FARNA *Systems*
904 2nd Avenue
Warner Robins, Georgia
31098-1029
"Supporting the Tandy Color Computer!"

# OS-9 Quick Reference and Programmers Guide

# for the Tandy Color Computer

by F.G. Swygert

Proofed by Rick Ulland

**the FARNA Fox!**
**FARNA Systems**

List filename >/p & -run printer in Background.

Shell i=/w& -open new window.

## CONTENTS

OS-9 Quick Reference Guide for the Tandy Color Computer
(Based on OS-9 Level II)

Distributed by
FARNA Systems
904 2nd Avenue
Warner Robins, GA 31098-1029
Phone 912-328-7859

Printing by CopyMasters
1304 Watson Blvd.
Warner Robins, GA 31093

## INTRODUCTION

One of the troublesome things about learning OS-9 is that bulky manual. It takes up lots of desk space, and it is sometimes hard to find that one simple code needed to complete a project. And one has to refer to the manual a LOT when first starting out- more than one cares to admit I'm sure!

This little Quick Reference Guide was designed to get that manual off your desk and back on the bookshelf. It isn't, however, a replacement for the full manual. Only a brief description of commands and codes are given, sometimes no more than the syntax for entering. Enough information is here to jog one's memory and get back on track, but the manual will still have to be referred to for learning and heavy duty programming chores.

## A Note on Redirection:

OS9 commands generally read from the keyboard and write to the current screen. Almost all of them, however, can be sent elsewhere, using the redirection symbols:

- < (input)
- > (output)
- >> (error output).

Some common redirections:

- echo Hello>/w7 - would print 'Hello' on window 7
- list file>/p - lists file to printer
- utility -? >>/p - would print the help file

(note that help info often follows the error path)

## 1 - SYSTEM COMMANDS

Commands are given in all capital letters. Items following in *italics* are required. Items enclosed in parentheses () are optional. COMPLETE path lists must be used in paths and names (file and directory) or current is assumed (**path** is used to mean **pathlist**).
* Commands with a leading asterics are found on the companion disk or Delphi OS-9 SIG.

**ATTR** ***filename*** **(permission)** examine or change file security attributes. filename is the name of the file to be examined or changed, permission is one or more of the options:

- a - display attributes
- d - file is a driectory
- e - only owner can execute
- r - only owner can read
- s - non-shareable file
- w - only owner can write

A "p" in front of e,r, or w means anyone (public) can access file. A minus sign (-) in front of a permission turns that permission off.

**BACKUP (options)(device1)(device2)(#nK)** backup data from one drive to another. If no drive specified, /d0 to /d1 assumed. If only device1 specified, single drive assumed. Options:

- e - cancel on read error
- s - use single drive
- -v - verify off
- #nK - memory used in kilobytes. Can specify up to 56K to speed backup process.

**BUILD** ***filename*** creates a text file by copying keyboard input to filename. Writes to file after enter is pressed. Press enter with no text to close file.

**CHD** ***path*** changes current data directory to directory named in path.

**CHX** ***path*** changes current execution directory to directory named in path.

**CMP** ***file1 file2*** compares binary values of data in the two files specified. Displays address and values of different bytes if encountered.

**COBBLER** *device* creates OS9 boot file. Device is the drive the boot file will be created on. Writes kernal to track 34 and remainder to the file OS9Boot, making an exact copy of the current system. Use with caution- if cobbler cannot write OS9Boot into one block,(for example on an almost full disk) the boot it makes will not work.

**CONFIG** create new system disk. Follow prompts, refer to manual for more info.

**COPY** ***path1 path2*** **(options)** copies from one file or device to another. Path1 is complete path and name of source, path2 complete path and name or destination. Options:

-s - single drive copy. Prompts for source and destination disks.
nK - memory used in kilobytes. Can specify up to 56K to speed copy process.

**DATE (t)** displays current system date. "t" displays current system time with date.

**DCHECK (-options)** ***drive*** checks disk file structure in drive specified. Options:

-b - suppress listing unused clusters
-m - saves allocation map work files
-o - prints valid DCHECK options
-p - prints path for questionable clusters
-s - counts and displays number of directories and files. Only checks file descriptors.
-w=path - specifies path for work files

Preceding dash used once, then all options can be together (no spaces between): DCHECK -bps /d0

**DEINIZ** ***device1*** **(device2) (etc.)** deinitializes and detaches specified device(s). One should only DEINIZ a device that was initialized with INIZ.

**DEL (-x)** ***file1*** **(file2) (etc.)** delete (erase) the specified file(s). -x assumes file is in current execution directory. Can only DEL files you have permission to write to (see ATTR).

**DELDIR** *directory* deletes specified directory along with all associated subdirectories and files.

**DIR (options) (path)** display contents of directory named in path. If no options or path specified, current data directory is assumed.
Options:

e - displays size, address, owner, permissions, and last date/time modified.
x - current execution directory

Options can be used without path and vice-versa.

**DISPLAY** ***hex#1*** **(hex#2) (etc.)** reads hexadecimal number hex#, converts to ASCII character, and writes to standard output device (always the current window unless redirected). Can redirect output to other devices, such as printer. For screen, 32=text, 33=background, and 34=border color. REF. Color Codes page 33, Device Windows page 34, & Window Text Commands page 35.

**DISPLAY 1b 33 08** changes the current window (1b) background (33) to white (08).

**DISPLAY 41 42 43** prints ABC on the screen (standard output path)

**NOTE:** It is legal to use the foreground color numbers for any of the 3 areas. OS-9 converts decimal to hex. Output can be redirected.

**Display Codes**

| Windows | G/P Buffers | Colors |
|---|---|---|
| 1b20 DWSet | 1b29 DfnGPBuf | 1b30 DefColr |
| 1b21 Select | 1b2a KillBuf | 1b31 Palette |
| 1b22 OWSet | 1b2b GPLoad | 1b32 ForeColor |
| 1b23 OWEnd | 1b2c GetBlk | 1b33 BackColor |
| 1b24 DWEnd | 1b2d PutBlk | 1b34 Border |
| 1b25 CWArea | 1b2e PSet | |
| | 1b2f LSet | |

| Graphics Text | Graphics Absolute | | Graphics Relative |
|---|---|---|---|
| 1b35 ScaleSw | | | |
| 1b36 DWProtSw | 1b40 | DPtr | 1b41 |
| 1b39 GCSet | 1b42 | Point | 1b43 |
| 1b3a Font | 1b44 | Line | 1b45 |
| 1b3c TCharSw | 1b46 | LineM | 1b47 |
| 1b3d BoldSw | 1b48 | Box | 1b49 |
| 1b3f PropSw | 1b4a | Bar | 1b4b |

(BOTH)
1b4e PutGC
1b4f FFil
1b50 Circle
1b51 Ellipse
1b52 Arc3P

*** DMODE** ***drive*** **(options)** modifies disk drive device descriptors. Use without options to display current drive parameters, follow option with desired **HEX** value to change. Use **COBBLER** to make permanent.Options are as follow:

drv - drive number
stp - step rate in ms (00=35, 01=20, 02=12, 03=6)
typ - drive type (20 indicates 5.25" CoCo format)
dns - density (tracks per inch- 1=40, 3=96)
cyl - cylinders (tracks- 23=35, 28=40, 50=80)
sid - number of sides
vfy - write verify on (0)/off(1)
sct - sectors per track (12=18)
tos OR t0s - sectors per track, track 0 ONLY
ilv - sector interleave factor (normally 3)
sas - number of sectors allocated at one time (normally 8)

To change the current 35 track drive 0 to 40 track double sided, 35ms access, use:

**DMODE /d0 stp=03 cyl=28 sid=2**
**DMODE /dd stp=03 cyl=28 sid=2**

**NOTE:** BOTH /d0 modules (/d0 & /dd) MUST be identical!

**DSAVE (options) (device) (path)** ***path*** copies (backs up) all files to specified directory. /d0 assumed if no device given. Path is a command procedure file DSAVE stores it's output in. Options:

-b - copies source disk's boot file (if present)
-b=path - same as -b except OS9Boot comes from specified path
-i - indents directory levels
-l - don't process directories below current level
-m - don't include MAKDIR in path
-s# - specifies amount of memory for copy to use in # of K
-v - verify copy by forking to CMP after each file

Can also be 'piped' to a shell for immediate execution. To copy the disk in /d0 to /d1 type: **chd /d0; dsave /d0 /d1 ! shell**

**ECHO** ***text*** prints typed text to standard output, usually the current screen (can de redirected). Used to create messages in procedure files or to send initialization strings to a terminal. Can be redirected to any device.

