Apple II Technical Notes

Developer Technical Support

Pascal #12: Disk Formatter Routine

Revised by: Cheryl Ewy & Dan Strnad November 1988
Revised by: Cheryl Ewy June 1985

This Technical Note documents the Apple II Pascal 1.3 Disk Formatter routine.

Introduction

Integrating the Pascal Disk Formatter utility into your application program will free the user from having to format Pascal disks prior to running your program. Error codes that specify any problems encountered during the formatting process are returned. The disk contains the following files:

FORMATTER.TEXT is a sample Pascal host program that illustrates the use of the formatter routine.

FORMDISK.TEXT is an assembly language function that is linked to your Pascal host program. It contains the code to format disks in ProDOS blocked devices and calls the `ASMFORMAT` function to format disks in Disk II drives.

ASMFORMAT.TEXT is the Disk II formatter, an assembly language procedure that must be specially handled (see below).

BOOTTRACKS.DATA is a data file that is used to create the formatter data file. It contains boot blocks for both Disk II drives and ProDOS blocked devices and a blank disk directory.

MAKEFMT.TEXT, MAKEFMT.CODE are a Pascal program that will create the required formatter data file.

FORMATTER.DATA is a complete formatter data file (identical to that supplied with the Apple II Pascal 1.3 Development System).

FORMATTER.CODE is the formatter program supplied with the Apple II Pascal 1.3 Development System.

All programs are supplied in source (and where appropriate, as code files) so that you may modify them for your own particular purposes.

ASMFORMAT – The Disk II Formatter Routine

The file ASMFORMAT.TEXT contains a proprietary subroutine that performs the actual formatting of Disk II disks. It is written in 6502 assembly language suitable for assembly by the Apple II or Apple /// Pascal Assembler. This code requires special handling by the host program to ensure a reliable format. It contains critical timing code, and because of this, it must be located on a page boundary in memory (a location of the form xx00, e.g., 3D00, 2000, etc.). To do this, it must be assembled `ABSOLUTE` and you must use `ORG` to place it on particular page boundary. It comes supplied at location 3D00, which is the location used by the formatter routine supplied with the Apple II Pascal 1.3 Development System (FORMATTER.CODE). If you need to move it to another particular location you must change the `.ORG` statement in the file to the new address. The formatter will not work reliably if it is not on a page boundary. The code itself is 1082 bytes in length.

Because of the special nature of this code, it must be loaded by the Pascal host program at the chosen location. The following sample code illustrates how this is done:

```
TYPE MEMARRAY = PACKED ARRAY [0..1535] OF 0..255;

       MEMPTR = RECORD CASE BOOLEAN OF
                     TRUE:  (ADDR: INTEGER);
                     FALSE: (MEM: ^MEMARRAY);
              END;

VAR  LOADPTR: MEMPTR;      {this is the pointer to the absolute memory
                           location where the Disk II formatter routine
                           will be loaded.}


       {the following code will load the Disk II formatter routine from the
formatter data file into memory at a fixed location}

       RESET(DATAFILE, '%FORMATTER.DATA');

       LOADPTR.ADDR := 15616;     {this value is the absolute memory location
                                  where the code is to be loaded.  In this
                                  example, 15316 is the decimal equivalent of
                                  the memory address 3D00.}

       BLOCKSREAD := BLOCKREAD(DATAFILE, LOADPTR.MEM^, 3);
{the above line will load three blocks (the Disk II formatter code) from the data file into the
memory space specified in LOADPTR}
```

The Disk II formatter routine assumes that the A register has been setup with the slot number and drive number of the disk which is to be formatted. `FORMDISK` sets up this information before doing a `JSR` to the Disk II formatter routine. The contents of the A register are defined as follows:

Bit 7	Drive number. 0=Drive 1, 1=Drive 2
Bits 6-4	Slot number. 100=4, 101=5, 110=6. No other slots are supported.
Bits 3-0	Reserved; must be set to zero.

After the Disk II formatter routine is called, it returns an error code in the A register. `FORMDISK` then returns this error code to the host program. The error codes are listed in the following section.

