I. FORMAT DISK
Follow instructions to format a disk.

J. DUPLICATE DISK
Follow the instructions to copy an entire disk from drive to drive or, by copying the contents of the disk into memory, to the same drive.

K. BINARY SAVE
FNM,SSSS,EEEE (RETURN) saves the contents of a specified area of memory, usually a machine code program. SSSS is the start address and EEEE the end address in four-digit hexadecimal.

L. BINARY LOAD
FNM (RETURN) loads a file saved with BINARY SAVE back into the locations from which it was stored.

M. RUN AT ADDRESS
On display prompt enter address at which to execute a BINARY LOADed file in four-digit hexadecimal. (RETURN) executes the program.

N. CREATE MEM. SAV
Follow the instructions to create a file called MEM.SAV. When the DOS menu is called, DOS automatically saves the contents of memory that would be overwritten by the menu and reloads it when RUN CARTRIDGE is selected.

O. DUPLICATE FILE
FNM (RETURN) then follow the instructions to copy files from one disk to another on a single drive. Wildcards are permissible.

The other commands controlling program files and data files are:
SAVE LOAD LIST ENTER RUN OPEN# CLOSE# PRINT#INPUT# NOTE# POINT# PUT# GET# STATUS# X10

Program Files:

SAVE FSP
This writes the specified program to disk in a 'tokenised' form.

LOAD FSP
This reads the specified tokenised program into user memory from the bottom of memory upwards.

LIST FSP,LN1,LN2
This stores a BASIC program IN ATASCII form (Atari's version of standard ASCII). Specifying FSP alone stores the whole program. LN1 and LN2 represent line numbers and can be used to specify the start and end line numbers of a portion of a program to be stored. Used in conjunction with ENTER to merge programs.

ENTER FSP
This reads the specified file, previously stored using LIST, into user memory and merges it with a program already held in memory. If there are duplicate line numbers, the lines contained in the newly read file take the place of those in memory.

RUN FSP
This reads the specified tokenised program into memory and automatically runs it.

Data Files:

OPEN#
This command controls access to special communication channels called I/O Control Blocks (IOCB) and links them to an appropriate device, in this case a disk drive and FSP, under specified conditions, as follows:

OPEN#IOCB,AC1,AC2,FSP

where IOCB is the I/O channel (1-5); AC1 is the auxilliary code 1 (specifies type of I/O operation according to a table in DOS manual); and AC2 is always 0 for disk.

The following commands relate to IOCBs OPENed in the manner previously explained.

CLOSE# IOCB
This releases the specified IOCB from the I/O conditions set above. CLOSEd IOCBs cannot be accessed.

PRINT#
This writes numeric (X,Y) or string (A$) data to the specified IOCB, for example:

PRINT#,X,Y or PRINT#, IOCB,A$

INPUT#
This reads numeric or string data from the specified IOCB, for example:

INPUT#IOCB,X,Y or INPUT#IOCB,A$

NOTE#
This is set before storing data with PRINT#. It provides a record of the sector number and byte number where the next byte is to be stored on disk. The resulting list can be stored as a table in another file to enable data to be accessed randomly, a byte at a time if required, by POINT#. For example:

NOTE#IOCB,A,B

where S is the sector number (1-719) and B is the byte number (0-124).

POINT#
This reads the specified byte of data, previously stored using NOTE#, into user memory thus:

POINT#IOCB,A,B.

PUT#
This writes a single byte to the specified IOCB. For example:

PUT#IOCB,N

where N = 1 to 255.

GET#
This reads a single byte stored by PUT# with:

GET#IOCB,N

STATUS#IOCB, ERROR
This gives a specified variable, in this case ERROR, the current error number for the last I/O operation on the IOCB. The resulting number can then be checked against the table of errors in the Atari DOS manual.

X10 CN,#IOCB,AC1,AC2,FSB
This is used to duplicate some of the functions of the DOS menu by use of the Command Number (CN) relating to the required function. The Atari DOS manual contains a list of Command Numbers and their associated functions.

PRICE DIFFERENTIAL

Since 1973, the cost of a computer has dropped 500 fold per bit of memory. If the Rolls Royce car had been developed at the same rate as the computer — while achieving a similar level of price reduction — today's vehicle would cost in the region of £1.50 and average about one million miles to the gallon