Asynchronous Communications on the Altos 8600 and the Altos 586

- How to transfer files between Altos Xenix systems
- How to communicate with non-UNIX systems

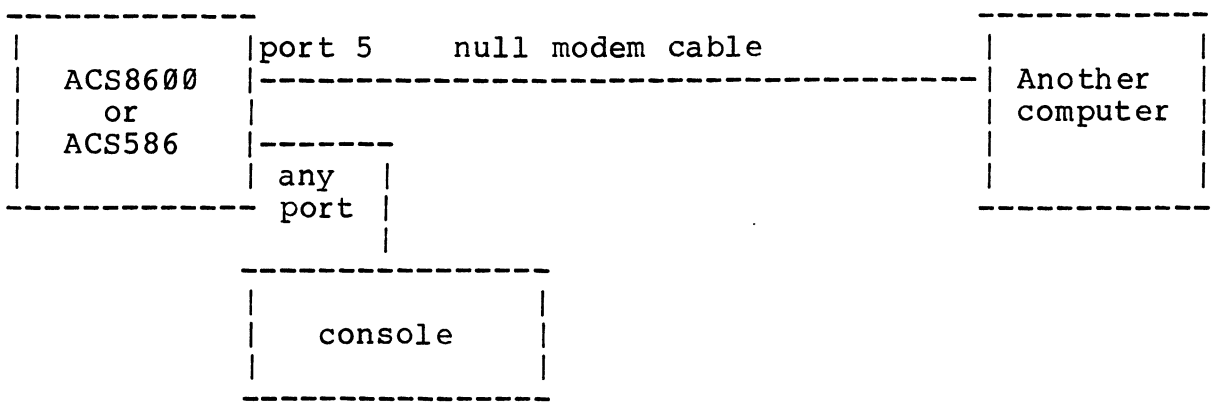
This document describes how to use the "cu" and "uucp" asynchronous communications packages. These packages are available as standard features on all Altos Xenix systems. Either package can be used to transfer files between the Altos 8600 and the Altos 586. Cu is easier to use, but it is not as powerful as uucp. Altos recommends that uucp be used to transfer files between the 8600 and the 586; the cu package should be used to communicate with non-UNIX systems. However, we recommend that anyone interested in asynchronous communication under Xenix learn to use both software packages.

Cu can be used for foreground asynchronous communications with almost any other computer system. Under light to medium system loads, communication is possible at 1200 baud. Under heavy loads, 600 baud or 300 baud is the maximum data rate. Facilities are provided for executing commands on the remote system, and transferring files to or from the remote system.

The uucp package can be used for background asynchronous communications with almost any other UNIX system. The baud rate can be up to 9600 baud. Uucp cannot be used for communications with a non-UNIX system. Two commands are provided with the uucp package: the uucp command allows one to transfer files to or from a remote system; the uux command allows one to execute commands on a remote system.

Section 1 of this document describes the cu package. Section 2 describes the uucp package. The use of modems with Altos Xenix systems is covered in section 3.

Section 1: The cu program



To use cu, follow these steps:

1. Determine the serial port to be used for communication. Any port not utilized by a terminal or a printer can be used; for purposes of demonstration, we will assume that port 5 is available.

2. Become the super-user (using the su command), then execute:

```
# disable /dev/tty5
```

This command will prevent a terminal from using port 5.

3. Execute the command:

```
# cu -t -s 1200 -l /dev/tty5 -a /dev/null
```

The cu program will attempt to establish a connection with the other computer, using serial port 5. The speed of the connection will be 1200 baud (cu will not function properly at baud rates greater than 1200 baud). The following message should be displayed after the connection has been successfully established:

```
Connected
```

All characters typed from this point onwards will be processed by the remote system, and all output from the remote system will be displayed on the screen. If a line begins with the character '~', it will be processed by the local UNIX system, and will NOT be passed to the remote system. The following commands are available when communicating with a UNIX or a non-UNIX remote system:

~.	Terminate the connection.
~<filename	Send the contents of the named file to the remote system, as if it was typed at the terminal.
~!command	Execute the specified command on the local system.
~$command	Execute the specified command on the local system, and send the output to the remote system.