**EDIT** see Text Editor Commands (page xx).

**ERROR** ***number*** displays message for error number.

**EX (file)** terminates current shell then runs file (if given). If no other shell open, OS-9 will crash. Must always be the last command on a line.

**FORMAT** ***device*** **(options) (name)** formats device using options and assigning name. Options:

r - don't display prompts
1 - format single sided disk
2 - format double sided disk
'#' - format # of tracks
:#: - sector interleave value #
name - disk name

If formatting a hard drive, first use the command "**tmode -pause**" to turn screen pause off. Otherwise format will stop whenever the screen fills with verified sectors. Standard CoCo floppy interleave value is 3. New drives may be capable of 2.

**FREE** ***drive*** displays number of unused sectors, name, date created, and cluster size of disk in specified drive.

**HELP** ***command1*** **(command2) (etc.)** displays help file for specified command(s). Many third party utilities have a built in help file. Use utilityname -? to view.

**IDENT** ***name*** **(options)** displays header information for file or module name. Options:

-m - assume name is a module in memory
-s - display edition byte, type language byte, module CRC, & module name on one line. If CRC verifies, a period will be displayed, question mark if not.
-v - don't verify module CRC
-x - assume file name is in execution directory

**INIZ** ***device1*** **(device2) (etc.)** initializes specified device driver(s).

*** IPATCH** ***patchfile oldfile newfile*** **(-v)** creates newfile from oldfile using changes specified in patchfile. Patchfile is created using **MAKPATCH** (see). File names may contain paths. -v enables display of installation process on screen.

**NOTE:** many patchfiles are available in the Delphi OS-9 SIG.

**KILL** ***processID*** terminates specified processID number. Can only terminate a process with your user number attached.

**LINK** ***module*** increases module link count by one. When a module is loaded, link count is 1. Count becomes 2 when module is run. When finished, count drops by 1, but module remains. Modules run from disk only has a count of 1, and will be dropped as soon as it's finished. Can switch to another window and link a module rather than reloading. Once the count is reduced to 0, module "disappears" from memory and must be re-loaded.

**LIST** ***file1*** **(file2) (etc.)** lists the contents of specified text file(s). Can list to screen or other device. May be redirected.

**LOAD** ***path*** loads module(s) specified in path into memory.

**LOGIN** provides security for timeshare systems. Requests username and password, checks against validation file. Automatically sets usernumber, execution & data directories, and executes a program in password file. Automatically called by **TSMON**.

**MAKDIR (path)** ***name*** makes directory name in current data directory unless path is specified.

*** MAKPATCH** ***oldfile newfile patchfile*** **(-v)** creates patchfile which describes differences between oldfile and newfile. Patchfile contains a header followed by a series of "patch entries". Each entry contains: an 8 bit type (0=deletion, 1=addition, 3=disparate size change, 4=done), and three 16 bit unsigned values (offset in oldfile where patch applies, size of old area to patch, size of new area). -v enables display of installation process on screen. Must be knowledgeable with m/l to use! **WARNING:** newfile will be overwritten if in current path.

**MDIR (e)** displays names of modules currently in memory. e lists extended physical address, size, type, revision level, re-entran attribute, link count, and name of each module. Numbers shown in hexadecimal.

**MERGE** ***(file1) (file2)*** **(etc.)** copies file(s) to standard output path. Output can be redirected to any device. **MERGE file1 file2 >file3** combines files1&2 into file3 in the current directory and drive.

**MFREE** displays block number, beginning and ending address, and size of unassigned RAM blocks.

**MONTYPE** ***type*** sets monitor type, where type is **r** for RGB, **c** for composite color, **m** for composite monochrome

**OS9GEN** ***device*** **(options)** creates a bootable disk by creating and linking required OS9Boot file. Device is the drive with the disk to be made bootable. Use ONLY on a newly formatted disk! Options:

-s - use single drive
#nK - memory used. Can specify up to 56K to speed OS9GEN

To use os9gen, create a text file with the modules desired, and place it in the MODULES directory. Chd to this directory and redirect os9gen's input with **os9gen /dx /dx <textfile**. Alternately, each file to be added can be input separately:

```
OS9: os9gen /d1  (/d1 is receiving drive)
OS9: /d0/os9boot (use all modules in current boot)
OS9: /d1/new.driver  (file to be added)
OS9: /d1/new.driver  (file to be added)
OS9: (press BREAK)
```

*** PCDOS** ***-command*** **(options)** ***device*** **(DOS path) (OS9 path)** displays a directory and transfers files to/from MS-DOS, Atari ST DOS disks, and OS9 disks. Blank MS/Atari disk MUST be pre-formatted. Requires patched version of CC3Disk or SDisk3 for OS-9 Level II (on CoCo 3). Automatically converts text files to proper format. Mainly used for ASCII text file transfers. Commands:

-dir - list directory of DOS disk. Can use
-del - delete file on DOS disk
-id - display MS-DOS BIOS id info
-get - import file FROM DOS disk
-put - export file TO DOS disk

Options:

-all - use ONLY after -dir to display hidden DOS files.
-raw - use ONLY after -put/-get to prevent text file conversion.

**PROCS (e)** displays list of processes currently running. Shows ID#, user#, priority, etc. e displays processes of all users.

**PWD** shows path from root directory to current data directory.

**PWX** shows path from root directory to current execution directory.

**RENAME** ***path name*** renames the file or directory found in path to name. Only the first parameter is a complete path.
**rename /d1/cmds/util util2**

*** RSDOS** ***-command*** **(options)** ***device*** **(DECB path) (OS9 path)** displays a directory and transfers files to/from DECB disks and OS9 disks. Blank DECB disk MUST be pre-formatted. Requires patched version of CC3Disk or SDisk3 for OS-9 Level II (on CoCo 3).
Commands:

-dir - list directory of DECB disk
-del - delete file on DECB disk
-get - import file FROM DECB disk
-put - export file TO DECB disk

Options (used ONLY with -put):

-b - BASIC binary program (type 0)
-d - BASIC data file (type 1)
-m - executable M/L program (type 2)
-t - text editor source file (type 3)
-a - ASCII file (text or BASIC)
-f=x - sets type to number x (0-255)

**NOTE:** default format is BASIC binary program (0).

*** SDUMP &** loads SDUMP, which waits in background until called with SHIFT-CTRL-ALT to print current text screen. Works with multiple windows, just run once.

**SETIME (yy/mm/dd hh:mm(:ss))** sets system date and time to year (yy), month (mm), day (dd), hour (hh), minutes (mm), and optionally seconds (ss). Date and time can also be separated by colons, spaces, or slashes, but NOT commas. No separaters need be used, just a space between date and time.
**SETIME 91:05:01:13:30 (91,May 1,1:30 pm)**
**SETIME 91/05/01 13/30 (same as above)**
**SETIME 910501 1330 (same as above)**

**SETPR** ***processID*** **#** changes process processID priority to #. Can set only for processe with your user number attached. Lowest priority is 1, highest 255.

**SHELL** ***list*** causes the shell to run list of commands, load list of parameters, and/or list of options. All can be combined. Refer to manual chapter 3.

