CodeHead Utilities Release #4 Notes

Release Date: Thursday, June 20, 1991

PLEASE REFER TO THE PRINTED ADDENDUM FOR A DESCRIPTION OF FEATURES OF RELEASE 3.

Remember that the following programs may be installed as desk accessories as well as being run as programs, simply by renaming their extensions from PRG to ACC:

Art Gallery

ShowMem4

Spooler Manager

MultiFile

There are four other desk accessories on the CodeHead Utilities disk which we've named with an extension of ACX. If you want to install any of these accessories, you'll have to rename their extensions to ACC (or load them into MultiDesk).

CODEHEAD RAM DISK

The CodeHead RAM Disk now allows you to install up to two different drives. The two drives are shown at the top of the main screen along with their sizes. If no second drive is selected, a dash will be shown as Drive 2.

The currently active drive letter will be shown in inverted text. You can change its size by moving the mouse or using the arrow keys to change the currently selected size. You can also change the size by hitting the appropriate key (1–9, A–X, F1–F4). Previously, pressing these keys would immediately install a RAM disk. Now they only change the size, allowing you to continue configuring before installing. The selected size will also be shown at the top of the screen.

Hitting the Help key will change the "current drive" selection between Drive 1 and Drive 2. This will also change the location of the cursor in the size field to reflect the size of the currently selected drive.

As before, you can set the drive letter you wish with Control A through Z (drives Q–Z are available but not recommended). Hitting Esc will cancel Drive 2 (denoted by a dash), even if Drive 1 is the current selection. Previously Esc would exit from the program without installing a RAM disk.

Only Drive 1 may be piggy–backed or have a CCP file loaded. You are also restricted from setting either drive to the same letter as the other drive.

When you are finished configuring the RAM disk, you can install it by pressing Return or the left mouse button. You can exit without installing by pressing Undo or the right mouse button.

Reset–Proof Print Spooler

The CodeHead RAM Disk also now sports a reset–proof print spooler! To our knowledge there are no other spoolers of this type for Atari computers.

The spooler is designated by an asterisk. To install a spooler in the RAM disk, hit the asterisk key (*). This will place an asterisk as the drive designator for Drive 2. The spooler takes the place of Drive 2; you are not allowed to install two drives and a spooler at the same time. The spooler may be any size from 1K up to the available memory in your computer (to obtain sizes other than the preset sizes you must use the variable "X" size). You must install a RAM disk in Drive 1 in order to use the print spooler. If you are only interested in installing a print spooler, install a 1K RAM disk for Drive 1.

Once installed, the spooler will intercept all BIOS output to the printer port. It will accept printer data as fast as the application can output it and feed it to the printer as fast as the printer can take it. As with the CodeHead Spooler (stand–alone spooler also in this package), the CodeHead RAM Disk spooler will also spool an "Alternate–Help" screen dump.

Bear in mind that the spooler will only accept the data as fast as the application can send it. In the case of a standard TOS screen dump, it still takes about a minute and 15 seconds for the ROM routines to dump the screen contents into the spooler (with a monochrome screen). This uses about 64K worth of data. Meanwhile, your printer will have started printing the screen dump and will be partially finished (depending on printing speed) before control is returned to you.

If you perform a warm reset (via hot keys or the reset button), the RAM disk will reinstall itself and the spooled data will continue printing where it left off. Of course, if your application has not finished printing all its data into the spooler at the time of the reset, only the data contained in the spooler will be printed after the reset.

If the reset occurs during ASCII printing there will be no characters lost and you will be unable to tell where the reset interrupted the printout. Graphics data seems to generally print correctly through a reset except for Alt–Help screen dumps. For some reason, these seem to contain a glitch at the point of the reset. This is unavoidable at this time and the only solution is to refrain from reseting during printouts of screen dumps.

There are two hot key combinations which affect the RAM disk's print spooler. As with the stand–alone CodeHead Print Spooler, you may clear the RAM disk spooler's buffer by pressing RightShift/Enter (that's "Enter" on the numeric keypad, not "Return"). Remember that if your printer has its own buffer, it may continue printing after you've cleared the spooler until its own buffer is empty. You can turn your printer off to clear its buffer.

