GETTING STARTED WITH OSS

CONGRATULATIONS !!!

You have purchased what we believe is by far the most advanced software development package available for the Atari 800 and Atari 400 personal computers.

This package will run on any Atari 800 or Atari 400 with at least 32K bytes of RAM. Since no OSS software uses any routines in any cartridge, you may fully utilize all the RAM in even a 48K byte Atari.

CAUTION: If you have ANY cartridge plugged into your Atari, you will not be able to utilize more than 40K bytes of memory. This a hardware feature of the Atari and can not be changed via software. If you need the power of 48K bytes of RAM, REMOVE ALL CARTRIDGES.

There are, however, some circumstances under which you may need a cartridge for your own development work. OSS CP/A is completely compatible with all known Atari cartridges.

HOW TO USE YOUR OSS PACKAGE

1. Check the contents of your package. If you ordered just BASIC A+, there should be a BASIC A+ manual (an addendum to the Atari Basic manual). If you ordered CP/A, there should be a CP/A manual and an EASMD (Editor/ASseMbler/Debug) manual.

2. There should be a license agreement. FILL THIS OUT NOW AND SEND IT TO US ! Aside from its obvious purpose, the agreement is YOUR ticket to ***SUPPORT***. Yes, we do answer phone questions. Yes, we do respond to bugs. BUT ONLY for those persons who send back their license !!!

3. Turn on your disk drive(s) and screen, leave the Atari computer off. If you purchased CP/A, place the CP/A disk in drive 1. If you purchased only BASIC A+, place an Atari DOS master disk in drive 1. Boot up the system by turning on the computer's power. That's all there is to it! Follow the manual directions for running the program you desire.
   Note: special instructions for running BASIC A+ under Atari DOS are in the beginning of the BASIC A+ manual.

4. We strongly urge you to immediately make a backup copy of your OSS diskette. You may do this using DUPDSK (see CP/A manual). [Or use the Atari DUPLICATE DISK menu DOS command.]

5. Sit back and enjoy the power of a REAL computer system.

OPTIMIZED SYSTEMS SOFTWARE

OS/A+

for the ATARI 800 (R)

COPYRIGHT (C) 1981, OSS

MAY 1981

Version 1.1

NOTICE

OSS reserves the right to make changes or improvements in the product described in this manual at any time and without notice.

This manual is copyrighted and contains proprietary information. All rights are reserved. This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from OSS.

OSS OS/A+ is Copyright (C) 1981, OPTIMIZED SYSTEMS SOFTWARE.

Optimized Systems Software
10379 Lansdale Avenue
Cupertino, CA 95014
Telephone: 408-446-3099

ATARI and ATARI 800 are registered trademarks of ATARI, INC.

TABLE OF CONTENTS

1.0 Introduction

OS/A+ is an abbreviation for Operating System/Advanced. Its purpose is to give its user an advanced type of command control over the software systems in the Atari personal computer. OS/A+ replaces the Atari menu driven DOS command processor with a less restrictive command line processor. The OS/A+ user types command words and parameters rather then invoking menu functions and responding to questions. The OS/A+ command set is easy to learn since most of the commands are the same as the functions desired, such as RENAME or DIRECTORY. The OS/A+ command processor allows for user written commands as well as the "batch" execution of commands from a file. OS/A+ replaces ONLY the MENU command processor of ATARI DOS. The Atari File Manager and Atari OS are used by OS/A+ without modification. This means that disk volumes and associated disk files are fully interchangable between Atari DOS and OS/A+. The only known incompatibility is that OSS BASIC A+ SAVE files are not compatable with ATARI BASIC SAVE files. ATARI BASIC ATASCII source (LIST, ENTER) files will run without modification under OSS BASIC A+. The DOS.SYS file on the OS/A+ disk is the Atari FMS (written by OSS) and OS/A+. OS/A+ disks do not have or need to have DUP.SYS or MEM.SAV.

