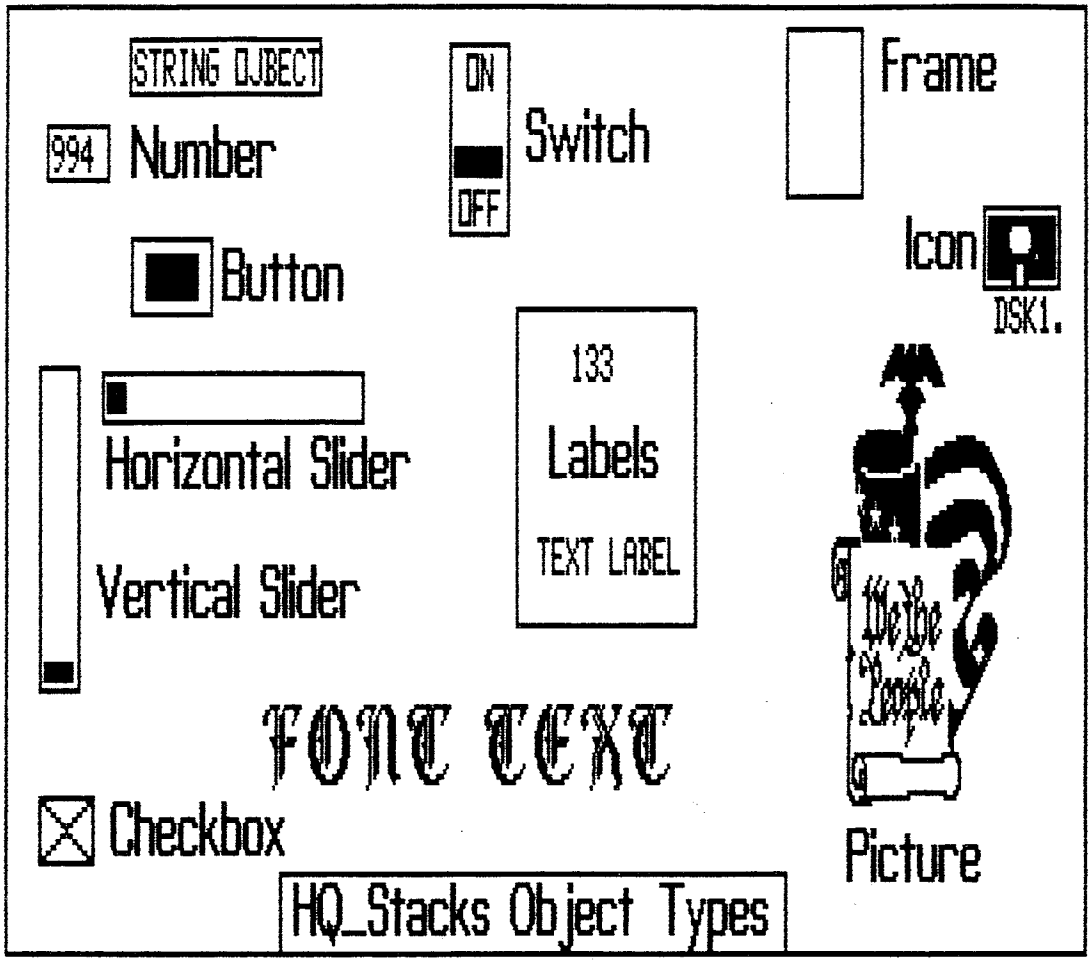
HQ_Stacks



Stackware Programming for the Geneve 9640.

McCann Software
4411 North 93rd Street
Omaha, NE 68134

Dear Fellow Geneve User,

Thank you for your continued support! This new version of HQ_Stacks increases by nearly forty the number of built-in messages in HQ. The new messages are, for the most part, in support of the ART stack which is included as part of the new version. Previously, ART was a menu option under the HQ browser menu. In order to demonstrate that stacks can be very powerful I moved all the art functions into messages which can be sent to built in handlers. This provides a large sample stack for study as well as a functional art program to produce art for use with HQ.

To run DEMO type &AUTOEXEC right after you boot up your Geneve. If you have any trouble there are some hints at the beginning of the manual. Use the right mouse button to "poke around." If you get to the HQ_Stacks Coloring Book you can turn on the crayon by clicking the right mouse button while the cursor is in the crayon box. Turn it off with the center mouse button (DEL). To draw hold down the right mouse button and drag the mouse. To change colors move the mouse to the last line on the screen and click with the left button (INS) right above the color you want.

I hope you enjoy this new version of HQ_Stacks and produce some beautiful and useful stackware. In fact please write some stacks and share them with other HQ_Stacks owners. However, don't break federal copyright law by distributing the files on the HQ distribution disk with your stackware. If someone sees your stackware in action and likes it please tell them my address and let me sell them a copy.

I am going to spend the next several years working on HyperQuit interpreters, of which HQ_Stacks is an example. I would like to continue to develop on the Geneve as a base and port to other machines. There are several hundred more built-in messages which I would like to develop and add to HQ_Stacks. The areas of file handling, sound, sprites, floating point numbers, user defined object types, inheritance, not to mention interaction with 9640 WINDOWS are still to be fully explored. This will take thousands of hours of work and to continue I have got to sell 25-30 copies per month. The only way I can sell this many programs in the Geneve community is by word-of-mouth support. Those who have already bought HQ are the only ones who can spread the word that there is still something going on with Geneve. Your continued support and encouragement of others to buy HQ is the only way I can produce more versions of the program. I guess you can look at your purchase of HQ as an investment, the more you encourage others to buy a copy the more version upgrades you will be able to participate in. Again, thank you!

Mike McCann

HQ_Stacks Version 2 Documents

New Art Messages

The new art messages include CRAYON, BLOCKS, BOXES, ALPHA, DALPHA, CFILL, ELLIPSES, LINES, MAGNIFY, MOVES, RAYS, SPLINES, TRIANGLES and ZAP. These correspond pretty well with the functions in the old Art menu section of version 1. The three mouse buttons function slightly differently. The buttons still correspond to keyboard keys with right as ENTER, center as DEL and left as INS. Generally, ENTER is used to select or perform an operation. Now, DEL or the center mouse button is used to leave the message function, where F9 was previously. INS or the left mouse button now toggles between "anchor and unanchor" in functions where before this was split between INS and DEL. INS is also used at the bottom line of the screen to select drawing colors.

We have also added some additional functions to the line based messages. Line based messages are: LINES, BOXES, ELLIPSES, RAYS, SPLINES and TRIANGLES. The additional functions are rotation and translation. When a "rubber band" figure is extended the numeric keypad keys (with the NUM LOCK depressed) are active. The 8, 2, 4, and 6 keys are used to translate the figure up, down, left and right respectively, The plus and minus keys are used to rotate the object counterclockwise and clockwise respectively.

Art and Font Load and Save Messages

In order to make an art program work you must be able to save and load art and fonts. In order to do this you must be able to obtain filenames from the user. The string object type will do this but you must then be able to easily use the string as a filename. The words ARTLD, HQLD, INSTLD, SAVEHQ, AFONTLD, TFONTLD and CFONTLD all will take an address of a string variable like OMESSAGE as a parameter and use that string as a filename. This filename is automatically appended to the volume name (i.e. DSK3.) stored in VOLUME to provide a full pathname to the file. PRINT also will take the printer device name from a string variable like OMESSAGE.

POP-UP Menu System

Also included are a set of words which implement a "pop-up" menu system for HQ. If you are shuttling quickly between cards or between objects and their functions, as in the ART stack, the necessity of redrawing menu choices becomes bothersome. The messages >MNU, MNU>, MNU>B and MNU@ support this pop-up menu system. One method of using this system is illustrated in the ART stack. These messages are flexible, however, and many uses may be found for them. The pop-up menu is stored in 64 pixel lines which reside at the very "bottom" of the graphics memory in VDP. The video memory, in mode 8 which HQ uses, can be thought of as consisting of 512 lines of pixels. We display the top 192 or 212. The lines between 212 and 255 are used for sprite control. The lines between 256 and 467 are used as a

hidden backup for the visible screen. This leaves us with the lines between 468 and 511 in which to save a pop-up menu while we are using the whole top screen for artwork or display.

The message >MNU (which I pronounce "push menu") essentially pushes the number of lines selected for the menu from the visible screen to the menu area. MNU> (which I pronounce "pop menu") pops the number of lines selected for the menu from the menu area to the visible screen. MNU>B (which I pronounce as "pop menu to bottom screen") pops the menu area to the bottom screen at the line corresponding to the menu line in the top screen. The message MNU@ is what I call an xy variable, one in which two values are stored. In this case the "first" value is the number of the line where the menu is displayed (i.e. line 0 in ART.) The "second" value is the number of lines in the menu (1 to 64.) In this case 1 functions as 0, as nothing is shown if 1 is used. You may set these as you choose. They are set to (0,1) whenever a stack is loaded.

The best way to understand how this pop up menu system works is to study the Art stack included with the distribution disk. This shows that the menu is saved and the screen restored before an action and the screen saved and menu restored after an action. The exception is CATALOG and handling that case is demonstrated in the Art stack.

Notes on the ART stack.

To load the ART stack merely enter its pathname (i.e. DSK6.ART) at the browser menu and press "L" for LOADSTACK. You may also enter this pathname as a command line parameter from MDOS. To choose a function from the menus merely move the cursor over the your choice and press enter. The only exception is CLEAR_SCREEN which you must click on with the center mouse button or DEL key. To see the screen without the menu move below the menu and press the left mouse button. Press any other key to turn it back on. (This was programmed in a stackscript.) When any menu, except the opening menu, is present move out of the menu and press the right mouse button to return to the previous menu. What's really nice about the art stack is that you can study it by printing it with Prntstk from the browser menu then you can modify it to work just like YOU want. I didn't have enough space on the distribution disk to implement all the functions but following the examples on the disk you sure can.

Documentation Notes:

In Appendix A the definition of OFILL should indicate that the parameters consist of a color and two xy pairs. The first pair is the pixel location of the upper left corner of the rectangular area to be colored. The second xy pair is the width in x and height in y of the block that will be colored.

Appendix A Addendum 5/91

The following messages are added to Appendix A with the release of Version 2 of HQ_Stacks.

>LLM (P --)

Hide the lower portion of the graphics screen. The scan lines between 192 and 212 may be shown and hidden as chosen by LLM> or >LLM respectively. See CLLM COMM to clear this screen area.

>MNU (P --)

Save the "pop up menu" area currently shown on the screen to the menu area on the hidden graphics screen. This may later be retrieved with MNU>. The xy variable MNU@ determines the location and height (1 to 64 scan lines) of this menu area. See MNU>, MNU@, MNU>B.

AFONTLD (P addr --)

Load the TI-Artist type font with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. AFONTLD was designed specifically for retrieving a filename stored in OMESSAGE and loading the corresponding font. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See TFONTLD, CFONTLD. A typical script which uses AFONTLD:

ON ENTER OBJECT NAME FONTFILENAME DROP OMESSAGE AFONTLD OFF

The "OBJECT NAME FONTFILENAME DROP" phrase searches the objects in the current card for one with the name FONTFILENAME. OMESSAGE is then placed as a parameter for AFONTLD to use.

ALPHA (P --)

This is an "art functions" message. ALPHA allows text to be entered on the screen. The text is typed in by the user at the current location of the cursor. The currently loaded font is used. The foreground color of the characters may be changed by clicking the left mouse button directly over the color selection bar on the bottom of the screen. To "anchor" the line of text press the left mouse button at the location you wish to begin typing. When anchored, pressing the right mouse button or enter key will move to the next line. The variables CHSP, BLSP, and LNSP control the character spacing, blank space width and vertical line spacing respectively. See DALPHA.

ARTLD (P addr --)

Load the TI-Artist type "_P" picture with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. ARTLD was designed specifically for retrieving a filename stored in

OMESSAGE and loading the corresponding picture. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See HQLD, INSTLD. A typical script which uses ARTLD:

ON ENTER OBJECT NAME ARTFILENAME DROP OMESSAGE ARTLD PICMOVE OFF

The "OBJECT NAME ARTFILENAME DROP" phrase searches the objects in the current card for one with the name ARTFILENAME. OMESSAGE is then placed as a parameter for ARTLD to use.

BLOCKS (P --)

BLOCKS is an "art functions message". BLOCKS paints a rectangular block of the selected size and color at the current cursor location. When the cursor first appears you must determine the size of the block. To do this, click the left mouse button then move the cursor to the right and down. This opens a "rubber band box". When the box is the size you want press the right mouse button. The block will appear in the current color "attached" to the cursor. Move this block around the screen and press the right mouse button wherever you wish to paint this block.

To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

BOXES (P --)

BOXES is an "art functions message". BOXES draws rectangles in the current color and linewidth. To "anchor" the corner of the box, press the left mouse button. Move the cursor to the right down to open a "rubber band box" which shows the size of the box to be drawn. Press ENTER or the right mouse button to draw the box. To release the "anchor" press the left mouse button again. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

CFILL (P --)

CFILL is an "art functions message". CFILL will "flood" fill an enclosed area with the current color. Move the cursor to the approximate center of the area you wish to fill and press the right mouse button. CFILL looks at the color under the cursor when the mouse button is clicked. It fills each pixel line on the screen from right to left until a pixel is found that is not the same as the one that was under the cursor when CFILL started. Press any key to stop the fill. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press DEL or the center mouse button to leave the function.

CFILL@ (P xyc--)

CFILL will "flood" fill an enclosed area centered by xy in the color "c".

CFONTLD (P addr --)

Load the CHARA type font with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. CFONTLD was designed specifically for retrieving a filename stored in OMESSAGE and loading the corresponding font. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See TFONTLD, AFONTLD. A typical script which uses CFONTLD:
ON ENTER OBJECT NAME FONTFILENAME DROP OMESSAGE CFONTLD OFF
The "OBJECT NAME FONTFILENAME DROP" phrase searches the objects in the current card for one with the name FONTFILENAME. OMESSAGE is then placed as a parameter for CFONTLD to use.

DALPHA (P --)

This is an "art functions" message. DALPHA allows text to be entered vertically on the screen. The text is typed in by the user at the current location of the cursor. The currently loaded font is used. The foreground color of the characters may be changed by clicking the left mouse button directly over the color selection bar on the bottom of the screen. To "anchor" the line of text press the left mouse button at the location you wish to begin typing. When anchored, pressing the right mouse button or enter key will move to the next line. The variables CHSP, BLSP, and LNSP control the character spacing, blank space width and vertical line spacing respectively. See ALPHA.

DOT (P xyc--)

DOT puts the color "c" at the pixel location "xy."

EDPAL (P --)

EDPAL is used to change the palette colors of the 9938A video processor. Each of the 16 colors in the video 8 mode which HQ uses is made up of 3 components, a red, blue and green component. EDPAL displays these component values, 7 being the lightest value and 0 being the darkest value. You may change any of these values and use the arrow keys to move among them. Press F9 to escape.

ELLIPSES (P --)

ELLIPSES is an "art functions message". ELLIPSES lets you draw elliptical figures on the screen in the current color and line width. Press the left mouse button to "anchor" the center of the ellipse. Move right and down from the anchor to change the shape of the "rubber band ellipse" Press enter or the right mouse button to draw the ellipse. To change the current color,

move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

HQLD (P addr --)

Load the HQ art type picture with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. HQLD was designed specifically for retrieving a filename stored in OMESSAGE and loading the corresponding picture. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See ARTLD, INSTLD. A typical script which uses HQLD:
ON ENTER OBJECT NAME ARTFILENAME DROP OMESSAGE HQLD PICMOVE OFF
The "OBJECT NAME ARTFILENAME DROP" phrase searches the objects in the current card for one with the name ARTFILENAME. OMESSAGE is then placed as a parameter for HQLD to use.

HQ> (P --)

HQ> will move from the graphics screen into the HQ Browser menus depending on whether the menu is LOCKed or UNLOCKed. This would be the same as using F9 to return to the Browser. To be sure you can exit to the Browser use the phrase UNLOCK HQ>. HQ> is used in conjunction with LOCK and UNLOCK so that if you LOCK a stack you can unlock it from a script and return to the Browser or you can control the exit with some messages proceeding HQ> in a script (i.e. 15 4 COLOR HQ> .)

