure

1. Run the following **xmodem** command on the remote system after logging in:
   ```
   xmodem -r newfile
   ```

   where `r` is the Xmodem flag to receive and *`newfile`* is the name of the file to be received. This name does not need to be the same as the file being transferred.
2. Press Enter.
3. The following message is displayed:

```
ate: 0828-005 The system is ready to receive file newfile. Use Ctrl-X to stop xmodem.
```

If the message is not displayed, the system may not have the **xmodem** program installed or located in its command PATH.

4. Press Ctrl-V to return to the ATE CONNECTED MAIN MENU.
5. Press `s` to send a file.
6. The following message is displayed:

   Type the name of the file you wish to send and press Enter. To use the last file name (), just press Enter.
7. Enter the name and full path of the file to be transferred.
8. Press Enter.
9. ATE will display the following message and begin to transfer the file:

```
ate: 0828-024 The program is ready to send file newfile. You will receive another
message when the file transfer is complete.
ate: 0828-025 The system is sending block 1.
ate: 0828-025 The system is sending block 2.
ate: 0828-015 The file transfer is complete.
ate: 0828-040 Press Enter
```

10. Press Enter when the transfer is complete.

Receiving a File Using ATE

Use the following procedure to receive a file transferred from a remote host.

Prerequisites

- A connection must already be established using the ATE program.
- The Xmodem file transfer protocol must already exist on both the local and remote systems. On the operating system, Xmodem is located in the **/usr/bin** directory.

Procedure

1. Run the following **xmodem** command on the remote system after logging in:

```
xmodem -s newfile
```

   where `s` is the **xmodem** command to send and *`newfile`* is the name and full path of the file to be transferred.
2. Press Enter.
3. The following message is displayed:

```
ate: 0828-005 The system is ready to send file newfile. Use ctrl-X to stop xmodem.
```

   If the message is not displayed, the system may not have the **xmodem** program installed or located in its command PATH.
4. Press Ctrl-V to return to the ATE CONNECTED MAIN MENU.
5. Press `r` to receive the file.
6. The following message is displayed:

   Type the name of the file you wish to store the received data in and press Enter. To use the last file name (), just press Enter.
7. Enter the name and full path of the file to be transferred.
8. Press Enter.
9. ATE will display the following message and begin to transfer the file:

```
ate: 0828-020 The program is ready to receive file newfile. You will
receive another message when the file transfer is complete.
ate: 0828-028 The system is receiving block 1.
ate: 0828-028 The system is receiving block 2.
ate: 0828-040 Press Enter.
```

10. Press Enter when the transfer is complete.

Troubleshooting Common ATE Problems

Problem:
When transferring or receiving files, the xmodem command appears to hang. A Ctrl-X corrects the problem.

Solution:
Examine the Alter menu to verify that xmodem protocol (or Transfer method) is being used.

Problem:
When transferring or receiving files, the file scrolls across the screen and a message is displayed stating that the transfer or receipt was complete when, in fact, it was not.

Solution:
Examine the Alter menu to verify that xmodem protocol (or Transfer method) is being used.

Problem:
When starting ATE, the user receives the following error:

```
ate: 0828-008 The system tried to open port /dev/tty0 but failed.  If the port name is not correct,
change it  using the Alter menu.  Or, take the action indicated by the system message shown below.

Connect: The file access permissions do not allow the specified action.
ate: 0828-040 Press Enter.
```

Solution:
The Connect: line in the error message narrows down the problem. Verify that the user attempting to run ATE is a member of the UUCP group. To check this, the user can enter *id* on the command line; **uucp** should appear in the output listing.

Problem:
When attempting to make a connection with ATE, the following error is received:

```
ate: 0828-008 The system tried to open port /dev/tty0 but failed. If the port name is not
correct, change it using the Alter menu. Or, take the action indicated by the system
message shown below.

Connect: A file or directory in the path name does not exist.
ate: 0828-040 Press Enter.
```

Solution:
An incorrect or unavailable tty was selected for use by ATE. Examine the Alter screen in ATE.

Problem:
The file transfers correctly, but the file size is larger than the original file.

Solution:
The xmodem protocol pads the file during transfer. To avoid this, use the **tar** command to compress the file and transfer it. This is also a means of overcoming another xmodem limitation where only one file is sent at a time. The user can **tar** several files together into a single tar image and transfer it using xmodem.

Basic Network Utilities

The Basic Network Utilities (BNU) is a group of programs, directories, and files that can be used to communicate with any UNIX system on which a version of the Unix-to-Unix Copy Program (UUCP) is running.

Note: To use baud rates higher than 38400, specify a baud rate of 50 in the **/etc/uucp/Devices** file for the desired tty, then change the SMIT configuration for that tty to reflect the actual baud rate desired.

For example. to run the **cu** command on tty0 with a baud rate of 115200, use the following procedure:

1. Ensure the hardware supports the baud rate.
2. Edit **/etc/uucp/Devices** to include the following line:
   ```
   Direct tty0 - 50 direct
   ```
3. Enter the **smit chtty** fast path.
4. Select tty0.
5. Change the baud rate to 115200.
6. Exit SMIT.

The following sections contain information about BNU:

- BNU Prerequisites
- How BNU Works
- National Support for BNU Commands
- BNU File and Directory Structure
- BNU Administrative Files and Directories
- BNU Security
- UUCP Login ID
- BNU Login IDs
- Adding BNU Login Shells to the Login.cfg File
- Security and the Systems and remote.unknown Files
- Security and the Permissions File
- BNU Daemons
- UUCP Configuration
- Setting Up Automatic Monitoring of BNU
- Setting Up BNU Polling Remote Systems
- Maintaining BNU
- Working with BNU Log Files
- Logging Files in the .Log and .Old Directories
- Other BNU Log Files
- Systemwide Log Files Used by BNU
- Using BNU Maintenance Commands
- Monitoring a BNU Remote Connection
- Monitoring a BNU File Transfer
- UUCP Conversation Flow Diagram
- UUCP Quick Setup Guide and Information Sheet

BNU Prerequisites

Before users on the system can run BNU programs, BNU must be installed and configured.

BNU is controlled by a set of configuration files that determine whether remote systems can log in to the local system and what they can do after they log in. These configuration files must be set up according to the requirements and resources of each system.

BNU must also be maintained. To maintain BNU, users must read and remove log files periodically and check the BNU queues to ensure jobs are transferring to remote systems properly. Users must also periodically update the configuration files to reflect changes in the system or remote systems.

BNU establishes communication between computer systems on local and remote networks and is on of the Extended Services programs that can be installed with the base operating system.

BNU is a version of UUCP, which was developed by AT&T and modified as part of the Berkeley Software Distribution (BSD).

BNU provides commands, processes, and a supporting database for connections to local and remote systems. Communication networks such as token-ring and Ethernet are used to connect systems on local networks. A local network can be connected to a remote system by hardwire or a telephone (modem) configuration. Commands and files can then be exchanged between the local network and the remote system.

How BNU Works

BNU uses a set of hardware connections and software programs to communicate between systems. A structure of directories and files tracks BNU activities. This structure includes a set of public directories, a group of administrative directories and files, configuration files, and lock files. Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs.

With the exception of the remote login commands, BNU works as a batch system. When a user requests a job sent to a remote system, BNU stores the information needed to complete the job. This is known as *queuing* the job. At scheduled times, or when a user instructs it to do so, BNU contacts various remote systems, transfers queued work, and accepts jobs. These transfers are controlled by the configuration files on each system and those of the remote system.

National Language Support for BNU Commands

All BNU commands, except **uucpadm**, are available for National Language Support (NLS). User names need not be in ASCII characters. However, all system names must be in ASCII characters. If a user attempts to schedule a transfer or a remote command execution involving non-ASCII system names BNU returns an error message.

BNU File and Directory Structure

BNU uses a structure of directories and files to keep track of their activities. This structure includes:

- Public directories
- Configuration files
- Administrative directories and files
- Lock files.

Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs as they run.

BNU Public Directories

The BNU public directory, **/var/spool/uucppublic**, stores files that have been transferred to the local system from other systems. The files wait in the public directory until users claim them with the **uupick** command. The public directory is created when BNU is installed. Within the public directory, BNU creates a subdirectory for each remote system that sends files to the local system.

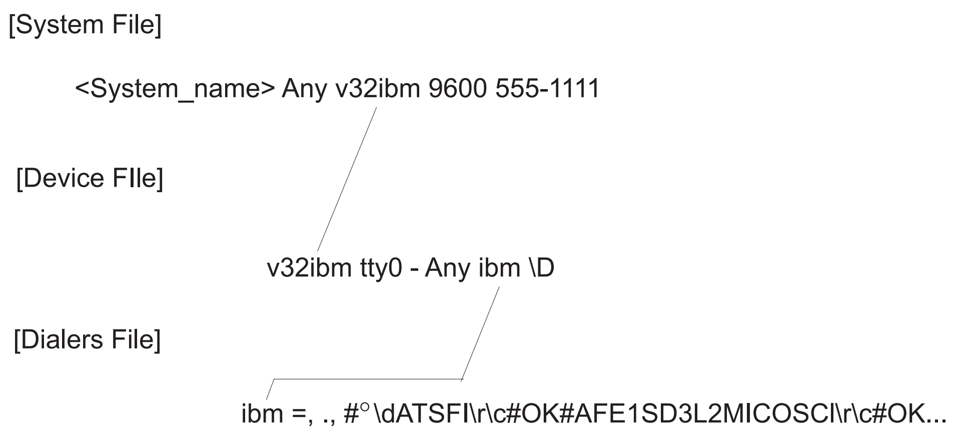
BNU Configuration Files

The BNU configuration files, also known as the BNU supporting database, reside in the **/etc/uucp** directory. The files must be configured specifically for your system. They are owned by the UUCP login ID and can be edited only with root authority. The configuration files contain information about:

- Accessible remote systems
- Devices for contacting the remote systems
- Times to contact the remote systems
- What the remote systems are allowed to do on the local host.

Some configuration files also specify limits on BNU activities which prevents the local host from becoming overloaded. The BNU configuration files include:

Devices	Contains information about available devices, including both modems and direct connections.
Dialcodes	Contains dialing code abbreviations, which allow the user to shorten phone numbers in the **Systems** file.
Dialers	Specifies calling command syntax for a specific modem type (dialer).
Maxuuscheds	Limits simultaneous scheduled jobs.
Maxuuxqts	Limits simultaneous remote command executions.
Permissions	Contains access permission codes. This file is the primary file for determining the security for BNU.
Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
Systems	Lists accessible remote systems and information needed to contact them, including the device to use and the user name and password combinations the user needs to log in. Also specifies the times when the systems can be contacted.



```
[System File]

    <System_name> Any v32ibm 9600 555-1111

 [Device FIle]

                   v32ibm tty0 - Any ibm \D

 [Dialers File]

                 ibm =, ., #° \dATSFI\r\c#OK#AFE1SD3L2MICOSCl\r\c#OK...
```

Figure 35. Correlation of BNU File

The above illustration shows that the configuration files cross-reference each other when BNU is in use. For example the:

- **Systems** file contains an entry for a *Class* of device. A device of each *Class* referred to in the **Systems** file must be defined in the **Devices** file.
- **Devices** file contains a *Token* field that refers to entries in the **Dialers** file.
- **Poll** file contains entries for systems the host system calls. Each of these systems must be defined in the **Systems** file.

Entries in the BNU configuration files depend on the types of connections between the user's host system and each remote system. For example, special entries must be made if TCP/IP or *direct* connections are used to contact other systems. If modems are used to contact other systems, the modems must be defined in the **Dialers** file.

The **Systems**, **Devices**, and **Permissions** files must be configured on the user's host system before they can contact remote systems using BNU. Other configuration files enable the user to use BNU capabilities, such as automatic polling. Many of the configuration files must be modified periodically to reflect changes to the user's host or remote systems.

Systems File:
The **/etc/uucp/Systems** file contains the primary information that is used by the **uucico** program to connect to the various systems. Users should consider this file the database of systems where their machine can connect.

system_name
Specifies a name that the BNU caller program (**uucico**) uses to associate the remote system to the dial-up and login information from the user's **Systems** file database. The **Systems** file contains all the information needed to call a specific system, and login as a specific user.