2.0 Running OS/A+

The OS/A+ Command Processor is invoked in the same manner as the Atari menu command processor. When the OS/A+ disk is booted, OS/A+ is immediatly entered. If the computer has a cartridge that works with the disk, such as BASIC, then the cartridge can be entered via the OS/A+ CARtridge command. Re-entry of OS/A+ from the cartridge is done in the same way as it is to Atari DOS. The BASIC command for this is DOS. Some cartridges do not allow DOS type exits and thus OS/A+ cannot be used with these cartridges.

When OS/A+ is entered it will clear the screen and display:

```
OSS OS/A+   ATARI version 1.0
Copyright (c) 1981  OSS

D1:<cursor>
```

The D1: is the command prompt. It serves two purposes. First it tells the user it is ready to accept a command. Secondly, it is a reminder of the default disk drive. The default drive, in this case, being the familar file spec for drive 1.

3.0 Default Drive and File Specs:

Most OS/A+ commands and parameters deal with files of one sort or another. The Atari Operating System requires files be specified with a filespec of the form:

        <device>: <optional-file-name>

The device for disk files is of the form Dn: where n=1,2,3,4. For example, D1: is the device name of the disk drive with the switch at the rear of the drive set for drive one. Other types of devices are: Printer=P:, Cassette=C:, Screen=S:, etc. The optional-file-name is used for named file accessing devices such as the disk units. To work with the disk file TEST.ORG on disk drive number 1, the operating system requires that the file spec D1:TEST.ORG be used. Having to always specify the D1: can be tedious, especially if most of the user's file work is on a single drive.

The OS/A+ system is designed to prefix all filenames appearing in a OS/A+ command line with the default drive - if a device has not been explictly specified. In the case of D1:TEST.ORG, the user could enter only TEST.ORG for a file name and allow OS/A+ to prefix it with the default drive. Thus TEST.ORG becomes D1:TEST.ORG in the OS/A+ system. If TEST.ORG happened to be on drive two and the default drive was drive one, the user could enter D2:TEST.ORG. OS/A+ would see that the user has explicitily specified a <device> and would thus not append the default drive device to that file name.

If the user needs to work a great deal with files on drive two, he can change the default drive so as to avoid the now necessary D2: prefix typing. Where the system prompts D1:<cursor>, the user can respond with D2:<return> to change the default drive to the D2: device. The next OS/A+ prompt line will show D2:<cursor>. Now files accessed on drive one will require the explict D1: prefix typing, while files on drive will not require prefix typing. Only devices of the form Dn: (where n = 0-9) are allowed as default drives. OS/A+ does not check to insure that the new default drive actually exists. The user's first indication of an invalid default drive will occur when OS/A+ attempts to access a file on the invalid device (via user command). The error message "INVALID DEVICE" will indicate the situation. The user should then set the default device to a valid disk unit. The default device change command is one of the many OS/A+ commands.

4.0 OS/A+ Commands

OS/A+ has three general classes or groups of commands. The classes are intrinsic commands, extrinsic commands, and execute commands. Intrinsic commands are executed by means of resident code in the OS/A+ monitor. Extrinsic commands are executed by means of loading and running programs. The execute subset of commands provide for the batch execution of OS/A+ commands from a file.

4.1 Intrinsic Commands:

The intrinsic commands are executed via code in the OS/A+ monitor. These commands do not require the loading of programs to perform their functions. The following is a summary of the OS/A+ intrinsic commands:

DIRECTORY	- List Directory
PROTECT	- Protect a file (from change or erase)
UNPROTECT	- Unprotect a file
ERASE	- Erase (delete) a file
RENAME	- Renames a file
LOAD	- Load a binary file
SAVE	- Save a binary file
RUN	- Execute a program at some address
CARTRIDGE	- Run Atari cartridge in the A cartridge slot

The default drive change command Dn: is also an intrinsic command. All intrinsic commands may be abreviated with the first three characters. As a matter of fact, OS/A+ only looks at the first three characters while testing for an intrinsic command. Each of the commands will be covered in detail later in this manual; however, to give you a feel of the intrinsic commands let's look at the DIRECTORY command. While looking at these examples, assume the D1: is the default device and has been placed on the screen by OS/A+.

```
D1:DIRECTORY     list entire directory of disk on drive one
D1:DIRECT          "      "      "      "    "    "    "    "
D1:DIRTY           "      "      "      "    "    "    "    "
D1:DIR             "      "      "      "    "    "    "    "
D1:DIR *.*         "      "      "      "    "    "    "    "
D1:DIR D1:         "      "      "      "    "    "    "    "
D1:DIR D1:*.*      "      "      "      "    "    "    "    "
D1 DIR D2:       list entire directory of disk on drive two
D1:DIR D2:*.*      "      "      "      "    "    "    "    "
D1:DIR *.OBJ     list all files with extension .OBJ on drive one
D1:DIR D2:*.ASM  list all files with extension .ASM on drive two
```

