Inter·Office Memorandum

To	Debugger Working Group	Date	October 27, 1978
From	Barbara Koalkin	Location	Palo Alto
Subject	Suggestions for stack window	Organization	SDD/SD

XEROX

Filed on: [IRIS]<KOALKIN>D5>DUISTACK.BRAVO

DRAFT

This memo contains thoughts on the stack window for the new debugger interface. This is based on my ideas and suggestions from the meetings, and is NOT a final design. This is only meant to stimulate discussion for the next meeting (I know we will come up with at least 6 other proposals).

* The debugger is created with (among other things) a window for manipulating the stack.
* Upon entering the debugger for the first time, this window is initialized with the context of whatever process is currently running. If you look at this window (i.e., make it "current", move the cursor into the window), it would contain all of the information associated with this context (configuration, module, global frame, local frame, procedure).

Each stack window:

* Has a (Tools style) menu containing the standard set of window operations.
* Is identified by its PROCESS which is displayed in the header of the window.
* Has a context that is current (selected, marked, call it what you like); this is the context in which expressions are interpreted.
* When you invoke the STACK command, giving it a PROCESS as a parameter (current selection perhaps), a stack window is created containing the information associated with this process (should it also become the current context for the debugger's symbol lookup?).
* When you are "in" this window, commands are invoked by single letters. The functions provided include: next procedure on the call stack, display the local variables, make this the current context, show the source line, load the source file into a window, jump *n* levels.
* We can also have single letter type-in commands in this window for directing program control, i.e., P for Proceed, Q for Quit, and K for Kill session.
* A subwindow of the stack window is reserved for showing context information about the current configuration, psb, module, global frame, and local frame. The rest of this subwindow is used to show the procedure call stack.

Input functions:

* stack commands such as next procedure on the stack, display the variables, make this the context, display the source line, jump *n* (or -*n*) levels on the stack
* input to the interpreter

Output functions:

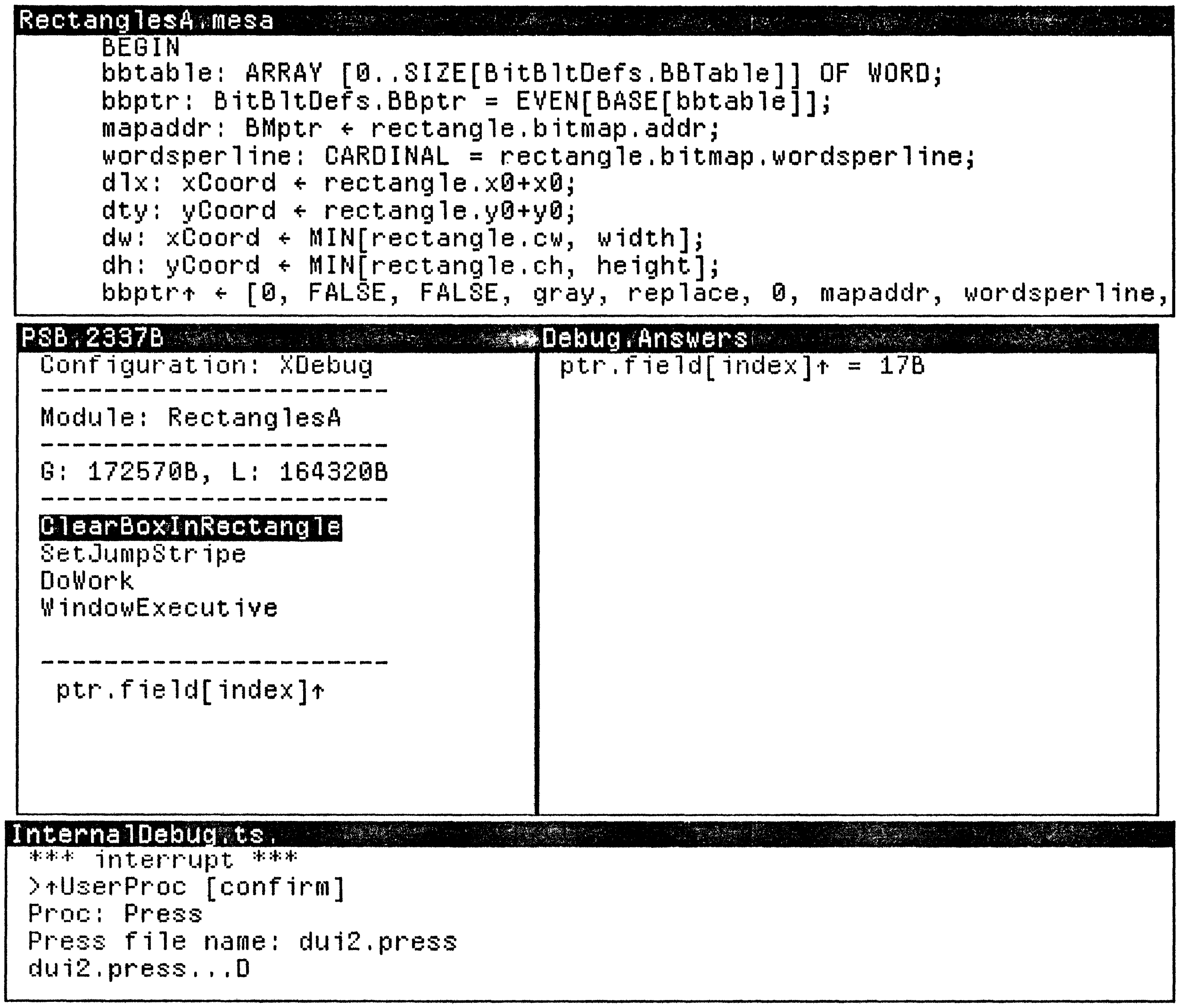
* display of local variables
* display the source line
* output from the interpreter
* messages like "No symbols for nnnnnnnB"

