DAI PERSONAL COMPUTER DOS - Preliminary Engineering Specifications

Power-on Sequence

i. Connect the disk system via the flat cable to the DCE-BUS connector at the back of the Personal Computer.

ii. Ensure that the doors of both diskette drives are open and then turn on power to the disk system.

iii. Insert the system diskette into drive 0 and close the door.

iv. Turn on power to the Personal Computer (press RESET if necessary).

The following sequence of steps will then be executed automatically:

1. The bootstrap file $DKBOOTS on the system diskette engaged in drive 0 will be loaded into the stack RAM memory from address 0F800H upwards. The bootstrap routine will occupy approximately 200 bytes. Control will then be passed to its entry point at address 0F800H.

   This bootstrap file has a special format and contains pure binary data, without the usual file-definition parameters which specify the start, end and entry addresses. This bootstrap routine is used to load and execute the diskette file $MSTRDOS, which normally contains DOS.

2. The file $MSTRDOS on the system diskette engaged in drive 0 will then be loaded and executed. This is a normal binary file which is approximately 5K bytes in length, and contains the full DOS. The binary data on this file is preceded by the three usual file-definition parameters indicating the start, end and entry addresses.

   Note: This file can also be a user program in the correct binary format. In this case, it will be automatically loaded and executed on power-on or system reset.

3. The binary file $USER. BIN will then be searched for on the system diskette.

   If no such file is found, execution will proceed to step 4.

   If the file is found, it will be opened for input and loaded into memory as specified in the file-definition parameters contained on it. Any loading error will cause execution to return to the normal BASIC initialization sequence (i. e. wait for any key input, and then display the BASIC sign-on message and prompt character).

   If the entry address specified in the file-definition parameters is equal to 0FFFFH (i. e. no entry address), execution will proceed to step 4.

   If a valid entry address is specified in the file-definition parameters (i. e. any value except 0FFFFH), the loaded program will be called as a machine code subroutine with that entry address. Upon executing the Return instruction code at the end of the routine, the contents of the zero flag in the CPU flag register will be tested. If this zero flag is set (=1), execution will proceed to step 4; if it is clear (=0), execution will return to the normal BASIC initialization sequence (i. e. wait for any key input, and then display the BASIC sign-on message and prompt character). The zero flag should therefore be set or cleared by the program stored in $USER. BIN to indicate the presence or absence of a BASIC program for automatic loading followed by running (step 4).

4. The BASIC program file $USER. BAS will then be searched for on the system diskette.

   If no such file is found, execution will return to the normal BASIC initialization sequence (wait for any key input etc.).

   If the file is found, it will be opened for input and loaded into memory as a BASIC program (semi-compiled code) plus symbol table, and then executed (i. e. RUN).

File Formats and Filereferences

Filereferences are similar to those in the industrial system (see Section 6.5.3 in the Software Handbook). However, BASIC and Utility commands which write files to diskette automatically generate implicit extension names.

Implicit extensions are given below:

filename.BAS — A BASIC program file (containing semi-compiled code), with the program code followed by symbol table. Generated by the BASIC command SAVE; can be read back via the BASIC command LOAD.

filename.BIN — A binary file preceded by file-definition parameters (see Section 6.6.36 in the Software Handbook). Each file-definition parameter is a 16-bit address in low-high order. Generated by Utility command W; can be read back via the Utility command R. If the execution address in the file-definition parameters is equal to 0FFFFH, this will be interpreted as no-entry. The Utility command W will produce execution address = 0FFFFH; if a valid entry address is desired, the DOS command DSAVE can be used. The Utility command R will read such a file and automatically enter it at the execution address if it is not equal to 0FFFFH (only if no address offset is specified in the R command). The DOS command DLOAD will read such a file, but will not transfer control to it.

The following files are generated via the BASIC command SAVEA. They can be read back via the BASIC command LOADA.

filename. FPT	A floating-point array file.
filename. INT	An integer array file.
filename. STR	A string array file.

Since the above extension names are automatically generated by the relevant Utility or BASIC commands, filereferences specified in such commands must not include these extensions. Subsequent Utility or BASIC commands used to load such files must also use filereferences excluding the extensions. However, subsequent DOS commands accessing such files must include the correct extension name in the filereference.