When **uucico** calls the remote system, it will actually login (like any other user) into a special user account that will invoke as its login shell. When the remote system invokes its login shell, it will present its *uuname* (see the **uuname** command) to the system calling in.

That *uuname* must match the name in the **Systems** file of the host calling in. If they do not match, the system calling in will report a `WRONG MACHINE NAME` message and the call will fail.

Call_time
Indicates the *time-of-day* that the user's system can call out to the specified remote site. Normally (for testing), this entry will read: **Any**, which means ″I can call this system **Any** time of the day.″

Other valid entries are:

- **Never**
- Day of the week in the format ″MoTuWeThFrSaSu″
- Weekdays only, indicated by ″Wk″
- Time of day can be specified by 24 hour times 1800 (6:00pm) 0800 (8:00am), and time of day can be used in ranges by listing two times separated by a dash (-) such as: 0600-1800.

Device
Specifies the name of a device that is listed in the BNU **Devices** file. There is no restriction on the name, other than it being no longer than 7 characters long and the fact that it must exist in the **Devices** file. If users do *not* wish to use the **Devices** file, they may use the dash (-) entry which indicates a valid **NULL** device.

speed Indicates the connection speed for serial line communications. If the system is using a direct connect serial line (that is, no modem just one host connected serially to the other) or modem lines, this field is required. If the system is using a **TCP** connection device type (a special **Devices** file type), this field should be a dash (-).

Chat script
Indicates the BNU **Systems** file entries. The UUCP **chat** utility is what allows the user's machine to login to other systems and start a BNU connection without user intervention. This sequence is broken into *send* and *expect* fields. For every *send*, there is a corresponding *expect* (until the last one). Basically it works like this: ″I send″, ″the other end responds″; ″I send″, ″the other end responds″, etc. until I am logged in. It can be thought of in this way: ″what would I type, and what would the remote modem or system respond with, if I were going to login to that system″. Normally this sequence starts with the phone number of the remote system (if the remote system is to be connected over a modem); and is then followed by the sequence used to login to the remote system. The login sequence will look something like:

```
in:--in: username word: password
```

login_sequence
Begins: `in:–in:` which is the last 3 digits in the **login:** prompt separated by two dashes (—) used to signify:

If I don’t get this within my default time out period, send the characters between the dashes (-) plus a carriage return and then wait for **in:** again. In the following example, wait for `in:`. If we don’t see it, send `hello` and wait for `howdy:` in response:

```
in:-hello-howdy:
```

The next part of the chat sequence is the *username* of the remote site’s UUCP account. This can be any login account the remote system administrator wants to set up to do UUCP, but the default program that it calls MUST be **/usr/sbin/uucp/uucico**.

Some sites accomplish this by creating a normal user and making the first line of the user’s **.profile** call **uucico**, but it is not recommended. The default directory of this user is usually **/usr/spool/uucppublic** but it is not a requirement.

The next part of the chat sequence is `word:` which is the last five characters of the `password:` prompt. If the remote system were to say `Enter your secret code:` instead of prompting for the `password:`, this entry would be `code:` in the chat sequence.

The last entry in the chat sequence is usually the actual password of the account you will be using for UUCP on the remote system. This entry should result in a completed login and direct access to the **uucico** program. Some systems have two (or more) passwords per account, so the user would simply add more *expect - send* entries onto the chat sequence to show this.

Refer to ″Security and the Systems and remote.unknown Files″ for more information on the **Systems** file.

Note: For quickest setup, it is probably easiest to copy the existing sample entry in the **Systems** file, and replace the system name, phone number, and account specific information (login name and password) from this default entry.

Permissions File:
The **/usr/lib/uucp/Permissions** file is used to create security control over machines attempting to communicate with the local host. This file has two distinct entries each of which does a different job. It should be noted that although the entries appear together in the file, they are *not* related to one another.

The first entry in the **Permissions** file is the `MACHINE=` entry. This entry is used to establish the base permissions for a given sitename.

MACHINE=
Indicates the remote systems that are allowed to access the user’s machine. This entry *must* have a corresponding *system_name* listed in the **/etc/uucp/Systems** file or it is ignored.

Several system names may be separated by colons (:) on one entry (all systems entered this way will have the same permissions, as dictated by the rest of the entry’s fields).

REQUEST=
Indicates whether the users from remote systems may initiate requests to perform BNU jobs on this system. A **yes** in this field also requires a **yes** value be placed in the `REQUEST=` field for the `LOGNAME=` entry in this file (this line will be discussed later).

An answer of **yes** means that remote users may spool up jobs for this system, and request files be transferred to and from this system. For added security, it is recommended that `REQUEST=no` be used so that only this machine can initiate job requests.

READ=
Indicates the directories, separated by colons (:), that the remote system may have read access to on this system. If the remote system is given access to the root directory (**/**), then remote users will be able to read *all* directories on this machine. The remote users will be subject only to the normal operating system file permissions

Note: Giving the remote system read access to the root directory (**/**) will allow it to read only those directories accessible to the UUCP account.

If the intention is to restrict access to only the **uucppublic** directory and the **tmp** directory, the `READ=` field would resemble the following:

```
READ=/var/spool/uucppublic:/tmp
```

WRITE=
Indicates which directories (separated by colons (:)) that the remote system may write to on this system. This works the same as the `READ=` field above, but determines write access.

COMMANDS=
Indicates the command that the remote system may execute on this system, separated by colons (:). These commands can be the command names alone or the complete path to the commands. To execute the commands listed here, however, the UUCP user account on this system must have execute permission. This field does note override normal file permissions permitting execution on this system.

The second line type in the **Permissions** file is used to establish the permissions for the login account that the remote system is using. Several of the fields for this line are the same as the `MACHINE=` fields of the same name.

The primary difference is that these entries are specific to the login account that the remote system is using, while the `MACHINE=` fields are specific to the remote system's *uuname*. This field is the `LOGNAME=` entry.

LOGNAME=
Indicates the username, separated by colons (:), which remote systems will use when logging in to this machine. When a remote system logs into this machine, its username will be checked against the **Permissions** file for an entry that has `LOGNAME=` as the first characters on the line. After **uucico** checks the `LOGNAME=` line, the permissions for that remotely logged in system will be set to the the permissions defined on the rest of the `LOGNAME=` line.

REQUEST=
Specifies permissions based on the login name, as opposed to the system name. It is exactly the same as the `MACHINE=` line entry of the same name.

SENDFILES=
Indicates when the local system may send files to a remote system. It is only used on the `LOGNAME=` line and can be set to either **yes** or **call**. When the field is set to **yes**, the local system may send files (and/or jobs) to the remote system when either system initiates the call.

When the field is set to **call**, the local system may only send files or jobs when the local system initiates the call.

In other words with `SENDFILES=` set to **call** this system can only receive files when the remote machine calls in. With this setting the local system can send and receive files only when it initiates the call to the remote machine.

READ=
Indicates the same as the `MACHINE=` field of the same name, but it sets permissions based on the login name, as opposed to the system name.

WRITE=
Indicates the same as the `MACHINE=` field of the same name, but it sets permissions based on the login name, as opposed to the system name.

Devices File:
The **/etc/uucp/Devices** file is used by several programs (including **uucico**) to determine which device on the system to use for a given connection attempt. For example, the **cu** program uses entries in this file for

Direct access to the tty port, Serial Line Internet Protocol (SLIP) uses the **Devices** file for determining access to tty's for TCP/IP connections over serial line, and ECS (the IBMLINK connection facility) uses the **Devices** file to make modem connections to IBMLINK.

The basic purpose of the **Devices** file is to define the device type, location, speed, and other communication parameters necessary for the dial-out programs (it is only used for dial-out connections).

device_name
Specifies a *user specified* name for the device used in dial out connections. Any name may be used in this field except for the reserved words **Direct** and **TCP** which are used for direct connect serial lines (hard-wired; no modems) and connections over a network involving TCP/IP, respectively.

By standardization the *device_name* **ACU** is used for modem dial-out connections. The name ACU stands for **Automatic Calling Unit**. An ACU device used to be required for autodialing the telephone numbers for modems. The device would be attached to a second line and would dial the phone for the modem, and then attach the telephone line to the modem for communication with the remote modem/system.

connection_port
Specifies the physical device to be used by UUCP. For serial communications, the entry is the name of a tty device (for example, tty2). If a *device_name* of **TCP** is specified, the `connection_port` entry should be a dash (-).

- Early versions of UUCP used this field to indicate which tty the autodialer, or ACU, was attached to. This entry became less critical however, as modem technology progressed. Now that modems can dial without the use of ACUs, the entry is no longer needed. A dash (-) is used here only as a place-holder.

speed Indicates the speed in bits per second (bps) of a serial line connection. If the *device_name* is **TCP** this field will be a dash (-). If the user does not wish to limit the speed of this device to a specific bps rate, they may specify the speed in the **Systems** file and set this field (in the **Devices** file) to **Any**. This allows the same device to be used for several different systems that need to connect at different speeds.

dialer Indicates the type of *dialer* (as specified in the **/etc/uucp/Dialers** file) that this device will use. Several default dialers are specified in the **Dialers** file, they are:

```
hayes, penril, ventel, rixon, vadic, micom, TCP, and direct
```

Each dialer specifies a different command set to use when attempting to dial the modem. Although several dialers are listed, the most commonly used entries are: **hayes**, **direct**, and **TCP**.

The default **hayes** entry, shown below, seems cryptic at first, but it can easily be translated using the information provided.

```
hayes =,-, "" \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Escape Characters for the UUCP Dialers File

\c	No carriage return or newline.
\d	Delay (1 to 2 seconds depends on UUCP version - BNU or BSD).
\D	Telephone number (do not use Dialcodes translation).
\e	Disable echo checking.
\E	Enable echo checking (use for slower devices).
\K	Insert a Break character (BNU only).
\n	Send a newline.
\p	Pause.

\r	Send carriage return.
\s	Send a space character.
\T	Send telephone number without Dialcodes translation.
\\	Send a backslash (\\) character.
=,-	Change an equal (=) or dash (-) in UUCP dial string to comma (,). Hayes uses a double quotation mark (″) for *pause*.
″	Expect nothing (null string).
EOT	Send an end-of-transmission character.

System administrators who are familiar with modem configuration could use these escape characters to build a command string capable of programming even the most complicated modems. The following Dialers entry is an example of one such string used to program a Hayes-compatible, 9600 baud modem.

Note: The following lines should be combined into a single line:

```
HayesProgrm9600 =,-, "" \d\dAT\r\c OK AT&F\r\c OK
ATM1\r\c OK AT&D3\r\c OK AT&K3&C1\r\c OK ATL0E0Q2\r\c OK
ATS0=1\r\c OK AT&W\r\c OK
```

The **direct** dialer is only a **dummy** entry, in that it does not specify any command set. Use of the direct dialer is limited to direct (non-modem) serial connections only.

The **TCP** dialer is effectively the same as the **direct** dialer, but is a keyword that TCP/IP and UUCP connections look for when making a call.

token This field indicates whether the **Devices** file should send the phone number *as it is listed* in the **Systems** file, or whether it should interpret the number through the **/usr/lib/uucp/Dialcodes** file first.

\D	Indicates that the number should be checked with the **Dialcodes** file first.
\T	Indicates that the number should be passed straight through to the **Dialers** file without interpretation.

The **Dialcodes** file is used to make *standardized* names for certain parts of a phone number. For example, if you made a lot of calls to a certain area code in San Francisco, you could create a Dialcodes entry that reads: `SFO 9,1415`. Which would be used in the phone number dial string like this: instead of - `9,14155551111`, you would use - `SFO5551111`. The `SFO` would be changed by the **Dialcodes** file for automatically.

It should be noted that users often include the **\D** token even though the **Dialcodes** file is rarely used by UUCP administrators.

BNU Administrative Files and Directories

The BNU administrative directories and files are in subdirectories of the **/var/spool/uucp** directory. These directories and files contain two types of information:

- Data waiting to be transferred to other systems.
- Log and error information about BNU activities.

Under the **var/spool/uucp** directory, BNU creates the following seven directories:

Admin
Contains the following four administrative files that retain error and log information about BNU activities:

- audit
- Foreign
- errors
- xferstats.

Corrupt
Contains copies of files that cannot be processed by the BNU program.

Log Contain log files from BNU transactions.

Old Contain log files from BNU transactions.

Status
Stores the last time the daemon tried to contact remote systems.

Workspace
Holds temporary files that the file transport programs use internally.

Xqtdir Contains execute files with lists of commands that remote systems can run.

Directories are also created under the **/var/spool/uucp** directory which contain the system_name of each remote system the host contacts. Note that the names of all the above directories start with a period (.). This means that they cannot be found with an **ls** command unless the **-a** flag is used. When the daemon is started, it searches the **/var/spool/uucp** directory for work files and transfers the files from any directory that is not hidden. The daemon sees only the system_name directories, not the other administrative directories.

The files in the hidden directories are owned by the UUCP login ID. These files can be accessed only with root authority or with a login ID which has a user ID (UID) of 5.

BNU Lock Files

The BNU lock files are stored in the **/etc/locks** directory. When BNU uses a device to connect to a remote computer, it places a lock file for that device in the **/etc/locks** directory. When another BNU program or any other program needs the device, that program checks the **/etc/locks** directory for a lock file. If a lock file exists, the program waits until the device is available or uses another device for the communication.

In addition, the daemon places lock files for remote systems in the **/etc/locks** directory. Before contacting a remote system, the daemon checks the **/etc/locks** directory for a lock file for that system. These files prevent other instances of the daemon from establishing duplicate connections to the same remote system.

BNU Security

Because other systems contact the user's system to login, transfer files, and execute commands, BNU provides a means to establish security. BNU security enables the user to restrict what users of remote systems can do on the local host. BNU runs several daemons to complete its activities and uses administrative directories to store the files it needs. BNU also keeps a log of its own activities.

BNU security works on several levels. When the user configures BNU, they can determine:

- Who on this system has access to BNU files.
- What remote systems this system can contact.
- How users on remote systems login to this system.
- What users on the remote systems can do on this system once logged in.

UUCP Login ID

When BNU is installed, all of the configuration files, daemons, and many of the commands and shell procedures are owned by the UUCP login ID. The UUCP login ID has a user ID (UID) of 5 and a group IDD (GID) of 5. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file to schedule automatic jobs for BNU.

Usually, logging in as user UUCP is not allowed. To change files that are owned by the UUCP login ID, log in with root authority.

> **Note:** Allowing remote systems to log in to the local system with the UUCP login ID seriously jeopardizes the security of the local system. Remote systems logged in with the UUCP ID can display and possibly modify the local **Systems** and **Permissions** files depending on the other permissions specified in the `LOGNAME` entry. It is strongly recommended that the UUCP administrator create other BNU login IDs for remote systems and reserve the UUCP login ID for the person responsible for administering BNU on the local system. For the best security, each remote system that contacts the local system should have a unique login ID with a unique UID number.

BNU Login IDs

The startup shell for BNU login IDs is the daemon **/usr/sbin/uucp/uucico**. When remote systems call your system, they automatically start the daemon on your system. Login IDs for BNU have a UUCP group ID of 5.

Login IDs used by remote systems need passwords. To prevent security from prompting a new BNU login ID for a new password when the remote system logs in, you must set the password as soon as you create the account. To do this, use the **passwd** command followed by the **pwdadm** command.

To set a password for the login ID `nuucp`, log in as the root user and enter the following commands:

```
passwd nuucp
pwadm -f NOCHECK nuucp
```

The system prompts you for a password for the `nuucp` login ID. Completing these steps allows the remote system to log in without being immediately prompted for a new password (which the batch-oriented `nuucp` login cannot provide).

After creating the login ID for a remote system, notify that system's BNU administrator and give them the login ID and password to access your system.

Creating a BNU Administrative Login ID

A user with root authority can set up a BNU administrative login ID. This is useful if you wish to delegate BNU administration duties to a user without root authority. The BNU administrative login ID should have password security, a UID of 5, and be in a UUCP group with an ID of 5.

The login shell for the administrative login should be the **/usr/bin/sh** program (instead of the daemon). Giving the BNU administrative login a UID of 5 causes it to have the same privileges as the UUCP login ID. Thus, for security, remote systems should not be allowed to login as the BNU administrator.

Adding BNU Login Shells to the login.cfg File

User configuration stanzas in the **/etc/security/login.cfg** file provide configuration information for programs that change user attributes or add new users. There are two stanzas of this type:

```
pw_restrictions
usw
```

The shells attribute in the `usw` stanza defines the valid shells on the system. The value is a list of comma-separated full path names such as:

```
/usr/bin/sh,/usr/bin/bsh,/usr/bin/csh,/usr/bin/ksh
```

Before using the System Management Interface Tool (SMIT) to add a new BNU user, add the program name **/usr/sbin/uucp/uucico** to the `usw` shells stanza. The new program name should be separated from the last entry by a comma and no blanks.

```
/usr/bin/sh,/usr/bin/bsh,/usr/bin/csh,/usr/bin/ksh,/usr/sbin/uucp/uucico
```

Note: SMIT will fail when specifying **/usr/sbin/uucp/uucico** as a user's login shell if the program name is not added to the **login.cfg** file located in the **/etc/security** directory.

Security and the Systems and remote.unknown Files

On most BNU systems, only remote systems listed in the **/etc/uucp/Systems** file can log in to the local system. The **/usr/sbin/uucp/remote.unknown** script is executed whenever an unknown system attempts to call the local system. This script refuses to let the unknown system log in, and makes an entry in the **/var/spool/uucp/.Admin/Foreign** file recording the time of the login attempt.

With root authority, or as a BNU administrator, you can modify the **remote.unknown** shell procedure to log more information about the remote system or to store the information in a different file. For example, you can modify the shell procedure to send mail to the BNU administrator whenever an unknown system tries to log in.

By taking away execute permissions on the **remote.unknown** shell procedure you enable unknown machines to log in. In this case, you should add a `MACHINE=OTHER` entry to the **/etc/uucp/Permissions** file to establish permissions for the unknown machines.

Your system can contact only remote systems listed in the **Systems** file. This prevents users on your system from contacting unknown systems.

Security and the Permissions File

The **/etc/uucp/Permissions** file determines:

- Remote login user names for logging in to the local system.
- Approved commands and privileges for remote systems logging in to the local system.

The **/etc/uucp/Permissions** file contains two type of entries:

`MACHINE`	Defines machine names and the privileges associated with them. `MACHINE` entries take effect when the local system contacts a remote system.
`LOGNAME`	Defines login names and the privileges associated with them. `LOGNAME` entries take effect when a remote system calls the local system and attempts to log in.

Options in the **Permissions** file enable you to establish various levels of security for each remote system. For example, if many remote systems share one login ID on the local system, use the `VALIDATE` options to require each remote system to use a unique login ID. The `SENDFILES`, `REQUEST`, and `CALLBACK` options specify which system has control, keeping the local system in control of transactions if necessary.

The `READ`, `WRITE`, `NOREAD`, and `NOWRITE` options define access to specific directories on the local system. These options also control where on your system remote users can place data. The `COMMANDS` option limits the number of commands users on remote systems can execute on the local system. The `COMMANDS=ALL` option allows total privileges to remote systems but seriously jeopardizes the security of your system; it should therefore be used only when necessary.

BNU Daemons

The BNU software includes four daemons stored in the **/usr/sbin/uucp** directory:

uucico
Facilitates file transfers.

uusched
Facilitates work request scheduling of files queued in the local spooling directory.

uuxqt Facilitates remote command executions.

uucpd Facilitates communications using TCP/IP.

uucico, **uusched**, and **uuxqt** are started by the **cron** daemon according to a schedule set by the BNU administrator. With root authority, you can also start these daemons manually. The **uucpd** daemon should be started by the TCP/IP **inetd** daemon.

Using the uucico Daemon

The **uucico** daemon transports the files required to send data from one system to another. The **uucp** and **uux** commands start the daemon to transfer command, data, and execute files to the designated system. The **uucico** daemon is also started periodically by the BNU scheduler, the **uusched** daemon. When started by the **uusched** daemon, the daemon attempts to contact other systems and execute the instructions in the command files.

How the Daemon Process Begins:
To run the instructions in the command files, the **uucico** daemon first checks the **/etc/uucp/Systems** file for the system to be called. The daemon then checks the **Systems** file entry for a valid time to call. If the time is valid, the daemon checks the *Type* and *Class* fields and accesses the **/etc/uucp/Devices** file for a device that matches.

After finding a device, the **uucico** daemon checks the **/etc/locks** directory for a lock file for the device. If one exists, the daemon checks for another device of the requested type and speed.

When no device is available, the **uucico** daemon returns to the **Systems** file for another entry for the remote system. If one exists, the daemon repeats the process of searching for a device. If another entry is not found, the daemon makes an entry in the **/var/spool/uucp/.Status/***SystemName* file for that remote system and goes on to the next request. The command file remains in the queue. The daemon attempts the transfer again at a later time. The later attempt is called a *retry*.

When the Daemon Reaches the Remote System:
When the **uucico** daemon reaches the remote system, it uses the instructions in the **Systems** file to log in. This causes an instance of the daemon to be invoked on the remote system as well.

The two daemons, one on each system, work together to make the transfer. The daemon on the calling system controls the link, specifying the requests to be performed. The daemon on the remote system checks the local permissions for whether they allow the request to be performed. If so, the file transfer starts.

After the daemon on the calling system has finished transferring all requests it has for the remote system, it sends a hangup request. When the remote daemon has transactions to send to the calling system, it denies the hangup request, and the two daemons reverse roles.

Note: Either the **/etc/uucp/Permissions** file on the local system or the **/etc/uucp/Permissions** file on the remote system can forbid the daemons to reverse roles, In this case, the remote system must wait to transfer files until it calls the local system.

When nothing is left to be transferred in either direction, the two daemons hang up. At this point, the **uuxqt** daemon is called to execute remote command requests.

Throughout the transfer process, the daemons on both systems log messages in the BNU log and error files.

Using the uusched Daemon

The **uusched** daemon schedules the transfer of files that are queued in the spooling directory on the local system. The spooling directory is **/var/spool/uucppublic**. When the **uusched** daemon is invoked, it scans the spooling directory for command files, then randomizes the files and starts the daemon. The daemon transfers the files.

Using the uuxqt Daemon

When a user issues the **uux** command to run a specified command on a designated system, the **uuxqt** daemon executes the command. After creating the necessary files, the **uux** command starts the daemon, which transfers those files to the public spooling directory on the specified system.

The **uuxqt** daemon periodically searches the spooling directory for command-execution requests on every connected system. When it locates such a request, the **uuxqt** daemon checks for necessary files and permissions. Then, if permitted, the daemon executes the specified command.

Using the uucpd Daemon

The **uucpd** daemon must be runnable on the remote system before BNU can establish communications with a remote computer with Transmission Control Protocol/Internet Protocol (TCP/IP). The **uucpd** daemon is a subserver of the TCP/IP **inetd** daemon and is started by the **inetd** daemon.

By default, the **inetd** daemon is configured to start the **uucpd** daemon. However, if this has been changed on your system, you may need to reconfigure the **inetd** daemon to start the **uucpd** daemon.

UUCP Configuration

Prerequisites

- BNU must be installed on the system. You must have root authority to edit the BNU configuration files.
- If you are using direct connections for BNU communications, the appropriate hard-wired connections between your system and the remote systems must be set up.
- If modems are used for the BNU connection, they must be installed and configured.
- If the connection uses TCP/IP, then TCP/IP must be running between your system and the remote site.

Configuring UUCP

To configure UUCP on your system, perform the following steps:

1. Edit the **/etc/security/login.cfg** file making the following changes:
   - Find the `shells =` variable in the `usw:` stanza.
   - Add `,/usr/lib/uucp/uucico` after the last entry.

   User configuration stanzas in the **/etc/security/login.cfg** file provide configuration information for programs that change user attributes or add new users. This entry must be made before attempting to add or change a user's login shell to `uucico`.

   ALTERNATE METHOD:

   The UUCP login herald may also be customized for a specific tty port. The system's default herald contains control sequences which may cause a **uucico** process to terminate login attempt. UUCP errors such as `Enough already` may indicate such a problem.

To avoid this situation the administrator should make the herald as short as possible. Use the following steps to accomplish this:

- In the **/etc/security/login.cfg** file, comment out the *default* stanza and its herald by placing an (*) in the leftmost column of each line.
- Add the following entries before the *default* stanza, substituting the correct tty number in place of `tty0`:

  ```
  /dev/tty0: herald = "\nuucp login: "
  ```

2. Create a UUCP administrator account. Edit the **/etc/passwd** file adding the following line after the **uucp** entry:

   ```
   uucpadm:!:5:5::/usr/lib/uucp:/bin/ksh
   ```

   This creates a user with the same group and user ID as UUCP which is helpful for BNU administration and debugging.
3. Add other machine logins. You have the option of maintaining separate logins, or having one login for all UUCP connections. The rule of thumb here is that if you need to maintain complete control over access by each individual machine, you must create separate login IDs as well as combine the `MACHINE` and `LOGNAME` entries in the **Permissions** file. A few examples are shown below:

   ```
   Umicrtk:!:105:5:micrtk
   uucp:/usr/spool/uucppublic:/usr/lib/uucp/uucico
   Ufloydl:!:106:5:floydl
   uucp:/usr/spool/uucppublic:/usr/lib/uucp/uucico
   Uicus:!:107:5:icus
   uucp:/usr/spool/uucppublic:/usr/lib/uucp/uucico
   Urisctkr:!:108:5::/usr/spool/uucppublic:/usr/lib/uucp/uucico
   ```

   If you want to have one set of permissions and do not want to maintain separate control for any of the UUCP connections, you can have a single login for all the machines. For example:

   ```
   nuucp:!:6:5::/usr/spool/uucppublic:/usr/lib/uucp/uucico
   ```

   Field requirements:

   The user ID (the third colon-separated field) must be unique to avoid would be a security risk. The group ID (the fourth colon-separated field) *must* be a 5, which is the same group as UUCP. You can change the home directory (the sixth field) to any valid directory, but the login shell (the seventh field) *must* be **/usr/lib/uucp/uucico**.
4. Make sure that the **/etc/group** file contains the new users added in step 3. For example:

   ```
   uucp:!:5:uucp,uucpadm,nuucp,luucp,Umicrtk,Ufloydl,Uicus,
   Urisctakr,ayalad
   ```

   Take this time to add any other users to the UUCP group who will be using modems to dial out with programs other than the **cu** command. The user name `ayalad` was added at the end of the UUCP group as an example. The user name `luucp` is added as a login name for systems being granted *limited* machine access.
5. After editing these files as **root**, set up a password for the new users with the command:

   ```
   passwd UserName
   ```
6. After changing the passwords, it is necessary to edit the `flags =` entry in the **/etc/security/passwd** file for each new UUCP user. Locate the following `flags` entry for each UUCP user:

   ```
   flags = ADMCHG
   ```

   Change it to:

   ```
   flags =
   ```

   Otherwise, when the remote **uucico** logs in, it will be prompted to enter a new password, which it cannot do. Hence the login will fail.

7. Activate **cron**. After logging in as **uucpad**, run the following command to read the current crontab for UUCP into a temporary file:

   ```
   crontab -l > /tmp/cron.uucp
   ```

   Next, edit **/tmp/cron.uucp** to uncomment entries. They should look like following:

   ```
   20,50 * * * /bin/bsh -c "/usr/lib/uucp/uudemon.poll >/dev/null"
   25,55 * * * /bin/bsh -c "/usr/lib/uucp/uudemon.hour> /dev/null"
   45,23 * * /bin/bsh -c "/usr/lib/uucp/uudemon.cleanu > /dev/null"
   48,8,12,16 * * /bin/bsh -c "/usr/lib/uucp/uudemon.admin > /dev/null"
   ```

   Changed the entries to suit your needs. Afterwards, read in the edited version into UUCP's crontab with this command:

   ```
   crontab /tmp/cron.uucp
   ```
8. Verify that your changes took effect by running the command:

   ```
   crontab -l
   ```

   Refer to *AIX 5L Version 5.1 System Management Guide: Operating System and Devices* for more information on **cron**.
9. Set up the BNU data files: **Systems**, **Devices**, **Permissions**, and **Dialers**. You can use the **/usr/sbin/uucp/uucpadm** command to initially set up the files, and then edit them to suit your exact needs.
10. Use the **uucheck** command to verify that everything is in place:

    ```
    /usr/sbin/uucp/uucheck -v
    ```

    The **uucheck** command verifies that the directories, programs, and support files are set up properly and that the **Permissions** file entries are consistent. Correct any errors reported by the **uucheck** command.

BNU Example

The following BNU file entries were made using the examples shown in this chapter. An optional entry is included to show how to program a modem (a Telebit T3000 in this case).

The **Systems** file:

```
Umicrtk Any ACU 9600 9,15551111 "" \d\r in:--in: nuucp word: nuucp
Ufloydl Any ACU 9600 9,15551234 "" \d\r in:--in: nuucp word: nuucp
Uicus Any ACU 9600 9,15554321 "" \d\r in:--in: luucp word: luucp
Urisctkr Any ACU 2400 9,15559876 ""\d\r in:--in: luucp word: luucp
telebit Nvr TELEPROG 19200
```

The **Devices** file:

```
ACU tty1 - 9600 hayes \D
ACU tty1 - 2400 hayes \D
TELEPROG tty1 - 19200 TelebitProgram
```

The **Dialers** file:

```
hayes =, -, "" \dAT\r\c OK \pATDT\T\r\c CONNECT
# NOTE: The following 4 lines should be made into one long line:
TelebitProgram =,-, "" \dAT&F\r\c OK
ats0=1s2=255s7=60s11=50s41=2s45=255s51=252s63=1s58=2s64=1\r\c OK
ATs69=2s105=0s111=30s255=0M0&C1Q2&D3&Q0&R3&S1&T5\r\c OK ATE0X12&W\r\c OK 00
```

The **Permissions** file:

