Caveat: Just because it's documented doesn't mean it works that way.

About Handebug

Handebug is a debugging tool specifically written for the Handy project. It was designed to allow downloading programs and data files, examining and changing memory and registers, breakpointing and stepping through code, and uploading and saving memory. It will also communicate with Handecraft, providing a limited emulation of the Suzy display hardware.

Handebug tries to provide a continuous display of important data. The display screen includes the current register contents, a list of current breakpoints and their status, selected memory locations and their contents, and a data display window that can be switched among CODE, DATA, SYMBOLS, and TRACE RAM. When Handebug is initially loaded all memory and register displays are zero. As soon as communications is established the register display is updated and, if the display window is set to CODE or DATA, will update the currently displayed page of memory.

When the program under development is running Handebug is not able to communicate with Handy. The display is normally not updated in this case. However, it will respond to special "slave mode" requests made by the software under development, allowing updating of specific displays at program controlled intervals. These "slave mode" requests are also the mechanism used to implement the Suzy emulation.

Handebug is also designed perform selected updates of the display at regular timed intervals by temporarily interrupting the running program to request the specified data. Currently this is not enabled because of potential problems with slave mode requests.

Getting Started

Handebug can only be run from the CLI. It is still under development, and occasionally debugging information may be printed in the CLI console window. To load Handebug, type "Handebug" or "Run Handebug" at the CLI prompt. Either method will work, although the second one is recommended since it will run the debugger as a background CLI task, leaving the CLI free to do other things. If you don't "Run" it, the CLI will be tied up executing Handebug, forcing you to open yet another CLI window to do other things, like editing or assembling files, and windows require screen (CHIP) memory which is at a premium when performing Suzy emulation. In general, the fewer windows open the better.

Handebug takes over the parallel port and opens a custom screen on the Amiga display. The screen can be pushed to the back or slid up or down using the drag bar (title bar) and positioning gadgets at the top of the display. This allows easy access to the WorkBench screen and the CLI windows, and to Handycraft's screen. At the upper left corner of the Handebug screen is a close box. To exit Handebug simply select this gadget with the pointer by pointing and clicking on it with the left mouse button.

Once the display is up, Handebug attempts to communicate with Handy. If Handy isn't talking yet there may be a delay of up to 3 seconds while the communications times out. At this point the debugger is functional.

If Handebug doesn't run there are a few things you can check. To initially load, Handebug requires about 40K of CHIP memory and about 100K of FAST memory. If there is not sufficient memory it will curently just quit. Once it is running you can shut down the custom screen and recover all but about 4K of the CHIP memory. Also, Handebug needs the parallel port. The current Amiga printer drivers hang onto the port, even if you aren't printing. Once you've used (or tried to use) the printer the port is tied up forever. To get around this you must reboot the Amiga.

The Handebug Display

Handebug initially creates a display on a custom screen, which seperates it from the Workbench and other screens. This has the benefit of not cluttering up the Workbench display, which will probably be cluttered enough with your own windows for editing, assembling, or whatever. Unfortunately, this also requires an additional 32K of precious CHIP ram. If the display isn't needed, this can be recovered by "hiding Handebug", as described later. Certain fields can be selected by clicking on them with the mouse. These are refered to as "transmitter" fields, and their contents are copied to the Select field immediately to the right of the "Go" gadget when they are selected. Other fields are refered to as "receptor" fields. These fields receive the contents of the Select field when they are selected.

The display screen consists of Memory Watch fields and Breakpoint fields on the right, a general-purpose data display on the left, a register content display at top-center, a set of fields to control strucure displays at the bottom right, and a set of operation gadgets in the middle. The data display has gadgets at the bottom that can be used to switch it among code, data, symbols,and trace-ram.

The Memwatch fields are used to monitor specific memory locations in Handy. The fields may be one or two bytes and may be displayed as a single byte, two individual bytes (low-byte, high-byte), or a 16-bit value (high-byte/low-byte). The address portion of the Memwatch fields is a receptor field, and if empty will receive the Select field contents when selected. It can also be manually edited, and can contain any valid address or defined symbol. The data portion is updated each time normal (not "slave") communications is re-established with Handy. This is a transmitter field, and can be used to select address data for other purposes. Also, the data field can be edited, and can contain any valid hex data. When the field is edited the corresponding memory in Handy is updated with the new value.

