HANDY RELEASE 0.4
27 October 1988

CONFIDENTIAL and PROPRIETARY

This document describing the V 0.4 release of Handy development environment includes these sections:

- Handebug V1.30
- Handycraft V1.25
- Revisions to 6502: Directory
- What You Must Do To Start Using This Release

Handebug V1.30

Handebug now talks with Handycraft, allowing you to enter Handebug function-key commands from Handycraft.

New function key definitions. See your new function-key template for details. The template is number V1.29.

If your .bin file is newer than a matching .sym file, your symbols now are thrown away. This means that if you assemble with the +s option and then download your file, causing the symbols to be loaded, and later assemble without the +s option and then download, this second download will correctly cause the old symbols to be discarded, leaving you correctly in a symbol-free state. Note that you can throw symbols away also by using SHIFT-F9.

The GO field now has an address that you can GO at. This address is set every time you download a file that has an assembler .RUN directive in it, just like the way the PC is loaded for you. The difference between the PC and the GO address is this: when you use CONTINUE, you begin executing at the current PC. When you use GO, you begin executing at the GO address. This will be especially useful for restarting your program from its initial address regardless of where it was executing when you stopped it.

Handycraft V1.25

When you ask Handycraft to load a sprite image file that doesn't exist, no longer will you be tormented by the ill-begotten *Out of darn memory!* error. Now Handycraft tells you *filename: couldn't open* and then skips the rest of your sprites. Really. Honest. Would I lie?

Handycraft now talks with Handebug, allowing you to enter Handebug function-key commands from Handycraft.

The 't' command no longer toggles the title bar. The title bar now is always there, whether you can see it or not. This means that you can always drag and depth-arrange the Handycraft screen whether or not you can see the gadgets. Also, the Handycraft is no longer frozen in the presence of the title bar. In fact, Handycraft now just ignores the screen's title bar altogether.

The Handy display window positioning hardware registers HOFF and VOFF are now emulated, using the 6502 macros HOFF8, VOFF8, HOFF16 and VOFF16 to get the instructions up to Handycraft.

The sprite control flag SKIP_SPRITE is now supported.

The sprite control flags HFLIP and VFLIP are now supported.

Revisions to 6502: Directory

Changed controls.src so that the keyboard macros will read the Apple keyboard correctly.

Changed sprite.i to the new sprite control block offsets, including adding the fields SCB_TILT and SCB_STRETCH. You must change your sprite control block definitions in order for your code to work correctly with the new release.

Changed sprite.i so that it now ends with a field named SCB_SIZEOF which constant contains the byte-size of the sprite control block (this constant was formerly named SCB_SIZE). If you are defining your own extended sprite control block, you could use SCB_SIZEOF to create SCB offsets that will track any future changes to the sprite control block (and there *will* be changes), like this:

```
MY_SCB_DATA0      .EQU  SCB_SIZEOF
MY_SCB_DATA1      .EQU  (MY_SCB_DATA0+1)
MY_SCB_SIZEOF     .EQU  (MY_SCB_DATA1+1)
```

By popular demand, the SPRITES and PSPRITE macros can now be passed either the address of the first SCB to be rendered or the address of a 16-bit variable that contains the address of the first SCB to be rendered. You describe which you are passing to the macro by passing a second argument to the macro, the *indirection* argument. The indirection argument can be either 0 or 1. If 0, you are declaring that the first argument is an absolute address of an SCB. If the indirection argument is 1, you are specifying that the first argument is the address of a variable that contains the address of your first SCB. So now, what used to be:

```
            SPRITES     TestSprite
      ; Following is a definition of a sprite control block
      TestSprite        ...
```

is now:

```
            SPRITES     TestSprite,0
```

and may also be:

```
            LDA         #<TestSprite
            STA         FirstSprite
            LDA         #>TestSprite
            STA         FirstSprite+1
            SPRITES     FirstSprite,1
      FirstSprite
            .DS         2
      ; Following is a definition of a sprite control block
      TestSprite        ...
```

In 6502:examples is applesound.src, a bit of a file from code by Peter that makes the Apple go woop-woop or beep or so depending on the value you have in A. Currently supports values from 0 to 2. Have a look, come up with your own sounds, what the heck. The only problem with this routine is that your Apple hangs while it's making your sound, and in the absence of an accelerator board (as is the environment of yours truly) the sounds last quite long.

harddefs.i now has the sprite control bits defined. Handycraft doesn't support all of these bits yet, but it does support a few.

New macros, HOFF8, VOFF8, HOFF16 and VOFF16, which you use to set the position of your display window within the display world. You must use some combination of these macros at the start of your program, or else the display window will be in an indeterminate position.

The testsprite.src file in 6502:examples now shows the new SCB layout, usage of the HOFF8 and VOFF8 macros, and the new technique required by the SPRITES and PSPRITE macros.

What You Must Do To Start Using This Release

Because this release will really break everyone's code, I won't install it on your system until you ask me to, so the first thing you must do to use this release is to ask me to install it on your system.

You must change your sprite control block data declarations to match the new definitions. If you don't, uploading to Handycraft will result in many unpleasant effects.

You must change your calls to the SPRITES and PSPRITE macros so that you pass these macros a second argument describing whether you are passing a pointer to an SCB or the address of a variable that contains the address of your first SCB. See the section *Revisions to 6502: Directory* above for details.

You must use either HOFF8 or HOFF16, and VOFF8 or VOFF16, at the start of your program before the first display is created.