4.1.1 PROTECT

The PROtect command is used to protect disk files from being modified or ERAsed. Files that have an asterisk to the left of the file name in the directory listing are protected files.

```
                PROtect         file spec
```

4.1.2 UNPROTECT

The specified files (PROtected or not) are unprotected. The unprotected files can now be modified or ERAsed.

```
                UNProtect       filespec
```

4.1.3 ERASE

The specified files are removed from the disk and the disk sectors occupied by the files become free to be used again by other files.

```
                ERAse           filespec
```

4.1.4 RENAME

Rename a file or files.

```
                REName  old-filespec new-filename
                REName  old-filespec,new-filename
```

The old-filespec specifies the file(s) that are to be renamed to new-filename. Either blanks or a comma may be used to separate the filenames. WARNING! Be careful using wild card renames. You can get more than one file with the same name and never be able to access the second same-named file. (See Appendix B)

4.1.5 SAVE

The SAVE command is used to write (copy) a section of RAM to a disk file. The area of RAM to be written is given as the two hexidecimal parameters start address (sa) and end address (ea).

```
                SAVe    filespec   sa   ea
        Example:
                SAV     TEST.OBJ  4000  4FFF
```

The sa and ea parameters are separated by blanks or a comma. The ea must be greater than or equal to sa.

OS/A+ will write a six byte header to the file before writing the data. This header consists of the binary file indicator, the sa, and the ea.

```
        Binary File Indicator (2 Bytes)                 $FFFF
        sa      (2 Bytes) least significant byte first  ($0040)
        ea      (2 Bytes) least singificant byte first  ($FF4F)
        data    (ea - sa) + 1 bytes
```

The saved file may be later loaded with the LOAD command.

4.1.6 LOAD COMMAND

The Load command is used to load binary files into RAM. The specified file is checked for the Binary File Indicator ($FFFF as the first two file bytes). If the indicator is present the next four bytes are assumed as the sa and ea of the data. OS/A+ will then copy the next ea-sa + 1 bytes of data from the file to RAM starting at ea. OS/A+ will also place sa in the OS/A+ RUNLOC cell. If OS/A+ does not recieve an end-of-file after loading the data it will assume another code segment is present. Each following code segment is like the first except that the $FFFF header is not present. OS/A+ will only place the sa from the first segment in RUNLOC.

```
        LOAD    filespec
```

OS/A+ also supports the Atari load and go scheme. If the load file has the proper INIT and RUN vectors, OS/A+ will perform the INIT and RUN functions (see Atari DOS 2.0 manual for details).

The OSS assembler (EASMD.COM) creates object files that are load-able as LOAD files.

4.1.7 RUN COMMAND

The Run command causes OS/A+ to call (JSR) a routine in RAM.

```
        RUN     optional-hex-address
```

If the optional hex adress is specified then OS/A+ will place the given hex adress in the OS/A+ RUNLOC and then call the routine via the address in RUNLOC. If the hex address is not specified then OS/A+ will call the address that is currently in RUNLOC. The address in RUNLOC may have been set by a previous LOAD or RUN command or via the execution of an extrinsic command.

4.1.8 CARTRIDGE

The parameterless CARtridge command causes OS/A+ to transfer control to the CARTRIDGE in the A cartridge slot. There are two ways OS/A+ will call a cartridge, either with a warm start or a with a cold start. The cartridge cold start tells the cartridge to reinitialize its memory and start cold. The warm start tells the cartridge to retain its memory as it was upon exit (via DOS command or RESET). The first OS/A+ cartridge call will always be a cold start. Sub-sequent OS/A+ calls will be warm starts unless OS/A+ has executed a memory changing command. Memory changing commands are LOAD and extrensic commands.

