Image BBS 1.2 Programmer's Reference Guide

Chapter IV: Disk File Formats

Chapter V: Variable Usage

Preface

This manual assumes the user has a working knowledge of the BASIC programming language. If you find you have trouble with areas of basic, it is recommended you refer to a good programming book, such as Commodore 64 Programmer's Reference Guide, published by Howard W. Sams & Company.

The main purpose of this manual is to familiarize the reader with areas of Image that may be accessed in order for the reader to write modifications or modules to enhance their system.

The scope of this book is intended to include: basic subroutines, reserved system variables, other important areas of memory in use, disk file handling, execution and use of mini-plus files, data formats of various system files.

This manual was made with every intent to provide a readable, accurate, and exciting way to program Image like never before. Enjoy the manual.

New Image Software

Welcome to the world of programing IMAGE BBS! This manual is designed to allow both the novice and advanced programmer the abilities to program IMAGE BBS modules. We will cover four major areas of IMAGE BBS, General Information, the basic program, machine language routines, disk file formats, and variable usage charts. Also included you will find a special question and answer section of the most often asked questions and the answers to those questions.

IMAGE BBS will support virtually all hardware available for the C64 including the Lt. Kernal (c) hard drive and the CMD HD series (c) hard drives. Also, the Swiftlink (c) rs232 cartridge by Dr. Evil.

IMAGE BBS is a modular type program consisting of both basic and machine language modules. A main basic module ("im") and a main machine language module ("ml 1.2") that both remain in memory at all times. Also used are basic modules (+.) plus files and machine language (++) modules.

The purpose of this manual is to provide you with necessary subroutines and variable considerations to make your own modules or modify the existing system to suit your needs.

I would like to thank all of those who worked hard in taking NEW IMAGE SOFTWARE to where it stands today.

DON GLADDEN (C Tuna) for the original BASIC program

RAY KELM (Professor) The ML whiz kid, for the ML.

FRED DART (The Chief) For hours upon hours of work and always being available for support.

JOHN MOORE (Little Work) For Continuous sleepless nights spent to better the software and its subsystems. Also for the much awaited 128 version.

BOB LEARY (Dr.BOB) For Countless hours of work racking my brains trying to get this manual done.

NISSA for providing undue support to all those sysops in their time of need.

Chapter I: General Information

There are several things which are often used and would fit the description of general information. The items covered in this chapter are used in almost every module you can find.

The Functions Keys:

Image bbs translates specific ascii characters into function keys to make the task of programming easier and also to allow for characters which could cause garbage in sequential data files.

```
F1 = comma (,)      F2 = question mark (?) F3 = colon (:)
F4 = equals (=)     F5 = quote (")         F6 =  chr$(13)
F7 = asterisk (*)   F8 =up arrow ()
```

The Drive Designators:

IMAGE BBS uses six drive identifiers to identify the system drive in which to find a specific file. The active drive is set by setting the variable dr to a value of one to six as I will describe below.

1 - System drive, this drive should contain all of your s.files. When writing your own modules all files which are usually read but NOT changed often (like intro screens) should carry the s. prefix and be stored on this device.

2 - E-mail drive, this drive contains all mail, force mail and network files.

3 - Etcetera drive, this one of the most active drives on your system. This drive contains all file which change often and should be used to store data which needs to change frequently.

4 - Directory drive, this drive holds all of the sub-directories used by the system.

5 - Plus File drive, this drive holds all of the IMAGE BBS program modules.

6 - User drive, this drive is used to store the user records and alphabetical user list.

Time and date format:

IMAGE BBS uses a 11 digit string to represent the the current time and date. This information is stored in the string variable d1$.

19011038045

The first character in the string represents the day of the week from 1 (sunday) to seven (saturday). The second and third digits together represent the year, fourth and fifth the month, six and seventh the date. The eighth and ninth digits represent the hour. If the value of this position is 00-11 then it is AM any value larger than eighty denotes PM and you must subtract 80 from the value to get the correct time. Last but not least the tenth and eleventh digits represent minutes.

Carrier Loss:

All prompts for user input must be protected against a LOST carrier! This is simple and easily forgotten. If the carrier is lost the system will automatically drop the users time remaining to zero. Time remaining is stored in the integer tr%. The simplest way to protect your system is to simply check the value of tr%, if it is less then one either goto line 1811 or, return if it is a subroutine.

User Flags:

Each user has a total of fifteen which decide whether or not they have access to different areas of the system. These flags are held in a fifteen digit string each digit representing a separate flag which can be checked by setting the variable a to the flag number (1-15) and issuing a gosub 1004 command. If a returns with a value of zero access is denied. The values of each flag are as follows in the chart below.

1 - Non-Weed Access	2 - Credit Ratio
3 - Local Maint	4 - Post/Respond
5 - U/D Access	6 - Editor Lines
7 - Unlimited Credits	8 - Remote Maint
9 - E-Mail Access	10 - User List Access
11 - BAR/Log Access	12 - Sub Maint
13 - Files Maint	14 - MCI Access

15 - Prime Time U/D Access

Access Levels:

IMAGE BBS uses a base two number system to determine which access groups (0-9) have access to separate areas of the system. In a base two number system the value of a actual number is numberm therefore 0 would be 1, one would equal 2, etc, etc. Below is a listing of the numbers 0-9 as used by IMAGE.

0 = 1	1 = 2	2 = 4
3 = 8	4 = 16	5 = 32
6 = 64	7 = 128	8 = 256
	9 = 512	

To figure out which access groups would have access to certain areas you would add the base two numbers of the access groups you wish included. For example if you wanted all access groups to have access to the function you would need to have a access code of 1023, now if you just wanted levels 8 and 9 to have access the code would now be 768.