Example:

'SAVE "PROGRAM:1"	This BASIC command will copy the BASIC program currently in memory onto the diskette in drive 1, as file PROGRAM. BAS.
LOAD "PROGRAM:1"	This BASIC command will load the BASIC program file PROGRAM. BAS from the diskette in drive 1 into system memory.

If, for example, this file is to be deleted via the DOS command DELETE, the filereference must include the extension, as shown below:

```
DELETE PROGRAM. BAS:1
```

DOS Commands via the Keyboard

DOS commands can be entered via the keyboard, in response to the usual BASIC prompt. They cannot be entered from the Utility or from the Editor.

Such a DOS command must be the first one on that line. Anything that follows a DOS command and its parameters on the same line will be ignored.

Including DOS Commands in BASIC Programs

Any DOS command and its parameters can be included in a BASIC program by containing it in a PRINT statement. Any single or any group of such DOS commands contained in PRINT statements must be preceded by the BASIC command:

```
POKE #131,3
```

and followed by the BASIC command:

```
POKE #131,0
```

The total number of characters generated by such a DOS command contained in a PRINT statement (upto the terminating carriage-return) must be less than or equal to 48.

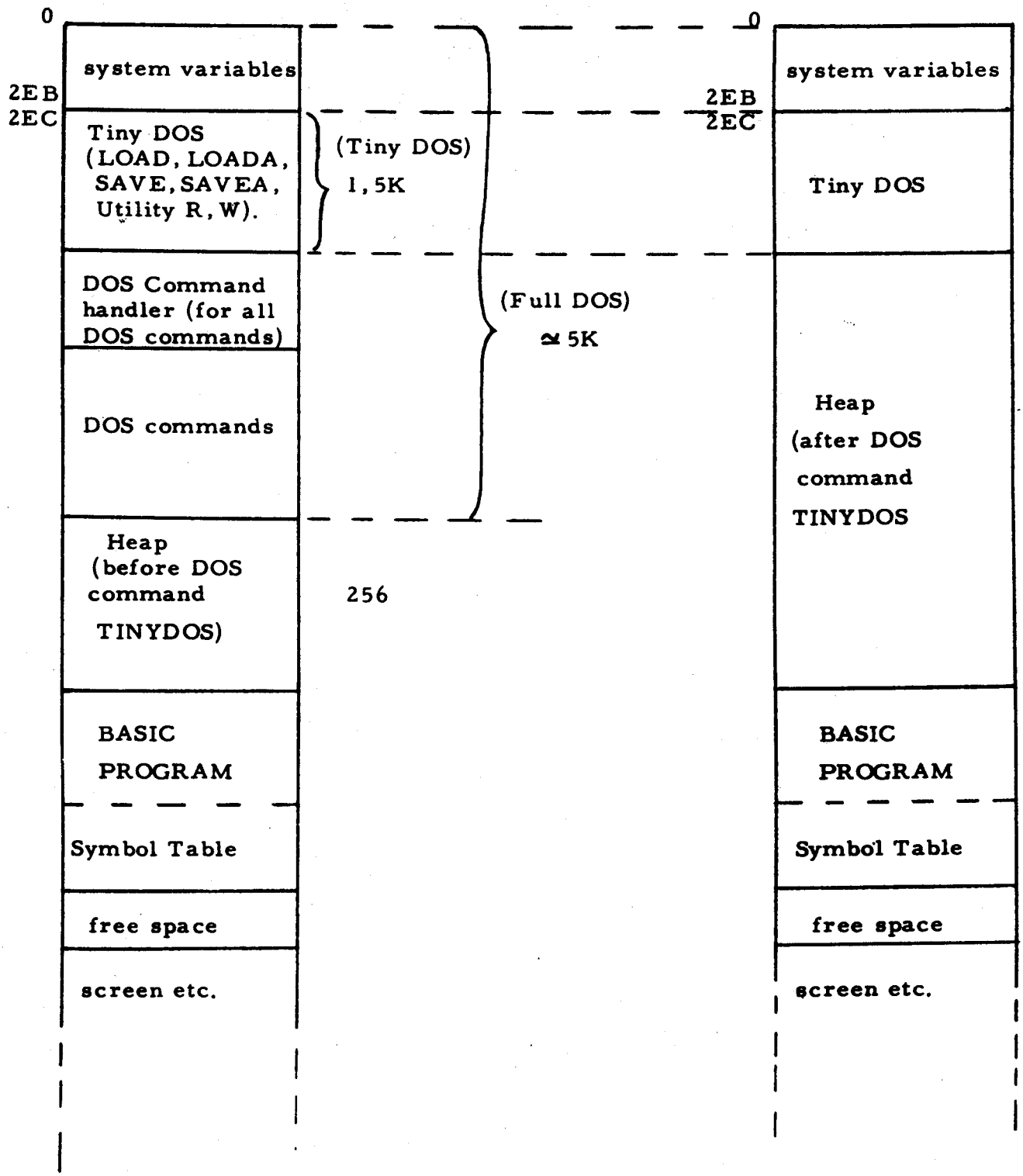
The command "POKE #131,3" will channel all subsequent characters output via the BASIC statement PRINT to the DOS command handler. They will be re-channelled to the printer on execution of the BASIC command "POKE #131,0".