FORMDISK – The Main Formatter Routine

The file FORMDISK.TEXT is an assembly language function that is assembled and linked to your Pascal host program. This function determines whether the drive containing the disk to be formatted is a Disk II drive or a ProDOS blocked device. If it is a Disk II drive, `FORMDISK` invokes the Disk II formatter routine with the required parameters as described in the previous section. If the drive is a ProDOS blocked device, `FORMDISK` sets up the proper parameters and executes a format call to the device. `FORMDISK` will return an error code back to the Pascal host after the formatting is complete. The call to this function is shown below:

```
 VAR  ERRCODE: INTEGER;                 {the error code returned}
         VOLNUM:  INTEGER;              {the volume (unit) number of the disk}
ERRCODE := FORMDISK(VOLNUM);            {the function call}
```

There are six possible error codes returned by `FORMDISK`. They indicate problems that may have occurred during the formatting process. They are as follows:

Error code	**Error**	**Possible causes**
00	No Error	Formatting successfully completed
39	Unable to format the disk	No disk in drive; drive door not closed; bad media
43	Disk is write-protected	Disk is write-protected; disk is pushed halfway into drive, activating the write-protect switch
47	No disk in drive	The disk drive is empty. This error is only reported for ProDOS block devices. If a Disk II drive is empty, error #39 is returned.
51	Drive speed is too slow	The drive motor speed requires adjustment, it is too slow. This error is only reported for Disk IIs.
52	Drive speed is too fast	The drive motor speed requires adjustment, it is too fast. This error is only reported for Disk IIs.

To use the `FORMDISK` function requires that you modify one `.EQU` statement in the source file (FORMDISK.TEXT) to specify the location of the Disk II formatter routine in memory. Currently, the statement reads as follows:

```
DO_FORMAT   .EQU   3D00   ;memory address of the Disk II formatter routine
```

If you decide to relocate the Disk II formatter routine, simply change this value to reflect the new memory address, then reassemble `FORMDISK`. The `FORMDISK` function does a `JSR` to this value to invoke the Disk II formatter routine.

Note: The value used in the `.ORG` in `ASMFORMAT` and the `.EQU` in `FORMDISK` **must** match.

Making a Formatter Data File

To use the formatter requires a data file that contains three pieces:

1. The Disk II formatter routine code, to be loaded into memory.
2. The boot code that is written to blocks 0 and 1 of the formatted disk.
3. A blank UCSD Pascal directory that is written to block 2 of the formatted disk.

The formatter disk comes with the second and third parts in the file BOOTTRACKS.DATA. This four-block file contains the boot blocks for Disk II drives and ProDOS blocked devices and the blank directory. Once the Disk II formatter routine has been assembled (to ASMFORMAT.CODE) it must be concatenated to the BOOTTRACKS.DATA file to make the formatter data file. The Disk II formatter routine code occupies the first 3 blocks of the formatter data file, which is then followed by the contents of the BOOTTRACKS.DATA file. Because the assembler puts special informational content blocks into a code file, a special program is required to copy only the blocks containing the code of the Disk II formatter routine. This is the program MAKEFMT.CODE. This program copies blocks 1, 2, and 3 of ASMFORMAT.CODE to blocks 0, 1, and 2 of the file FORMATTER.DATA. It then copies blocks 0, 1, 2, and 3 of the file BOOTTRACKS.DATA to blocks 3, 4, 5, and 6 of the file FORMATTER.DATA. This makes the required formatter data file (7 blocks in size) that will be used by the Pascal host program. `MAKEFMT` requires that the files ASMFORMAT.CODE and BOOTTRACKS.DATA be on the prefix volume. Set the Pascal prefix to this volume and `X(ecute MAKEFMT`. It will create the file FORMATTER.DATA on the same volume. The source for this program is included so that you may modify it as needed.