INSTLD (P addr --)

Load the TI-Artist type instance with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. INSTLD was designed specifically for retrieving a filename stored in OMESSAGE and loading the corresponding picture. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See HQLD, ARTLD. A typical script which uses INSTLD:

ON ENTER OBJECT NAME ARTFILENAME DROP OMESSAGE INSTLD PICMOVE OFF

The "OBJECT NAME ARTFILENAME DROP" phrase searches the objects in the current card for one with the name ARTFILENAME. OMESSAGE is then placed as a parameter for INSTLD to use.

LINES (P --)

LINES is an "art functions message". LINES allows "connected" lines (klines) to be drawn in the current color and line width. To "anchor" the line click the left mouse button. This will also "unanchor" the line. Move the cursor away from the anchor point to see a "rubber band line" representing where the line

will be drawn. To draw the line press ENTER or the right mouse button. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

LLM> (P --)
Save the "pop up menu" area currently shown on the screen to the menu area on the hidden graphics screen. This may later be retrieved with MNU>. The xy variable MNU@ determines the location and height (1 to 64 scan lines) of this menu area. See MNU>, MNU@, MNU>B.

LOCK (P --)

LOCK will prevent the user from going to the browser menu from the graphics environment by pressing F9. This somewhat isolates the user from the stack editing environment. UNLOCK is provided as well. CAUTION: LOCK also may lock the unwary author temporarily out of his own stack. A stack may be edited by re-entering the HQ environment without command line parameters

MAGNIFY (P --)

MAGNIFY is an "art functions message." MAGNIFY "zooms" in on a 32x32 pixel area of the screen for precise art editing. When the cursor appears, it is dragging a 32x32 box. Move this box around the area you wish to edit and press the right mouse button or ENTER. An expanded version of this area appears on the screen. Clicking the right mouse button "paints" a small square, which represents a pixel, in the currently selected color. The colors may be selected from the palette at the bottom of the screen. To erase, choose transparent, at the extreme lower left of the screen. Press DEL or the center mouse button to move back to the screen. Press DEL again to exit the MAGNIFY function.

MNU> (P --)

Restore the "pop up menu" area previously saved with >MNU to the hidden graphics screen to the visible screen. The xy variable MNU@ determines the location and height (1 to 64 scan lines) of this menu area. See >MNU, MNU@, MNU>B.

MNU>B (P --)

Restore the "pop up menu" area previously saved with >MNU to the hidden graphics screen to the top of the hidden graphics screen. This only makes sense if you work it out! The xy variable MNU@ determines the location and height (1 to 64 scan lines) of this menu area. See >MNU, MNU@, MNU>.

MNU@ (P -- addr)

This "xy" variable contains the location and height, repectively, in pixel lines, of the pop up menu. The location may be from line 0 to line 191. The height may be from 1 to 64

(not zero--but 1 acts like zero since this is a "pass through" to a 9938A function that starts counting at 1.) Use XY@ to get the values from this variable and XY! to place values there. LOADSTACK sets this to (0,1) so to use the pop up menu in a stack it may be necessary to set these values in the Stackscript. See >MNU, MNU>, MNU>B.

MOVES (P --)

MOVES is an "art functions message." MOVES lets you cut and paste an area of the screen. To cut the area move the cursor to the upper left hand corner of the area to be cut and press the left mouse button or INS key. Move the cursor to the right and down opening a bounding box around the area to be cut. Press ENTER to cut the object out. Now, as you move the cursor the cut portion of the screen is "attached" to the cursor. Press ENTER once to paste this cutout in another place on the screen. The cutout will "disappear" after cycling off and on. Move the cursor away and the cutout will be drawn on the screen and still attached to your cursor. Press F8 to "XOR" the cutout on to the screen. A cutout that is "XORed" to the screen may be "picked-up" by moving the cutout on the cursor over the cutout on the screen and pressing F8 again.

PASTEX@ (P xy-)

PASTEX@ is identical to the PASTE@ function except that it "XORs" the cutout area onto the screen. This way the same cutout area or picture may be XORed off the screen. PASTEX@ puts the cutout area at "xy." This area must have been just previously cut out by CUT or a STAGE art function. See PASTE, PASTE@, CUT, STAGEA, STAGEH, STAGEI.

PICMOVE (P --)

PICMOVE is an "art functions message". PICMOVE will "attach" a just previously cutout area of the screen or picture to the cursor. This cutout may then be pasted onto the screen at one or more chosen locations. The area must have just previously been cut out by CUT or a STAGE art function. See PASTEX@, PASTE, PASTE@, CUT, STAGEA, STAGEH, STAGEI.

POINT (P xy-c)

Get the color "c" of the pixel at screen location "xy."

POSTUP (P --)

An automatic message that is now sent by the new version of LOADCARD. There may be some situations in which you wish an activity to occur after CARDUP, after LOADOBJECTS but before the user is given control of the screen. If you need such a service put a POSTUP handler in your cardscripts.

PRINT (P addr --)

Print the graphics screen to the device with the filename located in the string variable addr. This string must have a

leading count. The OMESSAGE variable is such a string. PRINT was designed specifically for retrieving a filename stored in OMESSAGE and printing the screen. See PRINTCARD in APPENDIX A. A typical script which uses PRINT:
ON ENTER OBJECT NAME PRINTDEVICE DROP OMESSAGE PRINT OFF
The "OBJECT NAME PRINTDEVICE DROP" phrase searches the objects in the current card for one with the name PRINTDEVICE. OMESSAGE is then placed as a parameter for PRINT to use.

RAYS (P

RAYS is an "art functions message". RAYS allows lines with a common center point to be drawn in the current color and line width. To "anchor" the line click the left mouse button. This will also "unanchor" the line. Move the cursor away from the anchor point to see a "rubber band line" representing where the line will be drawn. To draw the line press ENTER or the right mouse button. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

SAVEHQ (P addr --)

Save a portion of the graphics screen to the file with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. SAVEHQ was designed specifically for retrieving a filename stored in OMESSAGE and saving art from the screen. The user is given the opportunity to open a box around the area to be saved. This is done by clicking the left mouse button to "anchor" the bounding box then moving to the right and down to surround the area to be saved. When the bounding box is correctly positioned press ENTER or the right mouse button to save. Press F9 to escape. See SVPIC in APPENDIX A for a similar function which uses different parameters. A typical script which uses SAVEHQ:
ON ENTER OBJECT NAME HQARTFILENAME DROP OMESSAGE SAVEHQ OFF
The "OBJECT NAME HQARTFILENAME DROP" phrase searches the objects in the current card for one with the name HQARTFILENAME. OMESSAGE is then placed as a parameter for SAVEHQ to use.

SCOLOR (P c --)

SCOLOR changes the screen color to "c."

SPLINES (P --)

SPLINES is an "art functions message". SPLINES allows an elliptically curved line segment to be drawn in the current color and line width. To "anchor" the line click the left mouse button. This will also "unanchor" the line. Move the cursor away from the anchor point to see a "rubber band line" representing where the line will be drawn. To draw the line press ENTER or the right mouse button. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press

DEL or the center mouse button to leave the function.

TCOLOR (P c --)

TCOLOR changes the text color to "c." This is the menu text color not FCOLOR the foreground color for pictures, instances and font text.

TFONTLD (P addr --)

Load the TPA type font with the filename located in the string variable addr. This string must have a leading count. The OMESSAGE variable is such a string. TFONTLD was designed specifically for retrieving a filename stored in OMESSAGE and loading the corresponding font. The volume (i.e. DSK6.) will be automatically attached to the front of the filename. This volume name is taken from the string currently in the string variable VOLUME. See CFONTLD, AFONTLD. A typical script which uses TFONTLD:
ON ENTER OBJECT NAME FONTFILENAME DROP OMESSAGE TFONTLD OFF
The "OBJECT NAME FONTFILENAME DROP" phrase searches the objects in the current card for one with the name FONTFILENAME. OMESSAGE is then placed as a parameter for TFONTLD to use.

TRIANGLES (P --)

TRIANGLES is an "art functions message". TRIANGLES allows right triangles to be drawn in the current color and line width. To "anchor" the triangle click the left mouse button. This will also "unanchor" the triangle. Move the cursor away from the anchor point to see a "rubber band triangle" representing where the triangle will be drawn. To draw the triangle press ENTER or the right mouse button. To change the current color, move the cursor to the bottom of the screen directly above the color you wish to choose and press the left mouse button. Press F8 to "XOR" the function onto the screen. Press DEL or the center mouse button to leave the function.

UNLOCK (P --)

UNLOCK will allow the user to go from the graphics screen to the browser menus by pressing F9. This is only to reverse the use of LOCK.

ZAP (P --)

ZAP is an "art functions message." ZAP lets you clear a rectangular area of the screen. To clear the area move the cursor to the upper left hand corner of the area to be clear and press the left mouse button or INS key. Move the cursor to the right and down opening a bounding box around the area to be cleared. Press ENTER to clear the rectangular area. Press DEL or the center mouse button to leave the function.

Table of Contents

Introductory Notes.

Loading HQ_Stacks(HQ)

Loading the HQ Demonstration Program:

On your HQ_Stacks distribution disk is an AUTOEXEC file. This file will serve to load and execute the HQ_Stacks demo stack from your A> prompt. To run the demonstration put your copy of the HQ_Stacks distribution disk in the A: drive and type &AUTOEXEC. This AUTOEXEC file will create the RAM_DISK E:, copy the HQ_Stacks demo files and artwork to E:, load the HQ_Stacks main program and run the demonstration. The demo stack will run itself and is self explanatory. Should any error occur in loading the demo, check your AUTOEXEC file. The AUTOEXEC file on your MDOS disk should not contain the commands TIMODE, RAMDISK, GPL or other possibly conflicting program names or commands.

Loading HQ Stacks:

To load HQ and an HQ stack at the MDOS prompt type the path name to HQ (i.e. B:HQ <stackname>). Where <stackname> is the device and filename of the stack. The stackname may be any valid device and filename (i.e. DSK6.COLORING). The HQ program will load, find the stackname on the command line and run the stack. Any artwork to be used by the stack must be in the proper drive. (Note: a stack's author may furnish a batch file, AUTOEXEC file or !README file for the purpose of moving art files to the proper drive.)

Loading HQ Alone:

At an MDOS prompt type the path name to HQ (i.e. F:HQ). HQ will load and present its browser/scriptor menu interface. From the menu interface you can run or browse existing stacks, create new stacks or edit and create artwork.

HQ_Stacks and Disk Drives:

A stack may reside on any device: diskette drive, ram disk or hard drive. The speed of HQ_Stacks is largely dependent on the performance of the device on which the stack and requisite artwork resides. McCann Software recommends that the stack and artwork for the stack reside on internal or external ram disk at all times. This facilitates the high level of disk activity used by HQ.

Printer Information:

HQ is capable of printing graphics screens as well as printing the listing of the stack database. For graphics printing, HQ will print only to Epson-type graphics printers. For full-page black and white graphics printing McCann Software recommends its program The Printer's Apprentice for MDOS. (Tip: If you wish to print a graphics screen similar to a printing program, turn all objects to transparent background color and black foreground color). The graphics printer filename is PIO.CR or PIO.CR.LF whichever works for your printer. To print a stack listing the proper printer filename is PIO.

Copying the Program Diskette:

McCann Software believes that your software investment should be protected. For that reason we do not copy protect our software. You are encouraged to make and use a back-up copy. Do not make any other copies. This is a violation of Federal criminal law and violation of our copyrights under civil law. If your friends want a copy make them buy one, they are worth it! The Purchase of HQ_Stacks is NOT a license to distribute the program.

NOTE: Geneve and MDOS are trademarks of MYARC Inc. TI-Artist is copyright by Insecbot Inc. 9640 Windows Copyright Beery Miller.

HQ_Stacks Diskette Contents:

The files HQ, HR, HS, and HT are necessary to properly load and run HQ_Stacks from MDOS. DEMO files are demonstration stacks. Any file which ends in the two letters "_H" are files of HQ specific artwork. The "_H" artwork format is published in an appendix in this manual.
Stack Design Tips:

For the best stacks, make a good stack plan. The stack should proceed logically from its title or opening card onward. Each card should be sketched out both for its visual elements and operational elements before construction begins. Create the artwork first and save it. Create each object or field and make sure of its placement, action and color. Next, create, attach and test each script for its correct operation. Have a kid test your stack for user friendliness! If you wish McCann Software to distribute your stackware please send us a copy and any restrictions on its distribution. We HOPE to provide collections of stackware to HQ_Stacks owners on an occasional basis for a small distribution charge.

HQ_Stacks(HQ) User's Guide.

As an HQ user there are very few things you need to know. The author of an HQ Stack will concentrate on making your work with the stack as easy as possible.

Stacks--The Concept:

A "stack" in HQ_Stacks is like a stack of note cards. The screen of your computer is the face of the current card that has been selected. Each card has a number of objects which may be fields, gadgets or artwork. You interact with these objects using the mouse or keys. Each type of object interacts in a different way. For instance, typically on a card there are objects that say "NEXT CARD-->" or "<--PREV CARD." By clicking the right mouse button while the mouse pointer or cursor is over this object the next card or previous card is selected.

Object Types and Operations:

There are twelve basic object types in HQ_Stacks:

1. **String Type.** Click on this object with the right mouse button to enter information. Several function keys are used in typing in information. F1 deletes the character under the cursor. F2 inserts a space under the cursor. F7 for help. F9 to escape the field. Press ENTER to enter the information into the field. The left and right arrow keys as well as Alt-E and Alt-D move the cursor left and right respectively.

2. **Integer Type.** Click on this object with the right mouse button to enter integer numbers (counting numbers.) The function keys work in this field just like in the String Type.

3. **String Label Type.** This object appears as a word on the card. Although it can be used simply as a label, it may respond to a mouse click or keypress. "Next Card-->" is usually this type of object.

4. **Number Label Type.** This object type is an integer number on the card. Although it can be used simply as a label, it may respond to a mouse click or keypress.

5. **Square Push Button Type.** This object turns off or on in response to click by the right mouse button. This could also be thought of as an indicator "LED" turned on or off by the mouse.

6. **Check Box Type.** An "x" marks this object when its on. The right mouse button turns this button on/off.

7. **Horizontal Slider Type.** As you click the right mouse button on the right side of this object its slider moves right. If you click on the left it moves left.

8. **Vertical Slider Type.** As you click the right mouse button on the top of this object its slider moves up. If you click on the left side it moves left.

9. **Light Switch Type.** The right mouse button clicks this on and off as you would expect.

10. **Frame Type.** This is a rectangular area on the card which may be shaded, bounded by a outline box or even invisible. An invisible frame type object might be hidden under the clown's nose in a picture on the card and when you click the mouse on the nose it honks. This object type is versatile you never know what an author might do with this.

11. **Picture Type.** This object type allows the stack author to load a picture onto a card. The picture may respond to mouse or keys or may not.

12. **Font/IconType.** This object allows the stack author to load font text or icons on the card. The area inside the font text or icon may respond to mouse or key action depending on how the author defined the object.

With just these few objects an enormous variety of cards can be created. If you can think of an object type that should be included in the next version of HQ_Stacks the author would be happy to consider your suggestion.

User Keys and Mouse Button Definitions:

Right Mouse Button: This is same as the ENTER key.

Center Mouse Button: This is same as the Del key and F2.

Left Mouse Button: This is the same as the Ins key and F1.

Arrow Keys: Move the cursor on the card just as the mouse does.

F1 Key: This is the same as the right mouse button and Ins key.

F2 Key: This is the same as the center mouse button and Del key.

F5 Key: Toggles the screen between 192 and 212 lines.

F7 Key: HQ_Stacks help Key.

F9 Key: Escape to HQ Browser/Scriptor level.

F10 Key: Change the cursor color.

All other keys may or may not function depending on what the Stack author had in mind.

HQ's Crayon Function:

A stackware author can attach the crayon function to any object. When you click the right mouse button on this object a palette of colors appears and your mouse will now spread color like a crayon. To color, hold the right mouse button down as you move the mouse. To change colors move the cursor to the bottom of the screen directly above the color you want. Click the left mouse button once and now you can draw with the chosen color.

HQ_Stacks and MDOS.

HQ can be directed to make the MDOS command line available to the user. You may also see MDOS functions (like disk catalogs) be used in a stack. Whenever an MDOS call happens HQ jumps to a text window and does the MDOS operation. This may involve some input on your part such as setting the date and time, etc. Once the MDOS operation is finished you will see the cursor blinking in the upper left hand corner of the card. Press any key and you will instantly jump right back to where you were in your stack.

