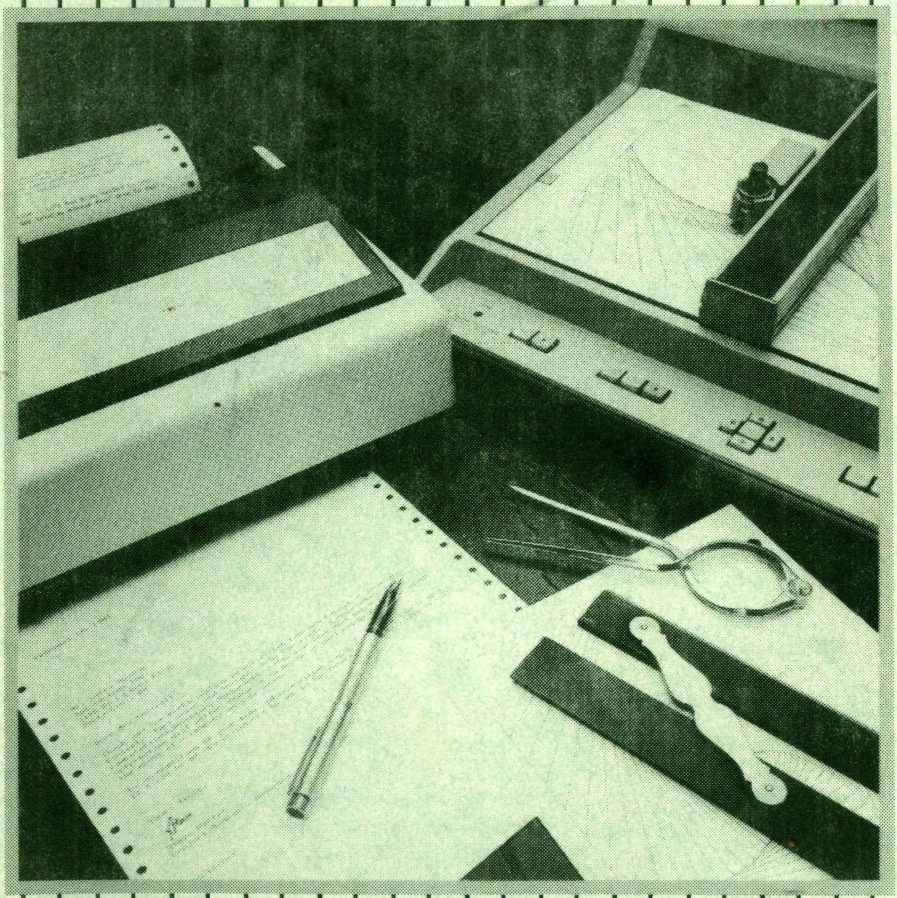
Universal Parallel Interface Card
Installation and Operating Manual

Apple III



Notice

Apple Computer reserves the right to make improvements in the product described in this manual at any time and without notice.

Disclaimer of All Warranties And Liabilities

Apple Computer makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Apple Computer software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Apple Computer, its distributor, or its retailer) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Apple Computer be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Apple Computer has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

Reorder Apple Product #A3L0009

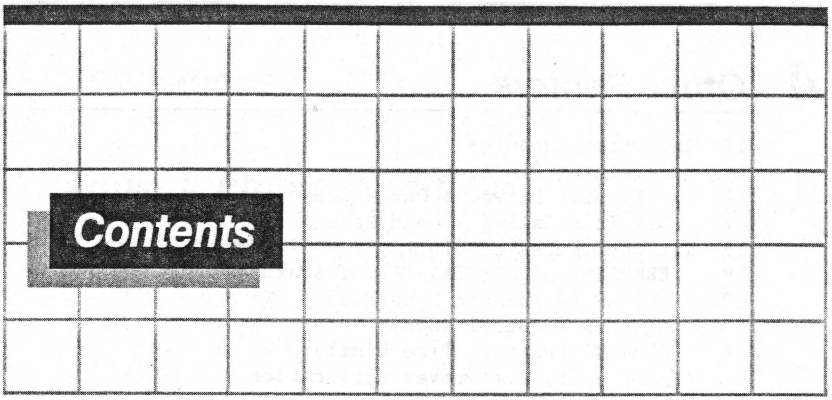
WARNING: This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Apple III

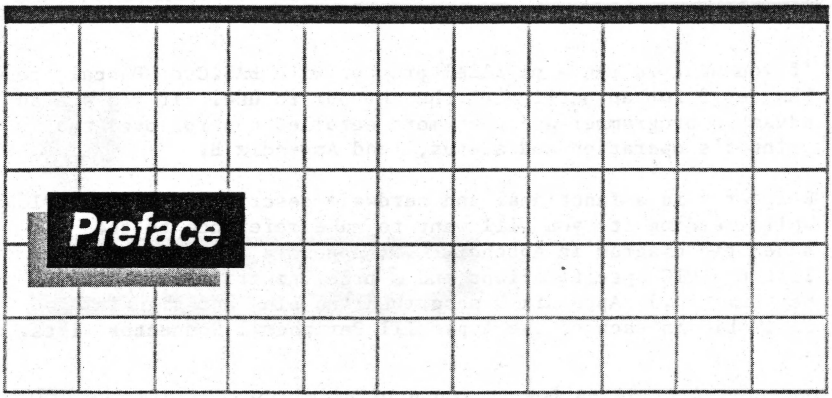
Universal Parallel Interface Card

Installation and Operating Manual

NOTE: PLEASE READ THIS MANUAL BEFORE ATTEMPTING TO INSTALL THE APPLE III UNIVERSAL PARALLEL INTERFACE CARD IN THE APPLE COMPUTER. INCORRECT INSTALLATION COULD CAUSE PERMANENT DAMAGE TO BOTH THE PERIPHERAL DEVICE AND THE APPLE COMPUTER.



Appendices



This manual tells you how to install the Apple III Universal Parallel Interface Card (UPIC) and how to use it to perform both simple and complex tasks. Before you read this manual, first read the Apple III Owner's Guide and Chapters 1 and 2 of the Apple III Standard Device Drivers manual.

This manual has six chapters and five appendices. If your plan is simply to install the UPIC, connect one of the commonly used printers, and then start using it immediately, Chapters 1 and 2 may be all you need to read. Chapter 1 explains how to install the UPIC in your Apple III. Chapter 2 gives general instructions for connecting the cable from the UPIC to a device (for example, to a printer or another computer), and for setting up the Device Driver needed for data transfers. Chapter 2 also contains all the specific information you need if you are going to use one of these commonly used printers:

Centronics 700, 730, 737 and 779
Anadex DP-8000
Printronix P300
IDS 440, 445, 460
Epson MX-80
Texas Instruments 810

If you wish to attach a different type of parallel printer, read Chapter 1, then Chapter 4, then Chapter 2. To connect your Apple III to another Apple III, another type of computer, or a sophisticated printer that requires a 40-pin connection, read Chapter 1, then Chapter 5, then Chapter 2.

If you want to run a parallel printer with BASIC or Pascal, read Chapter 3 for an outline of the methods to use. If you are an advanced programmer and want more detailed control over the printer's operation and status, read Appendix B.

Chapter 6 is a functional and hardware description of the UPIC. While reading it, you will want to make reference to the schematic diagram in Appendix C. (Appendix C also contains a list of UPIC specifications and a brief description of its performance.) Appendix D describes the pins and signals available in each of the Apple III Peripheral Connector slots.

Appendix A explains how to run a parallel printer via the UPIC when the Apple III is in Apple II Emulation mode.

Appendix E contains an ASCII conversion table.

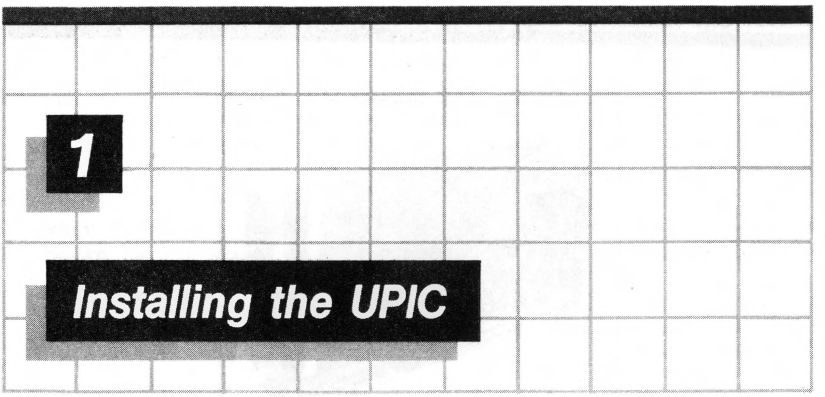
Appendix F explains step-by-step how to prepare the Printer or Parallel Driver and install it on your boot diskettes if you have Version 4.Ø of the System Utilities diskette.

The glossary has definitions for most of the important or unusual terms used in this manual.

This manual uses two symbols to point out paragraphs of special interest. The symbols are:

The hand points to paragraphs that contain especially useful or noteworthy information.

The eye gazes on paragraphs that explain an unusual feature to look out for.



1 Installing the UPIC

The Apple III Universal Parallel Interface Card (UPIC) is a simple yet powerful extension to your Apple III. Using nothing other than the UPIC, a cable, and what is supplied on the diskette that came with the UPIC, you can set up your Apple III to do eight-bit parallel data transfers either to a printer, or both to and from a terminal or another computer. This chapter describes how to unpack, prepare and install the UPIC.

Unpacking

The package your Apple III Universal Parallel Interface Card (UPIC) came in should include:

- the UPIC itself
- this manual
- a diskette containing the UPIC Driver Software
- a Warranty Registration Card
- an Apple III User Input Report Form
- a packing list

Fill out the Warranty Registration Card and mail it in right away. If any item is missing, contact the Apple dealership where you purchased the UPIC.

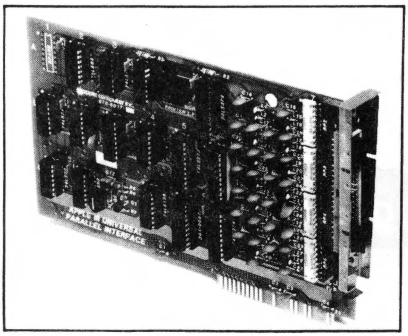


Figure 1-1. The Universal Parallel Interface Card (UPIC)

Preparing a Backup Diskette

The diskette that came with this product contains important software you will need for as long as you use the UPIC. Lest anything happen to your one and only copy of this diskette, we strongly urge you to make a duplicate copy. Use the Copy Diskette utility program described in the Apple III Owner's Guide. Then put the original diskette in a safe place, but not too safe: you may need to find it again. Your backup copy of the UPIC diskette is now the one you should use.

Preparing the UPIC

Set the switch marked Printer L.F. on the UPIC to NORM; that is, the same way it is set in Figure 1-1 (unless you plan to use the UPIC in Apple II Emulation mode, in which case refer to Appendix A). The UPIC is now ready to install.

Before opening the Apple III, be sure that the power is turned off. However, DO NOT unplug the Apple III. The ground connection through the power cord helps prevent damage to the Apple III's internal parts.

The Installation Procedure

First, you must remove the cover of the Apple III. Turn the Apple III over and locate the two cover attachment screws halfway up each side of the bottom panel. Loosen each screw a quarter turn, and then turn the Apple III right-side up with the keyboard facing you. Now, lift the top cover and carefully pull it toward you and off.

Figure 1-2 shows the four Apple III peripheral connector slots. You can insert the Apple III UPIC in any one of these slots, although for Apple II Emulation mode you ordinarily install it in slot 1 (see Appendix A).

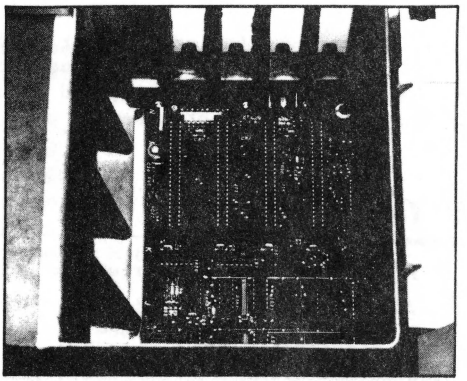


Figure 1-2. Peripheral Connector Slots

Remove the dummy card, if necessary, from the slot you intend to use. Then grasp the upper corners of the UPIC so that the gold fingers of the edge connector point down, and the printer cable connector is toward the back of the Apple III. Insert the UPIC in the chosen slot, making sure the metal Radio Frequency Interference (RFI) shield fits snugly against the Apple's case and the front edge of the UPIC slides into the vertical guide. Gently rock the UPIC back and forth until it is firmly seated. Figure 1-3 shows how the UPIC looks if installed in slot 1.

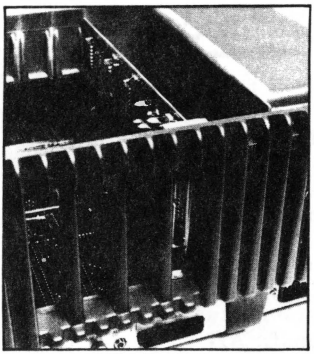


Figure 1-3. The UPIC Installed in Slot 1

Replace the top, carefully slipping the four tabs along the back of the cover into the corresponding four square holes in the case. Turn the Apple upside down again, and then push in and retighten the two cover attachment screws.

The Final Two Steps

There are two final steps you must perform before you can run a device with the UPIC:

- prepare and connect the cable
- set up the device driver

Chapter 2 explains how to perform these steps, and includes all the specifics necessary to prepare any one of the parallel printers most commonly used with the Apple III.

If you intend to attach and run some other kind of parallel printer, read Chapter 4 first, then follow the general procedure given in Chapter 2. If, on the other hand, you intend to attach your Apple III to a sophisticated printer or terminal, or to another computer--for example, a second Apple III--read Chapter 5 and then Chapter 2.

2 Attaching the Device

This chapter describes what you must do to set up your Apple III with any of these parallel printers:

- Centronics 779, 7ØØ, 73Ø or 737
- Anadex DP 8ØØØ
- Printronix P3ØØ
- IDS 44Ø, 445 or 46Ø
- Epson MX-8Ø
- Texas Instruments Model 81Ø

If you have another type of parallel printer, first read Chapter 4 to determine what special settings and connections you need, then follow the procedures in this chapter. If, on the other hand, you intend to connect a sophisticated printer, a terminal, or another computer (such as a second Apple III) via the UPIC, read Chapter 5 first, then this chapter.

The Simple Printer Connection: 20 Pins

You are going to attach your printer to the Apple III using the 2Ø center pins of the 4Ø-pin connector that now protrudes from the rear of the Apple III's case. The remaining pins are concealed and protected by two plastic shields, as shown in Figure 2-1. Note carefully the arrangement of the pin numbers.

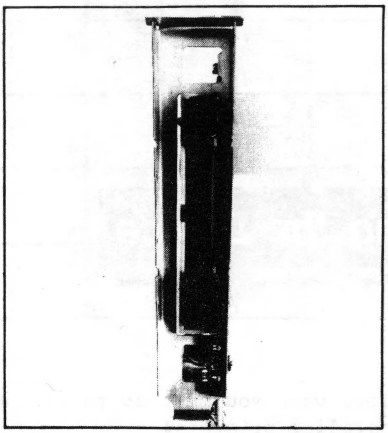


Figure 2-1. The 2Ø-Pin UPIC Male Connector

The Printer Cable

If you already have an Apple parallel printer cable (from an Apple II Parallel Printer Card) you can connect it to the Apple III without modification. However, you then must set the first two parameters in the DCB (see Table 2-2) to ØØ.

Your Apple dealer ordinarily has ready-made cables in stock for Centronics 737 and 779 printers, as well as the equipment necessary to prepare cables for connection to other devices.

If you have a cable with a 2Ø-pin connector already attached at the UPIC end (also available from your Apple dealer), and you want to attach the connector on the printer end of the cable, use Table 2-1 as a guide. The UPIC pin numbers are those stamped on the 2Ø-pin connector. The printer connector pin numbers refer to the lines as they are designated in each printer's manual.

If for any reason you are also furnishing your own 2Ø-pin female connector for the UPIC end of the cable, be sure it has a mating key (Figure 2-2) for the key slot on the UPIC's male connector. Unkeyed connectors can be plugged in the wrong way--and that can spell disaster.