Example:

```
INPUT "FILENAME=";N$
POKE #131,3
PRINT "CREATE " + N$ + ":1"
PRINT "COPY SOURCE:1 " + N$ + ":1"
PRINT "PROTECT " + N$ + ":1"
POKE #131,0
```

voir DAInamic N°14/47

RAM Memory Map

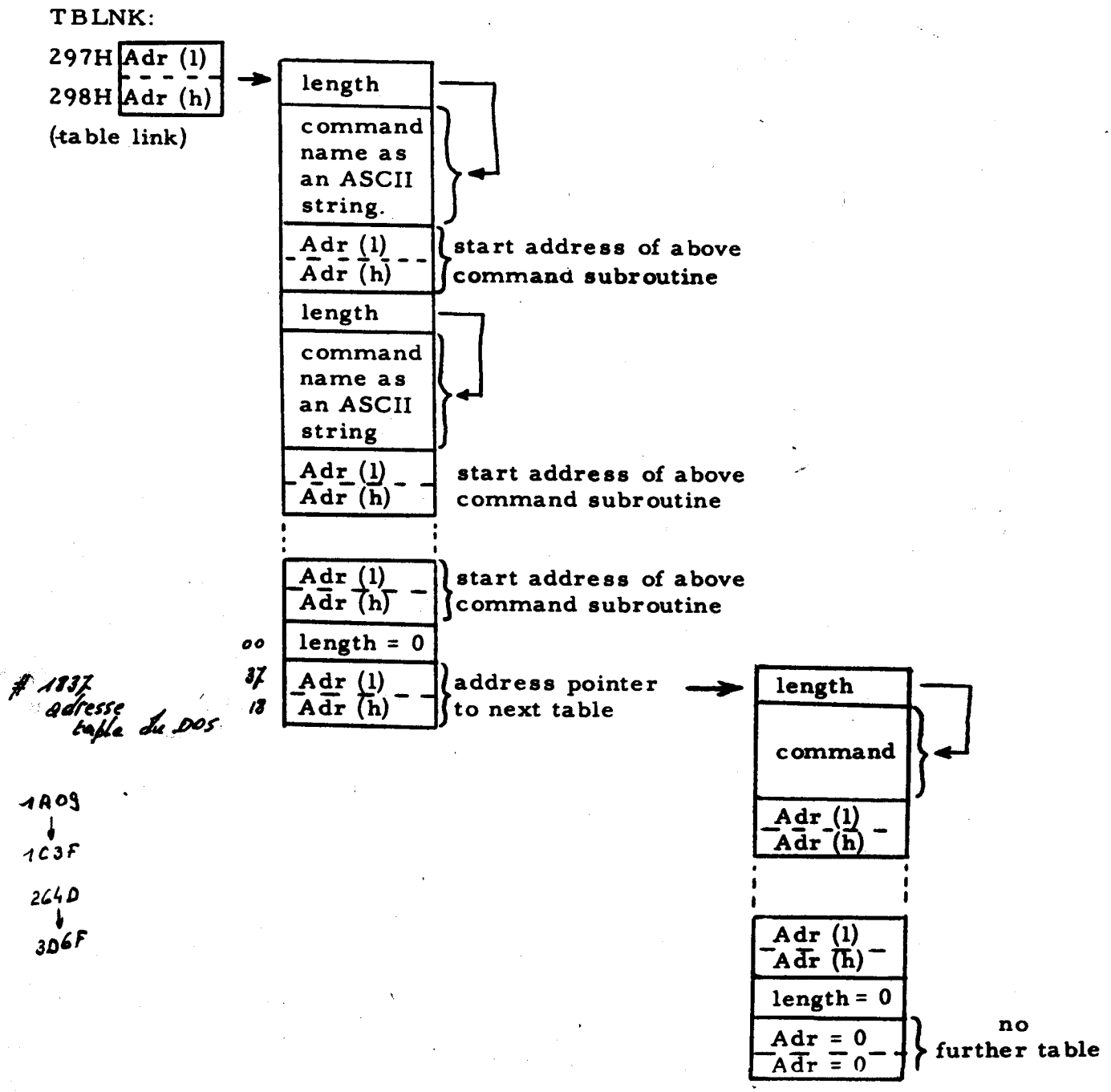


The power-on reset bootstrap sequence loads the full DOS (about 5K) into memory. At any subsequent time, this full DOS can be reduced to tiny DOS via the DOS command TINYDOS. Tiny DOS represents a cassette replacement, and only supports the BASIC commands LOAD, LOADA, SAVE, SAVEA and the Utility commands R,W. All DOS commands are processed via the DOS command handler. The DOS command TINYDOS will delete the DOS command area and the DOS command handler, by re-allocating these areas to the heap. The heap can then be reduced and the BASIC program area corresponding increased via the BASIC command CLEAR.

For example, the binary program file $USER.BIN loaded during the power-on reset bootstrap procedure can call the DOS command routine TINYDOS to reduce the full DOS to tiny DOS.

DOS Command Table Structure

The structure of the DOS command table need not be known for normal use of DOS commands. However, an understanding of this table structure enables the user to add machine code sequences which can be executed simply as new DOS-type command codes (after full DOS has been loaded). Whenever a DOS command is encountered (via keyboard or via the PRINT statement) control is passed to the DOS command handler, which will scan the DOS command table for the presence of that command.



The scheme above illustrates visually the structure of the DOS command table. The address locations 297H and 298H contain the pointer address (low, high order) to the start of the command table. The first entry in the command table contains a binary number specifying the number of ASCII characters which follow immediately and constitute the first command code name. The last character of the command code name is followed by the start address of a machine-code subroutine which must be called to execute that command. This address will be followed by the length of the next command code etc. If no more codes follow, the length will be set to zero. The two bytes following the last length parameter (zero) indicates the presence or absence of an address link to the next table. A zero address indicates no further table link. A no-zero address value will be interpreted as a pointer address to the next table (low, high order).

New user-defined commands can be linked into the DOS command table in the following way:

1. Set up the user command table in correct format. The last length parameter which is zero must be followed by an address pointer to the start of the normal DOS command table.