HQ_Stacks and string or number functions:

If you click on an object and a blinking cursor appears the stack is asking you to enter some data. This may be a string or a number. Type in your number or string and press ENTER to let the stack accept your entry. Press F9 to escape the field. The F1 key will delete the character under the cursor. The F2 key will insert a space under the cursor. The left and right arrow keys will move the cursor left and right.

HQ_Stacks and Errors

From time to time you may have a file input/output error occur. If so, you will see the "I/O ERROR PRESS ENTER" message at the bottom of your card. Press ENTER and a menu will appear. You can generally recover from the error by pressing "L" for Loadstack from this menu. If any other thing needs to be done it should be noted by your stackware author.

HQ_Stacks Reference Manual:

HQ is a graphical user interface(gui) program. It is designed to be used at three fundamental levels. The first level is the User Level. The author intended that stacks be developed that could be used and enjoyed by preschoolers using the mouse only. With this type of stack an adult would load the program then remove the keyboard leaving junior with the mouse alone. With a properly constructed stack junior could never access any but the "user" level of HQ. At the same time, this program could be very rich in content accessing most of Geneve's features, color, sound etc.

The second level of HQ use is the Browser Level. HQ_Stacks is designed with a graphical user interface(gui) at the center and supporting structures surrounding this interface like a scaffolding. The Browser lets users access the statistics and scripts which perform the actions of the Cards and Objects which operate in the gui. The Browser Level user moves from the gui into the scaffolding and begins to modify the objects and artwork that comprise an existing stack to suit his needs. HQ_Stacks was designed so that those authoring stacks could provide suggestions for user modifications to their stacks so the Browser level user could easily implement advanced modifications to those stacks.

The third level of HQ use is the Scripting or Authoring Level. HQ provides an opportunity for both advanced and beginning scriptors and authors to create useful and interesting stacks fast and easy. The graphics environment and scripting language of HQ is rich and allows high level access to MDOS command line interface as well as MDOS XOP calls. The scripting language is full of graphics primitives that aren't so primitive as to make scripting a chore. We even provide the advanced scriptor high level access to 9640 Windows (Copyright 1990, Beery Miller) commands and are working with Mr. Miller to implement full Windows compatibility. The Browser makes the scriptor's life easier by allowing point and click modification

of both location of and function of built-in objects. There are 12 built-in objects which are easy to implement and very flexible. We even provide high level access to 9938A VPD features not accessible through MDOS XOPs. At the beginning scriptor level we give examples of how to write a slide show program that is useful immediately.

Our exposition in this manual will follow the three levels. The User level section of the manual provides the user with the things he needs to know to successfully operate the gui. At the Browser level, we will lay out the entire supporting scaffolding which the Browser level user can operate within-including HQ_Art. The manual will also give examples of stacks that can be easily modified to make the Browser a power user in short order. The remainder of the manual is mainly for the Author or Scriptor and is filled with Tables and Appendices which are for reference. These references lay out HQ_Stacks structure, scripting language, HQart definition, and other technical details.

HQ_Stacks Browsing Menus

Main Menu

Dir Edit Loadstack Cardload Browse HQ Newstack Mdoscli Art Quit Stackname <stackname>

Each Menu in HQ_Stacks consists of a list of words with the first letter of each word capitalized. Each word represents a choice among functions or programs. When the cursor is flashing to the left of the first line of the menu pressing the key corresponding to the first letter of the menu choice will move you into that function.

Stackname <stackname>: The function "Stackname," chosen by pressing "S" will allow you to enter the name of the stack you wish to execute, edit or create. By pressing "S" the cursor moves into the field "<stackname>." You may enter the device name and stackname here (i.e. DSK1.FIRSTSTACK.) Press ENTER to accept your choice. Press F9 to escape this field. You may also use the "Activate" option of the Directory function to move a stackname to this position.

Edit, Loadstack, Cardload, Browse, HQ, and Newstack functions will return an error if there is no valid <stackname> in this field. The Dir, Mdoscli, Art, Stackname and Quit functions will operate properly without a valid <stackname> present.

Dir (Directory): Pressing "D" will activate the Directory function. A menu window will pop up in the middle of the screen similar to the following:

DSKn.
Please Enter Device
USED nnnn FREE nnnn
<filename>
<filename>
<filename>
<filename>
<filename>
<filename>
<filename>
Alt-E,Alt-X,F9,Act

The cursor will be placed in the device name field. Type in the proper device and number (i.e. DSK2.) then press ENTER. The Dir function will read the catalog of the device or subdirectory and display the filenames on that directory in the "<filename>" fields. The cursor will appear next to one of the <filename> fields. Pressing the up and down arrows will allow movement through all of the filenames in the directory. Pressing

F9 will escape the Dir function. Pressing "A" next to one of the <filename> fields will move that filename and the device name into the Stackname field.

Browse: Press "B" to move into the browse mode. The browse mode is very powerful in making changes and moving through the variables and scripts of a stack. When you press "B" the graphics screen of the current card is redrawn and the cursor appears. The left mouse button (Ins) and right mouse button (Enter) are active.

The left mouse button, if pressed while in the bounding box of an object will attach to that object and let you drag it to a new location. When you have moved the object where you want it, press the right mouse button. The object's new location will be written to its variables section. After you release the object you are returned to the HQ main menu.

The right mouse button, if pressed while in the bounding box of an object, will move you into the object editor. The object editor will be at the location in the stack of the object you clicked on ready to edit variables or scripts. After you edit variables and/or scripts and press F9 to exit the object editor you will be returned to the main screen to edit any other object. Note: Any change you might have made to an object while you were in the editor will not be effective since the card must be reloaded to correctly reflect changes made to an object.

Loadstack: Press "L" to load the current stack. The <stackname> field at the main menu must have a valid device and filename or an error will occur. Any pictures, fonts or icon files necessary for proper operation of the stack must be in the proper drive. Additionally, if pictures or fonts are to be used, the fonts must be loaded by the stackscript or cardscript. The stackscript or cardscript must use the SOURCE command to set the proper VOLUME name for the device used to load pictures for picture type objects.

When you press "L" the graphics screen appears, all objects on the current card are loaded as well as any other actions specified in the stack or card scripts. When the cursor appears the user may use the mouse or keyboard keys to begin to interact with the objects on the current card and stack. Press F9 to escape to the HQ main menu.

Cardload: Press "C" to load the objects and run the cardscripts of the current card. A stack must be present for the cardload command to function. When you press "C" the graphics screen appears, all objects on the current card are loaded and any actions specified in the cardscripts are performed. When the cursor appears, the user may use the mouse or keyboard keys to begin to interact with the objects on the current card and stack. Press F9 to escape to the main HQ menu.

Newstack: Press "N" to create a new stack. The stack created uses the stackname specified in the Stackname <stackname> field on the line below the main HQ menu. When you press "N" HQ creates the file and writes the necessary tables, headers and indexes to the file. HQ also creates one card, object and objectscript which are null type. After the stack is created you may use the edit menu to begin building stackscripts, cards or objects. After you have created one non-null object you may use the browser to edit objects.

HQ: Press "H" to move directly to the HQ graphics screen. This option does not load the stack or the current card or activate the browser. It will let you look at the graphics screen. If a card was previously loaded and no editing changes made, the objects will correctly respond to mouse and keyboard input.

Mdoscli: Press "M" to jump to a "command line interpreter" (cli) which sends MDOS commands to MDOS. In most respects you can interact with this cli just like the MDOS cli except that the key responses are a little different. The HQ cli will only interpret the command up to the current cursor position regardless of the number of characters in the line which is slightly different than MDOS.

Quit: To quit HQ, press "Q." When you press "Q" you will be offered the "Exit? Y/N?" option. Press "Y" to exit. Any other key will return you to the HQ menu.

Edit

Edit & Debug

Stackscript Card Object Memdmp Pstack Reset Var

Press F9 to leave the Edit and Debug menu.

The edit function allows you to access the three major divisions of the stack. At the top level are the stackscripts used to set up the environment that the stack is to run in. Next, are the cards, each card containing any number of scripts and objects. Finally, the objects, which are displayed on the screen and interact with the user. The edit menu allows you to choose which part of the stack you wish to edit by pressing the key corresponding to the first letter of the edit function.

The debug function lets you examine some of the inner workings of HQ in case of your stack is generating unexplained errors.

Script Menu

Edit View Save Restore Copy Zap 0 1 2 3 F4(Dn) F6(Up) F9(Back) Card# nn Obj# nn Scr# nn

There are scripts at each level of a stack. There are stackscripts, cardscripts and objectscripts. The editing of each type of script is controlled by the same menu. Each script consists of a block of 256 characters displayed as four lines of sixteen characters. You may type in any characters you wish into these scripts. Each object may have 124 scripts. Each card may have 124 scripts. Each stack may have 124 scripts. Since this version of HQ_Stacks allows 124 cards per stack and 124 objects per card...you do the math, that's a lot of scripts. Although scripts may hold any character data you wish there is a special use for scripts. If you enter a non-zero value in the OATTACH field of an object whenever that object is sent a message (like a mouse key press) all its scripts are searched for executable words from the scripting language.

Edit:

To edit an existing script press "E". Your cursor will appear in the script box at the top of the screen. The editing keys are as follows:

F1-delete character under the cursor.
F-insert blank under the cursor.
F3-delete the current line and move it to the 1-line buffer. F7-Help shows editing key functions.
F8-Insert a line from the 1-line buffer. F9-Escape.
F10-Blank to end of line. Remainder of line to 1-line buffer. Control8-Insert a blank line.
Tab Key-Tab 10 spaces.
Arrow keys-Up,Down,Left,Right.
Alt-E,X,S,D, same as arrow keys.

Note: Inserting a line at the bottom line takes an F8-F3-F8 cycle.

View

Changeto F4(Dn) F6(Up) F9(Back)
Card# nn Obj# nn Scr# nn

The view function of the script editor allows you to look at or copy from another script. Like the top window on the script editing screen, the middle window is the view window. The F4 and F6 keys will let you view preceding and succeeding scripts in the current object or card. Select the "Changeto" function to choose another card, object and script to view. The card, object and script numbers are shown in the script editing and view menus. F9 takes you back to the script editing menu.

Save

Pressing "S" will save any changes to your script. This is done automatically when you leave the editing menu.

Restore

Pressing "R" is an attempt to restore the current script to the way it was before any changes. If you have pressed save, left the editing menu or used any other menu function this may or may not be successful.

Copy

Pressing "C" will copy the script in the lower view window into the upper edit window.

Zap

Zap will remove this script from the stack. Zap is not reversible so the message Delete? is shown in case you want to change your mind.

0 1 2 3

Pressing 0, 1, 2, or 3 will copy that line from the view window into the 1-line buffer for later insertion (F8) into the edit window script.

F4(Dn)

Pressing the F4 key will move to the previous script, if any. This will be displayed, ready for editing, in the window at top of the screen.

F6(Up)

Pressing the F6 key will move to the next script, if any. This script will be displayed, ready for editing, in the window at top of the screen.

F9(Back)

Pressing the F9 key will take you back to the previous menu depending on which menu you entered the script editor from.

Card and Object Editor

Cards and objects both consist of a set of parameters (variables) and up to 124 scripts. Thus, cards and objects are edited through essentially the same menu. The only difference is the card menu shows the card name of the current card while the object menu shows the name of the current object. In the following explanation the term card/object means the information holds for either. When the card/object editor menu

is present, pressing the key corresponding to the first letter of the menu function word will activate that function. The menu is as follows:

New Scripts Variables Print Zap F9(Back) F6(Next) F4(Prev)
Card# nn Obj# nn Scr# nn Objname cccc Rec# nn

New

Pressing "N" will create a new card/object or a new script belonging to the current card/object. The message "New Object or Script?" / "New Card or Script?" appears. Press the key corresponding to your choice of card/object or script. Press F9 to escape without choosing. When you make your choice new entries are automatically added to the stack file and then your are presented with the parameters (variables) editing screen in order to define your new card/object.

Scripts

Press "S" to move to the script editor and edit the current script. If no scripts are present you will be asked "New Script? Y/N?. If you answer "Y" a new script will be added to the stack and you will be moved to the script editor. If you press any key other than "Y" you will escape from this choice.

Variables

Pressing "V" will move you to the variable editing screen for the current card/object. HQ variables are the factors which uniquely define each object or card itself. The variable editing screen provides a way to define, modify and observe these factors. The up and down arrow keys move you through each variable field. In the NAME and MSSG field you may enter characters, the rest are positive and negative integer values. The F1 key will delete the current character. The F2 key will insert a blank under the current character. The F7 key will display a help window keyed with help for this particular variable. F9 will escape the variables editing window. F9 will also escape changes to a field so make sure to use the arrow keys or ENTER to leave a field you want changed before using F9 to exit the variables window.

Variables

NAME	ODX	OBG
MSSG	ODY	OLFG
OBJX	OVAL	OLBG
OBJY	OMIN	OTYPE
OBJX1	OMAX	OSTEP
OBJY1	OFG	OATTACH

Each object type has special use for each of the object variables so see the particular object type definition for specifics.

Locate? Y/N?

On leaving the variables editing window you are given the Locate? Y/N? option. This allows you to define the size of and place the object you have defined on the screen. Press "Y" to move to the graphics screen and size and place your object. NOTE: on some objects if you do not size and place them they do not appear on the graphics screen. Also remember that if you intend to attach scripts to your object to put a 1 in the OATTACH field.

Print

Pressing "P" will present the Printfile <printername> option with the cursor positioned over the <printername> field. Type in a valid printer device name (i.e. PIO.) and press ENTER. The current card/object will be printed along with all its scripts. When printing a card the print function will print the card variables, the cardscripts, each object and all objectscripts. Press F9 to escape the <printername> field without printing. Press any key to terminate printing. If an i/o error occurs press ENTER to return to the main HQ menu.

Zap

Press "Zap" to delete this card/object and all its scripts. This is not reversible so the "Delete? Y/N" message is offered. Press "Y" to delete or any other key to escape the zap. NOTE: if you delete a card, all its cardscripts, objects and objectscripts are deleted as well. The filespace is reserved for reuse but not recovered.

F4(Prev)

Press F4 to move to the previous card/object.

F6(Next)

Press F6 to move to the next card/object.

F9(Back)

Press F9 to move out of the card/object editing menu.

The "Card# Obj# Scr# Cname Rec#" line shows where you are in the stack.

Memdmp

The Memdmp option will display thirty two-consecutive bytes of memory to assist you in debugging your stack. As with traditional memory dumps the display is in hexadecimal numbers and ascii characters. To select the beginning address of the memory dump type in the desired hexadecimal memory address and press Enter. The thirty-two memory locations will be displayed. To move to the next thirty-two, locations, press the up arrow. To move to the previous thirty-two locations press the down arrow. Although the address may be entered as either signed or unsigned hex numbers after the display the signed hex number is returned as reference.

Pstack

Pressing "P" will display the contents of the parameter stack. The "normal" display will show a "1" and no other value. Generally, if there is anything else on the stack it was probably left on the stack by a script that did not eat all its parameters. If there is nothing on the stack the script ate too much. If you are getting errors this may be the reason. To reset the stack press "R" for Reset at the Edit & Debug menu.

Reset

Press "R" to reset HQ. This is a non-destructive reset that will clear the stack, reset the internal pointers and generally recover attempt to recover from any difficulties that may have arisen.

Vars

Vars is a utility which lets you look at the contents of any variables you may be using. At the flashing cursor type in the message name of the variable. If it is a valid variable name, the hex address of the variable and its contents in hex and decimal will be displayed.

Art

The art option in HQ_Stacks gives you the opportunity to create artwork for use in this program. It is not a major artwork program with the vast array of features some have, but it will perform the basic functions. The art option will also import the popular TI-Artist "_P" files and "_I" instances for inclusion in your artwork. McCann Software intends to write separate artwork conversion programs which convert from other artwork formats to the HQ format whenever possible. We have published in this manual the HQ artwork format which is a simple 16 color, byte oriented RLE format. This will make it easy for you to write your own conversion utilities. There are dozens of color artwork formats and including even a sample of them overwhelms the limited space we have for useful programming. If you publish an art program or have COMPLETE details on a format and would like to cooperate on a conversion program, please send me the details of the format plus two representative files. I will try to write a conversion program that runs under MDOS and share the code with you.