You can "freeze" the spooler's output by pressing the combination LeftShift/Enter (also "Enter" on the numeric keypad). This allows you to contain an entire printout within the spooler so that you may save it to disk with the Spooler Manager. Pressing LeftShift/Enter will toggle "freeze" on or off each time you press it, depending on the current state of "freeze".

NOTE: The print spooler only intercepts data that is being sent to the parallel printer port. It will not catch data sent to the Atari Laser Printer through the DMA port.

Spooler Manager

The Spooler Manager is a separate program which communicates with the CodeHead RAM Disk's print spooler. It will not work with the stand–alone CodeHead Spooler. The Spooler Manager will run as either a program or a desk accessory. You need only change its filename between SPOOLMAN.PRG and SPOOLMAN.ACC to use it in the desired mode.

When you run SpoolMan (or open it as an ACC), it will show the current size of the spooler and the number of bytes left unprinted within the spooler. If there is no spooler currently installed, the size will be zero and some of the buttons will be disabled but you will still be able to print files if you wish.

SpoolMan allows you to "Clear" or "Freeze" the spooler. These commands provide the same function as the spooler's hot keys and you will see the "Bytes to print:" display change to zero or the "Freeze" button change when the appropriate hot keys are pressed.

There are two additional functions provided by SpoolMan. The most powerful option is the Save function. This allows you to save data contained in the spooler so that it can be printed at a later time (or anything else you want to do with it, such as editing).

The normal procedure for doing this will be to first freeze the spooler (by hot keys or from SpoolMan). Then perform a printing operation. This can be done from any application which outputs to the printer port, including DTP documents, word processing output, text file printouts from the desktop or MaxiFile, and screen dumps.

In order to successfully save spooled data to disk, the spooler must be large enough to contain the entire document. In the case of DTP and other graphics output, there may be more than 100K per printed page. If the

spooler fills up and there is still more data being sent, it will first pause for 30 seconds (as does the system in waiting for the printer to become ready), and then return an error condition to the application doing the printing. The application may or may not inform you of this so your only indication may be that everything has stopped (disk access, progress display on the screen, or whatever indicator the application has). When printing from the desktop or the Spooler Manager, you will get a message stating that the printer is not responding. To recover from a "full" condition, you can attempt to clear the buffer with the hot keys, abort the printing operation by means provided by the application, or reset the computer.

Once the data has been captured by the spooler, you can select the "Save" option and save it using any name you desire. It can later be printed by any method you like, with or without a spooler installed. This is very useful for reprinting graphics files which take a long time for the application to print (because of internal calculations). Once the data is saved as a disk file, it can be printed at the printer's speed instead of the application's speed.

NOTE: MaxiFile's print option is intended for text files and therefore handles tab commands . If you attempt to print a file containing graphics data from MaxiFile, any byte with a value of nine (the ASCII value of Tab) will be interpreted as a tab character, causing a glitch. This will be changed in the next version of MaxiFile, 3.1. In the meantime, it's best to use another method, such as SpoolMan.

The other additional feature of SpoolMan is the ability to print files. This option is available even if there's no spooler installed. Selecting the "Print a File" option allows you to select a file with the file selector. You can then enter the number of copies you'd like printed and tell SpoolMan whether a formfeed should follow each copy.

SpoolMan prints files with no translation whatsoever. In this sense it's similar to the desktop's "print" function, but printing is provided here for convenience. One use for this feature would be to print multiple copies of an address label which has been captured from a DTP document.

If you run SpoolMan as a program with a filename on the command line it will print one copy of that file and exit. This means that you can install it as an application, or just install it in your HotWire menu with a filename on the command line. You can then print a file by selecting that entry in the menu. If you want to install SpoolMan as an application, you can choose any extension you wish to use to denote spooled files, such as ".SM". If you install SpoolMan as an application for .SM files you can then just double-click on an .SM file to print it.