```
MACHINE=Umicrtk:Ufloydl REQUEST=yes READ=/ WRITE=/
COMMANDS=ALL MACHINE=Uicus:Urisctkr REQUEST=yes READ=/ WRITE=/  \
COMMANDS=vi:ls:rm
LOGNAME=nuucp REQUEST=yes SENDFILES=yes READ=/ WRITE=/
LOGNAME=luucp REQUEST=yes SENDFILES=yes  \ READ=/usr/spool/uucppublic:/tmp  \
WRITE=/usr/spool/uucppublic:/tmp
```

Setting Up Automatic Monitoring of BNU

Prerequisites

- BNU users and files must be correctly configured.
- Root authority is required to edit the **/var/spool/cron/crontabs/uucp** file.

Configuration

BNU uses the **cron** daemon to start BNU daemons and to monitor BNU activity. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file for instructions about when to start BNU procedures.

1. Log in as a user with root authority.
2. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file.
3. Uncomment the lines for BNU maintenance procedures, **uudemon.admin** and **uudemon.cleanup**. You can change the times these procedures are run if the system needs maintenance at more or less frequent intervals. It is best to run the **uudemon.admin** command at least once a day and the **uudemon.cleanup** command at least once a week.
4. You can use the **crontabs/uucp** file to schedule other BNU maintenance commands, such as the **uulog**, **uuclean**, or **uucleanup** commands. In addition, you can use the **/var/spool/cron/crontabs/uucp** file to instruct the **cron** daemon to start the **uucico**, **uuxqt**, or **uusched** daemons at specific times.

Setting Up BNU Polling of Remote Systems

Prerequisites

- BNU users and files must be correctly configured.
- Root authority is required to edit the **/var/spool/cron/crontabs/uucp** file and the **/etc/uucp/Poll** file.

Configuration

To enable BNU to poll remote systems for jobs, list the systems in the **/etc/uucp/Poll** file. In addition, run the **uudemon.hour** and **uudemon. poll** commands periodically.

1. Decide which remote systems to automatically poll. Decide how often you want to poll each one. Specify times for each system with the **Poll** file as seldom as once a day or as often as you wish.
2. Log in as a user with root authority.
3. Using an ASCII text editor or the **uucpadm** command, edit the **Poll** file. Add an entry for each system your system will poll. Any systems listed in the **Poll** file *must* also be listed in the **/etc/uucp/Systems** file.
4. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file. Remove the comment characters (#) from the lines that run the **uudemon.hour** and **uudemon.poll** commands. You can change the times these commands are run; however, be sure to schedule the **uudemon.poll** command approximately five minutes *before* you schedule the **uudemon.hour** command.

BNU will now automatically poll the systems listed in the **Poll** file at the times you have specified.

Maintaining BNU

BNU must be maintained to work properly on your system. The following checks should be performed in order to maintain the Basic Network Utilities:

- Read and remove log files periodically.

- Use the **uuq** and **uustat** commands to check the BNU queues to ensure jobs are transferring to remote systems properly.
- Schedule automatic commands which poll remote systems for jobs, return unsent files to user, and send you periodic messages about BNU status.
- Periodically update the configuration files to reflect changes in your system.

In addition, occasionally check with administrators of remote systems to keep up with changes on their systems that may affect your configuration. For example, if the supervisor of system `venus` changes your system's password, you will need to put the new password in the **/etc/uucp/Systems** file before your system can log in to system `venus`.

Working with BNU Log Files

BNU creates log files and error files to track its own activities. These files must be checked and removed periodically to keep them from filling the storage space on your system. BNU provides several commands for use in cleaning log files:

- **uulog**
- **uuclean**
- **uucleanup**
- **uudemon.cleanu**

Run these commands manually or use entries in the **/var/spool/cron/crontabs/uucp** files to run the commands by the **cron** daemon.

Logging Files in the .Log and .Old Directories

BNU creates individual log files in the **/var/spool/uucp/.Log** directory. A log file is created for each accessible remote system, using the **uucp**, **uuto**, or **uux** command. BNU places status information about each transaction in the appropriate log file each time someone on the system uses BNU. When more than one BNU process is running the system cannot access the log file. Instead, it places the status information in a separate file with a **.Log** prefix.

The **uulog** command displays a summary of **uucp** or **uux** requests, by user or by system. The uulog command displays the files. However, you can also have BNU automatically combine the log files into a primary log file. This is called *compacting* the log files, and can be done with the **/usr/lib/uucp/uudemon.cleanu** command (usually run by the **cron** daemon).

The **cron** daemon runs the **uudemon.cleanu** command, which combines the uucico and uuxqt log files on the local system and stores them in the **/var/spool/uucp/.Old** directory. By default, the **uudemon.cleanu** command saves log files that are two days old.

If storage space is a problem, consider reducing the number of days that files are kept. To track BNU transactions over a longer period of time, consider increasing the number of days that files are kept. To change the default time for saving log files, modify the shell procedure for the **uudemin.cleanu** command. This script is stored in the **/usr/sbin/uucp** directory and can be modified with root authority.

Other BNU Log Files

BNU also collects information and stores it in the **/var/spool/uucp/.Admin** directory. This directory contains the **errors**, **xferstats**, **Foreign**, and **audit** files. These files must be checked and removed occasionally to save storage space. BNU creates each file when it is needed.

When another system contacts your system with the **uucico** daemon's debugging mode on, it invokes the **uucico** daemon on your system with debugging turned on. The debugging messages generated by the daemon on the local system are stored in the **audit** file. This file can get quite large. Check and remove the **audit** file often.

The **errors** file records errors encountered by the **uucico** daemon. Checking this file can help you correct problems such as incorrect permissions on BNU work files.

The **xferstats** file contains information about the status of every file transfer. Check and remove this file occasionally.

The **Foreign** file is important to the security of your system. Whenever an unknown system attempts to log in to the local system, BNU calls the **remote.unknown** shell procedure. This shell procedure logs the attempt in the **Foreign** file. The **Foreign** file contains the names of the systems that have attempted to call the local system and been refused. If a system has been attempting frequent calls, use this information when considering whether to allow that system access.

Systemwide Log Files Used by BNU

Because many BNU processes need root authority to complete their tasks, BNU creates frequent entries in the **/var/spool/sulog** log file. Similarly, using the **cron** daemon to schedule BNU tasks creates multiple entries in the **/var/spool/cron/log** file. When using BNU, check and clean these files.

Using BNU Maintenance Commands

The Basic Networking Utilities contain several commands for monitoring BNU activities and cleaning BNU directories and files.

Cleanup Commands

BNU contains three commands that clean directories and remove files that have not been sent:

- **uuclean**

  Deletes all files older than a specified number of hours, from the BNU administrative directories. Use the **uuclean** command to specify a directory to be cleaned or a type of file to be deleted. You can also instruct the command to notify the owners of the deleted files. The **uuclean** command is the Berkeley equivalent of the **uucleanup** command.
- **uucleanup**

  Performs functions similar to the **uuclean** command. However, the **uucleanup** command checks the age of files based on days rather than hours. Use the **uucleanup** command to send a warning message to users whose files have not been transferred, notifying them that the files are still in the queue. The **uucleanup** command also removes files relating to a specified remote system.
- **uudemon.cleanu**

  A shell procedure that issues the **uulog** and **uucleanup** commands to compress the BNU log files and remove log and work files over three days old. The **uudemon.cleanu** command is run by the **cron** daemon.

Status-Checking Commands

BNU also provides five commands for checking the status of transfers and log files:

- **uuq**

  Displays jobs currently in the BNU job queue. Use the **uuq** command to display the status of a specified job or of all jobs. With root authority, you can use the **uuq** command to delete a job from the queue.
- **uustat**

Provides information similar to that provided by the **uuq** command, but in a different format. Use the **uustat** to check the status of jobs and delete jobs you own. With root authority, you can also delete jobs belonging to other users.

- **uulog**

  Displays a summary of **uucp** or **uux** requests, by user or by system. The **uulog** command displays the file names. See ″Working with BNU Log Files″ .
- **uupoll**

  Forces a poll of a remote system. This is helpful when work for that system is waiting in the queue and needs to be transferred, before the system is scheduled to be called automatically.
- **uusnap**

  Displays a very brief summary of BNU status. For each remote system, this command shows the number of files awaiting transfer. However, it does not show how long they have been waiting. The **uusnap** command is the Berkeley equivalent of the **uustat** command.

Shell Procedures

BNU is delivered with two shell procedures used for maintenance:

- **uudemon.cleanu**

  Is discussed under ″Cleanup Commands″
- **uudemon.admin**

  Issues the **uustat** command. The uustat command reports the status of BNU jobs. It sends the results to the UUCP login ID as mail. You can modify the **uudemon.admin** shell procedure to send the mail elsewhere, or use a mail program to reroute all mail for the UUCP login ID to the user responsible for BNU administration.

These shell procedures are stored in the **/usr/sbin/uucp** directory. If you wish to change what they do, copy the procedures and modify the copy. Run the procedures from the command line or schedule them to be run by the **cron** daemon.

To automatically run the **uudemon.cleanu** and **uudemon.admin** commands, remove the comment characters (#) from the beginning of the relevant lines in the **/var/spool/cron/crontabs/uucp** file.

Monitoring a BNU Remote Connection

Prerequisites

- The BNU program must be installed on the system.
- A link (hard-wired, modem, or TCP/IP) must be set up between the local and remote systems.
- The BNU configuration files, including the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file, must be set up for communications between the local and remote systems.

Procedure

The **Uutry** command can help monitor the **uucico** daemon process if users at the local site report file-transfer problems.

1. Issue the **uustat** command to determine the status of all the transfer jobs in the current queue as follows:

   ```
   uustat -q
   ```

   A status report similar to the following is displayed:

   ```
   venus 3C (2) 05/09-11:02 CAN'T ACCESS DEVICE
   hera 1C 05/09-11:12 SUCCESSFUL
   merlin 2C 5/09-10:54 NO DEVICES AVAILABLE
   ```

This report indicates that three command (C.*) files intended for remote system `venus` have been in the queue for two days. There could be several reasons for this delay. For example, perhaps system `venus` has been shut down for maintenance or the modem has been turned off.

2. Before beginning more extensive troubleshooting activities, issue the **Uutry** command as follows to determine whether the local system can contact the remote system `venus` now:

   ```
   /usr/sbin/uucp/Uutry -r venus
   ```

   This command starts the **uucico** daemon with a moderate amount of debugging and the instruction to override the default retry time. The **Uutry** command directs the debugging output to a temporary file, **/tmp/venus**.
3. If the local system succeeds in establishing a connection to system `venus`, the debugging output contains a good deal of information. However, the final line in this script, which follows, is the most important:

   ```
   Conversation Complete: Status SUCCEEDED"
   ```

   If the connection is successful, assume that the temporary file transfer problems are now resolved. Issue the **uustat** command again to make certain that the files in the spooling directory have been transferred successfully to the remote system. If they have not, use the steps in Monitoring a BNU File Transfer to check for file transfer problems between the local system and the remote system.
4. If the local system cannot contact the remote system, the debugging output generated by the **Uutry** command contains the following type of information (the exact form of the output may vary):

   ```
   mchFind called (venus)
   conn (venus)
   getto ret -1
   Call Failed: CAN'T ACCESS DEVICE
   exit code 101
   Conversation Complete: Status FAILED
   ```

   Check the physical connections between the local and remote systems. Make sure that the remote computer is turned on and that all cables are properly connected, that the ports are enabled or disabled (as needed) on both systems, and that the modems (if any) are working.

   If the physical connections are correct and secure, then verify all the relevant configuration files on both the local and remote systems, including the following:

   - Verify that entries in the **Devices**, **Systems**, and **Permissions** files (located in the **/etc/uucp** directory) are correct on both systems.
   - If a modem is being used, make sure the **/etc/uucp/Dialers** file contains the proper entry. If you are using dial-code abbreviations, be sure the abbreviations are defined in the **/etc/uucp/Dialcodes** file.
   - If a TCP/IP connection is being used, make sure that the **uucpd** daemon can be run on the remote system and that the configuration files contain the correct TCP/IP entries.
5. Once the physical connections and configuration files have been checked, issue the **Uutry** command again. If the debugging output still reports that the connection failed, it may be necessary to contact a local service representative for additional assistance. Save the debugging output produced by the **Uutry** command. This may prove helpful in diagnosing the problem.

Monitoring a BNU File Transfer

Use this procedure to monitor a file transfer to a remote system. Monitoring a file transfer is useful when transmission of files to the remote system is failing for unknown reasons. The debugging information produced by the **uucico** daemon (called by the **Uutry** command) can help determine the point of failure.

Prerequisites

- The BNU program must be installed and configured on the system.
- A connection to the remote system must already exist.

Procedure

Use the following procedure to monitor file transfers.

1. Prepare a file for transfer using the **uucp** command with the **-r** flag, by entering:

   ```
   uucp -r test1 venus!~/test2
   ```

   The **-r** flag instructs the BNU program to place the **test1** file in the queue but *not* to start the **uucico** daemon.
2. Issue the **Uutry** command with the **-r** flag to start the **uucico** daemon with debugging turned on by entering:

   ```
   /usr/sbin/uucp/Uutry -r venus
   ```

   This instructs the **uucico** daemon to contact the remote system `venus` overriding the default retry time. The daemon contacts system `venus`, logs in, and transfers the file while the **Uutry** command produces debugging output that monitors the uucico process. Press the interrupt key sequence to stop the debugging output and return to the command prompt.

