TR TOM SOLHEIM

Buffered Edit — helst men min 32K

STD Comm. Basic — må ha 48K

500Kr. Forms Handler — klarer seg med 16K. bør i enkelte tilfelle ha 32K

PL/M compiler fra MYCRON (7-8000 kr.) — må ha 2 diskette 48K

Service håndbok TDV 2114.

TOS 21 Users Guide

January 1977

CONTENTS

PART 2 TOS 21

Introduction.

The Tandberg Operating System (TOS21) is intended for use with the Tandberg TDV 2100 series display terminal. It makes available to the user generalized input/output routines, a command interpreter for program invocation and a general file assignment mechanism used both by system and user programs.

TOS may easily be reconfigured to suit paricular user needs, and if desired a user may redefine completely specific system functions.

The TOS21 system consists of two parts:

- a PROM monitor
- The RAM based i/o system, command interpreter and assignment interpreter

This publication is divided into two distinct parts: one discussing the monitor and one discussing the complete TOS21 system. Except for special debugging purposes, a TOS21 user need not be familiar with the details of the monitor.

PART 1

THE MONITOR

1.1 Kernel commands

The monitor residing in PROM will be entered normally either:

- When power is turned on and any key on the keyboard is depressed.
- When a breakpoint is encountered during execution (program under debugging)
- When Reset button is pressed or
- If a JMP to address zero is executed

The monitor will display current program counter (not when power is turned on) and will display a dot as a prompting symbol, ready to accept commands.
The commands are:

S Memory inspect/substitute
X Register inspect/substitute
G Start execution with or without brakpoints
D Display memory
F Fill memory
L Load binary file
R Load resident monitor .
A Display all possible codes
T Act as teletype
H Help - display a command summary

In the following CR means Carriage Return button, Sp means Spacing bar.
When a hexadecimal address or value is requested, any mumber of digits may be entered. Each hexadecimal digit represents four bits, and superfluous digits are shifted out. Left zeroes are inserted when too few digits are entered. To correct errors in typing: Simply go on until the error is shifted out.

When the monitor discovers an erroneous command character, it will respond with an * and revert to the prompting symbol.

1.1.1 S-command-Memory inspect and substitute

Key in: S<Address>Sp
The contents of the addressed memory location is displayed.
To display the contents of the next location:

Type: Sp

To change the contents of the displayed location and display the next:

Type: <value>Sp

To return to the prompting symbol:

Type: CR

To change the contents of the displayed location and return to the prompting symbol:

Type: <value> CR

1,1,2 X-command-Register inspect and substitute

Key in: X <register identifier>

This is adequate when a breakpoint in the user's program is executed, or after a button Reset. The contents of all registers are then stored and may be scrutinized.

The register identifiers are the following:

A, B, C, D, E, H, L:		Single registers
F	:	Status register (Flip-flops)
P	:	Program counter
S	:	Stack pointer
X	:	All registers

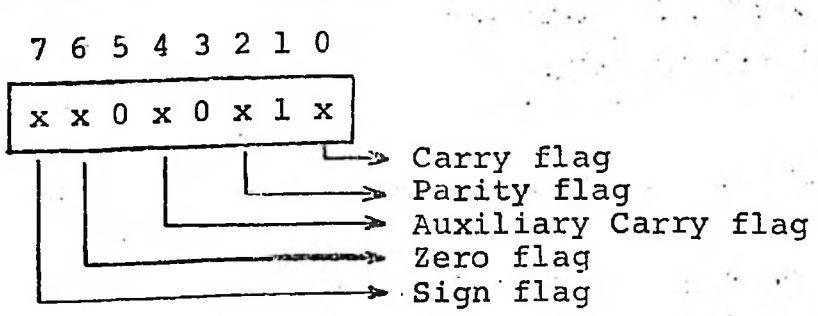
The contents of the selected register is displayed.
To return to the prompting symbol: Type CR
To change the register contents: Type <value> CR
The two double word registers P and S cannot be changed directly.
To change P:

The pseudo registers Q and R represents the upper and lower half of P

To change S:

The pseudo registers T and U represents the upper and lower half of S

The F-register (status flip-flops) contains the conditions flags packed as follows:



1,1,3 G-command - Start program execution

Syntax:

G CR
G <address 1> CR
G Sp <address 2> CR
G <address 1> Sp <address 2> CR
G Sp <address 2> Sp <address 3> CR
G <address 1> Sp <address 2> Sp <address 3> CR

Address 1 is the desired starting address. When it is omitted, the current address is used, i.e. restart after a breakpoint.

Address 2 and address 3 are breakpoints and will be set if those addressed are entered.

If one breakpoint is executed, both breakpoints will be reset. The breakpoints will also be reset if the program is interrupted by the Reset button.

To set a breakpoint, the monitor will substitute the user's program instruction by a RST 0. If this instruction is used in the program, it may cause confusion.

1,1,4 D-command - display memory

Syntax:

D <address>, <number> CR

Memory is displayed as lines starting with a 16 bits address followed by 16 8-bits values.

The typed starting address is rounded down to the nearest 10 (hexadecimal).
The displayed part of memory will always be in groups of 10 (hexadecimal) words.

A number exceeding 170 (hexadecimal) will be of no use, as the first displayed information then will be rolled out.

1.1.5 F-command - fill memory

Syntax:

F <lower address> , <higher address> , <value> CR

All memory locations from and included the lower address to and included the higher address, will have their contents replaced by the typed value. Note that the area FF00 to FFFF cannot be filled by use of the F-command.

1.1.6 L-command - load binary program file

Syntax : L[:Fn:] name[. extension] CR

n is unit (0-3) 0 is default
name is 1-6 alphanumeric character
extension is 1-3 " " "

1.1.7 R-command - load extended monitor, TOS

Syntax : Ri

This command is an abbreviation for

L:Fi:TOS21 CR

(Note that Carriage return is not required)

1.1.8 A-command - fill the screen with all codes

This is a hardware test facility.

Syntax : A CR

Function: Codes 00H to FFH are entered on the screen a number of times until all but the last position on the screen is filled (to avoid roll).

1.1.9 T-command - act as a teletype

Syntax : T CR

Function off line:

Each character entered from the keyboard is echoed on the screen

Function on line:

Characters entered from the keyboard are transmitted asynchronously.
Incoming characters are displayed.

1,2 User callable routines

The monitor kernel contains a number of user callable routine

These are entered by a CALL instruction(with exception of MON which should be entered by JMP) and returns to user (except MON, LINK and CLINK)

Address (hexadecimal)	Routine	Function
40	TTI	Read from keyboard
43	TTO	Write on screen
46	TTONC	Write on screen, no control char.
49	CURSOR	Position cursor
4C	MON	Enter monitor-NO RETURN
4F	ERROR	Display error message
52	PRINT	Print character
55	DTST	Test for completion of diskette operation
58	DCAL	Initiate Calibrate diskette
5B	DSRD	Initiate Read diskette sector
5E	DSWR	Initiate Write diskette sector
61	DSDL	Initiate Delete diskette sector
64	※)DSRDA	Initiate Read diskette sector and convert to ASCII.
67	※)DSWRA	Convert to EBCDIC and initiate write diskette sector
6A	INDISP	Read charater from screen
6D	DSWT	Wait for completion of diskette operation
70	DCALW	Initiate Calibrate and wait for completion.
73	DSRDW	Initiate Read and wait for completion.
76	DSWRW	Initiate Write and wait for completion.
79	DSDLW	Initiate Delete and wait for completion.
7C	※)DSRAW	Initiate Read convert to ASCII and wait for completion.
7F	※)DSWAW	Convert to EBCDIC, initiate write and wait for completion.
82	RCVI	Receive from host (UART)
85	LOCDIR	Locate Intel format directory entry.
88	LOAD	Load program from diskette.
8B	LINK	Load program and start its execution. NORMALLY NO RETURN.

※) Require TOS 21 support.

Address (hexadecimal)	Routine	Function
8E	CTST	Test for completion of cartridge operation.
91	CCAL	Initiate rewind (calibrate) cartridge.
94	CBRD	Initiate Read Cartridge block
97	CBWR	Initiate Write Cartridge block
9A	CBED	Initiate rewrite (edit) of a cartridge block.
9D	CBTM	Initiate write tape mark on cartridge
A0	CBSF	Initiate Search Forward on cartridge.
A3	CBSB	Initiate Search Backward on cartridge
A6	CUN	Initiate rewind and unload of cartridge.
A9	CRW	Initiate rewind of cartridge.
AC	CWT	Wait for completion of cartridge operation.
AF	CBRDW	Initiate read and wait for completion (cartridge)
B2	CBWRW	Initiate write and wait for completion (cartridge)
B5	CBEDW	Initiate rewrite and wait for completion (cartridge)
B8	CBTMW	Initiate write tape mark and wait for completion (cartridge)
BB	CBSFW	Initiate search forward and wait for completion (cartridge)
BE	CBSBW	Initiate search backward and wait for completion (cartridge)
C1	CLOCD	Locate directory entry (cartridge)
C4	CLOAD	Load program from cartridge
C7	CLINK	Load program from cartridge and start its execution (Normally no return)
CA	XMIO	Send to host (UART)

Note that in a system supporting diskette only, the cartridge routines will contain a jump to an error routine and vice versa.

TTI, TTO, TTONC, CURSOR, MON, ERROR, PRINT, INDISP, RCVI and XMIO are present in all systems.

Upon return to user, the carry flag is used to indicate whether an error occurred in the routine:

carry cleared - normal completion

carry set - error occurred, A-register contains error code.

With the exception specified above in the case of error return, and results as indicated below, all registers are saved and restord.

TTI (40H)

Read one character from keyboard.

The routine will not return until a character has been received.

The character is returned in A-register.

TTO (43H)

Write a character in the current cursor position on the screen.

The character to be written is in the A-register.

Charachers with value less than or equal to 31(1FH)are interpreted as control characters.

The cursor is advanced one position for other characters.

TTONC (46H)

Write a character in the current cursor position on the screen.

The character to be written is in the A-register.

All characters are interpreted as data characters.

The cursor is advanced one position.

CURSOR (49H)

Positions the cursor on the screen to the desired character position.

H - desired line (0-24)

L - desired character position within line (0-79)

Parameters outside the specified range will position to line 0, character 0.

MON (4CH)

Enters the monitor. Does not return to user.

ERROR (4FH)

Displays the message ERROR nn (where nn is a hexadecimal number) on the screen.

Register A contains the error number. Returns to user if carry was cleared.

PRINT (52H)

Print the character contained in the A-register.

DTST (55H)

Tests for status of diskette operation and set result in A and carry as follows:

Carry set - completed operation
A = 0 - normal completion
A ≠ 0 - error, A contains error code

Carry not set - operation is not yet completed.

DCAL (58H)

Initiate Calibrate diskette. Parameter: A-unit (0-3).
Initiates a calibrate operation (position to track 0) for the specified diskette unit.
Returns before operation is completed, the DSWT routine must be called prior to initiating another diskette operation.

DSRD (5BH)

Initiate Read of a diskette sector.
Parameters. A-unit, B-track, C-sector, HL-buffer address.
Initiates reading a specified sector from a specific unit.
Performs necessary seek operations. Returns before operation is completed, the DSWT routine must be called prior to initiating another diskette operation.

DSWR (5EH)

Initiate Write of a diskette sector.
As DSRD, but initiates output of a specified sector.
The output buffer must not be altered until completionhas been checked.

DSDL (61H)

Initiate Delete of a diskette sector.
As DSWR, but a deleted data mark is written in the sector data header.

DSRDA (64H)

Initiate Read of a sector and convert from EBCDIC to ASCII. As DSRD, but with conversion after read. Note that an error will occor, if the required conversion table is not available in RAM memony.

DSWRA (67H)

Convert from ASCII to EBCDIC and initiate write of a sector. As DSWR, but with conversion before write.
Note that the buffer contents are changed to EBCDIC, and that an error will occur if the required conversion table is not available in RAM momony.

INDISP (6AH)

Read a character from the screen (the current cursor Position). Cursor is moved to the next position.

DSWT (6DH)

Diskette wait for completion.
No parameters. Wait for completion of previously initiated operation and return with status in A.
Carry is set if an error occurred, and A contains the error code. Carry is cleared if the operation was completed normally. A is in this case set to zero.

DCALW (70H)

Initiate Calibrate diskette and wait for completion. Equivalent to a call on DCAL immediately followed by a call on DSWT.

DSRDW (73H)

Initiate Read of a sector and wait for completion. Equivalent to a call on DSRD immediately followed by a call on DSWT.

DSWRW (76H)

Initiate write a sector and wait for completion. Equivalent to a call on DSWR immediately followed by a call on DSWT

DSDLW (79H)

Initiate delete of a sector and wait for completion. Equivalent to a call on DSDL immediately followed by a call on DSWT.

DSRAW (7CH)

Initiate read of a sector, convert to ASCII and wait for completion.
Equivalent to a call on DSRDA immediately followed by a call on DSWT.