Plus Files (+.):

A plus file is a basic module which is loaded into memory when needed by the bbs. The maximum size that any one plus file can be is 56 cbm blocks. There are two distinct different types of plus files , the standard plus file and the mini plus file. The standard plus file can range from lines 1 to 999 under NO circumstances with any plus file should you allow your program to fall through past line 999. The last line of the program should always be a remark. Depending on the type of module it should return control to the main system via either a goto 1811 or a return. A mini plus file is a smaller module which may be loaded between the plus file and the main basic program. Care should be taken when using these types of modules. I will cover more on these modules further on this chapter.

ML modules (++):

A Ml module is normally used for such items a transfer protocols and copiers. However they can be used for any function that you wish. Al ML modules must reside in the c000 to ca00 range of memory.

Mini plus files:

A mini plus file is a module or series of modules which link to a plus file module in much the same way as a plus file module links to the main module. When using mini modules the size of the plus file is limited to 40 cbm instead of 56 cbm blocks. Also the mini module may range from one line higher than the main module (the first line must be a remark) to line 999. Mini plus files are loaded into memory via the &,7 ml call which is described later on in the ml portion of this manual. Below is a example of how a standard module using mini plus files should look.

```
lines 1 - 798  main basic module
line 799       remark
--------------------------------
line 800       remark (first line of LMP)
lines 801-998  basic mini module
line 999       remark
```

Chapter II: Basic Sub-routines

The purpose of this chapter is to provide you with all existing sub-routines available for your use. All routines are shown being accessed through the recommended entry points. Some routines may be accessed through several entry points while others may not. Failure to use the proper entry points of the routines provided could cause undue garbage on the CPU stack or other problems with your program.

1001 position record of relative file currently open as lfn #2. the variable x should be set to the number of the record desired prior to calling this routine.

1004 checks status of user flags 1-15. The variable 'a' must be set for the appropriate flag 1-15 before calling this routine. The routine will return either a 0 or a 1 in the variable 'a'. (0=not accessible, 1= access allowed). See Table 1 below.

Table 1

Flag (a)	Controls
1	Non-Weed
2	Credit Ratio
3	Local Maint
4	Post/Resp.
5	UD/UX Access
6	Max Editor Lines
7	Unlimited D/L's
8	Remote Maint.
9	E-Mail Access
10	User List Access
11	BAR/Log View
12	Sub Maint
13	Files Maint
14	MCI Access
15	U/D At Prime Time

1005 Upper/Lowercase input. Displays prompt currently stored in the variable 'p$' and waits for user input followed by a return. Characters allowed is determined by poking the length to location 53252,len (0-255). All input is returned in the variable 'an$'.

1006 Uppercase Input. Same as 1005 above only all input is forced uppercase only. Also all input is stored in 'u$' as well as 'an$' for command stacking.

1007 'Hotkey' Input, Waits for a keypress, value is returned in 'an$'.

1009 Set Drive, Sets active system drive # 1-6. The variable 'dr' must be set with a value of 0-6. The active device will be returned in dv%, active drive+":" will be returned in dr$. Refer to Table 2 below.

Table 2

dr	Sys. drive
0	Dev, Drive designated by PF,TF,MF,RF
1	System files disk (s., n.)
2	E-mail Disk (m., nm.)
3	Etcetera Disk (e.)
4	Directory Disk (d.)
5	Plus File Disk (+., ++, scn.)
6	User Disk (u.)

* The values in Table 2 apply to all references of the variable 'dr'.

1010 Closes and then (re)opens command channel (lfn#15) on device, drive requested by setting the variable 'dr' as explained above.

1011 Open File, Opens filename stored in a$ on device dr. The variables dr and a$ must be set prior to calling this routine. It will automatically set the drive, and open the command channel. A value of other than 0 returned in 'e%' indicates a drive error.

* When opening a sequential data file ,s,(r,w,a) must be specified in a$. When creating a relative file ,l,+chr$(x) must be specified.

1012 Read Error channel of currently active device,drive. a$ will return status message. e% = error #, e$ = error message, t% = track. s% = sector.

1013 Loads Plus file module located in a$ from plus file disk. This routine will also store the filename in cm$ and output it to the "AREA" window of the sysop screen. Please not that a$ should equal the filename WITHOUT the +. on the beginning. The +. and drive designators are automatically added by the sub-routine.

1016 Same as 1013 above only routine will load +.file and proceed to line 1 of the module if no error is encountered.

1023 Scratch and Replace, This routine should ONLY be used for sequential data files. The routine will scratch the file designated in a$ on device,drive dr and re-open it for sequential write on the same device, drive.

1024 Scratch File, This will scratch filename in a$ on device dr.

1025 BAR Stat, This routine will add the value of 'i' to the variable st(x) which holds the values of the various statistics of the Board Activity Register. The x should be set according to table 3 below to equal both the dimension and record number of the file e.stats in which the information is stored. 'i' should be set to the value you wish added to dimension (X) of the array.

* note that 1025 will fall through to 1026 before returning.

Table 3

	LAST	LOG	CURRENT	TOTAL
FBACK	1	12	23	30
Sys MAIL	2	13	24	31
Usr MAIL	3	14	25	32
POSTS	4	15	26	33
RESPS	5	16	27	34
U/L's	6	17	28	35
D/L's	7	18		36
NEW Usr	8	19	29	
CALLS	9	20		
USED	10	21		
IDLE	11	22		

x = 37 Total minutes system used (overall).
x = 38 Total minutes system idle (overall).

1026 Print the value of st(x) to record x of relative file currently open on lfn# 2. e.stats should be opened opened prior to calling this routine.

1027 Sets active device, drive to the etcetera drive and opens filename e.+b$+,s,+a$. b$ should be equal to the filename without the e. and a$ should equal either a,r or, w depending on if the file is needed for read ,write or, append.

1030 read currently open sequential file lfn # sr until the "" character is encountered, reset output defaults and return. sr should be set to the lfn #of the currently open file. Under normal circumstances the lfn# should be 2.

