Microtan TANDOS-R Manual

Appendix A - TANDOS Commands

Table of Contents

Appendix A - TANDOS Commands....1
BACKUP....2
COPY....2
Copying files....2
Wildcards....2
Superseding....3
Protection Attribute....3
Single disk drive copying....3
Merging Files....4
Memory Usage....4
CREATE....5
DBASIC....5
DEL....5
Delete Single File....5
Wildcards....5
DIR....6
Full Directory Listings....6
Paging....6
Selective Directories....7
Using Wildcards....7
DLOAD....7
DLOAD Parameters....7
Auto Load....8
DRV....8
DSAVE....9
DSAVE Parameters....9
DZAP....9
Command Summary....9
CONTROL COMMANDS....10
General Comment....11
FORMAT....11
INIT....12
KILL....12
REN....13
S....13
Errors....13
SYS....14

BACKUP

A disk copy utility
Syntax: BACKUP < CR >
Anatomy: Starts @ $B960. Ends @ $BB9B. Execs @ $B960.

This is a useful utility for single drive owners. It will copy whole disk contents without the need to do it file by file. It will first request the user inserts the source disk, then after reading as much data as available RAM allows, requests the user to insert the destination disk. This continues until the whole disk has been copied. Caution as it will write over anything already on the destination disk.

COPY

A file copy utility
Syntax: COPY n:DESTFN.EXT< m:SOURCE.EXT [S P N C M]< CR >
Anatomy: Starts @ $400. Ends @ $B10. Execs @ $530

The COPY command is a powerful command which permits copying of files from disk to disk. By using the various options it is possible to make copies of files on different disks, even if there is only one disk drive available. Options to delete old versions of files (supersede), merge files, change the write protection attribute are also available. Extensive wildcard facilities increase the usefulness of the Copy command even further.

Copying files

The simplest requirement is the transfer of a file from a disk on one drive to a disk on another drive. The command syntax for this is simply:-

COPY n:DESTN.EXT< m:SOURCE.EXT < CR >

The first file specification is the file destination and the second is the source. n and m may be the same, or omitted (default is taken to be the current drive). If n and m are the same then the filenames must be different. In the event that you require the new file to have the same name as the old (must be different disks) then you may omit the destination filename e.g.

COPY 2: < 0:FILNAM.EXT< CR >

makes a copy of FILNAM.EXT on the disk in drive 2. The user is notified when a file has successfully been created on the destination disk.

Wildcards

Wildcards may occur only on the source (right hand) side of the '<' symbol. There are no other restrictions. Thus:-

COPY 2:< 0:*.* < CR >

will copy everything from drive 0 to drive 2. Once again, the user is notified whenever a file is created on the destination disk. This is especially useful when using wildcards as the full

name of each file is printed permitting the user to follow the operation of the COPY instruction. Sometimes when using wildcards and you have true 'complex load-modules' on the disk you are copying from, the copy will fail and produce the message 'Insufficient space etc.' for one of the un-titled load modules. The reason it can't tell you what the title of the load module is, is quite simply because it doesn't have one! What has probably happened is that it has failed on one of the load modules after the first one which is the only one really named by the directory. There is really no way around this problem because of the way COPY treats a 'load module' when in the process of copying but it is as well to be aware of it.

Superseding

It is sometimes necessary to delete an old version of a file and create a new version with the same name. This type of operation is simplified by the use of the S parameter. The command:

COPY m:JOE.EXT< n:FRED.EXT S< CR >

will generate a file named JOE.EXT on disk drive m from a source file FRED.EXT on disk n. If a file called JOE.EXT already exists on drive m then this will be deleted before the new file is created. The message JOE.EXT SUPERSEDED is displayed as a reminder. Files cannot be superseded if their write protect attribute is set (since this prevents their deletion).