DSWAW (7FH)

Convert to EBCDIC, initiate write of a sector and wait for completion.
Equivalent to a call on DSWRA immediately followed by a call on DSWT.

RCVI (82H)

Read a character from host.
Waits until a character arrives and returns it in A-register.
(Up to 31 characters will be buffered automatically).

LOCDIR (85H)

Locate directory entry on Intel-formatted diskette.
Parameters: A-unit. HL-address of 9 character name area.
Returns:

HL-address of directory entry.

(For further information more detailed knowledge of the Intel directory format is necessary.
This routine is not normally used outside systems programming).

LOAD (88H)

Load a program from an Intel-formatted diskette.
Parameters: A-unit, HL-adress of 9 character name area.
Returns:

HL-starting address of program

LINK (8BH)

Load a program from an Intel-formatted diskette and start execution.
Parameters: as LOAD.
Returns only if an error occured.

CTST
(8EH) - Test status of cartridge operation

Returns:

Carry set - operation completed

A = 0 normal completion
A ≠ 0 error, A contains error code

Carry not set - operation is not yet completed

CCAL
(91H) - Rewind (calibrate) cartridge

A-unit.

Returns before operation is completed, the CWT routine must be called prior to initiating another cartridge operation.

CBRD
(94H) - Read a block from cartridge

A-unit B-track HL-buffer address.

Initiates reading of the next block on the specified track. Returns before operation is completed. The CWT routine must be called prior to initiating another cartridge operation or using the data in the buffer.

CBWR
(97H) - Write a block to cartridge.

A-unit B-track DE-block length HL-buffer adr.

Initiates writing of a block on the specified unit/track. The output buffer must not be altered until completion has been checked.

CBED
(9AH) - Rewrite a block.

A-unit B-track DE-block length HL- buffer adr.

Initiates rewrite (edit) of a block.

CBTM
(9DH) - Write tape mark

A-unit B-track

Initates writing of a tape mark.

CBSF
(A0H) - Search Forwards

A-unit B-track C-no of tapemarks

Initiates a forward search for the specified number of tapemarks

CBSB
(A3H) - Search Backwards

A-unit B-track C-no of tapemarks

Initiates a backward search for the specified no. of tapemarks.

CBUN
(A6H) - Rewind and unload.

Initiates a rewind and unload operation.

CRW (A9H)

Initiate rewind of cartridge. A subsequent operation Wait until the rewind is completed (up to 60 seconds).

CWT (ACH)

Cartridge wait for completion.

No parameters. Wait for completion of previously initiated operation and return with status in A. Carry is set if an error occured. and A contains the error code. Carry is cleared if the operation was completed normally. A is in this case set to zero.

CBRDW (AFH)

Initiate Read of a block from cartridge and wait for completion.

Equivalent to a call on CBRD immediately followed by a call on CWT.

CBWRW (B2H)

Initiate write of a block on cartridge and wait for completion.

Equivalent to a call on CBWR immediately followed by a call on CWT.

CBEDW (B5H)

Initiate rewrite of a block on cartridge and wait for completion.

Equivalent to a call on CBED immediately followed by a call on CWT.

CBTMW (B8H)

Initiate write of a tape mark on cartridge and wait for completion.

Equivalent to a call on CBTM immediately followed by a call on CWT.

CBSFW (BBH)

Initiate a forward search on cartridge and wait for completion.

Equivalent to a call on CBSF immediatley followed by a call on CWT.

CBSEW (BEH)

Initiate a backward search on cartridge and wait for completion.

Equivalent to a call on CBSB immediately followed by a call on CWT.

CLOCD (C1H)

Locate directory entry on cartridge

Parameters: A - unit, HL - address of 9 characters name area.

Returns: HL - address of directory entry.

(For further information, more detailed knowledge of the Tandberg cartridge format is necessary. This routine is not normally used outside systems programming).

CLOAD (C4H)

Load a program from cartridge

Parameters: A - unit, HL - address of 9 character name area.

Returns:

HL - starting address of program

CLINK (C7H)

Load a program from cartridge and start execution.
Parameters: as CLOAD
Returns only if an error occurred.

XMIO (CAH)

Send a character to host.
Parameter: A-character to be sent.

Note: The routine will wait if the transmit buffer is full.

1,3 Programming note.

Whenever no processing is required in paralell to diskette or cartridge transfer, the routines which wait for completion of the operations ought to be used. Note that when an operation is initiated, the data area must not be accessed until the completion of the operation has been checked. Errors when this condition is violated may be extremely difficult to detect, as the effect may be timing dependent, and two otherwise identical program executions may yield different results.

Note that if a diskette or cartridge routine is called prior to the completion of a previously initiated diskette operation, a delay will occur until the first operation is completed. If an error occurs in the first operation, it will not be reported to the user.

1,4 Error codes from monitor

The error codes returned by the monitor is a subset of TOS error messages. Refer to appendix A for codes.

PART 2

TOS 21 Operating System

This part gives partly an informal introduction to the use of TOS 21, partly a more formal description of the facilities available.

A number of details which is mainly of interest to system programmers, and some reference material, such as error codes, are included as appendices.

Section 3 contains a description of the utility programs supplied on the TOS 21 system diskette.

2,1 Loading TOS 21

When the power is turned on the TDV 2100, insert a system diskette in any diskette unit (or cartridge in cartridge unit).

TOS 21 will then be loaded automatically and identify itself by

TOS 21 VER x·y

where x·y identifies the particular TOS 21 version.

The command prompt (%) will then be give to signify that TOS 21 is expecting a command from the keyboard.

2,2 Commands- an introduction

Programs for use with the TOS 21 system are stored on diskettes in a special format (see Appendix E), and are identified by a name. (If the system was suitably configurated at system generation time, programs may also be loaded from cartridge).

Execution of a program are initiated by specifying its name. If the diskette containing the program is not on unit 0, the unit must also be specified.

In addition, most program requires specification of the name and location of the files (data and results) on which the program should operate. The program will refer to files logically by a name (there are eight such names, see below).

When execution of a program is started, the command must indicate which physical file should correspond to each different logical file. (Assignment need only be given for these logical units which are referenced by the program.) This is expressed by a sequence of assignments. The format of these will be discussed in detail later.)

That a program may be written refering to logical rather than to physical units both simplify programming (since the programmer usually need not think about what sort of files might be actually used) and use (since only one method of expressing files to be used by programs need be learned, rather than having different convension for each program.)

How this works is probably best illustrated with an example. Assume that we have some text information (as opposed to binary data or executable programs) stored on a diskette. This (physical) file will then contain a number of lines separated by carriage return and linefeed characters

A program called MOVE which is supplied with the TOS 21 system (on the same diskette) may be used to copy from one such file to another.

Assuming that the system diskette is on unit 0, we indicate that we want to execute the program by typing

MOVE ␣

(␣ is a blank character).

Next we must specify the file we want to copy from. Assuming that it is on the diskette on unit 1 and that its name is TEXT. we specify that it is the input file by typing

SI=:F1:TEXT

SI= specifies that we are assigning the Standard Input file. :F1: indicates that the file is on a diskette on unit 1, and that this diskette is in the so called "Intel" format (other letters are used for other formats and for other media, such as cartridge). The remaining part specifies the name under which the data is stored on diskette.

The MOVE program also need to know an output file where the copy of the input shall be written. Assuming that we want a copy on the system diskette, with the name TEXT.OLD we can continue by

typing:

,SO=TEXT.OLD

The comma is required to seperate the two individual assignments. SO= indicates that we are assigning the Standard Output file. The absence of a unit specification (:etc) indicates that we want the file on diskette unit 0 in "Intel" -format. (The system diskette has automatically this format). We could, with the same effect have written

,SO=:F0:TEXT.OLD

When we now type carriage return the command will be executed. It looks like: MOVE SI=:F1:TEXT,SO=TEXT.OLD

The actions of TOS 21 are now:

1. Check that a file named MOVE (i.e. the program) is present on the diskette om unit 0, load it into memory.
2. Check that a file named TEXT is present on the diskette on unit 1, allocate buffers and read the first sector of the file.
3. Check that no file named TEXT.OLD is present on the diskette on unit 0 (since it is an output file to be produced by the MOVE program), allocate buffers for the file and reserve storage of the diskette.
4. Start program execution.

After completion of the execution, TOS 21 will again regain control, and will

1. Release the buffers assigned to the input file
2. Release excessive storage reserved on the diskette for the output file, update the diskette directory with information about the file and release the buffers
3. Return to wait for the next command.

The same MOVE program may be used if we had a printer available, and wanted to print the file. The physical designation recognized by TOS 21 for a printer is :LP:, so our command would look like

MOVE SI=:F1:TEXT,SO=:LP:

This simplicity results from the use in programs of logical unit references.

One of the first things we need to know is how to correct our typing errors when typing commands. Commands are read from the logical unit CI and the file corresponding (by default) to CI is a so-called line-oriented file, i.e. a complete line of information is prepared at a time, and until a carriage return is received the line may be altered by two edit control characters:

DEL - will cause the line to be deleted
← - will cause the last character to be deleted.

Without going into details on how this is achieved (but when a user has more experience, it will be no problem for him to build a file structure with similar characteristics) the line or characters will disappear from the screen when editing is performed so that a user always sees his command exactly as TOS 21 will receive it.

Some programs may require parameters, or requests information during execution from CI (normally the keyboard). The editing facilities discussed above may be used also when entering this information. If the parameters to a program are few, they may be entered on the same line as the command. The assignment and the parameter information (to be read by the program) is then seperated by a $.

As an example. A standard program called DIR produces a listing of the names in the directory of an Intelformatted diskette. It requires as a parameter, the number of the diskette unit.

Assuming that we want the listing to appear on the screen and that the diskette is on unit2, the command required is:

```
DIR SO=:CO:$:F2:
(:CO: designates the screen)
```

If we at the present time have :CO: as the logical output file CO (which incidentally is the default when TOS 21 is started), we might write the assignment as

```
SO=/CO
```

SO= indicates that we are assigning the Standard Output file. /CO indicates that SO should be assigned to the same file as CO is assigned to (which happens to be the physical file :CO:, i.e. the screen.) The complete command will in this case be

```
DIR SO = /CO$:F2:
```

For this particular program (DIR) a default option has been built into the program, since a directory listing usually is desired on CO. If no assignment for SO is given, the assignment SO=/CO will be implicitly performed. We have thus the "short form":

```
DIR$:F2:
```

If we want a listing on a printer

```
DIR SO=:LP:$:F2:
```

will do that.

You can even put it on diskette for subsequent listing by use of MOVE:

```
DIR SO=DIRLST$:F2:
```

and print by

```
MOVE SI=DIRLST,SO=:LP:
```

Further examples on program invocation are found in the section "Utility programs".

Program names (such as DIR and MOVE in the examples above) may be written in lowercase letters provided that no file exists with the name in lowercase. TOS will first search for the program with the lowercase name, and if this is not found, it will try to find the name in uppercase.

Logical unit designators (e.g. SI, SL etc.) and physical unit designators excepting filenames (e.g.: LP:,:CO: etc.) may be written in lowercase with the same meaning as in uppercase.

If the program is contained on one diskette and data on another, and only one diskette drive is available, the program name and assignments may be separated as follows

```
programname/CR
(change diskette)
assignment (or $ if no assignment)
```

Example:

```
DIR/
SO=:LP:$:FO:
```

to produce a listing of the directory other than the one containing the DIR program.

CAUTION: Wait until the program loading is finished before changing diskette. The selected light in the diskette drive must be off.

2.3 Logical units and physical files

This section gives a more formal and complete description of commands, logical units, files and assignment.

It is assumed that the reader is familiar with the BNF syntax notation. A frequent reference to the examples at the end of the section may be useful for readers not familiar with this syntatic form of expression.

TOS 21 recognizes eight different logical units which may be assigned (and buffer space automatically allocated and deallocated) through the assignment interpreter.

These are.

CI - console input
CO - console output
SI - standard input
SO - standard output
SL - standard list
AI - auxiliary input
AO - auxiliary output
AL - auxiliary list

Very few programs will use all these logical units. Consult the individual program description for details.

The full list of physical files which may be assigned to the logical units are:

:BB: - byte bucket file
:NF: - null file
:CI: - keyboard
:CO: - screen
:CR: - séquential reader *)
:LP: - sequential output
:XM: - remote transmit
:RC: - remote receive

*) only if configurated on TOS generation.

```
filename          diskette file on unit 0
filename. ext     (Intel format)

:Fi:filename          diskette file on unit i (0<i<3)
:Fi:filename. ext     (Intel format)

:Ii:filename      diskette file on unit i
                  (0 < i < 3), IBM format
                  EBCDIC code
:Ai:filename      diskette file on unit i
                  (0 < i < 3), IBM format
                  ASCII code
:Mi:filename      cartridge file on unit i
:Mi:filename. ext (0 < i < 3), Tandberg format
```

The meaning of these physical file designations are:

:BB: is an output ("byte bucket") file which is always open. The informasjon written to this file is simply discarded.

