HANDY RELEASE 0.5
15 November 1988

CONFIDENTIAL and PROPRIETARY

This document describes the V 0.5 release of Handy development environment. It includes these sections:

- Handebug V1.33
- Handycraft V1.28
- Handypic V0.1
- Revisions to 6502: Directory
- Miscellaneous
- What You Must Do To Start Using This Release

In general, thanks to everyone for torturing us endlessly with the endless bug reports. Who rights this crummy cold anyway?

Handebug V1.33

Fill Mem is now implemented.

Note: don't fill from $F000 up, as you'll fill right over the monitor code. Also, don't fill from MONITOR_ZP_RESERVED, currently $F0, to $FF. And finally, for this release don't FillMem the stack. Next release it will be OK to do so. One day when the debugger hardware board is finished the monitor won't use the stack or any other Handy RAM at all so you can FillMem to your little heart's content.

By popular demand, to move the cursor to a new Handebug field you must now position the pointer over the field and click with the select button. The pointer still disappears when you start typing.

Handebug now enjoys a faster upload technique as suggested by Mike.

Bug fix: now refreshes all fields, including GO and Breakpoints, when you hide Handebug and then open again.

The Handebug Hidden window is smaller, with teeny weeny little gadgets, and the position and size of window is remembered and restored the next time you hide Handebug. So now take a hike all jerks who bugged us mercilessly about this.

Structure editor, first cut. Yeah! This one allows you to type in the address and structure ID of any structure of your making.

Congratulations, John. It's a good job.

You can edit any data field. Note that if you double-click in any field defined as a pointer to a structure, the structure editor displays that structure.

The structure definition file can be found at s:handebug.defs, which with this release contains only the Handebug structure editor's definition of a Sprite Control Block structure. You can define your own structures by adding them to the s:handebug.defs file using the rules defined in the structure editor listing that is included in the hardcopy of this release. Structures you define have ID numbers that start at $10 and go up to $FF. $00 is an error ID number. The system reserves $01 through $0F. The Sprite Control Block is type $01.

When you create a structure definition, you ought to try to keep the window completely within the upper 320x200 rectangle of the Handebug display. This is because tomorrow we're

going to allow the structure window to open on the Handycraft display, the dimensions of which are - guess what! - 320x200.

While you're not allowed to modify the system structure definitions, any system structure, such as the Sprite Control Block, can be legally *expanded* by you . The allows you to create your Expanded SCB structure without recreating the entire underlying SCB structure. To expand an existing system structure:

- leave the existing definitions alone, except that you can make Y coordinates smaller in order to pinch out the white space and pack more data into the window
- add new definitions below existing definitions by making the window larger
- position the window anywhere you want, except that the window must open and fit completely within the upper 320x200 rectangle of the Handebug display

Handycraft V1.28

Bug fixes:

- When editting, the Back Sprite command no longer wipes out all other sprites
- When in edit mode, any HOFF and VOFF values uploaded during emulation mode are now ignored
- Single-color sprites should work OK now
- Selecting a sprite while in edit mode should work correctly now
- The Out-of-Memory/Couldn't-Open-File confusion is fixed now once and for all, really, honestly, would I lie?
- The sprite color-redirection problem is fixed. You should be able to redefine any image color to any display color with wild abandon

There is now real sprite data output. You can write out sprite data three ways:

- 'O' is used to output image data of the currently-selected sprite
- 'w' is used to write the SCB and the image data of the currently-selected sprite
- 'W' is used to write the SCB of the currently-selected sprite

Also, if all you want is to find out how many bytes are required by the imagery of the current sprite, you can use the 'I' command.

Handycraft now writes out an extra byte of zero at the end of each data line as now required by the Handy hardware (see the new Handy Spec, page 19, for an explanation).

When writing the data for small sprites, you should check whether COMPACTED or COMPLETELY LITERAL creates the smallest amount of data. Surprisingly, there are times where it's more efficient to create COMPLETELY LITERAL data rather than COMPACTED. To test, edit the sprite, select to edit the particulars, select COMPACTED, exit the editor and then use 'I' to see how many bytes your data will take. Then edit the particulars again, select LITERAL, exit and use 'I' again.

Output your current editor display environment using 'o'. Input a saved environment using 'i'.

Emulation display is now 102 lines tall. Used to be 100 for very distant, very obscure reasons.

The Edit screen (the short one at the bottom of the Handycraft display) is now not created until it is needed. This is good because several of you found the screen to be a nuisance. The down-side of this is that trying to edit a sprite now may result in Out-Of-Memory error. Don't forget that you can use 'f' and 'F' commands to get back memory.

If a sprite is defined as BACKGROUND, all pens write into the display (nothing is transparent). If a sprite isn't BACKGROUND and you've selected that Pen-F is transparent, Pen-F will now be as transparent as air.

Sprites used to have 3 bits with which you could select if the sprite was BACKGROUND and/or COLLIDING and/or PENF-TRANSPARENT. Now in Handy hardware reality those 3 bits are used to define 8 *sprite types*, which types are described on page 14 of the Handy Spec and include 2 new sprite features: XOR sprites and sprites where Pen-E can be made opaque and non-colliding. These new types are not yet supported by the Handycraft emulator; next release, for sure.