4.1.8.1 RESET

If a cartridge has control and the RESET key is pressed, OS/A+ will be entered. If it is desired to re-enter the cartridge, simply enter the CAR command.

4.2 Extrinsic Commands:

The extrinsic commands are programs that are run by OS/A+. Any program file of the load file format and containing the .COM extension may be used as a OS/A+ extrinsic command. The OS/A+ COPY command is one such extrinsic command. If you DIR the OS/A+ diskette, you will see a file named COPY.COM . The program in the COPY.COM file is what is executed when the COPY command is entered. Assuming that D1: is the default device, the COPY command would look like:

```
D1:COPY <from-file-name> <to-file-name>

                    or

D1:COPY TEST.OBJ D2:TEST.OBJ
```

to copy TEST.OBJ from drive one to drive two.

Whenever any command is given to OS/A+ it first compares the command entered (first three characters only) to its intrinsic command list. If the command is not in the intrinsic list, it is assumed to be extrinsic. OS/A+ will process the extrinsic command by:

1) Prefix the command with the default device (if a device is not specified).
2) Attach the .COM extension to the command.
3) Open the generated file spec for input.
4) Test file for proper Load file format (see 4.1.6).
5) Load and execute the program.

The COPY command illustrated will execute only if the file COPY.COM exists on drive one and is of the Load file format. If any element of the procedure fails various error messages will result. Step 1 of the procedure implies that a device may be specified. If the default device is drive two and the COPY.COM program is on drive one, our example COPY would look like:

```
D2:D1:COPY D1:TEST.OBJ TEST.OBJ
```

which again copies TEST.OBJ from device one to device two. Never explictly specify the .COM extension as part of the command. The command COPY.COM will result in a file spec of D1:COPY.COM.COM, which is invalid. If the file is not of the proper format, the error message ADR RANGE ERROR will most likely appear.

The extrinsic command class contains an infinite number of commands. Some extrinsic commands (such as COPY) are supplied by OSS. Most extrinsic commands are user written. If you are intrested in writing your own extrinsic commands, see Appendix B.

4.3 Batch Processing:

The OS/A+ execute feature allows the user to execute one or many OS/A+ commands with a single command. Let's suppose that you wrote a set of BASIC programs that must be run in sequence. You could issue the OS/A+ extrinsic BASIC command (execute BASIC.COM), then from BASIC run each program one at a time. If the running time of the BASIC programs was very long you could sit at the key board for hours just to type RUN program-name every once in awhile. OS/A+ allows you to create and execute an EXECUTE file which contains one or many OS/A+ commands. You would then enter one command that would free you from the keyboard for more important (or fun) things.

4.3.1 Executing EXECUTE files:

Any text file with the filename extension .EXC can be used as a OS/A+ execute file. The execution of the file is invoked much like the extrinsic commands, except the command is preceeded with an AT "@" symbol. To execute the EXECUTE file DEMO.EXC on the D1: default device

```
        D1:@DEMO
```

OS/A+ will build the file spec D1:DEMO.EXC and read that file line by line executing the OS/A+ commands just as if they were being entered from the keyboard.

Humans are not quite perfect in the eyes of computers and sometimes make mistakes. OS/A+ commands specified in error will generate error messages. If OS/A+ discovers an error while executing an EXECUTE file, it will print the error message as usual and STOP executing the EXECUTE file.

Execution of an exexute file will also stop after the CARTRIDGE command is executed.

4.3.2 Execute File Format

An execute file is simply a text file. Each line of the text file will become a OS/A+ command when executed. The three basic rules of text file LINES are that:

1) they must contain valid OS/A+ commands,
2) they must be less than 60 characters in length
3) they must end in a carriage return (ATASCII $9B).

OS/A+ allows the commands in an execute file to be preceeded by numbers and blanks. This feature allows the command lines to be numbered for readability and thus document their purposes.
The execute file line:

```
        LOAD    OBJ.TEST <return>
```

and the line:

```
        100 LOAD    OBJ.TEST <return>
```

are the same to OS/A+ . OS/A+ scans the line for the first non-numericl, non-blank character before starting to scan the command word. The EDITOR of the OSS EASMD program can be used to create and modify execute files.