:NF: is an input file ("null file") which is always open, but will return end-of-file whenever referenced

:CI: is a line-image file representing the usual keyboard with echo on screen (:CO:).

:CO: is the usual console output (screen) operated as a TTY equivalent (except when screen control characters are used).

:LP: is a printer

:XR: is the UART used to transmit to host computer and

:RC: is receiving from the host.

The first four remaining options designes a file on a diskette unit Intel format and with specified name (max. 6 characters) and extension (max. 3 characters). Name and extension may consist of letters and digits only.

The I and A options designates a file on a diskette unit in IBM format and with specified name (max. 8 characters). Name may consist of letters and digits only. :Ii: indicates that the diskette contents is in EBCDIC representation and should be converted to ASC IIon input and from ASCII to EBCDIC on output. :Ai: indicates that the diskette contents is in ASCII representation

Note: Conversion is only possible if the conversion table was configurated on TOS generation.

M designates a file on a cartridge with Tandberg format. This format is designed to give the cartridge as good performance as possible compared to a diskette in Intel format. Appendix G describes the format.

To prevent excessive tape movement, TOS 21 will only accept one file assigned per physical cartridge unit.

We have already given some examples of commands. The remaining details are given here.

Formally, the command syntax are:

```
<command>: := <file> CR |
              <file>/CR <assign sequence>|
              <file> <blanks> <assign tail>|
              <file> <blanks> <assign sequence>
```

where <blanks> is one or more blank character. <assign tail> and <assign sequence> are defined later.

A command indicates that execution of a particular program should be initiated. The initiation occurs when the carriage return (CR) key is depressed.

The second form of the assignment may be used when the program should be loaded from one diskette (cartridge), data is to be taken from another diskette (cartridge), and only one diskette unit is available. When the / followed by a carriage return is encountered, the specified program is loaded. After loading, assignments is expected from CI.

> If a program is not found on the specified unit, TOS will try a second time with all letters converted to upper case. This does not apply to filenames given in assignments.

A command may extend over more than one line if the rules for continuation of the assignment part below are followed. The first part of the command is processed when the first carriage return is used. The execution of the program is not initiated until the whole command has been processed.

A command consists of a command part and an optional assignment part. The command part is a reference to a diskette or cartridge file.

The optional assignment part specifies which physical files should be assigned to the various physical units.

A command is terminated by CR or by a dollar sign which indicates that the remaining part of the CI lines will either be read by the invoked program, or when a command again is required and read from CI.

The ← character from CI causes deletion of one character and a echo of ← space ← on CO. The RUBOUT character causes the line to be deleted. The echo is ERASE LINE.

The (Horisontal) tab character causes positioning to the next character whose number is a multiple of 8. Blanks are inserted in the intervening characters.

Formally, the syntax of the assignment part is defined by:

```
<assignment> : := <logical unit> = <file designator>
<assign sequence> : := <assignment>  <assign tail> |
                       <assignment> , <assign sequence>
<assign tail> : := <carriage return> | $
<logical unit> : := CI|CO|SI|SO|SL|AI|AO|AL
<file designator> : := <empty> |
                      /< logical unit>|
                       <composite> |
                       <standard file> |
                       <file>
<standard file> : :=  :NF:|:BB:|:CI:|:CO:|:CR:|:LP:|:XM:|:RC:
<file> : := <unit designator>  <block file name>
<unit designator> : := <empty>|
                       :F <integer> :|
                       :I <integer> :|
                       :A <integer> :|
                       :M <integer> :
<block file name> : := <name>|
                       <name> . <name>
<composite> : := <echo>|
                 <double>
<echo> : := ( <file designator> / <file designator> )
<double> : := (<file designator> , <file designator> )
<name> : := <letter>| <digit>| <name> <letter> |<name> <digit>
```

An assignment sequence consists of a sequence of individual assignments separated by comma.

The format is

<logical unit> = <file designator>

An assign sequence is continued on the next line if a comma is immediately followed by a carriage return.

Note that with the exception of CI and CO, all assignment are performed one by one from left to right in the command. The sequence

.....SI=ABC,SO=/SI.....

has thus a different meaning from

.....SO=/SI,SI=ABC

Five different file designators are recognized:

empty — e.g. SI=
indicates that no file will be assigned to the specified logical unit. Any access to SO will cause an error.

/logical unit — e.g. SL=/SO
indicates that the file assigned to the right hand side logical unit is also assigned to the left hand one. In the example, SL will designate the same file as SO. (Merging output.

standard file — e.g. SL=:BB:
indicates that the file on the right hand side is assigned to the left hand logical unit.

block file — e.g. SI=:F1:ABC.X
indicates that the diskette file on the right hand side is assigned to the left hand logical unit. The same type of file designator is also used for cartridge assignments

composite — A composite file is further subdivided into two categories

echo e.g. CI=(:CI: XYZ)
indicates that input should be taken from the first file and echoed to the second.

double e.g. SO=(ABC.X,:CO:)
indicates that for output files, the output will be written on both, while on input it indicates concatenation of the two files.

3. Utility programs

The following utility programs are included on the TOS 21 system diskette:

FORMAT - initialize a diskette in Intel format
DIR - produce directory
DELETE - delete a file
RESCUE - reclaim a deleted file
ALIAS - define an alternate filename
ATTRIB - change the attributes of a file
ANALYZ - check a diskette for errors
DROP - mark sectors as not useable
MYLOAD - load from MYCRO-1 formatted program diskette.
MPACK - copy non-deleted files from one cartridge to another
MEMDMP - output a specified memory area as an executable program.
HEXBIN - convert from Intel "Hex" format to executable (binary) format.
MOVE - copy a line-oriented file
COPY - copy a block oriented file
EXEC - execute a "command library element"
DEBUG - execute program in debugging mode.
ASSIGN - assing a file for more than one command
ALLOC - allocate space for a file on an IBM formatted diskette
EDIT - sequential editor
XREF - produce cross-reference listing of an assembly program
ASM - assembler
MINIT - initialize a cartridge
RELEAS - release a file which was previously assigned
RENAME - rename a file
HEXLST - list a file in hexadecimal form
DCOPY - copy a diskette
DIRPAC - pack a diskette directory

For simplicity it is in all program descriptions and examples in this section assumed that the system diskette is mounted on unit 0. If it is mounted on another diskette unit or the program should be loaded from cartridge, the program name must be prefixed by :Fn: or :Mn: where n is the appropriate unit number.

3.1 FORMAT

The FORMAT program is used to initialize a diskette in Intel-format or cartridge in Tandberg format.
It is invoked by

```
FORMAT $ unit volume name, or
FORMAT $ unit volume name, S
```

where unit is in the form used in assignment (:Fi:, :Mi:), and volume name is the name to be given to the diskette. The format of the name is as for files, i.e. a name of maximum six alphanumerical characters followed optionally by an extension of up to three alphanumeric characters.

The latter form is used to create a system diskette. In this case all files having the S-attribute set (see ATTRIB program description, section 3,7) will be copied from the second unit to the first. If both units specified are diskettes, files having the F-attribute set will also be copied.

The first form of the command will format a diskette containing only usage map (ISIS.MAP), directory (ISIS.DIR) and label (ISIS.LAB).

Examples:

```
FORMAT $:F2:LIB.BM
```

will format a diskette on unit 2 to contain usage map and directory. The name of the diskette will be LIB.BM.

```
FORMAT $:F2:SYS.V3, S
```

will format a system diskette on unit 2, give it the name SYS.V3 and copy files with S and F attribute from the diskette on unit 0.

3,2 DIR

DIR will produce a listing of the names and characteristics of the non-deleted files on a diskette or a cartridge.

The program is invoked by

```
DIR  SO=outputfile $ unit
```

or

```
DIR  $unit
```

where unit is in the form used in assignment (:Fi:, :Ai:, :Mi:, etc.). The unit may be omitted if it is :F0:

The assignment may be omitted, and SO =/CO is then used as default.

The listing contains for an Intel-formatted diskette:

- the volume name
- the name of the file
- the number of blocks in the file
- the file length in bytes
- the attributes of the file (see ATTRIB program description in section 3,7). Files with the I attribute set will not be included in the list.
- alias names are noted as such
- the total number of blocks (sectors) in use on the diskette

The listing contains for an IBM-formatted diskette:

- the volume name
- the name of the file
- track/sector of beginning and end of the space allocated to the file and end of information written on the file
- listing of file flags (see IBM publication describing the diskette format.)

For a cartridge, the listing contains:

- the volume name
- the name of the file
- the number of blocks in the file
- the file length in bytes
- the attributes of the file (see ATTRIB program description in section 3,7)
- the block size
- the track and file number of the first file segment of the file.
- the number of filesegments and number of blocks on each of the tracks 1 through 3. (Also indication if the track is full).

Examples:

DIR $:F1:

will produce a listing on CO for the Intel formatted diskette on unit 1.

DIR SO=:LP:$:A2:

will produce a listing on printer for the IBM (ASCII code) formatted diskette on unit 2,

DIR SO=CAR1$:M1:

will produce a listing as a file on :F0: named CAR1 for the cartridge on unit 1.

3,3 DELETE

DELETE will delete specified files from a diskette or cartridge. For an Intel-formatted diskette, the space will be released for use by other files.

The program is invoked by

DELETE $ filespecifications

where filespecifications is a sequence of filedcsignators (as used in assignments) separated by comma. The command cannot be extended beyond one line.

Example:

To delete two files, ABC and DEF on unit 1, and the file TEMP on unit zero, the program could be invoked by

DELETE $:Fi:ABC,:Fl:DEF,TEMP

If the file designator specifies an IBM-formatted diskette, the dataset label sector corresponding to that file is rewritten with a "deleted data" mark. Since allocation of space is not automatic on IBM-formatted diskettes, no actual release of sectors take place.

If the file designator specifies a Tandberg-formatted cartridge, the file directory block for that file is marked as deleted. The space used for the file will not be reused.
Different file designators may be mixed in one delete statement, e.g

DELETE $:Fl:ABC,:M2:QPR,:A2:IBM

For Intel-formatted diskettes and Tandberg-formatted cartridges, the directory will be searched for alias-names, and these will also be deleted.

Messages from program (on CO):

filename NOT FOUND	The filename did not exist on the specified unit. Processing continues with the next name in the list.

filename PROTECTED — The attributes of the file specifies that it cannot be deleted. Processing continues with the next name in the list.

filename DELETED — The deletion completed normally. Processing continues with the next name in the list.

filename ALIAS — The specified filename is an alias for another name. The original name must be given to release the space, hence only the directory entry is deleted. Processing continues with the next name in the list.

aliasname DELETED ALIAS — The specified filename (shown in the last "DELETED" message) had an alias associated with it which was also deleted.

IMPROPER UNIT SPECIFICATION — The specified unit designator was not valid. Processing is aborted.

IMPROPER PARAMETER LIST — An error was detected when processing the parameter list. Processing is aborted.

3,4 RESCUE

NOTE: CAUTION SHOULD BE EXECISED WHEN USING THIS PROGRAM

If by mistake or accident a file was deleted. RESCUE may be used to re-establish the file under certain conditions:

Intel-formatted disette:

- No output files was created on the diskette since the deletion.

IBM-formatted diskette:

- No new data set labels was created on the diskette since the deletion
- The space occupied by the deleted file was not re-used by overlapping files.

There are no restirction on reclaiming a deleted file on a Tandberg-formatted cartridge.

The program is invoked by

RESCUE $ filename

Example:

```
DELETE $ :F2:XYZ
RESCUE $ :F2:XYZ
```

Messages from program:

filename RECLAIMED	The file has been reestablished -hopefully properly. At least the RESCUE program did not detect anything wrong.
filename NOT FOUND	No deleted file with the specified name was found in the file directory
filename LOST	Reclaiming of the file was not possible due to violation of one of the rules specified above.

3,5 RENAME

RENAME will change the name of a specified file on a diskette or a cartridge. The program is invoked by

RENAME $filename, newname

where filename designates the file to be renamed, newname indicates the new name in the usual format (without unit specification).

Example:

To rename a file ORANGE on unit 1 (Intel formatted) to APPLE:

RENAME $:Fi:ORANGE, APPLE

Note that the number of rename operations on one file on a cartridge should be limited as much as possible.

Messages from program:

filename NOT FOUND	The filename did not exist on the specified unit.
filename EXISTS	The new name already exists as filename on the specified unit. The old name is retained on the file.
filename RENAMED	The rename operation was performed.
filename PROTECTED	The attribute of the file specifies that it cannot be renamed.
IMPROPER UNIT SPECIFICATION	The specified unit designator was not valid. Processing is aborted.
IMPROPER PARAMETER LIST	An error was detected when processing the parameter list. Processing is aborted.

When RENAME is used on an alias, the new name is also on alias.