**TMODE (path#) (option1) (option2) (etc.)** displays or changes terminal parameters (options). Path# is one of the standard path numbers: .0-input, .1-output, .2-error output. Options (default value is hexadecimal unless otherwise noted):

bsb - backspace erases characters (default)
-bsb - backspace doesn't erase characters
bsl - backspace-space-backspace deletes terminal display line (default).
-bsl - disable backspace over a line
echo - input characters echo to screen (default)
-echo - disable echo
lf - turn on auto line feed to screen (default)
-lf - disable auto line feed
upc - uppercase characters only. Converts lower to upper.
-upc - upper and lower case characters
pause - turn on screen pause when full, press space to resume (default)
-pause - disable screen pause
abort=x - sets terminate character, normally CONTROL C
baud=x - sets baud rate, word length, and stop bits for software controlled interface. Bits 0-3 control baud rate (0=110, 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=19200 w/ACIAPAK, 7=32000 w/SIO). Bit 4 is reserved. Bit 5-6 is word length (00=8 bits, 01=7 bits). Bit 7 is stop bits (0=1, 1=2)
bell=x - sets bell character (output), default=07
bse=x - set backspace char. (output), default=08
bsp=x - sets backspace char. (input), default=08
del=x - sets delete line char. (input), default=18
dup=x - sets character to duplicate last input line, default=01
eof=x - sets end-of-file char. (input), default=1B
eor=x - sets end-of-record character (carriage return, input), default=0D
null=x - sets null count (number of nulls after carriage return), default=0 (decimal)
pag=x - sets number of video display lines (decimal). Affects pause.
psc=x - sets pause character, default=17
quit=x - sets quit character (normally CONTROL E)
reprint=h - sets reprint line character (hex)
type=x - use for ACIA initialization, default=00. Bit 0= true lowercase for TERM-VDG (1=yes, 0=no). Set to 80 for a window device for TERM-WIN. Bit 4 sets auto-answer modem feature (1=on, 0=off). Bits 5-7 set MARK(101), SPACE(111), no parity(000), even parity(011), or odd parity(001) for all ACIA devices.

**TSMON** ***(terminal#)*** monitors idle terminals on a timesharing system. Initiates a login sequence when an idle terminal is requested. Logoff by sending end-of-file character (BREAK/ESCAPE).
**TSMON /t10**
**TSMON (BREAK)**

**TUNEPORT (device) (-s=x)** allows testing and setting delay loop for serial port. Device to be tested/set is printer (/p) or terminal (/t1). Test by typing TUNEPORT (device). Current baud rate will be displayed and test data sent to printer. Current delay will be displayed and user prompted for a new value (in decimal). Continue until proper rate is found. Set new rate with **TUNEPORT (device) -s=x** where x is the new decimal value. Use **COBBLER** to make change(s) permanent.

* **UNDEL** ***(path)*** **(directory)** scans directory for deleted files/directories and prompts for the first letter of the deleted file/directory if recoverable (remainder of file name will be displayed). Use uppercase letter if undeleting a directory, lower case for files. Press enter to skip file. Current directory scanned if no path given. DO NOT USE UPPERCASE FIRST LETTER ON A FILE! A disk editor will have to be used to remove the directory attribute if this happens. If a directory is given a lowercase letter, DEL then UNDEL again using uppercase.

**UNLINK** ***module1*** **(module2) (etc.)** reduces named program or command link count by one. Will be unloaded from memory once count reaches 0. If a module is named that wasn't loaded or is being used by another process, that process will crash, usually with error 221 (module not found). Modules that are part of a merged file cannot be unlinked except for first module in file, which is the "master" file. Unlink the master and the entire group count will be reduced. All files merged in the group will show a count of 0. The file just before the 0s is the master file (shows count for group).

**WCREATE (-?) (-z)** ***/wpath*** **(-s=type)** ***xpos ypos xsize ysize forecolor backcolor*** **(border)** used to intitialize and create a window. /wpath - device name of window (W, W1, W2, and W5-W7). Options:
-? - display help message
-z - accepts input redirected from a file, either a named file or a following line in the same file.
**WCREATE -z < file** (window instructions in file)
-s=type - screen type: 1=40 col., 2=80 col., 3=640x192 2-color, 4=320x192 4-color, 5=640x192 4-color, 6=320x192 16-color
xpos - x coordinate for upper left corner of screen
ypos - y coordinate for upper left corner of screen
xsize - number of columns across screen (1-80)
ysize - number of lines in screen(1-24)
forecolor - foreground or lettering color
backcolor - background color
border - border color (default is black). Border must be specified if using -s=type.
* See page xx for color codes. Also see "Device Windows", page xx.
**WCREATE /w1 -s=2 0 0 80 24 7 4 1** (sets up 80 col. window as /w1)

* **WMODE (parameters) /wx** displays or changes window attributes of an initialized window, where /wx is the desired window descriptor. Use parameters as specified under WCREATE. Built-in help displayed if no parameters given. Does NOT change items specified under XMODE or TMODE.

**XMODE** ***device*** **(option1) (option2) (etc.)** displays initialization parameters of any SCF-type device (screen, printer, RS-232 port, etc.) if no options listed. Changes initialization parameters to those listed when option list is included. Similar to TMODE, but XMODE updates remain as long as the computer is on during current session. TMODE only works on open paths so effects are gone once current path is closed. Options are same as TMODE (see for list).
Use COBBLER to make changes permanent.

## 2 - Special Keys

| KEY(S) | FUNCTION |
|---|---|
| ALT x | Displays graphic characters on VDG screen, international characters in window (x is any key) |
| CLEAR | selects next window (when windows are used) |
| CTRL | control key |
| CTRL - | prints underline (displays as ) |
| CTRL , | prints left curly brace ( { ) |
| CTRL . | prints right curly brace( } ) |
| CTRL / | prints back slash ( \ ) |
| CTRL 0 | shift lock on/off |
| CTRL 1 | prints vertical bar ( \| ) |
| CTRL 3 | prints tilde ( ~ ) |
| CTRL 7 | prints up arrow |
| CTRL 8 | prints left bracket ([) |
| CTRL 9 | prints right bracket (]) |
| CTRL A | displays last line typed with cursor at end. Press ENTER to execute or edit by backspacing. Repeat to display line. |
| CTRL C or CTRL D | redisplay command line |
| CTRL E or BREAK | halts current program |
| CRTL H or CTRL W | moves cursor one space left, temporarily halts video. Any key restarts. |
| CTRL X or SHIFT | delete current line |
| CTRL BREAK | same as ESCape, sends end-of-file to program receiving keyboard input. Must be first on a line. |
| CTRL CLEAR | toggles "keyboard mouse" on/off. Uses arrow keys for directions, F1 and F2 for buttons. Normal arrow/F key operation suspended when keyboard mouse is active. |
| SHIFT BREAK | similar to CTRL C but only interrupts current screen, program continues running in background. |
| SHIFT CLEAR | selects previous window (when windows are used) |
| ENTER | carriage return or execute current command line |

## 3 - Keyboard Codes

Listing consists of **KEY-VALUE** (values in hexadecimal)

| NORM | SHIFT | CTRL | NORM | SHIFT | CTRL |
|---|---|---|---|---|---|
| 0-30 | 0-30 | | G-47 | g-67 | BEL-07 |
| 1-31 | !-21 | -7C | H-48 | h-68 | BSP-08 |
| 2-32 | "-22 | 00 | I-49 | i-69 | HT-09 |
| 3-33 | #-23 | -7E | J-4A | j-6A | LF-0A |
| 4-34 | $-24 | 00 | K-4B | k-6B | VT-0B |
| 5-35 | %-25 | 00 | L-4C | l-6C | FF-0C |
| 6-36 | &-26 | 00 | M-4D | m-6D | DR-0D |
| 7-37 | '-27 | 5E | N-4E | n-6E | CO-0E |
| 8-38 | (-28 | [-5B | O-4F | o-6F | CI-0F |
| 9-39 | )-29 | ]-5D | P-50 | p-70 | DLE-10 |
| :-3A | *-2A | 00 | Q-51 | q-71 | DC1-11 |
| ;-3B | +-2B | 00 | R-52 | r-72 | DC2-12 |
| ,-2C | <-3C | 7B | S-53 | s-73 | DC3-13 |
| –2D | =-3D | 5F | T-54 | t-74 | DC4-14 |
| .-2E | >-3E | 7D | U-55 | u-75 | NAK-15 |
| /-2F | ?-3F | \-5C | V-56 | v-76 | SYN-16 |
| @-40 | 60 | NUL-00 | W-57 | w-77 | ETB-17 |
| A-41 | a-61 | SOH-01 | X-58 | x-78 | CAN-18 |
| B-42 | b-62 | STX-02 | Y-59 | y-79 | EM-19 |
| C-43 | c-63 | ETX-03 | Z-5A | z-7A | SUM-1A |
| D-44 | d-64 | EOT-04 | | | |
| E-45 | e-65 | EMD-05 | | | |
| F-46 | f-46 | ACK-06 | | | |