Protection Attribute

Each file on a disk has a protection attribute. If this is set to 'N' (by default or by REN) then the file may be deleted. This status is shown in the directory listing by the absence of the 'P' attribute. If the attribute is set to 'write protect' ('P') then the file cannot be deleted by the DEL disk command or overwritten by any of the 'save' or copy disk commands (beware BACKUP, CREATE, FORMAT and INIT - these will completely erase a disk!). COPY normally assumes that files created by it should have the same protection attribute as the original file. If you wish to override this then you may specify N or P as parameters to any Copy command e.g.

COPY m:< n:*.* P < CR >

will copy all files from disk n to disk m and set the write protect attribute on all of them. It does not affect the write protect status of any files on disk n,

Single disk drive copying

COPY caters for operations which have to be performed on a single disk drive. It will prompt the user for Source and Destination disks to be loaded as necessary throughout the execution of any COPY command whenever the C parameter (C for 'Change disk') is issued as part of that COPY command e.g.

COPY n:FRED< n:JOE C < CR >

Note: Source and destination Drives need to be the same.

Command execution proceeds in the following manner:-
1. The COPY program is loaded from the system disk,
2. The user is prompted: "LOAD SOURCE disk & PRESS RETURN".

3. The source file is read.
4. The user is again prompted but with "LOAD DESTINATION disk & PRESS RETURN".
5. The source file is written to disk.

There will be at least one read operation (from the source disk) and one write operation (to the destination disk) for every file copied. If the file is too long to fit in memory at one time then multiple read/write operations are needed. COPY tries hard to prevent you getting the disks confused. It checks the disk names (see INIT) every time you load a disk and warns you if it is not the disk it was expecting. This check is rendered useless if both disks have the same name, (it warns you about that too) hence the plea to use sensible names when CREATE or INIT is used.

Merging Files

The user may occasionally wish to merge several files together into one larger file. This may readily be accomplished by using a COPY command with the M parameter. Other than disk size, there is no limit to the number of files which may be merged together. The command:

COPY n:FRED< m:JOE??? M < CR >

will merge all files with the first 3 letters of the filename being JOE on disk m into a single file called FRED on disk n. Note that merge operations require a filename on the destination side of the "<". COPY will inform the user of the progress of the command with a message each time the output file is written to (either its creation or for a file appended to it). FRED may or may not exist before the command is issued but it must not have the write protect attribute to set to 'P'. On completion of the above command COPY prompts with "NEXT:". If you want to append further files then simply type a new line specification at this point e.g.

NEXT: GEORGE.* < CR >

COPY will continue to merge and prompt with "NEXT:" until the user responds with only < CR >. One of the main uses for merged files is to create complex 'load modules' i.e. files which load and execute. You can use the DSAVE command to save various areas of memory separately (zero page, TANRAM, or even the VDU screen). See example below. After merging with COPY then all of these areas will be loaded at one go merely by issuing the appropriate DLOAD or Auto-load command. Note that the transfer address (see DLOAD) is taken from the first part of the composite file only. It is perfectly legal to merge a file with itself (though it could be argued that it is rather a strange thing to do). COPY will perform correctly provided that the source file(s) fit in memory. If multiple copy operations are necessary then the operation will eventually fail with a message explaining that the disk is full (it won't be listed when you do a DIR - the partially formed file is automatically removed and no harm is done to the disk). This is an inevitable consequence of the source file getting longer every time you merge a bit onto it.

Memory Usage

COPY uses memory from location $400 upwards (unlike nearly all the other TANDOS 65 commands). It will use almost all the available RAM during COPYing. If you have extension RAM (eg TANRAM) then copying long files will be quicker.

CREATE

Creates a new disk in Drive number n:
Syntax: CREATE n:
Anatomy: Starts @ $400. Ends @ $7E6. Execs @ $5A0.

This provides both the FORMAT and INIT utility operations in a single application. The Drive number must be specified. In all cases the user will be prompted to insert the disk in the specified drive. A listing of a version of CREATE is available in TANDOC 3, Page 15.

DBASIC

Loads and runs disk-based BASIC
Syntax: DBASIC
Anatomy: Starts @ $B960. Ends @ $BA38. Execs @ $B960.