3,6 ALIAS

ALIAS is used to establish an alternate name for a file. Except in the case of deletion (see DELETE), and alteration of attributes, an alias will act as if it were the original name of the file.

The program is invoked by

```
ALIAS $ filename, name
```

Example:

```
ALIAS  $:F3:STALIN,IRON.MAN
```

Messages from program:

filename NOT FOUND	The filename did not exist on the specified unit.
filename EXISTS	The alias already exists as filename on the specified unit. No alias is created.
name MADE ALIAS	The operation completed normally
IMPROPER UNIT SPECIFICATION	The specified unit designator was not valid. Processing is aborted.
IMPROPER PARAMETER LIST	An error was detected when processing the parameter list. Processing is aborted.

3,7 ATTRIB

ATTRIB is used to alter file attributes. A file have the following attributes which may be set or reset:

S - System attribute
F - Format attribute
W -Write protect attrobute
I - Invisible attribute.
A - Alias attribute

The attributes have the following effect:

S - attribute is set for system programs, i.e. TOS21 and the utility programs. They will be copied by the FORMAT program if creation of a system diskette is specified.

F - attribute is set for specific formatting system files,i.e. usage map, directory and error message file. They will be copied by the FORMAT program if creation if a system diskette is specified.

W - attribute is set if a file is to be protected from deletion or rename.

I - attribute is set if a file should not be included in the listing produced by the DIR command.

A - attribute is set for names created as an alias for other files (see section 3,6)

The ATTRIB program will allow alteration the S, W and I attributes only. It is invoked by

ATTRIB $ filename, attribute list

where attribute list is a sequence of specifications of the form + attribute or - attribute sepatated by comma (attribute is either S, W or I).

Examples:

ATTRIB $:F1:FIL.X,+I,-W

will set the invisible (I) attribute of the file FIL.X on unit 1 and clear the protect (W) attribute of the file.

ATTRIB $:M1:FILY,+S,+W

will set the system (S) and protect (W) attribute of the file FILY on the cartridge on unit 1.

Program messages:

filename NOT FOUND	the specified file was not found on the specified unit.
UNKNOWN ATTRIBUTE	the specified attribute was not I,S or W
IMPROPER UNIT SPECIFICATION	The specified unit designator was not valid
ATTRIBUTE LIST ERROR	The list of attributes was not in the correct format.

3,8 ANALYZ

ANALYZ will check a diskette to locate sectors with errors (usually CRC errors).

The program produces a list (on SO) of sectors with errors. The list may be used subsequently as input to DROP to mark these sectors as 'in use' so that they will not be allocated to files (Intel-formatted diskettes only):

The program is invoked by

```
ANALYZ  SO=outputfile $ unit
```

or `ANALYZ  $ unit`

where unit is in the form used in assignment (:Fi:, :Ai: etc.). The unit may be omitted if it is :F0:

The assignment may be omitted, and SO=/CO is then used as default.

Note: Sectors with deleted data is not considered as being in error, even through the occurrence of such sectors on an Intel-formatted diskette is illegal.

3.9 DROP

DROP will mark specific sectors as "in use" to prevent them from being allocated to new files. The DROP program applies only to Intel-formatted diskettes.

It reads a list of track and sector numbers from SI and sets the bit in the diskette map to mark the sector as "in use".

It is invoked by either

```
DROP  SI=inputfile$unit
```

or `DROP  $unit`

In the latter case, SI=/CI is assumed as default assignment.

Unit is of the form :Fi: where i is the unit number of the diskette unit.

The inputfile contains track/sector specification in the following format:

T/S, T/S, ... T/S carriage return

The input is terminated by an empty line.

Example:

1. To mark as "in use" track 10 sectors 4,8 and 9 of the diskette on unit 1:

```
DROP      $:F1:
10/4,10/8,10/9
```

2. Assume that we want to check the diskette on unit 1 and mark the sectors where errors are found:

```
ANALYZ  SO=TMP$:F1:
DROP    SI=TMP$:F1:
DELETE  $TMP
```

3. As 2, but assume that we also want a list on :LP:

```
ANALYZ  SO=(TMP,:LP:)$:F1:
DROP    SI=TMP$:F1:
DELETE  $TMP
```

3.10 MYLOAD

MYLOAD will load a program from a program diskette with the load format used for the MYCRO-1 microcomputer (produced by A/S MYCRON), and initiate its execution.

Loading is always from unit 0.

The program is invoked by

MYLOAD assignments $ programname

where assignments are the required TOS assignments if the MYCRO-1 program was written on that computer for use under TOS on TDV 2100.

Example: To load a program named MYC from a MYCRO-1 formatted diskette, assuming that the TOS diskette is on unit 1:

```
:F1:MYLOAD$MYC
If only one drive is available, use
MYLOAD/
Change diskette after program has been loaded
$MYC
```

3.11 MPACK

MPACK will copy all non-deleted files from one Tandberg-formatted cartridge to another.

It is invoked by

MPACK SO=outfile$inunit,outunit

or MPACK $inunit,outunit.

In the latter case, SO=/CO is assumed as default. The program reads from the cartridge on unit number inunit and writes those files which are not deleted onto outunit with the same name.

outunit must contain a cartridge which has been previously initialized with MINIT.

If a file with identical name already exists on outunit, MPACK will construct a new name.

Alteration of names and listing of the names of the copied files are output on SO.

Example:

To copy from unit 0 to unit 1, output on:LP:

```
MPACK  SO=:LP:$0,1
```

3.12 MINIT

MINIT initializes a cartridge with directory and special blocks required on "Tandberg-formatted" cartridges.

It is invoked by

```
MINIT  $unit,volumename
```

or

```
MINIT  $unit,volumename,R
```

where unit is a digit 0-3. The latter option indicates reinitialization.

The program will check to see if the cartridge is already initialized. If so, and the R-option was not present, the cartridge will not be reinitialized. Otherwise the required control blocks and tapemarks will be written onto the cartridge.

3.13 MEMDMP

MEMDMP may be used to dump an area of RAM as a binary file which may subsequently be loaded for execution.
It is invoked by

MEMDMP SO = outputfile$xxxx␣yyyy

where xxxx is the (hexadecimal) starting address, ␣ is a space and yyyy is the (hexadecimal) length of the area.
The program when loaded will start execution at xxxx.

Note: The area 2000H-27FFH cannot be written out.

3.14 HEXBIN

HEXBIN converts a program in Intel "hex" format on SI to a program in loadable format on SO (see Appendix E).
It is invoked by

HEXBIN SI = input file, SO = output file $ start address
or HEXBIN SI = input file, SO = output file $

In the former case, the created binary file will contain the specified starting address. The format is xxxx where x is a hexadecimal digit.

In the latter case, the start address is taken from the END-card of the assembly.

No subcommands from CI are required during program execution.
Program messages:

RECORD TYPE ERROR	Indicates that program found a hex-record with an unacceptable type code. The program terminates execution at this point. Normal return to TOS21.

3.15 MOVE

MOVE will copy lines from one file to another. It is invoked by

MOVE SI = infile, SO = outfile

Example:

MOVE SI = QUITO.LST,SO = :LP:

will copy from the file QUITO.LST to line printer.
(For further examples see section 2.2).

If an assignment has been made to AO, the same text is put in that file, but all lower-case letters are converted to uppercase.

Example:

If a printer only has uppercase letters, and we want to display a text on the screen using both lower and uppercase at the same time as printing it, we could use:

MOVE SI = infile, SO = :CO:, AO = :LP:

3.16 COPY

COPY will copy blocks from one file to another. It is invoked by

```
COPY  SI=infile,SO=outfile
```

Example:

```
COPY  SI=FIL1,SO=:M1:FIL1.BAK
```

will copy the file FIL1 (on diskette unit 0) to a new file named FIL1.BAK on cartridge 1.

NOTE: Acceptable assignments for COPY are only block oriented files, e.g. diskette or cartridge files.

3.17 EXEC

EXEC is used to invoke a sequence of commands residing on a diskette or cartridge file. When the end of file is reached, commands will again be requested from the keyboard.

The command sequence is initiated by

EXEC CI=infile

3.18 DEBUG

DEBUG is used to load a program and perform the required assignment, but not start its execution. Control is instead passed to the monitor kernel, where the commands of section 1 may be used to control execution of the program.

The program is initiated by

DEBUG $ progname assignment

(The right hand side of $ is a complete command).

Example:

DEBUG$:F1:NEWPROG SI=:NF:,SO=:CO:

will load the program NEWPROG from unit 1, perform the assignments and enter the monitor kernel.

3.19 ASSIGN

ASSIGN is used to perform an assignment lasting more than one command. Usually, a file is opened when a program is initiated, and closed when it is completed. Through the use of ASSIGN, a file will remain open until a release command is executed.

The command is simply

```
ASSIGN  assignments
```

Example:

```
ASSIGN  SO=:LP:
```

Programs may now be invoked without assigning SO.

3.20 RELEAS

RELEAS is used to close a file kept open by the ASSIGN command.

It is invoked by

RELEAS $logical unit list

where logical unit list is a list of logical units separated by comma.

Example:

RELEAS $SO,SL

3.21 ALLOC

(The specifications for this program is not yet complete).

3.22 EDIT

Edit is a sequential editor for use with TOS.

It is invoked by

EDIT SI=input file,SO=output file

The Editor copies information from the input file to the output file under control of commands from CI, the console input file.

With SI and SO are associated an input and an output buffer respectively. With each of the two buffers is associated a pointer. When the Editor is started, the first line is read from SI into the buffer, and it is displayed on CO as well. The position of the pointer (the next character) is also indicated on CO. Each time the Editor is waiting for a command, it will prompt the user by

ENTER COMMAND:

on CO.

The following commands are available:

B	set the pointers of the input and output buffer back to the first character in each. (Equivalent to BI and BO)
BI	set the pointer of the input buffer back to the first character (equivalent to a new scan of that line)
BO	set the pointer of the output buffer back to the first character (equivalent to clearing the output buffer)
C	copy the remainder of the input buffer to the output buffer, write the output buffer to SO, read next input line, clear output buffer
Cn	(0<n≤99) as C followed by a direct copy of the next n-1 input lines. (C1 is equivalent to C)
C+n	(0<n≤99) copy the next n characters from the input buffer to the output buffer. If the end of the input buffer is reached, the command is equivalent to C
C"text"	copy up to (but not including) the next occurrence of the specified text in the input file
DI	display contents and position pointer of the input buffer
DO	display contents of output buffer
H	end the editing, Remaining part of SO is skipped
I"text"	insert the specified text in the output buffer (The text cannot contain CR)
I	insert a number of lines in the output file (see "Insert mode" below)

S	skip the remainder of the current line. If the output buffer was not empty, the contents there is written out to SO as a line
Sn	(0<n≤99). Skip n-lines. Equivalent to S followed by Sn-1.
S+n	(0<n≤99) skip the next n characters in the input buffer. If the end of the input buffer is reached, the command is equivalent to S.
S"text"	skip up to (but not including) the next occurrence of the specified text on the input file
X"text"	(exchange) skip as many characters in the input buffer as there are characters in the specified text, and insert the characters of the text in the output buffer. If the end of the input buffer is reached, the text is inserted and the output buffer written on SO.

A command (but not the "text") may be written both in lower and uppercase letters.
Insert mode is entered when the I-command is given and when end of file is encountered on the input file.

In this mode, new lines may be entered one by one, each terminated by a carriage return. The keys ← and RUBOUT are available for editing the line, indicating backspace and delete line respectively → may be used to tabulate to the character whose number is the next multiple of 8.
CTRL C followed by
Carriage return indicates return from the insert mode to the normal edit mode. If insert mode was entered as the result of an end-of-file on input, a CTRL C followed by a carriage return indicates end of edit.

Examples on invocation of editor

1. Creating a new datafile.

   Invoke the editor by

   EDIT SI=:NF:,SO=output file

That the null file :NF: is used as input file, will cause the editor to enter insert mode immediately.

2. Update with listing.

The double-file feature can be used to produce a listing of the edited output:

EDIT SI=input file,SO=(output file,:LP:)

3. Two-file input.

The double-file feature can be used if more than one input file should be edited together:

EDIT SI=(infile 1, infile 2),SO=output file

4. Multiple input

If a large number of files should be edited together, it is more convenient to use the ASSIGN feature:

```
ASSIGN  SO=outputfile
EDIT    SI=file 1
EDIT    SI=file 2

EDIT    SI=file n
RELEAS $SO
```

3.23 XREF

(The specification for this program is not yet complete).

3.24 ASM

Calling sequence

ASM SI = input file, SO = hex file, SL = list file

The Assembler is a two pass assembler when used as specified above. It can be run on-line, as a one pass assembler if the SI specification is omitted. Input is then taken from console input. Code will be generated for each statement and stored in RAM.

The assembler is separately documented in "TDV 2100 Assembler - User's Manual".