### Function Keys

| KEY | NORMAL | SHIFT | CTRL |
|---|---|---|---|
| BREAK | 05 | 03 | 1B |
| ENTER | 0D | 0D | 0D |
| SPACE | 20 | 20 | 20 |
| LT ARROW | 08 | 18 | 10 |
| RT ARROW | 09 | 19 | 11 |
| DN ARROW | 0A | 1A | 12 |
| UP ARROW | 0C | 1C | 13 |

## 4 - User Mode Calls

System call format is: **name: mnemonic, software interrupt code2, request code** (in hexadecimal). See manual for more info.
ADDR = address, CLR = clear, CHR = character, REV = revision
# = number, DIR =directory, INI = initialization.

**Allocate Bits: F$AllBit 103F 13**- set bits in allocation bit map. Bit numbers from 0 to one less than number of bits in map.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| D= # of first bit | CC=carry set |
| X=start ADDR of bit map | B=error code |
| Y= # of bits to set | |

**Chain: F$Chain 103F 05**- load & execute new primary module, no new process created.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=language/type code | CC=carry set |
| B=size of data area (in pages) | B=error code |
| X=ADDR of module/file name | |
| Y=parameter area in pages | |
| U=start ADDR of parameter area | |

**Compare Names: F$CmpNam 103F 11** - compare two strings.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| B=length of first string | CC=carry CLR |
| X=ADDR of first string | |
| Y=ADDR of second string | |

**Copy External Memory: F$CpyMem 103F 1B**- read external memory into buffer.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| D=DAT image pointer | CC=C bit set |
| X=begin copy offset | B=error code |
| Y=byte count | |
| U=destination buffer | |

**CRC: F$CRC 103F 17**- calcualte module CRC

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=start byte ADDR | Updates CRC accumulator |
| Y= # of bits | |
| U=CRC ADDR | |

**Deallocate Bits: F$DelBit 103F 14**- clear allocation map bits.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| D= # of first bit | None |
| X=start ADDR of bit map | |
| Y= # of bits to set | |

**Exit: F$Exit 103F 06**- terminates calling.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| B=return status code | Process terminated |

**Fork: F$Fork 103F 03**- creates child process.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=language/type code | X=last name ADDR byte +1 |
| B=data area (pages) | A=new process |
| Y=parameter area (pages) | |
| U=start ADDR of parameter ID # area | |

**ERROR OUTPUT:**
B=error code
CC= carry set

**Get System Block Map: F$GBlkMp 103F 19**- get copy of block map.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=pointer to 1K buffer | D=bytes per block |
| **ERROR OUTPUT:** | Y=system block map size |
| CC=carry set | |
| B=error code | |

**Get Module Directory: F$GModDr 103F 1A**- get copy of system module directory.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| X=pointer to 2K buffer | CC=carry set |
| Y=end of copied DIR | B=error code |
| U=start of system DIR | |

**Get Process Descriptor: F$GPrDsc 103F 18**- get copy of process descriptor.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=process ID | CC=carry set |
| X=pointer to 512 byte buffer | B=error code |

**Intercept: F$Icpt 103F 09**- set signal intercept trap.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=ADDR of intercept routine | Signal sent to process causes intercept to be called, process not killed. |
| U=ADDR of routine memory | |

**Get ID: F$ID 103F 0C** - get process ID and user ID

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| None | A=process ID |
| | Y=user ID |

**Link: F$Link 103F 00** - link to named module.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=type/language | A=type/language |
| X=ADDR of module | B=ATTR/REV |
| **ERROR OUTPUT:** | X=last name ADDR byte +1 |
| CC=C bit set | Y=module entry point ADDR |
| | U=module header ADDR |

**Load: F$Load 103F 01** - load module(s) from file.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=language/type | A=language/type |
| X=pathlist ADDR | B=ATTR/REV |
| **ERROR OUTPUT:** | X=last path ADDR byte +1 |
| CC=carry set | Y=module entry point ADDR |
| | U=module header ADDR |

**Memory: F$Mem 103F 07** - change process data memory.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| D=memory size in bytes | Y=upper memory ADDR |
| **ERROR OUTPUT:** | D=memory size in bytes |
| CC=carry set | |
| B=error code | |

**Link to a Module: F$NMLink** - link to module, module not mapped in user address space.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=type/language | A=type/language |
| X=module ADDR | B=module REV |
| **ERROR OUTPUT:** | X=last module ADDR byte+1 |
| CC=carry set | Y=module memory requirement |
| B=error code | |

**Load a Module: F$NMLoad** - load module(s) from file, module(s) not mapped in user address space.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=type/language | A=type/language |
| X=path ADDR | B=module REV |
| **ERROR OUTPUT:** | X=last module ADDR byte+1 |
| CC=carry set | Y=module memory requirement |
| B=error code | |

**Print Error: F$Perr 103F 0F** - writes error message to standard path.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| B=error code | CC=carry set |
| | B=error code |

**Parse Name: F$PrsNam** - scan input string for valid OS-9 name.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=path ADDR | X=optional slash ADDR+1 |
| **ERROR OUTPUT:** | Y=ADDR of last name CHAR+1 |
| CC=carry set | A=trailing byte |
| B-error code | B=name length |
| | Y=ADDR of first non-delimiter |

**Search Bits: F$SchBit 103F 12**- search memory allocation bit map for free memory block of specified size.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| D=start bit | D=start bit |
| X=map start ADDR | Y=bit count |
| Y=bit count | **ERROR OUTPUT:** |
| U=map end ADDR | CC=C bit set |

**Send: F$Send 103F 08** - send signal to process.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=process ID | CC=carry set |
| B=signal code | B=error code |

**Sleep: F$Sleep 103F 0A** - temporarily turn process off.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=sleep time (ticks) | X=sleep time (ticks slept) |
| X=0 (indefinate sleep) | |
| X=1 (sleep through current time slice) | |
| **ERROR OUPUT:** | |
| CC=carry set | |
| B=error code | |

**Set Priority: F$SPrior 103F 0D** - change priority.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=process ID | CC=carry set |
| B=priority (0-255) | B=error code |

**Set SWI: F$SSWI 103F 0E** - set SWI2 and SWI3 vectors.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=SWI type | CC=carry CLR |
| X=ADDR of interrupt | B=error code |

**Set Time: F$STime 103F 16**- set system.time and date.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| X=ADDR of time packet | CC=C bit set |
| | B=error code |

**Set User ID: F$SUser 103F 1C**- change current user ID, doesn't check for errors or caller' ID.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| Y=new user ID | CC=carry set |
| | B=error code |

**Time: F$Time 103F 15**- get system date and time.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
| X=date/time storage area | X=date and time |
| **ERROR OUTOUT:** | |
| CC=carry set | |
| B=error code | |

**Unlink: F$UnLink 103F 02**- unlinks unused module with link count of 0.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| U=ADDR of header | CC=carry set |
| | B=error code |

**Unlink Module by Name: F$UnLoad 103F 1D**- decrements module link count, removes if result is 0.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| A=type | CC=carry set |
| X=name pointer | B=error code |

**Wait: F$Wait 103F 04**- temporarily turn off calling process.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| A=process ID | CC=carry set |

## 5 - I/O Mode Calls

System call format is: **name: mnemonic, software interrupt code2, request code** (in hexadecimal). See manual for more info.