The following commands are available only if communicating with a UNIX system:

~%takeremotefilelocalfile	Copy'remotefile'to 'localfile'
~%put localfile remotefile	Copy 'localfile' to 'remotefile'

If an output line FROM the remote system begins with the character '~', all output will be diverted to a local file. The following output-diversion commands are available:

~>filename	Divert output to the named file, and display each line as it is received.
~>>filename	Append output to the named file, and display each line as it is received.
~>:filename	Divert output to the named file, but do not display each line.
~>>:filename	Append output to the named file, but do not display each line.

All characters sent from the remote system to the local system will be diverted to the named file until the following line is encountered:

~>	Terminate output diversion.

The diversion commands can be used to transfer files between a UNIX system and a non-UNIX system. The exact way in which these commands should be used depends upon the non-UNIX operating system on the remote computer. Altos supplies a program called "TOXENIX" as part of its standard distribution package. This program can be used to transfer files from an MP/M system to a Xenix system. Similar programs can be

written to facilitate communication with other systems. Consult your local Altos dealer or Altos Customer Service for information on how to communicate with other systems.

Section 2: The uucp package

The uucp package is very complex. The following dissertation introduces the reader to the basic operations of uucp; more thorough documentation is included in several sections of the standard Xenix documentation (see volume 2B, sections 35 and 36 of the UNIX Programmer's Manual).

The implementation of uucp on the Altos Xenix system differs from the typical uucp implementation in one respect. The name of the local system, which is usually imbedded in the uucp source code, must be specified in the file /etc/systemid. The standard release assigns the name "altos86" to each system. The systemid file must be modified if more than one Altos system is included in a uucp network.

Each system in a uucp network is, at any given time, either an active system or a passive system. An active system can initiate communication with other systems; a passive system cannot initiate communication. Users on passive systems can still execute uucp commands; the only difference is that passive systems cannot initiate calls (i.e., they must be called from an active system). If a system is active, a modem is required for remote communication over telephone lines. A modem is not required for communication over distances shorter than a few hundred feet. If remote communication is desired, a user-written auto-dial program is needed to take full advantage of the auto-dial capabilities of most modems; such a program is not included as part of the standard Xenix release. The use of modems with cu and uucp is discussed in more detail near the end of this document.

The uucp package consists of a series of programs. The following programs are executed directly by the user:

/bin/uucp	This program requests that file(s) be copied to or from another system. The file(s) are not copied immediately; a request to copy them is placed in the /usr/spool/uucp directory, where the request is acted upon at a later time by the uucico program.
/bin/uux	This program requests that command(s) be executed on another system. As with the uucp command, the request is queued in the spool directory.

Other programs in the uucp package are usually initiated indirectly. These programs are:

/usr/lib/uucp/uucico — This program performs all communications between systems. On an active system, it does the following:

1. It scans the spool directory for work.
2. It attempts to login on each remote system.
3. If successful, it starts the uucico program on each remote system.
4. It executes each request from both systems.
5. It enters information about each transaction in the log files in the spool directory.

On an active system, uucico is automatically started whenever any user runs the uucp or uux programs. It can also be started by the /etc/cron program. On a passive system, uucico can only be started by another copy of uucico running on a remote system.

/usr/lib/uucp/uuclean — This program is used to remove old files from the spool directory.