The Breakpoint fields are used to set strategic breakpoints in Handy memory. These fields are receptor fields and will receive the contents of the Select field when selected. The address field can be manually edited, and can contain any valid hex address or defined symbol. To the right of the address field are two gadgets, Enable and Clear. The Enable gadget is automatically highlighted as soon as you enter or edit the address data. This indicates the breakpoint is enabled, and will be placed the next time the program is executed. Selecting this gadget will toggle the enabled status of the breakpoint, allowing you to temporarily disable a breakpoint without actually clearing it. The Clear gadget will clear the breakpoint and disable it. Entering an address of zero will also clear the breakpoint. Breakpoints are placed just before continuing execution, after all memory changes are downloaded, and are cleared each time normal

communications are re-established.

The Register fields are used to display and modify the register contents of the Handy CPU. These fields are updated whenever normal communications are re-established, and are downloaded just prior to continuing execution. If the Go operation was specified the PC is also loaded with contents of the Select field. All of the registers may be modified, and may contain any valid hex data. In addition, there are a set of singe-character gadgets at the bottom of the display that reflect the current state of the status (P) register. The P register may be modified by changing the states of these gadgets.

The Data field is a multi-purpose area of the display. It can be used to display one of four different sets of data. The cursor keys can be used to scroll the display up or down, and to move around the various fields in the display. The types and number of fields varies with the type of display selected.

When the Code option is specified the display will contain a disassembled listing of memory. The address fields may be edited to select particular addresses to disassemble. In addition, the hex opcode and operand fields may also be edited. Eventually, the instruction mnemonic will also be editable. Any changes are immediately downloaded to Handy. If an area of memory is selected that has not previously been fetched from Handy then that memory will be read from Handy and the display will be updated.

If the Data option is selected the display area will contain a hex display of Handy memory. The data is displayed as both hex and ASCII, and either field may be modified. Modified fields are immediately downloaded to Handy. When in this mode, the display is continually updated as long as normal communications are established. This allows monitoring certain hardware registers in real-time. To facilitate editing, the field currently selected for editing (indicated by the presence of the cursor) will not be updated until the cursor is moved from the field. In essence it is "frozen" to allow editing.

The Structure Display/Edit creates another window above the normal Handebug display that is used for intelligent displays of structure contents. Structures can be defined before loading Handebug, and the display will format the data and display it with appropriate titles.

Booting Handy

Before Handebug will communicate in any meaningful way with Handy, Handy (read "Apple][") must be bootstrapped with a communications "Monitor Program". This program communicates with Handebug using an established communications protocol. There are several versions of this code, each designed to work with a specific hardware configuration. Before attempting to bootstrap you should make sure that you have set up the correct version.

From the Handebug screen, select the "Bootstrap" gadget with the mouse or press function key F9 on the keyboard. This will cause the Bootstrap window to appear. There may be a delay of up to 3 seconds if Handebug thinks Handy is trying to communicate, so be patient. At this point RESET the Handy hardware to make sure it is in a known state.

NOTE to Apple][users: Power cycle the Apple][, or press Ctrl-Reset and type

PR#1<ret>.

The Bootstrap window will contain the names and paths of the bootstrap code and monitor program. If your Amiga has been set up in the recommended way then you will not need to change these fields. Otherwise, you may edit these strings to reflect your specific situation.

At this point, select the "OK!" gadget. Handebug will read the bootstrap code and update the display, then try to download it to Handy. If it is successful the procedure is repeated with the monitor program. If the display flashes, and does not update beyond the bootstrap code which is loaded at address $0200 then there is a communications problem and procedure should be repeated. If it still fails, there is a definite problem. Check to make sure you are using the correct bootstrap and monitor code.

If you really screwed up the file names, or if you just decide you don't really want to bootstrap (maybe you got here by accident), select the "Cancel" gadget and the operation will be canceled. The path and file names will also be restored to the state they were in when you originally selected Bootstrap.