**Attach: I$Attach 103F 80**- attach or verify a device to system.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
| A=access mode | X=updated device name |
| X=device name string ADDR | U=device table entry ADDR |
| **ERROR OUTPUT:** | **ACCESS MODE PARAMETERS:** |
| CC=carry set | 0=special device abilities |
| B=error code | 1=read only |
| | 2=write only |
| | 3=update |

**Change Directory: I$CHGdir 103F 86**- change working directory.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
| A=access mode | X=updated path |
| X=path ADDR | |
| ERROR OUTPUT: | **ACCESS MODE PARAMETERS:** |
| CC=carry set | 1=read only |
| B=error code | 2=write only |
| | 3=update |
| | 4=execute |

**Close Path: I$Close 103F 8F**- terminate I/O path.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
| A=path # | CC=carry set |
| | B=error code |

**Create File: I$Create 103F 83**- create and open disk file.

| | |
|---|---|
| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
| A=access mode (write or update) | A=path # |
| B=attributes | X=last path ADDR byte +1 |
| X=path ADDR | |
| **ERROR OUTPUT:** | **ATTRIBUTE BITS:** |
| B=error code | 0=read |
| CC=carry set | 1=write |
| | 2=execute |
| | 3=public read |
| | 4=public write |
| | 5=public exec. |
| | 6=shareable |

**Delete File: I$Delete 103F 87** - delete disk file.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=path ADDR | X=last path ADDR byte +1 |
| ERROR OUTPUT: | |
| CC=carry set | |
| B=error code | |

**Delete A File: I$DeletX 103F 90** - deletes from current data or execution directory.

| ENTRY CONDITIONS: | EXIT COND.: |
|---|---|
| A=access mode | X=last path ADDR byte +1 |
| X=path ADDR | |
| ERROR OUTPUT: | |
| CC=carry set | |
| B=error code | |

**Detach Device: I$Detach 103F 81** - remove device from system.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| U=device table entry ADDR | CC=carry set |
| | B=error code |

**Duplicate Path: I$Dup 103F 82** - second path number for same (duplicate) path (used to redirect I/O).

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A= # of path to copy | A=new path # |
| ERROR OUTPUT: | |
| CC=carry set | |
| B=error code | |

**Get Status: I$GetStt 103F 8D** - status of file or device.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=path | CC=carry set |
| B=function code | B=error code |

**Make Directory: I$MakDir 103F 85** - create and initialize directory.

| ENTRY CONDITIONS: | EXIT CONDITIONS: | |
|---|---|---|
| B=ATTR | X=last path ADDR byte +1 | |
| X=path ADDR | | |
| ERROR OUTPUT | ATTRIBUTE BITS: | |
| CC=carry set | 0=read | 6=single user |
| B=error code | 1=write | 7=any user/type |
| | 2=execute | |
| | 3=public read | |
| | 4=public write | |
| | 5=public exec. | |

**Open Path: I$Open 103F 84** - open path to existing file or device.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=access mode | A=path # |
| X=path ADDR | X=last path ADDR byte +1 |
| ERROR OUTPUT: | ACCESS MODE PARAMETERS: |
| CC=carry set | D=directory |
| B=error code | E=execute |
| | R=read |
| | S=not shareable |
| | PE=public exec. |
| | PR=public read |
| | PW=public write |

**Read: I$Read 103F 89** - read number of bytes from path.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=path | Y= # of bytes to read |
| Y= # of bytes to read | |
| X=storage ADDR | |
| ERROR OUTPUT: | |
| CC=carry set | |
| B=error code | |

**Read Line with Editing: I$ReadLn 103F 8B** - read line of text and activate line editing.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=path | Y= # of bytes to read |
| X=storage ADDR | |
| Y=max # of bytes to read | |
| ERROR OUTPUT: | |
| CC=carry set | |
| B=error code | |

**Seek: I$Seek 103F 88** - reposition file pointer.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=path | CC=carry set |
| X=MS 16 bits of file | B=error code position |
| U=LS 16 bits of file position | |

*MS=most significant, LS=least.

**Set Status: I$SetStt 103F 8E** - set status of file or device.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| U=path | CC=carry set |
| B=function code | B=error code |

**Write: I$Write 103F 8A** - write to file or device.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=path | Y= # of bytes written |
| X=write data start ADDR | |
| Y= # of bytes to write | |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Write Line: I$WritLn 103F 8C** - write to file or device until carriage return.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=path | Y= # of bytes written |
| X=write data address | |
| Y=maximum # of bytes to write | |

**ERROR OUTPUT:**
CC=carry set
B=error code

## 6 - System Mode Calls

System call format is: name: mnemonic, software interrupt code2, request code (in hexadecimal). See manual for more info.

**Alarm Set: F$Alarm 103F 1E**- set alarm.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| X=time packet address | CC=carry set |
| | B=error code |

**Allocate 64 Bytes: F$All64 103F 30** - dynamically allocate 64 byte memory blocks.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| X=page table base ADDR | A=block # |
| (0=table not allocated) | X=page table base ADDR |
| | Y=block ADDR |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Allocate High RAM: F$AllHRam F$AllHRam** - allocate system memory from high memory.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| B= # of memory blocks | CC=carry set |
| | B=error code |

**Allocate Image: F$AllImage 103F 3A**-allocate RAM for DAT image

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=starting block # | CC=carry set |
| B= # of blocks | B=error code |
| X=descriptor pointer | |

**Allocate Process Descriptor: F$AllPrc 103F 4** - allocate and initialize 512 byte process descriptor.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| none | U=descriptor pointer |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Allocate RAM: F$AllRAM 103F 39** - allocates desired number of contiguous RAM blocks.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| B= # of desried blocks | CC=carry set |
| | B=error code |

**Allocate Process Task Number: F$AllTsk 103F 3F** - determine if task has number, if not allocate one.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| X=descriptor pointer | CC=C bit set |
| | B=error code |

**Insert Process: F$AProc 103F 2C** - insert process for execution into active process.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| X=process descriptor ADDR | CC=carry set |
| | B=error code |

**Bootstrap System: F$Boot 103F 35** - links boot module (or other module named in the INIT module).

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| none | CC=C bit set |
| | B=error code |

**Bootstrap Memory Request: F$BtMem 103F 36** - allocates requested memory (in blocks) in system address space.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| D=requested # of bytes | D=bytes granted |
| **ERROR OUTPUT:** | U=memory granted pointer |
| CC=C bit set | |
| B=error code | |

**Clear Specified Block: F$ClrBlk 103F 50** - marks requested blocks in DAT area as unallocated.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B= # of requested blocks | none |
| U=first block ADDR | |

**DAT to Logical Address: F$DATLog 103F 44** -convert DAT image and block offset to logical address.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B=DAT image offset | X=logical ADDR |
| X=block offset | |
| **ERROR OUTPUT:** | |
| CC=carry set | |
| B=error code | |

**Deallocate Image RAM Blocks: F$DelImg 103F 3B** - deallocate image RAM blocks.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| A=starting block # | CC=C bit set |
| B= # of blocks | B=error code |
| X=descriptor pointer | |

**Deallocate Process Descriptor: F$DelPrc 103F 4C** - free process descriptor memory for other use.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| A=process ID | CC=C bit set |
| | B=error code |

**Deallocate RAM Blocks: F$DelRAM 103F 51** - free RAM block in memory block map.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B= # of blocks | none |
| X=starting block # | |

**Deallocate Task Number: F$DelTsk 103F 40** - release specified task number.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| X=process descriptor pointer | CC=C bit set |
| | B=error code |