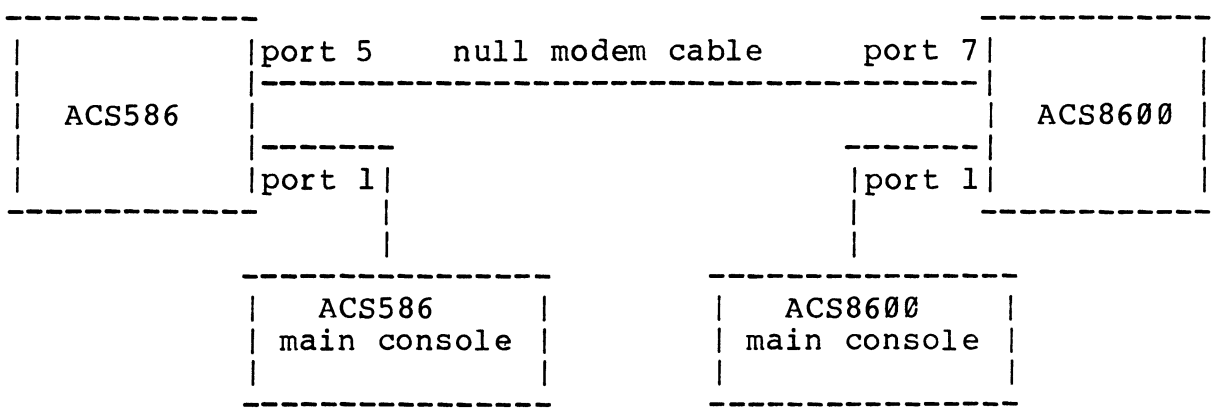
/usr/lib/uucp/uuxqt — This program is started by uucico to execute remote commands.

/bin/uulog — This program is normally executed every few days (or weeks, depending upon frequency of uucp usage). It cleans up the log files in the /usr/spool/uucp directory.

The following instructions describe how to connect an 586 to a 8600, and transfer files between the two systems using uucp. For simplicity, it is assumed that the two systems are connected by a single null-modem cable. The 586 will be configured as an active system, and the 8600 will be configured as a passive system. The reader of these instructions is expected to have some experience with basic Xenix utilities, and knowledge of at least one editor on the system.

Uucp Installation Instructions

1. Attach a CRT to port 1 on the 8600 and attach another CRT to port 1 on the 586. Then, attach a null modem cable between port 5 on the 586 and port 7 on the 8600. The resulting system configuration is shown below.



2. Boot both systems, and bring them up to multi-user (consult the Xenix installation instructions if you do not know how to do this).

3. Login as root on both systems. You must be logged in as root to install uucp. After it has been installed, any user can execute the uucp or uux commands.

4. On the active system (i.e., the 586), execute the command:

```
# disable  /dev/tty5
```

This ensures that the serial line tty5 will be used exclusively by uucp. On the passive system (i.e., the 8600), execute the command:

```
# enable  /dev/tty7
```

This ensures that there is a login process running on serial

line tty7. The uucico program running on the 586 will use this serial line to login on the 8600 whenever it needs to transfer files.

5. Use an editor to modify the file /etc/systemid on each system. This file must contain the name of the system on which the file resides. For simplicity, use the name Altos8600 for the 8600 system, and the name Altos586 for the 586 system.

6. Edit the file /usr/lib/uucp/L-devices. This file must contain at least one entry for each serial line used by uucp. Each entry has the form:

```
device  0  speed
```

where:

```
device isthe name of the serial line (e.g. tty5).
speed  is the baud rate of the line (e.g. 1200).
```

For our example, the L-devices file should contain:

```
tty5  0  9600
```

on the 586, and

```
tty7  0  9600
```

on the 8600.

7. Edit the file /usr/lib/uucp/L.sys. This file should contain one entry for each system which can be accessed. The information in this file indicates which systems are passive, and which systems are active. Each entry has the form:

```
system  time  device  speed  phone  login
```

where:

```
system  is the name of the remote system (e.g. Altos8600).
time    is a string indicating the time when the remote
        system should be called (e.g., MoTh0800-1730).
device  is the name of the serial line (e.g. tty5).
speed   is the baud rate of the line (e.g. 1200).
phone   is the same as device (e.g. tty5).
login   is a series of fields indicating how to login
        on the remote system.
```

The L.sys file on the 8600 should contain:

```
Altos586  Never  tty7  9600  tty7
```

This entry means that the 8600 is a passive system, so it should never attempt to call the 586.