1034 Deletes user handle contained in an$ from alphabetical index. an$ must be equal to name of user to remove prior to calling this routine.

1038 Insert user handle contained in an$ to alphabetical index. an$ must be equal to name of user to insert prior to calling this routine.

1046 Search alphabetical index for user name in an$. an$ must be set to the name to search prior to calling this routine. The variable 'i' will return a value of 0 if no match is found. Otherwise, it will return the id number of the user name in an$.

1060 opens filename e.stats on your system etcetera disk.

1062 opens e-mail file for desired user. tt$ should equal the name of the user in which to send mail and a$ should equal either "r" or "w" depending on whether the file is to be opened for read or write.

1063 opens relative file e.data on your etcetera disk.

1064 opens the daily log. a$ should be set to a,r,w depending on either you want the file to read, write or append.

1065 opens relative file u.config on the user file disk

1067 load plus file module in a$, print filename in "AREA" window and proceed to line 1 of module.

1070 assembles user online's stats into a$ (each separated by a chr$(13)), positions relative file pointer and prints a$ to the u.config file. The u.config file must be opened prior to calling this routine.

1075 clear screen and output sequential file in a$ from device,drive dr to screen and modem.

1076 output sequential file in a$ from device,drive dr to screen and modem.

1079 reads blocks free and updates sysop screen for device, drive dr. Blocks free is returned in the variable bf.

1089 Disk directory for device, drive dr.

1093 clears arrays all programmable arrays.

1095 prints a$ to system log and active printer at device 4.

1096 prints a$ to printer at device 4.

1098 Resets system to default outputs, prints current value of p$ to the "AREA" window and waits for uppercase input of up to 38 characters.

1099 Prints first 11 characters of p$+":"+first 4 characters of an$ to "AREA" window.

1300 Loads module contained in z$ (minus the +.) and GOSUBS to line 1 of the module. This routine is for ALL level command type modules like +.ST, +.XP, +.LD, etc. These type of modules must return control to the main program with a RETURN.

1301 Loads module contained in z$ (minus the +.) and continues to line 1 of the module unless an error is encountered then it will return to the main prompt. This is a good routine to load main level command modules like +.BB and +.CP.

1348 Outputs sequential file s.new user to screen and modem.

1349 Outputs sequential file s.config to screen and modem.

1350 Clears screen and outputs sequential file s. + a$ to screen and modem.

1351 Clears screen and outputs s.menu+ (lc) to screen and modem and is automatically followed by s.menu 1 if lc is greater than 1.

1354 Read seq file from drive x. an$ must be set to Command/device,drive (Example RD8,0). If no device or drive is designated then read will be from device 8,0.

1360 Print cm$ to the sysop screen 'AREA' window.

1375 Print user online's computer type to chat message window.

1376 Print an$ to chat message window

1450 Prompt for device and drive, then prompt for disk command before returning. Note:if n0 or s0 you will be prompted with an 'Are you sure?' '@' will return status of specified drive.

1460 Will add credits in the amount specified to user online.

1470 Input routine for device, drive prompt.
(A value of 1-6 will return system drive,
A value of 7-15 will return device number.
If the device number is followed by a comma and a number of 0-255, the second number represents the drive/lu. The value of the device will be returned in d1%. The drive value will be returned in d2%. Default drive is drive 8,0.

1490 Print a$ to daily log.

1500 Translate IMAGE encoded F keys to text.

1520 Time of day clock routines.

1604 Editor entry routine. All lines typed into the editor will be stored in tt$(x). kk will equal the number of the last line entered. If kk = zero when returning from this routine then either the editor was aborted or no text was entered.

1610 Alternate Editor Entry. This entry point for the editor will NOT clear tt$(x) or kk on entry. Any text already loaded into tt$(x) will become text in the editor. This entry point should ONLY be used in cases where text is needed in the editor at entry like a EDIT function.

1634 Change Access, This routine changes the users flags to that of the access group it has been changed to via the Acs checkmark. The users new access is stored in the integer ao% upon entry and ac% on exit.

1640 Chat Request, this is the subroutine for a chat request. The first time a user requests chat he will be asked for a reason for chat. Every time after that he will be told that the page is on.

1656 Time & date, Prints current logon time and minutes left this call.

1678 Feedback. This routine lets the user leave feedback to the sysop. Each user is allowed to leave 2 feedbacks per call.

1694 Logoff, This routine will log the user off of the system and reset it for incoming calls. If O or OFF is entered the user will be asked if he wishes to Logoff. If O is followed by a # then the users last call date will NOT be updated. If the last character of the command is ! then the user will not be asked to logoff but will be immediately disconnected and the system reset.
Note that on a normal logoff the user last date is updated to be the time that that user logged on the bbs, not the time of logoff.

1736 Load ML Module, This is the routine which loads the ML modules into memory at c000 (49152). the variable a should be equal to the number of the ML module you wish to load.

a = 0 Punter (++ 0)	a = 1 Xmodem (++ 1)
a = 2 Copier (++ 2)	a = 3 R-Punter (++ 3)
a = 4 Indexer (++ 4)	a = 5 Clock (++ 5)

1811 Main System prompt entry (clear arrays)

1812 Alternate Main system prompt entry.

1901 Output "Are You Sure" and fall through to the getkey Yes / No routine.

1902 Getkey, waits for user to hit a key (Y/N). If Y is entered Yes! will be output and the variable a will equal one. If any other key is hit NO will be output and the variable a will equal zero.