3.25 HEXLST

HEXLST will list a file in hexadecimal form, 16 bytes per line in groups of 8 lines.
It is invoked by

```
HEXLST  SI = infile, SO = outfile
```

Example:

```
HEXLST  SI = FIL1, SO = :LP:
```

will list the file FIL1 on printer.

3.26 DCOPY

DCOPY will copy an entire diskette regardless of its contents. The diskette may not contained deleted data sectors or sectors with CRC errors etc.

The program copies from unit 0 to unit 1. Be careful so that your diskettes are in the proper unit.
It is invoked by

```
DCOPY
```

The program gives a message and wait for an operator reply before copying is started.

Any other reply than Y will cause return to TOS.

and S (SINGLE UNIT TOS 1.4)

3.27 DIRPAC

When a file is deleted through use of the DELETE program, its space in the diskette directory will not be used. This means that after a number of deletions, the directory might be full (200 entries), but with a number of deleted ones. The program DIRPAC will pack the directory so that the space might be reused.

It is invoked by

```
DIRPAC$ unit specification
or DIRPAC$
```

where unitspecification is of the form :Fx:. When the second option is used, unit 0 is assumed.

Example: To pack the directory on the diskette on unit 2:

```
DIRPAC$:F2:
```

4. Programming using TOS21

4.1 Storage layout

TOS21 occupies the high end of the memory address area while the PROM Monitor Kernel occupies the lower end of memory space. Fig. 1 shows the storage layout. The start of the RAM area available for user programs depends on the amount of RAM in the TDV 2100.

The start of TOS code is determined when the system is generated. Below that, buffers for i/o units are dynamically allocated and deallocated, so that the RAM available for user programs may vary according to space required for the assigned files.

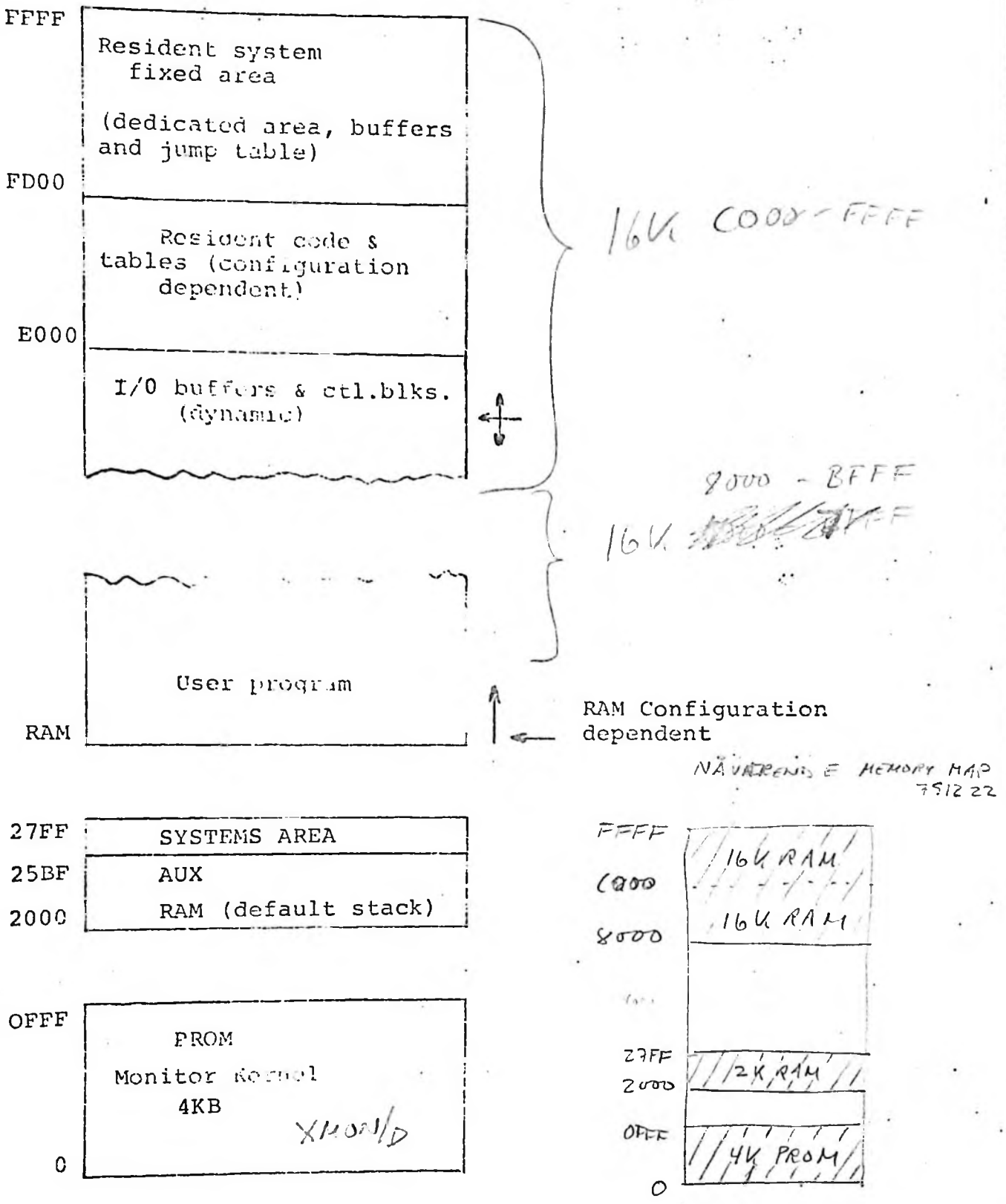


Fig. 1 Storage layout

4.2 Systems area

The area from FD00 to FFFF is reserved for systems use. The layout is shown in fig. 2

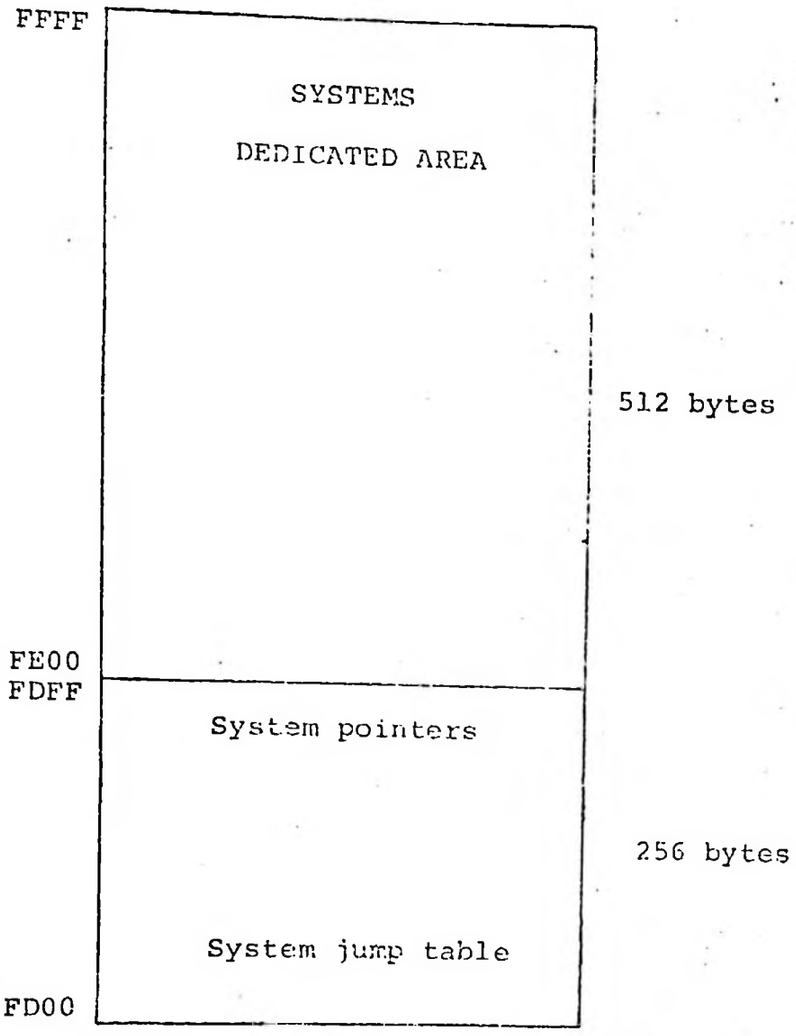


Fig. 2. Reserved system area

The area FF00 through FF0F contains the current assignments for the eight logical units:

FF00	CI
FF02	CO
FF04	SI
FF06	SO
FF08	SL
FF0A	AI
FF0C	AO
FF0E	AL

FD00 through FDFF contains a jump table which is used to access system functions, and system pointers.

FD00	- FD3F	User available TOS21 functions
FD40	- FDCC	Basic function jump table
FDCD	FDEB	Internal system functions
FDEC	- FDFF	System pointers

The addresses of the internal system functions and system pointers are shown in Appendix D. These functions are not intended for other purposes than systems programming.

The following TOS21 functions are available:

FD00	TOS21	FD03	ERROR	FD06	LOAD
FD09	INCHAR	FD0C	OUTCHAR	FD0F	GET
FD12	PUT	FD15	PUTSTR	FD18	GETBIN
FD1B	PUTBIN	FD1E	OPEN	FD21	READ
FD24	WRITE	FD27	CLOSE	FD2A	EDBH
FD2D	EDWH	FD30	EDHB	FD33	EDHW
FD36	BUFALLOC	FD39	BUFDEALLOC	FD3C	STIMER

The basic function jump table has the same structure as the jump table of the monitor kernel. For systems where the kernel contains both diskette and cartridge routine, the jump table will simply refer to the relevant address in the kernel jump table. For systems where only one of the devices are supported by the monitor, the other device may have its routines in the TOS21 RAM-part. References to the basic function table of TOS rather than directly to the kernel will ensure that programs may be run under other versions of the kernel without alteration.

4.3 Use of TOS21 functions

A TOS function is called by giving a CALL instruction specifying the jump table address of the desired function. The appropriate parameters must have been placed in registers prior to the call. Upon return, the carry flag will indicate whether the routine execution completed normally (carry cleared) or an error occured (carry set). In the latter case, the A-register contains an error code (see Appendix A). Note that when carry is set and A is zero, this indicates end-of-file (on input) or end-of-space (on output).

Routines involving input/output operations require as a parameter (in HL) the address of a file control block. (When normal assignment is used, the user need not bother with the control block formats, but a rough outline is given in Appendix B). A user will normally use a LHLD instruction to fetch the contents for the particular unit in the device assignment table (FF00-FF0F).

In the following, a brief description is given of the functions available in TOS. Unless otherwise specified, all registers are saved and restored except for those returning a result.

FD00 TOS21 Return to TOS21 command mode. May be entered by JMP or call as it does not return to user.
No parameters.

Note: Return to TOS should normally be done by a RET instruction as this will close all files opened as the result of an assignment. Entering TOS through TOS21 will cause the files to remain open.
When returning to TOS through a RET instruction, carry should be cleared for normal return, and set if an error occured. A is then assumed to contain the error code.

FD03 ERROR Display message ERROR xx on CO, and return to user if carry was not set on entry.

FD06 LOAD Load a program from an Intel-formatted diskette.
Parameters: as LOAD of monitor kernel.

The jump table entries from FD09 through FD27 contains the generalized i/o handling routines.

It is assumed that assignment to the logical units (CI, CO, SI, SO, SL, AI, AO, AL) have been performed by TOS21. The assigned logical units will then have been opened by the system.

The following functions are available:

- read the next character (INCHAR)
- write a character (OUTCHAR)
- read text up to next CR LF into a user specified buffer (GET)
- write a text from a user specified area up to CR (PUT, PUTSTR)
- read a specified number of characters into a user specified buffer (GETBIN)
- write a specified number of characters from a user area (PUTBIN)
- read a block (READ)
- write a block (WRITE)
- close a block (CLOSE)
- (- open a file (OPEN), only used when open was not performed by system.)

All these routines requires a pointer to a file control block in HL. For the normal files, and assuming equivalences defined previously, this pointer may be conveniently loaded by the LHLD instruction.

	Routine	Input parameters	Result
FD09	INCHAR	HL - file control block	A-character If carry set, error or eof
FD0C	OUTCHAR	HL - file control block	If carry set, error or eoe
FD0F	GET	HL - file control block DE - user buffer	If carry set, error or eot. DE altered.
FD12	PUT	HL - file control block DE - user buffer	If carry set, error or eoe. DE altered.
FD15	PUTSTR	as PUT	
FD18	GETBIN	HL - file control block DE - user buffer B - count	If carry set, error or eof. DE altered.
FD1B	PUTBIN	HL - file control block DE - user buffer B - count	If carry set, error or eoe. DE altered.
FD1E	OPEN	HL - file control block A - open code (1=input, 2=output)	If carry set, error
FD21	READ	HL - file control block	If carry set, error or eof
FD24	WRITE	HL - file control block	If carry set, error or eoe
FD27	CLOSE	HL - file control block	If carry set, error

The difference between PUT and PUTSTR is that PUT will insert CR LF while PUTSTR will not.