4.3.3 Execute Intrinsic Commands

OS/A+ has four special intrinsic commands designed for use with execute files. These commands are:

REMARK	Remark or comment (does nothing)
SCREEN	Allows execute commands to echo to the screen. This is the default mode.
NOSCREEN	Turn off Echo of execute file command lines to the screen.
END	Stop executing the execute file and return OS/A+ to keyboard entry mode.

4.3.3.1 REMARK

The REMARK command provides a means of commenting and documenting an execute file. OS/A+ will ignore all characters on the REMARK command line and proceed to the next command file line. The command file:

```
        100     REM     BACKUP DAILY TRANSACTION FILE
        110     BASIC   TFBACKUP.BAS
        120     REM     PRINT TRANSACTION REPORTS
        130     BASIC   TREPORTS.BAS
        140     END
```

uses OSS BASIC A+ to work with some transaction BASIC programs. The REMARK statements explain the process. LINE 110 will load and execute the OSS BASIC A+ (BASIC.COM on default drive) which will in turn run the TFBACKUP.BAS BASIC A+ program (SAVED on default drive).

4.3.3.2 SCREEN/NOSCREEN

OS/A+ normally echos the command lines to the screen so that it appears as if they were typed in as keyboard commands. The NOSCREEN command can be used to prevent the echo process. After NOSCREEN has been executed, no further EXECUTE file command will appear on the screen until:

1) the SCREEN command is executed or
2) the EXECUTE file stops for some reason.

4.3.3.3 END

The END command provides a documentable END to the execution of of an execute file. It may also be used to stop the file's execution before the actual end-of-file.

4.3.4 PROGRAM CONTROLLED EXECUTE FILE STOP

It is sometimes desirable for a program in a chain of executing programs to stop the execute process. The usual reason for this is that the program has detected an error severe enough to invalidate the processes performed by the following program(s). The continued execution of the execute files is provided for by a single byte flag within OS/A+. If a program sets this byte to zero, then upon returning to OS/A+ via DOS or CP (BASIC statements) the execute file execution will immediately stop. The execute flag is located 11 bytes from the start of OS/A+. The address of OS/A+ is pointed to by memory location 10 ($0A). The following BASIC A+ program segment will turn off the execute file and return to OS/A+.

```
        1000    CPADR = DPEEK(10)
        1010    EXCFLG = CPADR + 11
        1020    POKE  EXCFLG,0
        1030    DOS
```

4.3.5 STARTUP.EXC

The execute filename STARTUP.EXC has special meaning in the OS/A+ system. When the system is first booted (power up), OS/A+ will search the directory of the booted disk volume for a file named STARTUP.EXC. If STARTUP.EXC is on the booted volume, OS/A+ will execute that file before requesting keyboard commands.

5.0 SYSTEM INTERFACE GUIDE

The writer of assembly language code will most likely need to interface with the Atari Operating System (OS). If the assembly code is to become an extrensic command, there may be a need to interface to OS/A+ . The Atari OS manual provides a proliferation of information about the Atari Operating System which will not be covered here.

5.1 SYSEQU.ASM

Every OS/A+ master disk contains an assembler source file, SYSEQU.ASM, that has various commonly used Atari OS and OS/A+ system equates. This file may be included in an assembly languae program via the OSS EASMD include function (.INCLUDE #D1:SYSEQU.ASM)

5.2 OS/A+ MEMORY LOCATION

OS/A+ is designed to be placed just after the Atari File Manager. Since the acatual location of OS/A+ may vary with different versions of a file manager, a fixed location has been assigned to point to OS/A+. The location (CPALOC=$0A) is the same one Atari uses to point to DUP. All Atari programs that use a DOS exit vector through $0A.