SpoolMan Keyboard Commands

The following keys may be used while the SpoolMan dialog box is on the screen:

P	Print a File
F	Freeze on/off
Clr Home	Clear spooler
C	Clear spooler
S	Save spooler data
Return	OK (Exit)
Alternate	Abort printing (only during "Print a File" operations)

CCP File Loading

CodeHead RAM disk now allows you to specify a CCP file to load rather than be restricted to always loading CODE_RAM.CCP. When you select F5 for CCP file loading, a file selector will appear allowing you to choose any CCP file. Your chosen file will appear at the bottom of the screen and if you save your configuration that filename will be saved with it.

NOTE: When the AUTO folder programs are executed during bootup, the file selector is not yet initialized so you will not be able to specify a filename when CodeRam runs from the AUTO folder. If you select "Load CCP File" while in the AUTO folder, the CCP filename last saved into CodeRam will be used (CODE_RAM.CCP if you've never saved one).

You can also include a CCP filename on the command line. This allows you to have entries in your HotWire menu which will create different RAM disks complete with the desired files already loaded into them. See below for the command line syntax.

Command Line Syntax

There are a few new command line options for dealing with the new features in CodeHead RAM Disk.

If the –I command is followed by "2=", it defines the installation of Drive 2, otherwise Drive 1 will be installed.

–D0 will clear Drive 2 (cause it not to be installed).

–D2x will assign drive letter x to Drive 2.

–D* will install a print spooler as Drive 2.

–L2"FILENAME.CCP" can be used to load any CCP file. The filename must be enclosed within quotes and a full path may be used.

A Help Screen showing the entire command line syntax can now be viewed from within the program by hitting any non–command key from the main screen (such as the space bar). Here is a listing of all command line options:

i	install (if followed by '2=', install #2, otherwise install #1)	
	i	alone, install using the size saved into CODE_RAM.PRG
	ixxx	install where xxx is the number of "K" in size
	is	shrink to fit
	ifx	install a floppy size (x = 1 through 4)
	i2=	install drive 2, followed by any of the above commands
d	drive [A–Z]	
	dn or d1n	set drive #1 to n
	d0	clear drive #2
	d2n	set drive #2 to n
r	remove any existing RAM disk	
c	cold boot	
w	write (save) config	
n	no warnings	
l	CCP loading [0][1][2]	
	l0	don't load a CCP file
	l1	load CODERAM.CCP
	l2"filename"	load the specified CCP file
a	ACC/RAM [r][n]	
	ar	load ACCs from RAM disk
	an	load ACCs from normal location
b	blitter [0][1]	
	b0	don't use blitter
	b1	use blitter
q	quit after executing command line	

Commands must be preceded by a dash ("-") with optional spaces between commands. Commands may be in upper or lower case. Any switches not explicitly set on the command line will use the values saved into CodeHead RAM Disk. Here are some example command lines:

-N-DP-I200-Q	Give no warnings, Install Drive 1 as letter P, 200K and exit.
-d1z-i1-d*-i2=80	Install a 1K RAM disk as Drive Z and an 80K print spooler.
-DB -IS -L"E:\HOTSET.CCP"	Install a RAM disk as Drive B (piggybacked) and load HOTSET.CCP, making the RAM disk just large enough to contain it.

NOTE: Because you may be removing a drive which has the same letter as one that you are installing, CodeRam must allow you to use the command line to set both Drive 1 and Drive 2 to the same drive letter, or to set Drive 2 to an existing drive. It is up to you to make sure that you DON'T do this.

Delay Setting

From the Help screen (See Command Line Syntax above), you can now set the delay time used for automatic installing/bypassing when running CodeHead RAM Disk from the AUTO folder. The current delay time will be shown at the bottom of the screen and you can set a new time by hitting "T". Changing the delay time will enable the F10 "Save" mode so that your new setting will be saved (you can cancel this by hitting F10).

