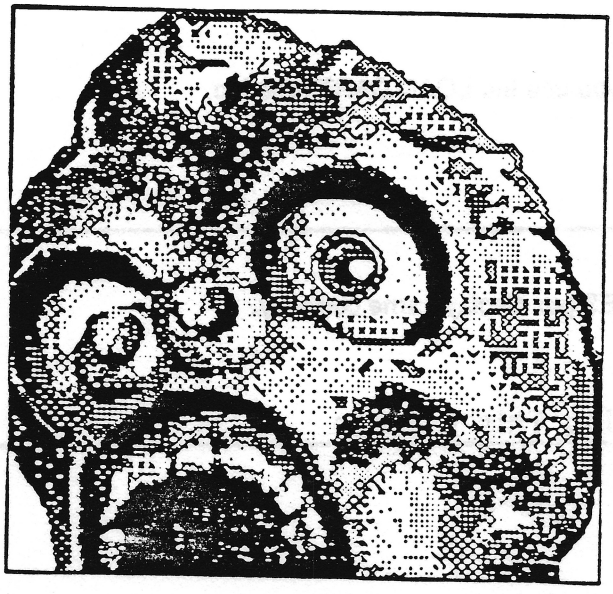
HANDY RELEASE 0.8
2 May 1989

CONFIDENTIAL and PROPRIETARY



This document describes the V 0.8 release of the Handy development environment. It includes these sections:

- HMUSIC and New HSFX
- CART I/O System and HandyROM 0.06
- Other 6502: Changes
- HandyAsm V1.01
- AsmStrip
- GREP
- Handebug 1.57
- Handycraft 1.57
- What You Must Do To Start Using This Release

You want a release?
I got your release right here!

There's a lot of stuff here. Take the time to read it all carefully, really.

HMUSIC and New HSFX

The HMUSIC driver now is released. See the document HMUSIC for a description and examples of usage. Also, 6502:examples/testhmusic.src shows you everything you need to do to play a song or two. To learn more about HSPL, the Handy Sound Programming Language (our own version of Epyx' SPL), refer to the HSPL document distributed with this release.

For your info, a while ago as a test case I added music to Jet (which already had HSFX). It worked just fine, without any display tearing. Also, I got an estimate of the sizes involved: the HMUSIC code was something like 1600 bytes big, and the Liberty March (Monty Python theme) song data was 2200 bytes big.

VERY IMPORTANT: the HMUSIC and HSFX drivers have been rewritten to be "lower-priority" interrupts. Any other interrupt handler will now be able to run regardless of whether music and sound effects are being created. The upside is that the display double-buffering that occurs during the end-of-frame interrupt handling will no longer be held up by music and sound effects, thereby (poof) getting rid of the display shudders that we've seen. Also, the RedEye code will be able to run without risk of losing data as a result of this change. But this "feature" isn't without its downside: heavy use of other interrupts can cause music and sound effects to glitch a bit.

We now suggest that every sound effect and every song voice get a unique priority number, with music getting the even and sound effects getting the odd priority numbers. If they don't get unique numbers, HMUSIC and HSFX can get headaches and heartburn.

Also, the most recent versions of the music tools HSPL and HKCS are being distributed with this release to make sure that everyone is up to date.

CART I/O System and HandyROM 0.06

Here's the first release of the CART system code and the HandyROM cartridge image builder.

To use the CART support macros and code, you must include cart.mac, cart.src and cart.i. Also, there's lots of stuff you have to know, and many things you must define. Read the document "CART - Game Cartridge I/O System" for complete details.

Currently the CART system works with only the cartridge 0 port, which means that currently the CART macros can handle only read-only carts that are a maximum size of 1M bytes. Sometime in the future port 1 will be supported too, and writing to the cartridge will be formalized.

Very important: now more than ever it's important for us to keep the sense of the team. We've never used this game cartridge stuff before, it's going to be complicated to integrate it into our code. Please take notes about the techniques you evolve so that we can meet later in the week and compare notes. It will be extremely valuable to share with one another our experiences and the tricks we discover.

In this release of CART, there are these restrictions:

- You should leave the cartdefs.i definitions alone. You're supposed to copy this file into your directory and then edit it to suite your needs. Don't edit it for now, just accept the defaults.
- Don't even think about calling any of the CART routines directly. Go through the macros only. The routines are guaranteed to change around significantly over the next several sub-releases, so do yourself a favor and do the right thing.
- The ability to preload the directory into RAM and then reference it there isn't supported yet, but it will be real soon now, honest.

Other 6502: Changes

New system macro and source files. You now must include the macro file 6502:macros/sys.mac and the source code file 6502:src/sys.src. Also, before anything else in your program, you should invoke the macro INITSYS. See any of the source files in 6502:examples for an example of this.

Serial registers IODAT and SERCTL have bit definitions in harddefs.i.

New display macro: SETDISP_75, for setting your display rate to 75 Hertz.

If you want to detect the current state of FLIP, the correct way to do it is to read the DISPCTL_RAM variable and AND it with #DISP_FLIP. If the result is non-zero, then the display is currently flipped.