Problems:

* There needs to be a notion of THE current context, in much the same way as there is now. How to display this: a current context stack window, as distinguished from stack "viewers" that allow you examine the stack of a particular process without changing the overall context? Having *n* stack windows complicates the proposed scheme for changing contexts to something other than what is visible (i.e., to a different configuration). In addition, when you do change to a new configuration, the process context is set to NIL so there is no identifier for the stack window.
* How much feedback to user type-in? What gets entered into a log?
* We need to be able to change THE current context both within an exisiting stack window as well as switching between stack windows. Having multiple stack windows makes it easy to rapidly switch contexts between different processes. All we need is an indication of which stack window is THE current one (how about a graphic indication like an asterisk or using THE current selection, whatever that is).

I have included some sample interfaces that we can use as a basis for discussion (please disregard the InternalDebug.ts window, the position of the cursor, and some of the file names; these are all for use in creating the press files):

* Figures 1 and 2 demonstrate the proposal for having a small input feedback window at the bottom of the stack window. Figure 1 shows it echoing the Variable command and displaying the variables in the window to the right (it should be called Debug.Answers). Figure 2 shows interpreter input (invoked by SP) in the feedback window and output in the Debug.Answers window. Both of these commands were evaluated in terms of the current context (highlighted) in the stack window.
* Figures 3 and 4 demonstrate the proposal for having a Debug.Interpreter window. Figure 3 has the window to the right of the stack window (or part of it) and Figure 4 has it on the bottom (the L-shape we discussed). Both of these commands were evaluated in terms of the current context (highlighted) in the stack window.