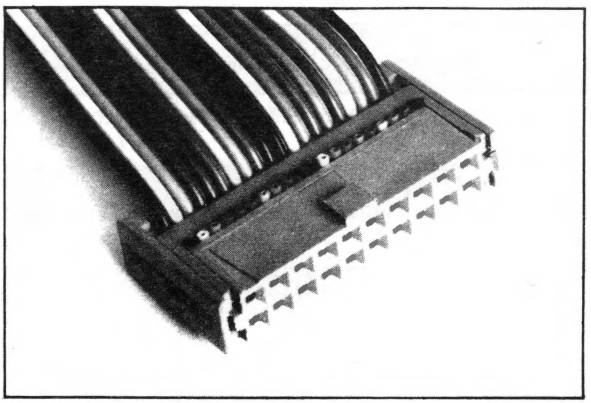


Figure 2-2. A Keyed 2Ø-Pin Female Connector

If you are making the interconnecting cable yourself, use a good grade of stranded ribbon cable of 26-28 gauge wire. Stranded wire can withstand much more flexing than solid wire, and will give you far less trouble.

Figure 2-3 shows a sample cable with a Printronix connector.

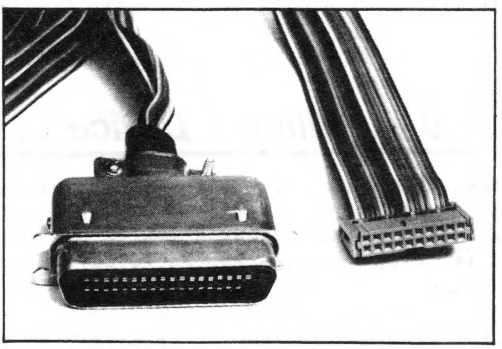


Figure 2-3. A Sample Cable

If the 2Ø-pin connector on your cable does not have a mating key, you must be absolutely certain that the connectors are properly aligned. Improper alignment can result in severe damage to your UPIC.

UPIC Pin #	Printer Connector Pin Number						
	Centronics (779/7ØØ)	(73Ø/737)	Anadex DP-8ØØØ	Printronix P3ØØ	IDS (44Ø/45/6Ø)	Epson MX-8Ø	TI 81Ø
1	14	27	14	14	7	9,19	19
2	1Ø	19	1Ø	1Ø	22	1Ø	1Ø
3							
4							
5							
6						32	32
7							
8	1	1	1	1	3	1	1
9	12		12	12	12	12	12
1Ø	2	3	2	2	14	2	2
11	3	5	3	3	13	3	3
12	4	7	4	4	12	4	4
13	5	9	5	5	11	5	5
14	6	11	6	6	1Ø	6	6
15	7	13	7	7	9	7	7
16	8	15	8	8	15	8	8
17	9	17	9				9
18	13	25	13	13	4	13	13
19	18	28	18	18		35	18
2Ø	16	31	16	9,16	7	3Ø,33	16,27

Table 2-1. Pin Assignments for Commonly Used Printers

Preparing and Installing a Device Driver

The Apple III does not "talk" directly to whatever is connected to it. Instead, the Apple uses a set of computer instructions called a device driver to pass information to and from whatever is connected to it. Two of these device drivers are furnished on the diskette that comes with the UPIC:

-One is for sending information to parallel printers (a driver called .PRINTER).

-One is for exchanging information with a remote terminal or another computer, such as another Apple III (a driver called .PARALLEL).

This section gives the general procedure for tailoring a driver to the device you are using, and then adding it to the collection of drivers on each of your "boot" diskettes. This section also provides all the specific settings you need to tailor the Printer Driver (.PRINTER) to any one of the printers listed at the start of this chapter.

Why Device Drivers?

Each time you start up ("boot") your Apple III, it loads into memory not only the operating system (SOS), but also all the device drivers from the file SOS.DRIVER on the boot diskette you are using. Each driver is designed to handle or "drive" one of the devices connected to the Apple III. The Apple III can communicate only with those devices represented by device drivers, and to no others. Thus, every boot diskette you want to use with a parallel device must contain either the Printer Driver (.PRINTER) or the Parallel Driver (.PARALLEL).

The General Technique

The actual sequence of keystrokes required to perform the procedure explained in this section can be found in the Apple III Standard Device Drivers manual.

If you have Version 4.Ø of the Utilities Diskette, refer to Appendix F for the exact keystrokes to use.

First make a backup copy of the UPIC diskette (Chapter 1) and store it in a safe place. To prepare the boot diskettes:

1. Bring the appropriate driver into the Apple III. Insert the System Utilities diskette in the built-in drive and boot the Apple III. From the menu, select the System Configuration Program (SCP) and the Add a Driver function.

2. Modify the driver's Configuration Block for your specific device. Table 2-2 (below) gives the correct values for a number of commonly used printers. Chapters 4 and 5 explain how to derive these values for other printers or parallel devices.

If you connect a printer to the UPIC with an Apple II Parallel Printer Card cable, you must set each of the first two parameters of the Configuration Block to ØØ.

3. If the UPIC is in a slot other than #1, change its Peripheral Slot Assignment.

4. Save the configured driver on the UPIC diskette.

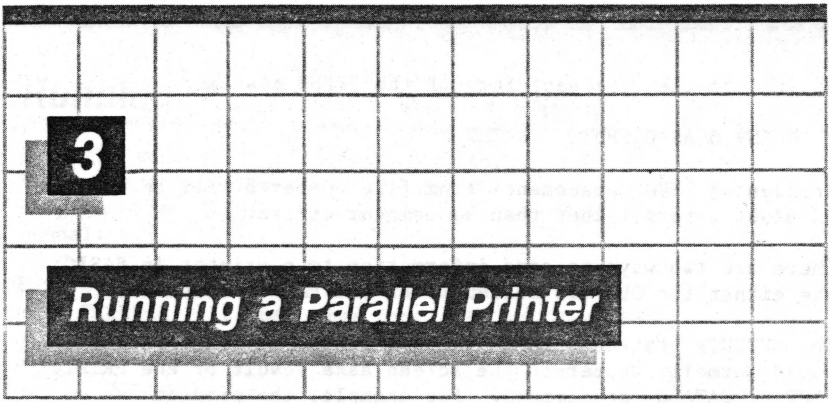
5. Prepare each boot diskette for the new driver by removing or renaming, if necessary, any existing driver named .PRINTER that is on that diskette.

6. Bring the configured driver into the Apple III from the UPIC diskette.

7. Generate a new system for the boot diskette, with the new driver in it. If there are errors, repeat the first four steps given above, following closely the procedures given in the Apple III Standard Device Drivers manual.

8. If there are no errors in the new system configuration, save it on the boot diskette you are changing. Repeat steps 5 through 8 until you have reconfigured all the boot diskettes you plan to use with the .PRINTER or .PARALLEL driver.

In Apple II Emulation mode, the system does not use device drivers. Therefore, you do not need to add a driver to any diskettes you use in that mode.



3 Running a Parallel Printer

This chapter describes the commands to put in BASIC and Pascal programs to run a parallel printer. It assumes that you know how to write programs in at least one of the two languages; however, even if you don't, you can type in the BASIC examples to see how they work.

For these examples to work, you must first follow all the procedures in Chapter 2 — including preparation of the .PRINTER driver to work with your specific printer. Also, if you have given .PRINTER a different name, you should modify the sample programs accordingly.

Running the Printer with BASIC

This section illustrates general techniques for using your printer with BASIC. It gives examples of commands, but does not explain all their variations. For details, read the Apple III Business BASIC manual.

Before a BASIC program can send anything to the parallel printer, the program must assign a file reference number to it with an OPEN# statement.
For example:

```
OPEN# 5, ".PRINTER"
```

This assigns file reference number 5 to the device named .PRINTER. After this particular OPEN# statement, all information sent to file number 5 will be directed to the printer, unless you CLOSE# 5 or OPEN# 5 as another device. Note that the file reference number has no relation to the number of the slot where you installed the UPIC.

If you use the alternate form of the OPEN# statement

```
OPEN# 5 AS OUTPUT, ".PRINTER"
```

accidental INPUT statements from file number 5 will be flagged as BASIC errors rather than as printer errors.

There are two ways to send information to a printer in BASIC: use either the OUTPUT# statement or the PRINT# statement.

The OUTPUT# statement sends to the printer all information that would normally appear on the screen as a result of the PRINT, LIST or CATALOG statements. For example, the program

```
10  OPEN# 5 AS OUTPUT, ".PRINTER"
20  OUTPUT# 5
30  LIST
```

prints its own three lines on the printer. A program that uses the OUTPUT# statement to send data to a device cannot send characters to the screen with the PRINT, LIST or CATALOG statements until another statement in the program, such as OUTPUT# Ø, redirects the information to the console.

The second way to send information to the printer is to use the PRINT# statement. Although you must specify the file number each time you use PRINT#, you can send information to the screen more easily than with OUTPUT#. For example the program

```
10  OPEN# 5, ".PRINTER"
20  PRINT# 5; "This is all the news that's fit to PRINT."
30  PRINT "This is a screen test."
```

sends the first quoted sentence to the printer, and the second one to the console. Try it!

Use a CLOSE# statement to remove the links produced by the OPEN# statement. For example, use CLOSE# 5 to "close" the .PRINTER in its guise as file number 5.

If you are a reasonably experienced programmer, and you want to have speedy and detailed control over the workings of the printer, you can use the control and status requests provided on the UPIC diskette in the form of invocable modules. Appendix B explains the requests, and shows you how to invoke them.

Running the Printer with Pascal

Like the section on printing with BASIC, this section gives a few general examples of how to run the printer from a Pascal program. For details, read the Apple III Pascal Programmer's Manual.

Before your program can communicate with the file that it will call PRINTER:, it must give that file an identifier and a type. So first define it as a variable with identifier P (for example), and as type "text" or "file of characters," using one of the following declarations:

```
VAR P:TEXT;     or
VAR P:FILE OF CHAR;
```

Next, equate the identifier P with the driver name .PRINTER using the REWRITE command:

```
REWRITE(P,'PRINTER:');
```

This command also prepares the printer to receive information from the program.

Pascal automatically drops the dot before Apple III driver names, and adds a colon after it (for example, changing .PRINTER to PRINTER:). This makes the names consistent with the pathnames in UCSD Pascal, from which Apple III Pascal is derived.

Now you can send information to the printer using WRITE statements; or you can use the WRITELN statement, which automatically sends a carriage return at the end of the information to print. For example

```
WRITELN(P,'If Peter Piper printed Pascal programs painlessly');
WRITE(P,'Where''s a painlessly printed Pascal program');
WRITELN(P,' Peter Piper printed?');
```

causes your tongue to tangle on two lines of a printed page, while

```
WRITE('She sells shilling sea shells');
WRITE('on the sharp-sloped seashore');
```

claims but one of the console's 24 lines. (Note: if you don't specify P, The information automatically goes to the console.)

When you have finished sending information to the printer, close its file with the statement

```
CLOSE P;
```

That's that.

Here is a small but complete Pascal program to try:

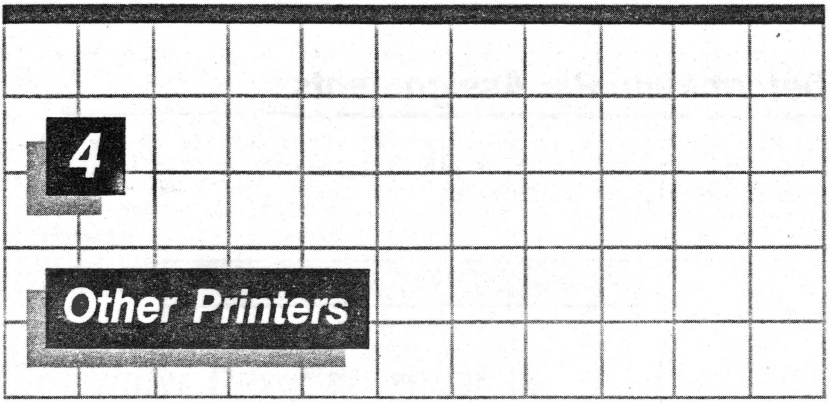
```
PROGRAM PRINT_IT;
VAR P:TEXT;
    Y,Z:INTEGER;                { Y and Z are indexes for loops}
BEGIN
  REWRITE(P,'PRINTER:');          { Prepare the printer.    }
  FOR Y := 1 TO 5 DO BEGIN        { On 5 separate lines     }
    FOR Z := 1 TO Y DO WRITE(P,' ');{   print Y-1 blanks    }
    WRITELN(P,'*')                {     followed by *.      }
  END;
  CLOSE(P)                        { Close the printer file.}
END.
```

This program produces the following on the printer:

```
*
 *
  *
   *
    *
```

Status and control requests (Appendix B) are available on the UPIC diskette in the form of invocable modules for use with the standard Apple III Pascal procedure 'UNITSTATUS.'

The .PRINTER driver can also run a printer in Apple II Emulation mode. Read Appendix A for details.



This chapter gives you the information you need if you want to configure the Printer Driver to a parallel printer other than the kinds listed in the Preface. The first section discusses the UPIC device drivers in general. The next section gives the pin assignments for construction of a cable; the final two sections explain the Printer Driver and its Configuration Block values.

The exact purpose of each wire coming out of the Apple III UPIC is specific to each application. The design of the UPIC enables the driver to control the relationships between the signals coming into the UPIC from a device and the signals going out of the UPIC to that device.

The Driver Modules

Two code modules, known as device drivers, come with the Apple III UPIC. Each is written for a special purpose, and each sends out and expects to receive signals and data according to a certain format.

The first module, the Printer Driver (.PRINTER), sends data to parallel printers, which receive one character at a time. This driver uses a 20-pin connection. The remainder of this chapter tells you how to connect the UPIC to a printer--even ones not listed in the Preface--and what values to set for .PRINTER to run it. Once you have determined these connections and values, follow the procedures given in Chapter 2.

The second module, the Parallel Driver (.PARALLEL), uses a 40-pin connection. This driver is discussed in Chapter 5.

Determining Pin Assignments

The Printer Driver requires only the center 2Ø pins on the UPIC connector. The 2Ø connector pin assignments, as used by the Printer Driver, are as follows.

Pin Number	Function
1	Signal Ground
2	ACKNOWLEDGE INPUT
3	Data Input Bit Ø
4	Data Input Bit 1
5	Data Input Bit 2
6	Printer In Check Input
7	Printer Ribbon Out Input
8	STROBE OUTPUT
9	Printer Out of Paper Input
1Ø	Data Output Bit Ø (LSB)
11	Data Output Bit 1
12	Data Output Bit 2
13	Data Output Bit 3
14	Data Output Bit 4
15	Data Output Bit 5
16	Data Output Bit 6
17	Data Output Bit 7 (MSB)
18	Printer Online Input
19	Printer Power On Input
2Ø	Signal Ground

Table 4-1. The 2Ø-Pin Connection

Table 2-2 (in Chapter 2) shows pin assignments for several commonly used printers, namely:

- Centronics 779 or 7ØØ
- Centronics 73Ø and 737
- Anadex DP 8ØØØ
- Printronix P3ØØ
- IDS 44Ø or 445 or 46Ø
- Epson MX-8Ø
- Texas Instruments Model 81Ø

If your printer is one of these, Chapter 2 lists all the pin connections and driver values for you. If you have some other type of printer, look in the printer's instruction manual to

determine which of the pins shown in Table 4-1 corresponds to each of the pins on your printer. Use this information to fashion a connecting cable from the UPIC to that printer.

If you are not sure what all the signal lines are, or if your printer's instruction manual calls them by other names, read on. The next section describes the function of each of the lines. You should be able to deduce the proper line connections by comparing these descriptions with those in your printer's instruction manual.

The Printer Driver's Handshake

The Printer Driver (.PRINTER) is designed to pass information between an Apple III and most types of parallel printers. This driver is recorded on the diskette that comes with the Apple III UPIC. The section of Chapter 2 called The General Technique explains how to add .PRINTER to your system configuration.

The following sections describe the exchange of signals (called a "handshake") that takes place between the Printer Driver and a printer—in other words, the way the Printer Driver uses the lines you connect via the cable.

Why It's Called a Handshake

Each character that the Apple III UPIC sends to a printer is sandwiched in an exchange of signals on the STROBE OUTPUT and ACKNOWLEDGE INPUT lines. This exchange is very much like one person handing a dollar to another via a handshake:

Person A (extending a hand with the bill hidden in it):
"Hi, how are you!"

Person B (shaking hands, receiving bill, and reacting automatically): "Just fine, thanks. Got any more?"

When the UPIC is ready to send a character to the printer, it places that character on the eight data lines, waits 5 microseconds for the data signals to be set up, and then sends the STROBE OUTPUT signal. When the printer detects the STROBE OUTPUT signal, it receives the character for printing. The printer then sends the ACKNOWLEDGE INPUT signal to indicate that it has accepted the character and is ready for a new one. This is the Printer Driver's Handshake (Figure 4-1).