   The **Uutry** command also stores the debugging output in the **/tmp/***SystemName* file.

UUCP Conversation Flow Diagram

UUCP error messages can be linked to a specific phase in the conversation flow. Use the following diagram and the message descriptions on the following pages to help diagnose your UUCP problems. Some of the messages described may not be sent from the operating system's UUCP (BNU) version but are included in case another UUCP version is in use on a different system platform.

PHASE 1 Status Messages

Assert Error	The system is having problems. Check the error report for possible causes (for example, `errpt -a | pg`).
System not in Systems	If you supply a remote system name that is not found in the **Systems** file, this status message is created. UUCP will terminate. Use the **uuname** command to check the system name again.
Wrong time to call	The **Systems** file has restrictions on times to allow outgoing calls. UUCP will keep trying until the time is right. Check the **Systems** file.
Callback required	The network has restricted usage either for security or economic reasons and access is denied at this time.
Cannot call or No call	UUCP recently tried to call the remote system and failed. It will not immediately try again. Can also be caused by an old system status file being left around thus keeping uucico from trying again.

PHASE 2 Status Messages

Dialer Script Failed	Your **Dialers** file script did not complete successfully.
No Device Available or Can't Access Device	The modem or the outgoing phone line from your system is busy. Check for an error in the device entry of the **Systems** file. Also, check the **Devices** and **Dialers** files to be sure logical devices have physical devices associated with them. Is the device in use by some other program? Check the **/etc/locks** directory for lock on port. If a lock file exists (for example, `LCK..TTY0`), remove it (for example, `rm /etc/locks/LCK..TTY0`). Also check permissions on the port.

Dial Failed or Failed (call to system)	Appears when your system dials another successfully but the other system does not answer. It may also indicate a problem in the **Devices** file. Enter the `uucico -r1 -x6 -s` *`system name`* command. It could be that UUCP is expecting some string that it is not receiving. Make connection by hand to find out what needs to be incorporated into the **Systems** file entry to satisfy the request. Please keep timing in mind; perhaps some delays are needed. This could also mean that the port is busy, you dialed a wrong number, or UUCP lost ownership of port.
OK or Auto Dial	These are information messages only.

PHASE 3 Status Messages

Handshake Failed (LCK)	The device is being used by someone else; the process could not create the **LCK** file. Sometimes **LCK** files must be manually removed by the administrator. After a number of retries, see your system administrator. See if another process has control of the port (for example, another uucico).
Login Failed or Timeout	Login failed due to a bad connection or slow machine. Similarly, you might see Timeout, which means the remote system did not respond within a set period of time. This could also indicate a problem with the chat script.
Succeeded (Call to System) or BNU	These are informational messages only and do not indicate a problem.

PHASE 4 Status Messages

Startup Failed or Remote reject after login	After login, uucico is started on the remote system. If there is a problem in initiating a conversation between the two systems, these messages are created. You may also have logged in using the wrong UUCP account. Initial **shere** handshake failed.
Wrong machine name	You called the wrong machine or the machine's name was changed.
Bad login/machine combination	Login for remote system failed. Problem could be a wrong phone number, login/password, or chat script.
Remote has a LCK file for me	Remote was trying to call at same time and your request will fail temporarily.
You are unknown to me	**System** or **Permission** file may have wrong permission or owner or may not have correct system name within. Use **uuname** on both machines.
OK or Talking	These are informational messages only and do not indicate a problem.
Get the login or password prompt but it is in all capital letters	Modem may be in echo mode (E1 on Hayes compatibles). This causes the modem to echo back, or send, a RING to your system when an incoming call is received. The getty process receives the string and accordingly changes the `login:` or `password:` into all caps. To fix this problem, change echo mode on modem to off (**ATE0** for Hayes compatibles). **Note:** Keep in mind that once this change is made, you should use *ATE1* in your **Dialers** file chat script or you will not get the expected **OK** back from the modem.
Remote port set for ″delay or ″getty -r and chat script expects key input	Ports set for delay are expecting one or more carriage returns before proceeding with the login. To fix this problem, try beginning the chat script on dialing system to the following: `"" \r\d\r\d\r\d\r in:--in: ...` Interpreted, the above reads to expect nothing and send return, delay, return, delay, return, delay, return.

PHASE 5 Status Messages

Alarm	**uucico** is having trouble with the connection. Either the connection is bad or xon/xoff is set to yes on modem.

Remote access to path/file denied or copy (failed)	Permission problem; check file/path permissions. Try giving read, write, and execute permission to the receiving directory (`chmod 777`) as a test.
Bad read	Remote system ran out of space, most likely in the spool area. Can also mean uucico could not read or write to device.
Conversation failed	The modem is powered off, its cable pulled or loose, or the remote system crashed or shutdown. A telephone disconnection can also cause this error.
Requested or Copy (succeeded)	These are informational messages only and do not indicate a problem.

PHASE 6 Status Messages

OK (Conversation Complete)	The remote can deny the hangup request and reverse the roles (meaning the remote has work for the local system to do). Once the remote uucico and the local uucico agree that no more work exists, they hang up.
Conversation succeeded	These are informational messages only and do not indicate a problem.

UUCP Quick Setup Guide and Information Sheet

(H)	Denotes HoneyDanBer versions only (or BNU).
(B)	Denotes Berkeley (BSD) versions only.
( )	Denotes common across all versions.

Chat Script Parameters		
Parameters	**Version Type**	**Description**
″ ″	( )	Null string.
\N	(H)	Null character.
\b	(H)	Backspace character.
\c	( )	Suppress carriage return sent at the end of every send string.
\d	(H)	Delay 2 seconds.
\d	(B)	Delay 1 second.
\e	( )	Disable echo checking.
\E	( )	Enable echo checking.
\p	(H)	Pause for approximately 1/4 second.
\K	(H)	Send a break.
\n	( )	Newline character.
\N	(H)	Send a null character. Use \0 on the other UUCP versions.
\p	(B)	
\r	( )	Carriage return character.
\s	( )	Space character.
\t	(H)	Tab character.
\ \	( )	Backslash character.
ABORT	(B)	Arm abort trap in expect string.
BREAK	( )	Generate break signal (approximately 1/4 second).
BREAK*n*	(B)	Generate *n*/10 seconds break signal.
CR	(B)	Carriage return character.
EOT	(H)	Send Ctrl-D newline Ctrl-D newline.

Chat Script Parameters		
EOT	(B)	Send Ctrl-D newline.
NL	(B)	Newline character.
PAUSE	(B)	Delay for 3 seconds.
PAUSE*n*	(B)	Delay for *n* seconds.
P_ODD	(B)	Use odd parity for future send strings.
P_ONE	(B)	Use parity 1 for future send strings.
P_EVEN	(B)	Use even parity for future send strings.
P_ZERO	(B)	Use parity 0 for future send strings.
\ddd	( )	Send octal character.
\M	(H)	Open dial-out port with O_NDELAY (clocal on).
\m	(H)	Reopen port with clocal off.

Appendix A. 128-Port Async Adapter Configuration Worksheet

The following worksheet is designed to help you plan for your 128-port async controller configuration. You will need one copy of the worksheet for *each* line.

128-Port Async Adapter Configuration Worksheet

Adapter:	
Line:	
Line Speed:	
Cable Type 8/4:	

Node ID	1n	2n	3n	4n
Connection T ype (comm mode)				
RAN Port	Device Type			
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

Figure 36. 128-Port Asynch Adapter Configuration Worksheet. This worksheet has space to write out adapters, lines, line speed, and cable type. It also has areas for the connection types for each Node (1n through 4n) and the device type for each RAN port (1 through 16) on each node.

Sample Worksheet

The following illustration is a sample of a partially completed worksheet.

128-Port Async Adapter Configuration Worksheet

Adapter:	`1 cmxa0`
Line:	`1`
Line Speed:	`56Kbps`
Cable Type 8/4:	`8 wire`

Node ID	*1n*	*2n*	*3n*	*4n*
Connection Type (comm mode)	`direct`	`422_modem`	`422_modem (virtual RAN)`	`direct (virtual RAN)`
RAN Port	*Device Type*			
1	`3151 Terminal`	`3151 Terminal`		
2	`3151 Terminal`	`Modem`		
3	`Printer`	`YT 100 Terminal`		
4		`Plotter`		
5	`3151 Terminal`			
6	`3151 Terminal`			
7	`Printer`			
8	`Modem`			
9	`3151 Terminal`			
10	`Printer`			
11	`3151 Terminal`			
12	`3151 Terminal`			
13	`3151 Terminal`			
14	`3151 Terminal`			
15	`3151 Terminal`			
16	`3151 Terminal`			

Figure 37. 128-Port Asynch Adapter Configuration Worksheet — Partially Completed

The above worksheet shows the following:

- **Adapter** is `1 – cmxa0`
- **Line** is `1`
- **Line Speed** is `56Kbps`
- **Cable Type** is `8-wire`
- The first node is a `direct connection` type.
- The first node's 16 available RAN Ports are assigned the following devices:
  1. 3151 Terminal
  2. 3151 Terminal
  3. Printer
  4. None
  5. 3151 Terminal
  6. 3151 Terminal

7. Printer
8. Modem
9. 3151 Terminal
10. Printer
11. 3151 Terminal
12. 3151 Terminal
13. 3151 Terminal
14. 3151 Terminal
15. 3151 Terminal
16. 3151 Terminal

- The second node is a `422_modem` type.
- The second node's 16 available RAN Ports are assigned the following devices:
  1. 3151 Terminal
  2. Modem
  3. YT-100 Terminal
  4. Plotter
  5. None
  6. None
  7. None
  8. None
  9. None
  10. None
  11. None
  12. None
  13. None
  14. None
  15. None
  16. None

Suggested Reading

128-Port Asynchronous Adapter Subsystem

Planning for Your 128-Port Async Adapter.

Appendix B. Ethernet

Note: The following appendix is not applicable to the Itanium-based platform.

The following information describes:
- Ethernet logical communication protocol.
- Ethernet types and constraints.

Ethernet Overview

Ethernet can be very simple or relatively complicated depending on the network.

Ethernet is logical communication protocol that permits multiple devices to talk to each other over a variety of physical media. This logical protocol is called *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD).

CSMA/CD is a broadcast mechanism where one station speaks while all others listen. When two stations try to speak at the same time, a collision takes place. With CSMA/CD, both stations detect the collision, back off, and retry later. In practice, collisions rarely happen.

A simple form of an Ethernet local area network (LAN) is illustrated by the following illustration.

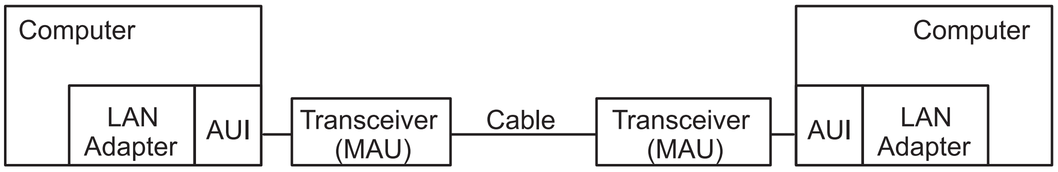


Figure 38. Simple Ethernet LAN

The LAN adapter provides the interface between the computer and the Ethernet. The transceiver, also called a medium attachment unit (MAU), provides the physical interface to the LAN medium. The interface between the LAN adapter and transceiver is the adapter unit interface (AUI). This interface has a standard connector (DB-15 with slide latches) and electrical characteristics. The Ethernet medium is a cable. The type of cable depends on the type of Ethernet.

In many cases, a single electronics board combines the LAN adapter and the transceiver.

There is a distance limit on the cable between two transceivers. This limit depends on the type of Ethernet. A repeater amplifies and retimes the Ethernet signals enabling transmission over longer distances. A repeater can have more than two ports. A multiple port repeater is commonly called a hub, particularly when used with 10BaseT. The following illustration depicts a network using a repeater with two ports.

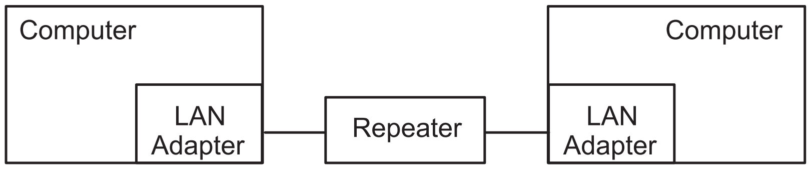


Figure 39. Ethernet LAN with Repeater

Ethernet Constraints

Constraints are guidelines specified by Ethernet architects to guarantee proper operation at a particular installation. You can guarantee proper operation at your installation by not violating any of the constraints. One indicator of constraint violation is a higher than normal error rate. Because of the error recovery mechanisms inherent in communication protocols, a high error rate is often only apparent to the user as lower performance.

There are two types of constraints on Ethernet:

- Limitations on the physical medium
- Architectural constraints on the system as a whole

Limitations on Physical Medium

There are three limitations on a physical medium:

- A length or distance limit on the amount of cable you can use to connect two transceivers.
- A maximum number of transceivers you can attach to a single medium cable.
- A maximum number of repeaters between any two points.

These particular limitations ensure the integrity of the signal between any two stations regardless of Ethernet implementation.

Architectural Constraints

Architectural constraints derive from timing parameters of the CSMA/CD protocol. For the collision detection and recovery mechanism to operate properly, the time interval between when two LAN adapters recognize a collision is bound. As a result, the overall size of the system is bound by the propagation time of the signals over the longest path.

Types of Ethernet

There are three types of Ethernet:

10BaseT	Uses unshielded twisted-pair cables as its transmission medium. It depends upon an active hub to resolve collisions between stations trying to talk at the same time.
10Base2	Uses inexpensive coaxial cable as its transmission medium. It depends upon transceiver logic in the host adapter to resolve collisions.
10Base5	Uses a special double-shielded coaxial cable as its transmission medium. It depends on an external transceiver to resolve collisions.

10BaseT

10BaseT uses a star topology. Each of the communication devices connects to a centrally located hub that implements the CSMA/CD protocol. A star topology Ethernet using twisted-pair is easier to maintain than coaxial types of Ethernet because:

- All connections occur at a central location.

- Each link (device-to-hub) can be maintained independent of other links.
- A failure on one link does not affect other links.
- Hubs typically provide diagnostic tools to help an administrator understand the state of the hub's connections.

For example, one of the simplest diagnostic tools is the link integrity indicator. A link integrity indicator is a light on the hub and adapter cards that remains on as long as the link integrity signal is not compromised. A hub and its devices constantly transmit a link integrity signal. This is a simple way to verify a proper physical connection.

In a more sophisticated implementation, an intelligent hub can provide link integrity status to network management software. This eliminates the need to go to the hub or device to see the status of the indicator light.

Hubs: A normal 10BaseT hub provides a fixed number of ports, such as 16. You can expand the total number of ports on a network by attaching multiple hubs together. When attaching hubs together, you can use twisted-pair (10BaseT technology) or coaxial (10Base2 or 10Base5 technology) cable. When connecting multiple hubs, the network is usually organized in a tree, as shown in the following illustration.

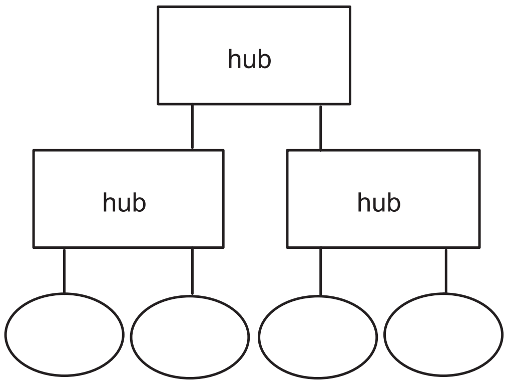


Figure 40. Multiple Hub Tree Structure

Constraints: The maximum cable length between any two end points on a 10BaseT Ethernet is 100 meters (end points are active devices, such as host adapters, transceivers, or hubs). An end point can be either a 10BaseT transceiver or a hub. This 100-meter limitation assumes that you are using level 3 cable and a limited number of interconnects. Interconnects include plug-in sockets or punch-down blocks. Using inferior cable or an excessive number of interconnects reduces the effective length.