If everything was successful and Handebug is now talking to Handy the LEDS will be flashing. This indicates that Handebug is polling Handy for current data for the current display area (note that this only occurs when DATA is selected).

NOTE to Apple][users: The LEDS are simulated on the Apple][display. You can observe them by attaching a composite monitor to the Apple composite video output.

If there were any symbols loaded the symbol table is cleared when you bootstrap.

Downloading Files

Downloading data to Handy is a breeze. First, select the "Download" gadget or press the corresponding functionkey . This will bring up the "Download File" requestor. From this requestor you may select the file to download by selecting names or gadgets or typing names and/or paths.

When the requestor appears it will may immediately begin loading file names from a a drive/volume and directory. This will probably not be the directory or volume you want. You do not have to wait for it to finish. You can specify another volume or directory at any time by selecting gadgets, names, or typing in the appropriate field.

You can get a file from any directory without ever taking your hands off the mouse. You can stroll through the disks by continually selecting the "Next Disk" gadget. Each time you do it will abort what it was doing and start loading file names again. As soon as you can find the name of the directory in the file name area you can select it. Then when the actual file name appears you can select it also.

If you prefer, you can directly specify any or all of the disk, directory, and filename by selecting the appropriate string fields and typing them yourself.

YOU DO NOT HAVE TO WAIT FOR FILE NAME LOADING TO STOP.

Once you have entered or selected the filename, press "OK" or double-click the filename to proceed. Handebug will open the file and transfer it to Handy. If a symbol file was created by using the +S option of the assembler Handebug will also read the symbol file and merge the symbols into the symbol table. If an execution address was specified with the .RUN directive the PC register will be changed to the new execution address.

If there is a problem because the file is the wrong format or Handy is not communicating, the display will stop updating and will flash after 3 seconds when the communication times out.

Unformatted files may be downloaded by selecting the "Raw Data" gadget in the "Download Options" window that opens along with the "Download File" requestor. You may also clear the current symbol table before downloading the file by selecting "Clear Symbols".

Selecting "Cancel" will abort the download process.

Downloaded data is also stored in Handebug's internal image of Handy's memory. It may then be changed by editing the data in the Data display window.

Entering and Editing Data

Handebug is conceptually oriented around fields and groups of fields. The cursor follows the mouse through fields; when the mouse is positioned over a field, the cursor appears in that field and the field may be edited immediately. When the mouse leaves a field, the cursor remains in the field until the mouse encounters a new field. Also, as soon as key data is entered the pointer disappears until the mouse is moved, getting it out of the way while data is being entered.

If communications are currently established with the monitor program in Handy the new data will be transmitted to Handy. In most cases, edited data that reflects the current Handy state is transmitted to Handy as soon as the cursor leaves the field. The exceptions are the register display and editing with the '+' and '-' keys.

Registers are only updated when the program is told to execute via selection of the "Go", "Continue", Step", or "Step Flat" gadgets, or by pressing function key F5. The register values are sent to Handy immediately before the command to continue execution.

When a data field is changed with the '+' or '-' key (increment or decrement value) the new data is immediately transmitted to Handy.

Running (and Stopping) a Program

The whole purpose of Handebug is to allow debugging of programs written for and running on Handy. You write a program, assemble it, download it, now you need to start it running.

There are several ways to run a program once it has been downloaded. All of them first restore the registers from the display, allowing you to change the registers as desired.

"Go" and "Continue" establish all the breakpoints, begin execution at the current PC location, and continue execution until some mysterious force, such as a breakpoint or NMI, causes the program to stop. "Go" has the additional feature of first setting the PC from the SELECT display field immediately to the right of the "Go" gadget if it is not empty.

The "Single Step" and "Step Flat" gadgets can be used to execute the next instruction and then return to Handebug. "Single Step" will always execute only one instruction and then return. "Step Flat" will also execute a single instruction. However, if the instruction is a JSR it will break on the return from the subroutine, rather than on the first instruction in the subroutine. This can be very useful if you are trying to follow a procedure that calls a lot of subroutines but you aren't interested in following execution through each and every subroutine. The normal breakpoints are established.