**Link Using Module Directory Entry: F$ELink 103F 4D** - link using pointer to directory entry.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B=module type | U=module header ADDR |
| X=directory entry | Y=module entry point pointer |
| **ERROR OUTPUT:** | |
| CC=C bit set | |
| B=error code | |

**Find Module Directory Entry: F$Fmodul 103F 4E** - return pointer to module directory entry.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| A=module type | A=module type |
| X=pointer to name | B=module rev. |
| Y=DAT image pointer | X=update name |
| ERROR OUTPUT: | U=directory entry pointer |
| CC=C bit set | |
| B=error code | |

**Find 64 Byte RAM Block: F$Find64** - find 64 byte RAM block.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| A=block # | Y=block ADDR |
| X=block address | CC=carry set if not in use or allowed |

**Get Free High Block: F$FreeHB 103F 3E** - search DAT image for highest free blocks of requested size.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B=blocks requested | A=starting block # |
| Y=DAT image pointer | |

**ERROR OUTPUT:**
CC=C bit set
B=error code

**Get Free Low Block: F$FreeLB 103F 3D** - search DAT image for lowest free blocks of requested size.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B=blocks requested | A=starting block # |
| Y=DAT image pointer | |

**ERROR OUTPUT:**
CC=C bit set
B=error code

**Compact Module Directory: F$GCMDir 103F 52** - compact entries in module directory.
**FOR INTERNAL OS-9 USE ONLY! DO NOT CALL FROM PROGRAM!**

**Get Process Pointer: F$GProcP 103F 37** - get process pointer.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| A=process ID | B=process pointer |

**ERROR OUTPUT:**
CC=carry set
B=error code

**I/O Delete: F$IODel 103F 33** - delete unused I/O module.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| X=I/O module address | CC=carry set |
| | B=error code |

**I/O Queue: F$IOQu 103F 2B** - insert calling process into another process I/O queue, put calling process to sleep.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| A=process # | CC=carry set |
| | B=error code |

**Set IRQ: F$IRQ 103F 2A** - add or remove device from polling table.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| D=device status register ADDR | CC=carry set |
| | B=error code |

X=0 to remove device
X=ADDR of packet to add device
(address at X=flip byte, X+1=mask byte, X+2 =priority byte)

**Load A from Task B: F$LDABX 103F 49** - load A from 0,X in B.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| B=task # | A=byte at 0,X |
| X=data pointer | |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Get One Byte: F$LDAXY 103F 46** - load A from X,Y

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| X=block offset (X) | A=byte at Y,X |
| Y=DAT image pointer | |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Get Two Bytes: F$LDDDXY 103F 48** - load D from D+X,Y

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| D=offset to offset in DAT image | D=bytes at D+X,Y |
| X=offset in DAT image | |
| Y=DAT image pointer | |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Map Specific Block: F$MapBlk 103F 4F** - map block into process space.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| X=start block # | U=first block ADDR |
| B= # of blocks | |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Move Data: F$Move 103F 38** - move data from one address to another.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=source task | CC=carry set |
| B=destination task | B=error code |
| X=source pointer | |
| Y= # of bytes | |
| U=destination pointer | |

**Next Process: F$NProc 103F 2D** - execute next process in active queue.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| none | control not returned to caller |

**Release a Task: F$RelTsk 103F 43** - release task from use, make hardware available for another task.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| B=task # | CC=carry set |
| | B=error code |

**Reserve Task Number: F$ResTsk 103F 42** - reserve DAT task number.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| none | B=task # |

**ERROR OUTPUT:**
CC=carry set
B=error code

**Return 64 Bytes: F$Ret64 103F 31** - deallocate 64 byte block of RAM.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=block # | CC=carry set |
| | B=error code |

**Set Process DAT Image: F$SetImg 103F 3C** - copy all or part of DAT image into process descriptor.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=start image block | CC=carry set |
| B= # of blocks | B=error code |
| X=process pointer | |
| U=new image pointer | |

**Set Process Task DAT Registers: F$SetTsk 103F 41** - write to hardware DAT registers.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| X=process descriptor pointer | CC=carry set |
| | B=error code |

**System Link: F$SLink 103F 34** - add module to current address space from outside.

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| A=module type | A=module type |
| X=module pointer | B=module rev. |
| Y=DAT image pointer name | X=updated name pointer |
| **ERROR OUTPUT:** | Y=module entry point |
| CC=carry set | U=module pointer |
| B=error code | |

**Request System Memory: F$SRqMem 103F 28** - allocate RAM from top of available RAM (for system use only).

| ENTRY CONDITIONS: | EXIT CONDITIONS: |
|---|---|
| B=module type | U=header ADDR |
| X=directory entry pointer | Y=entry point |

**ERROR OUTPUT:**
CC=C bit set
B=error code

**Return System Memory: F$SRtMem 103F 29** - deallocates block of RAM (for system use only).

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| U=start block ADDR | CC=C carry set |
| D= # of bytes requested | B=error code |

**Set SVC: F$SSvc 103F 32** - add or replace a system call. See manual for table format.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| Y=system call init. table address | CC=C bit set |
| | B=error code |

**Store Byte in Task: F$STABX 103F 4A** - store byte A at 0,X in task B. Similar to M/L instruction STA 0,X, except that X is in the task address space, not current.

| ENTRY CONDITIONS: | ERROR OUTPUT: |
|---|---|
| A=byte to store | CC=carry set |
| B=task # | B=error code |
| X=storage address | |

**Install Virtual Interrupt: F$VIRQ 103F 27** - install virtual interrupt handling routine. For use with Multi-Pak or similar device. Refer to technical reference chapter 2 for details.

| **ENTRY CONDITIONS:** | **ERROR OUTPUT:** |
|---|---|
| D=starting count value | CC=carry set |
| X=0 to delete entry, 1 to install | B=error code |
| Y=5 byte address | |

**Validate Module: F$VModul 103F 2E** - check header parity and CRC of a module.

| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
|---|---|
| D=DAT image pointer | U=module dir ADDR |
| X=new block offset | |

**ERROR OUTPUT:**
CC=carry set
B=error code

---

## 7 - Screen Color Codes

| **HEXADECIMAL CODE** | | |
|---|---|---|
| **FOREGND** | **BACKGND** | **COLOR** |
| 00 | 08 | white |
| 01 | 09 | blue |
| 02 | 0A | black |
| 03 | 0B | green |
| 04 | 0C | red |
| 05 | 0D | yellow |
| 06 | 0E | magenta |
| 07 | 0F | cyan |

10 or more (decimal - actually a hex number greater than 0F, which is nonexistant) also produces black. If the foreground color is used instead of back for a background, OS-9 will automatically translate to correct color.

## 9 - Device Windows

A device window can run a program or utility. Each window acts as if it were an individual terminal. Use the CLEAR key to page between device windows. Use ÇúINIZ /wx to open a device window, where x is one of the following window types:

| No. | Size | Type | Colors | RAM used |
|---|---|---|---|---|
| 0 | 32x16 | VDG | (requires VDGINT) | |
| 1 | 40x24 | text | 16 | 1.6K |
| 2 | 80x24 | text | 16 | 4K |
| 5 | 640x192 | gfx | 2 | 16K |
| 6 | 320x192 | gfx | 4 | 16K |
| 7 | 640x192 | gfx | 4 | 32K |

NOTE: Window numbers 8-15 are on Multi-Vue disk.

The window parameters (type, start position, and size) can be changed by following **INIZ /wx** with the **WCREATE** or **DISPLAY** commands (see below). Windows CANNOT overlap! A shell must then be forked to the new window with the **SHELL** command. **SHELL i=/wx** will create an immortal window, or one that will not die if you exit with CTRL-BREAK (refer to manual, chapter 3 for SHELL parameters).

```
 DISPLAY 1b 20 02 00 00 50 18 01 00 00 >/wx
WCREATE /wx-s= 02 00 00 80 24 01 00 00
               |  |  |  |  |  |  | border
               |  |  |  |  |  | background
               |  |  |  |  | foreground
               |  |  |  | length in rows
               |  |  | width in columns
               |  | starting Y position
               | starting X position
              window type code
```

**WCREATE** accepts decimal input, while **DISPLAY 1b 20** requires hex. **DISPLAY** is, however, always in RAM (no loading).