Also not yet supported:

- Palette reload, HVSize/Stretch/Tilt reload control flags
- Stretch and Tilt
- Hardware collision
- "Start drawing up" and "Start drawing left" control flags
- READROM macro

Handypic V0.1

First release of Handypic, which can be found in HANDY:. Sorry, but I'm too tired to get those extra features in. Right now, Handypic cuts out the top-left 160 x 102 rectangle from the iff file that you specify as an argument. Takes iff_filename and creates iff_filename.src, a file that contains the palette and display data of a picture. Cut the palette into a separate file and then change a copy of 6502:examples/video.src to include your new file. Assemble it with HANDYIO defined, take the .bin into the lab, download on Handy hardware, hit go and voila.

Next release of Handypic will allow you to cut up the picture into mutliple rectangles, and will have an option to create a whole display .src program and also to create directly a .bin file for you. Real soon now.

Revisions to 6502: Directory

New file, 6502:examples/video.src, which you use in conjunction with Handypic to create a still picture on the Handy display from any old IFF ILBM file.

The Apple keyboard routines used to give you info only if a key was currently being pressed when the routine was called. Now if a key was pressed and released between calls to the routine you will now get data about the key that was pressed, thanks to some logic by Larry.

There's lots and lots of new hardware register definitions and register flag bit definitions in 6502:include/harddefs.i.

New macro file, 6502:macros/zpage.mac, which has macros that must be used now when declaring zero-page variables. Logic and text mostly by Mike. The system code needs these macros, and you must use these same macros. I've including hardcopy of zpage.mac with this release for your perusal.

The monitor now doesn't upload to the emulator sprites that have the SKIP_SPRITE flag set.

New file, 6502:src/sysdata.src, which declares any data that is needed by the system and emulator routines. You must include this file in your source code before including any other 6502:src file.

Handy math is now emulated using the 6502:macros/handymath.mac macro file and the 6502:src/handymath.src source file. The logic for the math routines was carved out by Chuck - good job, Chuck - and expanded upon with comments added by RJ. An example of using the math macros can be found in 6502:examples/testmath.src. Hardcopy of that file is included with this release.

The math emulation currently allows you to set the SIGNMATH and ACCUMULATE flags in the SPRSYS register. However, the result flags overflow/divide by zero and carry aren't implemented yet. Next release, honest.

To use the math you must do this:

- Call RESETMATH to initialize the registers
- If you want to accumulate the results of multiplies, you can either preset the accumulator or use the CLEARACCUM macro to reset the accumulator
- Set any hardware math register directly, except for MATHA and MATHE which respectively trigger the multiply and divide hardware automatically. To write to either of these registers use the macros STA_MATHA, STX_MATHA, STY_MATHA, STZ_MATHA or STA_MATHE, STX_MATHE, STY_MATHE, STZ_MATHE.
- You might want to call WAITMATH to wait for the results, or you might want to go off and do other stuff while the math hardware churns

Miscellaneous

I apologize for the fact that lots of information is being thrown at you this way. It's got to be hard for you to wade through so many changes. Soon I'm going to release a new version of the programmer's doc, which will be up to date and will have an index for easy reference to the mounds of information that's coming available. Hang in there.

You get new copies of Handy Specification and Handy Appendix 2. Look for the vertical bars along the left edge of the pages; these mark where text is new or changed.

You get a Handy display sketch sheet, with nine rectangles that approximate the Handy display. Ruler along the bottom is a pixel ruler.

I have a question. The math routines currently declare their own copies of MATHA - MATHP registers. I would rather use the real register locations even when running on the Apple. This would mean having the Handy hardware registers live at Apple RAM starting at $FC00. Is there any reason why we can't do this? Someone answer please.

There's a hardware register named SPRSYS which has bits that mean different things depending on whether the register is read or written. The implication of this is that if one routine of your code wants to set or clear a specific SPRSYS flag, for instance a multiply routine that wants to set the SPRSYS SIGNMATH bit, it can't simply LDA SPRSYS, set the flag and STA SPRSYS. Instead, you need to keep a RAM copy of SPRSYS around which you always modify whenever you want to modify SPRSYS.

To support SPRSYS and any other hardware registers that behave this same way, I have created a convention. The RAM copy of any register will have the name of the register with a "_RAM" suffix. Any RAM copies of registers such as this will be declared in the 6502:src/sysdata.src file.

It's important that you follow this convention, as the system code is depending on it.

Using SPRSYS as an example, to set a flag in SPRSYS you should now do this:

- LDA SPRSYS_RAM

- ORA #SIGNMATH
- STA SPRSYS_RAM
- STA SPRSYS

What You Must Do To Start Using This Release

First, this release breaks your software again so before I do anything you must give me permission to install the new release on your system. There are several minor changes I'd also like to make to your A2000, all of which should be benign. Please be sure in advance that I can make these changes to your system:

- I want to add C2: to your startup-sequence assigns
- I want to make these changes to your SYS:C2 directory:
  - Add tsize, an application that tells you the number of bytes found in a directory
  - Give you a new copy of the goodshow program
  - Update the readme file
- I want to distribute a new file, s:handebug.defs, which has the Handebug structure editor's definition of a Sprite Control Block structure

You must include 6502:macros/zpage.mac before any zero-page memory allocations and before any 6502:src files are including in your program. Also, if you make zero-page memory allocations, you must surround these allocations with the macros described in zpage.mac (I've supplied you with hardcopy of this file).

You must include 6502:src/sysdata.src in your program before any other 6502:src file is included.

You probably have to set some sprite Control0 and Control1 flags if you haven't been doing so already. The old testsprite.src program didn't set any sprite Control flags. The current testsprite.src does set some flags.