1903 Same as above only you must add your own getkey (gosub1007) to the routine.

1908 Prints current time to screen and modem. Also Minutes left if user does not have unlimited time.

1910 SAY, read a random quote from the e.say data file and output it to the screen and modem.

1914 RESET, resets all system output to default parameters

1915 ZZ, check for console local mode.

1920 Abort, outputs ABORTED! to screen and modem

1980 Outputs text from record #x of e.text.

1985 Outputs Sorry + b$ + Limit Exceeded.

1989 Outputs Illegal Command message.

2000 - 2015 error trap

Chapter III: Machine language Routines

Jump Table Routines (& commands) V1.2

0 - outastr

Description: Output routine, handles MCI

Usage: &

Examples:
```
&        (print a$)
&,0      (print a$)
&"text"  (print "text")
```

1 - inline

Description: Input Routine. Handles word wrap, MCI access, line length, and and graphics. Before calling, use: "poke53252,line len" to set the line length. W$ is text that is wrapped, and AN$ holds the inputted text.

Usage: &,1,x,y

x= Bit/Val	Use
0/1	Graphics Allowed
1/2	"." on Column 1 Exits
2/4	Disable P$ Prompt
3/8	Allow MCI
4/16	Text Wrap On
5/32	Edit Mode On
6/64	Ignore Time Expired
7/128	Delete on Column 1 Exits

y=

Examples:
```
poke53252,10:p$="Name?":&,1
(Prompts user for name, and allows 10 characters to be entered)

poke53252,20:w$=na$:p$="Handle":&,1,32:ifan$<>""thenna$=an$
(prints prompt, then old handle, and then the prompt again, and
 allows the user to enter a new handle)

poke53252,30:p$="New Prompt":&,1,9:po$=an$
(allows user to enter a new main prompt, and allows graphics and
 MCI to be entered)
```

2 - dskin

Description: File Input routine, will input a maximum of 80 characters into the variable A$. Also allows bytes to be read directly from a file into a variable.

Usage:
```
&,2,File #,0  - normal input, stop at chr$(13)
&,2,File #,N  - input up to N bytes into A$, ignore chr$(13)
```

3 - read0

Description: File Read Routine, also allows an optional speed variable for Movie Files. Using a speed of 0 uses the normal file read, whereas using any other value uses the movie file read, with an appropriate slowdown based on the speed value.

Usage:
```
&,3,File #        - Normal file read
&,3,File #,speed - View Movie File
```

4 - getmdm

Description: Get a character from the modem. This returns the character that was received in location 780. This routine does no ASCII translation, and no high bit stripping. It gets the character directly from the RS232 routines.

Usage:
```
&,4:a=peek(780)
```

- getversn

Description: Get the version # information that is embedded into the ML. This puts the revision # into A%, and the revision date into A$.

Usage:
```
&,5
```

6 - password

Description: Password input. Sets text length to 14 characters, and inputs a password. Mask character is printed rather than the real character that is typed. Password is returned in AN$.

7 - prg

Description: Loads modules into memory. The filename and drive number are in A$, and the device number is passed directly in the command.

Usage:
```
dr=5:gosub1010:a$=dr$+"+.filename":&,7,dv%
(loads a + file from the + file disk)
dr=5:gosub1010:a$=dr$+"+.mm.file":&,7,dv%,1
(loads a Mini-Module from the + file disk)
dr=5:gosub1010:a$=dr$+"+.
```
See the section on "Using Modules" for more information.

8 - dskdir

Description: This will read the directory from a file that was opened as a directory channel.

Usage:
```
dr=1:gosub1010:open2,dv%,0,"$"+dr$+"*":&,8,2:close2
(This reads the directory from the system disk)
dr=2:gosub1010:open2,dv%,0,"$"+dr$+"m.*":get#2,a$,a$:&,8,2,1:
&,8,2,1
(this reads the entry for the first Email file into A$)
```

9 - btmvar

Description: This displays the contents of a variable at the bottom of the screen. The variable displayed can be either A$ or AN$.

Usage:
```
&,9     - Display AN$
&,9,1   - Display A$
```

10 - term

Description: This is the terminal mode that is used in Image Term. To exit term mode, press the C= and Ctrl keys together.

Usage: &,10

- clrarr

Description: This is used to erase the contents of an Array. Any of the arrays can be cleared (except BF())

Usage:
```
&,11      - Clear TT$()
&,11,1    -
&,11,2    -
```

12 - newuser

Description: This is the same as the normal file read, except that the file read is non-abortable, and the speed option is not supported.

Usage: &,12,File #

13 - inchr

Description: This is the Get Character routine. Like the MCI Get (#G). It waits for a character from either the keyboard or the modem, and returns it in AN$.

Usage: &,13

14 - bell1

Description: Sounds a bell

Usage: &,14

15 - convan

Description: Converts AN$ from 11 byte date to text.
The resulting text string is placed into AN$.

Usage: &,15

16 - sys49152 (call proto)

Description: This does a SYS49152

Usage: &,16

17 - sys49155 (call proto)

Description: This does a SYS49155

Usage: &,17

18 - setmode

Description: Set the screen mode (1=split,0=full)

Usage: &,18,0 - Set full screen mode
&,18,1 - Set split screen mode

19 - disp1

Description: Display top for User Online. This loads the appropriate "scn." files.

Usage: &,19,Device#,Drive#

20 - disp2

Description: Display top for System Idle

Usage: Same as 19.

21 - disp3

Description: Display bottom.

Usage: Same as 19.

22 - tenwait

Description: Time Delay. This will delay for any interval between .1 second to 25.5 seconds, in 1/10 second steps.

Usage: &,22,Tenths

23 - xgetin

Description: This is a get character routine that should be used when writing ML routines that need to get a character from the user. The character is returned in location $FE.

24 - xchrout

Description: This is an output character routine that should be used when writing ML routines that need to output a character to the user.

25 - sound

Description: Makes various sounds.

Usage:
```
&,25     (Beep)
&,25,1   (Siren)
&,25,2   (Bell)
```

26 - getmdm

Description: Same as 4.

27 - arraysav

Description: Save variable pointers. This saves the pointers that tell where BASIC Variables and Arrays start and end, so they can be later restored to erase unnecessary variables.