5.3 EXECUTE PARAMETERS

The OS/A+ execute flag is located CPEXFL ($0B) from the start of OS/A+. The CPALOC may be used as an indirect pointer to access the execute flag.

```
        LDY     #CPEXFL         ;GET DISPL TO FLAG
        LDA     (CPALOC),Y      ;LOAD FLAG
```

The Execute Flag has four bits that control the execute process:

EXCYES	$80	If one, an execute is in progress
EXCSCR	$40	If one, do not echo execute input to screen
EXCNEW	$10	If one, a new execute is starting. Tells OS/A+ to start with first line of the file
EXCSUP	$20	If one, a cold start execute is starting. Used to avoid file-not-found error if STARTUP.EXC is not on boot disk.

OS/A+ performs the execute function by opening the file, POINTing to the next line, reading the next line, NOTE the new next line and closing the file. To perform these functions, OS/A+ must save the execute file name and the three byte NOTE values. The filename is saved at CPEXFN ($0C) into OS/A+. The three NOTE volues are saved at CPEXNP($1C) into OS/A+. (CPEXNP+0=ICAUX5; CPEXNP+1=ICAUX4; CPEXNP+2=ICAUX3). By changing the various execute control para- meters, a programmer can cause recursion and/or changing of ex- ecute files.

5.4 DEFAULT DRIVE

The OS/A+ default drive file spec is located at CPDFDV ($07) into OS/A+. The Default Drive here is ATASCII Dn: where 'n' is the ATASCII default drive number.

5.5 EXTRINSIC PARAMETERS

The extrinsic commands may be called with parameters typed on the command line. The OSS command

```
        D1:COPY FROMFILE  D2:TOFILE
```

is an example of this. The entire parameter line is saved in the OS/A+ input buffer located at CPCMDB ($40) bytes into OS/A+ and is available to the user. Since most command parameters are file names, OS/A+ provides a means of extracting these parameters as filenames. The routine that performs this service begins at CPGNFN ($03) bytes into OS/A+ . The routine will get the next parameter and move it to the filename buffer at CPFNAM ($21) bytes in OS/A+. If the parameter does not contain a device prefix, then OS/A+ will prefix the parameter with the default drive prefix. The first time COPY calls CPGNFN the file spec "D1:FROMFILE" is placed at CPFNAM. The second time COPY calls CPGNFN the file spec "D2:TOFILE" is placed in CPFNAM. If CPGNFN were to be called again then the default filespec would be set into CPFNAM at each call. To detect the end of parameter condition, the user may check the CPBUFP ($0A into OS/A+) cell. If CPBUFP does not change after a CPGNFN call then there are no more parameters. The filename buffer is always padded to 16 bytes with ATASCII EOL ($9B) characters. The following example sets up a vector for calling the get-filename routine.

```
        CLC
        LDA     CPALOC          ;ADD CPGNFN
        ADC     #CPGNFN         ;TO CPALOC VALUE
        STA     GETFN+1         ;AND PLACE IN
        LDA     CPALOC+1        ;ADDRESS FIELD
        ADC     #0              ;OF JUMP
        STA     GETFN+2         ;INSTRUCTION
        ...
        ...
GETFN   JMP     0
```

The following routine then gets the next file name to CPFNAM.

```
        LDY     #CPBUFP         ;SAVE CPBUFP
        LDA     (CPALOC),Y      ;VALUE
        PHA
        JSR     GETFN           ;GET NEXT FILE PARM
        LDY     #CPBUFP
        PLA                     ;TEST FOR NO NEXT
        CMP     (CPALOC),Y      ;PARM
        BEQ     NONEXT          ;BR IF NO NEXTPARM

        LDY     #CPFNAM         ;ELSE GET FILE
        LDA     (CPALOC),Y      ;NAME FROM BUFFER
        ...
        ...
```

5.6 RUNLOC

The OS/A+ RUNLOC ($3D into OS/A+) is used as the OS/A+ vector to routines with the RUN, LOAD and extrinsic commands. An application that allows exits to OS/A+ can change RUNLOC to provide a warmstart re-entry to the application (if the user enters RUN with no para-lmeters). If the application is not reusable and wishes to forbid re-entry, the high order byte of RUNLOC ($3E into OS/A+) should be set to zero.

```
        LDY      #RUNLOC+1         ;FORBID RE-ENTRY
        LDA      #0                ;TO ME
        STA      (CPALOC),Y
```

5.7 EXITS

OS/A+ calls all programs (except cartridges) via the 6502 JSR instruction. A called OS/A+ program may return back to OS/A+ via the RTS instruction or via a JMP (CPALOC). If the JMP (CPALOC) is used, OS/A+ will close IOCB zero and re-open it to the E: device (which clears the screen). Either the JMP (CPALOC) or the RTS return will cause OS/A+ close IOCBs one through seven.