The delay allows you to automatically install or bypass CodeHead RAM Disk when it runs from your AUTO folder so that bootups do not require any user input. If you enter a delay time of zero, the main screen will not even appear (unless you've hit a key before it runs).

TT and Big Screen Compatibility

CodeHead RAM Disk is now fully compatible with the Atari TT030 computer as well as all other large screen monitors (such as the Viking Moniterm).

If you run CodeRam on the TT or any other computer which has "fast RAM", you will have an extra option available. Below the F10 command will be an F11 option for installing in "fast RAM". Hitting F11 will toggle between "Use Fast RAM" and "Normal RAM". Installing CodeHead RAM Disk in "fast RAM" will provide faster RAM disk access.

XBRA Protocol

CodeHead RAM Disk now uses XBRA protocol. This allows it to function correctly with POOLFIX4 installed (see below). CodeRam will also now correctly install if you are using either FATSPEED or POOLFIX3.

Poolfix4 Problem

The popular POOLFIX4.PRG has a bug in its reset routine which prevents CodeHead RAM Disk from reinstalling itself after two warm resets. A patched version of POOLFIX4 is provided in the FREEWARE folder on your CodeHead Utilities master disk. The only change in this version of POOLFIX4 is that a $424 has been changed to $42A.

Bugs Fixed

The state of piggybacked drives is now saved so that you can reboot or install a piggy-backed drive from the AUTO folder without being asked for confirmation. On a warm reset a piggy-backed drive will be selected if it contains AUTO programs, otherwise the existing drive will be selected first.

The F3 preset floppy size will now give you the correct sized drive.

ART GALLERY

Mega STe/TT030 Compatibility & Large Screens

Art Gallery 3.0 is now fully compatible with the Atari Mega STe and TT030 computers, and is also compatible with large screen monitors. If you have a large screen monitor, you can view any type of picture that matches the current resolution, or has the same number of planes but lesser pixel resolution. For example, if you have a Moniterm monitor (or a TT030 in TT High Resolution), you will be able to view any pictures created for ST high resolution.

If you run Art Gallery on a TT030 computer, you will be able to view pictures created by Neochrome, DEGAS, DEGAS Elite, Art Director, TNY pictures, and Prism Paint pictures in any of the 5 "normal" TT screen resolutions, which includes the three standard ST resolutions -- low, medium, and high. This means that you can view any existing pictures in their native resolution, no matter what the current resolution is when you open Art Gallery. (This is vastly preferable to converting pictures between color and monochrome.)

The TT030 vs. Spectrum 512

Spectrum 512 pictures cannot be viewed on a TT030. The Spectrum 512 picture display technique depends on very tight timing; in fact, the timing is tied directly into the number of machine cycles a given instruction will use. But the 68030 chip used in the TT030 has an onboard cache that makes it impossible to predict machine cycles with the necessary degree of accuracy.

Prism Paint Pictures

Art Gallery now supports the picture format created by Lexicor's Prism Paint program. Prism Paint is a very nice full-screen paint program that works in any resolution. Prism Paint pictures from any standard TT or ST resolution can be viewed, and if you have a TT030 you can view pictures from any resolution (except TT high res, unless you have the Atari high res monitor) no matter what the current resolution is. If you have a large screen, you can view Prism Paint pictures created for that screen.

Command Line Support

Art Gallery 3.0 can be "remotely controlled" with input through the "command line". If you wish to use this feature from the GEM desktop, you can install Art Gallery as an application for your most commonly used picture types. (See your computer's user manual for more information about the "Install Application" feature of GEM.)

If you use CodeHead Software's HotWire program-launcher, you can set up customized command lines to build your own slideshows using multiple types of pictures if you desire, or build Hotwire menus containing your favorite pictures, to load and display with a single keypress or mouse click.

If you pass Art Gallery a command line consisting solely of a filename (including the full path information), it will simply load that picture. (Assuming it __is__ a valid picture, of course.)