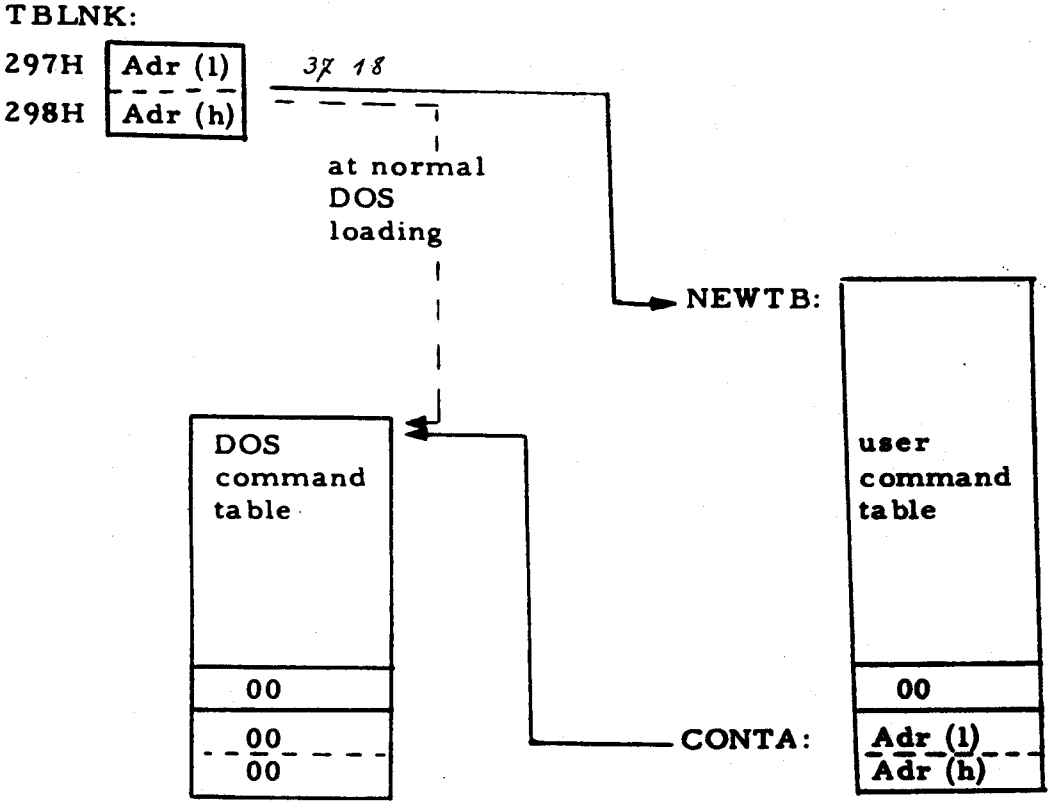
   Example:

   Set the label corresponding to the start of the user-defined command table be NEWTB, and let the label corresponding to the address pointer at the end of it (pointing to the DOS command table) be CONTA.

   The user table can be linked to the DOS command table in the following way:

```
INIT:
    PUSH    H
    LHLD    TBLNK
    SHLD    CONTA
    LXI     H,NEWTB
    SHLD    TBLNK
    POP     H
    RET
```

Such a sequence must be executed once only. If not, the link to the DOS command table will be lost. The locations TBLNK and TBLNK+1 will be initialized to the start of the DOS command table when DOS is loaded into memory.



DOS Commands

These are mostly similar to the commands described in the Software Handbook. Some have been slightly modified.

DIR n — Display directory of diskette in drive 'n'. Deleted files, track and sector numbers are not displayed.

RESETD — Reset diskette controller.

DELETE filereference1 filereference2 . . .
Delete one or more diskette files. This command and parameters can be upto 4 lines in length.

COPY filereference (with diskunit = 0, or omitted)
Copy file from diskette in drive 0 to the diskette in drive 1..

COPY filereference (with diskunit = 1)
Copy file from diskette in drive 1 to the diskette in drive 0.

COPY sourcefileref destinationfileref
Copy the specified source file to the specified destination file.

Note: All three formats of the COPY command automatically open and close the destination file. If the destination file does not already exist, it is created to be 1 sector longer in length than the source file. This feature can be used to extend the length of an existing file easily. If the same file length is desired, the destination file should first be created via the CREATE command.

TINYDOS — See the section "RAM Memory Map"

LOCAL — Closes cassette motor control contacts to enable use of cassette unit in local mode.

REMOTE — Opens cassette motor control contacts to enable automatic motor control.

AUTO nnn — Enables automatic line numbering for keyboard entry of BASIC programs, starting at line number 'nnn' (must be non-zero), with steps of 10.

AUTO carriage-return

Disables the above automatic line numbering and resumes manual line numbering. Before the command is typed, the line number which automatically appears at the start of the line must be deleted.

BACKUP n
0,1

Backtup (total copy) from diskette in drive 'n' to the diskette in drive '1-n'. Automatically formats the destination diskette, but does no checking.

COMPACT n
0,1

Similar to BACKUP except that deleted files are not copied.

CREATE

See section 6. 6. 14 of Software Handbook.

Note: DOS commands are always operative with diskettes. The ASSIGN commands below are for the BASIC commands LOAD, LOADA, SAVE, SAVEA and the Utility commands R and W. For example, they can be used for copying a program or an array from diskette to cassette via memory, and vice versa.