Art Menu

Artwork Fonts Drawing Print Clear Textcolor

By pressing the key corresponding to the first letter of the word that stands for a menu option you move to that option. Pressing F9 at this menu returns to the HQ main menu.

Artwork Menu

Directory Loadart Saveart Fcolor Bcolor

T I HQart <filename>

The artwork menu allows loading and saving artwork to the screen. There are three formats available. The HQ format is both read and written by this program. The TI-Artist pictures and instances are read only. To load or save HQart the word HQart must be visible with a valid <filename> when Loadart or Saveart menu options are chosen. Pressing "H" will move your cursor to the <filename> field where you may enter a valid filename (i.e. DSK1.PICTURE_H). Note: HQ_Stacks uses the "_H" filename ending convention for HQart. HQart is stored in I/V254 format. Press ENTER to accept the filename you have typed in. Press F9 to escape the field and cancel any changes made in the <filename> field. Pressing "T" will bring up TIArtist <filename> for loading the "_P" portions of TI-Artist pictures. Pressing "I" will bring up Instance <filename> for loading TI-Artist instances.

Directory

The directory function operates here as in the HQ main menu. If you need information on the directory function please refer to that section. Pressing "A" while the cursor is adjacent to a filename in the directory window will move that filename to the <filename> field in the artwork menu.

Loadart

Pressing "L" will load the art from the <filename> art file. TI-Artist pictures and instances load with foreground color and background color based on choices made with the FCOLOR and BCOLOR options.

Make these choices before loading your artwork. After you press "L" and the art file loads and the cursor appears in the upper left corner of the screen. The artwork is "attached" to this cursor and may be moved around the screen.

To place the artwork press ENTER (right mouse button) or F8. If you push the art off the right edge or bottom and press ENTER, only the portion visible will be loaded. When you press ENTER or F8 the picture "cycles" and then disappears. When you move the cursor again the picture will still be attached but a copy will be left where you pressed ENTER. Place as many copies of the artwork on the screen as you like. If you use the ENTER key the picture will cover anything under it. If you use F8 the picture will perform a logical "XOR" with anything under it. This may be a useful technique for you. F8 may also be used to "pick-up" a badly placed piece of art placed with F8 (this must be done while the art is still attached to the cursor) just realign the art on the screen with that on the cursor and press F8. If you load a large object, moving it may be sluggish as it must be redrawn after every move. To make moving objects easier after your art is "hooked" on the cursor, press Del (center mouse button). The picture disappears and you can move the cursor with ease to your intended location. When located, press Ins (left cursor button) to "turn on" the picture again. Press F9 to escape the load screen. Finally, should any error occur in loading press ENTER to return to the HQ main menu.

Saveart

Pressing "S" will save the art from the screen to the <filename> HQart file. When you press "S" the cursor appears on the screen. This will let you "clip" the rectangular portion of the screen you want saved to the file. To clip the portion to be saved place the cursor at the upper left hand corner of that portion and press Ins (left mouse button). Move the cursor to the right and down expanding a box around the area to be clipped and saved. When you have this clipping area correct to your satisfaction press ENTER (right mouse button) to save the art. If you wish to reposition the upper left corner of the clipping box, move the cursor to the new position and press Ins (left mouse button) again. To cancel the save press F9. (Note: If you cannot see the clipping rectangle after you click the left mouse button and move down and right the drawing color has been set either at transparent or screen color.) To remedy this move to the Draw function and choose another color. Should any error occur in saving your artwork press ENTER to return to the HQ main menu. Choose "A" for Art and try again.

Fcolor

Press "F" to choose the foreground color of TI-Artist pictures and instances. Do this prior to loading or the foreground color will be the one last chosen in another operation. When you press "F" a color palette will appear along with the cursor. Move the cursor directly over the color you wish to choose and as far down the screen as the cursor will travel. Press ENTER (right mouse button). The foreground color is now selected.

Bcolor

Press "B" to choose the background color of TI-Artist pictures and instances. Do this prior to loading or the background color will be the one last chosen in another operation. When you press "B" a color palette will appear along with the cursor. Move the cursor directly over the color you wish to choose and as far down the screen as the cursor will travel. Press ENTER (right mouse button). The background color is now selected. Note: On the far left of the palette is a color block that is the same as the screen background color. This is the transparent background. It is not the same as the screen color but lets the screen color show through.

Fonts

HQ allows you to load a font for use in the art program. This same font is carried over into HQ (they occupy the same buffer space). This is not the font used in the menus although that font is the default and

Drawing

The drawing functions of HQ provide the basics needed to create useful art.

Drawing Menu

A/Dalpha Fatpix Crayon Objmov Palette Zapobj
Ray Kline Magnify Box Triangle Spline Ellipse

A/Dalpha

Press "A" to type characters on the screen from the currently loaded font. Press "D" to type characters "down" the screen. The spacing, foreground color and background color of the font characters depends on the settings in the variables table of the font menu.

Fatpix

Press "F" to set the pixel width in the pixel table. This consists of two values "Hl nn Wide nn" Enter a value in each. Press ENTER to move through the table. Press F9 to escape. This sets the width in pixels of the lines which are drawn in the Ray, Kline, Crayon, Box, Triangle, Spline and Ellipse options. The larger the value in the Fatpix table the fatter the lines are.

Crayon

Press "C" to spread color with the mouse like a color crayon. When you press "C" a color palette and cursor will appear on the screen. The color of the crayon is chosen by moving the cursor down the screen as far as it will go. Then move it directly over the color in the palette you want and press the left mouse button or Ins key. To draw, place the cursor where you want to start drawing, hold down the right mouse button and move the mouse. Lift your finger off the button to stop drawing. Holding down the F8 key while moving the mouse gives a slightly different effect. Use the transparent color at the far left edge of the palette to erase. If your cursor is hard to see press the F10 key to cycle through the cursor colors.

Objmov

Press "O" to move or copy one rectangular area of the screen to another. When you press "O" the cursor will appear on the screen. Move the cursor to the upper left corner of the object you wish to move and press Ins or the left mouse button. Move the cursor to the right and down from this point opening a "clipping rectangle" around the area to be moved. When you have opened the box around the area to be moved press Enter or the right mouse button. The cursor will move to the upper left and the object you want to move will be "attached" to the cursor. Move the cursor to the place you want the object to be and press Enter (right mouse button) or F8. The object will "cycle" and disappear (if you have trouble try pressing Enter). Now move away with the cursor. You will still have the object attached and a copy will remain where you dropped it. Repeat as many times as desired. Press F9 to escape and release the object.

Using F8 instead of enter gives a different effect called XOR. To "pick-up" the object again, move a copy right over the top of the object on the screen until they are aligned. Press F8 to "pick-up" the copy off the screen. This only works if you did not use F8 and "mix" your object with another below it on the screen. For a really unusual technique, pick up a small object using Objmov. Hold down the right mouse button until the object flashes and then move the mouse around. This can be done with F8 for another effect.

Palette

Press "P" to see or change the color palette. You can mix your own colors with the palette option. The colors you see on the screen are formed by mixing a red, green and blue component. Each of the sixteen colors in the graphics mode used in HQ allows eight levels of red, blue and green for each color. When you press "P" you will see the sixteen colors (the one on the far left is transparent) and the three values that make them up. The cursor can be moved by the arrow keys to any of these colors and by pressing the number keys 0-7 you will change the colors. Caution, if you change the text and screen colors so they are the same, you can't see the text for the screen. Press F9 to go back to the draw menu. The palette will remain in your selected colors until it is turned off or colors are reset. The HQ script message PALETTE also sets the palette.

Zapobj

Press "Z" if you want to zap or erase a part of the screen. When you press "Z" the cursor appears on the screen. Move the cursor to the upper left corner of the area you want to erase. Press the left mouse button (Ins) and move to the right and down opening a clipping rectangle around the area you wish to erase. Press the right mouse button (Enter). The area will be erased. Erase is not reversible. To escape the erase function press F9 before you press Enter. To reposition the upper left corner of the rectangle before erasing move the cursor to the new position and press the left mouse button and repeat the above process.

Ray

Press "R" to draw one or more rays. A ray is a line or series of lines emanating from a single point. When you press "R" the cursor appears and you may move it to any location on the screen. Press the left mouse button (Ins) to "anchor" the point where the ray(s) will be drawn from. Move the cursor away in any direction and a "rubber band line" will follow. When this rubber band line matches the line you wish to draw press the right mouse button (Enter). If you move the mouse away you may draw more lines originating from the same point. You may also move the initial point by pressing Ins (left mouse button) again. The line color is dependent on the last color chosen from the color palette at the bottom of the screen. To choose a color move the cursor to the bottom of the screen as far as the it will go. Move directly over the color of your choice and press the left mouse button (Ins).

Kline

Press "K" to draw "Konnected" lines. Each time a kline is drawn the initial point is moved to the end point and so each new line is connected to the last. The other information is the same as in Ray. If you hold the right mouse button down and move the mouse the technique is similar to that used in Crayon.

Magnify

Press "M" to use the magnifier. This "zooms" in on a 32x32 pixel area of the screen for really precise work. When "M" is pressed, the cursor appears dragging a 32x32 box with it (if the box is not visible press Ins or choose a more visible color from the palette). When you then press enter the expanded version of this area is drawn along with its small representation to the upper right. Clicking the right mouse button "paints" a small square, which represents a pixel, in the currently selected color. The colors may be selected from the palette at the bottom of the screen. To erase, choose transparent at the lower extreme left. Press F9 to return to the regular screen. You may now repeat this process for another 32x32 area or press F9 to escape back to the menu.

Box

Pressing "B" for box moves to the rectangular box drawing function. When you press "B" the cursor appears on the screen. Move the cursor to the upper left corner of the area you want the box drawn around. Press

the left mouse button (Ins) and move away. You will see a "rubber band box" expand. When this meets your specifications press the right mouse button (Enter). Repeat this process for as many boxes as you need. Colors for the box may be selected from the palette at the bottom of the screen. Move the cursor down the screen as far as it will go. Move the cursor directly over the your choice of color. Press Ins (left mouse button) to select the color. Press F9 to return to the draw menu.

Triangle

Pressing "T" for triangle moves to the triangle drawing function. When you press "T" the cursor appears on the screen. Move the cursor to the upper left corner of the area you want the triangle drawn around. Press the left mouse button (Ins) and move away. You will see a "rubber band triangle" expand which represents what your triangle will look like. When this meets your specifications press the right mouse button (Enter). Repeat this process for as many triangles as you need. Colors for the triangle may be selected from the palette at the bottom of the screen. Colors for the box may be selected from the palette at the bottom of the screen. Move the cursor down the screen as far as it will go. Move the cursor directly over the your choice of color. Press Ins (left mouse button) to select the color. Press F9 to return to the draw menu.

Spline

Pressing "S" for spline moves to the spline or elliptical curve drawing function. When you press "S" the cursor appears on the screen. Move the cursor to the upper left corner of the area you want the spline drawn around. Press the left mouse button (Ins) and move away. You will see a "rubber band spline" expand which represents what your spline will look like. When this meets your specifications press the right mouse button (Enter). Repeat this process for as many splines as you need. Colors for the spline may be selected from the palette at the bottom of the screen. Move the cursor down the screen as far as it will go. Move the cursor directly over the your choice of color. Press Ins (left mouse button) to select the color. Press F9 to return to the draw menu.

Ellipse

Pressing "E" for ellipse moves to the ellipse and circle drawing function. When you press "E" the cursor appears on the screen. Move the cursor to the center of the area you want the ellipse drawn around. Press the left mouse button (Ins) and move away. You will see a "rubber band ellipse" expand which represents what your ellipse will look like. When this meets your specifications press the right mouse button (Enter). Repeat this process for as many ellipses as you need. Colors for the ellipse may be selected from the palette at the bottom of the screen. Move the cursor down the screen as far as it will go. Move the cursor directly over the your choice of color. Press Ins (left mouse button) to select the color. Press F9 to return to the draw menu.

Print

Printfile <filename>

Press "P" to use the print function. The graphics are printed by "dithering" which simulates colors with a black and white effect. Enter the proper printer name (i.e. PIO.CR or PIO.CR.LF). Press Enter to print or press F9 to escape this field and return to the menu without printing. Once printing has begun press any key to stop.

Clear

Press "C" to clear the screen. All graphics are permanently erased. The "erase the screen? Y/N" option appears. Press "Y" to erase. Press any other key to escape.

Screencolor

Pressing "S" will allow you to choose the screen color. This carries over to all of HQ and may be changed by moving to this menu. There are also scripting words which allow color changes from scripts. When you press "S" the palette and cursor appear on the screen. Move the cursor down the screen as far as it will go. Move the cursor directly over the your choice of color. Press the right mouse button or Enter to select the color.

Textcolor

Pressing "T" will allow you to choose the text color. This is the color that the menu text appears in. This carries over to all of HQ and may be changed by moving to this menu. There are also scripting words which allow text color changes from scripts. When you press "T" the palette and cursor appear on the screen. Move the cursor to the bottom of the screen as far as it will go and directly over the color you wish the text to be. Press the right mouse button or Enter to select the color. Do not select transparent which is all the way to the left and do not select the same color as the screen color!

HQ_Stacks(HQ) Scriptor and Browser's Guide to Objects

As an HQ browser or scriptor you can make stacks that are easy to use. Stacks are made of one or more cards. Cards are made up of various objects that the user sees and interacts with. This chapter provides descriptions and details of the different object types. Many of the characteristics which make objects unique are controlled by object variables. In this chapter you will see many references to object variables. Each variable has a name which usually is capitalized and begins with the letter "O."

Each object as it is displayed on the screen, has a rectangular area of the screen that it occupies called its bounding box. If you want to know the limits of this bounding box for a particular object, go to the edit menu and look in the variables of the object. The numbers in the object variables OBJX and OBJY represent the pixel location of upper left hand corner of the bounding box. The numbers in the object variables OBJX1 and OBJY1 represent the pixel location of lower right hand corner of the bounding box. ODX and ODY are the width and height of the bounding box respectively.

When the user clicks a mouse key or presses a keyboard key a message is sent through the stack. Each object on the current card is examined to see if the cursor is in the bounding box of that object. If so, the object is told to execute its built-in action. In the case of the switch type object the built-in action of the switch is to throw the switch from one position to another.

Before it executes an object HQ saves the OATTACH object variable numeric value. After the object's built-in action has been executed the OATTACH value is checked. If the numeric value of OATTACH is non-zero the object's scripts are searched for the presence of the message handler (or method) to match the message sent by the mouse button or key clicked by the user. If the user pressed the right mouse button or the Enter key the ENTER message is sent. The first script which is found to have the ENTER message handler has the remainder of the words in that script executed. Thus scripts act as short programs (called message handlers or methods by cs guys) which allow you to extend the user's simple mouse click to any number of actions using any of the resources of the computer.

If the first word in the script is ON the script is a message handler. The message it handles is identified by the message name that follows ON (i.e. if the first two words in a script are "ON ENTER" that script handles the ENTER message).

Object Types, Operations and Variables:

There are twelve basic object types in HQ_Stacks:

1. **String Type.** Click on this object with the right mouse button to enter information. Several function keys are used in typing in information. F1 deletes the character under the cursor. F2 inserts a space under the cursor. F7 for help. F9 to escape the field. Press ENTER to enter the information into the field. The left and right arrow keys as well as Alt-E and Alt-D move the cursor left and right respectively.

The string type object will display the OMESSAGE (MSSG) in a field on the card. If the user clicks inside the bounding box he can enter characters in the field up to the limit. The absolute limit is 31 characters. You may set the limit by entering characters into the OMESSAGE field when the object variables are defined or by setting ODX to a specified value. Since this type uses the built in MDOS font and MDOS text calls, it is limited to text line boundaries. Even though this field accepts character data from the user you can still set OATTACH to a non-zero number and process the OMESSAGE string or anything else.

Variables:

NAME--For reference and search-try to be unique for search.
MSSG--Displayed on screen and edited.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--This is fixed at 8.
OVAL--NA
OMIN--NA
OMAX--NA
OFG--Color the text will be in.
OBG--Background of the field.
OLFG-Outline of the field.
OLBG-NA
OTYPE-String type is 1.
OSTEP--NA
OATTACH-Optional (if zero no search, if non-zero script search done.)

2. **Integer Type.** Click on this object with the right mouse button to enter integer numbers (counting numbers). The function keys work in this field just as in the String Type.