There's a new macro, INITINT, which allows you to define an interrupt handler that overrides the monitor's interrupt handler. This allows you to take over all of Handy RAM once your program starts running because once you control the interrupts it's safe to wipe out the monitor completely ... but guess what, no more debugging!

Anyway, sooner or later when we start running on the Howard board we'll all have to start using this macro, but while we're still on the prototypes you can skip this if you want.

The INITINT macro expects two arguments: the address of a 16-byte buffer that will be used for the interrupt vectors, and the address of an RTS instruction in your code. INITINT, among other things, sets up all vectors in the table to point to the RTS. If you use interrupts, you should set the address of your code into your table after the INITINT call. Also, you should do

INITINT before invoking any other system macros (such as INITEOF) which might connect with the interrupt handler.

See the 6502:examples/testint.src file for sample usage.

There is now a mechanism for having the end-of-frame handler count end-of-frame events for you. The new code will play with a variable named DisplayFrameCount for you, in one of two ways depending on whether you've defined FRAMECOUNT_UP or FRAMECOUNT_DOWN.

If you want DisplayFrameCount to increase with every frame then you should define FRAMECOUNT_UP. DisplayFrameCount will be incremented every end-of-frame.

If you want DisplayFrameCount to decrease with every frame then you should define FRAMECOUNT_DOWN. DisplayFrameCount will be decremented every end-of-frame until it reaches zero. Once it reaches zero it will no longer be decremented. This allows you to hang in your WAITEOF loop until DisplayFrameCount hits zero, and if one time your logic goes too long you don't have to worry about DisplayFrameCount looking like 255 or something.

Note that if you define neither count constant then no extra code is added to the end-of-frame handler.

The game button definitions PAUSE_BUTTON and FLABLODE_BUTTON are gone. Now there are two new GETJOY button definitions: OPTION1_BUTTON (also named RESTART_BUTTON) and OPTION2_BUTTON (also named FLIP_BUTTON). Also, there's a new GETSWITCH definition: PAUSE_SWITCH.

You should support the new PAUSE_SWITCH. Also, you should implement these combination presses: if PAUSE_SWITCH and RESTART_BUTTON are both pressed then restart the game; if PAUSE_SWITCH and FLIP_BUTTON are both pressed then flip the display by using the FLIP macro.

The IODIR and IODAT hardware register bit definitions now have names that reflect their current reality, not the distant distortions.

In 6502:include/monitor.i, the monitor's zpage-usage constant MONITOR_ZP_RESERVED is changed from $F0 to $F8. This frees up 8 zpage bytes, which hopefully will hold you until we get up on the Howard boards.

Bug fixes:

- Local labels should work correctly, and should generate error messages correctly. This is in answer to *many* different bug reports
- The comments after the RELOAD_ sprite definitions in harddefs.i now say "reload" rather than "reuse." Sorry, Chuck
- The HPRINT source code now sets MATHC to get around the hardware bug, and the sprites in the 6502:examples/testhprint.src file are now non-colliding
- SETDISP_50 PCOUNT value is changed from $32 to $31

HandyAsm V1.01

Sorry, you guys missed the 1.00 release. Here's 1.01 for you.

Performance improvements, 5 - 10% depending on your code.

New "rule" that formalizes current behavior of HandyAsm: you can define and redefine a constant as many times as you like using .= (aka .SET), but once you use .EQU to define or redefine a constant the constant becomes fixed and cannot be redefined.

This allows us to create system constants with default values that can be safely redefined by the programmer. For example, the HMUSIC include file now does this:

```
        #IFNDEF HMUSIC_CHANNELCOUNT
HMUSIC_CHANNELCOUNT .= 4
        #ENDIF
```

If you have defined HMUSIC_CHANNELCOUNT before you include hmusic.i, then the constant won't be redefined. If you haven't already defined it, then the constant is created with a default value, and it's allowable for you to define it using .EQU with some other value after including hmusic.i.

Several programmers have reported that using -X to get a cross-reference of their code has pointed out dusty corners with code no longer used. You might give it a whirl and see if there's excess out there that could be trimmed.

AsmStrip

This is a little utility program in HANDY: that strips assembler output data packets, generating a raw binary file for you.

GREP

A new and improved GREP has been copied to your C2: directory. A copy of the GREP documentation is distributed with this release.

Handebug 1.57

Faster parallel port operations, really really faster, you bet. Easily 5 times faster than before, maybe as much as 10 times faster. Lots faster, get the idea?

Now when the display flashes at you there's a text message printed at the CLI too, so you can go see what Handebug is screaming about.

New function keys implemented to support the CART system: Shift-F4 is the LOAD CART function, and Shift-F5 is the BOOT CART function. Refer to the CART documentation for details about what these new functions do for you.

Currently, LOAD CART doesn't give you any options regarding where to load your file into the game cartridge. The rom file you specify gets loaded to the cartridge starting at page 0, offset 0. The fancier features will be coming real soon now, honest, would John lie?

Handycraft 1.57

New version number.

What You Must Do To Start Using This Release

You must support the new button redefinitions.

You must include 6502:include/sys.i, 6502:macros/sys.mac, and 6502:src/sys.src. Also, the very first thing your code should do is call the INITSYS macro, like this:

```
            .RUN    Start
Start       INITSYS
```