## Window Text Commands

The following hexadecimal codes are used with the ÇúDISPLAY command to control cursor and text in alpha and graphics windows (unless otherwise stated).

| CODE | FUNCTION |
|---|---|
| 01 | HOME- put cursor in upper right corner. |
| 02 x y | CURSOR XY- move cursor to row X of line Y. Add 20 to positions to get hex values. |
| 03 | ERASE LINE |
| 04 | ERASE TO END OF LINE (from cursor) |
| 05 20 | CURSOR OFF |
| 05 21 | CURSOR ON |
| 06 | CURSOR RIGHT (one position) |
| 07 | RING BELL |
| 08 | CURSOR LEFT (one position) |
| 09 | CURSOR UP (one line) |
| 0A | CURSOR DOWN (line feed) |
| 0C | CLEAR SCREEN |
| 0D | RETURN (carriage return) |
| 1F 20 | REVERSE VIDEO ON |
| 1F 21 | REVERSE VIDEO OFF |
| 1F 22 | UNDERLINE ON |
| 1F 23 | UNDERLINE OFF |
| 1F 24 | BLINKING ON (alpha only) |
| 1F 25 | BLINKING OFF (alpha only) |
| 1F 30 | INSERT LINE |
| 1F 31 | DELETE LINE |
| 1F 3C 00 | TRANSPARENT OFF- draw alphanumeric characters with back and fore ground (default- graphics only). |
| 1F 3C 01 | TRANSPARENT ON- draw alphanumeric characters using current background color (graphics only). |
| 1F 3D 00 | BOLD OFF (graphics only) |
| 1F 3D 01 | BOLD ON (graphics only) |
| 1F 3F 00 | PROPORTIONAL OFF- proportional character spacing (default- graphics only) |
| 1F 3F 01 | PROPORTIONAL ON (graphics only) |

## 10 - Text Editor Commands

The following commands are used with OS-9s packaged Macro Text Editor. "Pointer" refers to the "edit pointer".

**EDIT (oldfile) (newfile)** - loads and starts the built-in text editor. Oldfile is file being read in to edit, newfile a newly created file. A temporary file called **Scratch** will be created for output. Oldfile will be updated upon leaving the editor.
**MACRO (file)** - executes macro (file).
**!** - places comments, ignores following text (similar to REM).
**space** - insert line before pointer.
**ENTER** - move pointer to next line.
**+x** - move pointer x lines forward.
**-x** - move pointer x lines backward.
**+0** - move pointer to end of line.
**-0** - move pointer to start of line.
**>x** - move pointer x characters forward.
**<x** - move pointer x characters backward.
**CTRL 7** - move pointer to start of text. (CTRL space for terminals)
**/** - move pointer to end of text.
**[commands] x** - repeat commands x times.
**:** - skip to end of inermost loop or macro if fail flag not on.
**Ax** - set SEARCH/CHANGE anchor to column x. Restricts search/change to strings starting in column x.
**A0** - returns anchor to normal mode (undo Ax).
**Bx** - make buffer x primary buffer.
**Cx $1 $2** - changes next x occurences of $1 to $2.
**Dx** - delete x lines from pointer.
**Ex $1** - add $1 to end of next x lines.
**Gx** - get x lines from secondary buffer and inserts before pointer in primary buffer (starts from top of secondary).
**Ix $1** - insert line with x copies of $1 before pointer.
**Kx** - kill x characters at pointer.
**Lx** - list x lines starting at pointer.
**Mx** - change workspace size to x bytes.
**Px** - put x lines at pointer in primary buffer to pointer in secondary.
**Q** - quit editor.
**Rx** - read x lines from input file.
**Sx $1** - search for next x occurences of $1.
**Tx** - tab to column x of present line.
**U** - truncate (unextend) line at pointer.
**V (on or off)** - verify mode on/off.
**W x** - write x lines to output file.
**Xx** - display x lines before pointer.

## Built-In Macros

**.CHANGE x $1 $2** - change x occurences of $1 to $2.
**.DEL (.file)** - delete macro (.file).
**.DIR** - buffer/macro directory.
**.EOB** - test for end of buffer.
**.EOF** - test for end of file.
**.EOL** - test for end of line.
**.F** - exit innermost loop or macro & set fail flag.
**.LOAD (file)** - load macro (file). Path may be specified in (file).
**.MAC (file)** - opens macro (file) for definition. If no name specified, new macro started.
**.NEOB** - test for not end of buffer.
**.NEOL** - test for not end of line.
**.NEW** - write all text to current line to initial output file, read equal amount of text from initial input and ad to end of edit buffer.
**.NSTR $1** - test $1 for matching character at current pointer.
**.READ $1** - open $1 for reading. $1 may contain complete path.
**.S** - exit innermost loop/macro & clear fail flag.
**.SEARCH x $1** - search for x occurences of $1.
**.SAVE $1 $2** - save macro $1 on file specified by $2 ($2 may contain path).
**.SHELL command(s)** - execute OS-9 shell command(s).
**.SIZE** - display memory used and available for use in workspace.
**.STAR x** - test x for asterics (infinity).
**.STR $1** - test $1 for match at pointer.
**.WRITE $1** - open file for writing at path ($1 is path).
**.ZERO x** - test x for 0.
[- starts at macro loop (CTRL 8).
]- ends at macro loop (CTRL 9).
**[caret]** - move pointer to buffer start (CTRL 7).

## Editor Error Messages

**BAD MACRO** -illegal name.
**BAD NUMBER** -numbers can't be over 65,535
**BAD VAR NAME** - illegal variable name. May have omitted number or used $ or # signs in parameter list.
**BRACKET MISMATCH** -brackets not in pairs or nested to deep.
**BREAK** - CTRL-C or CTRL-Q pressed.
**DUPL MACRO** -attempt to close macro definition with existing name. Try different name!
**END OF FILE** -at end of buffer.
**FILE CLOSED** -attempted write to closed file. Specify file when starting editor or open with .WRITE.
**MACRO IS OPEN** -macro definition must be closed before use.
**MISSING DELIM** -couldn't find matching delimiter. String must be on one line.
**NOT FOUND** -string/macro not found.
**UNDEFINED VAR** -variable not in macro's parameters.
**WHAT??** -unrecognized command. Check spelling!
**WORKSPACE FULL** -buffer full. Increase size or delete text.

## 11 - Mouse Get Status System Calls

Get Status System calls are used with the RBF manager routine GETSTA. The Get Status routines pass the register stack and specified code to the device driver.

SS.Mouse passes information on the current mouse (or joystick) and its fire button(s). A 32 byte data packet is created by SS.Mouse that provides position and button information. "A"= Button A, "B"= Button B. Button B is the second button on two button devices (black on Deluxe Joystick, right on two-button mouse). The mouse is scanned every time the keyboard is scanned. Support modules are CC3IO, GrfInt, and WindInt.