Usage: &,27

28 - arrayres

Description: Restore array pointers

Usage: &,28

29 - usevar
Description: Get contents of a variable. This is the routine to call to read the value of a variable from machine language. The X register holds the variable # to access. (see variable table) The contents of the variable is read into the buffer at $61.

30 - putvar
Description: Put value into a variable. This stores the contents of the buffer at $61 into a variable. This routine is meant only to be called from ML routines. The X register holds the variable #. (see variable table)

31 - zero
Description: Set value to Floating Point 0. This stores the floating point equivalent of 0 into the buffer at $61.

32 - minusone
Description: Set Value to Floating Point -1. This stores the floating point equivalent of -1 into the buffer at $61.

33 - getarr
Description: Get descriptor for TT$(X). This gets the descriptor (length and pointer) of an element of the TT$ array. The element # is passed in the X register. The descriptor is stored in the buffer at $61.

34 - putarr
Description: Put descriptor for TT$(X). This stores the buffer at $61 as an element of the TT$ array. The element # is passed in the X register.

35 - getln

Description: Get string for TT$(X) into buffer. This gets the descriptor for an element of TT$() into the buffer at $61, AND gets the corresponding string into the general text buffer at $ce77.

36 - putln

Description: Put string in buffer to tt$(X) (used in protos)

37 - trapon

Description: Turn error trap on

Usage: &,37

38 - trapoff

Description: Turn error trap off

Usage: &,38

39 - prtln

Description: Print tt$(X) (used in protos)

40 - forcegc

Description: This routine will force a fast garbage collect. This can be useful when you need to find out exactly how much memory is available.

41 - setbaud

Description: This sets the baud rate that Image will use. Note that you should not change the baud rate while someone is connected, since this only changes the rate at which IMAGE sends/receives, and the modem will not follow.

Usage: &,41,X
X=0, 300 Baud
X=1, Not Used
X=2, 1200 Baud
X=3, 2400 Baud

42 - reserved (Internal Usage)

43 - reserved (Internal Usage)

44 - chatchk

Description: Checks for the presence of the CHT check mark.

45 - prtvar

Description: Will print a variable.

46 - prtvar0

Description: Will print a variable with MCI forced OFF.

47 - carchk

Description: Will check for the presence of a carrier.

48 - getkbd

Description: Gets a character from the keyboard.

49 - getmod

Description: Gets a character from the modem, with ASCII translation.

50 - outscn

Description: Outputs a character to the Image window.

51 - outmod

Description: Outputs a character to the modem, with ASCII translation.

52 - chkflags

Description: This is the BASIC interface to the LightBar.
Usage: Described in the section called "The LightBar".

53 - disp4

Description: Loads the display for the term menu.

54 - editor

Description: This is the BASIC interface to the Editor.
Usage: Described in the section "The Editor".

NOTES:
A=1:
The flags that can be used for the input routine are in the form of bits, each value turns one thing on or off:

```
  1 = Allow Graphics Characters
  2 = Disable "." on column 1
  4 = Disable P$ (prompt)
  8 = Disable "£" key
 16 = Word Wrap on
 32 = Edit mode on
 64 = Ignore time out
128 = Disable DEL on column 1
```

A=9,29,30,45,46:

The variables that can be accessed by these functions are:

```
 0 AN$   1 A$    2 B$    3 TR$   4 D1$   5 D2$   6 D3$   7 D4$   8 D5$   9 LD$
10 TT$  11 NA$  12 RN$  13 PH$  14 AK$  15 LP   16 PL   17 RC   18 SH   19 MW
20 NL   21 UL   22 QE   23 RQ   24 AC%  25 EF   26 LF   27 W$   28 P$   29 TR%
30 A%   31 B%   32 DV%  33 DR$  34 C1$  35 C2$  36 CO$  37 CH$  38 KP%
```

Locations used By Image ML:

	Name	Use
97	var	store value of variable for usevar/putvar/zero/minusone/getarr/ putarr/getln/putln etc.
830	idlemax	Maximum idle time allowed
2024	mjump	# line to skip for £j,£e,£d
2025	mresult	Result of £t,£a
2026	mspeed	Print Speed
2027	mprint	print mode
2028	mcolor	Current Color
2029	mprtr	Printer flag (enabled with £L1)
2030	mreverse	Reverse mode flag
2031	mci	MCI ON/OFF flag
2032	mdigits	# digits for £%
2033	carrst	Carrier Flag.
2034	tsp1	Transmit speed lo byte
2035	tsp2	Transmit speed hi byte
2036	chks	Checkmark flags for LightBar (left=2036, right=2037)
2038	readmode	Store the unabortable read flag
2039	filenum	Stores the File # for read0/dskin
2041	abtchr	Alternate Abort Character. (works like " ")
2042	clock	Turns on BIG clock
2043	filetyp	Filetype for punter
16384	tempbott	Temporary storage for bottom of screen (for full screen)
16464	tempbotc	Temporary storage for bottom color mem (full screen)
16544	pmodetbl	Print mode definition table. (for £px)
16640	tempscn	Temporary storage for top of screen (full screen)
16880	tempcol	Temporary storage for top color mem (full screen)
17124	curdsp	Current display # (for top of screen)
17138	mask	Password mask character
17139	scnmode	Screen mode (Full or Split)
53248	local	flag for forced local mode. (won't output to modem)
53249	case	Upper case lock flag
53250	editor	Flags for the &,1 routine
53251	tsr	Time Remaining
53252	llen	Line Length for &,1
53254	chat	flag for returning from & for an abort
53256	chatpage	turns on the flashing page
53257	access	shadow for ac%
53258	mxor	Carrier XOR value
53259	mkolor	Kolorific mode flag
53260	mupcase	Uppercase mode flag
53261	irqcount	IRQ slowdown flag
53262	trans	Ascii Translation Flag
53263	index	length of line returned by inline

Using Modules:

In Image BBS, the "main" program, also known as the "im" file, resides at $301 in memory. Other modules, called "Plus Files" load at $0801. The area designated for "Plus" files is $0801-$4000, which is about 14k. (This is about 56 CBM (1541) "blocks") The main file starts at Line # 1000. Modules start at line 1. When you enter a module, it is ALWAYS entered with either a GOTO1, or a GOSUB1.