If the standard BASIC ROMs are installed, then the enhanced BASIC incorporating the various disk commands may be invoked by using the DBASIC command. This loads and runs a short initialization program from the system disk which patches a RAM resident subroutine in the zero page, and then jumps into the existing BASIC ROMs. DBASIC adds a further 4 commands and 6 new BASIC statements to the ROM version. Further details are available in the TANDOS Manual Chapter 4.

DEL

Delete a file.
Syntax: DEL [n:]FILNAM.EXT < CR >
Anatomy: Starts @ $B960. Ends @ $BBDC. Execs @ $B980.

The DELete command is used to erase files from the disk. The space previously occupied by the file itself and its directory information becomes available for use in storing other files. DEL will not operate on any disk which has the write protect tab fitted (delete is treated as a write operation). You will also be prevented from deleting any file with the 'Protect' attribute set (giving a 'P' in the directory listing after the file length). You may change the protection attribute with the REName command. Alternatively use the KILL utility.

Delete Single File.

If the delete command is used in conjunction with an explicit file specification (no wildcards) then only that file is deleted, (assuming it exists and is not protected, of course).

Wildcards

Delete may be used with wildcards. In this case the directory is searched for all files which match the specification. For each of these, the full filename is displayed and a yes/no prompt given. In the latest version, you do not have to press return after answering Y or N.

DIR

Directory Listing on disk in Drive n:
Syntax: DIR [n:]< CR > ....or.... DIR [n:]< LF >
Anatomy: Starts @ $B960. Ends @ $BB9A. Execs @ $B988.

The directory command is used to list the files stored on a particular disk. It will also show the size of the files as a number of sectors (each sector is 256 bytes long). The listing will indicate if files are write protected and how much room is left on the disk for more files. For directory listings of drive n: without page breaks, use the < LF > terminator.

Full Directory Listings

To obtain a full directory listing of the current disk you need only type DIR < CR > (current disk is 0 after a reset - see the DRV command for more information). A typical screen display is then:

```
DIR
O:PAULK1 DIRECTORY
PIMS      . BDT      1   PIRATE . BAS      65
STRTRK    .BAS       34  LED      . BAS    16
PIMS      . BAS      25  ELIZA    . BAS    23
ZODIAL    . BAS      87  OTHELO.BAS        13
FORWAR. OBJ          1   DIR               3P
268 USED.                90 FREE OUT OF 358
```

The 'O:PAULK1'at top left is not a filename but a disk name (see INIT command). If you name your disks in an organised manner, you can use this feature to check that you have loaded the correct disk. The file names are the names of the files on the current disk. The numbers immediately to the right are the length of the files in sectors (in decimal notation). The 'P' following one of the numbers tell you that the file is protected from accidental erasure (it is "Write Protected" and hence any activity which attempts a write operation to that file will be blocked). The 'Used' and 'Free' refer to the total number of sectors used for files and which remain available for files. The 'Out of' tells you how many sectors are available for file use in total on the disk. This is not quite the same as the total number of sectors on the disk. TANDOS 65 uses 1 sector for the system sector and one directory sector for every 15 files on disk.

Paging

It is only possible to get the directory information for 24 files on the screen at once. If you have more than this number of files on the disk then DIR will provide two (or however many it needs) pages of directory information. It will not display the next page until you press < CR > in response to the >> prompt. < ESC > terminates the directory display. On older versions, whilst pressing < LF > instead of < CR > got rid of the >> prompt, it still caused gaps to

appear on the listing which is undesirable on printer output. This has been taken care of in a re-write. Pressing < LF> will now simply give a continuous listing as it comes off the disk. This facility is also useful for those using the VDU screen with its extra line space which of course allows more files on the screen at once.

If you are using a printer, you may find it useful to avoid the next page prompt.

Selective Directories.

If you want the directory information for just one file (perhaps to check it is there or to see how long it is) you may not want to see the whole directory listing. You can obtain directory listings for any particular file by specifying the filename in the DIR command (in addition to the disk unit if necessary). e.g. DIR FILNAM.EXT. The display is exactly as before but only information about that particular file is shown (if there is no such file then no file information is displayed).