APPENDIX A

OSS EXTRINSIC COMMANDS

A-1. COPY

```
COPY            from-file-spec          to-file-spec
```

The copy command will copy one file, the from-file-spec, to the to-file-spec. COPY does NOT allow a change of diskettes while copying: both source and destination must be mounted when the COPY command pauses after loading.

A-2. INIT

```
INIT            (no parameters)
```

The INIT command is used to:

1) FORMAT A DISK (OR)
2) FORMAT A DISK AND WRITE DOS.SYS (OR)
3) WRITE DOS.SYS

INIT is menu driven and will give you the oportunity to change disks before executing. DOS.SYS is the OS/A+ boot loader and is required to make the OS/A+ boot from the disk.

A-3. DUPDSK

```
DUPDSK          (no parameters)
```

DUPDSK is used to duplicate an entire disk. It can be used with a single drive. It will format the destination disk for you if you choose to do so. When you are finished with DUPDSK, you must insert a system disk (a disk with DOS.SYS) because DUPDSK will (purposefully) re-boot the system.

APPENDIX B

FMS POKES

There are several 'pokes' that can be done to the Atari FMS that comes with OS/A+. These pokes are used to change certain FMS parameters. The changes can be made permanent by using INIT to write (re-write) DOS.SYS after the poke is done.

B.1 NUMBER OF FILE BUFFERS

The FMS allocates space for file buffers. One file buffer is required for each open disk file. The number of file buffers allocated is the number of files that can be open at the same time. The OS/A+ system is shipped with three (3) file buffers allocated. Three is the recommended minimum. The nul). can be changed by poking a new value at $709 (decimal 1801). The maximum usable value is 7 (any value greater than 7 wastes space). The changed value does not go into effect until the system is booted. This means that you MUST rewrite (or write) DOS.SYS on the disk and then reboot the disk.

B.2 NUMBER OF DRIVES

The FMS drive byte is used to tell FMS how many drives you have on your system. The FMS is shipped with the drive byte set for two drives (D1: and D2:). Each drive allocated via this value consumes an additional 128 bytes of RAM for a drive buffer. If you have more or less than 2 drives, you will probably want to change this value. This value, like the value for the number of file buffers, does not go into effect until the system is booted. The drive byte is located at location $70A (decimal 1802). The appropriate values are:

```
        1 drive = $01   (decimal 1)
        2 drives= $03   (decimal 3)
        3 drives= $07   (decimal 7)
        4 drives= $0F   (decimal 15)
```

B.3 FAST DISK WRITE

The Atari disk can be commanded to write sectors with verify or without verify. The write WITH verify causes the drive to read each sector immediately after writing it; this process assures that data on the disk is valid but causes write operations to run about half as fast as they could run if the write was done without verify. Depending upon your patience, the importance of your data, and your objective view of the reliability of your drives and disks, you can choose either write-with-verify (slow) or write-without-verify (fast). The FMS location to change is $779 (1913 decimal). The write-with-verify value is $57 (87 decimal). The write-without-verify (default, faster write) is $50 (80 decimal).

B.4 RENAME WOES

If you happen to rename several files (for example, with the use of a wild card rename) in such a way that you end up with two files of the same name, you need to remember this section. The problem: after ending up with two files of the same name, all further accesses to that filename will access only the first file that appears in the directory. Even a wildcard rename will not work: both files are again renamed to the same name.

The solution: You may patch FMS to alter the RENAME code. The patch causes RENAME to change only the first file in the directory that matches the given filespec, not all matching filenames. To make the patch, POKE a zero ($00) to location $C2E (decimal 3118). To restore RENAME to normal functioning, poke $B8 (decimal 184) to the same location.

CAUTION: because this patch affects ALL renames, and will not now allow multiple RENAMEs, etc., it is probably not advisable to make the patch permanent.