When a modules starts to become too large for the 14k buffer, you can further divide it into Sub-Modules. The most common way is to use "Mini-Modules". These are modules that can be up to 4k in length, or about 17 disk blocks. The Mini-Modules load at $3001 in memory. When one is loaded, it is effectively appended onto the end of the Plus file that is currently in memory. Therefore, you cannot start a Mini-Module at line 1. All of the line numbers in the Mini-Module MUST come AFTER the line #'s in the Plus File. If there is an overlap, it may crash the system, forcing you to re-boot. For this reason, be VERY careful when you program using modules.

There are a few restrictions that you must be aware of. First, the Mini-Module and the Plus file must share the same area of memory. So, even though the maximum size of a Plus file is 14k, if you are using a Mini-Module, the maximum size for the plus file becomes 10k, and the Mini-Module is 4k, which is a total of 14k. There are other sizes of modules, and here is a chart to explain them:

Size of Modules	Plus File Length	Module Length	Module Address
No Module	14 k/56 blks	----------	-----
Mini-Module	10 k/41 blks	4 k/17 blks	$3001
Small Module	6 k/25 blks	8 k/33 blks	$2001
Large Module	2 k/ 9 blks	12 k/49 blks	$1001

Using ML Modules:

If you wish to use ML modules, more commonly known as "++" files, the &,7,dv%,2 command will load them for you. You can call any of the normal Image & routines by loading the Accumulator with the & number, and X and Y with the parameters. Then a JSR $DD01 will call the routine. The area set aside for ML modules is from $C000-$CA7F

RS232 Routines:

The RS232 routines that Image uses reside in memory from $4300 to $45FF. There is a jump table at $4300 which has the ONLY safe entry points into these routines. NOTE: Calling some of these routines will result in "strange" things happening. Be VERY careful.

$4300 - Install

This is the install routine that is called when the system is booted, DO NOT call this routine, since it WILL crash your system.

$4303 - Enable

This routine enables the NMI interrupts that perform the RS232 I/O. Note that during such things as disk access these must be turned off, and then turned back on when the disk access is finished. Image handles this for you, and you should never need to call this routine.

$4306 - Disable

See above for "Enable".

$4309 - RSGet

This gets a character from the RS232 device. Note that RS232 has a 256 byte "ring" buffer so that characters that are received when the system is busy are not lost. The character is returned in the Accumulator.

$430C - RSout

This will output a character to the RS232 device. The character is passed in the Accumulator. Note that the routines use a technique called "double buffering" which means that while one character is being transmitted, another can be put into a "holding" buffer, until the first character is done. This allows the system to output a character, and then go back to doing more processing while it is being sent. This keeps the screen cursor synchronized with the cursor on the users' side, while still gaining the benefit of a buffered I/O system.

$430F - SetBaud

This will set the baud rate as defined by an internal table. See the listing for &,41.

The "Swapper":

Image uses a sort of virtual memory mechanism to allow the 12K of RAM that is "hidden" in the 64 to be used. RAM from $D000-$FFFF is used to store many of the routines that Image does not need to use as often as others. Examples are the disk input routine (&,2) and the entire editor system! These routines are accessed using the "swapper". This is a routine that swaps the needed routines into memory, and swaps whatever is in that memory to the hidden RAM. The routines are then executed, and then swapped back to hidden RAM. Thus, the 12K of hidden RAM can be used to hold more ML for the Image system. In Image 1.2, all but 2K of this RAM is used, and in future versions that last 2K will be used! The swapper is located at $CA80. To use it, you set the Accumulator and the Y register with the starting page numbers to swap, and the X register to how many pages of RAM to be swap. Thus to swap memory from $C000-$C2FF with the memory from $C300-$C5FF, you would do this:

```
LDA #$C0
LDY #$C3
LDX #$03
JSR $CA80
```

To swap it back again, repeat the same sequence of commands. Be very careful when using this routine, since it does not care WHAT it swaps, and will try to swap anything you tell it to!

The LightBar:

The LightBar is controlled by IRQ routines which read the function keys, and make the necessary screen/memory changes that are requested. There is an & which allows you to get or set the status of the LightBar. This is &,52.

There are several functions implemented in this command in version 1.2. You can clear/set/toggle/read any individual checkmark. Here are some examples:

```
&,52,0,0    Clear the check on the LEFT side of SYS.
&,52,0,1    Set the check on the LEFT side of SYS.
&,52,0,2    Toggle the check on the LEFT side of SYS.
&,52,0,3    Read the check on the LEFT side of SYS into A%.
```

The first parameter after the 52 is the position number, and the second is the function. Valid positions are:

Pos	L/R	Pos	L/R	Pos	L/R	Pos	L/R
Sys	0/ 1	Cht	8/ 9	Asc	16/17	Fn4	24/25
Acs	2/ 3	New	10/11	Ans	18/19	Fn3	26/27
Loc	4/ 5	Prt	12/13	Exp	20/21	Fn2	28/29
Tsr	6/ 7	U/D	14/15	Fn5	22/23	Fn1	30/31

The Editor:

The programming of the Image editor is best left to those who know a LOT about the Image system. There is an & which will make the necessary calls into the editor ML. This ML resides from $D000-$DFFF, and is swapped into memory from $1000-$1FFF by the swapper when it is called. There are 3 legal entry points, which are accessed with "&,54", "&,54,1", and "&,54,2". The first entry point is the "normal" entry. This puts the user into the editor with an empty buffer. The second entry does not clear the buffer. The last entry is the entry that you would use in an extended command if the command that was typed was not a recognized command.