The integer type object will display the OVAL (object value) in a field on the card. If the user clicks inside the bounding box of the field he can enter a number in the field up to the limit. The numeric limit is -32767 to 32768. You set the character limit by setting ODX to 6 (character width) times the number of characters in the field. So if you want a two digit number set ODX to 12. This type uses the built in MDOS font and MDOS text calls and is limited to text line boundaries. Even though this field accepts number data from the user you can still set OATTACH to non-zero and process the OVAL or anything else.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.

OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--This is fixed at 8.
OVAL--This integer value is displayed and can be edited. OMIN--NA
OMAX--NA
OFG--Color the text will be in.
OBG--Background of the field.
OLFG-Outline of the field.
OLBG-NA
OTYPE-Integer type is 2.
OSTEP--NA
OATTACH-Optional (if zero no search, if non-zero script search done.)

3. **String Label Type.** This object appears as a word or words on the card. Although it can be used simply as a label, it may respond to a mouse click or key press. Usually an object like "Next Card-->" is a string label.

The string label type object will display the OMESSAGE (MSSG) in a field on the card. This is a "display only" object, the user cannot change the contents as with the string type. This type uses the built in MDOS font but is not limited to text line boundaries. When you locate this field you can drag it anywhere. The field width is determined by the number of characters in the field. This object type is useful for action targets like "NEXT CARD-->", "DIR B:", etc.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--Displayed on screen.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--This is fixed at 8.
OVAL--NA
OMIN--NA
OMAX--NA
OFG--Color the text will be in.
OBG--Background of the field.
OLFG-NA
OLBG-NA
OTYPE-String label type is 6
OSTEP--NA

4. **Number Label Type.** This object type is an integer number on the card that may or may not respond to your mouse or key clicks.

The number label type object will display the OVAL variable in a field on the card. This is a "display only" object, the user cannot change the contents as with the string type. This type uses the built in MDOS font but is not limited to text line boundaries. When you locate this field you can drag it anywhere. The field width is determined by the number of characters in the field. This object type is useful for displaying number values.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--This is fixed at 8.
OVAL--Value is displayed on the screen.
OMIN--NA
OMAX--NA
OFG--Color the text will be in.
OBG--Background of the field.
OLFG-NA
OLBG-NA
OTYPE-Number label type is 11.
OSTEP--NA

5. **Square Push Button Type.** This object turns off or on in response to click by the right mouse button. This could also be thought of as an indicator "LED" turned on or off by the mouse.

The square push button type object will display its "on" state if OVAL is a non-zero number and display "off" state if OVAL is zero. The change of state is in response to the user pressing the Enter key or the right mouse button. If OATTACH is non-zero a script can be written to allow the object to respond to any key or mouse button but its built-in action is only affected by Enter. You may set or reset OVAL and ODRAW this object if you want it to change states from a script. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODy--The field width can be set manually, with Locate or Browse.
OVAL--If non-zero button is "on". If zero button is "off".
OMIN--NA
OMAX--NA
OFG--Color the "light" will be.
OBG--Background of the "light" field.
OLFG-NA
OLBG-NA
OTYPE-Button type is 3
OSTEP--NA

6. **Check Box Type.** An "x" marks this object when it's on. The right mouse button turns this button on/off.

The Check box type object will display its "on" state if OVAL is non-zero and display "off" state if OVAL is zero. The change of state is in response to the user pressing the Enter key or the right mouse button. If OATTACH is non-zero, a script can be written to allow the object to respond to any key or mouse button but its built-in action is only affected by Enter. You may set or reset OVAL and ODRAW this object if you want the object to display a change made in its OVAL from the script. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODy--The field width can be set manually, with Locate or Browse.
OVAL--If non-zero box is "on". If zero box is "off".
OMIN--NA
OMAX--NA
OFG--NA
OBG--Background of the check box.
OLFG-Color the "x" will be if the box is "on".
OLBG-NA
OTYPE-Check Box type is 7.
OSTEP--NA

7. **Horizontal Slider Type.** As you click the right mouse button on the right side of this object its slider moves right. If you click on the left it moves left.

The horizontal slider type object will display a "bar" at a location within the slider "body" in relation to the current state or values of OMIN, OMAX, OVAL and OSTEP variables. The change of state is in response to the user pressing the Enter key or the right mouse button within the slider body. OVAL can range between OMIN and OMAX at the rate of OSTEP. OMIN must be set at a lower value than OMAX. OSTEP must be 1 or greater. If OATTACH is non-zero a script can be written so that the object will respond to any key or mouse button but its built-in action is only affected by Enter. You may set OVAL to any value and ODRAW this object if you want it to change states from a script. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODy--The field width can be set manually, with Locate or Browse.
OVAL--Determines the relative value of the slider "bar".
OMIN--Minimum value for OVAL, must be lower than OMAX.
OMAX--Maximum value for OVAL, must be higher than OMIN.
OFG--Color the "bart" will be.
OBG--Background of the slider field.

OLFG-Outline color around the slider.
OLBG-NA
OTYPE-Horizontal slider type is 5.
OSTEP--Differential added to or subtracted from OVAL on mouse click.

8. **Vertical Slider Type.** As you click the right mouse button on the top of this object its slider moves up. If you click on the left side it moves left.

The vertical slider type object will display a "bar" at a location within the slider "body" in relation to the current state or values of OMIN, OMAX, OVAL and OSTEP variables. The change of state is in response to the user pressing the Enter key or the right mouse button within the slider body. OVAL can range between OMIN and OMAX at the rate of OSTEP. OMIN must be set at a lower value than OMAX. OSTEP must be 1 or greater. If OATTACH is non-zero a script can be written to make the object respond to any key or mouse button but its built-in action is only affected by Enter. You may change the value of OVAL and ODRAW this object if you want it to change states from a script. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODy--The field width can be set manually, with Locate or Browse.
OVAL--Determines the relative value of the slider "bar".
OMIN--Minimum value for OVAL, must be lower than OMAX.
OMAX--Maximum value for OVAL, must be higher than OMIN.
OFG--Color the "bart" will be.
OBG--Background of the slider field.
OLFG-Outline color around the slider.
OLBG-NA
OTYPE-vertical slider type is 4.
OSTEP--Differential added to or subtracted from OVAL on mouse click.

9. **Light Switch Type.** The right mouse button clicks this on and off as you would expect.

The light switch type will display its "throw" at the "on" position if OVAL is non-zero; and displays the "throw" at the "off" position if OVAL is zero. The change of state is in response to the user pressing the Enter key or the right mouse button. If OATTACH is non-zero, a script can be written to make the switch respond to any key or mouse button but its built-in action is only affected by Enter. You may set or reset OVAL and ODRAW this object if you want it to change states from a script. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.

ODX--The field width can be set manually, with Locate or Browse.
ODy--The field width can be set manually, with Locate or Browse.
OVAL--If non-zero switch is "on". If zero switch is "off".
OMIN--NA
OMAX--NA
OFG--Color the "on" and "off" words will be.
OBG--Background of the switch field.
OLFG-Color the "throw" will be.
OLBG-NA
OTYPE-Light Switch type is 10.
OSTEP--NA

10. **Frame Type.** This is a rectangular area on the screen which may be shaded, bounded by a outline box or even invisible. An invisible frame type object might be hidden under the clown's nose in a picture on the card and when you click the mouse on the nose it honks. This object type is versatile, you never know what an author might do with this. If OATTACH is non-zero a script can be written to allow the object to respond to any key or mouse button but, this type has no intrinsic built-in action. This object may be any size and be located anywhere on the card.

Variables:

NAME--For reference and search try to be unique for search.
MSSG--NA.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--The field width can be set manually, with Locate or Browse.
OVAL--NA
OMIN--NA
OMAX--NA
OFG--NA
OBG--Background color of the "frame".
OLFG-Outline color of the box around the frame.
OLBG-NA
OTYPE-Frame type is 8

11. **Picture Type.** This object type allows the stack author to load a picture onto a card. The art type is HQ. The ONAME field is the filename. The MSSG field can be the Volume name or device name. (This is optional and not recommended as it is better to set the VOLUME or device name using the SOURCE command in the stackscript when the stack loads or cardscript when the card loads. Setting the VOLUME in the stack or card scripts prevents your objects from being too device dependent.) The picture object has no intrinsic or built-in action except that when the card is loaded the picture is loaded from the storage device. If OATTACH is non-zero a script can be written to allow the object to respond to any key or mouse button. This object may be any size and be located anywhere on the card.

Variables:

NAME--Filename Only (no device i.e. PICTURE not DSK4.PICTURE).
MSSG--Optionally the device name (i.e. DSK1) if blank VOLUME is used.

OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.

OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--The field width can be set manually, with Locate or Browse.
OVAL--NA
OMIN--NA
OMAX--NA
OFG--NA
OBG--NA
OLFG-NA
OLBG-NA
OTYPE-Picture type is 9

12. **Font/IconType.** This object type allows the stack author to load font text or icons onto a card. Unlike the picture type, the font or icons must have been loaded by a stackscript or cardscript prior to execution of the object load or the current font will be used. The MSSG field can contain the text printed to the card in the chosen font. Or, if the MSSG field is left blank, OVAL may be set to an ASCII value of a single character or icon. This allows a program depending heavily on icons to have all 128 characters defined as up to 32x32 pixel icons. The font/icon object has no intrinsic or built-in action except that when the card is loaded the font text or icon is displayed on the screen. If OATTACH is non-zero a script can be written to allow the object to respond to any key or mouse button. This object may be any size and be located anywhere on the card.

Variables:

NAME--Object Name make it unique for easy search.
MSSG--Optionally font string to be print when object is loaded.
OBJX--The bounding box may be set manually, with Locate or Browse.
OBJY--The bounding box may be set manually, with Locate or Browse.
OBJX1-The bounding box may be set manually, with Locate or Browse.
OBJY1-The bounding box may be set manually, with Locate or Browse.
ODX--The field width can be set manually, with Locate or Browse.
ODY--The field width can be set manually, with Locate or Browse.
OVAL--If MSSG field is blank the ASCII value of OVAL is displayed.
OMIN--NA
OMAX--NA
OFG--Color of the font text foreground.
OBG--Color of the font text background.
OLFG-NA
OLBG-NA
OTYPE-Font/Icon type is 12.

Stack Author's/Scriptor's Guide to Automatic Stack Loading

If you are creating a stack you may wish to assume that the stack user has very little knowledge of the details of how to get your stack up and running. For this reason, HQ will start itself from the command line and the user will see only the opening card of your stack ready to run, displayed on the graphics screen. He'll never see the HQ browser/scriptor main menu.

The first thing to know in automatic stack loading is that the MDOS command line must contain "HQ" followed by the stackname:

```
HQ E:MYSTACK
```

When you are going to run MYSTACK from drive E:, the file MYSTACK must be on drive E:. If MYSTACK requires any fonts, icons or pictures those should either be on drive E: or the stackscript or cardscript must use SOURCE to set the VOLUME to the proper device where the fonts, pictures or icons are located.

Once HQ is loaded, the HQ files HQ, HR, HS, HQHELP are no longer needed so those files do not have to be present in the same device, directory or subdirectory as the stack files.

When HQ loads it will parse the command line. If there is a parameter following HQ on the command line, HQ will obtain that filename. Next, it checks to see if the file is present. If the file is present, HQ then checks for the correct file type (DF128). Finally, the stack is opened and the "LOADSTACK" message is sent which sends "STACKUP" and "LOADCARDS."

NOTE: All the following details are provided as an in-depth explanation of how a stack is started. A template stack TEMPSTACK is provided on the HQ distribution disk which contains the mentioned scripts, cards and objects. To begin your adventure with HQ you might wish to copy TEMPSTACK, use it and examine its operation before starting your own stack.

The stack author may provide a stackscript which includes the message handler "ON STACKUP.....OFF." The STACKUP handler will include the messages that the author wishes to use to present his stack to the user. The "ON STACKUP" handler can be used to set up user preferences such as, screen color or text color, etc. A STACKUP handler that will set the screen color and menu text color is:

```
ON STACKUP 1 6 COLOR OFF
```

This STACKUP handler sends the message COLOR which takes two parameters. The two parameters for COLOR are the foreground color (text) and background color (screen). In this case 1 is chosen for the text color and 6 for the background color. The 16 colors were not given names in HQ because the stack scriptor may use the PALETTE command to change the hues of all 16 colors.

When the stack is loaded the LOADSTACK message is sent. LOADSTACK really consists of only two messages STACKUP which searches the stackscripts for and "ON STACKUP" handler and LOADCARD which loads the current card. In your STACKUP handler you may point to any card for LOADCARD to load. The default is card 1. The LOADCARD message has two parts as well, the CARDUP message and LOADOBJS. Each card may set up its operating environment and defaults by using a CARDUP handler. To illustrate a simple "ON CARDUP" the BUZZ sound effect is included in the handler to signify audibly that the CARDUP message has been sent.

```
ON CARDUP BUZZ OFF
```

These examples of message handlers, one on a stackscript and one on a cardscript are simple. These scripts could be much more complex, setting the default drive for artwork, setting the default color palette, loading a font, etc. The scriptor needs the most flexible way to automatically load the set up of his stack environment. The STACKUP and CARDUP messages get this done in a straight forward way, but offer the possibility of unlimited flexibility. These messages are all that are used in the TEMPSTACK template stack and will get you started exploring HQ. Use and modify simple examples TEMPSTACK until you have mastered the basics of HQ and are branch out in more adventurous directions.

Stack Author's/Scriptor's Guide to Scripting

Every stack, card and object may have up to 124 scripts. Each script contains up to 256 characters. The scripts may contain any data. Only scripts that begin with the message handler indicator "ON" as the first word of the script will be interpreted by the script interpreter.

The HQ scripting messages which may be recognized and interpreted by the message interpreter are blank delimited strings of characters. This means that messages in the scripting language resemble regular English words as each message must be separated by blanks in order to be recognized. Of course English breaks this rule, but doesn't it break every rule? Besides messages, the HQ message interpreter also recognizes integer numbers. These then are the only things the interpreter understands blank delimited messages and numbers.

A message is like a black box. It does what it does and does it right every time. You don't need to know how the black box does what it does, you just use it because it does what you want. If you know that the CARDUP message in a stackscript will be sent to the ON CARDUP message handler and get your stack running you really don't need to know what, if anything, is the mechanism behind CARDUP.

Some messages don't stand alone like CARDUP. These messages use or produce parameters. A parameter is a number which a message uses to modify its action. If the COLOR message changes the color of the text and screen it must have parameters telling it which colors to change to. Since there are 16 colors in the graphics mode HQ uses, it would follow that the colors are numbered 0-15. If we provide the message COLOR with the numbers 15 and 4 as parameters it changes the text to color 15 and the screen to color 4 (i.e. 15 4 COLOR.)

Some messages use parameters, some messages produce results which are parameters for other messages, some do both. An example of a message which does both is the addition message '+' (plus). Plus always takes two parameters and leaves one result for another message to use. In the appendix there is a list of all HQ built-in messages, what parameters they use, results they produce and their actions. Whenever you write a script you must take into account the parameters needed by and produced by every message. THE PRIMARY RULE OF SCRIPTING IS THAT EVERY SCRIPT IS A CLOSED SYSTEM. EVERY MESSAGE IN A SCRIPT THAT NEEDS PARAMETERS MUST BE FURNISHED THE CORRECT NUMBER OF PARAMETERS. EVERY MESSAGE THAT PRODUCES PARAMETERS MUST CONTAIN ANOTHER MESSAGE THAT USES THEM.

Where to "put" parameters?

Like every right thinking individual, we put our scrap in a heap, our junk in a pile, and our dinner plates in a stack. Since parameters are as important as dinner plates we put them in a stack. The last parameter put on the stack is the first one used (i.e. the top dinner plate). How does the stack work? Who cares. As long as it works! You sent a message that produced a result, it put the result on the top of the stack. You need to use another message that takes a parameter. Where does it get that parameter? Right off the top of the stack, of course! Messages take care of the stack as long as you take account of the balance of parameters produced and consumed.

I said there were only messages and numbers in HQ scripting language, I lied. There are only messages. What looks like a number when you type it into your script is really a message!! That string of numeric characters is a message to the HQ interpreter which needs no parameters but produces a result. The result of message 444 is that the number 444 is now on the top of the stack ready to be eaten by another hungry message. So, the HQ script language consists only of words separated by blanks, some look like numbers but now you know about numbers! The messages take care of the stack as long as you feed them properly and let other messages clean up after them properly.