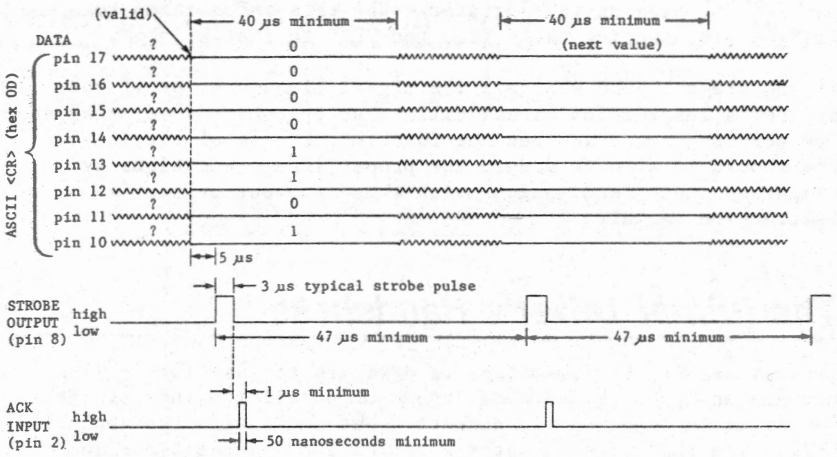


Figure 4-1. The Printer Driver's Handshake

There is another kind of exchange between the UPIC and the printer. Stretching the handshake analogy a bit further:

Person A: "Here's a report form. Tell me what condition you're in to use what I've given you."

Person B: "I've filled out the report. Knowing me as you do, decide for yourself."

This sort of followup conversation is like a control or status request and its reply (Appendix B).

The order of events in the Printer Driver Handshake, or in a control or status request, never varies. However, certain other things do vary from one printer to another. These variable elements, or "parameters," are stored in the Device Configuration Block (DCB):

- what bits of information to check when evaluating the status of the printer (ERRMASK; DCB position Ø)
- whether each bit of an OK status from the printer should be ON or OFF (ERRSTAT; DCB position 1)
- whether or not to suppress sending to the printer any <LF> that immediately follows a <CR> (AUTOLF; DCB position 2)

- how long to sustain the STROBE OUTPUT signal; what polarity to interpret as meaning that the STROBE OUTPUT and ACKNOWLEDGE INPUT signals are present (CTRLWRD; DCB position 3)

- how long to wait for an ACKNOWLEDGE INPUT signal reply from the printer (TIMEOUT; DCB position 4)

You may find some of these parameters a bit strange. Be patient: the next section will explain how they work.

If you examine Table 4-2, you will notice that it is the same as Table 2-2, with the addition of the names of each of these five parameters. Once you have determined their respective values for a given printer, you are ready to put them into the DCB and install the Printer Driver following the procedures given in Chapter 2.

Printer	Device Configuration Block (DCB) Values				
	ERRMASK ØØ	ERRSTAT Ø1	AUTOLF Ø2	CTRLWRD Ø3	TIMEOUT Ø4
Centronics 779/7ØØ	EØ	CØ	4Ø	ØØ	ØA
Centronics 73Ø/737	CØ	CØ	ØØ	ØØ	5A
Anadex DP-8ØØØ	EØ	CØ	ØØ	ØØ	5A
Printronix P3ØØ	EØ	CØ	ØØ	ØØ	ØA
IDS 44Ø/445/46Ø	6Ø	4Ø	ØØ	ØØ	5A
Epson MX-8Ø	E8	C8	ØØ	ØØ	ØA
TI 81Ø	E8	CØ	ØØ	ØØ	ØA
Any printer connected with Apple II cable	ØØ	ØØ	(final 3 same as above)		

Table 4-2. DCB Values for Commonly Used Printers

The Configuration Block

A group of parameters called a Device Configuration Block (DCB) contains the information that tailors each driver to the device connected to it. In the case of .PRINTER, these DCB parameters are the five listed in Table 4-2 and discussed below: ERRMASK, ERRSTAT, AUTOLF, CTRLWRD, and TIMEOUT.

ERRMASK and ERRSTAT (Error Status)

.PRINTER monitors printer status through the eight lines of the input port. These eight lines correspond to pins 3, 4, 5, 6, 7,

9, 18 and 19 of the UPIC connector. Each bit of the byte ERRMASK corresponds to one of these input port lines (see Table 4-3). The Printer Driver monitors each status line whose corresponding bit is set to 1 in ERRMASK.

The eight bits of ERRSTAT correspond to exactly the same input port lines. Each bit should be set to the value that the printer sends over the corresponding line under error-free conditions. Unused bits must be set to Ø.

No matter what printer you use, if you connect it via an Apple II Parallel Printer cable, you must set both ERRMASK and ERRSTAT to ØØ.

Table 4-3 shows the pin number and interpretation of the five bits of ERRMASK and ERRSTAT that .PRINTER uses.

Bit:	7	6	5	4	3	2	1	Ø
Printer:	Power On	On Line	Out of Paper	Ribbon Out	In Check			
Pin:	19	18	9	7	6	5	4	3

ERRMASK -- If bit = 1, corresponding status line is checked
ERRSTAT -- Bits should equal values normally expected

Table 4-3. ERRMASK and ERRSTAT Bits

The Printer Driver tests for error conditions as follows. It ANDs the eight bits of the input port with ERRMASK. This turns off the the unused input bits if they happen to be on. It then EXCLUSIVE-ORs that result with ERRSTAT. If any bits of the result are still set to 1 after all that, those bits indicate printer errors.

You can use SOS status requests to determine exactly which printer error has occurred, and have your program take the appropriate action. Appendix B explains these control and status requests.

AUTOLF (Automatic Linefeed)

If the printer you are using generates its own Linefeed (<LF>) character automatically each time it encounters a Carriage

Return (<CR>), set AUTOLF to 4Ø. .PRINTER will then automatically suppress sending to the printer any <LF> that immediately follows a <CR>.

If, on the other hand, the printer does not have this "Auto Linefeed" feature, or you have turned it off, set AUTOLF to ØØ: .PRINTER will simply send all <LF> characters, no matter where they occur.

CTRLWRD (Control Word)

CTRLWRD determines the duration of the STROBE OUTPUT pulse (any odd whole-number value from 1 through 15 microseconds), and the polarity of the STROBE OUTPUT and ACKNOWLEDGE INPUT pulses: either negative and clocked on the rising edge, or positive and clocked on the falling edge.

The usual setting of CTRLWRD is ØØ; in fact, CTRLWRD is reset to zero whenever you reboot the system. As a result, CTRLWRD is rigged up so that when set to zero it sets pulse polarities at negative and strobe duration at 3 microseconds. The reason? Those printers most frequently used with the Apple III run just fine when CTRLWRD is set to zero. Check Table 4-4 and see.

The strobe bits are tricky. Look closely at the interpretation of bits Ø, 1 and 2 in Table 4-4. Remember that all zeros means 3 (that is, 3 microseconds). Believe it or not, you can set those bits to mean any odd number from 1 through 15. For example, to get 13, set all three bits to 1 (3 - 2 + 4 + 8).

Bit:	7	6	5	4	3	2	1	Ø
"1" =			Strobe Positv		Ack In Positv	Strobe +8 µs	Strobe +4 µs	Strobe -2 µs

Table 4-4. Meaning of Ones in CTRLWRD

TIMEOUT (Waiting Time Limit)

TIMEOUT determines the maximum time .PRINTER is to wait for the ACKNOWLEDGE INPUT signal from the printer. If the Printer Driver is trying to send a character to the printer, and it doesn't receive the acknowledge signal within (TIMEOUT x 11) microseconds, it sets interrupts on the UPIC, and then returns to whatever it was doing previously. When UPIC interrupts are set, the Apple III resumes sending characters (and turns the interrupts off) as soon as it receives the ACKNOWLEDGE signal.

A parallel printer generally loads characters rapidly into a temporary storage area called a line buffer until the entire line is in this buffer; then it prints out the line, which takes between Ø.2 and 5 seconds. The printer cannot send the ACKNOWLEDGE INPUT signal until it has printed the entire line.

The Apple III can spend this Ø.2 to 5 second interval doing much more productive things than waiting for the ACKNOWLEDGE signal; for example, it could be processing data in an applications program. Therefore, TIMEOUT should represent more time than the printer needs to accept a character, but less time than it needs to print a line.

For example, for printers such as the Centronics 779 and Printronix P3ØØ, the interval between the STROBE pulse and the ACKNOWLEDGE pulse is typically less than 1Ø microseconds. A waiting period of 11Ø microseconds (TIMEOUT = $ØA) is adequate to ensure that not too much time is spent waiting for the ACKNOWLEDGE signal while a line is being printed.

It is much better to have a TIMEOUT that is too big than one that is too small. If the WAIT is less then the printer's response time, an interrupt will be generated for every character transferred. This would seriously degrade the performance of the system.

A Sample Printer Driver Application

Appendix B explains how to use the control and status requests to get the best possible performance from your printer. The following is merely an example.

Normally, if you send data to the printer while it is turned off, neither the printer nor the program will do anything.

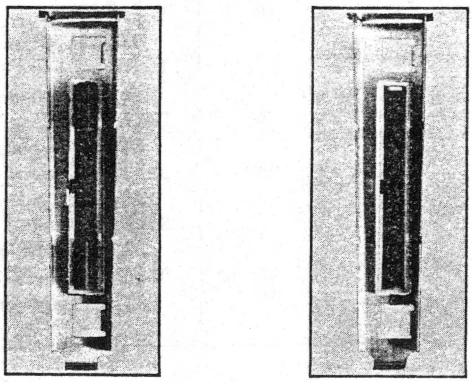
You can avoid this condition using Printer Driver status request number 3. The printer will send back not only its status, but also the number of characters currently in the output buffer and the buffer's size. As a result, your program can always check for possible errors before sending data, and arrange to send only as many characters as the output buffer currently has room for.

5 Other Parallel Devices

This chapter explains the pin connections from the UPIC to a sophisticated printer, a terminal, or another computer. It also describes how to set up the Parallel Driver (.PARALLEL). Once you have done this, follow the procedures in Chapter 2.

The Parallel I/O Connection: 40 Pins

The Parallel Driver uses a 40-pin connector. Before using it, you need to remove the two plastic shields from the UPIC connector by grasping each one by the tab and pulling firmly. They should pop right out. Figure 5-1 shows how the connector looks before and after removing them. Table 5-1 shows the pin numbering schemes and signal assignments.



Before (20 pins) After (40 pins)

Figure 5-1. The UPIC Connector

20-pin Connection		40-pin Connection		Signal Definition
		1	2	Port B output DØØ Port B output DØ1
		3	4	Port B output DØ2 Port B output DØ3
		5	6	Port B output DØ4 Port B output DØ5
		7	8	Port B output DØ6 Port B output DØ7
		x	x	(pin removed) (pin removed)
1	2	11	12	Signal Ground ACKNOWLEDGE INPUT
3	4	13	14	Port B input DIØ Port B input DI1
5	6	15	16	Port B input DI2 Port B input DI3
7	8	17	18	Port B input DI4 STROBE OUTPUT
9	1Ø	19	2Ø	Port B input DI5 Port A output DØØ
11	12	21	22	Port A output DØ1 Port A output DØ2
13	14	23	24	Port A output DØ3 Port A output DØ4
15	16	25	26	Port A output DØ5 Port A output DØ6
17	18	27	28	Port A output DØ7 Port B input DI6
19	2Ø	29	3Ø	Port B input DI7 Signal Ground
		x	x	(pin removed) (pin removed)
		33	34	DATA READY OUTPUT Signal Ground
		35	36	Signal Ground Signal Ground
		37	38	Signal Ground DATA READY ACK IN
		39	4Ø	Signal Ground Signal Ground

Table 5-1. UPIC Connector Pin Assignments

Notice that a driver program has a considerable number of signals available for use. It can simultaneously send up to sixteen bits of information to a parallel device through output ports A and B, or receive up to eight bits of data at a time through input port B. Furthermore, the driver can regulate the timing of transfers to and from the card with the four lines ACKNOWLEDGE INPUT, STROBE OUTPUT, DATA READY OUTPUT, and DATA READY ACKNOWLEDGE INPUT (pins 12, 18, 33 and 38, respectively, of the 40-pin connection).

The Parallel Driver's Dual Handshake

The Parallel Driver has two pairs of handshaking lines: one pair for data transfers to the device, and another pair for transfers from the device. The section of Chapter 2 titled Why It's Called a Handshake explains this transfer method.

A convenient and practical way to illustrate the Parallel Driver's dual handshake is in terms of the Apple III to Apple III connection. Table 5-2 shows the pin assignments for connection of two Apple IIIs that both have UPICs installed in them. Notice that the connections are symmetrical: Apple A pin 1 is connected to Apple B pin 13; Apple B pin 1 is connected to Apple A pin 13, and so on.

There are two separate handshakes for parallel I/O transfers: one for output (using STROBE OUTPUT and ACKNOWLEDGE INPUT), and one for input (using STROBE INPUT and ACKNOWLEDGE OUTPUT). All handshake and data line names indicate in what direction the data flows in the transfer (A to B; B to A). The handshake line names also indicate the direction of the signal with respect to Apple A (OUTPUT or INPUT).

In the descriptions that follow, all line and pin numbers refer to Apple A, and numbers in parentheses refer to points on the corresponding timing diagrams. Time intervals in the two diagrams are not to scale.

Apple A Pin Number	Apple B Pin Number	Function
1	13	Data A to B Bit Ø
2	14	Data A to B Bit 1
3	15	Data A to B Bit 2
4	16	Data A to B Bit 3
5	17	Data A to B Bit 4
6	19	Data A to B Bit 5
7	28	Data A to B Bit 6
8	29	Data A to B Bit 7
9		Unused
1Ø		Unused
11	11	Signal Ground
12	33	A to B ACKNOWLEDGE INPUT
13	1	Data B to A Bit Ø
14	2	Data B to A Bit 1
15	3	Data B to A Bit 2
16	4	Data B to A Bit 3
17	5	Data B to A Bit 4
18		Unused
19	6	Data B to A Bit 5
2Ø	38	A to B STROBE OUTPUT
21		Unused
22		Unused
23		Unused
24		Unused
25		Unused
26		Unused
27		Unused
28	7	Data B to A Bit 6
29	8	Data B to A Bit 7
3Ø	3Ø	Signal Ground
31		Unused
32		Unused
33	12	B to A ACKNOWLEDGE OUTPUT
34	34	Signal Ground
35	35	Signal Ground
36	36	Signal Ground
37	37	Signal Ground
38	2Ø	B to A STROBE INPUT
39	39	Signal Ground
4Ø	4Ø	Signal Ground

Table 5-2. Apple III to Apple III Pin Connections

The Output Handshake

The output handshake proceeds like this (Figure 5-2):

1. When Apple B is ready to accept data, it sets A to B ACKNOWLEDGE INPUT high (1).

2. This sets into motion the output sequence. After waiting a minimum of 44 microseconds, the Parallel Driver places data on the Data A to B lines (2), and 13 microseconds later signals that the data is there with an 8 microsecond pulse on A to B STROBE OUTPUT (3-4).

3. When Apple B detects valid data, it reads the data. Within 12 microseconds after it receives A to B STROBE OUTPUT, Apple A sets A to B ACKNOWLEDGE INPUT low (5). This pulse should stay low for at least 5Ø nanoseconds (6) if the receiver will not immediately be ready for the next byte.

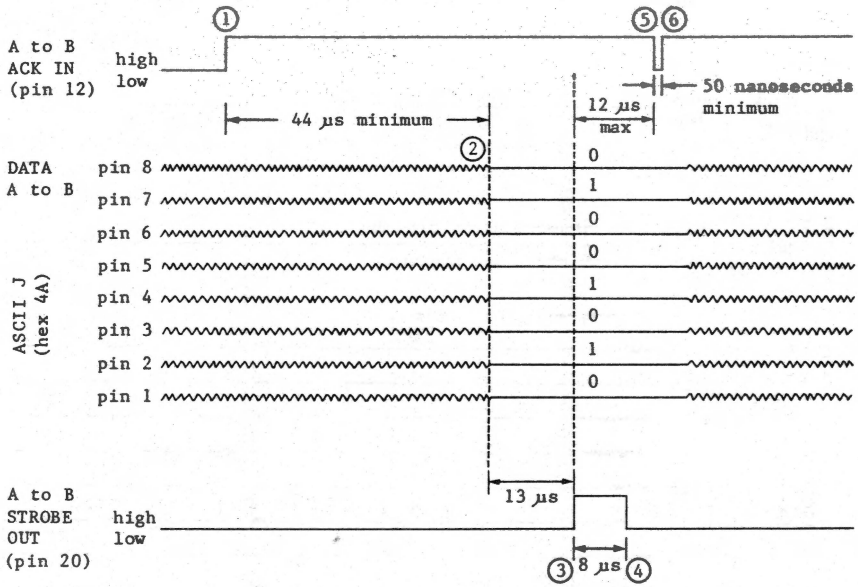


Figure 5-2. Parallel Output Handshake

The Input Handshake

The procedure for data input is similar (Figure 5-3):

1. As soon as Apple A is ready to read data, the Parallel Driver sets B to A ACKNOWLEDGE OUTPUT high (7).

2. When Apple B detects that B to A ACKNOWLEDGE OUTPUT is high, it waits at least 1 microsecond before sending data to the input port (8). Apple B indicates to Apple A that there is data on the bus by setting the B to A STROBE INPUT line high for at least 5Ø nanoseconds (9). The data must remain valid for at least 1 millisecond or until Apple A sets B to A ACKNOWLEDGE OUTPUT high.