Using Wildcards.

Wildcards may be used when requesting a selective directory listing. Thus, to obtain a directory of all the Basic programs on the current disk, type DIR.BAS < CR > This file specification refers to any file with an extension BAS. Any legal combination of characters and wildcard characters may be used to give a powerful method of extracting the directory information needed.

DLOAD

Load file from disk
Syntax: DLOAD [n:]FILNAM.EXT [D N P S] < CR >
Anatomy: ROM Resident @ $B000

DLOAD is the most general way of loading files into memory - if the file is capable of execution, TANDOS 65 will execute it on load completion of the load operation. DLOAD transfers the specified file from the disk (either specified or the default current 'unit' drive) into memory. The addresses used to perform the transfer are stored as part of the file. An attempt to load to memory which the system does not believe to be present will result in an error message (NO MEMORY) and the load operation aborted.

DLOAD Parameters

DLOAD accepts various parameters to increase the flexibility of file loading.

D Display the start, end and transfer of control addresses. These are displayed in order in Hex beneath the DLOAD command (default is no display).

N No run. The file will not be executed on load completion - regardless of whether or not it would have been possible (the default is for executable files to be run on load completion).

Pn Page n. The file will be loaded to memory page n regardless of what page the file was saved from (which is the default).

SHHHH Start at location HHHH. H is a Hex number. The file will be loaded starting at address HHHH regardless of what location it was saved from (which is the default).

Auto Load

A simple version of the DLOAD command is made available by merely typing the file specification for the file you wish to load e.g. FRED < CR > will load (and if possible, run) FRED from the system disk. This is how disk resident system commands are implemented.

Parameters are not accepted for Auto Load (all the defaults apply). Under auto-load, anything following the file specification will be ignored by the load routines. The loaded program may use that data for its own purposes (just like commands for TANDOS 65 utilities).

Auto-load loads files from the system disk (as distinct from the current disk), unless you override this default by specifying the disk unit number you may auto-load from any disk e.g. 2:DIR will run the directory command from disk 2:

The TANDOS Manual on Page 12 gives details of how to make full use of the auto-load facility.

All the TANDOS 65 commands (except DLOAD which is ROM resident) may be invoked in this way, as may programs you have written.

DRV

Select Drive specified
Syntax: DRV n < CR >
Anatomy: TANBUG/TUGBUG Command

The DRV command permits the user to select any disk unit number to be the default in subsequent disk commands.

DRV must be followed by any legal unit number (in the range 0 to 7). No check is made at this stage to ensure that the particular unit is available (that check is made by all TANDOS 65 commands which use the unit number).

The current drive number is reset to 0 (the system disk) whenever a reset is performed.

Note that programs executed by using the auto load command will always be loaded from the system disk (drive 0) unless the drive number is given explicitly. The current drive number is not used to define the auto load disk. Thus:

DRV 2
DIR

will run the copy of DIR from disk 0 and display a directory of disk 2.

DSAVE

Save memory as file
Syntax: DSAVE [n:]FILNAM.EXT [THHHH Pn]< CR >
Anatomy: Starts @ $B960. Ends @ $BB8C. Execs @ $B9AF.

DSAVE is the mechanism provided for saving areas of memory in disk files. The command prompts for two Hex addresses - Start and End. These are the addresses at which the save process should begin and end respectively. A check is made to ensure that End is > = Start.

The resultant file is not normally executable i.e. DLOAD will not attempt to cause the processor to jump into the loaded memory image. If you require an executable file then you must use DSAVE with the T parameter.

DSAVE Parameters

THHHH Transfer address. HHHH specifies an address in Hexadecimal to which the computer should jump in order to execute the saved program after it has been loaded.

Pn Page n. where n is a page number from 0 to 7. DSAVE will save the memory from page n. Default is Page 0. The page selection logic is left pointing to Page n on termination of the command.

You cannot DSAVE memory in the special disk board RAM at $B800 to $BBFF. This is because TANDOS 65 loads the DSAVE programs itself into that area which, of course, will overwrite the previous memory contents. Use the TANBUG memory COPY command to move the contents of $B800-$BBFF to any convenient location and then DSAVE from there.