ASSIGN DISK

Assigns diskette for read and write operations associated with LOAD, LOADA, SAVE, SAVEA, R,W commands.

ASSIGN CASSETTE

Assigns cassette for read and write operations associated with LOAD, LOADA, SAVE, SAVEA, R,W commands. Normal default is to cassette unit 0 after reset; if this has been changed subsequently, the above command will re-instate the last selection.

ASSIGN CASSETTE n

Assigns cassette unit n(=0 or 1) for read and write operations associated with LOAD, LOADA, SAVE, SAVEA, R,W commands.

ASSIGN TO DISK
CASSETTE
CASSETTE n (n = 0 or 1)

Similar to above ASSIGN commands, but for write operations only.

ASSIGN FROM DISK
CASSETTE
CASSETTE n (n = 0 or 1)

Similar to above ASSIGN commands, but for read operations only.

ASSIGN OUTPUT TO SCREEN
PRINTER
DISK

Assigns the system data outputs to the TV, TV + Printer, or diskette, respectively. If the DISK option is specified, the system output will be written to a currently open ASCII diskette file (name to be specified later).

ASSIGN INPUT FROM KEYBOARD
RS232
DISK

Reads the system input data from the keyboard, RS232 channel, or diskette, respectively. The first two options will generate automatic echo of the data read on the TV screen. If the DISK option is specified, the data will be read from a currently open ASCII diskette file (name to be specified later).

VERIFY DISK (n)

Verify contents of the diskette in drive n (n = 0 or 1). In case read errors are detected, suitable error messages will be displayed, and the checking will continue. See section 6. 6. 38 of Software Handbook for details and format of error messages.

VERIFY DISKS Similar to above command, except that the contents of diskettes in both drives will be verified.

VERIFY (filereference)

Verifies the contents of the specified filereference. Any error will stop the checking process.

IDISK n Initializes the diskette in drive n (n = 0 or 1). This command writes 55H to all sectors, and the original contents of the diskette are therefore lost. Does not verify the diskette (can be done subsequently via the VERIFY command).

RECOVER filereference

Recovers the contents of the deleted file 'filereference', or, unprotects the protected file 'filereference'.

PROTECT filereference

Write-protect the file 'filereference'. Write-protected files will be shown in the directory with a 'W' against them.

RENAME oldfilereference newfilereference

Rename the 'oldfilereference' as 'newfilereference'.

DSAVE filereference minadr maxadr entryadr

Save a binary file on diskette. See Section 6.6.36 of Software Handbook for details.

DLOAD filereference

Load a binary file from diskette. No loadaddress can be specified in the command. See section 6.6.23 of the Software Handbook for details.

DOS Error Codes (see section 6.7 of Software Handbook)

1	INVALID COMMAND CODE
2	INVALID SYNTAX
3	INVALID DISK UNIT
4	FILE ALREADY EXISTS
5	FILE TOO LONG OR DISK OVERFLOW
6	INVALID ACCESS MODE
7	FILE WRITE-PROTECTED
8	FILE NOT IN DIRECTORY
9	FILE NOT OPENED
0AH	DIRECTORY FULL; FILE CANNOT BE CREATED
0BH	TOO MANY PARAMETERS
0CH	NO PARAMETER
0DH	FILE ALREADY OPENED
0EH	END-OF-FILE REACHED
0FH	INVALID FILE NUMBER
10H	TOO MANY OPEN FILES
11H	INVALID RECORD NUMBER

81H	INVALID DISK COMMAND
82H	INVALID DISK PARAMETER
83H	SELECT FAULT
84H	FAILED TO FIND SECTOR
85H	FAILED TO FIND TRACK
86H	CRC ERROR IN IDENTIFICATION BLOCK
87H	CRC ERROR IN DATA
88H	ILLEGAL DATA MARK
89H	DELETED DATA MARK
8AH	WRITE FAULT
1FH	BLOCK SAVE AREA TOO SMALL OR DESTINATION FILE TOO SMALL WHEN COPYING A FILE TO ANOTHER
21H	SYNTAX ERROR CODE
31H	BLOCK OVERFLOW ERROR INDICATES THAT FIXED LENGTH RECORDS FILE DOES NOT CONSIST OF EXACT RECORDS
30H	NO DISK UNIT IN FILE NAME
22H	INVALID ASSIGNMENT
23H	NO EXTENSION IN FILENAME (NOT AN ERROR)
24H	INVALID PROM TYPE

Any DOS error can be handled (for example, within a BASIC program) in the following way:

PEEK #9B7

The result will be zero (if no error), or it will give the detected error code.