Only the mouse status is referred to due to its frequent use. Refer to the manual for additional Get Status system calls

**SS.Mouse** - function code $89

| | |
|---|---|
| **ENTRY CONDITIONS:** | **EXIT CONDITIONS:** |
| A=Path # | X= data storage area ADDR |
| B=$89 | |
| X=data storage area ADDR | |
| Y=mouse port select | **ERROR OUPUT:** |
| 0=automatic | CC=carry set |
| 1=right port | B=error code |
| 2=left port | |

**DATA PACKET:**
**Pt.Valid** (rmb 1) 0=info not valid, 1=info valid.
**Pt.Actv** (rmb 1) 0=port off, 1=right active (default), 2=left active
**Pt.ToTm** (rmb 1) initial timeout (countdown) value.
**Pt.TTTo** (rmb 1) time until timeout
**Pt.TSSt** (rmb 2) time since counter start
**Pt.CBSA** (rmb 1) "A" state(0=button open)
**Pt.CBSB** (rmb 1) "B" state(0=button open)
**Pt.CCtA** (rmb 1) "A" click count
**Pt.CCtB** (rmb 1) "B" click count

**Pt.TTSA** (rmb 1) "A" time this counter
**Pt.TTSB** (rmb 1) "B" time this counter
**Pt.TLSA** (rmb 1) "A" time last counter
**Pt.TLSB** (rmb 1) "B" time last counter

**TTSx** is the number of clock ticks that have passed during the current button state as defined by Pt.CBSx. Starts at one and increases as long as button remains in same state. TLSx is the number of ticks that a button is in the opposite state. Using these counts, one can determine how a button was even if the system returns the packet changes after the button state changes. Use to define clicks, drags, holds, etc. (along with state and click count).

**RESERVED (rmb 2) FOR FUTURE USE**
**Pt.BDX** (rmb 2) button down frozen (X). Copy of Pt.AcX when button changes states(open to closed/closed to open).
**Pt.BDY** (rmb 2) button down frozen (Y). Copy of Pt.AcY when button changes states(open to closed/closed to open).
**Pt.Stat** (rmb 1) window pointer type location. 0=current working area, 1=control region (for Multi-View), 2=out of current window. If Pt.Valid=0, then Pt.Stat=0 and is not accurate.
**Pt.Res** (rmb 1) resolution. 0=low (X:10,y:3), 1=high (x,y:1). The Hi-Res adapter or keyboard mouse must be used for high resolution.
**Pt.AcX** (rmb 2) actual X resolution value. 0-639 for high, 1:10 for low.
**Pt.AcY** (rmb 2) actual Y resolution value. 0-191 for high, 1:3 for low.

**Pt.WRX** (rmb 2) window relative X
**Pt.WRY** (rmb 2) window relative Y
Values read from mouse minus starting coordinates of the current window working area. Divide window relative values by 8 for absolute character positions.

**Pt.Siz** (equ.) packet size

**SPt.SRX** (rmb 2) screen relative X
**SPt.SRY** (rmb 2) screen relative Y
**SPt.Siz** (equ.) size of packet
**FOR SYSTEM USE ONLY**

## 12 - Error Codes

OS-9 built-in error codes are listed. Entries are **decimal number, (hexadecimal number), name.**

**001 ($01) UNCONDITIONAL ABORT**- unrecoverable error has occured, all processes terminated.
**002 ($02) KEYBOARD ABORT**- BREAK pressed.
**003 ($03) KEYBOARD INTERRUPT**- SHIFT BREAK was pressed.
**183 ($B7) ILLEGAL WINDOW TYPE**- wrong type window or parameters for given command.
**184 ($B8) WINDOW ALREADY DEFINED**- tried to create window using an existing designation.
**185 ($B9) FONT NOT FOUND**- window font not loaded or in specified/current path.
**186 ($BA) STACK OVERFLOW**- not enough stack space free.
**187 ($BB) ILLEGAL ARGUMENT**- wrong parameters or commands
**189 ($BD) ILLEGAL COORDINATES**- coordinates off screen.
**190 ($BE) INTERNAL INTEGRITY CHECK**- system modules or data no longer reliable.
**191 ($BF) BUFFER SIZE IS TO SMALL**- data larger than buffer.
**192 ($C0) ILLEGAL COMMAND**- command form not OS-9.
**193 ($C1) SCREEN OR WINDOW TABLE IS FULL**- system can't keep track of any more windows or screens.
**194 ($C2) BAD/UNDEFINED BUFFER NUMBER**- illegal buffer.
**195 ($C3) ILLEGAL WINDOW DEFINITION**- parameters illegal.
**196 ($C4) WINDOW UNDEFINED**- tried access to undefined window.
**200 ($C8) PATH TABLE FULL**- can't track any more files.
**201 ($C9) ILLEGAL PATH NUMBER**- number to large or does not exist.
**202 ($CA) INTERRUPT POLLING TABLE FULL**- interrupt can't be handled, no room for more entries in table.
**203 ($CB) ILLEGAL MODE**- device can't perform function.
**204 ($CC) DEVICE TABLE FULL**- no more devices can be added.
**205 ($CD) ILLEGAL MODULE HEADER**- sync code, header parity, or CRC of module bad.
**206 ($CE) MODULE DIRECTORY FULL**- modules can't be entered.
**207 ($CF) MEMORY FULL**- no more available memory.
**208 ($D0) ILLEGAL SERVICE REQUEST**- issued system call has illegal code.
**209 ($D1) MODULE BUSY**- non-shareable module in use.
**210 ($D2) BOUNDARY ERROR**- memory allocation or deallocation not on a page boundary.
**211 ($D3) END OF FILE**- read terminated.

**212 ($D4) RETURNING NON-ALLOCATED MEMORY**- attempt to deallocate non-allocated memory.
**213 ($D5) NON-EXISTING SEGMENT**- file structure of device bad.
**214 ($D6) NO PERMISSION**- user doesn't have access to perform specified operation.
**215 ($D7) BAD PATHNAME**- syntax error in path.
**216 ($D8) PATH NAME NOT FOUND**- can't find path.
**217 ($D9) SEGMENT LIST FULL**- file to fragmented.
**218 ($DA) FILE ALREADY EXISTS**- file exists in directory
**219 ($DB) ILLEGAL BLOCK ADDRESS- device file structure bad.**
**220 ($DC) PHONE HANGUP - DATA CARRIER LOST**- no carrier on RS-232 port.
**221 ($DD) MODULE NOT FOUND**- module not in directory.
**223 ($DF) SUICIDE ATTEMPT**- attempt to return to stack.
**224 ($E0) ILLEGAL PROCESS NUMBER**- non-existant process.
**226 ($E2) NO CHILDREN**- wait service issued but no dependants.
**227 ($E3) ILLEGAL SWI CODE**- software interrupt <1 or >3.
**228 ($E4) PROCESS ABORTED**- current process terminated.
**229 ($E5) PROCESS TABLE FULL**- no more processes can be run.
**230 ($E6) ILLEGAL PARAMETER AREA**- fork passed bad boundaries.
**231 ($E7) KNOWN MODULE**- module for internal use only.
**232 ($E8) INCORRECT MODULE CRC**- bad module CRC.
**233 ($E9) SIGNAL ERROR**- receiving process has signal pending.
**234 ($EA) NON-EXISTANT MODULE**- module can't be found.
**235 ($EB) BAD NAME**- illegal name used.
**236 ($EC) BAD HEADER**- module parity header bad.
**237 ($ED) SYSTEM RAM FULL**- bootfile (OS9Boot) to large.
**238 ($EE) UNKNOWN PROCESS ID**- incorrect ID number.
**239 ($EF) NO TASK NUMBER AVAILABLE**- all in use.
**240 ($F0) UNIT ERROR**- device unit doesn't exist.
**241 ($F1) SECTOR ERROR** sector # out of range.
**242 ($F2) WRITE PROTECT**- device write protected.
**243 ($F3) CRC ERROR**- bad CRC on read/write verify.
**244 ($F4) READ ERROR**- disk read data error or SCF input overrun (terminal)
**245 ($F5) WRITE ERROR**- error during device write.
**246 ($F6) NOT READY**- device not ready.
**247 ($F7) SEEK ERROR**- seek attempted on non-existant sector.
**248 ($F8) MEDIA FULL**- not enough free disk space.
**249 ($F9) WRONG TYPE**- attempt to read incompatible disk.
**250 ($FA) DEVICE BUSY**- non-shareable device in use.
**251 ($FB) DISK ID CHANGE**- disk changed with files still open.
**252 ($FC) RECORD IS LOCKED OUT- record is being used.**
**253 ($FD) NON-SHAREABLE FILE BUSY**- file being used.

**PERSONAL NOTES:**