DZAP

Disk Sector Editor
Syntax: DZAP< CR >
Anatomy: Starts @ $B960. Ends @ $BB89. Execs @ $B960.

DZAP is the result of a much rewritten version of the original TANDOS utility called SECDMP and its re-written version SECFIX.

The main change from SECDMP to DZAP is that the screens (text input and sector output) are much more separated in the way they are controlled. The graphics cursor that can roam about the lower half of the MICROTAN screen, is now controlled in real time and this control has no affect on the upper (text output) part of the display (except in the case of CONTROL O).

Command Summary

U(drive)

Select disk drive. Drive must be in the range 0-7. No check is made to ensure that the drive actually exists.

R(sector,track)

Sector read primitive. This downloads the specified sector into the lower half of the screen. Note that the order of entering the sector, track number has been reversed from SECDMP. The main reason for this is that it is now unnecessary to re-enter the track number if the sector required is on the same track as the last entry, ie R5 will read sector five from the same track as last time. Similarly R will read the same sector,track as last time.

W(sector,track)

Sector write primitive. Comments as for sector read in every way except this is obviously the inverse function.

Exxxxx

Enter an ASCII string specified in "xxxxx" into the screen buffer (lower). Note that only what is on the screen line can be transferred so if the cursor 'falls" off to the next line, that text will NOT be transferred and the statement error "?" will be generated. The text will enter from the current buffer cursor position.

HFF(,FF,FF)

Transfer the Hex bytes specified to the screen buffer. Leading zeros are ignored and as many bytes it can fit on one screen line may be transferred at once. The bytes will be entered from the current buffer cursor position onwards and its position automatically updated.

L (0-FF)

Locate the screen buffer cursor at the specified position in the lower half of the screen. Note that the parameter is in Hex and that the display now includes an "index" line immediately above the lower screen in order to help with positioning. This command has largely been made redundant by the change in nature of cursor control which is not now as cumbersome and faster.

F

F will F(ill) the screen buffer with the specified ASCII character following the F on the screen. Note that F< CR > will clear the screen buffer under index line.

X

X will invoke a tidy exit from the program DZAP.

CONTROL COMMANDS

The graphics cursor in the lower half of the screen is controlled in real time by our old 'Control key' friends ^R, ^L, ^U, ^D for right, left, up and down respectively. This cursor control can occur at anytime and will NOT affect in any way what it going on in the upper text part of the screen. For example you may be in the middle of typing in a string of characters after the E command ready to transfer when you realise that the buffer cursor is in the wrong position. No problem. Just change the cursor position using the control keys and carry on typing your text.

The biggest change from the old SECDMP program is in the P command which is no longer valid and has been replaced by ^O (O is for output). As in the cursor control functions this key is also 'live' in that it can be used at any time and from any position. On first use of ^0, the byte underneath the cursor in the buffer will be output to the upper text screen as a hex number followed by a ','. The cursor in the buffer moves right one position. You have now

entered the output mode and will remain in it until a < CR > is issued from the keyboard. Further use of the ^O key will result in the next byte being output etc. This means that whole group of bytes may be output to a single line on the text screen for easier checking when applicable. Note that these CNTRL functions 'wrap around'. i.e. the graphic cursor will drop onto the next line if using ^R or ^0 etc.

General Comment

DZAP resides in the 'transient program area' of the RAM on the DOS card (as was SECFIX) and NOT in normal RAM so in most cases it will not upset or infringe on normal program operation. If the VDU 80/82 board is resident in the system and you are running under TUGBUG, then the VDU board will have its own screen cleared and then further output to it will be suppressed on start up of DZAP. The X command will reinstate the VDU board. Beware of errors occurring due to the DOS. In most cases DZAP will trap out any scrolling into the lower part of the screen by virtue of its own screen handling but should a DOS error occur then it is possible for the message that gets output by the DOS to upset the screen formatting, the easiest way around this is to issue a reset then type in GB960 for an orderly return to DZAP operation.