There is a maximum of four repeaters between any two stations. A 10BaseT hub counts as a repeater.

Troubleshooting 10BaseT	
Symptom	**Action**
The link integrity indicator is off on one or both sides of the connection.	Verify the cable. Verify the pinout of the cable.
The link integrity indicator is on for both devices, but the devices still don't communicate.	Verify the quality of the cable.
The transfer rate is very slow.	Verify the quality of the cable. Verify the integrity of the interconnect.

Verify the Cable: Verify that the devices are firmly plugged in. Verify that the ends of the same cable are securely in the jack.

Verify the Pinout of the Cable: Check that you are using a straight-through cable. A normal 10BaseT is a straight-through cable. This means that pin 1 at one end of the cable connects to pin 1 at the other end. Similarly pin 2 connects to pin 2, and so on for all connector pins.

Check the cable pairs. The cable pairs within the twisted pair must also connect properly. Connector pins 1 and 2 attach to one pair while pins 4 and 6 connect to the other.

Verify the Quality of the Cable: Check for any occurrence of flat silver satin conductor wire. This wire is commonly sold for use with telephones and is not appropriate for 10BaseT installation due to incorrect electrical characteristics.

All of the cables in your 10BaseT Ethernet, including jumper cables, must be LAN-grade cables (Level 3 ATP). Even a short length of silver satin cable can cause a connection malfunction.

Verify the Integrity of the Interconnect: A continuity tester may show that a connection is electrically correct. However, in terms of a transmission, the connection has excessive impedance mismatches. These impedance mismatches result from discontinuity. A cable splice or a plug in socket are two items that cause discontinuity. Each occurrence degrades the signal.

You can use a time domain reflectometer (TDR) to test for discontinuity. The TDR identifies each occurrence by indicating the distance from the end of the cable to the discontinuity.

10Base2

10Base2 is a simplified form of the original Ethernet that uses T-connectors in place of the original Ethernet tapped-style transceivers. This form of Ethernet is inexpensive and easy to install for small networks. On a larger network, thin coaxial is difficult to maintain. In a typical 10Base2 installation, one electronics board contains both the transceiver and the LAN adapter.

10Base2 is a linear or bus topology that extends from device to device throughout a network. Each end of the bus must be capped off with a terminator. Between these terminated end points, you can attach up to four repeaters. The cable must be either RG58A/U or RG58C/U coaxial cable.

Most transceivers contain a switch to enable or disable a function called SQE heartbeat.

Constraints: The maximum segment length for a 10Base2 Ethernet is 185 meters or 600 feet. The maximum number of stations for each segment is 30. You can use up to four repeaters. When you use four repeaters, only three of the five segments can have taps. The remaining two segments must be direct links. The minimum distance between stations is 0.6 meters or 1.5 feet.

Troubleshooting 10Base2	
Symptom	**Action**
No stations talking.	Verify cable termination. Split the network into segments to isolate problems.
One station doesn’t talk.	Verify the Ethernet adapter in the station. Replace the T-connector.
Whole network slow.	Split the network into segments to isolate problems. Verify cable termination.
One station slow.	Verify the Ethernet adapter in the station.

Verify Cable Termination: Verify that a cable terminator is attached to both ends of the cable. If the network still does not work, remove the terminator from one end and use an ohmmeter to measure the resistance between the center tap of the cable and the shield. The resistance should be 50 ohms if the terminator at the opposite end is correct. Replace the terminator and repeat the experiment at the opposite end.

Split the Network: Somewhere near the middle of the cable, remove one cable end from the T-connector, and replace it with a terminator. Test the new shorter segment. If it works, replace the cable, and put a terminator on the opposite side of the T-connector. Now test the other segment. If either side of the T-connector fails, focus on that segment. Keep repeating this experiment, making the defective segment smaller and smaller, until you isolate a defective cable, node, or T-connector.

Replace the T-Connector: T-connectors are defective more often than you might think. If you have narrowed the problem to a segment with a single T-connector, try replacing the T-connector.

Verify the Ethernet Adapter in the Station: If you remove the T-connector from an Ethernet adapter or transceiver and everything else on the network works, try replacing the isolated Ethernet adapter or transceiver.

10Base5

10Base5 is the original form of Ethernet. It uses a special double-shielded coaxial cable. Transceivers are separate from the Ethernet adapters and tap into the cable.

10Base5 is a linear or bus topology that extends through a building, usually in the ceiling. Devices attach to the transceivers tapped into the cable with an AUI cable. Like 10Base2, a 10Base5 network must be terminated at both ends, or the network will not function.

10Base5 requires a special coaxial cable designed exclusively for Ethernet and readily identifiable by its yellow jacket. A variant of this cable is available with a fire-retardant jacket for use in plenums. This variant has an orange jacket. AUI cables are constructed from shielded coaxial cable.

10Base5 is not recommended for new installations. It is significantly more expensive than either 10BaseT or 10Base2 and has similar maintenance problems to 10Base2. It does have one advantage: individual segments can be up to 500 meters (1640 feet) long.

Constraints: The maximum segment length for a 10Base5 Ethernet is 500 meters (1640 feet). The maximum number of stations per segment is 100. You can extend segments with up to 4 repeaters, but only three of the segments can have taps on them. The other two segments must be direct links. AUI cables can be up to 50 meters (164 feet) long.

Troubleshooting 10Base5	
Symptom	**Action**
No stations talking.	Verify cable termination. Split the network into segments to isolate problems.
One station doesn’t talk.	Verify the Ethernet adapter in the station. Replace the transceiver.
Whole network slow.	Split the network into segments to isolate problems. Verify cable termination.
One station slow.	Verify the Ethernet adapter in the station. Replace the transceiver.

Verify Cable Termination: Verify that a cable terminator is attached to both ends of the cable. If the network does not work, remove the terminator from one end and use an ohmmeter to measure the resistance between the center tap of the cable and the shield. The resistance should be 50 ohms if the terminator at the opposite end is correct. Replace the terminator you removed, and repeat the experiment at the opposite end.

Split the Network: Somewhere near the middle of the cable, remove one cable end from the transceiver and replace it with a terminator. Test the new shorter segment. If it works, replace the cable and put a terminator on the opposite side of the transceiver. Now test the other segment. If either side of the transceiver fails the test, focus on that segment. Keep repeating the experiment, making the defective segment smaller and smaller, until you isolate a defective cable, node, or transceiver.

Replace the Transceiver: If you have a tapped transceiver, you can simply remove it from the cable. Otherwise, unscrew the coaxial connectors, and replace the transceiver with another one.

Verify the Ethernet Adapter in the Station: If you remove the AUI cable from an Ethernet adapter and everything else on the network works, try replacing the isolated Ethernet adapter or transceiver.

Appendix C. Wiring Serial Devices

For many installations, the most troublesome part of installing communications equipment is making the physical connection work. The following information describes the terminology and issues involved in making a successful connection:

- Terminology
- Communications standards
- Null modems
- Connectors

Terminology

Two types of devices can connect to a serial port: Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). As you might guess from the name, a terminal is a DTE. A modem is the most common example of a DCE. These terms derive from a telephone company perspective as seen in the following illustration:

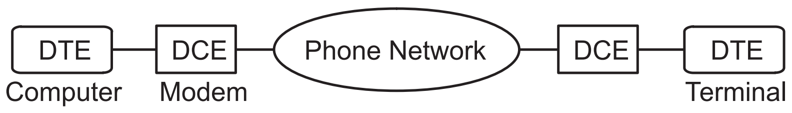


Figure 41. Phone Network

A DTE is any device that drives the transmit data signal and listens to the receive data signal. A DCE is any device that listens to the transmit data signal and drives the receive data signal.

A null modem is a connection device that eliminates the need for the modems and phone networks between like devices (two DTE devices). For more information on null modems, refer to ″Null Modems″.

Communications Standards

For two devices to communicate, they must speak a common language (use common electrical signals). To provide a common language for all manufacturers, the Electronic Industries Association (EIA), a U.S. standards body, devised a series of standards. The three most common standards in use for serial devices (terminals, printers, and modems) are:

- EIA 232, the most commonly used standard.
- EIA 422, used for high-speed, long-distance operation.
- RS-423, a compromise between EIA 232 and EIA 422.

In Europe, CCITT, an international organization, has published equivalent standards as shown in the following table:

Communications Standards	
EIA	**CCITT**
EIA 232	V.28
RS-423	V.10
EIA 422	V.11

Circuits

Common communications standards, such as EIA 232, define many different circuits as part of the interface between the data communication equipment (DCE) and the data terminal equipement (DTE). Terminals do not use many of these circuits. Furthermore, not all terminal manufacturers interpret and use these circuits in the same way. Terminals commonly use the circuits shown in the following table.

Commonly Used Signals			
Signal	**Name**	**Direction**	**Function**
TD, TxD XMT	Transmit Data	From DTE	The terminal sends information on this circuit.
RD, RxD RCV	Receive Data	To DTE	The terminal receives information on this circuit.
DTR	Data Terminal Ready	From DTE	Indicates that the terminal is operational. Some terminals use this circuit for flow control.
DSR	Data Set Ready	To DTE	Input to the terminal indicating that the DCE is ready. Some terminals will not receive if this circuit is not asserted.
RTS	Request to Send	From DTE	On most terminals, this circuit is always asserted.
CTS	Clear to Send	To DTE	Indicates that the DCE is ready to receive. Most terminals will not transmit if this circuit is not asserted.
DCD, CD	Data Carrier Detect	To DTE	Indicates that the DCE has a carrier, a fact that is of no significance to ASCII terminals. Some terminals will not receive if this circuit is not asserted.
GND, SGND	Signal Ground		Common reference point for all data and control circuits.
TD REF, TXD	Transmitter Reference	From DTE	Reference circuit for transmitter circuits in RS-423 and EIA 422 implementations.
RD REF, RXD	Receive Reference	To DTE	Reference circuit for receive circuits in RS-423 and EIA 422 implementations.
PGND	Protective Ground		Used for the shield of a shielded cable.

Usually, you can get a terminal talking if you connect transmit, receive, and ground, leaving the other signals unconnected. Adding other control signals can sometimes increase the performance or functionality of the terminal.

For the EIA 232 standard, there is a single reference point for all circuits called *signal ground*. Conceptually, its voltage relative to the earth ground is zero. However, in a realistic setting (a building with many electrical distribution paths), the voltage of signal ground relative to earth ground may not be zero and may even be different between the DCE and the DTE. This voltage difference between the DCE and the DTE, referred to as *ground shift*, can make all other circuits, especially data circuits, less reliable and prone to errors.

Newer standards (such as RS-423) separate the reference points of the transmit and receive circuits to reduce ground shift. When using separate transmit and receive ground references, the receive reference of the DCE connects to the transmit reference of the DTE and vice versa.

The RD REF circuit is the reference point for RD, CTS, and DCD. When communicating with an EIA 232 terminal, these two circuits are tied together at the terminal to the signal ground circuit of the terminal.

The EIA 232 standard contains another circuit, called protective ground that has nothing to do with any of the other circuits. Generally connected to the chassis of the device, it may also connect to the shield in a shielded cable.

Null Modems

When connecting a DTE-wired terminal to another DTE-wired device, you must make a circuit that replaces the modems and phone network. A null modem performs this function. To make a null modem, reverse the transmit and receive signals.

A null modem permits data flow between devices but does not pass control information. The control signals provide additional functionality useful with some applications.

Connectors

Two types of connectors are commonly used for serial connections on terminals:
- D-type connectors
- MMJ connectors

D-type Connectors

D-type connectors (named because of their shape) come in 25-pin (DB-25) and 9-pin (DB-9) configurations. Most terminals use the DB-25 connector. The EIA 232 standard says that the DTE should have a male connector, but many terminal vendors use a female connector instead.

Some terminals have two DB-25 connectors that are often labeled `modem` and `aux`. In most terminals, the port labeled `modem` is DTE-wired regardless of what the connector looks like. A terminal adapter (Feature Code 7904) is available for use with terminals with female DB-25 connectors such as the 3151 or Wyse.

Most PCs use a male DB-9 connector wired as a DTE for their COM1 port. Some terminals also use DB-9 connectors. The DB-9 uses the same circuits as the DB-25 but connects them to different pin numbers.

MMJ Connectors

Some Digital Equipment Corporation (DEC) terminals (and clones) use a connector that DEC calls a *modified modular jack (MMJ)* connector. To attach a terminal with this type of connector, you need a special adapter.

Appendix D. Wiring and Interconnection

This appendix describes how to:

- Plan your wiring installation.
- Select appropriate cables.
- Select appropriate connectors.

Structured Wiring

For many communications applications, the wiring is the most troublesome and expensive aspect of the installation. You can minimize wiring costs and problems by adopting a standard methodology and using it consistently throughout your installation. In recent years, several organizations have proposed standardized wiring systems that specify the types of cables and interconnects to use and the rules for using this equipment.

The following information presents a recommendation for wiring, derived from the Electronic Industries Association/Telecommunications Industry Association (EIA/TIA) standard 568. If you follow these recommendations, you can utilize your investment in wiring for a variety of communications applications including serial, local area network (LAN), and voice communications. The recommendations presented here have been used for many years for commercial telephone installations. In this case, EIA/TIA Standard 568 recommends a better grade of cable than that typically used for telephone installations.

A structured wiring system includes the following components:

- Station termination
- Standard information outlet
- Horizontal distribution
- Distribution frames
- Building main distribution