3. When it detects B to A STROBE INPUT, Apple A immediately sets B to A ACKNOWLEDGE OUTPUT low (1Ø). It spends at least 43 microseconds processing the character before it sets B to A ACKNOWLEDGE OUTPUT high to signal that it is ready for the next transfer (11).

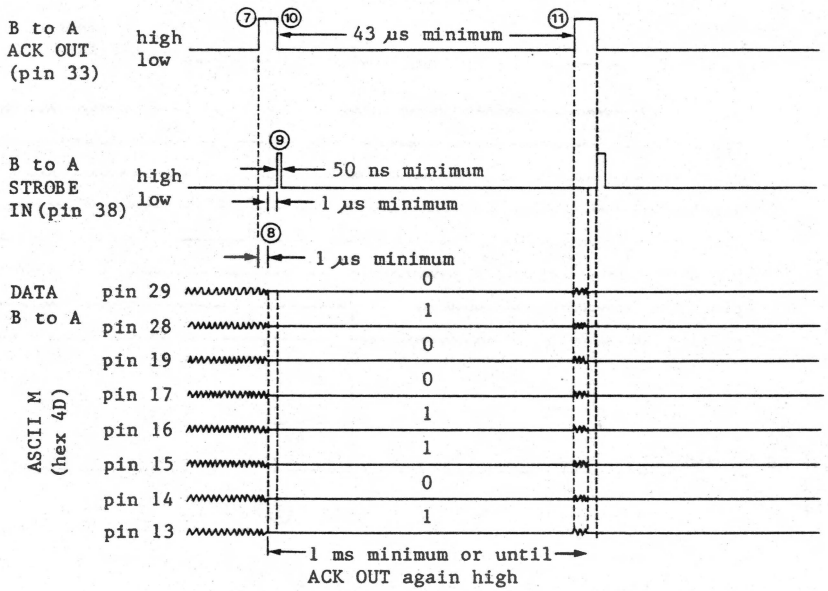


Figure 5-3. Parallel Input Handshake

The Configuration Block

The Device Configuration Block (DCB) for the Parallel Driver has only three parameters:

- LFREM removes <LF> whenever it immediately follows <CR> (DCB position Ø)
- IMREAD determines whether or not the driver should wait for enough characters or simply read what is currently in the input buffer and quit (DCB position 1)
- CTRLWRD sets the polarity of the handshaking signals (DCB position 2)

When two Apple IIIs with UPICs are connected together, set all three parameters to ØØ (their default setting) using the System Configuration Program. The sections that follow describe how to determine the settings of the Parallel Driver parameters for other applications.

LFREM (Remove <LF>)

Pascal and BASIC expect each input statement to end with a <CR> but no <LF>. Both languages mistakenly interpret <LF> after <CR> as the first character of the next command. Set LFREM to 4Ø to have .PARALLEL automatically remove any <LF> that immediately follows a <CR>.

If the Apple III is to transmit normal 8-bit data, set LFREM to ØØ.

IMREAD (Immediate Read)

If you set IMREAD to 4Ø, the Parallel Driver reads whatever characters are currently in the input buffer, (however few) and then returns control immediately to the program that made the read request. If you set IMREAD to ØØ, this does not occur.

No matter which way IMREAD is set, two other conditions cause data reading to stop:

- the requested number of characters have been transferred
- a byte called IS_NEWLINE contains FF and the driver receives the same character as the one stored in a field called NEWLINE

Control request #2 (described in Appendix B) determines the contents of both IS_NEWLINE and NEWLINE.

CTRLWRD (Control Word)

CTRLWRD sets the polarities of the handshaking signals the UPIC uses. If a bit is set to 1, the corresponding two signals are negative; that is, clocked on the falling edge (ØV is "high"). If a bit is set to Ø, the two signals are positive--clocked on the rising edge (5V is "high").

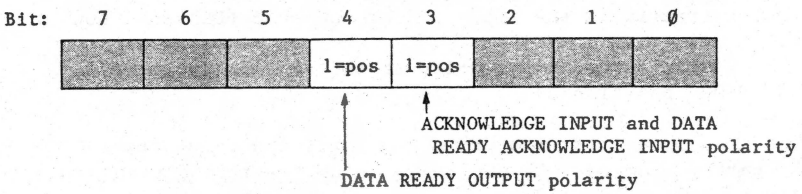


Table 5-3. CTRLWRD Bit Assignments

Parallel Driver Applications

This section explains several advanced applications of the UPIC used with the Parallel Driver. These applications make use of status and control requests (Appendix B); thus, unless you are an experienced programmer, you may have difficulty using them.

Apple III to Apple III

The .PARALLEL driver doesn't define any conventions for the start or termination of a file transfer. It merely writes characters when the receiver indicates that it is ready, and reads characters when Apple A is ready and data is valid. Thus your sending and receiving programs must signal the start and end of files.

The simplest way to transfer text files is to use the GET statement with a string variable to get one character at a time. When the receiving program reads a string of length zero, a carriage return has been read and it should write a carriage return to the destination file. The sending program should send a <CTRL-C> as an end-of-file character. <CTRL-C> is read by

the GET statement as a valid character, and it cannot normally be in a text file. For this type of transfer, set both IMREAD and LFREM to 4Ø.

Data files are more difficult to transfer since the data stream can contain any of the control characters. In this case you should probably send the length of the block of data in the first bytes of the transfer. For this type of transfer, set both LFREM and IMREAD to ØØ.

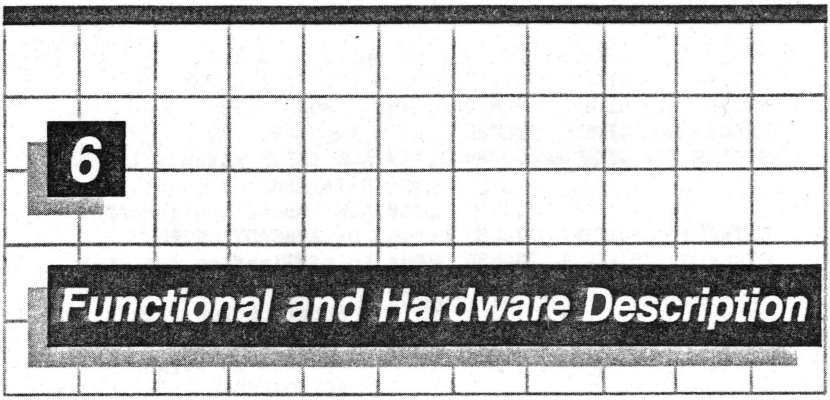
Terminal Emulator

If you use an Apple III as a terminal for another Apple III, you can turn IMREAD and LFREM on, and use the following Business BASIC program to read characters entered at the other Apple's keyboard and display them on the screen:

```
 1Ø  OPEN#1,".PARALLEL"              :REM  TURN ON CARD
 2Ø  ON KBD GOTO 1ØØ                 :REM  GOTO 1ØØ IF KEY PRESSED
 3Ø  GET#1 A$:PRINT A$;:GOTO 3Ø      :REM  READ CHARACTERS
1ØØ  PRINT#1;CHR$(KBD);              :REM  SEND TYPED CHARS
11Ø  GOTO 2Ø
```

Output Port

You can convert the Apple III UPIC into a 16 bit output port (using Output Ports A and B) without too much difficulty. The advantage of a port is that it holds a character stable on the output lines until the next character is sent to it. Output Port A is at address CØ8Ø + sØ; Output Port B is at address CØ81 + sØ (in both cases, s is the slot number). Refer to the section of Chapter 6 called Summary of Hardware Addresses for further details.



This chapter presents the Apple III UPIC's circuits in order of function: first the control circuitry, then the I/O circuitry, and finally the handshaking logic. Refer to the schematic in Appendix C as you read the descriptions. References to integrated circuits are by their card coordinates. These coordinates are written next to the part numbers for the chips on the schematic. Appendix D gives the pin assignments of the peripheral connectors on the main board inside the Apple III computer. All the signals that begin on the left side of the schematic are taken from pins in the peripheral connector slot where you installed the UPIC.

In all the descriptions that follow, s denotes the slot number.

Addressing Conventions

There are certain addresses that you can write to or read from to control the operation of the UPIC. These hardware addresses are in the range

C0B0 + s0 to C0B7 + s0

For example, if you installed the UPIC in slot 3, you should write to the addresses from C0B0 through C0B7.

To make this section easier to read, hardware addresses are given as if they were declared in an assembly language program written for the TLA (Pascal) assembler, the same one used for the two UPIC drivers. The following assembler directives assign a number to each of the constants for reference purposes.

```
PORTA     .EQU    ØCØ8Ø  ;Output Port A latch
PORTB     .EQU    ØCØ81  ;Output PORT B latch
STROBE    .EQU    ØCØ82  ;Send STROBE OUTPUT
READB     .EQU    ØCØ83  ;Read Input Port B
STATUS    .EQU    ØCØ84  ;DL7 ACK INPUT status, 1 = high
                         ;DL6 DATA READY status, 1 = high
                         ;DLØ ACK input, 1=high code
DATARDY   .EQU    ØCØ85  ;Send DATA READY OUTPUT
CTRLREG   .EQU    ØCØ86  ;Set initialization register
                         ;Bit#  Function
                         ;  Ø   Subtract 2us from strobe,1=on
                         ;  1   Add 4us to strobe width,1=on
                         ;  2   Add 8us to strobe width,1=on
                         ;  3   ACKNOWLEDGE polarity,1=positive
                         ;  4   DATA READY OUTPUT polarity,
                         ;        1=positive
                         ;  6   1=enable DATA READY ACKNOWLEDGE
                         ;        Interrupt Request; Ø=disable
                         ;  7   1=enable ACKNOWLEDGE INPUT
                         ;        Interrupt Request; Ø=disable
                         ;NOTE: Register initialized to zero
CLEAR     .EQU    ØCØ87  ;Clear ACKNOWLEDGE, DATA READY OUTPUT
```

Thus, to read the value of Port B into the Accumulator, just do the 65Ø2 assembly language operations

```
LDY       DEVOFF        ;Put slot # offset into Register Y
LDA       READB,Y       ;Read port B into the Accumulator
```

The second LDA operation adds the value in the Y register (the slot # offset) to the value in READB (the base address) and reads data from that address. But we are getting ahead of ourselves. Let's take a look at the hardware.

Address Decoding

Operations that the UPIC can perform are initiated by read or write operations to certain hardware addresses. IC 4C (a 74LS138) is a 3-to-8 decoder that selects one of its outputs depending on the state of the address lines. When an address between PORTA (CØ8Ø + sØ) and CLEAR (CØ87 + sØ) is on the address lines, the decoder output that corresponds to the value of the address lines A2, A1, AØ goes low. The rest of the outputs remain high. Each of the eight output lines of the

decoder goes to a different part of the circuitry, turning on various functions. The signals are defined as follows:

Address Lines A2	A1	A0	Selected by Address	Name	Circuitry Selected
0	0	0	C080	PORTA	Output Port A latch
0	0	1	C081	PORTB	Output Port B latch
0	1	0	C082	STROBE	Initiate STROBE OUTPUT
0	1	1	C083	READB	Read Input Port B
1	0	0	C084	STATUS	Read status bits
1	0	1	C085	DATARDY	Set DATA READY OUTPUT
1	1	0	C086	CTRLREG	Set control register
1	1	1	C087	CLEAR	Clear DATA READY OUTPUT and ACKNOWLEDGE INPUT

For example, the circuitry to Output Port B can be selected by the following code.

```
LDY     DEVOFF          ;Put slot # offset into Register Y
STA     PORTB,Y         ;Send contents of Accum. to Port B
```

A more complete summary of the effects of writing to the various addresses on the UPIC is presented at the end of the chapter.

Control Register

IC 5B (a 74LS273) is an 8-bit latch that stores the current status of the UPIC, such as the strobe pulse duration, interrupt mode status, and handshaking polarity.

Data is gated from the data bus into the status latch whenever CTRLREG (C086 + s0) is written to. The contents of the latch are set to zero whenever the $\overline{\text{IORES}}$ line goes low. This occurs when the power is turned on, or when a <CTRL-RESET> is done. The meaning of the information stored in the latch is as follows:

Bit #	Function
Ø	Subtract 2 µsec from strobe width, 1 = on
1	Add 4 µsec to strobe width, Ø=none added, 1 = on
2	Add 8 µsec to strobe width, Ø=none added, 1 = on
3	Set ACKNOWLEDGE and DATA READY ACKNOWLEDGE INPUT polarities; 1 = positive
4	Set DATA READY OUTPUT polarity; 1=positive
5	Set STROBE polarity, 1=positive
6	1=enable DATA READY ACKNOWLEDGE Interrupt Request
7	1=enable ACKNOWLEDGE Interrupt Request

The parallel driver uses this register in two ways. Bits Ø through 5 are the parameters set by the variable CTRLWRD from the Device Configuration Block. The drivers set these when the UPIC is opened. For example

```
LDY         DEVOFF          ;Put the slot # offset into Y
LDA         CTRLWRD         ;Put UPIC's parameters into Accum.
STA         CTRLREG,Y       ;Save the parameters in latch
```

will set the value of the control register to the UPIC parameters that you specified in the DCB.

The driver uses bits 6 and 7 to maintain the current status of the UPIC. Bit 6, the Enable DATA READY ACKNOWLEDGE bit, enables or disables interrupts for read requests. Bit 7, the Interrupt Request bit, enables or disables interrupts for write requests. You can disable interrupts for read requests as follows:

```
LDY         DEVOFF          ;Put the slot # offset into Y
LDA         CTRLWRD         ;Get current status of the UPIC
AND         #ØBF            ;Turn off bit 6, disable reads
STA         CTRLWRD         ;Update current status variable
STA         CTRLREG,Y       ;Save new status in status latch
```

The current status of the control register must always be maintained in software (CTRLWRD) since there is no way to read the contents of CTRLREG.

Input Port B

IC 6B (a 74LS244) is an 8-bit buffer that is used to transfer signals from the peripheral device to the Apple III. Data is driven from the UPIC's input lines (lines 29,28,19,17-13) onto

the Apple III's data lines (slot pins 42-49) whenever READB (CØ83 + sØ) is addressed.

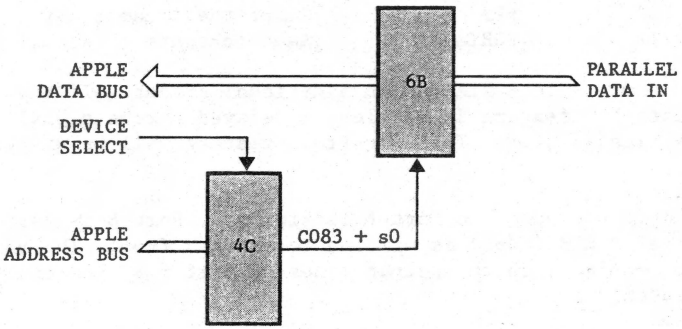


Figure 6-1. Input Port Block Diagram

Output Ports A and B

ICs 6A and 6C are 8-bit latches (74LS374) that are used to transfer data from the Apple III's data lines to the external device. Each of them latches, or stores, the data that is on the data lines when it sees the signal on its pin 11 go from a logic low to a logic high.

Data is latched into Port A whenever the address PORTA (CØ8Ø + sØ) is detected, and when the Read/$\overline{\text{Write}}$ signal (pin 18 of the peripheral connector slot) is low; that is, data is latched into Port A whenever PORTA is written to. This operation is performed by the NOR gate 2A (a 74LSØ2) which is followed by an inverter, chip 4B (a 74LSØ4). The resulting operation is PORTA AND R/$\overline{\text{W}}$. It goes low when a write to PORTA is performed, and goes high (latching the data) when the R/$\overline{\text{W}}$ line goes high again.