FD2A EDBH — Edit byte hex

Parameters: A-Byte, HL-address of two byte receiving area where edited hex value (in ASCII) is stored.

FD2D EDWH — Edit word hex
Parameters: DE-word, HL-address of four byte receiving area.

FD30 EDHB — De-edit hex as byte
Parameters: HL-address of two byte area.
Result: A-byte.

FD33 EDHW — De-edit hex as word
Parameters: HL-address of four byte area.
Result: DE-word.

FD36 BUFALLOC — Allocate a dynamic buffer
Parameter: HL-desired size
Result: HL-address of buffer.

FD39 BUFDEALLOC — Deallocate a dynamic buffer
Parameter: HL-address of buffer.

FD3C STIMER — Set interval timer
Parameters: DE-desired time interval in 20 ms units, HL-address where control should be passed on expiration of the limit.

4,4 User handling of interrupt under TOS.

When an interrupt occurs, the monitor will execute a fixed sequence of instructions. For interrupts on the following levels, the user may gain control after the standard sequence by storing the address of his interrupt routine in standard locations.

Level	Interrupt	Location to store address
2	Sync. interface	027A4H
4	Cluster interface I	027A8H
5	Cluster interface II	027AAH

These locations are cleared to zero when power is turned on, and if non-zero exit will be made to the user when the interrupt occurs. The user is responsible for saving and restoring of all registers, and must enable interrupts prior to executing a RET instruction to exit from the interrupt routine.
Example: If we in a (assembly) program wants to capture interrupts on level 4, and when an interrupt occurs enter a routine named INTR4, we might use

```
LXI  H,INTR4
SHLD 027A8H
```

All interrupts occurring on level 4 after this sequence of instructions has been executed, will cause the interrupt routine INTR4 to be entered.

The clock interrupt (on level 1) each 20 ms is not directly available to the user. The user has, however, available an interval timer feature, which permits him to request an interrupt after a specified number of 20 ms. periods.

Under TOS, a routine call STIMER is available to set the interrupt condition.

If the monitor only is used, the interrupt address is stored in 27BOH, the desired interval(as a word) in 27BDH. Note that the modification if these two location must be done with interrupts disabled.

Interrupts on level 3 are passed on to the user only if they are relevant for the UART and is not 'data available' or 'transmitter buffer empty'. The address of the user interrupt routine must be stored on location 27A6H.

Interrupts on level 6 (diskette) and 7 (cartridge) are in general handled by the monitor. If an interrupt address is provided for one of these levels, control will pass to the user only when the complete operation (read, write etc.) are completed, and not when seek, retry etc. will be done.

Interrupts on level 0 cannot be captured by the user.

5. USAGE HINTS

5.1 How to patch a program

To patch a program and store it (under a new name) the following method might be used:

1. Load TOS (if not already running)
2. Use the TOS command
   DEBUG$program name
   (without any fileassignments)
3. Patch the program using monitor commands
4. Restart TOS by R0 command
5. Execute the MEMDMP program with suitable assignment and parameters.

5.2 How to reopen an input file

(As an example, the code sequences are shown for SI).

```
        LHLD    SI
        INX     H
        MOV     A,M
        ANI     20H
        MVI     A,44H
        JZ      L1
        MVI     A,67H
L1:     ANA     M
        MOV     M,A
        INX     H
        INX     H
        INX     H
        MVI     M,128
        INX     H
        MVI     M,0
        MVI     A,1
        LHLD    SI
        CALL    OPEN
        JC      .....
```

Note that if a file control block should be reused, the BUFFER EXT field must be reset prior to opening the file. Note further that this method will not work for a double file assignment, e.g. SI=(ABC, XYZ)

Appendix A

SYSTEM ERROR CODES.

Upon exit from a system routine, the carry bit will be set if an error occurred within the routine, cleared otherwise. The A-register contains an error code.

Hex value	
00	End-of file or end-of-extent
01	Sector missing on diskette
02	CRC error on diskette
03	Timing error during read/write data transfer.
04	No address mark
05	Diskette operation timeout
06	Timing error in read operation
07	Busy-bit timeout
08	Drive not ready
09	Name not in directory and no room for it.
OA	Name not in directory
OB	Program name not found
OC	Illegal directory entry
OD	Directory size error during load
OE	Data size error during load
OF	Diskette map cannot be located
10	Non-valid diskette command
11	Improper unit specification
12	Conversion table missing
13	Drive no. not 0-3
14	(Number not in use)

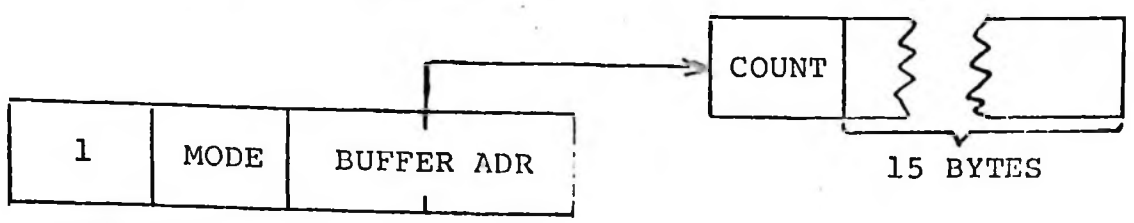
15	De-editing of non-hex digit
16	Input attempted from non-input file
17	Attempt to read after end-of-file
18	Attempt to write on non-output file
19	Unassigned device
1A	Unknown device type
1B	Command or filename did not start with letter or digit
1C	Command or filename longer than 6 characters or extension longer than 3 characters
1D	Command did not refer to Intel diskette or Tandberg formatted cartridge as device
1E	TOS not loaded
1F	Command not terminated by CR, $ or space
20	Deleted data record
21	Logical unit name unknown
22	Write attempted on filetype other than 5
23	No write or read routine supplied in control block
24	Read attempted on filetype other than 4 or 5
25	Line overflow for filetype 4
26	(Number not in use)
27	File already open
28	No open routine supplied in control block
29	File was not open when close was called
2A	No close routine supplied in control block
2B	Buffer chain destroyed
2C	Buffer header improper
2D	Improper file designator
2E	Improper assignment format
2F	No more space on diskette
30	(Nomber not in use)
31	Non-existent filename for input file
32	Output file already on diskette
33	No more space on diskette
34	Filename longer than 8 characters
35	Track no > 76
36	Sector no=0 or > 26
37	Buffer address zero
38	Routine not available
39	Cartridge directory block not 32 bytes
3A	Volume header missing
3B	Directory block error

3C	Output file already on cartridge
3D	No more space on cartridge
3E	Input file overrun
3F	Extra or misplaced tape mark
40	End of tape error
41-4F	I/O error on cartridge
41	Cartridge CRC error
42	Fatal cartridge error
43	41 + 42
44	Non valid cartridge command or write protected
45	41 + 44
46	Reading erased tape
47	41 + 42 + 44
48	Tape mark detected
49	41 + 48
4A	42 + 48
4B	41 + 42 + 48
4C	44 + 48
4D	41 + 44 + 48
4E	42 + 44 + 48
4F	41 + 42 + 44 + 48
50	Illegal unit number (cartridge)
51	Illegal track number (cartridge)
52	Block size illegal (cartridge)
53	Track change error (cartridge)
54	Cartridge unit not ready
55	Illegal buffer address (cartridge)
56	Cartridge not supported
57	Extra blocks in cartridge file
58	Continuation header missing

Appendix B

FORMAT - FILE CONTROL BLOCKS

CHARACTER DEVICE (INTERRUPT)



Mode values:

1 - opened for input
2 - opened for output
4 - do not suppress LF character
10H - lbbit (next read will give end-of- file or end of extent)
20H - auto (file was allocated by the assignment interpreter and will be closed, and buffer space released, when return to monitor).
40H - don't close (file should not be closed).
80H - eofbit (end of file/end of extent has been detected).

CHARACTER DEVICE (DEMAND)

2	MODE	INCHAR	OUTCHAR

INCHAR and OUTCHAR are subroutine addresses. If input file, INCHAR is expected to return with character in A and carry cleared. Carry is assumed set if an error occured.

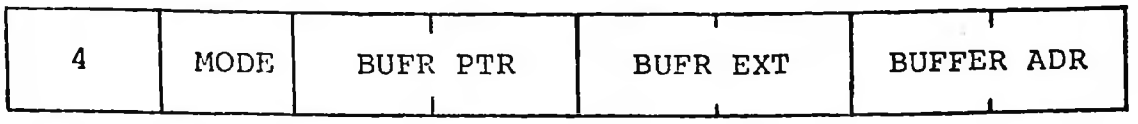
If output file, OUTCHAR will be called with character in A. All registers (except PSW for INCHAR) should be saved and restored. Normally, only one of the fields contain an address.

ECHO DEVICE

3	MODE	INPUT DEV.	OUTPUT DEV.

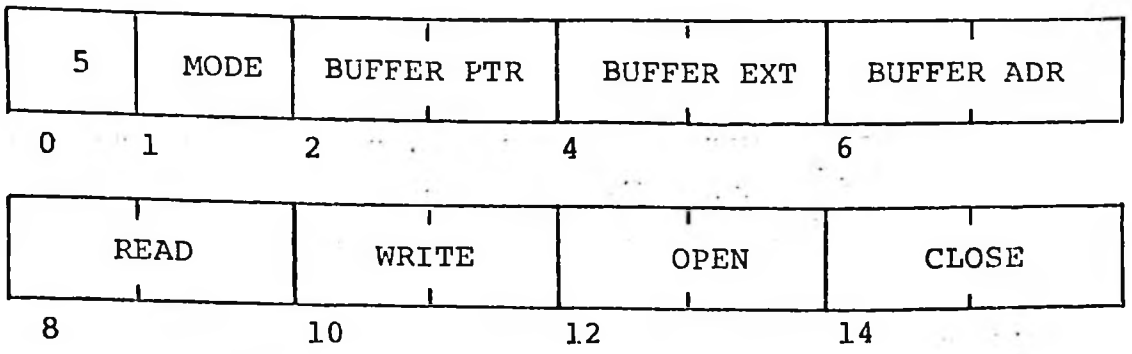
INPUT and OUTPUT DEV are addresses of other control blocks. As character is read from the input device, it will be echoed on the output device.

LINE ORIENTED DEVICE



BUFR PTR	character count showing next character in buffer (starts on zero)
BUFR EXT	length of buffer
BUFR ADR	address of buffer
CHAR OR BLK DEV	pointer to another control block which will be called to receive characters

BLOCK ORIENTED DEVICE



DEV TYPE 0 - diskette
1 - cartridge

BUFFER PTR Relative position within buffer of next character

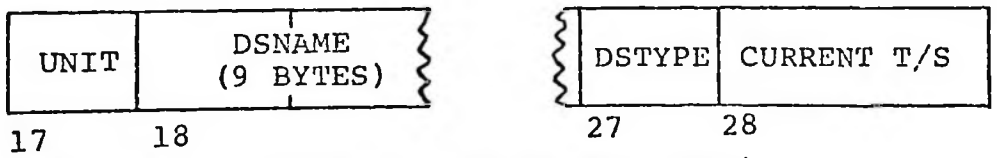
BUFFER EXT Size of buffer

BUFFER ADR Address of buffer

READ, WRITE, OPEN, CLOSE: Device dependent routine addresses to read, write, open and close. Normally, only one of read or write is non-zero, and open and close addresses depend on whether it is an input or output file

DEV TYPE device type
0 - diskette
1 - cartridge

DISKETTE: (PART 2)



UNIT physical unit number (0-3)

DSNAME dataset name on diskette. For INTEL-formatted diskettes, the first 6 characters are the name, the last 3 extent. For IBM-formatted diskettes the first 8 characters are the name. The 9th is reserved for future extension (accesscharacter)

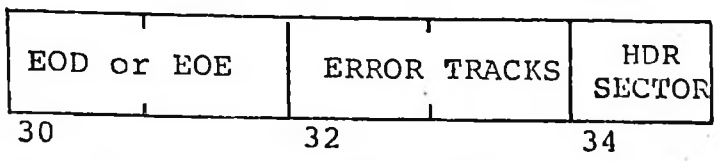
DSTYPE
80H EBCDIC
40H 512 bytes ≠)
20H 256 bytes ≠)
10H not used
8 binary seg ≠)
4 set S and I flags
2 IBM
1 INTEL

≠) Future extension

CURRENT T/S Sector and track for current directory (INTEL-format) or next record (IBM-format).