The L.sys file on the 586 should contain:

```
Altos8600 Any tty5 9600 tty5 ogin:-^M-ogin:-^M-ogin: uucp
```

This entry means that the 586 can communicate with the 8600 using serial line tty5 at 9600 baud. Up to two carriage returns (^M) will be "typed" by uucico before it stops trying to login on the 8600. Uucico will login on the 8600 as user uucp, thus there must be an entry for uucp in /etc/passwd on the 8600.

Note that the symbol ^M indicates a RETURN (CTRL-M). Two RETURN's must be embedded in the file. If you are using vi to create the L.sys file, you must use the control-V option to embed control-M's in the file.

8. Edit the file /usr/lib/uucp/USERFILE. Each entry in USERFILE describes the accessibility of files on the local system. The information in this file is used to limit the accessibility of other users or systems. In our example, this file should contain the single entry:

```
root, /
, /usr /tmp
```

This will allow any remote user to read or write files under the directories /usr and /tmp. Only the user named root is allowed to access other files.

9. Edit the file /usr/lib/crontab. On an active system (i.e., the 586), the uucico program should be executed periodically to ensure that uucp requests initiated by the passive system (i.e., the 8600) are processed in a timely fashion. Also, on every system, the spool directory should be cleaned every few days. To run uucico on the 586 once every hour, enter this line in the crontab file:

```
00  0-23  *  *  *  /usr/lib/uucp/uucico -rl -sAltos8600
```

To remove all spool files older than three days every morning at 4:00AM, use this line:

```
00  04  *  *  *  /usr/lib/uucp/uuclean -pTM
```

10. The uucp package is now ready for use. To summarize, the following files should have the following data:

```
----------------------
|  Files on the 586  |
----------------------

/etc/systemid:
      Altos586

/usr/lib/uucp/L-devices:
      tty5  0  9600

/usr/lib/uucp/L.sys:
      Altos8600 Any tty5 9600 tty5 ogin:-^M-ogin:-^M-ogin: uucp

/usr/lib/uucp/USERFILE:
      root,  /
      ,  /usr /tmp

/usr/lib/crontab:
      0  0-23  *  *  *  /usr/lib/uucp/uucico -rl -sAltos8600
      0  4  *  *  *  /usr/lib/uucp/uuclean -pTM

-----------------------
|  Files on the 8600  |
-----------------------

/etc/systemid:
      Altos8600

/usr/lib/uucp/L-devices:
      tty7  0  9600

/usr/lib/uucp/L.sys:
      Altos586  Never  tty7  9600  tty7

/usr/lib/uucp/USERFILE:
      root,  /
      ,  /usr /tmp

/usr/lib/crontab:
      0  4  *  *  *  /usr/lib/uucp/uuclean -pTM
```

Example of a uucp transaction

To verify that uucp is working properly, execute this command on the 586:

```
uucp  /etc/passwd  Altos8600\!/tmp/passwd
```

This command move the file /etc/passwd from the 586 to the 8600, where it is stored in the directory /tmp. The following actions should occur:

1. The uucp program stores two files named C.Altos86n0001 and D.Altos86n0001 in the directory /usr/spool/uucp on the 586. The file prefixed with "D" is a copy of /etc/passwd; the other file contains control information. The message "QUEUED (C.Altos86n0001)" is entered in LOGFILE.

2. The uucico program is started by the uucp program, and the uucp program terminates.

3. Uucico checks the L.sys file on the 586, which tells it that it should immediately attempt to login on the 8600. It sends a carriage return to the 8600, which should respond with a login message. Uucico logs in on the 8600, and starts uucico on the 8600. Uucico creates two files in the directory /usr/spool/uucp. These files are prefixed with "LCK". They will be removed at the end of the transaction. The message "SUCCEEDED (call to Altos86)" is entered in LOGFILE.

