this applies to Apple Pascal 1.0 & may not be totally incorrect!!
Not quite, but locations & drives etc are almost certainly different.
Read the 1:1 Attack BIOS file dump first

THE PRELIMINARY APPLE PASCAL GUIDE TO INTERFACING FOREIGN HARDWARE

10 DEC 1979 (minor revision)

NB Keep this

Some of the alphabet are not repeated elsewhere

This document is intended to direct users of the APPLE II who are interfacing to their machine hardware other than Apple peripheral cards. Its primary target is the community of peripheral card manufacturers who have developed a product for the Apple II under BASIC, and wish to develop an end-user Pascal interface. It is assumed that the reader is familiar with Apple II hardware and is an experienced programmer at the machine level, and has programmed in Pascal.

How Apple Pascal looks at the outside world

Currently, Apple Pascal is capable of recognizing the presence of an Apple (brand) peripheral card in slots 1 through 7, which it does at boot time by scanning the ROM in the slot address space. The purpose of this scan is (a) to determine if any card exists in the slot, since an open slot yields a random response; and (b) to find out what kind of Apple card, if any, is there, since the four types of Apple cards (printer, com, disk, and serial) have distinct values at byte locations Cn05 and Cn07. A foreign (other manufacturer's) card generally fails this test and so is disqualified in the Pascal system's list of active slots. Let us say, for example, that the Apple user has installed a foreign printer card in slot 1, and attempts to do a WRITELN to the printer. Pascal formats the request and sends it to the interpreter, which in turn reformats it and sends it to the low-level I/O package, which in Apple Pascal is called the BIOS for Basic I/O Subsystem. At the BIOS level, the slot list is checked to see what kind of card is in slot 1 (the only legal slot for the printer). It is at this point that the output request fails, since slot 1 is marked invalid unless an Apple printer, com, or serial card is in place. It is important to note in what follows that the BIOS driver routine is the only routine in the system that accesses the slot.

The BIOS

The term BIOS not only refers to the set of I/O drivers installed in the file SYSTEM.APPLE in the Pascal system, but also to the written specification for these drivers. This precise specification of the input and output to Pascal was created by the University of California, San Diego (UCSD) to make it easy to interface UCSD Pascal (parent of Apple Pascal) to any number of machines and machine configurations. The type of things specified in the BIOS includes data and control parameters, calling sequence, unit numbers, and error handling requirements.

For example, an Apple-specific version of the BIOS for the printer contains the following information:

(1) Transfer to the printer device is one character at a time.

(2) The following special characters must be recognized:
CR (carriage return) (hex 0D) Prints the line and returns the carriage to the first column. An automatic line feed should NOT be done.
LF (line feed) (hex 0A) Sent only after a CR. Should do a simple line feed (no return) if possible, else a CRLF.
FF (form feed) (hex 0C) Advance the paper to top-of-form (if possible) and perform a carriage return.
(3) There are only two calls to the Printer BIOS, a write and an initialization call. The initialization call should perform carriage return-line feed IF desired, but should not do a form-feed. The printer buffer should be flushed.

Interface Routines	Parameters
Printer write	data byte in register A return completion code in register X (zero if no error)
Printer init	completion code in register X (zero if online, nine if offline)

The completion code is identical to the Pascal IORESULT.

(End of Printer BIOS)

A complete description of the BIOS specification is beyond the scope of this document. The BIOS specification is available at cost from the UCSD Pascal distributor (Softech Microsystems of San Diego).

Changing the BIOS

At Apple, a BIOS has been written according to the UCSD specification to interface with all Apple peripherals. It is possible to modify the BIOS module within SYSTEM.APPLE to accommodate foreign types of peripheral cards, or to alter the behavior of an existing device such as the screen. This document contains the information necessary to enable one to manipulate the BIOS in the Pascal system in order to add non-Apple devices.

Pascal units for I/O drivers

For many types of additional hardware, such as an external clock, it is better not to alter the BIOS but to construct a Pascal unit containing routines to interface with the hardware. This unit (which may consist of assembly language routines) accesses the Cxxx addresses directly to initialize, command, and retrieve data from the hardware. The Pascal user then simply "uses" the unit and has available the routines he needs, without a system reconfiguration. (Note: The Cxxx address space may be accessed directly from Pascal with a "trick" record:

```
DEVICEBYTES = PACKED ARRAY [0..n] OF CHAR;
TRIX = RECORD CASE BOOLEAN OF
          FALSE:(ADDR: INTEGER);
          TRUE :(CONTROLREC: ^ DEVICEBYTES)
```

```
END;
VAR CARD: IRIX;
```

One then accesses data by the sequence

```
CARD.ADDR := (address of hardware);
CARD.CONTROLREC[0] := CHR(INITMASK)        etc.
```

The type CHAR is used to ensure that only the desired byte is accessed (not the whole word).)

Patching the BIOS.

Some applications such as printers really are required to be built into the system (i.e. so that WRITELN statements access the proper output device). APPLE PASCAL is aware of the following I/O devices: CRT screen, keyboard, printer, graphics output, remote in, remote out, and block devices (disks). Remote input and remote output are general ports for serial or parallel communication. Any device that fits into one of these categories can be handled by APPLE PASCAL, by inserting a customized I/O driver into the BIOS. The procedure to do so consists of the following steps:

1. Write the I/O driver according to the BIOS specification.
2. Patch the I/O driver into SYSTEM.APPLE at a specified location (see below).
3. Patch the appropriate BIOS vector in SYSTEM.APPLE.