The Pascal Host Program

It is up to you to write the Pascal host program. On the disk is a sample program (the Apple II Pascal 1.3 Formatter) that you may study. It illustrates the above techniques. The primary functions of the Pascal host are to:

1. Open the FORMATTER.DATA file.
2. Read blocks 0 – 2 into a memory location that is on a page boundary.
3. Read blocks 3 – 6 into a 2,048 byte buffer.
4. Call the assembly language function `FORMDISK` with the volume number of the drive containing the disk to be formatted.
5. Examine the error code returned. If there is an error then report it to the user, otherwise continue.
6. Use `UNITSTATUS` to determine whether the drive is a Disk II or a ProDOS blocked device and how many blocks are on the disk.
7. Use the number of blocks returned by `UNITSTATUS` to update the maximum

block number information in the blank directory.

8. If the drive is a Disk II, use `UNITWRITE` to write blocks 0 – 2 from the buffer to blocks 0 – 2 on the newly formatted disk.
9. If the drive is a ProDOS blocked device, use `UNITWRITE` to write block 3 from the buffer to block 0 on the newly formatted disk, then use it again to write block 2 from the buffer to block 2 on the disk.

The following code is an example of how to read in the blocks from the FORMATTER.DATA file, determine the drive type, update the directory, and write the boot blocks and directory to the newly formatted disk:

```
TYPE BYTARRAY = PACKED ARRAY [0..1] OF 0..255;

VAR  BUFFER: PACKED ARRAY [0..2048] OF 0..255;

NUMBLOCKS : INTEGER;

TRIX : RECORD CASE BOOLEAN OF
                    TRUE  : (INT : INTEGER);
                    FALSE : (BYT : BYTARRAY);
             END;

{read in the boot blocks and directory}
NUMBLOCKS := BLOCKREAD (DATAFILE, BUFFER, 4, 3);

{determine type of disk drive and number of blocks on the disk}
UNITSTATUS (VOLNUM, NUMBLOCKS, 1);

{update maximum number of blocks in blank directory}
TRIX.INT := NUMBLOCKS;
BUFFER[1038] := TRIX.BYT[0];
BUFFER[1039] := TRIX.BYT[1];

{write out the boot blocks and directory to a Disk II disk}
UNITWRITE (VOLNUM, BUFFER, 1536, 0);

{write out the boot block and directory to a ProDOS blocked device disk}
UNITWRITE (VOLNUM, BUFFER[1536], 512, 0);
UNITWRITE (VOLNUM, BUFFER[1024], 512, 2);
```

A dynamic variable can also be used as the buffer so that your program can reclaim the buffer space for its own use after the formatting is completed:

```
TYPE BUFFER = PACKED ARRAY [0..2048] OF 0..255;

VAR BUFPTR : ^BUFFER;
OLDPTR : ^INTEGER;

 {mark the beginning of usable space}
MARK (OLDPTR);
 {allocate space for the buffer}
NEW (BUFPTR);
 {read in the boot blocks and directory}
NUMBLOCKS := BLOCKREAD (DATAFILE, BUFPTR^, 4, 3);
{write out the boot blocks and directory to a Disk II disk}
UNITWRITE (VOLNUM, BUFPTR^, 1536, 0);
{release the space used by the buffer}
RELEASE (OLDPTR);
```

In Review

The following is a step-by-step review of how to use the formatting routine.

1. Determine where in memory you wish to load the Disk II formatter routine. Remember it must be on a page boundary.
2. Edit the file ASMFORMAT.TEXT, and change the value in the `.ORG` statement to be the memory address chosen.
3. Assemble ASMFORMAT.TEXT to ASMFORMAT.CODE.
4. `X(ecute MAKEFMT` to make the required FORMATTER.DATA file.
5. Edit the file FORMDISK.TEXT and change the line

   ```
   DO_FORMAT   .EQU   3D00
   ```

   to reflect the new memory location (same value as in the `.ORG` statement above).
6. Assemble FORMDISK.TEXT to FORMDISK.CODE.
7. Write the Pascal host program using the above techniques for loading the Disk II formatter routine, calling the `FORMDISK` function, updating the blank directory, and writing the boot blocks and directory. Remember error reporting.
8. Compile the Pascal host.
9. Link the Pascal host to the file FORMDISK.CODE, thus linking the `FORMDISK` function into your program.
10. With the linked Pascal host program and the FORMATTER.DATA file you can now format disks.