The scripting language is "case-sensitive," if the message in the appendix is spelled "LINE" and you use "Line" or "line" you will have the dinner plates that were for the LINE family going to who knows who, and bad results to boot.

Every script ends with the "OFF" message. It takes no parameters and obviously doesn't produce any results (which message would use them?) Any words after "OFF" which are in the script are not seen or

acted on by the interpreter. If "OFF" is not at the end of the script the interpreter tries to read whatever it finds after you stopped writing script and interpret it, usually with bad results.

There are many categories of messages in the HQ scripting language. Those that deal with arithmetic, logic, control, i/o, graphics and those that deal with the stack both the "card" stack and the "parameter" stack and those that deal with the objects on the cards.

There are two major appendices, one that deals with the HQ specific messages and one that deals with arithmetic, logic, etc. Each of these appendices show what parameters each message produces or consumes and a short description of what the message does.

HQ Author/ScriptorGuide to Message Authoring:

HQ is totally extensible. That means that in addition to all the messages built into HQ you may create as many messages as you need, which are comprised of existing messages. You don't really create a message, you create a message handler and another script sends the message which your new message handler handles. To create a message handler, you make up the message word that follows ON in a message handler in a script. For example:

New Message Handler: ON MYMESSAGE BEEP HONK BUZZ OFF
New Message Script: ON LMOUSE MYMESSAGE OFF

Your newly created message handler uses the existing HQ sound words BEEP, HONK and BUZZ. Notice that immediately after "ON" the first message is not an HQ built-in message but your very own new message word. Notice that the new message script below your new message handler recognizes the built-in HQ message "LMOUSE" which stands for left mouse button. The next word in the new message script is MYMESSAGE which is sent through the scripts until your MYMESSAGE handler is found, in turn, the BEEP, HONK and MOAN messages are sent by your handler.

BEEP, HONK and MOAN could be substituted for MYMESSAGE in the LMOUSE handler and the same action would result when the left mouse button was pressed.

Message handler "visibility".

If you put MYMESSAGE in an object script, only that object can respond to it. If you put the MYMESSAGE handler in a card script then every object in the card can send MYMESSAGE. If you put MYMESSAGE in a stackscript, every card and every object can send MYMESSAGE. In that sense, where you place MYMESSAGE determines which objects or cards can "see it." All built-in message handlers and messages can be sent and received by all stack, card and object scripts. The only exception is if you "intercept" a message that is normally sent to a built-in handler. You do this by naming your message handler the same name as another. The messages "trapped" by this handler depend on the level of "visibility" that your handler lives at. If you put your named alike handler at the stackscript level it will stop the message at your handler. If you send MYMESSAGE from an object whose scripts don't have MYMESSAGE to a card without MYMESSAGE and there's no MYMESSAGE handler in any stackscript you get a big fat error message.

Your message handler may accept parameters and produce results. This seems like a violation of the PRIMARY RULE OF SCRIPTING, but this is message authoring or handler authoring. You must make sure when you send your message it puts the proper parameters on the stack and eats the leftovers!

Appendix A

HQ Scripting Messages:

Following each message in this list is a diagram. Enclosed in parenthesis is the parameter stack "picture," this follows the capital letter "P." The stack picture shows which parameters MUST be present for proper execution of the message. Following two dashes "--" is the result left on the parameter stack after execution. This result may consist of none, one or more parameters which must be consumed by another message before the end of the script.

Also in this parenthesis may be the capital letter "T," which stands for "trailing" string. Presence of a "T" indicates the message must be followed by a trailing string. A message uses a trailing string like a parameter except, the parameter is a number, the trailing string is a string of characters like a filename. For example, ARTFONT is a message which loads a font. It uses a trailing string which represents the filename of the font to be loaded (i.e. HELVETICA). The message and trailing string combination would be: ARTFONT HELVETICA. In a perfect world there would be no seeking for truth.

Trailing strings come in two types, "blank" delimited trailing strings and "quote" delimited trailing strings. A blank delimited trailing string cannot have a blank character within it, it must be one "word". A quote delimited trailing string may or may not have a blank character within it. In this list we have given them names like type, record, address, byte, number and character to give some idea of the intended function of parameters.

#1 (P type -- record)

This message directs HQ to go to record number 1. #1 must be preceded by the message CARD, OBJECT, SSCRIPT, CSCRIPT, OSCRIPT. It is from these messages that #1 obtains the parameter "type" which it consumes. Using #1 you can direct HQ to go to CARD #1, OBJECT #1, OSCRIPT #1, etc. If you don't need the result "record" use the word DROP to remove it from the parameter stack. DROP makes an unneeded parameter vanish. See GOTO, NXXT and PRRV.

$1 (P -- address)

This is a 32 byte string variable available for your use. If you put something in it you can check its contents with the Memdmp in the Edit & Debug menu. Otherwise, use the Forth vocabulary words to manipulate it.

$2 (P -- address)

This is a second 32 byte string variable for your use. See $1 above.

4TH (P --)

4TH changes the message search path to only search HQ's built-in message vocabulary. It must be "balanced" in the script by the word HQI and used as the "4TH....HQI" pair. 4TH speeds up execution by limiting searching but will not find user defined message handlers. If you have messages that you have created they must not appear between 4TH and HQI in a script.

8CLS (P --)

Clear the graphics screen. HQ uses MDOS graphics mode 8. No graphics remain after this command. You must turn the screen back on with >8 or >8^.

>8 (P --)

Enter the MDOS graphics mode 8 (512x212 pixels 16 colors) . >8 shows 212 lines.

>8^ (P --)

Enter the MDOS graphics mode 8 (512x192 pixels 16 colors) >8^ shows 192 lines.

.CUR (P --)

Display the graphics mouse pointer at the pixel location in the xy variable XLOC.

ARTFONT (T filename --)

Load a "TI-Artist" type font into the font buffer. "Filename" is a blank delimited trailing string and must follow the message ARTFONT in the script (i.e. ARTFONT HELVETICA). The device (i.e. DSK5.) to load from must have been previously set with the SOURCE command (i.e. SOURCE DSK5.)

ARTPIC (T filename --)

Load a "TI-Artist" type picture. The picture is loaded at the current cursor location (XLOC). "filename" is a blank delimited trailing string and must follow the message ARTPIC in the script (i.e. ARTPIC ENTERPRISE). The device (i.e. DSK5.) to load from must have been previously set with the SOURCE command (i.e. SOURCE DSK5.)

BC (P -- address)

BC is the screen background color when the graphics mode is changed. BC is also the background color for regular text (not-font text, that is controlled by BCOLOR).

BCOLOR (P -- address)

The background color for loading ARTPICS or INSTANCES and for printing font text is set in the variable BCOLOR. To change the color use the "!" command (i.e 11 BCOLOR !).

BEEP (P --)

Sound effect.

BLSP (P -- byte-address)

The BLSP byte variable contains the amount of pixels between words when the FTYPE@" message is used to print font text to the screen. Since this is a byte variable "C@" must be used to read the value to the parameter stack and "C!" must be used to store the value. (i.e 4 BLSP C!).

BODY (P -- address)

BODY is an "object" variable which gives the record number of the object body. The object body is where object variables are stored. Not to be trifled with.

BUZZ (P --)

Sound effect.

CARD (P -- type)

CARD produces the CARD descriptor parameter. This parameter is needed by words like #1, GOTO, NXXT and PRRV which direct travel through the stack.

CARDUP (P --)

A message sent to the current card to find any CARDUP message handler. This message tells the card that it is being loaded onto the screen. If the card needs to initialize colors, set VOLUME to a specific device or load a font, a CARDUP message handler should be on one of the cardscripts.

CATALOG (P --)

This message pops up a window in the graphics 8 mode that allows the user to do a catalog of any directory or subdirectory. The window restores the environment after use. An alternative to the DOS and TODOS" commands.

CHSP (P -- address)

CHSP is a variable which controls the inter character space when font text is written to the screen. The value in this variable is the spacing between characters when the FTYPE or FTYPE"@ messages are used. This can be set with the "!" message (i.e. 4 CHSP !).

CMOUSE (P --)

This message is sent when the center mouse button or Del key has been pressed. If you have an object that you would like to have respond to the CMOUSE command include a handler on one of the object's scripts.

COLOR (P fc bc--)

The color message sets the text and background/screen colors. COLOR consumes two parameter from the stack. The foreground color and background color are the values 0-15 in the MDOS graphics mode 8 which HQ uses. The Art program has a color palette which displays the colors. These of course depend on the color palette which may be set with the PALETTE command.

CRAYON (P --)

When the user clicks the right mouse button on the object the CRAYON message pops up the palette and turns the mouse into a coloring/drawing tool. Holding down the right button colors with the currently selected color and pixel width. Moving the cursor back over the object and clicking the left mouse button cancels the CRAYON mode. To change the color move the cursor to the bottom line of the screen directly over the color and press the left cursor button.

CSCRIPT (P -- type)

CSCRIPT message puts the card script type on the parameter stack. This should be used by words like #1, GOTO, PRRV, or NXXT which are the stack's travel director messages.

CUT (P x y x y --)

CUT takes four parameters which are the x and y coordinates of a bounding box. This box surrounds a screen area you want to "cut" for later pasting elsewhere with the PASTE or PASTE@ commands.

CUT&SV (P x y x y -- T filename)

CUT&SV takes four parameters which are the x and y coordinates of a bounding box. This box surrounds a screen area you want to cut and save to the filename trailing the command (i.e. 10 10 40 40 CUT&SV SMALLBOX). The device (i.e. DSK1.) on which SMALLBOX is to be saved must be installed in VOLUME with the SOURCE command prior to execution of CUT&SV. This artwork is saved in the HQ format.

DOS (P --)

This message mimics the MDOS command line interpreter.

DOS" (T MDOS command)

Send the quote delimited string to MDOS. (i.e. DOS" DIR B:") See TODOS" a variation on this command.

ELLIPSE (P x y x y color --)

Ellipse takes five parameters and draws an ellipticallyshaped figure on the screen in the specified color. The first xy pair sets the center of the ellipse. The next xy set the "width and height" of the ellipse.

ENTER (P --)

The enter message is sent when the right mouse button (Enter) is pressed.

FC (P -- address)

FC is the variable which controls the foreground color. See BC.

FCOLOR (P -- address)

The foreground color for loading ARTPICS or INSTANCES and for printing font text is set in the variable FCOLOR. To change the color use the "!" command (i.e 1 FCOLOR !).

FLAT (P -- address)

FLAT is a variable. FLAT controls the number of line segments that are drawn to connect ELLIPSE or SPLINE. The default value is 10.

FONTHI (P -- address)

FONTHI is a variable. FONTHI contains the height (in pixels) of the current font.

FTYPE (T string blank delim.)

FTYPE will type a trailing "blank delimited" string to the screen in the current font. The output begins at the current cursor location controlled by XLOC. (e.g. FTYPE one-word).

FTYPE@" (P x y T:string " delim.)

FTYPE@" will type a trailing "quote delimited" string to the screen in the current font. The output location is specified by the parameters x and y. (e.g. 10 20 FTYPE@" More Than One Word")

GOTO (P type number --)

The GOTO message takes a "type" parameter produced by a word like CARD, OBJECT, SSCRIPT, CSCRIPT and OSCRIPT and a "target" number. The message makes the "target" object the current object. (i.e. CARD 4 GOTO).

HONK (P --)

Sound effect.

HQ (P --)

HQ is the "emergency" recovery word. If you find yourself in a strange place and all else fails try typing HQ<Enter>.

HQFILE (P -- address)

A string variable which contains the name of the current stack.

HQI (P --)

The HQI message "turns on" the HQ interpreter after it has been "turned-off" by the 4TH message. If the 4TH message is used to speed execution HQI must be sent before the end of the script, or before any user defined message is sent or before OFF is sent.

HQPIC (T filename)

Load an "HQ" artwork type picture. The picture is loaded at the current cursor location (XLOC). "Filename" is a trailing string and must follow the message HQPIC in the script (i.e. HQPIC SPACESHIP). The device (i.e. DSK5.) to load from must have been previously set with the SOURCE command (i.e. SOURCE DSK5.)

HYPER# (P -- address)

An object variable which contains the number which was assigned to the current object when it was created. This value may be used but should not be changed.

INDEX (P -- address)

An object variable that contains the index record of records belonging to this object. This value may be used but should not be changed.

INSTANCE (T filename)

Load a "TI-Artist" instance. The instance is loaded at the current cursor location (XLOC). "Filename" is a trailing string and must follow the message INSTANCE in the script (i.e. INSTANCE DIAMOND). The device (i.e. DSK5.) to load from must have been previously set with the SOURCE command (i.e. SOURCE DSK5.)

KEYS (P --)

This message is sent whenever a key other than left, center or right mouse buttons are pressed. You may retrieve the value from the variable OKEY.

LFCR (P --)

When using FTYPE or FTYPE@", LFCR will move to the next line below the current line of type. The distance between the lines of font text will be determined by value in the LNSP variable.

LINE (P x y x y color --)

LINE takes five parameters and draws a line between the two points defined by the xy pairs in "color."

LMOUSE (P --)

The LMOUSE message is sent whenever the left mouse button or the Ins key is pressed.

LNSP (P -- address)

LNSP is a variable. It contains the amount of space that the LFCR command will put between successive lines of font text typed on the screen.

LOADCARD (P --)

The LOADCARD message is sent to a card. It includes both the CARDUP message and the LOADOBJS message.

LOADOBJS (P --)

The LOADOBJS message loads all the objects belonging to the current card.

LOADSTACK (P --)

LOADSTACK is set to HQ. It loads the stack specified in the HQFILE as the current stack and sends the message STACKUP.

MPL (P -- address)

This command gives the address of the master palette. These are 48 consecutive character values between 0-7 which determine the hue of colors displayed on the screen. Change these values with the PALETTE command.

NAME (P type T string bl delim. -- t/f)

NAME searches the designated CARD or OBJECT type for a match between that the ONAME field and the trailing "blank delimited" string. If the named string object is found NAME produces a non-zero and if not found a zero is produced.

NAME" (P type T string quote delim. -- t/f)

NAME" searches the designated CARD or OBJECT type for a match between that the ONAME field and the trailing "quote delimited" string. If the named string object is found NAME produces a non-zero and if not found a zero is produced.

NXXT (P type -- body)

NXXT takes a type parameter produced by CARD, OBJECT. SSCRIPT, CSCRIPT, or OSCRIPT. It moves to the next stack object corresponding to "type."

OATTACH (P -- address)

An object variable. If OATTACH is non-zero and the object is sent a message the scripts attached are searched for message handlers. If OATTACH is zero no search is made.

OBJ@ (P --)

Make the current object's variables available for use

OBODY (P n --)

OBODY uses the number of a CARD or OBJECT body record. It uses that number to make the objects variables available for use.

OBJECT (P -- type)

Put the OBJECT type descriptor on the parameter stack. The type descriptor is used by words like #1, NXXT, PRRV and GOTO to direct travel through the stack.

OBJECT# (P -- address)

An object variable which indicates which object this is.

OBJX (P -- address)

An object variable. The "x" pixel location of the upper left corner of the bounding box around the objects screen area.

OBJX1 (P -- address)

An object variable. The "x" pixel location of the lower right corner of the bounding box surrounding this object on the screen.

OBJY (P -- address)

An object variable. The "y" pixel location of the upper left corner of the bounding box around the objects screen area.

OBJY1 (P -- address)

An object variable. The "y" pixel location of the lower right corner of the bounding box surrounding this object on the screen.

OCARD# (P -- address)

An object variable. This is the card number that this object belongs to. Do not alter this variable.

OCLR (P color--)

This message "clears" an area on the stack to the specified color. The area is bounded on the upper left corner by the values in the xy variable XLOC and on the lower right by the values in the variable XYXY.

OCOL (P -- address)

An object variable. This variable contains the foreground color and background color of the object. It is set using the OFG and OBG values in the variable editing page.

ODRAW (P --)

ODRAW will draw the current object using the current values of its variables. To ODRAW an object you must have the object pointer set for the object. To set this pointer you may use a sequence such as OBJECT #1 DROP, OBJECT NXXT DROP, OBJECT PRRV DROP, OBJECT NAME GORKBOX DROP, etc... (note: the DROP is used to discard the "body" parameter produced by each sequence).

ODX (P -- address)

An object variable. This gives the width, in pixels, of the bounding box which surrounds the object on the screen.