The "specified locations" in step 2 are in two parts. First, one may make use of the space used by the existing driver if there is one. These spaces are listed in Table I below. Generally, it is preferable to use this space in order to avoid wasting it.

If the space shown in Table I is insufficient, a small amount of unused space exists at locations hex DA8E - DB7F. This is block 5, byte 190 to block 5, byte 383 (decimal), inclusive.

The BIOS vectors to patch to point to the new routine are given by Table II. Note that the method of routine entry is via a JSR, with the return address on the stack.

The actual patching of SYSTEM.APPLE must be done by a user-supplied patcher; an easy method is to use BLOCKREAD to read in the SYSTEM.APPLE file and the new driver object file, then MOVELEFT to copy the new driver in and BLOCKWRITE to write the new interpreter file. A supplier should provide to the user, as a package, such a program along with his hardware and altered BIOS routine, making it easy for an Apple user to add several modifications to his single Apple Pascal system.

Folding of the Language card memory is taken care of automatically by the interpreter.

When writing BIOS routines, there is one additional consideration: the keyboard input to Apple Pascal is done by polling, not by interrupts; therefore, at convenient locations throughout the BIOS there exist calls to the character-available check routine located at location D681 in the interpreter. When replacing the BIOS routines with customized counterparts, those substitute routines should each contain a call to the check routine.

TABLE I

BIOS DRIVER AREAS IN FILE SYSTEM.APPLE

AREA	RAM LOCATION LO-ADDR	HI-ADDR	SYSTEM.APPLE FILE LO-BLK,BYTE	HI-BLK,BYTE
CONSOLE INIT & READ	D681	D787	3,129	3,391
CONSOLE WRITE	D7D0	D7F6	3,464	3,502
SCREEN DRIVER	D87B	DA3D	4,123	5,189
PRINTER INIT	D788	D790	3,392	3,400
PRINTER WRITE	D830	D84D	4,48	4,77
REMOTE INIT	D79C	D7A2	3,412	3,418
REMOTE READ	D84E	D87A	4,78	4,122
REMOTE WRITE	D809	D810	4,9	4,16
COMMON ROUTINES FOR PRINTER & REMOTE INIT	D791	D79B	3,401	3,411
	D7A3	D7CF	3,419	3,463
SERIAL CARD WRITE	D7F7	D808	3,503	4,8
PRINTER CARD WRITE	D811	D81E	4,17	4,30
COM CARD WRITE	D81F	D82F	4,31	4,47
DISK ROUTINES	D000	D569	0,0	2,361
GRAPHIC WRITE	none			

Notes on Table I.

RAM addresses are given in hex. Locations in the file SYSTEM.APPLE are given in decimal as a block & byte offset from the beginning of the file (starting at zero).

Console init routines contain routines used by various other parts of the system, including console write. It is best not to destroy these. Replacement of the Apple keyboard will entail replacement of the console character available check routine mentioned above, with a hook at the original location. Note that the console is highly reconfigurable from the PASCAL level (program SETUP); thus, one should be very thoughtful before putting out the effort to change the console BIOS.

Console write calls the screen driver, unless an external terminal is attached.

Both the printer and remote init routines use both of the common routines listed.

Printer write uses printer card write, serial card write, and com card write. Remote write uses serial card write, com card write, and printer write (sorry about the nonmodularity).

The console and disk drivers are considered replaceable and not extendable. Therefore, the (large) space allotted in Table I is considered sufficient for console and disk needs. Users wishing to interface with the existing console or disk drivers should do so with the cooperation of Apple Computer.

TABLE III

BIOS VECTORS

VECTOR	RAM LOCATION	SYSTEM.APPLE FILE BLK,BYTE
CONSOLE READ	FF7B	23,379
CONSOLE WRITE	FF84	23,388
CONSOLE INIT	FF8D	23,397

PRINTER INIT	FF9B	23,406
DISK WRITE	FF9F	23,415
DISK READ	FFA8	23,424
DISK INIT	FFB1	23,433
REMOTE READ	FFBA	23,442
REMOTE WRITE	FFC3	23,451
REMOTE INIT	FFCC	23,460
GRAPHICS WRITE	FFD5	23,469
	FFDE	23,478

All vectors are the 16-bit argument of a JSR instruction. The address given is the lo-byte address; the high byte goes in the (given address) + 1.

Zero Page Usage

If zero page storage is needed, the following information is useful Pascal makes use of variables stored at $50 thru $FF. The disk routines use variables at $36-$4F. Locations $00-$35 are considered to be pure temporaries, i.e. information may be stored in these locations but may be destroyed by other routines. Also, locations $200-$399 form a temporary area used by the disk and screen-scrolling routines, and may be used as a temporary buffer space for any other routine.

It should be noted that the drawback to all this patching is that it is dependent on the current system version. It is Apple's intention, however, to provide a standard, generalized method for system reconfiguration before the next system release occurs.

BIOS listing

Attached is a listing of the Apple Pascal BIOS, which provides the exact specification of the inputs and outputs of the drivers.