FORMAT

Formats the disk in the drive specified
Syntax: FORMAT n:< CR >
Anatomy: Starts @ $400. Ends @ $6BD. Execs @ $589.

The format utility program lays down the basic magnetic pattern of sectors and gaps on the disk surface so that future operations can find particular areas of the disk (broken down or 'mapped' into tracks and sectors). Details of this formatting are provided in the Floppy Disk Controller Data sheets. It is always necessary to format new disks before using them for anything at all (even non-TANDOS 65 things like DFORTH needs to use formatted disks. You should not need to use FORMAT again unless you have suffered a really catastrophic disk crash. Obviously, the FORMAT utility will destroy, beyond recall, any files or other information previously stored on the disk.

FORMAT makes use of the information held in the system definition on the system disk to determine the number of tracks on the disk to be formatted. To format a disk, issue the command FORMAT n: Note that you must give the disk unit explicitly (this is to avoid nasty accidents). The utility will be loaded from disk and will then issue a prompt to 'LOAD disk n & PRESS RETURN'. You should then load the disk to be formatted, check that you have given the correct command and, if all is well, press < CR >. Formatting will then take place and after about 10 seconds (for a 40 track disk). A message will inform you that formatting is complete. If you wish to escape from formatting after issuing the TANDOS 65 command then you should press < ESC > when requested to 'LOAD disk n & PRESS RETURN'. This will cause a return to the monitor prompt without writing to your disk. Note that FORMAT is one of the few TANDOS 65 commands to use memory from $400 upwards.

INIT

Initialise a disk in Drive n.
Syntax: INIT n:< CR >
Anatomy: Starts @$B960. Ends @ $BB46. Execs @ $B9CB.

Initialising the disk after formatting is important because the disk will just appear to be completely empty to the DOS and therefore will not be able to make any sense of it. This process involves the following activities. Firstly the set up the system sector in the first sector on Track 0. The includes the System Definition and Page Definitions (see SYS command), the Disk name, the location of the first Directory sector. Next the identification of the first Directory sector on the fourth sector of Track 0. All the other sectors are 'daisy-chain' linked together creating a 'free space chain'. All existing files are lost, regardless of their protect status. It is not, however, possible to INIT a disk with the write protect tab in place on the disk itself.

INIT loads from the system disk and then prompts the user to load the disk to be initialised. This provides a suitable opportunity, if needed, to remove the system disk and reload the drive with another disk. This is especially important for users with only a single disk drive. The user is next prompted for a disk name. This may be any combination of up to 9 alphanumeric characters. Try and use unique names for all your disks so that you can distinguish between them later. Again this is more important for users with single disk drives - under some circumstances TANDOS 65 attempts to check that the correct disk is loaded - it cannot do this if disks have the same or no name. After the disk name is terminated by < CR >, the initialisation process is started. This takes about 40 seconds for a 40 track disk.

For double-sided disks, each side must be initialised separately.

The system sector information (number and type of disk drives, amount of RAM) on a newly initialised disk will be the same as the current system disk. Run SYS if it needs to be changed.

Make absolutely certain that you do not INIT a disk by accident. There is no way that the old files can be recovered since every disk sector is overwritten during the initialisation sequence. You may escape from INIT, without harming your disk by pressing < ESC > in response to the 'disk Name' prompt.

KILL

Latest Delete file utility
Syntax: KILL [n:]FILNAM.EXT < CR >
Anatomy: Starts @ $B96O0. Ends @ $BBAE. Execs @ $B970

All comments about the DEL command also apply to this new command KILL. It was born simply as a short cut to enable the deletion of files with the protection status set SO BEWARE! Operation using wildcards is identical to DEL.

REN

Rename a file
Syntax: REN [n:]NEWNAM.EXT < [n:]OLDNAM.EXT [P N]< CR >
Anatomy: Starts $B960. Ends @ $BA69. Execs @ $B985

The RENname command is used for two different purposes.

The main use is simply to change the name of a file. However you cannot rename a file which is 'protected'.