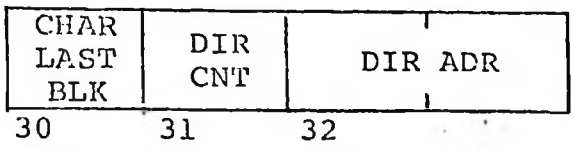
(PART 3):

IBM-MODE



EOD or EOE Sector and track for last block of input data set or last allocated block of output data set

INTEL-MODE



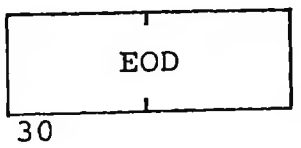
CHAR LAST BLK number of information characters in last block (input file)

DIR CNT relative pointer to directory buffer, showing sector and track of next data block

DIR ADR address of directory buffer

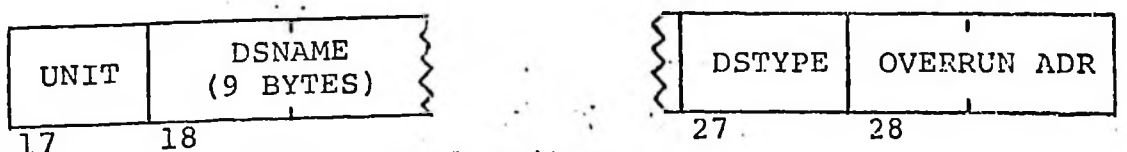
BINARY SEG*)

*) Future extension



EOD Sector and track for last block of data set

TAPE CARTRIDGE: (PART 2)

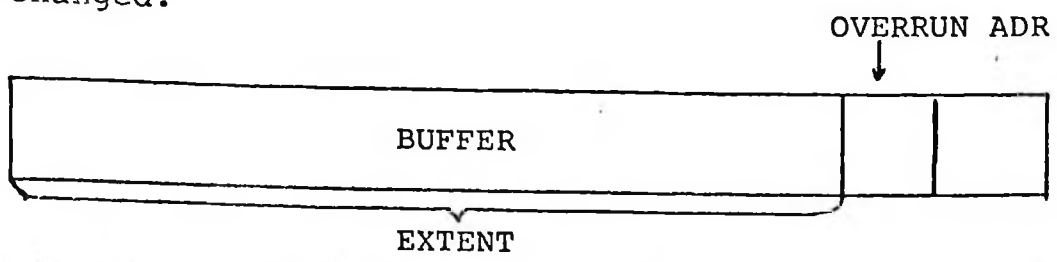


UNIT is physical unit no.

DSTYPE:

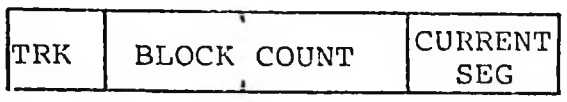
80H	EBCDIC
40H	not used
20H	
10H	
8	
4	
2	ECMA format
1	TANDBERG format

OVERRUN ADR is used to check for read overruns. Two bytes with special values are placed <u>after</u> the buffer before read, and it is checked after read to see that they have not been changed:



(PART 3):

TANDBERG FORMAT



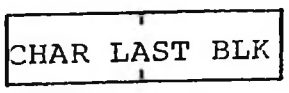
TRK track currently being read from/written to

BLOCK COUNT
is incremented for output files, decremented for input files

CURRENT SEG
indicates the number of the segment currently being read/written.

(PART 4):

TANDBERG FORMAT, INPUT



CHAR LAST BLK
indicates the number of information characters in the last block.

TANDBERG FORMAT,OUTPUT

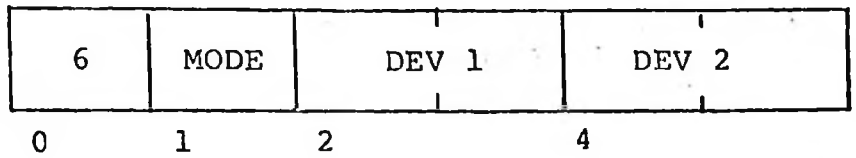
FIRST TRK the number of the track where the first segment of the file was written

FILE NO the file number of the first segement on that track

TRACK USE INFO

is a copy of bytes 1-15 of the Cartridge Usage Block (see appendix G).

DOUBLE DEVICE



DEV1 and DEV2 are pointers to other control blocks (both input or both output files).

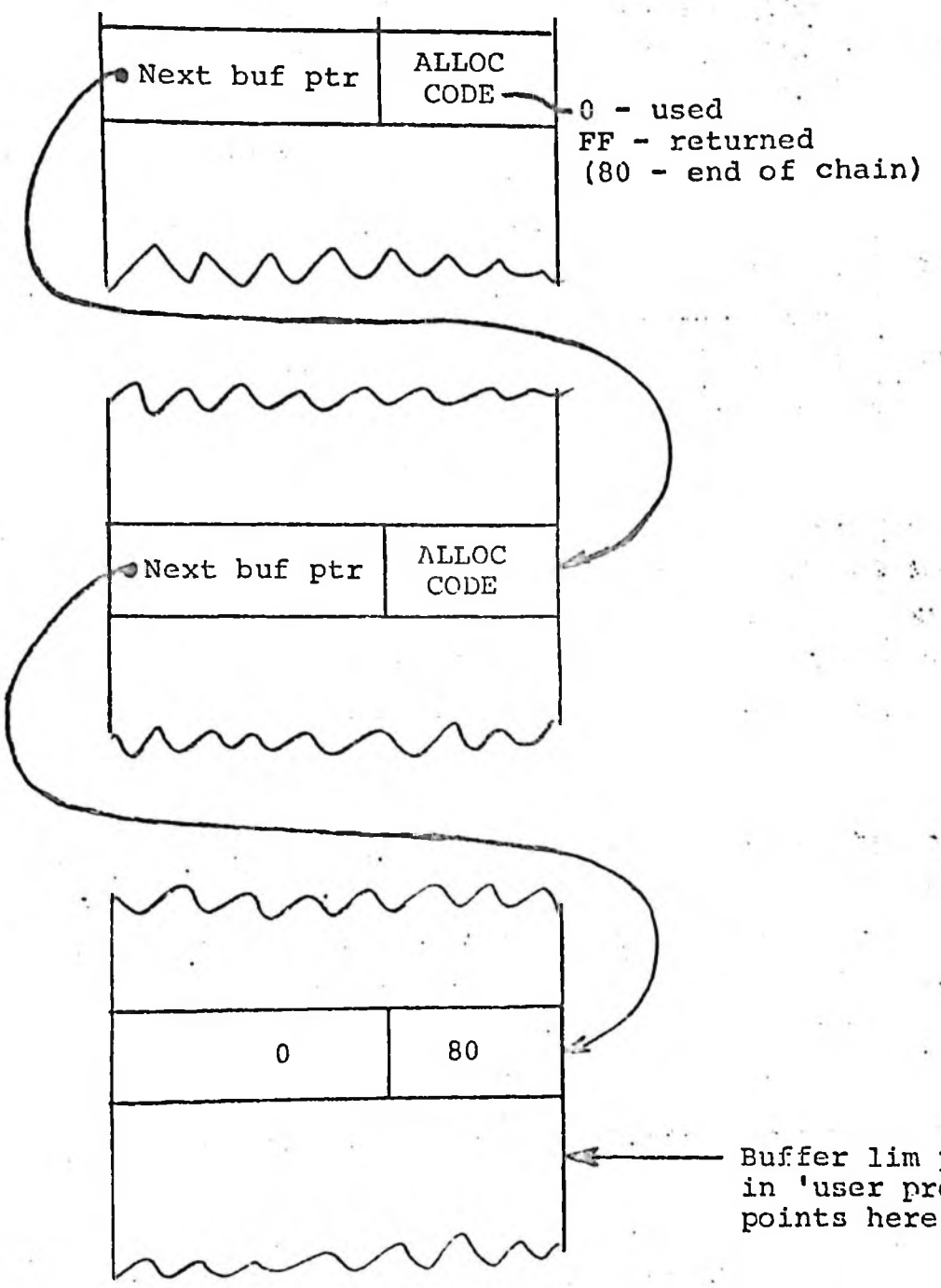
Appendix C

Internal formats

The formats shown here are for documentation purposes only. A user should never alter these areas, since to do so properly, requires knowledge of the internal operation of the TOS system.

The formats are further subject to change without notice.

BUFFER AREA


Next buf ptr
ALLOC CODE
0 - used
FF - returned
(80 - end of chain)
Next buf ptr
ALLOC CODE
0
80
Buffer lim ptr in 'user prog blk' points here

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
27B8	DAY CTR (0)	HR CTR ≠ (24)	MIN CTR ≠ (60)	SEC CTR ≠ (60)	20 MS CTR ≠ (50)	Interval Counter (0)		Diskette timer (0)	TIMER DATA
27B0	INTERVAL TIMER INTERRUPT ADR (0)					YEAR (0)	MTH (0)	DAY OF MTH (0)	

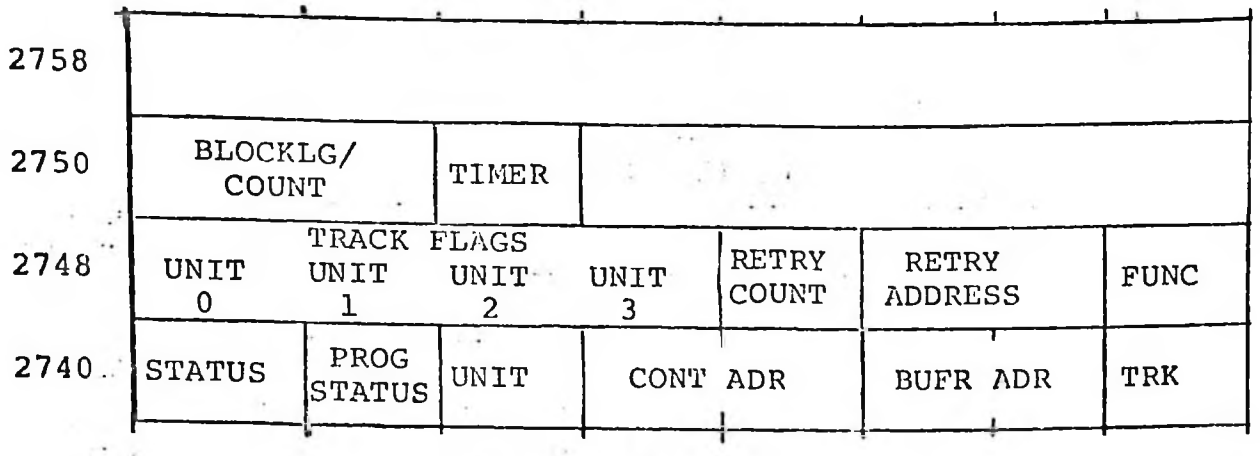
Interval counter max = 1310.72 sek≈21 min ≠) Reversed count (down)
( ) Bootstrap value

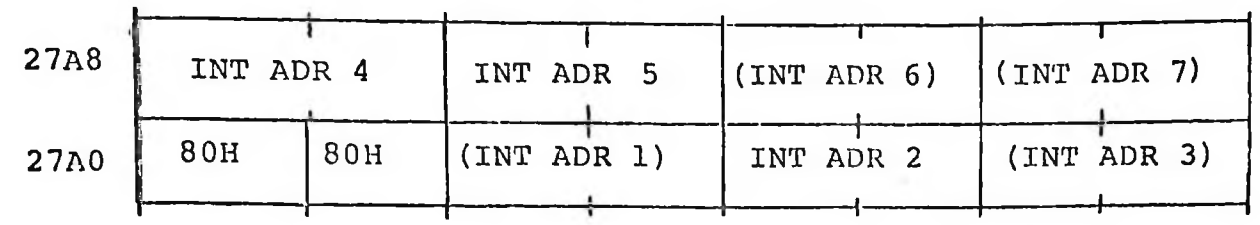
2768	SECT	FUNCTION	CONV	LG CODE	RETRY COUNT	RETRY ADDRESS	(RESERVED)	Diskette ctl
2760	STATUS	PROG STATUS	UNIT	CONT. ADR.		BUFR ADR	TRK	

2738	B COUNT	MAP 1	MAP 2	AUTO LOAD MSG SWITCH	UNAM	System work area 1
2730	UNIT	TRK	NAME	DCNT	BADR	

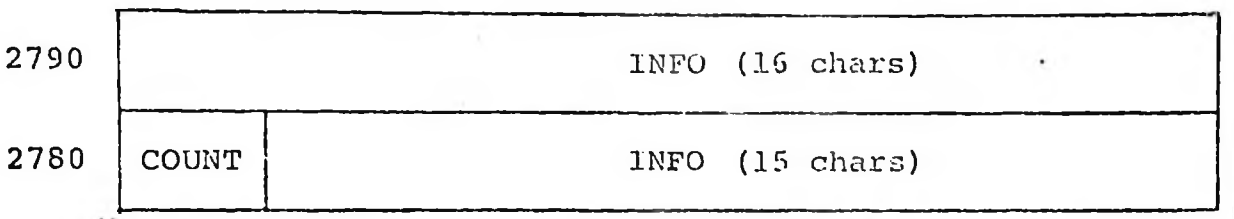
2718	FILE EXT	CISAVE	COSAVE	ROLL SWITCI	System work area 2
2710	CTL BLK	FILE NAME			