Port A should be used whenever additional strobe lines are needed. Use of Port B lines for software-controlled strobes will result in multiple strobing because of the address "lookahead" feature of the microprocessor.

The value ØØ can be placed on the output lines of Port A by the following instructions:

```
LDY         DEVOFF          ;Put slot # offset into Register Y
LDA         #ØØ             ;Clear the Accumulator
STA         PORTA,Y         ;Send contents of Accum. to Port A
```

Doing a write to PORTA has an additional effect. If the "autostrobe" feature is enabled, a delayed strobe signal is automatically sent. The autostrobe feature will be discussed later.

The logic connected to Port B is simpler. Port B is latched whenever PORTB (CØ81) is written to or read from. The value $FF could be placed on the output lines of Port B by the following instructions:

```
LDY         DEVOFF          ;Put slot # offset into Register Y
LDA         #ØFF            ;Put an $FF into the Accumulator
STA         PORTB,Y         ;Send contents of Accum. to Port B
```

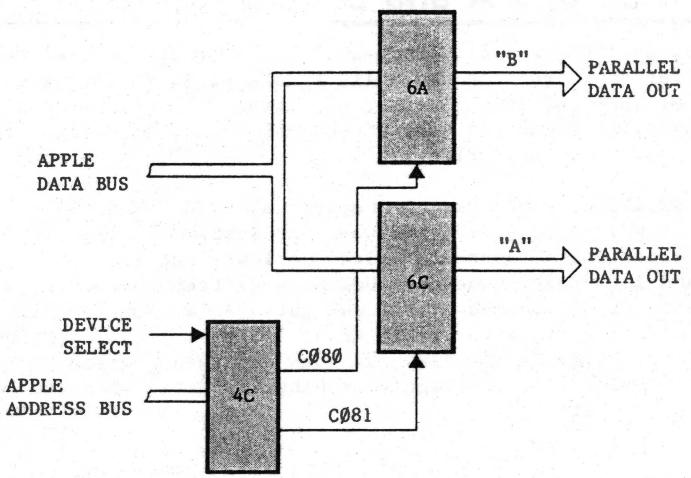


Figure 6-2. Output Port Block Diagram

Read Only Memory (ROM)

IC 5C is a 512x8 ROM containing the software the UPIC uses in Apple II Emulation mode. The code in the ROM is divided into two portions. The low 256 bytes is a printer driver for printers that have automatic linefeed (for example, Centronics). The high 256 bytes contain a printer driver that adds linefeeds after carriage returns. These drivers are identical to those on the Apple II Parallel Printer and Centronics Printer cards.

When a PR#s is done from Apple II Emulation mode, address CsØØ is automatically set as the address of the character output routine. The code in the ROM is being addressed whenever the I/O SELECT line is low; that is, when the address on the address lines is between CsØØ and CsFF. (Recall that s is the slot number.) The lower half of the code is selected when the PRINTER L.F. switch (pin 19 of the ROM) is set to AUTO (the open position). The upper half is selected when the switch is closed (that is, in the NORM position).

Through a technique called address line remapping, the state of the ACKNOWLEDGE INPUT signal is monitored. This allows different code to be executed, depending on the state of the ACKNOWLEDGE INPUT signal.

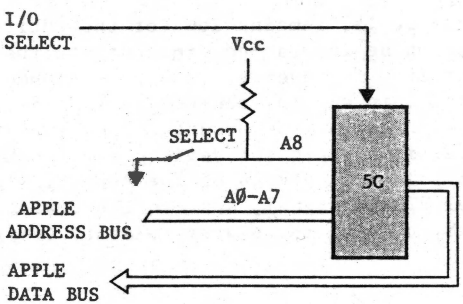


Figure 6-3. Read Only Memory

Strobe Circuitry

IC 3B (a 74LS161) is a presettable counter used to generate strobe pulses from 1 ±Ø.5 to 15 ±Ø.5 microseconds in duration. It is clocked from the system clock, C1M, which has a period of about Ø.98 microseconds.

The maximum count is set by pins 5, 6 and 9 of the 74LS273 control register, as explained in a prior section of this chapter. The lowest count bit is wired low, giving a minimum strobe width of 1 microsecond.

The count cycle is started by a high to low transition at pin 9 of the counter. This transition can be caused by two conditions:

1. A write operation to STROBE
2. A write operation to PORTA when autostrobe is on

The autostrobe feature is controlled by IC 1B, a 74LS74 flip-flop. Autostrobe is turned on when address space Csxx is accessed. It can be turned off by a read from or a write to CLEAR (CØ87 + sØ). The autostrobe feature is primarily intended to provide automatic strobes for Apple II Emulation mode; however, it can be used to help write more compact code for other applications.

The output line, pin 15, remains low for the entire count cycle; when the maximum value is reached, the output line goes high. Since the output line is inverted, and then connected directly to the enable pin (pin 4), the counter is turned off, and the output remains high until a new count cycle is started. Before the output signal is sent to the device, it is EXCLUSIVE-ORed with the STROBE polarity, pin 16 of the control register. This causes the final STROBE signal, sent to the device through connector pin 18, to have the proper polarity assigned to it.

Handshaking

IC 3A (a 74LS74) is a D-type flip-flop that is used to handle much of the handshaking. Both sides of the IC are cleared by a write to the address CLEAR (CØ87 + sØ) or by a low signal on $\overline{\text{IORES}}$ (pin 31). The DATA READY OUTPUT (pin 33) is generated when DATARDY (CØ85 + sØ) is addressed. The peripheral device responds by placing a signal on the DATA READY ACKNOWLEDGE line (pin 38), which clocks the flip-flop, removing the DATA READY

signal, and setting the D6 data line to a logic high. This response can be detected by reading from STATUS (CØ84 + sØ), and testing bit 6. This sequence is generally used for input handshaking. A simple one-character read could be performed as follows:

```
                STA     DATARDY,Y       ;Set DATA READY OUTPUT
                                        ;Clear D.R. ACKNOWLEDGE IN
GET_STATUS      LDA     STATUS,Y        ;Get status bits
                ASL     A               ;Move bit 6 to bit 7
                BPL     GET_STATUS      ;No ACKNOWLEDGE; try again
                LDA     READB,Y         ;ACK sent; read byte into
                                         Accum.
```

Output handshaking is generally handled with the STROBE and ACKNOWLEDGE lines (pins 18 and 12). A write to STROBE (CØ82 + sØ) sends the strobe signal telling the peripheral that valid data is on the data lines. A STROBE signal automatically clears the ACKNOWLEDGE bit of the flip-flop. Once the peripheral has read the data, it responds with the ACKNOWLEDGE (ACK) signal telling the computer that it has received the data. The acknowledge signal is placed on data line DIØ, where it can be detected by doing a read from STATUS (CØ84 + sØ) and testing bit Ø. A single-byte write operation can be done through Port A as follows.

```
                STA     PORTB,Y         ;Place character in Port A
                STA     STROBE,Y        ;Send STROBE OUTPUT
                                        ;Clear ACKNOWLEDGE INPUT
GET_ACK         LDA     STATUS,Y        ;Read status bits
                LSR     A               ;Put ACK into Carry bit
                BCC     GET_ACK         ;ACK not set; try again
```

The polarity of the handshaking signals (STROBE OUTPUT, ACKNOWLEDGE INPUT, DATA READY OUTPUT, and DATA READY ACKNOWLEDGE INPUT), is inverted by EXCLUSIVE-OR gate IC 4A. The state of the polarity is determined by the values present in bits 3, 4, and 5 of the Control Register.

Connector Filters

All of the signals on the peripheral connector are filtered to reduce interference by the computer with other electronics equipment. This filtering is done by routing each of the input and output lines through a ferrite core inductor and capacitor (LC) filter.

Summary of Hardware Addresses

PORTA C0800 + s0

1. A write to PORTA will latch whatever is on the data lines into IC 6C. This data remains on the 40 pin connector lines 20-27 until PORTA is written to again.

2. If autostrobe is enabled, a write to PORTA will also send a STROBE signal.

PORTB C0810 + s0

1. A read or a write to PORTB will latch whatever is on the data lines into IC 6A. This data remains on the 40 pin connector lines 1-8 until PORTB is written to again. For a read operation, the data on the lines is undefined.

STROBE C0820 + s0

1. A write to STROBE will generate the STROBE signal with duration and polarity as set in the control register.

2. When the STROBE signal is generated, it causes the ACKNOWLEDGE status to be cleared (see STATUS).

READB C0830 + s0

1. A read of READB allows the computer to read Port B (IC 6B). Input Port B is 40-pin connector lines 13-17, 19, and 28-29.

STATUS C0840 + s0

1. A read from STATUS will return the values of three of the UPIC's signals:
   DL7 = The ACKNOWLEDGE (ACK) status:
   Turned on by ACKNOWLEDGE INPUT.
   Turned off by STROBE signal or reset.
   DL6 = The DATA READY status:
   Turned on by a read or write to DATARDY.
   Turned off by a pulse on the DATA READY ACKNOWLEDGE INPUT line.
   DL0 = The ACKNOWLEDGE signal:
   Mirrors the ACKNOWLEDGE INPUT, but always is positive true logic (uses polarity).

2. If the interrupts are enabled (by setting bits 7 and 6 of the control register), then ACK status (DL7) and/or DATA READY status (DL6) will result in an IRQ signal.

DATARDY C085 + sØ

1. A write or read to DATARDY sets the Data Ready flip-flop.

2. DL6, the read interrupt bit, is turned off.

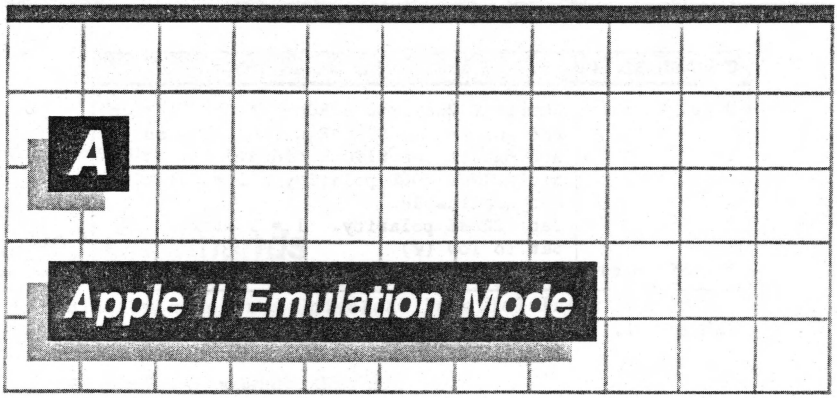
CTRLREG C086 + sØ

1. A write to CTRLREG causes the data on the data lines to be latched into IC 5B. This is the control register for the UPIC. See its description above.

CLEAR C087 + sØ

1. A read or write to CLEAR clears the handshaking flip-flop.

2. A read or write also disables the autostrobe feature.



A feature of the UPIC used with the Printer Driver is the full support of Apple II Emulation mode. If you have any Apple II software that uses your printer, and if your printer is one of those listed in the preface, simply plug your 2Ø pin printer connector into the UPIC (as explained in Chapter 2), put the Apple III into Emulation mode, and let it run. How to use the Apple III's Apple II Emulator is explained in the Apple III Owner's Guide.

In the explanations that follow, s is the number of the slot in which you have installed the UPIC; <CTRL-keyname> means "hold down the CTRL key while pressing the key called keyname"; <RETURN> denotes the RETURN key, etc.

Setting the Strobe

Turning the Apple III on or pressing <CTRL-RESET> automatically sets up the UPIC for the most commonly used printers; that is, the UPIC issues negative STROBE pulses that are 3µs long, and it expects to receive negative ACKNOWLEDGE pulses. All of the commonly used printers listed in the preface expect the same setting (CTRLWRD = ØØ).

If your printer requires a STROBE pulse of a different length, or if it uses STROBE or ACKNOWLEDGE signals of positive polarity, then you must POKE the proper value into the control word (CTRLWRD) at address CØ86 + sØ on the UPIC. (CTRLWRD is simply a name for DCB parameter 3, shown in Table 2-4.) Table A-1 shows how the bits in the control word, and hence in the value you must POKE, are assigned.

CTRLWRD Bit #	Function
Ø (right bit)	Subtract 2µs from STROBE width. 1 = on
1	Add 4µs to the STROBE width. 1 = on
2	Add 8µs to the STROBE width. 1 = on
3	Set ACKNOWLEDGE polarity. 1 = positive
4	Not applicable
5	Set STROBE polarity. 1 = positive
6	Set to low (Ø)
7 (left bit)	Set to low (Ø)

Table A-1. CTRLWRD (Control Word) Bit Assignments

For example, if your printer requires positive handshaking, and a strobe length of 5µs, you want to set the control word bits to ØØ1Ø1Ø11. This corresponds to a decimal value of 43. If your UPIC is in slot 1, then the desired address is CØ86 + 1Ø = CØ96, which is 493Ø2 in decimal. So do this:

```
POKE 493Ø2,43   :REM  INITIALIZE CONTROL REGISTER
```

Turning the UPIC On and Off

The following commands all pertain to Apple II Emulation mode only. They have no effect when the Apple III is not in Emulation mode.

You can turn on the printer from the keyboard with the command

s<CTRL-P><RETURN>

(UPIC is in slot s), and turn it off with the command

Ø<CTRL-P><RETURN>

From BASIC the command

PR#s <RETURN>

turns on the UPIC. All subsequent output will go to the printer as well as to the screen.

When you use the command

PR#Ø <RETURN>

all subsequent output will go to the screen only.

UPIC Commands

In Emulation mode, the Universal Parallel Interface Card commands take the form

<CTRL-I> <command> <RETURN>

In the case of the Change Line Width (N) command, there is also a number before the command indicating the new line width to use.

Each command must be preceded by <CTRL-I> (or another control character; see next paragraph) and followed by <RETURN>.

You can change <CTRL-I> to any other control character from <CTRL-A> through <CTRL-Z> by simply typing <CTRL-I> followed by the new control character; typing the two in reverse order changes it back. For example, typing <CTRL-I><CTRL-Q> changes the control character to <CTRL-Q>; typing <CTRL-Q><CTRL-I> changes it back. This is useful if you want to list on the printer a program that contains <CTRL-I>.

You can type in these commands at the keyboard or embed them in programs (for example, in a BASIC PRINT statement).

Change Line Width (line-width N)

The N command changes the number of characters printed per line from its current value to the one specified by line-width. Legal values of line-width are from 4Ø through 255. The default line width is 4Ø. This command also turns off the video screen.

For example, to print data on an 8Ø-column printer, type in

<CTRL-I>8ØN<RETURN>

Restore Video Display (I)

The I command restores the video display as the output device, and changes the line width back to 40.

Toggle the Linefeed Switch (K)

If the Printer L.F. switch on the UPIC is set to NORM, the K command causes the UPIC to suppress linefeed characters normally sent to the printer after each carriage return character. Type the command again, and the UPIC will resume sending linefeeds.

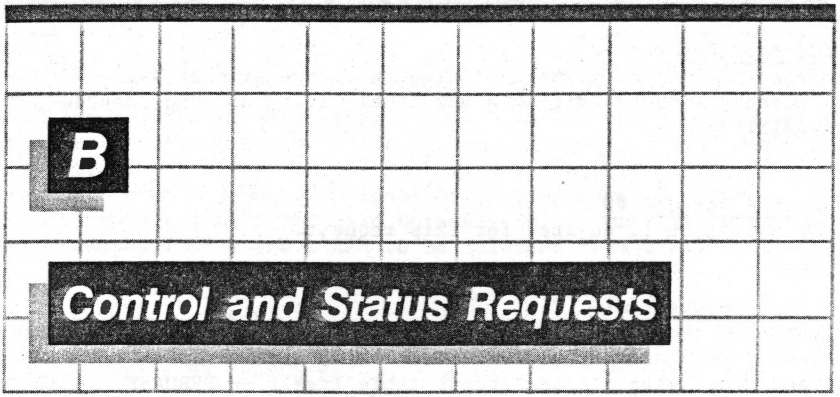
If the Printer L.F. switch is in the AUTO position, this command has no effect.

Some printers don't print out a line until they receive the linefeed character. Turning off linefeed may cause some printers to stop printing altogether.