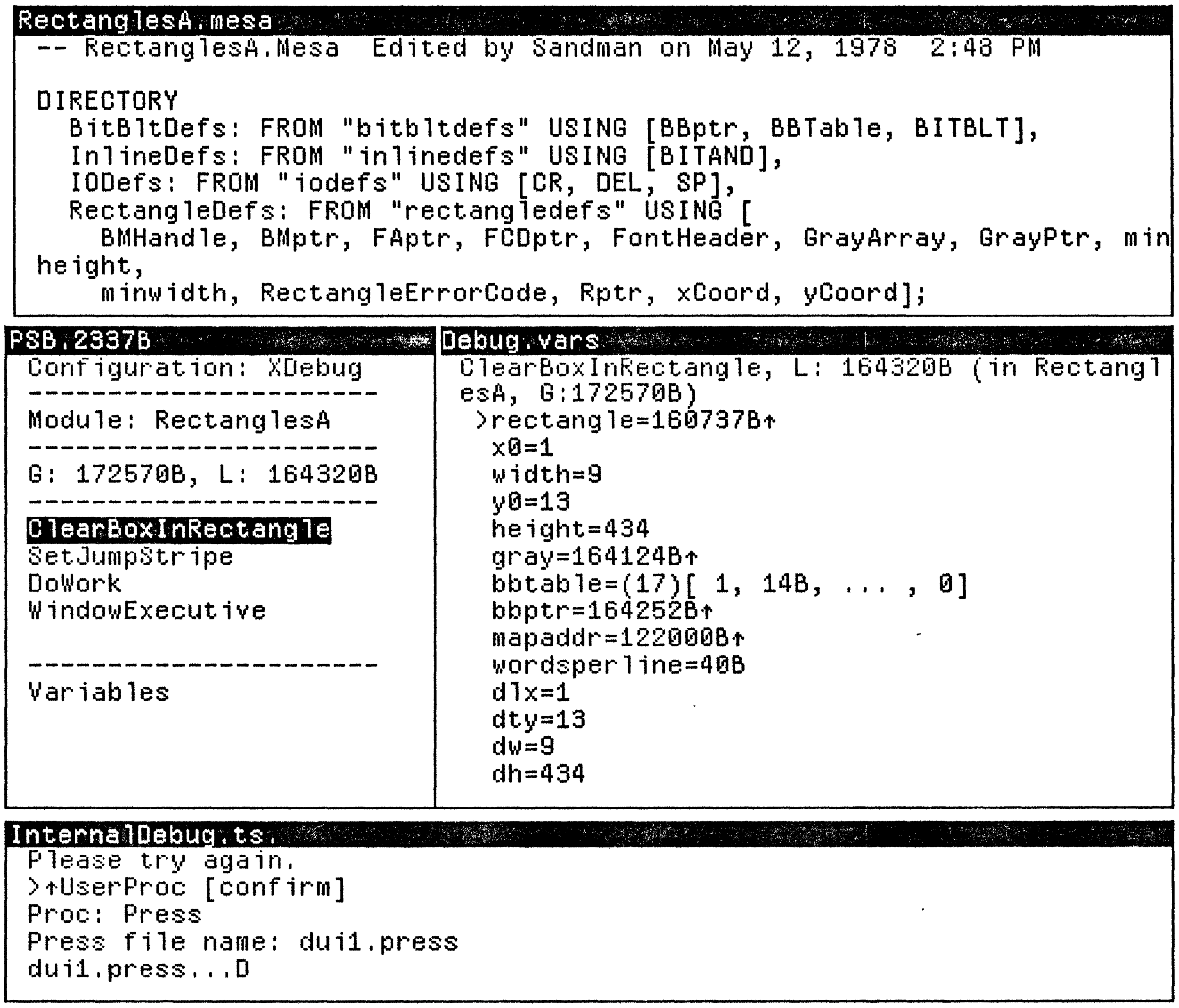


```
RectanglesA.mesa
     BEGIN
     bbtable: ARRAY [0..SIZE[BitBltDefs.BBTable]] OF WORD;
     bbptr: BitBltDefs.BBptr = EVEN[BASE[bbtable]];
     mapaddr: BMptr ← rectangle.bitmap.addr;
     wordsperline: CARDINAL = rectangle.bitmap.wordsperline;
     dlx: xCoord ← rectangle.x0+x0;
     dty: yCoord ← rectangle.y0+y0;
     dw: xCoord ← MIN[rectangle.cw, width];
     dh: yCoord ← MIN[rectangle.ch, height];
     bbptr↑ ← [0, FALSE, FALSE, gray, replace, 0, mapaddr, wordsperline,
```

```
PSB.2337B
 Configuration: XDebug
 ----------------------
 Module: RectanglesA
 ----------------------
 G: 172570B, L: 164320B
 ----------------------
 ClearBoxInRectangle
 SetJumpStripe
 DoWork
 WindowExecutive

 ----------------------
  ptr.field[index]↑
```

```
Debug.Answers
 ptr.field[index]↑ = 17B
```

```
InternalDebug.ts.
 *** interrupt ***
 >↑UserProc [confirm]
 Proc: Press
 Press file name: dui2.press
 dui2.press...D
```