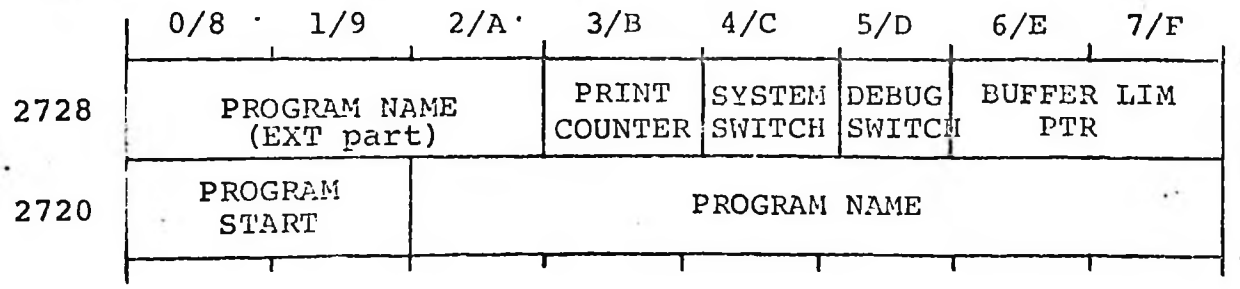
2778	Diskette header buffer
2770	Buffer for header read-diskette

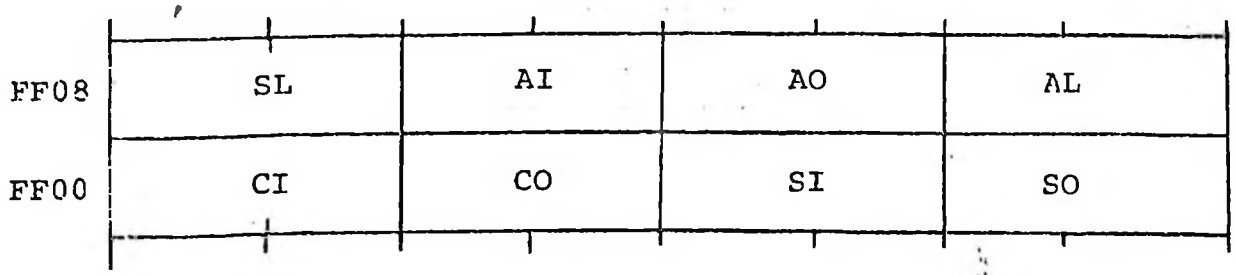


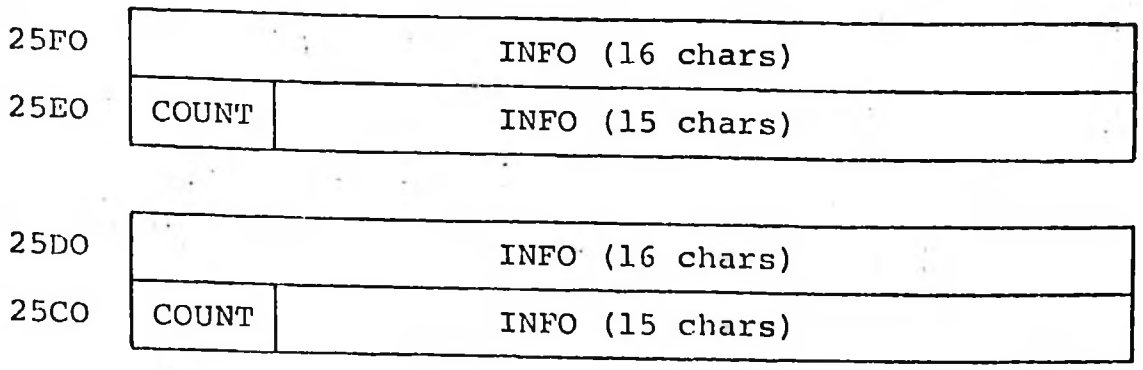


( ) normally not used









Appendix D

Internal system functions

The following addresses are assigned to internal system function jump entries and addresses

Jump entries:

FDCD	FILENAME
FDD0	IBFIL
FDD3	RELN
FDD6	RELBLK
FDD9	GETN
FDDC	GETBLK

Addresses:

These refers to separate six entry (18-byte) jump tables to handle file functions in different format. These have the general format

0	Open input file
3	Open output file
6	Read input block
9	Write output block
0C	Close input file
0F	Close output file
FDEA	Intel-format diskette table
FDE8	IBM-format diskette table
FDE6	Tandberg-format cartridge table

The following four routines assumes that the unit number is stored in UNIT (2730) and that MAP1 and MAP2 (2739, 273B) contains track/sectors of map on this unit.

FDD3	RELN	Release a number of diskette sectors A-number HL-address of list of sectors
FDD6	RELBLK	Release a diskette sector BC-track/sector
FDD9	GETN	Allocate a number of diskette sectors A-count HL-desired address of first entry
FDDC	GETBLK	Allocate a diskette sector BC-track/sector

The location of system pointers are:

	0/8 1/9	2/A 3/B	4/C 5/D	6/E 7/F
FDF8	EBCDIC	ASCII	RAM MEM START	SYSTEM START
FDF0	:NF:	:BR:	:CO:	:CI:
FDE8			:RD:	:PR:

Appendix E

Intel diskette format

Diskette layout:

Track	Sector	
0	1	Available for data. (may be used for system program)
0	25	
0	26	Diskette volume label
1	1	Diskette directory
1	26	
2	1	Diskette block usage map
2	3	
2	4	Available for data
76	26	

Diskette volume label

Char 1-6 Volume name.
Char 7-9 Name qualifier.

Diskette directory

Track 1 Sector 1 Directory block for directory dataset
Track 1 Sector 2 through sector 26: Diskette directory.

Each sector consists of eight 16 byte entries with the following possible formats:

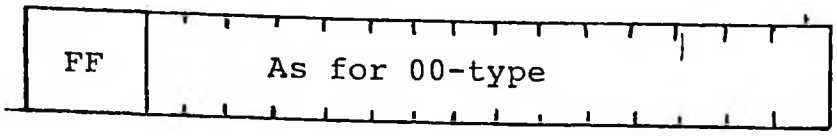
1	7	10	11	12	14
0	NAME	EXT	FLAG	NB	NBLK-1	DSDIR

Dataset is present

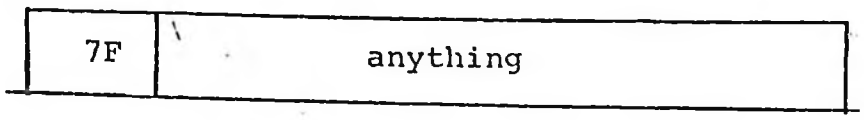
NAME - dataset name
EXT - ext part of name
FLAG - flags (for meaning, see Intel documentation)

80H F ← 8 A (Alias, used by TOS21 only)
4 W
2 S
1 I

NB - number of data bytes in last block
NBLK-1 - number of data blocks minus 1
DSDIR - sector and track of first dataset directory.



Deleted dataset entry
(The information in the last 15 bytes provide history only, since the space for the dataset has been released).



Unused directory entry

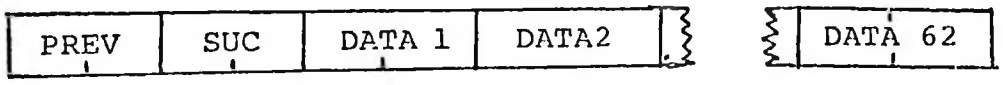
Diskette block usage map

Track 2 Sector 1 Directory block for Map data set.
Track 2 Sectors 2 and 3: One bit corresponding to each sector on the diskette,

Sector 2 contain the map for track 0 sector 1 through track 39 sector 10 while sector 3 contains the remaining part of the map.

Dataset directory.

A dataset directory block tha the following format:



PREV - sector and track of previous directory block (if any)
SUC - sector and track of next directory block (if any)
DATA1 - sector and track of 1st. data block belonging to this directory block.

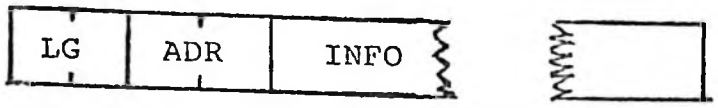
⋮

DATA62- sector and track of last (62nd) data block belonging to this directory block.

Note: If less than 62 datablocks belong to this directory, remaining entries are zero.

Intel load format

An executable program consists of a sequence of segments, each with the format



LG - length of INFO in this segment

ADR - address where info should be loaded

LG = 0 indicates that this is the last segment of an executable module. ADR indicates in this case the starting address.

Appendix F

IBM diskette format

(For more comprehensive information refer to "The IBM Diskette General Information Manual." published by IBM)

Diskette layout

Track	Sector	
0	1	Reserved
0	4	
0	5	Error map
0	6	Reserved
0	7	Volume label
0	8	Data set labels
0	26	
1	1	Available for data
74	26	
75	1	Alternate track 1
75	26	
76	1	Alternate track 2
76	26	

In the following, only the parts relevant to use of IBM-format diskettes with TOS has been included. Positions not specified must be assumed to be reserved.

Error map

Track 0 Sector 5

Positions 1-5: ERMAP

7-8 : Blank or number of first defective track (decimal)

11-12: Blank or number of second defective track (decimal)

Volume label

Track 0 Sector 7

Positions 1-4	:	VOL 1
5-10	:	volume identifier
11	:	volume accessibility field *)
38-51	:	owner identifier field *)
76	:	physical sector length*) of tracks 1 through 7 blank - 128 bytes 1 - 256 bytes 2 - 572 bytes
77-78	:	physical record (sector) sequence code*)

*) will probably be used in future extensions.

Data set label

Track 0 Sectors 8 through 26

Positions 1-4	:	HDR1
6-13	:	Data set identifier
29-33	:	Beginning of extent (BOE) First sector allocated to dataset 29-30 track number 32-33 sector number
34	:	Physical record length *) See position 76 of Volume label
35-39	:	End of extent (EOE) Last sector reserved for this dataset Format as BOE
42	:	Data set security *) Access character.
43	:	Write protect *) blank - not protected P - protected
48-53	:	Creation date YYMMDD
54-57	:	Record length *)
67-72	:	Expiration date YYMMDD
75-79	:	End of data (EOD) Address of next unused sector Format as BOE

Appendix G

TANDBERG tape cartridge format for use with TOS21

The tracks are numbered 0-3. Track 0 is used for the Cartridge Directory. Tracks 1 through 3 are used for data. Assuming 128 byte blocks, this gives a total data capacity for each cartridge of approximately 750,000 bytes.

Track 0 contains:

- 1 volume header block
- one file directory block for each file on the cartridge (whether deleted or still present)
- 1 cartridge usage block
- Tape mark.

Tracks 1-3 contains each a sequence of file segments. The format for each segment is

- 1 file header block
- a number of data blocks
- Tape mark
- 1 file trailer block
- Tape mark

After the last file segment of a track:

- 1 track end block
- Tape mark

Normally, a file segment and a file will correspond. The exception is when the end of a track is encountered during output of a file. In this case, the output is continued on another track if there is room on any of them. A special continuation file trailer and file header will be used to ensure proper reading of the file. Note, however, that continuation is time consuming both during input and output, and should be avoided whenever possible.

The block size may be chosen by the user, but is minimum 32 bytes, maximum 2048 bytes. The different types of control blocks (volume header, file directory, cartridge usage, file header, file trailer, track end) are of fixed size 32 bytes.

The format of the different types of control blocks are:

Volume header block:

Byte	
0	88H
1-9	Volume name
10-31	(Unused)

File directory block

Byte	
0	0-file present 0FFH-deleted
1-9	Filename
10	Track number (of first file segment)
11	File number (of first file segment)

File directory block

Byte	
12-13	Block size (min. 32, max. 2048) (least significant 8 bits in pos. 12)
14-15	Bytes in last block (least significant 8 bits in pos. 14)
16-17	Number of blocks in file (least significant 8 bits in pos. 16)
18	Attribute byte (see appendix E, "FLAGS")
19-31	(Unused)

Cartridge usage block:

Byte	
0	7FH
1-5	Usage information track 1 1- Number of filesegments 2-3- Number of blocks on track (least significant 8 bits in pos.2) 4- Track full indicator: ≠ 0 if no more room on trk. 5- Unused
6-10	Usage information track 2 (Same as track 1)
11-15	Usage information track 3,(same as track 1)
16-31	(Unused)

File header block

Byte	
0	'(' if first segment '-' if continuation
1-9	Filename
10	Segment number (first=0)
11-12	Block size
13-31	(Unused)

File trailer block

Byte	
0	')' if last segment '+' if continuation
1-9	Filename
10	Number of next segment(0 if last)
11	Continuation track (0 if last)
12-31	(Unused)

Track end block

Byte	
0	'/'
1-31	(Unused)