Here is a BASIC program that sets the printer width to 132, prints a line on the printer only, and then prints another line on both the screen and the printer. Notice that the line width is reset to 40 when output to the screen is turned back on.

```
10   PR#1 :REM  TURN ON CARD IN SLOT 1
20   I$ = ""             :REM  SET I$ TO CONTROL-I
30   PRINT I$;"132 N"    :REM  SET LINE WIDTH TO 132
40                        REM  ALSO TURNS OFF SCREEN
50   PRINT "THIS LINE PRINTS ON THE PRINTER ONLY"
60   PRINT I$;"I"        :REM  RESTORE VIDEO, 40 COLUMNS
70   PRINT "THIS LINE PRINTS ON THE PRINTER AND SCREEN"
80   PR#0                :REM  TURN THE CARD OFF
90   END
```

For more information about use of the Apple III UPIC in Emulation mode, please refer to the Apple II Parallel Printer Card Manual or Centronics Printer Card Manual.



A number of SOS control and status requests are available for controlling the operation of a device and checking its condition. The next two sections explain their significance as used by the Printer Driver and the Parallel Driver. The section after that discusses how to make your own Pascal requests. The final sections list the demonstration programs.

Printer Driver Control and Status Requests

Using these control and status requests, you can write programs that constantly monitor the status of the printer. If the printer is turned on, has a good ribbon, and has paper, then your program can safely send information to it; otherwise the program should send a diagnostic message to the screen.

Control Request #Ø
This request resets the Printer Driver, clearing its output buffer of all information currently waiting to be printed. This request also turns off interrupts on the UPIC; thus if the printer sends an ACKNOWLEDGE signal after the Apple sends Control Request Ø, the UPIC will not respond. (Normally, when the UPIC receives the acknowledge signal, the Printer Driver responds by sending characters from the buffer to the printer.)

Control Request #1
This SOS request sends six bytes to the printer: first a byte containing the number of bytes that follow (Ø5); then a five byte status table -- ERRMASK, ERRSTAT, AUTOLF, CTRLWRD, and TIMEOUT, in that order. All five of these bytes are discussed in Chapter 4.

Control Request #2
This SOS request normally indicates to the printer what character should trigger a new line. It is not implemented in .PRINTER.

Status Request #Ø
No operation is defined for this request.

Status Request #1
This SOS request causes the printer to send back six bytes: the first byte contains the byte count (Ø5) of the status table; the remaining bytes are the status table itself -- ERRMASK, ERRSTAT, AUTOLF, CTRLWRD, and TIMEOUT, in that order. These five bytes are explained in detail in Chapter 4.

Status Request #2
This SOS request normally causes the printer to send back the new line indicator character. .PRINTER places $ØØ in the location your call's pointer selects.

Status Request #3
This request causes the printer to send back five bytes containing its (error) status and buffer sizes.

Byte:	5	4	3	2	1
	high	low	high	low	ERRSTAT
	Number of characters in output buffer		Size of the output buffer		See Table 4-4

Parallel Driver Control and Status Requests

You can use the control and status requests described below to monitor the state of the device connected to your Apple III via the .PARALLEL driver. Later sections explain how to use these requests with Pascal and BASIC programs.

Control Request #Ø
This request resets the Parallel Driver: it clears the output and input buffers, destroying any information not yet read or written. It also turns off the interrupts on the card. As a result, if you issue Control Request #Ø, the UPIC will not respond to an ACKNOWLEDGE signal by sending more characters to the buffer (which it would normally do). Control Request #Ø also causes IMREAD and LFREM to be reloaded from the DCB.

Control Request #1
This SOS request sends the parallel device four bytes: the first byte contains Ø3 (the byte count of what follows); the remaining three bytes contain the new values for LFREM, IMREAD, and CTRLWRD. All three of these bytes are explained in Chapter 5.

These values override those in the DCB until a later Control Request #Ø resets the driver or a BASIC command opens the driver again.

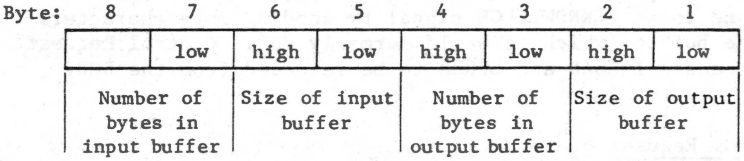
Control Request #2
This request always sends two bytes to SOS: the first byte sets the switch IS_NEWLINE to either $FF (terminate an SOS read request if it reads the same character as the one stored in NEWLINE) or $ØØ (ignore NEWLINE). The second byte places the character in NEWLINE that will prematurely terminate an SOS read request if IS_NEWLINE is set to $FF.

Status Request #Ø
No operation is defined for this SOS request.

Status Request #1
This SOS request causes the driver to return four bytes: the first byte contains Ø3 (the number of bytes that follow); the remaining three bytes contain the current values of LFREM, IMREAD and CTRLWRD. These three bytes are explained in Chapter 5. The values returned by Status Request #1 can be changed with Control Request #1.

Status Request #2
This request has a pointer that indicates where to store the current value of IS_NEWLINE (the first byte) and NEWLINE (the second byte) that the driver is using.

Status Request #3
This request returns eight bytes describing the current state of the buffers:



Making Your Own Pascal Requests

This section explains how to make your own Pascal status and control requests for any of the functions outlined above.

In Pascal, the information needed for a status or a control call can be represented by the following record declaration:

```
STATUS = RECORD
  UNIT_NUMBER:INTEGER;
  STATUS_BYTES:PACKED ARRAY[Ø..59] OF CHAR;
  CONTROL_CODE:CONTROL_FORMAT
END;
```

for which the CONTROL_FORMAT has been declared to be

```
CONTROL_FORMAT = PACKED RECORD
  IODIR:(OUT,IN);
  CALL_TYPE:(STATUS,CONTROL);
  REQUEST_CODE: Ø..2Ø47
END;
```

A control or status request is made using the UNITSTATUS function, a standard Apple III Pascal procedure. For example

```
UNITSTATUS(STAT.UNIT_NUMBER,STAT.STATUS_BYTES,STAT.CONTROL_CODE);
```

could be used to make a status or control request from a Pascal program. STAT is a record of type STATUS, and the three

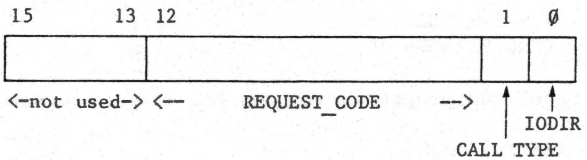
parameters referenced are as declared in the record above. Notice that the second parameter is a pointer to the 6Ø byte buffer.

For a control request the value of UNIT_NUMBER must be set to the standard Pascal unit number of the device. If your UPIC is the Pascal unit PRINTER:, then the value of UNITSTATUS is

```
STAT.UNIT_NUMBER := 6;
```

Information sent or received by the request is transferred in the 6Ø-byte buffer, STATUS_BYTES. For control requests, you must assign values to certain bytes of STATUS_BYTES, as required by that particular control request. Status requests cause the driver's status routine to place values in the buffer.

The boolean CONTROL_CODE.CALL_TYPE tells the driver whether the call is a control or a status request. The value of CONTROL_CODE.REQUEST_CODE indicates to the driver which control or status request is desired. Finally, CONTROL_CODE.IODIR indicates whether the call applies to an output or input channel for a unit. The format of the control code is as follows.



CALL_TYPE should be Ø for a status call and 1 for a control call. IODIR should be Ø if the call applies to an output channel for a unit or 1 if the call applies to an input channel for a unit.

For example, CONTROL_CODE for Control Request #2 to PRINTER: (unit #6) would be:

```
STAT.UNIT_NUMBER := 6;
STAT.CONTROL_CODE.IODIR := OUT;
STAT.CONTROL_CODE.CALL_TYPE := CONTROL;
STAT.CONTROL_CODE.REQUEST_CODE := 2;
```

This is the Parallel I/O Driver control request which is used to set the NEWLINE character. The entire call would be

```
WITH STAT DO BEGIN
  UNIT_NUMBER := 6;                    {Request to printer}
  STATUS_BYTES[Ø] := 255;              {Turn IS_NEWLINE on}
  STATUS_BYTES[1] := 33;               {set NEWLINE to "!"}
  CONTROL_CODE.IODIR := OUT;           {output channel     }
  CONTROL_CODE.CALL_TYPE := CONTROL;{Control request    }
  CONTROL_CODE.REQUEST_CODE := 2;    {Control request 2 }
  UNITSTATUS(UNIT_NUMBER,STATUS_BYTES,CONTROL_CODE);
END;  {WITH STAT DO}
```

Demonstration Software Listings

The UPIC diskette contains, besides the .PRINTER and .PARALLEL drivers, three demonstration software programs written in BASIC.

To run one of these modules (for example, DEMO.PRINTER), from the keyboard, type:

```
PREFIX$=/PARALLEL.CARD/
RUN  DEMO.PRINTER
```

To call one of the programs from another BASIC program, use:

```
1Ø PREFIX$=".dl"
2Ø INVOKE"REQUEST"
3Ø RUN"DEMO.PRINTER"
```

In the program listings that follow, some lines have been arbitrarily split into two or three shorter lines to make them fit more easily into the formatted width of the manual's pages. Each unnumbered line is the continuation of the line that precedes it.

DEMO.PRINTER Listings

```
 5  HOME
 1Ø  INVOKE"request"
 2Ø  DEVICE$=".PRINTER"
 3Ø  OPEN#1,DEVICE$
 4Ø  GOSUB 3ØØØØ
 5Ø  ON KBD GOTO 5ØØØØ
```

```
9Ø  GOSUB 4ØØØØ
95  IF chars.in>Ø GOTO 9Ø
1ØØ  PRINT#1;"This is line number "line.no"
     being sent to the printer "
11Ø  line.no=line.no+1
115  ON KBD GOTO 5ØØØØ
12Ø  GOTO 9Ø
29999  REM
3ØØØØ  REM Request Device Configuration parameters
3ØØØ1  REM
3ØØ1Ø  req.num=1
3ØØ2Ø  PERFORM status(%req.num,@string$)device$
3ØØ3Ø  errmask=ASC(MID$(string$,2,1))
3ØØ4Ø  errstat=ASC(MID$(string$,3,1))
3ØØ5Ø  autolf=ASC(MID$(string$,4,1))
3ØØ6Ø  ctrlwrd=ASC(MID$(string$,5,1))
3ØØ7Ø  timeout=ASC(MID$(string$,6,1))
3Ø199  REM
3Ø2ØØ  REM Test for printer error
3Ø2Ø1  REM
3Ø2Ø5  PRINT:PRINT"ERRMASK is $"HEX$(errmask)"
       ERRSTAT is $"HEX$(errstat)
3Ø21Ø  IF errmask=Ø GOTO 3Ø4ØØ
3Ø215  PRINT:PRINT"The following error conditions
       can be generated"
3Ø217  PRINT" by the parallel printer:":PRINT
3Ø22Ø  IF errmask>127 THEN PRINT TAB(1Ø);
       "Printer power off":errmask=errmask-128
3Ø23Ø  IF errmask>63 THEN PRINT TAB(1Ø);"Printer off line":
       errmask=errmask-64
3Ø24Ø  IF errmask>31 THEN PRINT TAB(1Ø);"Printer out of paper":
       errmask=errmask-32
3Ø25Ø  IF errmask>15 THEN PRINT TAB(1Ø);"Printer ribbon out  ":
       errmask=errmask-16
3Ø26Ø  IF errmask>7 THEN PRINT TAB(1Ø);"Printer in check    ":
       errmask=errmask-8
3Ø27Ø  IF errmask>Ø THEN PRINT TAB(1Ø);"Printer receives
       undefined error"
3Ø4ØØ  PRINT:PRINT"Auto linefeed is ";:IF autolf=64
       THEN PRINT"enabled":ELSE PRINT"disabled"
3Ø41Ø  PRINT:PRINT"CTRLWRD is $"HEX$(ctrlwrd)
3Ø5ØØ  PRINT:PRINT"The printer driver will timeout after waiting
       "timeout*11" microseconds"
3Ø6ØØ  PRINT:PRINT"Would you like to change any of these
       parameters (Y/N) ?";
3Ø61Ø  GET a$:PRINT a$
3Ø62Ø  IF a$<>"Y" AND a$<>"y" THEN RETURN
3Ø63Ø  PRINT:PRINT"What value should the ERRMASK be ( ØØ- FF) ";
```

```
3Ø64Ø  INPUT a$
3Ø65Ø  errmask=TEN(a$)
3Ø66Ø  PRINT:PRINT"What value should ERRSTAT be (ØØ-FF) ";
3Ø67Ø  INPUT a$
3Ø68Ø  errstat=TEN(a$)
3Ø69Ø  PRINT:PRINT"Should autolinefeed be enabled (Y/N) ?";
3Ø7ØØ  GET a$:PRINT a$
3Ø7Ø1  IF a$="Y" OR a$="y" THEN autolf=64:ELSE autolf=Ø
3Ø7Ø3  PRINT:PRINT"What value should CTRLWRD be (ØØ-FF) ";
3Ø7Ø5  INPUT a$
3Ø7Ø7  ctrlwrd=TEN(a$)
3Ø72Ø  PRINT:PRINT"How many microseconds should the driver
       what till timeout ";
3Ø73Ø  INPUT a
3Ø74Ø  timeout=a/11
3Ø75Ø  IF timeout>255 THEN timeout=255
3Ø76Ø  string$="5"+CHR$(errmask)+CHR$(errstat)+CHR$(autolf)
       +CHR$(ctrlwrd)+CHR$(timeout)
3Ø77Ø  PERFORM control(%req.num,@string$)device$
3Ø9ØØ  GOTO 3ØØØØ
39999  REM
4ØØØØ  REM  Request error status and buffer size
4ØØØ1  REM
4ØØ1Ø  req.num=3
4ØØ2Ø  PERFORM status(%req.num,@string$)device$
4ØØ3Ø  error.status$=MID$(string$,1,1)
4ØØ4Ø  error.status=ASC(error.status$)
4ØØ5Ø  buf.size.low$=MID$(string$,2,1)
4ØØ6Ø  buf.size.high$=MID$(string$,3,1)
4ØØ7Ø  buf.size=ASC(buf.size.low$)+ASC(buf.size.high$)*256
4ØØ8Ø  chars.in.low$=MID$(string$,4,1)
4ØØ9Ø  chars.in.high$=MID$(string$,5,1)
4Ø1ØØ  chars.in=ASC(chars.in.low$)+ASC(chars.in.high$)*256
4Ø199  REM
4Ø2ØØ  REM Test for printer error
4Ø2Ø1  REM
4Ø21Ø  IF error.status=Ø GOTO 4Ø4ØØ
4Ø22Ø  IF error.status>127 THEN PRINT"Printer power off":
       GOTO 4Ø3ØØ
4Ø23Ø  IF error.status>63 THEN PRINT"Printer off line":GOTO 4Ø3ØØ
4Ø24Ø  IF error.status>31 THEN PRINT"Printer out of paper":
       GOTO 4Ø3ØØ
4Ø25Ø  IF error.status>15 THEN PRINT"Printer ribbon out  ":
       GOTO 4Ø3ØØ
4Ø26Ø  IF error.status>7 THEN PRINT"Printer in check    ":
       GOTO 4Ø3ØØ
4Ø27Ø  PRINT"Printer in undefined error":GOTO 4Ø3ØØ
4Ø3ØØ  PRINT"Press RETURN when fixed "
```

```
4Ø31Ø  GET a$
4Ø32Ø  GOTO 4ØØØØ
4Ø4ØØ  PRINT chars.in" characters in buffer -
       Room for "buf.size-chars.in" more"
4Ø41Ø  RETURN
5ØØØØ  END
```