You can also control Art Gallery's slideshow features with command lines. If your command line starts with a '-' (hyphen) character, Art Gallery assumes that you wish to start a slideshow instead of showing a single picture. Following the '-' character, you may have a number from 0 to 9, which tells Art Gallery how many seconds to pause between pictures.

The number (0-9) is optional; if you don't include it Art Gallery will use its default delay setting.

Another optional parameter can follow the number (or the hyphen, if you don't include a delay time) -- the letter 'X'. If your command line includes an 'X', Art Gallery will display all the pictures in the indicated directory once, then exit. If you don't include the 'X' parameter, Art Gallery will continuously cycle through the pictures in the indicated directory until you abort it manually.

If you use both the delay time and the 'X' parameter, the delay time must come first.

After the optional parameters you should type a single space, followed by the name of the directory that contains the pictures you wish to display. When you pass a command line indicating a slideshow, Art Gallery decides which type of pictures it is going to show by the extension you provide at the end of the directory name.

Here are some examples of valid command lines for Art Gallery:

C:\PICTURES\ALOHA.TNY	Displays the picture ALOHA.TNY
-9 C:\PICTURES*.TNY	Runs a continuously cycling slideshow of all .TNY pictures in the \PICTURES folder on drive C:, with a delay time of 9 seconds.
-2X C:\PICTURES*.PC?	Shows all DEGAS Elite pictures in the \PICTURES folder with a delay time of 2 seconds, then exits after one pass.
-X C:\PICTURES*.PNT	Shows all Prism Paint pictures, with Art Gallery's default delay time, and exits after one pass.

AUTO ORGANIZER

A bug was fixed in Auto Organizer which would cause an error when AUTO folder clusters were located further than 16 megabytes from the beginning of the drive.

FONT TRICKS II

Font Tricks has been made TT-compatible and should now work fine on any Atari ST/TT computer in any resolution (including large screen monitors).

STICK SHIFT

Stick Shift is a new desk accessory designed for handicapped people. Stick Shift changes the function of the shift keys (Left Shift, Right Shift, Control, and Alternate) into "toggled" keys. This means that shift-key combinations can be typed with one hand (or even a mouthstick) by first pressing the shift keys to "stick" them down, then pressing the alphanumeric key, and finally pressing the shift keys once more to release them.

To use Stick Shift, install it as a normal desk accessory or load it into MultiDesk (see manual for more info). Once installed, Stick Shift will display a little symbol showing the current state of the shift keys. This symbol is the same one that's used by HotWire for its shift key combinations. The symbol can be placed at any of the four corners of the screen (see below).

If any of the shift keys are pressed, the appropriate section of the shift key symbol will be blackened until that key is pressed once more to release it.

Stick Shift Dialog Box

To open the dialog box, select Stick Shift from the desk accessory menu or the MultiDesk window. The dialog box contains a box representing your monitor screen with shift key symbols in each corner. You can click on any of the corners to change the location of the symbol. (It will only show up if "Enabled" is selected).

You can enable or disable Stick Shift by selecting "Enabled" or "Disabled". Selecting "Disabled" will clear any keys that are currently "stuck" down and erase the symbol display (this may leave a white hole on your screen).

Selecting "Save" will save the selected screen corner and the state of the Enabled/Disabled buttons. This information is saved directly into STIKSHFT.ACC and you will not be able to save it if you've renamed the accessory file.

NOTE: If you clear Stick Shift from MultiDesk while any shift keys are "down", the system will think they are still down until you manually press them to inform the system.

SIREN

SIREN.ACC is in the FREEWARE folder. It's an fun little accessory which periodically sends a little police car driving across the bottom of your screen, complete with siren and engine sounds.

The automatic patrolling of your screen has been set to once an hour, but there's a separate program called SETSIREN.PRG which allows you to set this time to any number of minutes and seconds. This is especially useful in high-crime areas where more police protection is desired.

SIREN works in any resolution, including large screen monitors and TT resolutions. It also waits until the cops are long gone before making any keypress sounds but keypresses are usually stored until the coast is clear.