ODY (P -- address)

An object variable. This gives the height, in pixels, of the bounding box which surrounds the object on the screen.

OFF (P --)

OFF must come at the end of every script to balance ON. It turns off the HQ interpreters and moves to the next message.

OFILL (P color x y x y --)

OFILL takes five parameters and fills the box bounded by the two xy pairs with the specified color.

OKEY (P -- address)

OKEY stores the value of the last keypress. This value is the "eight" bit value produced by the keyboard.

OKNOBX (P -- address)

An object variable. The "x" location of the knob in the object if it has one. This variable is not available on the variable page it is set automatically by the object that uses it.

OKNOBY (P -- address)

An object variable. The "y" location of the knob in the object if it has one. This variable is not available on the variable page it is set automatically by the object that uses it.

OLCOL (P -- address)

An object variable. The "line color" foreground and background for any outlines or other lines used in creating an object. This variable is set by the OLFG and OLBG variables in the variables editing screen.

OMAX (P -- address)

An object variable. The maximumvalue which OVAL can attain. This is used in objects like sliders which need to limitthe value of OVAL.

OMESSAGE (P -- address)

An object variable. This is a string which is used in label and string objects for the message they put on the screen. In picture types it can be used as the alternative to VOLUME as a device designator.

OMIN (P -- address)

An object variable. The minimumvalue which OVAL can attain. This is used in objects like sliders which need to limitthe value of OVAL.

ONAME (P -- address)

An object variable. This is the object name. The search messages NAME and NAME" look for a match between their trailing string and ONAME. In Font/Icon and Picture types it is the name of the file to be loaded.

OSCRIPT (P -- type)

OSCRIPT message puts the object script type on the parameter stack. This should be used by words like #1, GOTO, PRRV, or NXXT which direct travel through the stack.

OSTEP (P -- address)

An object variable. In slider object types the OSTEP controls the amount of increase or decrease in OVAL caused by the movement of the slider knob.

OTYPE (P -- address)

An object variable. This sets the object type. There are currently 12 built in object types their number is entered in this field.

1. String.
2. Integer Number
3. Square Button.
4. Vertical Slider.
5. Horizontal Slider.
6. Text Label.
7. Check box.
8. Frame.
9. Picture.
10. Light Switch
11. Number Label
12. Font/Icon.

Additionally, zero is a "null" type and is useful in debugging your stack. If you have objects clash set one temporarily to type zero to suppress its built in action.

OVAL (P -- address)

An object variable. This is the integer numeric value of this object. Different objects make different use of this variable.

PALETTE (T palette string --)

The PALETTE message takes a blank delimited trailing string. It moves that string into the master palette at MPL and changes the hues of colors. The string must be 48 bytes or less and composed of only the characters 0-7. (i.e. PALETTE 777000060373117237510067711733660664040625555777) Each group of three characters sets the red, blue and green hue of one screen color respectively. Generally, the higher the value the lighter the component intensity of the resulting shade. The string may include up to 48 values (16 x 3) and will change every value according to the string so that if you wanted to change color 1 to white use PALETTE 777777. However, to change color 15 to white all 48 characters would be necessary. Since this string is also used for dithering values in printing, it is recommended that color 0 always be set at 777 so the background of printing is "white," this makes no difference on the screen since 0 is transparent regardless of palette. The above string is the "standard" palette.

PASTE (P --)

A message which copies the currently saved rectangular screen image copied to the screen buffer by CUT to the xy location pointed to by XLOC.

PASTE@ (P x y --)

A message which copies the currently saved rectangular screen image copied to the screen buffer by CUT to the xy location given by the parameters x and y.

PCHR (P char --)

PCHR puts the character indicated by the parameter char on the screen. This is based on the current font loaded into the font buffer. "Char" is the ASCII value of the character to be written. The character is written at the current cursor location indicated in XLOC.

PIXHI (P -- address)

A variable which contains the height in pixels of the line height. Before drawing a line you may change this value to make a thicker line. Maximum value is 7.

PIXWD (P -- address)

A variable which contains the width in pixels of a graphics line. You may alter this value to change the width of the line. Maximum value is 7.

PRINT (T filename --)

Print the graphics screen to the filename given in the blank delimited trailing string. Filename must be a valid output device like PIO.CR or PIO.CR.LF. Output is in Epson graphics, double density.

PRRV (P type -- body)

Move to the previous CARD, OBJECT, or script, depending on which type is indicated. The body of the object is produced for use by OBODY or it can be DROPped.

RECT (P x y x y color --)

RECT takes five parameters and draws a rectangle with upper left corner and lower right corners defined by the xy point pairs. The color is specified for the lines drawn. The "fatness" of the lines is set by PIXWD and PIXHI.

SINDEX (P -- address)

An object variable giving the script index record. Do not alter this value.

SOUND (P --)

Produce a sound by using the values for the MDOS sound command. These values must have previously been stored in the SOUNDBOX.

SOUNDBOX (P -- address)

A set of sixteen consecutive bytes of memory which may be loaded with the values for producing a sound with MDOS calls.

SOURCE (T devicename)

SOURCE stores the blank delimited trailing string into the variable VOLUME. VOLUME is used by any message that reads or writes a disk file. VOLUME is the devicename to which the filename is appended to make a complete pathway to a file. (i.e. SOURCE DSK4.)

SPLINE (P x y x y color--)

SPLINE takes five parameters. The two xy pairs are the ends of a curved line that SPLINE draws in the color specified. "Fatness" of the line is determined by PIXWD and PIXHI.

SSCRIPT (P -- type)

SSCRIPT produces the type descriptor that is used by #1, NXXT, PRRV or GOTO to move to a particular stack script.

STACKUP (P --)

A message sent to a stack when the stack is being loaded. If you wish to have the stack set up a SOURCE device, or set up screen colors etc., you will need to provide a handler for STACKUP.

STAGEA (T filename)

Load the "TI-Artist" picture named in the trailing string "filename" into the artwork "staging" area. From that position it may be brought to the screen by PASTE or PASTE@.

STAGEH (T filename)

Load the HQ art picture named in the trailing string "filename" into the artwork "staging" area. From that position it may be brought to the screen by PASTE or PASTE@.

STAGEI (T filename)

Load the "TI-Artist" instance named in the trailing string "filename" into the artwork "staging" area. From that position it may be brought to the screen by PASTE or PASTE@.

STDFONT (P --)

Load the standard font, resident in the Geneve into the font buffer for use by PCHR, FTYPE or FTYPE@".

SVPIC (P x y x y T filename --)

SVPIC takes four parameters and a trailing string. The message saves the screen area bounded by the two xy pairs of points to the "filename."

TO (P x y --)

TO takes two parameters. TO draws a line from the current location in the xy variable XYTO to the current location of XLOC. The line is drawn in the current color. The current location of XLOC is then put into XYTO. This allows a progression of lines to be drawn by changing one point set.

TODOS" (T quote delim. string --)

TODOS sends the trailing quote delimited string to MDOS. The graphics screen is reserved and the DOS command is executed on the text screen for increased speed. When the command is finished press any key to return to the graphics environment.

TPAFONT (T filename --)

TPAFONT loads a font from the trailing blank delimited string "filename." The font is loaded into the font buffer for use by PCHR, FTYPE or FTYPE@." The CHSP, LNSP AND BLSP variables are set by the values in the font.

V (P n -- address)

V is a user variable. It uses the parameter "n" to compute an offset into the VARRAY. The V produces sixteen consecutive addresses for use as storage variables in the user's scripts. The variables are numbered 0-15. (i.e. 8 V puts the address of the ninth variable word on the parameter stack).

VARRAY (P -- address)

VARRAY is a block of sixteen consecutive variables for use by the V message which provides the addresses offset into VARRAY.

VOLUME (P -- address)

VOLUME is the location of the device name which, when combined with a filename, creates a complete path to a disk files. A device name is stored into VOLUME by SOURCE.

XLOC (P -- address)

An xy variable pair that contains the location of the cursor on the screen. The screen is 512 pixels wide by 192 pixels high. Valid values for XLOC are 0-511 in the x and 0-191 in the y. Use the Forth words XY@, and XY! to easily manipulate this useful xy variable.

XY (P n -- address)

XY is a message which uses a parameter n to produce 8 xy pair variables. The values 0-7 are valid. These variables are useful for storing screen point location sequences. (i.e. 4 XY yields the address of the 5th xy pair in the address array.)

XYARRAY (P -- address)

XYARRAY is a block containing 8 consecutive xy pair variables. XY is used to produce addresses in this block.

XYTO (P -- address)

XYTO is an xy variable pair which contains the last point to which a line was drawn with the TO command. This may also be loaded before the TO command is given to start a progressive series of connected lines.

YLOC (P -- address)

YLOC is the y portion of the XLOC xy pair. It is given a separate name for convenience and access to just the y value if needed.

begin (P --)

"Begin" is one element in a "conditional" structure used to control the flow of a program. There are two types of looping structures which begin with begin:

"pre-test" structure:

begin.....while.....repeat:

"post-test" structure:

begin.....until:

In the begin...while "pre-test" structure, the statements in the script between begin and while are executed and while consumes the parameter on top of the stack. If the parameter is zero, while directs the script to jump to the next message after repeat. If the parameter is non-zero, the words after while through repeat are executed, then repeat sets the execution back to begin to run through the cycle again.

In the begin....until "post-test" structure, the statements between begin and until are executed. Then, until consumes the parameter on top of the stack. If the parameter is zero, the structure executes again if non-zero control passes back to begin to execute again.

repeat (P --)

Repeat sends control of execution back to begin.

until (P t/f --)

Until requires a "true or false" parameter. Until (and any conditional like if, or while) considers a non-zero parameter "true" and a zero parameter "false." If the parameter is true, until sends control of execution to the message after until. If false, execution is sent back to begin.

while (P t/f --)

While requires a "true or false" parameter. While (and any conditional like if, or until) considers a non-zero parameter "true" and a zero parameter "false." If the parameter is true, while sends control of execution to the message after while. If false, execution is sent to the message after repeat.

do (P hi lo--)

The do...loop and do....+loopstructures are called "indexed" loops. Do consumes two parameters from the stack and uses them internally. The first parameter is the low number of the two and is the initial value. The second is the high number and is the terminal value. The difference between the initial value and terminal value is the number of times the messages between do and loop are

executed. (i.e 10 0 do ..... loop). If the parameters are 10 and 0 the loop is executed 10 times. Care should be taken to get the numbers in the proper order.

I (P -- n)

The "i" message is available inside the do...loop, do...+loopand is the current index value. After the loop begins the initial value is increased each time through the loop and becomes the "index" value of the loop. The message "i" reflects this value and produces it on the stack.

loop (P --)

loop terminates the do...loop structure. It increases the index value by one then compares it with the terminal value. If the values are equal the loop terminates. That is the word after loop is the next message executed.

+loop (P n --)

This message is an alternative way to change the value of the index in a do...loop. A parameter must be supplied to the message "+loop"each time through the +loop. +loopadds the value of the parameter to the +loop index. +loopchecks the loop index and if it becomes equal to or greater than the terminal value the +loopterminates. (i.e. 40 0 do 10 +loop)In this case the +loopis done four times as the index is increased 10 each time instead of 1 as in the loop.

If (P t/f --)

If is used to control the flow of execution of messages in a script. If consumes a parameter. This parameter is a true or false value. The parameter is considered true if it is non-zero, false if zero. If the parameter is a true the words following "if" are executed. If false control is passed to then or else. This is slightly different than if..else...then structures in other languages.

Types of "If" conditional structures:

if.....then

if.....else...then

else (P -)

else is a marker for "if" to pass control to in the false case. Else also passes control to then after the true case executes.

then (P -)

then is a marker for "if" or "else" in passing the flow of control conditionally.

Appendix B:

Level 2 Scripting Messages:

The level 2 scripting messages are very much like their counterparts in the Forth language. The list shows each message followed by a parameter stack "picture." The "P" at the beginning of the picture indicates parameters which must be present before the message is sent. Following two dashes in the picture are the values produced on the stack after the message has executed. If the picture includes a "T" this indicates the message must be followed by a trailing string. Trailing strings can be strings of characters to be printed to the screen. For example, the message '.'" (called dot-quote) prints a quote delimited string to the screen (i.e. ." Hi There"). Additionally, any message which has a star at the end of the picture is a message that has a full description in a Forth related text or reference like "Starting Forth" by Leo Brody or the TI-Forth manual. We have not given extensive descriptions of Forth-like words since many are not necessary for use of HQ, they are but provided for completeness. HQ specific words are not "starred" and are explained in detail.

Following the tradition of Forth, the following abbreviations are used in the stack pictures to help indicate the intended function of parameters. Remember all parameters are numbers.

addr....address
b.....byte
ch.....ascii character code.
cccc....ascii character string.
n.....number
d.....double precision number (two number values never used by HQ)
xy.....two consecutive values on the stack. Usually refer to a pixel location on the screen.
u.....unsigned number.
vaddr...video processor address.

! (P n addr -- *) Store the number n at addr.

!" (P addr-- T cccc") Store the quote delimited string cccc at addr with a leading count

(P -- *) Advanced number formatting word. See a Forth text.

#> (P -- *) Advanced number formatting word. See a Forth text.

#S (P -- *) Advanced number formatting word. See a Forth text.

((P -- *)

* (P n1 n2 -- n3 *) Multiply n1 by n2 and produce n3.

*/ (P n1 n2 n3 -- n4 *) Multiply n1 times n2 and divide by n3 producing n4.

*/MOD (P n1 n2 n3 -- n4 n5 *) Multiply n1 times n2 and modulo divide by n3. Produce n4 remainder n5 quotient.

+ (P n1 n2 -- n3 *) Add n1 and n2 and produce n3.

+! (P n addr -- *) Add n into addr.

+- (P n1 n2 -- n3 *) Apply the sign of n2 to n1 and produce n3.

- (P n1 n2 -- n3 *) Subtract n2 from n1 and produce n3.

-DUP (P n1 -- n1 n1 *) if n1 =0 (P n1 -- n1 *) if n1 <>0 Duplicate if not zero.

-TRAILING (P addr cnt -- addr cnt *) Remove the trailing blanks from the string at addr for cnt chars. Leave the new addr and cnt.

. (P n1 -- *) Print the value of n1.

." (P -- T cccc" *) Print the trailing quote delimited string.

.R (P -- *) Advanced number formatting, see Forth text.

/ (P n1 n2 -- n3 *) Divide n1 by n2 and produce the quotient n3.

/MOD (P n1 n2 -- rem n3 *) Modulo divide n1 by n2 and produce rem and n3 quotient.

0 (P -- 0 *) Produce zero.

0< (P n -- f *) Compare n to zero. f is non-zero if n is less than zero. f is 0 if n is greater than or equal to zero.

0= (P n -- f *) Compare to zero. f is non-zero if n is zero. f is 0 if n is not.

1 (P -- 1 *) Produce 1.

1+ (P n1 -- n2 *) Add one to n1 and produce n2.

1- (P n1 -- n2 *) Subtract 1 from n1 and produce n2.

2 (P -- 2 *) Produce 2.

2* (P n1 -- n2) Double n1 and produce n2. An arithmetic left shift.

2+ (P n1 -- n2 *) Add 2 to n1 and produce n2.

2- (P n1 -- n2 *) Subtract from n1 and produce n2.

2DROP (P n n --) Drop 2 numbers.

2DUP (P n1 n2 -- n1 n2 n1 n2) Duplicate two top items.

2MOVE (P addr1 addr2--) Move two numbers from addr1 to addr2.

2OVER (P n1 n2 n3 n4 -- n1 n2 n3 n4 n1 n2) Copy 2 numbers to top of parameter stack.

2SWAP (P n1 n2 n3 n4-- n3 n4 n1 n2) Jump 2 numbers.

3 (P -- 3 *) Produce 3

< (P n1 n2 -- f *) If n1 is less than n2 produce a non-zero f, otherwise produce zero.

<# (P -- *) Advanced number formatting message see Forth text.

<> (P n1 n2 -- f) If n1 is not equal to n2 produce a non-zero f, otherwise produce zero.

= (P n1 n2 -- f *) If n1 equals n2 produce a non-zero f, otherwise produce zero.

=CELLS (P n1 -- n2 *) Add 1 to n1 if it is odd to make it even and produce n2.

> (P n1 n2 -- f *) If n1 is greater than n2 produce a non-zero f, otherwise produce zero.