Loading Errors may be detected when accessing diskette files via the BASIC commands LOAD, LOADA or the Utility command R.

LOADING ERROR	0	: File not found.
	1	: Block too large (i. e. not enough memory). Data will be loaded till the end of the available memory block. For example, LOAD allocates from top of heap to bottom of screen area for loading programs.
	2	: Bad syntax, or invalid file type.
	3	: Any other disk error ("PEEK #erradr" will give the error code).

In all cases, the actual DOS error code will be stored at the memory address "erradr" (to be specified later), which can be read and processed.

All other DOS errors will be displayed on the TV screen as:

+++ DISK ERROR nn +++

where 'nn' is the 2-digit hexadecimal error code specified before. These messages will be accompanied by an explicit error message (details later).

Program Overlays

Large programs can be divided into smaller overlay modules, and executed. Since all BASIC variables will be cleared when each program module is loaded, variables common to the overlay modules must be saved (say on scratch file as arrays). When successive overlay modules are loaded (using DOS commands within the program modules), the relevant common variable files can then be loaded and the program variables suitably initialized

An additional feature is provided by DOS to enable a BASIC program file to be loaded and executed when its filename is stored in a BASIC string variable. This is supplementary to the normal LOAD "file" command which only allows a string constant as the filereference.

This feature is evoked by the sequence: CALLM 300, A$ where A$ can be any array variable.

ASCII INPUT AND OUTPUT FILES

In order to provide the user a more flexible file format outside the constraints of the array SAVE and LOAD commands (LOADA, SAVEA), sequential ASCII files can be used. These operate, and are interfaced by the user as follows:

1. ASCII INPUT FILES

The file must first be opened by using the 'OPENI' command. The format of this is:

*OPENI FILEREFERENCE OPENI FILENAME.BAS

This command automatically closes any file previously opened for ASCII input.

After the execution of the command: 'ASSIGN INPUT FROM DISK' anytime that the system would normally access the keyboard for character input these will come, in sequence, from the opened file. Thus, immediate commands, program entry, utility commands may all be prepared as are responses to 'INPUT' or 'GETC' statements. When the end-of-file error is reported, the input stream automatically returns to the keyboard, thus preventing any serious system crashes that may have been caused.

An example of using the ASCII input mode follows:

```
10   CLEAR 1000:POKE  131,3:PRINT "OPENI  $DKDIR"
15   PRINT "ASSIGN INPUT FROM DISK":PRINT "ASSIGN OUTPUT TO SCREEI
20   FOR I=1 to 18:REM 18 SECTORS IN THE DIRECTORY.
30   FOR J=1 to 6: REM 6 ENTRIES PER SECTOR
40   A$=""
50   FOR K=1 to 21:REM 21 BYTES PER ENTRY.
60   A$=a$+CHR$ (GETC):REM NEXT CHARACTER FROM DISK
70   NEXT K
80   PRINT LEFT$ (A$,11):REM FILENAME + EXTENTION
90   NEXT J
100  K=GETC+GETC:REM PURGE 2 PAD CHARACTERS
110  NEXT I
120  POKE  131,3:PRINT "ASSIGN INPUT FROM KEYBOARD"
130  STOP
```

2. ASCII OUTPUT FILES

A system of ASCII control characters has been defined to direct the command character stream ('PRINTED' after 'POKE 131,3') to either the DOS command handler, the ASCII file manager, or the currently selected ASCII output file. These control characters, and their functions are as follows:

SOH CHR$(1) Closes current output file (with adjust) switches character stream to file manager (normally used to precede filereference).

Special Case:

SOH + ETX, closes file but does not open a new file.

STX CHR$(2) Used to select the ASCII file to receive characters. When STX follows the filereference (preceded by SOH), the file is created if necessary, then opened for output.