NOTE: If you load SIREN.ACC into MultiDesk, you must enable the "TIMERS" button in order to have it drive by at the appointed intervals. But if you have any other ACCs loaded into MultiDesk that also use a timer event, MultiDesk will use the shortest timer event that it finds. Some accessories (sometimes needlessly) use a zero length timer which will result in repeated, continuous police activity. (You can detect an ACC's timer event by holding control while clicking on it from within MultiDesk. It will show evnt_multi with "T" selected.) It's best to load SIREN.ACC into its own menu slot.

Enjoy...and don't get busted.

LEONARD6

Leonard6 is in the FREEWARE folder. It's a replacement for the little bombs that appear when you have a system crash. With Leonard6 installed you'll get little heads with zeros and ones (kind of like our CodeHead logo) instead of bombs. Further documentation of Leonard6 is contained in the FREEWARE folder.

CodeCopy

Users of Stacy computers may have run into problems with CodeCopy. This is due to a serious bug in the AHDI hard disk driver. It has a time-delay auto-parking feature which normally parks your hard drive after about 5 minutes of usage. But if the timeout occurs while the floppy disk is being accessed, both hard disk and floppy disk access will fail. Since CodeCopy spends so much of its time doing floppy disk access, many users have thought that this is a bug in CodeCopy.

Even though it's not CodeCopy's fault, we've added a fix in CodeCopy to solve the problem. If CodeCopy detects the AHDI hard disk driver, it will save the current timeout value and set it to one hour. It continues to reset the timeout to an hour after every CodeCopy command, so there should be no way that the hard drive will park itself. The only way you could experience a problem would be if you let CodeCopy sit unused for 59.9

minutes and then read a disk ‹grin›. When you exit from CodeCopy, your previous timeout value will be restored.

A bug in CodeCopy has been fixed which would sometimes not allow the file selector to come up to save a CCP file.

CodeCopy will now correctly run from the AUTO folder (if you want), although you will not be able to access the file selector for loading or saving CCP files.

CODEHEAD ED

CodeHead ED is our desk accessory version of the popular programmers' text editor, Micro EMACS. Besides running as an accessory, CodeHead ED also makes use of the GEM file selector to make it even easier to use. The documentation for CodeHead ED is included separately as it may not be of interest to all that read this README file.

RSC –› ASM

RSC_ASM will convert a standard Atari resource file (created by either Atari's Resource Construction Set or Hi–Soft's WERCS) into 68000 assembly source code, complete with labels which can eliminate a lot of the drudgery of using GEM from assembly language. This allows you to imbed your resource files directly into your programs and eliminates an extra hassle for your users in dealing with RSC files.

RSC_ASM requires only an RSC file, but if an .H file also exists (created by setting the RCS's output for the C language), its labels will be used with the source code. RSC_ASM creates a text file with an extension of ".R" which can be merged or "included" directly into your assembly source code, either in the "text" segment or the "data" segment.

If you run RSC_ASM without a command line a file selector will appear allowing you to choose an RSC file. If an .H file exists in the same directory as your RSC file, its definitions will be used. You'll then be asked if you want to output to Disk, Printer, or Screen. Output will be the same to all three devices with the only difference being that you can pause screen output with a key or mouse button.

You can also call RSC_ASM with a command line. It expects to receive the name of an .H file (not an .RSC file). This is done so that you can install RSC_ASM as an application for .H files and still install the Resource Construction Set as an application of .RSC files. When RSC_ASM is run with an .H file on the the command line it will automatically assume "disk" mode and will create an .R file in the same directory and with the same name as the .H and .RSC files.

There is a bug in the Resource Construction Set which causes it to sometimes create a tree with more than one "last object" (LASTOB). RSC_ASM catches this and fixes it in the .R file. If this happens, an alert box will appear informing you of the number of "last object" errors it fixed, and that they were not fixed in the RSC file. You can usually fix these errors by simply re–sorting the objects in the offending tree.

RSC_ASM does not require that you "Order Binding File". By turning this switch off in the RCS, the saving of larger files will speed up immensely.