Station Termination

In wiring language, a station is the device with which you are trying to communicate, such as a terminal on your desk. Station termination is the cable from the wall outlet to the station and its associated adapter, if necessary.

Standard Information Outlet

The standard information outlet is the wall outlet of the system. If it is installed and used properly, this outlet can be used for many types of data depending on the signals carried over the wires. You should use the RJ-45 jack as the standard information outlet. This jack is available from any wiring equipment vendor.

Horizontal Distribution

Horizontal distribution is the wire that runs from the wall jack to a central wiring closet. It is called horizontal because that is the main direction that it runs, although the first part is normally vertically up the wall, then horizontally through the ceiling to the wiring closet. Special types of wiring and supporting structures, such as conduit or raceways, may be needed to satisfy local building or fire safety codes. Some of the options and requirements are described in ″Twisted Pair Cable″.

Distribution Frames

In most installations, the horizontal distribution cables from a region of the building come together in a central location, the wiring closet. At this point, the horizontal distribution cables are spliced together with other cables such as the building main distribution or cables that attach directly to some communications equipment. The interconnection equipment that ties the cables together is called the intermediate distribution frame. Recommendations on the type of equipment to use are presented in ″Interconnection Systems″.

Building Main Distribution

Building main distribution refers to the equipment that ties together all the wiring closets in the building. In a telephone installation, it is usually one or more 25-pair cables that terminate at the private branch exchange (PBX) or leave the building to the phone company central office. In data applications, the building main distribution is usually a LAN that ties together the hubs or serial concentrators.

Twisted Pair Cable

Communications wiring uses many types of cable; however, all new installations should use 100 ohm unshielded twisted-pair (UTP) cable conforming to EIA/TIA standard 568, Category 3 or higher. This type of cable is widely available and relatively inexpensive and offers performance as good as or better than many of the other cables used for serial communications. This same type of cable is recommended for 10BaseT applications and is suitable for voice telephones.

Each pair in this cable consists of two solid copper conductors enclosed in a polyvinyl chloride (PVC) sheath. The conductors are twisted around one another to minimize interference and crosstalk with other electrical signals. All installations should use 4-pair cable for horizontal distribution unless a large number of signals are being routed together to an intermediate distribution frame, in which case a 25-pair or larger bulk cable can be used.

The recommended cable is not shielded. Shielded cable is more difficult to install and more expensive and provides no technical benefit except in cases of extreme electrical interference.

For station termination, telephones generally use a flat-, 4-, 6-, or 8-conductor untwisted cable in a silver or gray PVC sheath, known as silver satin. This type of cable should not be used for data cables, even for short patch cables. It has the wrong electrical characteristics and will severely degrade the performance of your installation.

Note: At high data rates, the type of cable you use is critical. You should use UTP cable specified at level 3 grade or higher according to EIA specification 568.

Color Codes

Adopting and maintaining a consistent wiring pattern will simplify the maintenance of your installation. The communications industry has adopted a standard naming and numbering convention for twisted-pair cables. For historical reasons, one conductor in each pair is the tip conductor and the other is the ring conductor. The terms *tip* and *ring* originated when phones were switched by operators using phone-jack plug-and-socket patch cables. The tip conductor attached to the tip of the plug. The ring conductor attached to the metal ring in the plug.

Each pair in the cable is numbered sequentially, starting at one. Stripes of color in the sheath of each conductor uniquely identify that conductor and its pair number. The stripes consist of a primary color and a secondary color. A 4-pair cable is numbered as shown in the following table.

4-Pair Cable Color Code		
Pair	**Primary**	**Secondary**
1	White	Blue
2	White	Orange
3	White	Green

4	White	Brown

In the tip conductor, the primary color stripes are wide and the secondary color stripes are narrow. In the ring conductor, the primary color stripes are narrow and the secondary wide.

Cable Grades

The EIA/Telecommunications Industries Association (TIA) standard 568 defines different categories or levels of cable for different applications, as shown in the following table.

Cable Applications		
Level	**Typical Bandwidth**	**Application**
2	< 1 MHz	Telephone
3	< 10 MHz	LAN (10BaseT)
4	< 20 MHz	LAN (16 Mbps UDP)
5	< 100 MHz	LAN (FDDI)

The cost increment of using a high grade of cable generally isn't a significant part of your overall installation costs, so you should standardize on Level 3 or higher. This permits you to use any of your installed wiring for telephone, serial communications, or 10BaseT LANs as needed. If you select Level 5 cable, you may be able to use your cables with the emerging high-performance LAN standards such as fiber distributed data interface (FDDI) on copper twisted pair and 100 Mbps Ethernet.

Cable Distances

Cable distances have been optimized around common baud rates. The following table shows maximum baud rates and optimal slew rate settings.

Slew Rate Settings			
Distance (ft) RS-423/EIA 422	**Distance (ft) EIA 232**	**Maximum Baud Rate (bps)**	**Slew Rate Setting**
< 2300	< 200	9600	slow
< 625	< 200	38400	medium
< 150	< 50	115,200	fast
< 130	N/A	Reserved	super

Plenum Cables

If you plan to route your horizontal distribution cables through a plenum, that is, a duct used for circulating air in a building, you should use plenum-rated cable. This twisted-pair cable uses a sheath material that is more fire retardant than PVC, such as Teflon. plenum-rated cable is more expensive but is much safer than PVC cable. If the building were to catch fire, PVC would give off toxic fumes, which would spread quickly through the plenum.

Note: Fire officials in many communities regard the open space above raised ceiling tiles as a plenum and require plenum-rated cables, unless the cable is enclosed in electrical conduit.

Interconnection Systems

The horizontal distribution cable connects your information outlet to a central point such as a wiring closet, where it attaches to an active concentrator like a 10BaseT hub, or attaches to other cables for routing to another point in the building. A distribution frame ties the cables together in the wiring closet.

There are two types of distribution frames. When splicing cables together in a stable configuration environment, the preferred method uses a punch-down block, also known as a cross-connect block. When tying equipment together in an environment subject to frequent configuration changes, the preferred method uses a patch panel constructed from RJ-45 connectors.

For high-bandwidth applications such as 10BaseT, you need to minimize the number of cable splices and connectors between your LAN adapter and the hub. Each time you go through a plug-and-jack connector or cable splice, you disturb the electrical characteristics of the cable by introducing signal reflections that increase the probability of data errors. Using proper, good-quality connectors and punch-down blocks minimizes these disturbances, but they cannot be eliminated. Each such disturbance effectively reduces the maximum distance of a cable span. Exceeding this distance increases the error rate to unacceptable levels.

Main Building Distribution

Telephone systems usually tie all phones to a centrally located PBX. Multipair cables tie intermediate distribution frames to the PBX located in a phone room or directly to the telephone central office if a PBX is not used.

A similar approach can be used for LAN systems, but the distance limitations on serial lines and especially on LAN wiring make it more practical to locate the 10BaseT hub at the intermediate distribution frame. For these systems, the LAN replaces multipair cables as the main building distribution.

The same reliability and maintenance considerations that make 10BaseT LANs the wiring of choice for horizontal distribution apply to main building distribution as well. However, the 100 meter distance limitation on 10BaseT wiring may cause problems in tying together intermediate distribution frames. When this is the case, use standard Ethernet with repeaters. Standard Ethernet becomes more practical when it ties together a limited number of 10BaseT hubs at the intermediate distribution sites.

For longer runs between the intermediate distribution sites, particularly if some of the runs are outdoors, you should use fiber optic Ethernet. The use of fiber optics for your main distribution network will make your whole building much less susceptible to electrical disturbances.

Appendix E. Data Transmission

Note: The following appendix is not applicable to the Itanium-based platform.

Data transmission refers to the structure and timing of data. It includes the following parameters:

- Data rate
- Parity
- Stop bits
- Flow control
- Modems
- Serial Printers

Data Rate

Data rate is the speed at which data travels between devices, measured in bits per second. To successfully transmit data, two devices must transmit at the same data rate; otherwise you may get distorted data or no data at all.

The physics of cables dictates that the faster you transmit data on a cable, the less reliable is its reception at the other end. Many other factors also reduce data integrity. These include:

- Cable type
- Cable length
- Electronics in the devices
- Interference from other electrical devices

Since you generally have little control over these factors, you can minimize their effect by using a slower data rate.

Terminal Data Rates

A terminal interface usually supports higher communications data rates than the speed at which the terminal processes data. Therefore, if you set the line speed to a substantially higher data rate than the terminal can process, you reduce the reliability of your connection without improving the observed response time.

To choose the optimal speed for the terminal, transmit data at the highest possible rate without introducing errors, but no faster than the terminal can actually process. You can use parity checking to verify the integrity of the data.

Slew Rate

Slew rate is the rate of change on a communications line as it transitions from 1 to 0 and from 0 to 1.

Note: Data rate specifies the duration of the ones and zeroes, not the transition time.

Slew rate is not directly related to data rate, since theoretically a line should be able to transition from 0 to 1 infinitely fast regardless of the actual data rate.

In practice, however, there is a relationship. Signals with a fast slew rate tend to generate interference with other signals, and this interference can limit the transmission distance of a communications signal. If the slew rate is too slow relative to the data rate, the signal does not reach a sufficiently high voltage to be recognized by the receiver on a 1 bit before it starts transitioning to a 0 bit. Consequently, there is an optimal slew rate for each data rate.

Parity

Parity is a coding scheme for checking the validity of data characters. An extra bit transmitted with each character indicates whether the sum of the other bits in the character is even or odd. You can configure most terminals to transmit data, using even, odd, or no parity bits (none).

Parity checking detects reliability problems when transferring data. However, because most terminals ignore parity errors on data sent to the terminal, parity checking on data sent to a terminal is useless.

You can use the **cnsview** command to report on the number of errors observed on a port. If you observe parity errors on a port, you should take corrective action, such as:
- Lowering the data rate on the line.
- Improving the cabling on the line.
- Using a limited distance modem on the line.

Stop Bits

Stop bits delineate the end of a character in the data stream. Supported stop bits are one and two.

Flow Control

Flow control is the process of pacing data transmission so that the receiver has a chance to process all incoming characters before the transmitter sends additional data. Terminals usually require flow control to display data as fast as possible because some operations like clearing the screen take longer than simply adding a character to the screen. If you set the speed slow enough to accommodate the slowest operation, you unnecessarily delay more common operations. If you set the speed to match the fastest operation, the terminal may drop characters if it encounters a series of slow operations. Flow control enables you to set a high speed without dropping characters when the terminal is doing slow operations.

You can manage flow control using either software or hardware. Software flow control uses ASCII control characters to start and stop character transmission. Hardware flow control uses modem control signals to start and stop transmission.

Flow Control	
Advantages	**Disadvantages**
Software flow control uses only transmit and receive data circuits and does not require circuits in your cable.	Software flow control uses the same Ctrl-S and Ctrl-Q characters that some applications use.
Hardware flow control is completely independent of any application or interpretation of the data stream.	Hardware flow control requires additional wires and cables.

Modems

A modem is a data communication equipment (DCE) that uses EIA 232. EIA 232 defines many different circuits as part of the interface between the DCE and the data terminal equipment (DTE). Modems commonly use the circuits shown in the following table.

Modems

Signal	Name	Direction	Function
TD, TxD XMT	Transmit Data	To modem	The modem receives information on this circuit.
RD, RxD RCV	Receive Data	From modem	The modem sends information on this circuit.
DTR	Data Terminal Ready	To modem	The terminal asserts DTR when it wants the modem to answer or dial out and lowers DTR when it wants the modem to hang up.
DSR	Data Set Ready	From modem	The modem asserts DSR when it is on.
RTS	Request to Send	To modem	The modem uses RTS for flow control.
CTS	Clear to Send	From modem	The modem also uses CTS for flow control. The modem asserts CTS when it is ready to send more data.
DCD, CD	Data Carrier Detect	From modem	Indicates when a phone call connects. The modem lowers DCD when a dial-in connection hangs up or a dial-out connection is lost.
RI	Ring Indicator	From modem	Not used on Hayes-style modems.
GND, SGND	Signal Ground		Common reference point for all data and control circuits.
RD, REF RXD	Receive Reference		A circuit that attaches to a signal ground on the modem.
PGND	Protective Ground		Used for the shield of a shielded cable.

Control signals provide additional functionality, which is useful with some applications.

Modems use DB-25 connectors. Using a modem adapter (DB-25 to RJ-45) to convert the DB-25 connector on your modem to an RJ-45, enables you to use twisted-pair cable between the modem and the terminal.

Serial Printers

Serial printers are normally configured as DTEs; that is, they expect to receive data on the receive data line and transmit data on the transmit data line.

Serial printers default to EIA 232 connections and use DB-25 D-type connectors. Many printers also support EIA 422 connections.

Appendix F. Notices

This information was developed for products and services offered in the U.S.A.
IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Index

Printed in U.S.A