>YMM (P -- regs) The message sequence >YMMCOMM writes the top graphics screen to the lower, hidden screen. YMM>COMM brings it back. See YMM> and COMM.

? (P addr -- *) Print the number in addr on the screen.

?KEY (P -- ch *) Produce the 7 bit value of any key that is being pressed or 0.

?KEY8 (P -- ch *) Produce the 8 bit value of any key that is being pressed or 0.

@ (P addr -- n *) Produce the number stored at addr.

@! (P addr1 addr2--) Move the number at addr1 to addr2.

A+! (P addr1 addr2 cnt-- *) Add the numbers at addr1 into addr2 for cnt addresses.

ABS (P n1 -- n2 *) Take the absolute value of n1 and produce n2.

AND (P n1 n2 -- n3 *) Produce n3, the logical bitwise AND of n1 and n2.

ASCII (T cccc -- n1 *) Take the first character from the blank delimited string cccc and produce its ascii value n1.

BASE (P n -- *) Change the number BASE.

BL (P -- 32 *) Produce the "blank" ascii value 32.

BLANKS (P addr n -- *) Store n blanks beginning at addr.

C! (P n1 addr -- *) Store the low byte of n1 at addr.

C@ (P addr -- n1 *) Produce the byte at addr into the low byte of n1.

CLLB (P -- regs) CLLB COMM will clear the hidden video screen.

CLLL (P -- regs) CLLL COMM will clear the top 212 lines of the video screen.

CLLM (P -- regs) CLLM COMM will clear the lines between 192 and 212.

CLLU (P -- regs) CLLU COMM will clear the lines between 0 and 191.

CLR (P addr --) Clear the address (i.e. set it to zero).

CLS (P -- *) Clear the screen for text.

CMOVE (P addr1 addr2 cnt -- *) Move the cnt number of bytes from addr1 to addr2. Move from addr1 first.

CMOVE> (P addr1 addr2 cnt-- *) Move the cnt number of bytes from addr1 to addr2. Move from addr1 cnt + 1- first.

COMM (P regs --) Write the regs to the 9938A video processor. See >YMM,YMM>,CLLL, CLLB, CLLU, CLLM.

COS (P n1 -- n2) Produce the cosine n2 of the angle n1. n2 is cos(n1) times 16384.

COUNT (P addr1 -- addr2 cnt *) Produce the addr and count from the counted string stored at addr1.

CR (P -- *) Send a carriage return, line feed pair to the screen.

CUR. (P --) Display the graphics cursor at the xy pixel location stored in XLOC.

CURPOSX (P -- addr) The "x" or column address of the text cursor. May be retrieved as xy pair with CURPOSY using XY@ and XY!

CURPOSY (P -- addr) The "y" or row address of the text cursor.

D+ (P d1 d2 -- d3 *) Add the double precision numbers d1 and d2 to produce d3.

D+- (P d1 n -- d2 *) Apply the sign of n to d1 and produce d2.

D. (P d1 -- *) Print the double number d1.

D.R (P d1 n -- *) Advanced number formatting. See a Forth text.

DABS (P d1 -- d2 *) Produce d2 the absolute value of d1.

DEC (P addr --) Subtract one from the number in addr.

DECIMAL (P -- *) Set the number base to decimal (base 10).

DMINUS (P d1 -- d2 *) Produce d2, the negative of d1.

DOSIO (P -- *) The MDOS XOP call 8. Load the workspace regs at REG0 before DOSIO.

DPL (P addr -- *) Forth number formatting control. See a Forth reference.

DROP (P n -- *) Remove the number n from the parameter stack.

DUP (P n1 -- n1 n1 *) Duplicate n1.

EMIT (P ch -- *) Print the seven bit value ch.

EMIT8 (P ch -- *) Print the eight bit value ch.

ERASE (P addr n -- *) Store zero in n consecutive bytes starting at addr.

FILL (P addr cnt byte -- *) Store byte at cnt addresses beginning at addr.

GOTOXY (P x y -- *) Move the text cursor to the row and column.

HCUR (P --) Hide the graphics cursor.

HEX (P -- *) Set the number base to hex (base 16)

HOLD (P -- *) Advanced Forth number formatting. See Forth text.

INC (P addr --) Add one to the number at addr.

KEY (P -- ch *) Wait for a key to be pressed, produce its 7 bit value and print it on the screen.

KEY8 (P -- ch) Wait for a key to be pressed, produce its 8 bit value and print it on the screen.

KSTOP (P --) Halt execution until a no key is being pressed.

M* (P n1 n2 -- d1 *) Multiply n1 by n2 and produce the double number d1.

M/ (P d1 n1 -- rem quot *) Divide the double number d1 by n1 and leave remainder and quotient.

M/16K (P d1 -- quot) Divide the double number d1 by 16384 and produce quotient.

M/MOD (P ud1 u2 -- u3 ud4 *) Divide unsigned double ud1 by unsigned u2 and produce rem u3 and unsigned double quotient ud4.

MATH (P --) The MDOS XOP call 10. Load workspace registers at REG0 before MATH.

MAX (P n1 n2 -- n *) Compare n1 and n2 and produce the maximum n.

MDOS (P --) Jump immediately to MDOS, no goodbyes.

MEMORY (P --) MDOS XOP call 7. Load workspace registers at REG before MEMORY.

MIN (P n1 n2 -- n *) Compare n1 and n2 and produce minimum n.

MINUS (P n -- n *) Produce the negative of n.

MOD (P n1 n2 -- n3 *) Divide n1 by n2 and leave only remainder n3. n3 has sign of n1.

MODE (P n --) Change the MDOS graphics mode to n. HQ operates in mode 8 only.

MOUSE? (P -- 0) no mouse button press.
(P -- ch ch) mouse button pressed. If the mouse is being pressed return 2 copies of button value else return 0. The current location of the mouse is in XLOC as an xy pair.

MOVE (P addr1 addr2 cnt-- *) Move cnt words from addr1 to addr2.

MSP (P -- addr) This address holds the mouse speed 0-7.

MSTOP (P --) Halt execution till the fool stops fiddling with the mouse.

NEG (P addr --) Make the number in addr negative.

NOP (P -- *) YUP.

OPQ (P --) Opposite of XSE

OR (P n1 n2 -- n3 *) Logical bitwise OR of n1 and n2 produces n3.

OVER (P n1 n2 -- n1 n2 n1 *) Copy the second item to top.

PAUSE (P -- ch) Produce ch value of key pressed. If F4 is pressed halt till another key is pressed. Zero is produced otherwise.

PICK (P cnt -- n1 *) Copy the cnt item from the stack and produce it.

PORT3 (P -- addr) A VDP port address.

REG! (P ...cnt --) Store the cnt+1numbers into REG0. This useful for MDOS commands. (i.e. 0 8 4 2 REG! VIDEO will set the graphics 8 mode.) If you are not an exburt on MDOS calls please don't frustrate yourself with this.

REG0 (P -- addr) Workspace register set used for MDOS XOP calls only. Post mortems possible using Memdmp utility in Edit & Debug menu.

REGE (P -- *) Erase 16 registers at REG0

ROT (P n1 n2 n3 -- n2 n3 n1 *) Rotate the n1 to the top position.

S->D (P n -- d *) Make n a double number. Doubles take up two stack positions.

SIGN (P -- *) An advanced Forth number formatting message. See Forth Ref.

SIN (P n1 -- n2) Produce n2 the sine of n1. n2 is actually sin(n1)*16384.

SKIP (P addr1 cnt1 byte--addr2 cnt2) From addr1 skip any bytes that match byte and adjust to addr2 cnt2.

SLA (P n1 n2 -- n3) Produce n3 by shifting n1 arithmeticallyleft n2 positions. 0 shift does not shift.

SPACE (P -- *) Print a space.

SPACES (P n -- *) Print n spaces.

SPAT (P -- addr *) Graphics 8 mode sprite address table.

SPCT (P -- addr *) Graphics 8 mode sprite color table.

SPGT (P -- addr *) Graphics 8 mode sprite graphics table.

SRA (P n1 n2 -- n3 *) Produce n3 by shifting n1 arithmeticallyright n2 places.

SRC (P n1 n2 -- n3 *) Produce n3 by shifting n1 circularly right n2 places.

SRL (P n1 n2 -- n3 *) Produce n3 by shifting n1 logically right n2 places.

SWAP (P n1 n2 -- n2 n1 *) SWAP the two top numbers on the stack.

SWPB (P n1 -- n2) Produce n2 by swapping the bytes of n1.

SWPS (P n1 n2 -- n3 n4*) Swap the bytes within n1 and n2 leaving n3 and n4. Equivalent to SWPB SWAP SWPB SWAP.

TAB (P n --) Move to the nth column in the text screen.

TOGGLE (P addr b -- *) Set the bits in addr corresponding to the 1 bits in byte b.

TRIM (P a -- *) Remove leading and trailing blanks from a "counted string" in addr."

TYPE (P addr cnt -- *) Print the cnt bytes at addr to the screen.

U* (P u1 u2 -- ud *) Produce the unsigned double ud from unsigned u1 times u2.

U. (P n -- *) Print n as an unsigned number.

U.R (P u n -- *) Advanced number formatting message. See a Forth reference.

U/ (P ud u1 -- u2 u3 *) Unsigned divide double ud by u1 leaving unsigned rem and quot.

U/L (P ch -- ch *) Change an upper case letter to its lower case equivalent.

U< (P u1 u2 -- f *) If unsigned u1 is less than u2 produce f as non-zero, otherwise zero.

UD. (P d -- *) Print a double as unsigned.

UD.R (P d n -- *) Advanced Forth output formatting. See Forth reference.

UKEYS (P -- *) MDOS XOP 5. Load REG0 with workspace reg values before sending UKEYS.

UTIL (P -- *) MDOS XOP 9. Load REG0 with workspace reg values before sending UTIL.

VAND (P b vaddr --) AND the byte and the video address.

VDPRD (P -- addr) Video port address.

VDPST (P -- addr) Video port addr.

VDPWA (P -- addr) Video port addr.

VDPWD (P -- addr) Video port addr.

VFILL (P vaddr cnt byte -- *) Write cnt bytes of value "byte" beginning at vaddr.

VIDEO (P -- *) MDOS XOP call 6. Load the workspace regs at REG0 before VIDEO.

VMBR (P vaddr addr cnt -- *) Read cnt bytes from vaddr to addr.

VMBW (P addr vaddr cnt -- *) Write cnt bytes from addr to vaddr.

VPAGE (P vaddr -- vaddr *) Vpage consumes vaddr, strips the high order bits, writes them into VDP register 14 and returns the low order vaddr bits. This should immediately proceed a video read or write operation. Follow this operation with 0 VPAGE DROP to reset the video page to 0. To write to the lower 64K of video store 4 in the variable HI, make the read/write and store 0 in HI. (i.e. byte vaddr 4 HI ! VPAGE VSBW 0 HI ! 0 VPAGE DROP .)

VRA (P vaddr addr -- *) Read bytes from vaddr to addr using the count stored by VWP.

VRP (P cnt -- *) Prepare to read cnt bytes from video. This prepares a "fast read array" that can be used by VRA any number of times in one script. Cnt cannot exceed 128 bytes or a system crash may occur.

VSBR (P vaddr -- byte *) Read the byte at video address vaddr.

VSBW (P byte vaddr -- *) Write byte to vaddr.

VWA (P addr vaddr -- *) Write the bytes from addr to vaddr using the count stored by VWP.

VWCR (P color reg# -- addr *) Write color to video color register.

VWP (P cnt -- *) Prepare to write cnt bytes to video. This prepares a "fast write array" that can be used by VWA any number of times in one script. Cnt cannot exceed 128 bytes or a system crash may occur.

VWTR (P byte n -- *) Write byte to video register n.

VXOR (P byte vaddr -- *) XOR the byte into the vaddr

WINDOWS (P -- *) MDOS XOP 11. The XOP for 9640 WINDOWS by Beery Miller. The current version of HQ may not respond well to WINDOWS commands but Mr Miller and I are working on making future versions fully compatible.

XOR (P n1 n2 -- n3 *) Produce n3 from the xor of n1 and n2.

XSE (P -- *) Opposite of OPQ.

XY! (P x y addr -- *) Store x and y pairwise in two consecutive words at addr.

XY+ (P n1 n2 n3 n4-- n5 n6 *) Add n1 to n3 to produce n5, add n2 to n4 to produce n6.

XY>CR (P x y -- c r *) Take the xy pixel location and make a text column row location.

XY@ (P addr -- x y *) Fetch x and y pairwise from addr.

XYXY (P -- addr *) An xy pair variable. Most often holds the lower right xy pixel location pair of a bounding box.

Y@ (P addr -- y *) Fetch the y value from a xy pair variable like XLOC or XYXY.

YMM> (P -- addr *) YMM>COMM moves the 212 lines of graphics in the hidden graphics page in the VDP to the visible page.

HQ_Stacks Artwork Storage Format.

There are many, many color artwork formats available today. There is GIF, TIFF, ZIF, RLE, MYART, and TI-Artist just to name a few. None of these types offered the simplicity, single purpose utility and full public disclosure that would have made it an easy choice to store HQ artwork.

Rather than start another proprietary art format and try to make HQ load every other proprietary art format, practicality dictated a straight forward, published format. This will allow any interested programmer to build a cheap conversion program for a particular format into HQ and back.

As indicated elsewhere, we plan to write as many conversion utilities which run right out of MDOS as time and information allow. If you have the COMPLETE details on a format and wish to share either the format and a couple samples or a conversion program please send them to us. Please don't send us anything that will get us in trouble like proprietary or copyrighted information, programs or artwork.

HQ artwork is stored in the IV254 format. The convention we have adopted is to end the filename with "_H". HQ uses the 16 bit color MDOS mode 8 graphics. The data is stored in two parts, a 12 byte header file, and color data files compressed with a simple, byte-oriented run length encoding (RLE).

The header file consists of 5 parts:

1. The hex string >0347,>5236 as an identifier.
2. The "x" pixel location where the art originated--2 bytes.
3. The "y" pixel location where the art originated--2 bytes.
4. The width in pixels of the artwork--2 bytes.
5. The height in pixels of the artwork--2 bytes.

The color data files consist of 2 records for each horizontal line of color pixels in the artwork. The first record comprises the RLE compression of the first 128 bytes, the second record the next 128 bytes of the line.

This particular RLE format, has two types of storage uncompressed strings of bytes and compressed runs of bytes:
An uncompressed string consists of a count byte followed by that number of bytes of data. A string is indicated if the count byte is less than 128.

A byte run consists of a count byte followed by a run value. A run is indicated if the count byte is greater than 127 (i.e. the high bit is set.) To obtain the run count take the 2's complement of the count byte (i.e. 255 XOR 2+ .)

Updates and Warranty

The enclosed registration card is your ticket to future updates and warranty service on HQ_Stacks(HQ). You must complete the card and return it to us in order to become a registered user. If you buy HQ from a registered user, you may write to us and apply for registered user status.

Update policy: All registered users are eligible for any updates of HQ_Stacks. To receive your update, you will be required to return your original HQ system diskette with its serial number label intact. A small amount may be charged to cover shipping, materials handling and the upgrade fee itself.

Warranty: Any defective diskette or booklet will be repaired or replaced if returned to McCann Software within 90 days. McCann Software will make the judgement as to whether the diskette or software is defective and will either repair or replace it at our option. If you accidentally erase or damage the original HQ system diskette, return it with $10 for replacement. PLEASE MAKE AND USE A BACK UP COPY! This software is like a do-it-yourself book. It is for your personal use. You are the sole judge of its usefulness and accuracy. McCann Software is not responsible for any damage incurred by you, your associates, or customers as a result of the use of HQ_Stacks.

Fair Use Agreement: Your purchase of this software is not a license to make copies or distribute copies to any other person. Under the copyright laws this software is like a book. You may lend the original HQ system to a friend for use on his machine, but you and your machine cannot run a copy at the same time. You may sell your original copy of the software, but you cannot retain even one copy. You may use it on any computer, but not on more than one computer at a time. You may not put your name on any part of the software as its author.

Note on Software Piracy: We will pay a modest reward for information leading to the arrest and conviction of, or recovery of damages from, any software pirate engaged in pirating McCann Software products. Please write us if you have any questions about this or any of our products.

McCann Software
4411 North 93rd Street
Omaha, NE 68134