Chapter IV: Disk File Formats

This chapter will give you the formats for all important files on your system. the maps will include the name of each file and the information stored in each record and the associated system variables.

1. u.config (user statistics) REL 254 bytes per record

Field	variable	description
1	na$	Handle
2	pw$	Password
3	ff$	First Name
4	ll$	Last Name
5	ph$	Phone Number
6	ld$	Last Call Date (new msg)
7	ac%	Access Level
8	ct%	Calls Today
9	tc%	Total Calls
10	co%	Comp type (co$(co%) to see text)
11	ll%	Line length
12	ul	Upper/lower case Flag (1 = on)*
13	ll	Line feed flag (1 = on)
14	em	Expert Mode Flag (1 =On)
15	dc	Total FILES downloaded
16	uc	Total FILES uploaded
17	bd	Total BLOCKS downloaded
18	bu	Total BLOCKS Uploaded
19	cr	Credit Points
20	ps	Total Posts
21	rp	Total Responses
22	d5$	TRUE Last call date
23	fl$	Users Flags
24	jn$	Joined/Unjoined

* Note that the ul flag is used for default transfer protocol if your system uses the TURBO Rel Subs.

2 e.data (system data) REL 40 bytes per record.

Rec #	Var	Description
1	ca	Total calls to the system
2	ag$	Access group name (ag$) for group 0. Preceded by 4 control codes which designate time per call, calls per day, d/l's per call, and max idle time.
3		Access group 1 (see rec# 2)
4		Access group 2 " "
5		Access group 3 " "
6		Access group 4 " "
7		Access group 5 " "
8		Access group 6 " "
9		Access group 7 " "
10		Access group 8 " "
11		Access group 9 " "
12	ur	highest user account
13	oc$	30 character string of 0's and 1's for SB.
14		Same as record 13 for UD.
15		Same as record 13 for UX.
16	uh	Number of active user accounts
17	d3$	Last caller on system
18	pp$	system remote password
19	d6$	Time of last user logoff (11 digit DYYMMDDHHMM)
20	p1%,p2%,p3%	Time allowed during prime time (p1%), PT start (p2%), PT end (p3%).
21	fl$(0)	Default flags for group 0.
22	1	" " 1.
23	2	" " 2.
24	3	" " 3.
25	4	" " 4.
26	5	" " 5.
27	6	" " 6.
28	7	" " 7.
29	8	" " 8.
30	9	" " 9.
31	l1	next available user account
32	mt$	modem command string

* Note that in records 2 - 10 the access group name is preceded by four (4) control characters which cannot be seen without the +.access editor. These characters are as follows.

1 Time Allowed on System (0=unlimited)

2 Calls Per Day (0=unlimited)

3 Idle Time

4 Downloads per Call (0=unlimited)

3. e.stats (system stats) REL 20 bytes per record

	LAST	LOG	CURRENT	TOTAL
FBACK	1	12	23	30
Sys MAIL	2	13	24	31
Usr MAIL	3	14	25	32
POSTS	4	15	26	33
RESPS	5	16	27	34
U/L's	6	17	28	35
D/L's	7	18		36
NEW Usr	8	19	29	
CALLS	9	20		
USED	10	21		
IDLE	11	22		

x = 37 Total minutes system used (overall).
x = 38 Total minutes system idle (overall).

4. bd.data SEQ

bd.data is a seq file which your system reads once during boot. If any changes are made to this file you must re-boot your system to make these changes active.

Position	Var	Description
1		System Disk Device
2		System Disk Drive
3		E-Mail Disk Device
4		E-Mail Disk Drive
5		Etcetera Disk Device
6		Etcetera Disk Drive
7		Directory Disk Device
8		Directory Disk Drive
10		Plus File Device
11		Plus File Drive
12		User File Device
13		User File Drive
14	cc$	System Identifier
15		new user credits
16		highest device number
17		highest drive number
18	bn$	BBS Name
19	po$	Main Level prompt.

5. U/D directory file format SEQ

Pos	Var	Description
1	rn	number of files in directory (60 MAX)
2	f%(x)	file size in CBM blocks
3	nn$(x)	user id number padded with 0's + Handle
4	dt$(x)	u/l date + last d/l date + fname,type
5	ed$(x)	description of file
6	c%(x)	times d'loaded
7	d%(x)	comp type 0-9 (0 = Any)

Positions 2-7 are repeated through once for each file in the directory. The number of times is determined by the first file entry (rn).

6. SB directory format SEQ/REL

Pos	Var	Description
1	rn	number of posts in directory (60 Max)
2		title of original message
3		id# (or net id)of poster + handle
4		date of message + date of last resp
5		total number of responses

Positions 2-5 are repeated through once for each file in the directory. The number of times is determined by the first file entry (rn).

7. e.Sub, e.U/D, e.U/X file formats REL

record #1 Total number of subs, libraries defined.

records #2-901

Field	Var	Description
1		password for PW board
2	ac%(x)	Access code
3	bn$(x)	Board Name
4		unused
5		device
6		drive
7		open/close flag

8. E-Mail Format

Pos	Description
1	Handle of Sending User
2	ID of sending user
3	Date sent
4	[title]
5	text of message
6	(at end of file)

Chapter V: Variable Usage

Within Image, there are certain variables which can be considered 'reserved'. This does not mean that you cannot use them, per say, but that they can only be used in conjunction with specific purposes. Some variables may be used anytime, but have a specific purpose. Some variables can be used with certain subsystems, but not with others. Some variables may be used anywhere, but change continually. This is explained in detail in the following paragraphs.

An example of a variable used for a specific purpose is na$. This variable is used to print the handle of the user online. Storing something for a module in this variable would cause an undesirable effect. Basically, these types of variables are used to control system statistics and are best left alone, only to be used to output information. These variables are sometimes used as interfaces between the basic and ml, an example being pl. Setting pl to 0 will cause all user input to be in the form of upper and lower case characters. When you set this in basic, it causes the ml input routines to recognize the upper/lower case characters.