Once a program is running it has control of the machine and will continue to run until the hardware dies, the program destroys itself, or something Handebug did causes it to abort. There are several ways for Handebug to stop a program.

The trivial case is executing with "Single Step" or "Step Flat". The reasons should be obvious.

The most common and most useful method is by establishing breakpoints at strategic locations in the program. When a program's execution hits a breakpoint it causes control to return to Handebug, which updates its display to reflect the register contents and any changes to the page of memory currently being displayed. All breakpoints are also temporarily removed so that memory can be freely examined or changed.

Similar to breakpoints, yet totally different, is the Bus Monitor. The Bus Monitor is hardware that monitors the system bus for a variety of conditions that can be specified in Handebug. When the specified condition is met, control is forcibly returned to Handebug. This is incredibly useful when trying to track down random memory trashing, for instance.

Finally, the rudest (and most effective) method of interrupting a program is via the F10 function key, which causes an NMI (Non-Maskable Interrupt). The NMI will stop Handy in its tracks, no matter what it's doing (hopefully). Generally this is most useful if the program's execution has escaped and gone to the land of no return.

Breakpoints

Memwatch

Fill Memory

Function Keys

Hiding Handebug

Structure Display/Edit

Structure types 0 through F are reserved for standard system structures. You are free to define any of your own starting with number 10.

Structure definitions must be contained in a file called "handebug.defs" in the HANDY: directory.

A structure definition consists of a sequence of definition entries. An entry consists of a keyword, sometimes followed by parameter data. Not all entries are required.

A structure definition begins with the keyword STRUCTURE, and terminates when when either the next STRUCTURE keyword or the end of file is encountered. A structure consists of one or more fields. Each field definition begins with the keyword FIELD, and terminates when the next FIELD keyword is encountered or the structure definition is terminated. Each field can also contain one of more tities. Each title definition begins with the keyword TITLE, and terminates when the next TITLE keyword is encountered or when the field is terminated.

If a structure definition contains an error, then an error message is displayed in the CLI window that Handebug was started from. The message will indicate the line number containing the error and the name of the structure if possible. That structure definition will be discarded, and everything up to the beginning of the next structure definition will be ignored.

Blank lines are ignored, and may be used freely in a definition. Also ignored are lines that begin with a semi-colon, and anything following the keywords and their occasional parameters. Commenting a definition is thus not only possible, but highly encouraged.

STRUCTURE	- designates the start of a structure definition
NUMBER xx	- specifies HEXADECIMAL structure number
NAME string	- specifies the name of the structure (optional)
POSITION x,y	- specifies screen position of window (optional)
SIZE w,h	- specifies size of window (optional)
OFFSET n	- a DECIMAL number that is an offset from the specified address to determine the starting address
FIELD	- designates the start of a field definition
POSITION x,y	- specifies character coordinates of the field

```
    ¦TYPE HEX           - specifies field to be displayed as HEX data
one ¦TYPE DEC           - specifies field to be displayed as DECIMAL data
TYPE¦TYPE BIN           - specifies field to be displayed as BINARY data
only¦TYPE TEXT          - specifies field to be used as a pointer to a string
    ¦TYPE POINTER       - field to be displayed as HEX data with bytes swapped
    ¦TYPE STRUCT x      - field is a POINTER to another structure of type x
one ¦SIZE 8             - specifies field is only 1 byte long
SIZE¦SIZE 16            - specifies field is 2 bytes long
only¦SIZE string length - for TYPE TEXT, the number of characters to display
    OFFSET n            - specifies a DECIMAL offset relative to the starting
                          address of the structure (optional)

    TITLE               - designates the start of a field title (optional)
      POSITION x,y      - specifies the character coordinates of the title
      TEXT string       - the actual title data
```

Not Yet Implemented

As Handebug is a continually evolving program and still under development, the following features are not yet implemented:

MemWatch
Symbol/Value selection
Bus Monitor
Trace Buffer
Upload
Fill Memory (partially implemented)
Timed Interval Updates
Adjustable Timeout Periods