ETX CHR$(3) Returns character stream back to the DOS command handler. This is the only character which can suspend the character stream going to the output file. Hence, it is also the only exception to the otherwise totally transparent nature of the ASCII input/output mode.

These characters, and the filereference and data stream are sent by normal PRINT statements whilst the output has been assigned to disk.
I.e. POKE 131,3.

Normal output to screen or printer may be freely interspersed with disk data. DOS commands may also be issued but caution must be taken in avoiding the use of any DOS command which accesses the diskettes. These can cause the file to be closed prematurely and without adjust. Thus the final sector of the dataset may not be written onto the diskette. This caution also applies to the ASCII input mode.

An example of ASCII output follows:

```
1     CLEAR 1000:SOH$=CHR$(1):STX$=CHR$(2):ETX$=CHR$(3)
10    POKE  131,3:PRINT "RESETD"
20    PRINT SOH$+"FILENAME"+STX$+ETX$
30    PRINT "ASSIGN OUTPUT TO SCREEN"
40    PRINT "THIS TEXT IS ON THE SCREEN"
50    POKE  131,0:PRINT STX$+"THIS TEXT IS TO THE DISK FILE"+ETX$
60    PRINT STX$+"THIS IS SOME MORE TEXT"
70    PRINT:PRINT:PRINT "THEN SOME BLANK LINES"+ETX$
80    PRINT SOH$+"ANOTHER"+STX$+"DATA TO ANOTHER FILE"+ETX$
90    PRINT SOH$+ETX$:REM CLOSE LAST FILE
100   PRINT "ASSIGN OUTPUT TO SCREEN"
110   PRINT:PRINT "BYE - BYE"
120   STOP
```

Standard System Diskette: DSK-PCD V1.0

This diskette provides 4 system programs to the user.

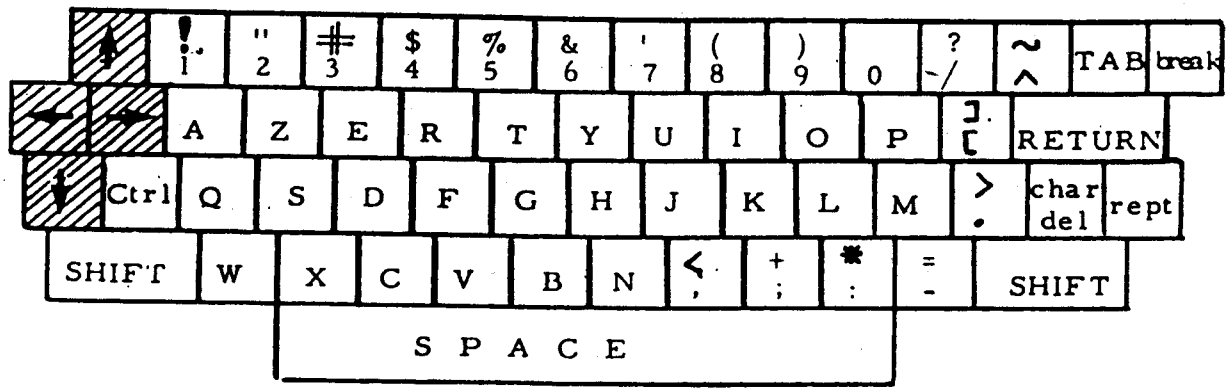
1) $DKBOOTS Bootstrap routine. See power-up details.

2) $MSTRDOS DAI Personal Computer DOS V1.0

3) $USER.BAS DAI "AUTO-MENU" program executed on power-up.

4) $AZERTY.BIN Modified Keyboard map to transform the Keyboard into an AZERTY (European) format. Must be renamed $USER.BIN before use. The Keycaps must be re-arranged into the following layout:

Keyboard Layout



The keys are assigned to rows and columns.

ROWS / Output lines (FF07)	0	1	2	3	4	5	6
0	0	8	re-turn	H	P	X	↑
1	1	9	Q	I	A	Y	↓
2	2	/	B	J	R	W	←
3	3	M	C	K	S	.	→
4	4	;	D	L	T	[	Tab
5	5	^	E,	. .	U	space bar	ctrl
6	6	:	F	N	V	rept	break
7	7	-	G	O	Z	char del	shift

COLUMNS

Input lines (FF01)