DEMO.PARALLEL Listings

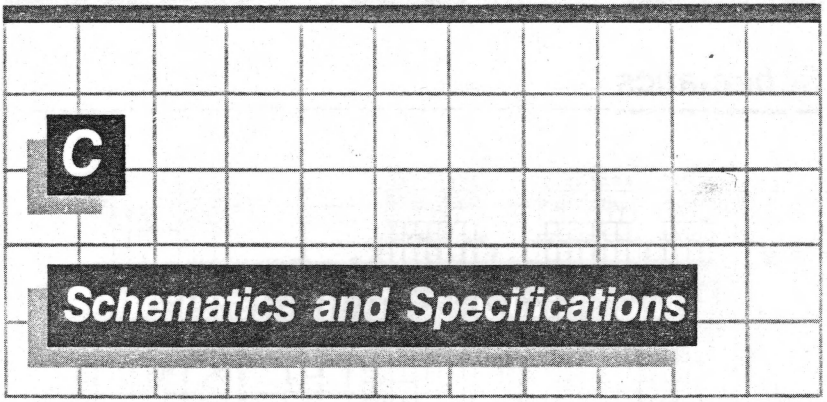
```
1Ø  INVOKE"request"
2Ø  DEVICE$=".Parallel"
3Ø  OPEN#1,DEVICE$
5Ø  HOME
1ØØ  PRINT"Do you want to remove linefeeds ":
     PRINT"that immediately follow carriage returns (Y/N) ? ";
11Ø  GET lfrem$
115  PRINT LFREM$:PRINT
12Ø  PRINT"Do you want to recieve the contents ":
     PRINT"of the input buffer immediately
     instead of after newline character (Y/N) ? ";
13Ø  GET imread$
14Ø  PRINT IMREAD$:PRINT
145  PRINT:PRINT"What value do want to set CTRLWRD (ØØ-FF) ";
147  INPUT a$:ctrlwrd$=CHR$(TEN(a$))
148  PRINT
15Ø  IF lfrem$="y" OR lfrem$="Y" THEN lfrem$=CHR$(64):
     ELSE lfrem$=CHR$(Ø)
16Ø  IF imread$="y" OR imread$="Y" THEN imread$=CHR$(64):
     ELSE imread$=CHR$(Ø)
17Ø  table.string$=CHR$(3)+lfrem$+imread$+ctrlwrd$
175  vnew.string$=CHR$(Ø)+CHR$(Ø)
18Ø  PRINT"Do you want input terminated by ":
     PRINT"something other than the character count (Y/N) ? ";
19Ø  GET is.newline$
195  PRINT IS.NEWLINE$:PRINT
2ØØ  IF is.newline$="y" OR is.newline$="Y"
     THEN PRINT"What character (Hit character) ? ";:
     GET newline$:vNEW.string$=CHR$(255)+newline$
2Ø5  PRINT
21Ø  req.num=1
22Ø  PERFORM control(%req.num,@table.string$)device$
23Ø  req.num=2
24Ø  PERFORM control(%req.num,@vNEW.string$)device$
39999  REM
4ØØØØ  REM  Request error status and buffer size
4ØØØ1  REM
```

```
40010  req.num=3
40020  PERFORM status(%req.num,@string$)device$
40050  o.buf.size.low$=MID$(string$,1,1)
40060  o.buf.size.high$=MID$(string$,2,1)
40070  o.buf.size=ASC(O.buf.size.low$)+ASC(O.buf.size.high$)*256
40080  o.chars.in.low$=MID$(string$,4,1)
40090  o.chars.in.high$=MID$(string$,3,1)
40100  o.chars.in=ASC(O.chars.in.low$)+ASC(O.chars.in.high$)*256
40150  i.buf.size.low$=MID$(string$,5,1)
40160  i.buf.size.high$=MID$(string$,6,1)
40170  i.buf.size=ASC(I.buf.size.low$)+ASC(I.buf.size.high$)*256
40180  i.chars.in.low$=MID$(string$,7,1)
40190  i.chars.in.high$=MID$(string$,8,1)
40200  i.chars.in=ASC(I.chars.in.low$)+ASC(I.chars.in.high$)*256
40250  PRINT
40300  PRINT o.chars.in" characters in output buffer -
       Room for "o.BUF.SIZE-O.CHARS.IN" characters"
40310  PRINT i.chars.in" characters in input buffer -
       Room for "i.BUF.SIZE-I.CHARS.IN" characters"
45000  REM New line
45010  req.num=2
45020  PERFORM status(%req.num,@string$)device$
45030  PRINT
45050  IF ASC(MID$(string$,1,1))>127
       THEN PRINT"Newline character active -
       Character is$"HEX$(ASC(MID$(string$,2,1)))"
       decimal "ASC(MID$(string$,2,1))"     '"MID$(string$,2,1)"'"
55000  REM Status table
55010  req.num=1
55020  PERFORM status(%req.num,@string$)device$
55030  PRINT
55050  IF ASC(MID$(string$,2,1))>63
       THEN PRINT"Line feed remove on output active":
       ELSE PRINT"Line feed remove inactive"
55055  PRINT
55060  IF ASC(MID$(string$,3,1))>63
       THEN PRINT"Immediate read on input":
       ELSE PRINT"Immediate read inactive"
55070  PRINT:PRINT"CTRLWRD is $"HEX$(ASC(MID$(string$,4,1)))
```

DEMO.DEVINFO Listings

```
5  ON ERR GOTO 90
10  INVOKE"request"
50  HOME
90  PRINT
100  INPUT"Name of device driver (Include period) ? ";device$
```

```
11Ø  IF device$="" THEN END
12Ø  GOSUB 3ØØØØ
13Ø  GOTO 9Ø
3ØØØØ  REM Device Information
3ØØ1Ø  PERFORM devinfo(@string$)device$
3ØØ2Ø  slot=ASC(MID$(string$,1,1))
3ØØ3Ø  devtype=ASC(MID$(string$,3,1))
3ØØ4Ø  subtype=ASC(MID$(string$,4,1))
3ØØ5Ø  manid=ASC(MID$(string$,8,1))
3ØØ6Ø  vernum$=RIGHT$(HEX$(ASC(MID$(string$,11,1))),2)+RIGHT$
       (HEX$(ASC(MID$(string$,1Ø,1))),2)
3ØØ7Ø  PRINT
3Ø1ØØ  IF devtype=97 AND subtype=1 THEN PRINT".CONSOLE driver"
3Ø11Ø  IF devtype=66 AND subtype=1 THEN PRINT".GRAPHICS driver"
3Ø12Ø  IF devtype=65 AND subtype=Ø THEN PRINT"parallel printer
       driver"
3Ø125  IF devtype=65 AND subtype=1 THEN PRINT".SILENTYPE driver "
3Ø13Ø  IF devtype=65 AND subtype=2 THEN PRINT"serial printer
       driver"
3Ø14Ø  IF slot=Ø THEN PRINT"Not in a slot":
       ELSE PRINT"In slot "slot
3Ø15Ø  IF manid=1 THEN PRINT"Manufactured by Apple Computer"
3Ø16Ø  PRINT"Version number "vernum$
3Ø3ØØ  RETURN
```



Specifications

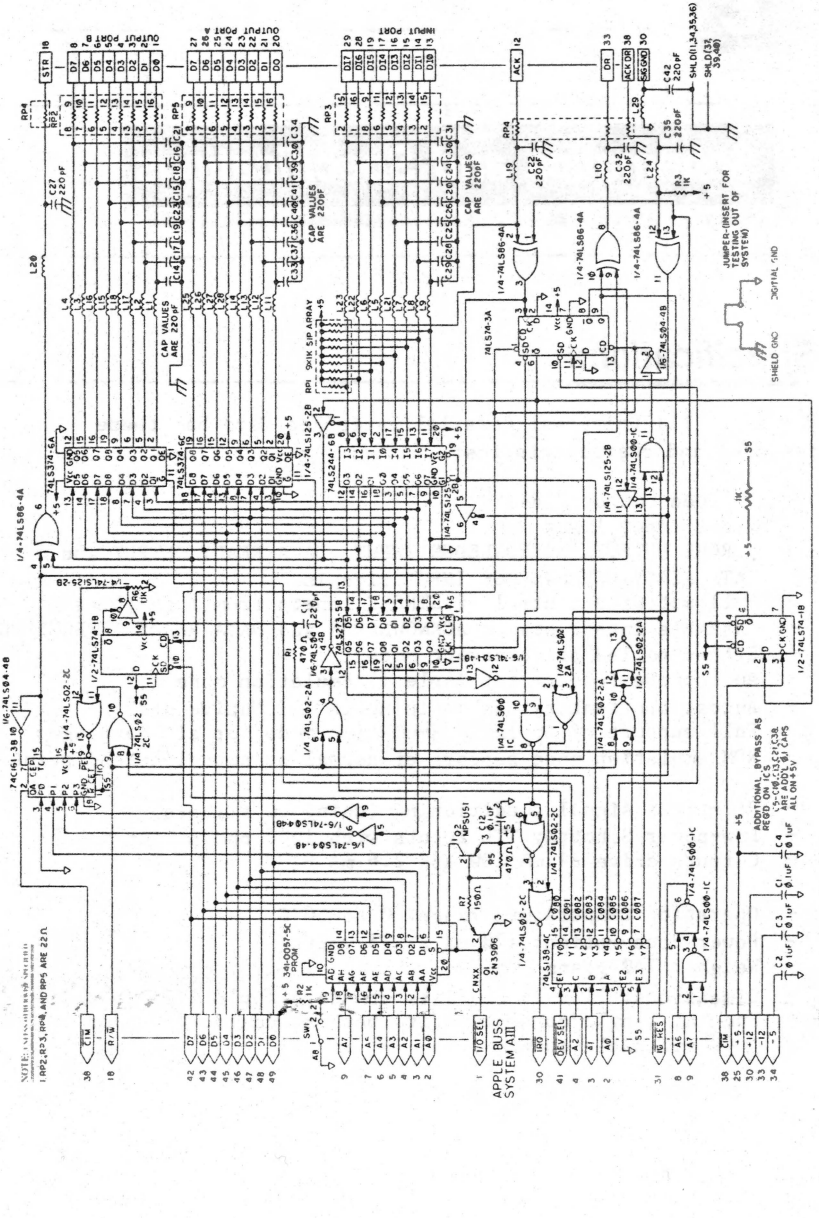
The Apple III Universal Parallel Interface Card has these features and characteristics:

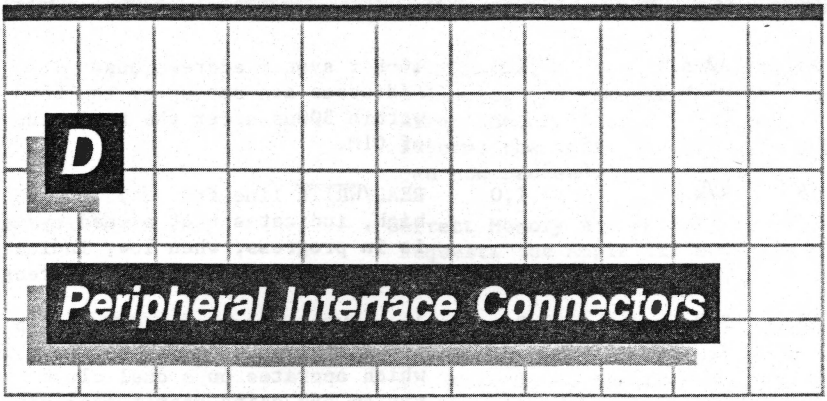
- Sixteen output lines
- Eight input lines
- STROBE OUTPUT, ACKNOWLEDGE INPUT, DATA READY OUTPUT and DATA READY ACKNOWLEDGE INPUT signals
- Full software control of STROBE length from 1 to 15 μs
- Maskable interrupts on ACKNOWLEDGE and DATA READY ACKNOWLEDGE
- Either polled or interrupt mode
- Autostrobe feature for Apple II Emulation mode
- Autostrobe feature usable in Apple III Native mode
- Full support of Apple II Emulation mode for all languages
- ROM emulation code support of normal or auto-linefeed printers

- RFI-proof signal conditioning (using LC filters)
- Low-power Schottky TTL inputs with 1ØØØ ohm pullup resistors
- Outputs drive 24 mA LOW and 2.6 mA HIGH

- Operation temperature range: Ø to 5Ø°C
- Power consumption: 17Ø mA at +5 volts
- Weight: 14Ø grams
- Dimensions: 11 x 19 cm (fits standard Apple III I/O slot)
- Special single connector fits both standard 2Ø-pin or full 4Ø-pin cable connections

Schematics





SIGNAL DESCRIPTION FOR PERIPHERAL I/O CONNECTORS

Pin #	Pin Name	In or Out**	Description
1	$\overline{\text{I/O SELECTx}}$	O	256 addresses are set aside for each peripheral connector A read or write at such an address will send Pin 1 on the selected connector low during the micro PHØ (Phase Ø; nominally 5ØØns in 1MHz mode; 25Øns in 2MHz mode).

2-17	AØ-A15	I,O	16-bit system address bus. Addresses are set up by the 65Ø2 within 3ØØns after the beginning of C1M.
18	R/$\overline{W}$	I,O	READ/$\overline{WRITE}$ line from 65Ø2. When high, indicates that a read cycle is in progress; when low, indicates that a write cycle is in progress.
19	PHØ	O	1 or 2MHz signal coincident with the operation mode of the Apple III, which operates on a dual clock system and gains operational speed by using a 2MHz clock at certain times.
2Ø	$\overline{I/O\ STROBE}$	O	Pin 2Ø on all peripheral connectors will go low during the micro PHØ of a read or write cycle to any address $C8ØØ-$CFFF.
21	RDY	I	"Ready" line to the 65Ø2. This line should change only during C1M, and when low will halt the microprocessor at the next READ cycle. This line has a 1K ohm pullup to +5V.
22	$\overline{TSADB}$	I	A low on this line from the peripheral will cause the address bus to tri-state for Direct Memory Access (DMA) applications. Has a 1K ohm pullup to +5V.
23		NA	Not used in Apple III (no daisy chaining of peripherals).
24		NA	Not used in Apple III.
25	+5V	O	Positive 5-volt supply*, 2.Ø amps total for all peripheral boards together (but note limit of 1.5 W per board).
26	GND	NA	System circuit ground. Ø volt line from power supply. Do not use for shield ground.

27	DMAOK	O	ACKNOWLEDGE signal to the peripheral following its request for the special Direct Memory Access (DMA) mode. Informs the peripheral that the DMA can now proceed.
28	$\overline{\text{DMAI}}$	I	Direct Memory Access (DMA) Interrupt. Requests the Apple III's DMA mode. Has a 1K ohm pullup to +5V.
29	$\overline{\text{IONMI}}$	I	Input/Output Non-Maskable Interrupt. The non-maskable interrupt does not go directly to the processor, so it can be masked by the system reset lock function.
3Ø	$\overline{\text{IRQx}}$	I	Interrupt request line. Each peripheral signal goes to an individual gate input and therefore can be driven by a normal TTL output.
31	$\overline{\text{IORES}}$	O	Input/Output Reset signal used to reset the peripheral devices. Pulled low by a power on or Reset in the Emulation mode or a Control-Reset.
32	$\overline{\text{INH}}$	I	Inhibit line. When a device pulls this line low, all system memory is disabled . This line has a 1K ohm pullup to +5V.
33	-12V	O	Negative 12 volt supply*, 2ØØmA total for <u>all</u> peripheral boards together.
34	-5V	O	Negative 5 volt supply*, 2ØØmA total for <u>all</u> peripheral boards together.
35	SYNC	O	The 65Ø2 opcode synchronization signal. Can be used for external bus control signals.
36	C7M	O	Seven MHz high frequency clock.
37	Q3	O	A 2MHz (nonsymmetrical) general purpose timing signal.

38	$\overline{\text{C1M}}$	O	Complement of C1M clock.
39	$\overline{\text{IOCLR}}$	O	Provides the C8ØØ space disable function directly without address decoding (CFFF was used as the address for disabling the expansion ROM). It is addressed at CØ2X.
4Ø	C1M	O	Phase C1M clock. This is a constant 1MHz regardless of system operational mode. When the system is in the 1MHz mode, this is the same as the microprocessor 1M clock.
41	$\overline{\text{DEVICE SELECT}}$x	O	Sixteen addresses are set aside for each peripheral connector. A read or write to such an address will cause Pin 41 on the selected connector to go low during the micro PHØ (5ØØns in 1MHz mode; 25Øns in 2MHz mode.)
42-49	D7-DØ	I,O	8-bit system data bus. During a write cycle, data is set up by the 65Ø2 less than 3ØØns after the beginning of C1M. During a read cycle the 65Ø2 expects data to be ready no less than 1ØØns before the end of C1M.
5Ø	+12V	O	Positive 12 volt supply*, 3ØØmA total for <u>all</u> peripheral boards together.

*Note: Total power drawn by any one peripheral board is not to exceed 1.5 watts.

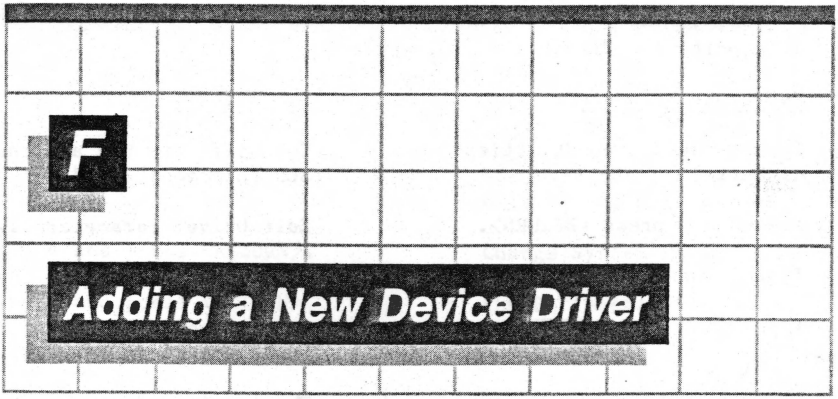
** Indicates the direction of the signal: I means input to the Apple III from the peripheral; O means output from the Apple III to the peripheral; I,O means either direction is possible (e.g., R/$\overline{\text{W}}$ or data).