The secondary use is to change the protection status of a file using P (protected) or N (non-protected) optional parameters.

Both the source and destination file specifications (i.e. those on either side of the '< ' symbol) must refer to the same disk unit. (It should be obvious that merely renaming a file cannot transfer it to a different disk).

Wildcards are not accepted by RENname.

S

An alternate to the DSAVE utility
Syntax: S [n:]FILNAM.EXT START END [THHHH RHHHH Pn]< CR >
Anatomy: Starts @ $B960. Ends @ $BBFC. Execs @ $B9A7

START and END are the start and end addresses of the data to be saved. These are mandatory. T, R, P are optional and may be entered in any order.

THHHH is the transfer address when the data is loaded the loader will jump to these address.

Px is the Page Number that the data will be saved from and subsequently re-loaded to.

RHHHH enables the data to be re-located when next loaded from the disk. This can be used for saving programs that have been relocated in order to assemble them.

When using 'R' the 'T' address must be the correct one. It is not adjusted by the save routine.

If the filename is already on the disk, the program will prompt "BAK (Y,N/B) ?". Respond Y(es) to delete the old copy from the disk, B(ackup)to make the original version a backup by adding the extension 'BAK', or N(o) to abort the operation.

Errors

There are 3 errors that can occur:-
1) A syntax error in the command
2) There is not enough space on the disk
3) One of the files is write protected
The errors are reported as 1) SYN 2) NSP & 3) WRP

This new version of DSAVE first appeared in the TUG newsletter No.26 Page 16 along with notes explaining the rationale.

SYS

Runs the system definition Utility on specified disk
Syntax: SYS n: < CR >
Anatomy: Starts $B960. Ends @ $BB7F. Execs @ $B9A1

The SYStem definition utility provides a way for the user to keep the system informed of any changes to the disk configuration and RAM in use. TANDOS 65 keeps a record of this information in the System Definition and Page Definition areas respectively of the system disk.

The SYS utility inspects each entry of the System Definition and Page Definition in turn allowing the user to change the value if required. For the System Definition each entry is the number of tracks present on the indicated drive (Unit 0 to 7). A zero entry represents no drive fitted. For the Page Definition each entry is the size of RAM available in Kbytes on the indicated page (numbered 0 to 7).

Paged random access memory is available when TANRAM memory boards are fitted in a Tangerine motherboard. A TANRAM fitted in Slot 0 is accessible by selecting Page 0. TANRAM fitted in Slot 1 is accessible by selecting Page 1 etc. Page 0 memory is the default memory available to the system following RESET. Pages 1-7 memory are only available if a Tangerine motherboard (or equivalent) is fitted. TANRAM provides memory above $1FFF up to an upper limit at $BBFF. However, as TANDOS uses memory $A800-SBBFF, the maximum memory available when TANRAM is fitted in any slot is 42 Kbytes.

If the system only has fully populated TANEX memory available then the upper limit of user memory will be $1FFF (ie 8 Kbytes). TANEX memory is accessible no matter which page is selected. This is the reason why the SYS utility requires a minimum value of 8 to be entered for every entry in the Page Definition.

If the system has Tanex-Plus memory available or an additional board fitted (eg 64K RAM board), then the upper limit for Page 0 will be $EFFF (ie 42 Kbytes) as described above. However, if the ROM area $C000-$DFFF is replaced with RAM, then the upper limit should be set to $EFFF (60 Kbytes). If this is done however, the TANDOS system will not prevent a user from loading a program/file into the I/O area ($BC00-BFFF).

Note: TANDOS does not inhibit saving data to disk from above these limits, however it will display the error message 'No memory-P0' if loading a file into an area above these limits is attempted.

If the stored values do not correspond with what is actually present on the system, then either system performance will be reduced, or worse still, it may not work properly. It is important to keep this information up to date. For example, if an old 40 Track disk drive is exchanged with a new 80 track disk drive, any new system disks created will continue to be formatted for 40 Tracks until the System Definition is updated. If it is necessary to update this information, all system disks will need to be updated to ensure the system accesses the same information whichever system disk is in use.