The main variables that can be used sometimes are arrays. Depending on what subsystem you are in, the arrays may or may not be in use. If they are not in use, it is safe to use them. The only exceptions to this is st(x), dv%(x), and tt$(x). St(x) holds the bar stats, changing the values of this array should be reserved for updating the BAR stats 1. Dv%(x) is the system device designator. Altering this will change the device accessed by the system. Tt$(x) cannot be used in a module that calls the editor. All text stored in the editor is put into tt$(x).

Some variables are intended to continually change. These include variables that will print text to the screen and modem, as well as variables that form links between the basic and ml portions of the program. Examples are a$ and an$. When used in conjunctions with & or &,0, the value of a$ will be printed to the screen and modem. All response entered at a prompt will be stored in an$. These variables have a set purpose, but are intended to change.

Virtually any single character alphabetical string (A-Z) can be used as a temporary storage variable with the exception of x$ which holds the system drive/LU indentifiers, u$ which holds the commands stacked at a prompt and p$ which is a ml reserved variable which contains the text of the last active prompt. One of the other advantages to using the single character strings is the capability of outputing the current definition of the variable using the £$var MCI command. Also note that any definition of a$ should be temporary and avoided for all purposes except for output to the screen and modem.

The current reserved string variable listing for Image v1.2 is as follows:

a$ Reserved for output to screen and modem.
ag$ Access group name of user currently online. (also £vm)
ak$ Prints line across screen (" "+"@"*11%-2+r$, also £vj)
an$ Last user input. (also £v7)
bn$ Current name of your BBS. (also £v5)
c1$ Chat mode entry message.
c2$ Chat mode exit message.
c3$ Returning To The Editor Message.
cc$ 2 character system identifier. (also £vn)
cm$ Current location in AREA window.
d1$ Current time and date information in 11 digit format. (also £v0)
d2$ Time and date of last logoff, also Library name at entry. (also £v8)
d3$ Handle of last user on the system. (also £v9)
d4$ Current ml protocol in memory. (also £v1)
d5$ TRUE last call date of user online in 11 digit format. (also £vk)
d6$ Logoff time of last user.
dd$ System identifier + user id number.
dr$ Currently active drive/LU number +":".
ff$ REAL first name of user online.
fl$ 15 character string which determines the user onlines "flags".
i1$ Access level + Handle of the sysop.
i2$ Expert Flag + Phone Number + First Name+" "+Last Name of sysop.
i3$ Access group name of sysop
ld$ Last call date of user online in 11 digit format. (used for new message reads)
ll$ REAL last name of user online.
lt$ Logon time of user online.
na$ Handle of current user online. (also £v2)
nl$ chr$(.) null
nm$ Last network sort in 11 digit format.
p$ Current prompt text.

ph$ Phone number of current user online. (also £v4)
po$ Text for system Main level prompt.
pp$ System password (change with PC command)
pr$ Name of current +.file in memory.
pw$ Password of current online user.
qt$ chr$(34) quotation mark
r$ chr$(13) return
rn$ REAL name of user online. (ff$+" "+ll$, also £v3)
sy$ Current subsystem active.
ti$ C= TOD clock.
u$ Reserved for command stack.
x$ System drive/LU designators.

The current reserved integer variables for Image v1.2 are as follows:

ac% Access level (0-9) of user online.
ao% Access level of user at login.
co% Computer type of user online (1-9)
ct% Number of calls today by user online.
d1% Currently active device number.
d2% Currently active drive/LU number.
d3% Currently active drive/LU number.
da% Number of D/L's allowed per call (0 = Unlimited)
dc% Number of D/L's this call by user online.
dv% Active device number.
i% Instant mode flag.
kp% Last key pressed.
ll% line length (38-80) of user online line.
p1% Time allowed during prime time.
p2% Time that prime time begins.
p3% Time that prime time ends.
pt% Prime time flag (1 = active)
tc% Total calls to the system by user online.
tr% Time remaining on system.

The current reserved floating point variables for Image v1.2 are as follows:

bd Total number of CBM blocks D/L'ed by user online.
bu Total number of CBM blocks U/L'ed by user online.
ca Total number of calls since system start.
cn Total number of calls since last re-boot.
cr Total credit points of user online.
dc Total number of files d/l'ed by user online.
dr currently active SYSTEM device. (1-6)
el reserved for future expansion.
em Expert mode flag.

f1 System flag.
f2 System flag.
f3 System flag.
f4 System flag.
id ID number of user online.
l1 next available account.
l2 Flag for reserved system.
l3 Flag for reserved system.
lc Flag for active subsystem menu.
le Editor lines allowed for user online.
lf Linefeed flag (default proto for Turbo Rels)
lp Output control flag.
mf Reserved for ml use.
nl C/G mode flag.
nm Network Flag.
pl Input control flag
pm Prompt Mode Flag.
pr Active ml proto in memory.
ps Total posts by user online.
qb Modem Speed.
qe Reserved for ml use.
rc Abort flag
rp Total responses by user online.
rq Reserved for ml use.
sh Checks for space bar.
sr Reserved for ml use.
st Status
uc Total files U/L'ed by user online.
uh Number of active user accounts.
ul Upper/lower case flag.
ur highest user id number +1

The following arrays are dimensioned by the bbs most can be used for your own program's.

ac%(31) Can be used for ANY +.file outside of UD,UX,SB.
bf(x,y) Blocks free on system drives. Should NEVER be used.
co$(9) Text for computer types supported. Should NEVER be used.
dv%(x) Device numbers for system drives. Should NEVER be used.
fl$(9) Default flags for access groups 0-9. Should never be used.
so%(31) Can be used for any +.file outside of UD,UX,SB.
st(60) Status, Should NEVER be used.
tt$(254) can be used anywhere that DOES NOT use the editor.