4. Uucico checks the spool directory, and discovers that it should transfer a file from the 586 to the 8600. The message "REQUEST (S /etc/passwd /tmp/passwd username)" is entered in LOGFILE on the 586 and the 8600.

5. The file is transferred to the directory /usr/spool/uucp on the 8600. After the file has been completely transferred, it is moved to /tmp/passwd. The message "REQUEST (SUCCEEDED)" is entered in LOGFILE on both machines.

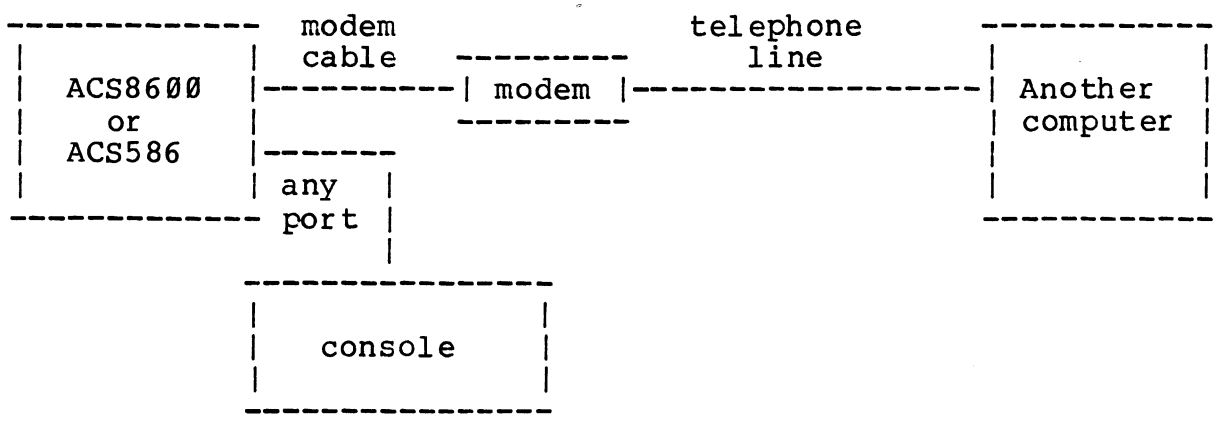
If uucp fails to transfer the file successfully, a message will probably be entered in LOGFILE on one or both machines. Failure is usually due to one of the following reasons:

1. The cable connecting the two machines was not built properly.

2. The files described in the Uucp Installation Instructions are not correct.

Section 3: How to use modems with Altos XENIX systems

Cu and uccp can be used for local dedicated point-to-point communication between computers in the same office, or for remote communication over telephone lines, in which case a modem must be used at both ends of the connection. Most commercially available asynchronous modems can be attached to an Altos Xenix system using a standard computer-to-modem cable. Examples of modems which have been used successfully with Altos systems include Racal-Vadic, Cermatek, and Hayes.



A special cable (usually called a "computer to modem" cable) must be used to attach a modem to either the 8600 or the 586. Most modem manufacturers and cable manufacturers supply these cables as off-the-shelf items. However, if the proper cable cannot be found, it may be necessary to build your own cable. To assist you, the RS-232C pins used by the 8600 and the 586 are shown below:

RS-232C connector on the 586 or the 8600.	Signal name		
	TD	--------> 2	usually connected to pin 3 on the modem
	RD	<-------- 3	usually connected to pin 2 on the modem
	RTS	<-------- 4	usually connected to pin 5 on the modem
	CTS	--------> 5	usually connected to pin 4 on the modem
	DSR	<-------- 6	usually connected to pin 20 on the modem
	GND	--------- 7	usually connected to pin 7 on the modem
	DCD	--------> 8	usually not connected
	DTR	<-------- 20	usually connected to pin 6 on the modem

Modems can be used with cu or uucp. To use a modem without auto-dial capability, manually dial the desired number to establish the connection. Then execute the cu command or one or more uucp commands. Cu and uucp will operate normally. At the conclusion of the session, you must manually dis-establish the phone connection (i.e., hang up the phone).