You'll need about 460K of free memory to run RSC_ASM.

RSC_ASM has been tested with the output of both the Atari Resource Construction Set and Hi–Soft's WERCS. The .R file created by RSC_ASM will assemble with no errors under Atari's Madmac and Hi–Soft's Devpac assemblers.

Format of an .R File

We've created a simple resource which contains only a dialog box, an editable string and an OK button. Here's a listing of its .R file created by RSC__ASM:

```
* ---------- Source code conversion from RSC__TEST.RSC ----------

* ---------- Equates --------------------

tree2           =       0       ; tree
exit2           =       1       ; object in tree2 tree
edit2           =       2       ; object in tree2 tree

num__tree       =       1
num__obs        =       3

G__BOX          =       20
G__TEXT         =       21
G__BOXTEXT      =       22
G__IMAGE        =       23
G__PROGDEF      =       24
G__IBOX         =       25
G__BUTTON       =       26
G__BOXCHAR      =       27
G__STRING       =       28
G__FTEXT        =       29
G__FBOXTEXT     =       30
G__ICON         =       31
G__TITLE        =       32

NONE            =       %000000000
SELECTABLE      =       %000000001
DEFAULT         =       %000000010
EXIT            =       %000000100
EDITABLE        =       %000001000
RBUTTON         =       %000010000
LASTOB          =       %000100000
TOUCHEXIT       =       %001000000
HIDETREE        =       %010000000
INDIRECT        =       %100000000

NORMAL          =       %000000
SELECTED        =       %000001
CROSSED         =       %000010
CHECKED         =       %000100
DISABLED        =       %001000
OUTLINED        =       %010000
SHADOWED        =       %100000

                even
```

```
* ---------- Ted Info Structures --------------------

eedit2:
        dc.l    str1,str2,str3
        dc.w    3,6,0,$1180,0,-1,7,12

* ---------- Object Trees --------------------

ttree2:  ; (tree 0)
obj0:
        dc.w    -1,1,2,G__BOX,NONE,OUTLINED
        dc.l    $21100
        dc.w    0,0,18,6
oexit2:
        dc.w    2,-1,-1,G__BUTTON,%00000111,NORMAL
        dc.l    str0
        dc.w    4,3,8,1
oedit2:
        dc.w    0,-1,-1,G__FTEXT,%00101000,NORMAL
        dc.l    eedit2
        dc.w    3,1,12,1

* ---------- Strings --------------------

null:   dc.b    0
sexit2:
str0:   dc.b    "OK",0
sedit2:
str1:   dc.b    "_____",0
str2:   dc.b    "EDIT:_____",0
str3:   dc.b    "X",0

        even
```

If you study this text you'll learn several things. It begins with equates for your tree and object numbers. These are the equivalent of the #define statements in an .H file and in fact are taken directly from the .H file.

Next are two equates for the number of trees and objects: num__tree and num__obs. The constant "num__obs" is particularly useful for "fixing up" your resource file. The GEM call "rsrc__load" will load a RSC file and adjust all of the coordinates for the current resolution. Since you are including your resource directly in your source code, you'll need to make calls to "rsrc__obfix" to adjust your objects. The following code will fix all objects in your .R file:

```
        move.w  #num__obs-1,d5
        move.l  #obj0,addrin
fixloop:
        move.w  d5,intin
        bsr     rsrc__obfix
        dbf     d5,fixloop
```

(You'll need to create your own subroutine for calling rsrc__obfix).

RSC__ASM Labels

The key to RSC__ASM's power is the fact that you can access your objects and strings without using rsrc__gaddr (resource get__address). This is possible because RSC__ASM creates labels from the object name by prefacing it with various letters. The letters used are as follows:

t	tree address
o	object address
e	ted__info address
s	string address
i	image address

This means if you give an editable text object the name "name", you'll find its object structure at address "oname", its ted__info structure at "ename", and its p__text string at address "sname". Likewise, the tree address of a dialog box named "mainbox" can be found at "tmainbox" and the image data of an icon or bitblock named "logo" can be found at address "ilogo".