E

ASCII Conversion Tables

Dec	ASCII	Octal	Hex	Binary 76543210	Dec	ASCII	Octal	Hex	Binary 76543210
0	NUL	000	00	00000000	32	SP	040	20	00100000
1	SOH	001	01	00000001	33	!	041	21	00100001
2	STX	002	02	00000010	34	"	042	22	00100010
3	ETX	003	03	00000011	35	#	043	23	00100011
4	EOT	004	04	00000100	36	$	044	24	00100100
5	ENQ	005	05	00000101	37	%	045	25	00100101
6	ACK	006	06	00000110	38	&	046	26	00100110
7	BEL	007	07	00000111	39	'	047	27	00100111
8	BS	010	08	00001000	40	(	050	28	00101000
9	HT	011	09	00001001	41	)	051	29	00101001
10	LF	012	0A	00001010	42	*	052	2A	00101010
11	VT	013	0B	00001011	43	+	053	2B	00101011
12	FF	014	0C	00001100	44	,	054	2C	00101100
13	CR	015	0D	00001101	45	-	055	2D	00101101
14	SO	016	0E	00001110	46	.	056	2E	00101110
15	SI	017	0F	00001111	47	/	057	2F	00101111
16	DLE	020	10	00010000	48	0	060	30	00110000
17	DC1	021	11	00010001	49	1	061	31	00110001
18	DC2	022	12	00010010	50	2	062	32	00110010
19	DC3	023	13	00010011	51	3	063	33	00110011
20	DC4	024	14	00010100	52	4	064	34	00110100
21	NAK	025	15	00010101	53	5	065	35	00110101
22	SYN	026	16	00010110	54	6	066	36	00110110
23	ETB	027	17	00010111	55	7	067	37	00110111
24	CAN	030	18	00011000	56	8	070	38	00111000
25	EM	031	19	00011001	57	9	071	39	00111001
26	SUB	032	1A	00011010	58	:	072	3A	00111010
27	ESC	033	1B	00011011	59	;	073	3B	00111011
28	FS	034	1C	00011100	60	<	074	3C	00111100
29	GS	035	1D	00011101	61	=	075	3D	00111101
30	RS	036	1E	00011110	62	>	076	3E	00111110
31	US	037	1F	00011111	63	?	077	3F	00111111

Dec	ASCII	Octal	Hex	Binary 76543210	Dec	ASCII	Octal	Hex	Binary 76543210
64	@	100	40	01000000	96	`	140	60	01100000
65	A	101	41	01000001	97	a	141	61	01100001
66	B	102	42	01000010	98	b	142	62	01100010
67	C	103	43	01000011	99	c	143	63	01100011
68	D	104	44	01000100	100	d	144	64	01100100
69	E	105	45	01000101	101	e	145	65	01100101
70	F	106	46	01000110	102	f	146	66	01100110
71	G	107	47	01000111	103	g	147	67	01100111
72	H	110	48	01001000	104	h	150	68	01101000
73	I	111	49	01001001	105	i	151	69	01101001
74	J	112	4A	01001010	106	j	152	6A	01101010
75	K	113	4B	01001011	107	k	153	6B	01101011
76	L	114	4C	01001100	108	l	154	6C	01101100
77	M	115	4D	01001101	109	m	155	6D	01101101
78	N	116	4E	01001110	110	n	156	6E	01101110
79	O	117	4F	01001111	111	o	157	6F	01101111
80	P	120	50	01010000	112	p	160	70	01110000
81	Q	121	51	01010001	113	q	161	71	01110001
82	R	122	52	01010010	114	r	162	72	01110010
83	S	123	53	01010011	115	s	163	73	01110011
84	T	124	54	01010100	116	t	164	74	01110100
85	U	125	55	01010101	117	u	165	75	01110101
86	V	126	56	01010110	118	v	166	76	01110110
87	W	127	57	01010111	119	w	167	77	01110111
88	X	130	58	01011000	120	x	170	78	01111000
89	Y	131	59	01011001	121	y	171	79	01111001
90	Z	132	5A	01011010	122	z	172	7A	01111010
91	[	133	5B	01011011	123	{	173	7B	01111011
92	\	134	5C	01011100	124	|	174	7C	01111100
93	]	135	5D	01011101	125	}	175	7D	01111101
94	^	136	5E	01011110	126	~	176	7E	01111110
95	_	137	5F	01011111	127	DEL	177	7F	01111111



The chart that follows has two columns. The left-hand column gives you the exact keystroke sequence to use if you have Version 4.Ø of the Utilities diskette and are using only the built-in diskette drive. If you want further explanation of what you are doing each step of the way, read the right column.

1. Bring in the Driver from the UPIC Diskette

Keystroke Sequence	Explanation
Insert the System Utilities diskette in the built-in Apple III drive and close the drive door.	Be sure this diskette is write-protected.
Turn on the Apple III, or press <CTRL-RESET>.	Boot the Apple III.
When you see the list of utilities available, type 3 and press <RETURN>.	Select the System Configuration Program (SCP).
Type 1 and press <RETURN>.	Add a Driver File...
Insert the UPIC diskette.	...from the backup copy.
Type .D1/PRINTER or .D1/PARALLEL and press <RETURN>.	Select the appropriate driver (usually .PRINTER).
Press <RETURN> a second time.	Return to SCP menu.

Go on to step 2.

2. Modify the Driver for a Given Device

Keystroke Sequence	Explanation
Insert the System Utilities diskette.	This diskette has all the SCP functions on it.
Type 3 and press <RETURN>.	Edit Driver Parameters...
Type 1 and press <RETURN>.	...of the only driver there.
Type 5 and press <RETURN>.	You are going to change the Configuration Block data.
Type the proper two digits in each position, following directions given at the bottom of the screen display.	These values come from Table 2-2 for the commonly used printers, or from the Configuration Block section of Chapter 4 or 5.
Press <RETURN> three times, slowly.	Return to the SCP menu.

Go on to step 3.

Printer	Device Configuration Block (DCB) Values*				
	Ø	1	2	3	4
Centronics 779/7ØØ	EØ	CØ	4Ø	ØØ	ØA
Centronics 73Ø/737	CØ	CØ	ØØ	ØØ	5A
Anadex DP-8ØØØ	EØ	CØ	ØØ	ØØ	5A
Printronix P3ØØ	EØ	CØ	ØØ	ØØ	ØA
IDS 44Ø/445/46Ø	6Ø	4Ø	ØØ	ØØ	5A
Epson MX-8Ø	E8	C8	ØØ	ØØ	ØA
Texas Instruments 81Ø	E8	CØ	ØØ	ØØ	ØA
Any printer connected via an Apple II cable	ØØ	ØØ	(final 3 same as above)		

*These numbers are in hexadecimal; hence some of them contain letters of the alphabet.

Table F-1. DCB Values for Commonly Used Printers

3. Change the Slot Assignment If Necessary

Keystroke Sequence	Explanation
Is the UPIC in slot 1? If so, skip to step 4.	You need to perform this step only if the UPIC is not in slot 1.
Type 4 and press <RETURN>.	Change System Parameters:
Type 2 and press <RETURN>.	the Peripheral Slot Assignment...
Type 1 and press <RETURN>.	...of the only driver there.
Type the number of the slot where you installed the UPIC.	The assumed slot number, 1, is currently posted for the card.
Press <RETURN> three times.	Return to the SCP menu.
Go on to step 4.	

4. Save the Configured Driver on the UPIC Diskette

Keystroke Sequence	Explanation
Type 5 and press <RETURN>.	Generate New System.
If the question Perform System Validation Check? appears, reply Y.	The validation check may take place automatically. Since there is currently only one driver, warning messages will appear.
Ignore any warning messages that may appear.	This time around you simply want to save a configured driver on the UPIC diskette.
Put the UPIC diskette into the drive.	As always, use the backup copy.
Type .D1/CEN737 or .D1/IDS445 or similar, then press <RETURN>.	You want to give this configured driver a filename on diskette that reminds you of what it is for.
Press <RETURN> again.	Return to the SCP menu once again.
Do you want to prepare other driver configurations? If so, go back to Step 2. If not, go on to step 5.	If you intend to use the UPIC for more than one purpose, now is a good time to prepare various versions of the drivers.

5. Prepare Each Boot Diskette for the New Driver

Keystroke Sequence	Explanation
Insert the System Utilities diskette and press <CTRL-RESET>.	To guard against slip-ups, reboot for each new diskette.
Type 3 and press <RETURN>.	Select SCP once again.
Type 1 and press <RETURN>.	Add a Driver File.
Insert the first/next boot diskette.	
Type .D1/SOS.DRIVER and press <RETURN>.	Bring in all current drivers and examine them.
Insert the System Utilities diskette.	
If you are adding .PRINTER and there is already a driver named .PRINTER on this boot diskette, decide if you want to rename and keep it (step 5a) or remove it (step 5b). If there is none, skip to step 6. In any case, press <RETURN> before continuing.	After examining the list of drivers, be sure to press <RETURN> to return to the SCP menu before going on to step 5a, 5b or 6.

5a. Rename the Serial Printer Driver

Keystroke Sequence	Explanation
Type 3 and press <RETURN>.	Edit Driver Parameters.
Type the number you see next to the name .PRINTER, then press <RETURN>.	Specify the driver you are going to rename.
Type 1 and press <RETURN>.	Device Name is to change.
Type .SPRINTER (for "Serial Printer") and press <RETURN>.	This is only a suggested name.
Press <RETURN> twice more.	Return to the SCP menu.
Skip to step 6.	

5b. Delete the Serial Printer Driver

Keystroke Sequence	Explanation
Type 2 and press <RETURN>.	Delete a Driver.
Type the number you see next to the name .PRINTER, then press <RETURN>.	This is the old Serial Printer Driver.
Press <RETURN> twice.	Return to the SCP menu.

Go on to step 6.

6. Add the New Driver

Keystroke Sequence	Explanation
Type 1 and press <RETURN>.	Add a Driver File...
Insert the UPIC diskette.	...from the backup copy.
Type .D1/CEN737, or whatever filename you gave to the modified driver, then press <RETURN>.	No matter what filename it had on diskette, the driver again has its name .PARALLEL or .PRINTER back.
Insert the System Utilities diskette and press <RETURN>.	Return to the SCP menu.

Go on to step 7.

7. Generate a New System with the New Driver in It

Keystroke Sequence	Explanation
Type 5 and press <RETURN>.	Generate a New System.
If you see the question, Perform System Validation Check?, reply Y and press <RETURN>.	This validation check may take place automatically. If there are warnings, redo all the steps, referring to the Apple III Standard Device Drivers manual where necessary.
Reinsert the boot diskette you are changing.	
Type .D1/SOS.DRIVER; press <RETURN>.	

If there are more boot diskettes to change, go back to step 5.

For testing purposes, either make a backup copy of each boot diskette before adding a driver to it, or use the procedure outlined in the Generate New System section of the Apple III Standard Device Drivers manual. Test each new boot diskette to make sure the new configuration is correct and works OK.

Glossary

Acknowledge: n. A signal arriving (ACK IN) or sent (ACK OUT) on a specific connector pin to indicate that the Apple III or attached device is ready to receive a byte of data (.PARALLEL handshake) or has just successfully received a byte of data (.PRINTER handshake).

Buffer: n. A memory area in a computer or other device that can hold information temporarily. Buffers improve the performance of computer systems by compensating for differences in speed between one device and another, or between one type of activity (single-byte transfers) and another (block transfers). For example, a computer program may produce a burst of 15Ø characters every 1Ø seconds, but the printer may be capable of printing only 15 characters per second. By supplying the printer with a 15Ø-character buffer, the computer can send all its characters rapid-fire and then go back to doing something else while the printer lopes along at its own pace.

Carriage Return: A specific ASCII character (ØD in hexadecimal) that ordinarily causes a printer or display screen to place the subsequent character at the beginning of the next line of text. On a manual typewriter, carriage return and linefeed usually go together: the platen is shifted to the right and the paper is advanced one or more lines in a combined motion. Computer people, being analytical, always treat them separately.

Character: n. Any symbol that has a widely-understood meaning. In computers, letters, numbers, punctuation marks, and even what are normally just concepts (such as carriage returns) are all characters.

Device: n. A piece of computer hardware, such as a disk drive or printer or terminal. "Device" is short for "Peripheral Device." "Peripheral" is also short for "Peripheral Device." The multiplicity of names is another attempt to confuse new users, creating a rite of passage, as it were.

Device Configuration Block: A set of hexadecimal values that specifies to a device driver the characteristics of the device connected to that driver, for example how fast it can receive characters.

Device Driver or Device Handler: A small program that acts as a communications link between a device and the computer's operating system.

Diskette: n. A flat, circular piece of flexible plastic, coated with a fine metallic powder, onto which information is recorded magnetically.

Driver: See Device Driver.

Emulation: Techniques using software or microprogramming in which one computer is made to behave exactly like another.

Handshake: n. A kind of communications protocol in which the receiving device, when it has successfully gotten a character or block of characters, sends back an acknowledging signal, thereby triggering the next transmission.

In Check: An error condition somewhere in a device (usually a printer) of sufficient severity that the computer should not attempt to transmit data to that device.

Input: n. Information (data) arriving at a computer or device.
-- v. To type information into the computer (obsolete jargon).

Interface: n. 1. The electronic components that allow two different devices, or the computer and a device, to communicate. 2. The part of a computer program that interacts with the user. "If we change this program, how will the user interface be affected?"

Invocable Module: A self-contained unit of program code that has specific information in precise locations (usually at the beginning of the module) so that a user program (written in BASIC) can call and use it in a standardized way.

LSB: Least significant (that is, rightmost) bit of a number.

Linefeed: n. An ASCII character (hexadecimal ØA to be exact) that causes a printer to advance the paper one line. Without linefeeds, the printer would keep printing over and over again on the same line.

Module: n. A self-contained series of computer instructions that performs some specific function.

MSB: Most significant (leftmost) bit of a number.

Online: adj. Under control of the Apple III; opposite of offline, or under control of the human operator.

Output: n. Data that have been, are being, or are to be transmitted from the Apple III to some other device. -- v. The act of transmitting data (obsolete jargon).

Parallel Interface: A type of interface in which all bits of a given character are transferred simultaneously, using a separate data line for each bit.

Parameter: n. A variable that can have one of a specific set of values.

Peripheral Connector Slot: In an Apple III, a 5Ø-pin slot designed to hold, and transfer signals to and from, an interface card.

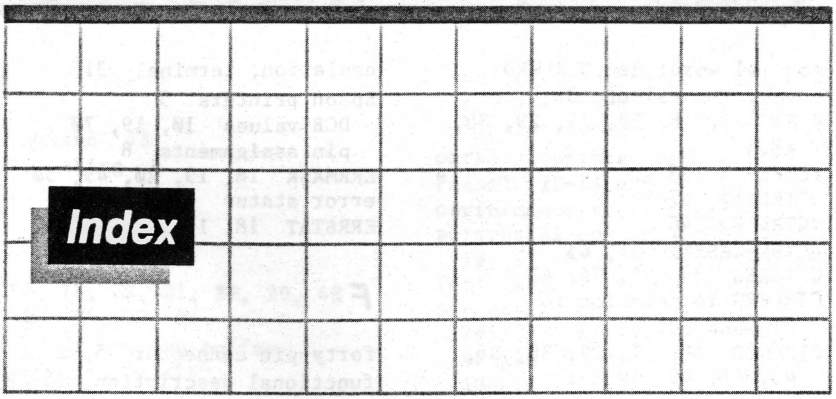
Radio Frequency Interference (RFI): Electromagnetic noise at frequencies that cause disturbances in nearby televisions, radios, and other radio frequency receivers.

ROM: Read-Only Memory, an integrated circuit on the UPIC that contains programs that can be read and used, but not rewritten or changed.

Serial Interface: A type of interface in which all bits of a given character are transmitted along the same data line in a stream, one after the other. (See Parallel Interface.)

Strobe: n. A brief signal pulse arriving (STROBE IN) from a transmitting device or sent (STROBE OUT) by the Apple III to a receiving device to indicate that a valid byte is present on the data lines, ready to be read.

UPIC: Universal Parallel Interface Card.



A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X – Y – Z

Distributed By
Sun Remarketing