Modems with auto-dial capability can also be used. To use cu, simply execute the cu command. Most modems will ask one or more questions in response to a particular key being pressed. The modem will then automatically establish the connection.

A special program (named "dial") is necessary to use uucp with an auto-dial modem. An example of such a program is given on the next page. For example, the dial program can be invoked by the following entry in the file /usr/lib/crontab on the 586:

```
Ø Ø * * * dial /dev/tty5 4Ø85551234 /usr/lib/uucp/uucico-r1-sAltos86
```

This entry means that the dial program will be run every night at midnight. The dial program automatically dials the number 4Ø8-555-1234, and then starts the uucico program.

```
/*************************************************************************/
/*  The dial command has the format:
/*
/*     dial ttyline number program
/*
/*  where:
/*
/*     ttyline  is the name of the serial line
/*     number   is the phone number
/*     program  is the name of the program to run
/*
/*  For example:   dial /dev/tty5 4085551234 /usr/lib/uucp/uucico -rl -sAlto
/*
/*  This version of dial works only with the Cermatek 212A.
/*************************************************************************/
#include <stdio.h>
#include <signal.h>
#include <sgtty.h>
struct sgttyb termio;

main (argc, argv)
int argc; char **argv;
{
        int cflag, fd;
        char *p;

        signal (SIGHUP, SIG_IGN);          /* Ignore hang-up signal */
        setbuf (stdin, (char *)0);         /* Don't buffer standard input */
        setbuf (stdout, (char *)0);        /* Don't buffer standard output */
        setbuf (stderr, (char *)0);        /* Don't buffer standard error */
        signal (SIGALRM, SIG_DFL);         /* Allow 60 seconds to make */
        alarm(60);                         /*    the connection */

        if ((fd = open (argv[1], 2)) < 0) {
                printf ("Can't open %s\n", argv[1]);
                exit(1);

        ioctl (fd, TIOCGETP, &termio);  /* get the terminal i/o structure */

        /* if called as ldial, use 300 baud else use 1200 baud */
        termio.sg_ispeed = termio.sg_ospeed = (((p = rindex(argv[0], '/')) ?
                *++p : argv[0][0] ) == 'l') ? B300 : B1200;
        termio.sg_flags &= ~(ECHO | CRMOD); /* turn off echo and crmod modes
        termio.sg_flags |= CBREAK;         /* set cbreak mode */
        ioctl (fd, TIOCSETP, &termio);  /* restore the terminal i/o structure

        output ("\n\r\n\r", fd);           /* enter interactive mode of modem */
        wait ("NUMBER?...", fd);           /* wait for response from modem */
        output (argv[2], fd);              /* dial number */
        output ("\r", fd);                 /* output cariage return */
        wait ("DATA MODE.", fd);           /* wait for response */

        alarm (0);      /* connection made, turn off alarm */
        execvp (argv[3], &argv[3]); /* execute the remaining part of command
}
```

```
/* Output characters S L O W L Y */
output (p, fd)
char *p;
int fd;
{
        char c;

        while (c = *p++));
                sleep (2);
                write(fd, &c, 1)
        }
}

wait (s, fd)
char *s;
int fd;
{
        char c;
        char *p;
        int unget = 0;

        while (1) {
                p = s;  /* point to the strings to be compared */
                /* read until the first character matches */
                do {
                        if (unget)
                                unget = 0;
                        else {
                                read (fd, &c, 1);

                } while (c != *p);
                while (1) {
                        /* done if last character */
                        if (*++p == '\0') {
                                return;
                        }
                        read (fd, &c, 1);  /* read in the next character
                        /* if the next character doesn't compare ... */
                        if (*p != c) {
                                /* start all over again */
                                unget = 1;
                                break;
                        }
                }
        }
}
```