This setup gives you some powerful options. For instance, you can deselect an object named "okbutton" in one instruction:

```
bclr    #0,ookbutton+11
```

You can change the text of a string object named "function" by pointing it to another string:

```
move.l  #newstring,ofunction+12
```

You can change the text of a ted__info named "editstr" by pointing its p__text to another string:

```
move.l  #lastentry,eeditstr
```

Or you can easily look directly at the text of a ted__info named "town":

```
lea     stown(pc),a0
```

Unnamed objects, strings, and ted__infos are given sequential number, both for your convenience and the efficiency of RSC__ASM. Many times an address may have two labels in order for you to be able to access the label by object name while RSC__ASM uses the sequential number. See above, where object edit2's structure oedit2 points to ted__info eedit2 where the p__text points to str1 which is the same address as sedit2.
The output of object data from RSC__ASM is shown in many different forms. The forms are chosen for the best readability. The "next", "first", and "last" children are displayed in decimal (with a −1 where appropriate). The object types, flags, and states are displayed using equates when possible or bit fields when more than one flag is set. The object specification will be either a label pointing to another address, or a hexadecimal long word. The x, y, width, and height are shown in decimal if less than 256, otherwise they are shown in hexadecimal so that the upper byte (pixel adjustment) can be easily viewed.

NOTE: All zero−length strings will point to a single zero with the label "null." You should be aware of this if you decide that you want to change any of them. All p__valid strings which use a string of repeated characters are represented as a single character followed by a null terminator:

XXXXXXXXXXXXXXXXX,0 = X,0

999,0 = 9,0

AAAAAAAAAAAAAA,0 = A,0

This is done to save memory and it does save a considerable amount especially if you have many long, editable strings. This practice works on all known TOS versions and although it is not documented to behave in this fashion, it will probably not change in the future.

Source Code

RSC_ASM is written in GFA Basic and although we have not released the source code publicly, we will consider doing so on an individual basis if requested. We are also open to suggestions for improvements.

SUPPORT AND UPGRADES

Technical Support by Mail

We are very busy at CodeHead and would appreciate it if when writing to us with problems you would please include your phone number. It is much easier, faster, and costs us less money to pick up the phone and call you with the solution, than to take the time to compose a letter of response.

Many letters do get answered (in due time), but you will have much better chances of getting a speedy response if you include your phone number. Thank you.

CODEHEAD Software Update Policy

At CodeHead, we update all of our software frequently; but due to postal expenses we are not able to notify our users regularly. You may find out the latest version number of any CodeHead Software product by calling us at the number below, or by contacting us on GEnie, Compuserve, or Delphi. The latest update for any product may be obtained by returning your original master disk plus $10.00 to:

CodeHead Software

P.O. Box 74090

Los Angeles, CA 90004

Voice: (213) 386-5735

FAX: (213) 386-5789

BBS: (213) 461-2095

From time to time we offer free updates for certain versions (if we've released something with bugs, etc.). You will be informed (or your money returned) if this is the case. If an update requires a manual there will be an extra charge. Prices are subject to change without notice.

****

Are you in possession of stolen mechandise? All of our software has hidden serial numbers which can be traced back to the purchaser. If we find copies of our software in circulation, we will take steps to prosecute anyone in possession of such a copy as well as the original purchaser of that software. You can protect yourself by not lending our software to anyone or making any copies which might fall into someone else's hands.

Please, remember that software theft hurts EVERYONE. If you have a legitimate copy of our software, please accept our sincere thanks for purchasing our product.

If you have accepted an illegal copy of our software (you know if you have), you might want to give some thought to the consequences of your actions. We will not continue to produce software for the ST if we can't make a living at it. If you steal a copy of any of our programs (by using it without buying it, or allowing others to use it without buying it), besides breaking federal and local laws and leaving yourself open for criminal prosecution, you're also quite literally stealing the food right out of our families' mouths and helping to drive us out of the Atari marketplace.

Think about it.