HANDEBUG SPECIFICATIONS DOCUMENT
Created: 28 June 1988
Last Modified: 28 June 1988
Authors: -RJ Mical-
John Meissen

This is a very early draft of this document. The intention of this document is to provide a specifications guide for the creation of the Handebug program. Ultimately, this document will evolve into a Handebug User's Guide.

LARGE DISPLAY BOX
- hex dump
- trace dump
- symbolic disassembly
- symbol table display

The large display box is used for various things. Which it's currently being used for is selectable with switches beneath the box. Along the right edge of the box is an autoknob prop gadget and arrow gadgets to allow stepping through the display.

The Trace buffer will be uploaded when the Trace function is selected for the display. The Trace buffer is not edittable.

The user should be able to edit the data in the large display box simply by positioning the mouse over an edittable area of the display box and then typing. If, for instance, the user positions the mouse over the hex dump fields, typing should fill in successive fields until the mouse moves again (actually, to allow for a little accidental jiggling, typing should go to successive fields until the mouse moves more than 8 ticks vertically or horizontally from its "start typing" position). As another example, if the user positions the mouse over the ascii display area of the hex dump, then the characters typed should be entered as ascii values. Also, in the disassembly display, where the disassembled text appears as well as the code bytes, positioning over the code bytes and typing should cause the bytes and the disassembly display to change, while positioning over the disassembly text and typing should invoke the mini-assembler, which should cause the instructions to be assembled in place (this will be a tricky job, with a lot of user-happy details that need working out).

BREAKPOINTS
- List of currently defined breakpoints (symbolic)
- Add a breakpoint
- Delete a breakpoint
- Gadget to delete all breakpoints

The breakpoint list shows all currently defined breakpoints, with the symbolic name or absolute address as entered by the user including simple equations (such as _main+10).

To add a breakpoint, use the mouse to point to an open slot and start typing.

To delete a breakpoint, point at it with the mouse and hit the DEL key.

To delete all breakpoints, select the Delete All Breakpoints gadget.

MEMORY MONITOR (SYMBOLIC)

- List of currently defined memory locations (symbolic)
- Add an address
- Delete an address
- Gadget to delete all addresses from the list (maybe)

The memory monitor list shows all currently defined memory locations to be monitored, with the symbolic name or absolute address as entered by the user including simple equations (such as _main+10). Every time the system breaks to the debugger, the current values of the memory monitor locations are fetched and displayed.

To add a memory monitor, use the mouse to point to an open slot and start typing.

To delete a memory monitor, point at it with the mouse and hit the DEL key. (poof)

To delete all memory monitor locations, select the Delete All Memory Monitors gadget.

Editting the data field of a watched location changes the memory in the machine being debugged.

DOWNLOAD/UPLOAD

Selecting the Download or Upload gadgets causes the fileio requester to be presented for the user to name a file.

With Download the symbol data is replaced and the other fields of the debugger are updated to reflect the new data. Breakpoints and memory watches are retained. Register contents?

With Upload, a second requester appears which allows the user to specify:

- Address range to be uploaded
- Format for upload (hex dump, assembler-compatible listing, binary)

BUS MONITOR

- Set the data AND value (mask)
- Set the lower and upper bound of data
- Set whether we're looking for data that's in or out of bounds
- Set the lower and upper bound of address
- Set whether we're looking for an address that's in or out of bounds

When the conditions of the bus monitor are satisfied, the system starts to record trace memory. When the Trace Deferred Stop Count value counts down to zero, the system breaks to the debugger.

The fields of the bus monitor need to be edittable in the usual way.

SET TRACE DEFERRED STOP COUNT

The Trace Deferred Stop Count is used to describe how many memory accesses should be allowed to take place after the bus monitor triggers a trace.

SINGLE-STEP

- Gadget to execute next instruction
- Gadget to execute next instruction flat
  (don't descend into subroutines)

Once in the debugger, we want to be able to execute the next instruction without setting the breakpoint. We need two gadgets, one simply to execute the next instruction, and another to execute the next instruction "flat" which means that the program will not single-step through the code of a subroutine if the current instruction is a JSR.

XTEXT

We should use XText to get fast updates.

TEXT/NUMERIC ENTRY

The user should be able to move the pointer over any number that is edittable and then simply type to edit that number. This would involve tracking the mouse with a cursor whenever the mouse is over any field that can be editted.

SYMBOLICS

Wherever possible the symbolic names should be used for memory locations. This includes showing symbolic names in the program disassembly, allowing the user to enter symbolic names for breakpoints and memory monitor locations,

UNDO

As much as possible actions should be undo-able. This would require some fancy programming, but if we think about undo-ability from the very beginning then it won't be so bad.

CURSOR KEYS

Use of the cursor keys should allow the user to move the cursor left, right, up and down from the current position in the current text box.

Shift-cursor keys are used to move the cursor a large jump. Shift-left cursor and shift-right cursor should move the cursor to the leftmost and rightmost position of the current text box respectively. The shift-up and shift-down cursor strokes should behave in different ways depending on the current text box: in the large display box, shift-cursor up and down displays either the previous or the next page respectively; in text boxes that don't scroll, the shift-cursor up and down should move the cursor to the top-left and bottom-right corners of the box.

FILL

Selecting Fill will cause a requestor to appear that will have fields for fill value, modifier, and range (upper and lower addresses). The modifier is a signed value that will be added to the fill value.

CURRENT FIELD BOX (?)

When the mouse is positioned over a valid field, if the left button is pressed then the contents of the field are copied into this box. If the mouse is positioned over an empty field (i.e., breakpoint or memory watch) and the left mouse button is pressed then the contents of this box are transfered to the new field. The field is edittable (sp) and must be a valid symbol or address expression. Data may be entered manually. If

the return key is pressed while the mouse is over this box the display will position to that address. The box may also be used to specify the starting address for the GO button.