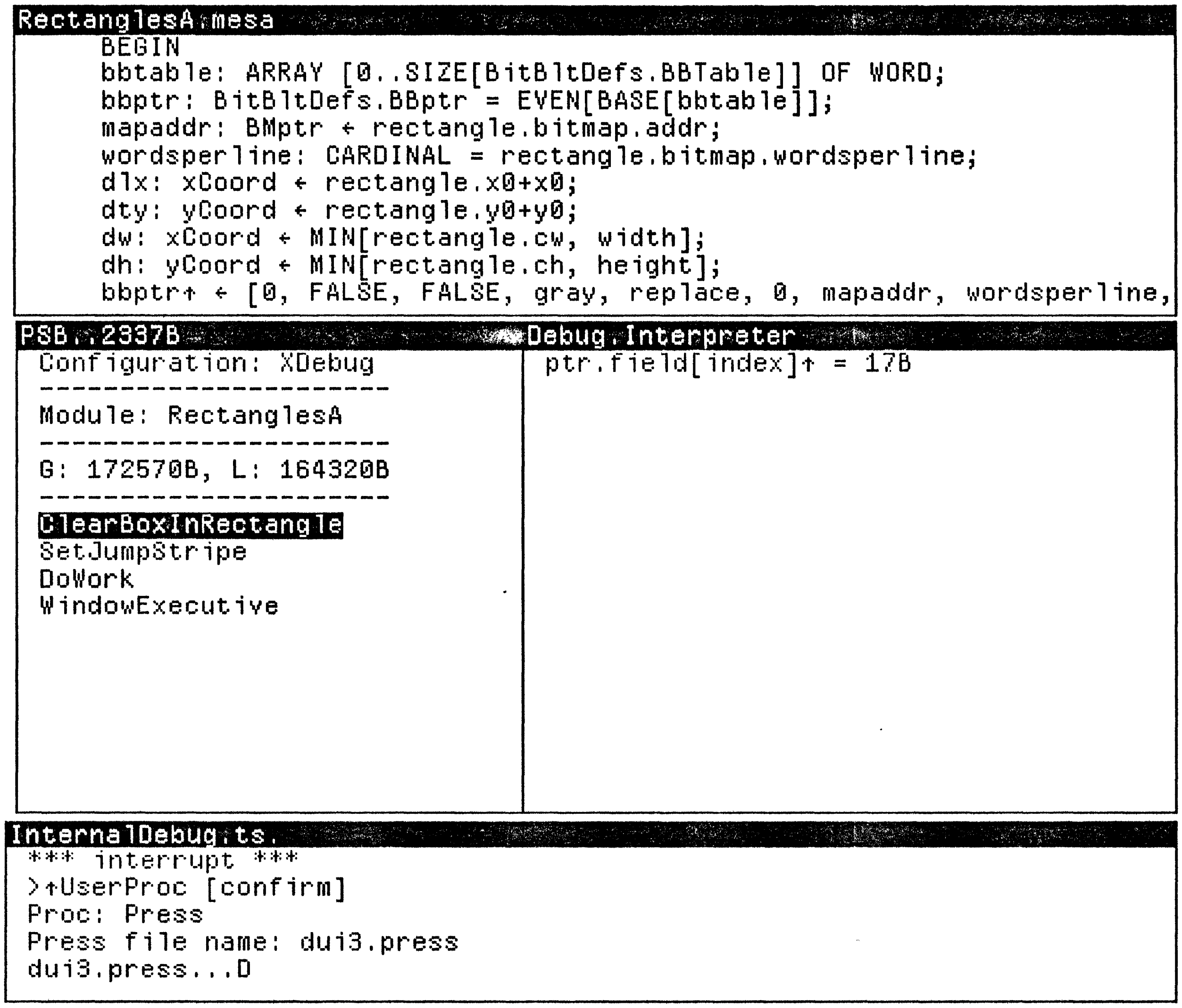


```
RectanglesA.mesa
 -- RectanglesA.Mesa  Edited by Sandman on May 12, 1978  2:48 PM

 DIRECTORY
   BitBltDefs: FROM "bitbltdefs" USING [BBptr, BBTable, BITBLT],
   InlineDefs: FROM "inlinedefs" USING [BITAND],
   IODefs: FROM "iodefs" USING [CR, DEL, SP],
   RectangleDefs: FROM "rectangledefs" USING [
     BMHandle, BMptr, FAptr, FCDptr, FontHeader, GrayArray, GrayPtr, min
 height,
     minwidth, RectangleErrorCode, Rptr, xCoord, yCoord];
```

```
PSB.2337B
 Configuration: XDebug
 ----------------------
 Module: RectanglesA
 ----------------------
 G: 172570B, L: 164320B
 ----------------------
 ClearBoxInRectangle
 SetJumpStripe
 DoWork
 WindowExecutive

 ----------------------
 Variables
```

```
Debug.vars
 ClearBoxInRectangle, L: 164320B (in Rectangl
 esA, G:172570B)
  >rectangle=160737B↑
   x0=1
   width=9
   y0=13
   height=434
   gray=164124B↑
   bbtable=(17)[ 1, 14B, ... , 0]
   bbptr=164252B↑
   mapaddr=122000B↑
   wordsperline=40B
   dlx=1
   dty=13
   dw=9
   dh=434
```

```
InternalDebug.ts.
 Please try again.
 >↑UserProc [confirm]
 Proc: Press
 Press file name: dui1.press
 dui1.press...D
```

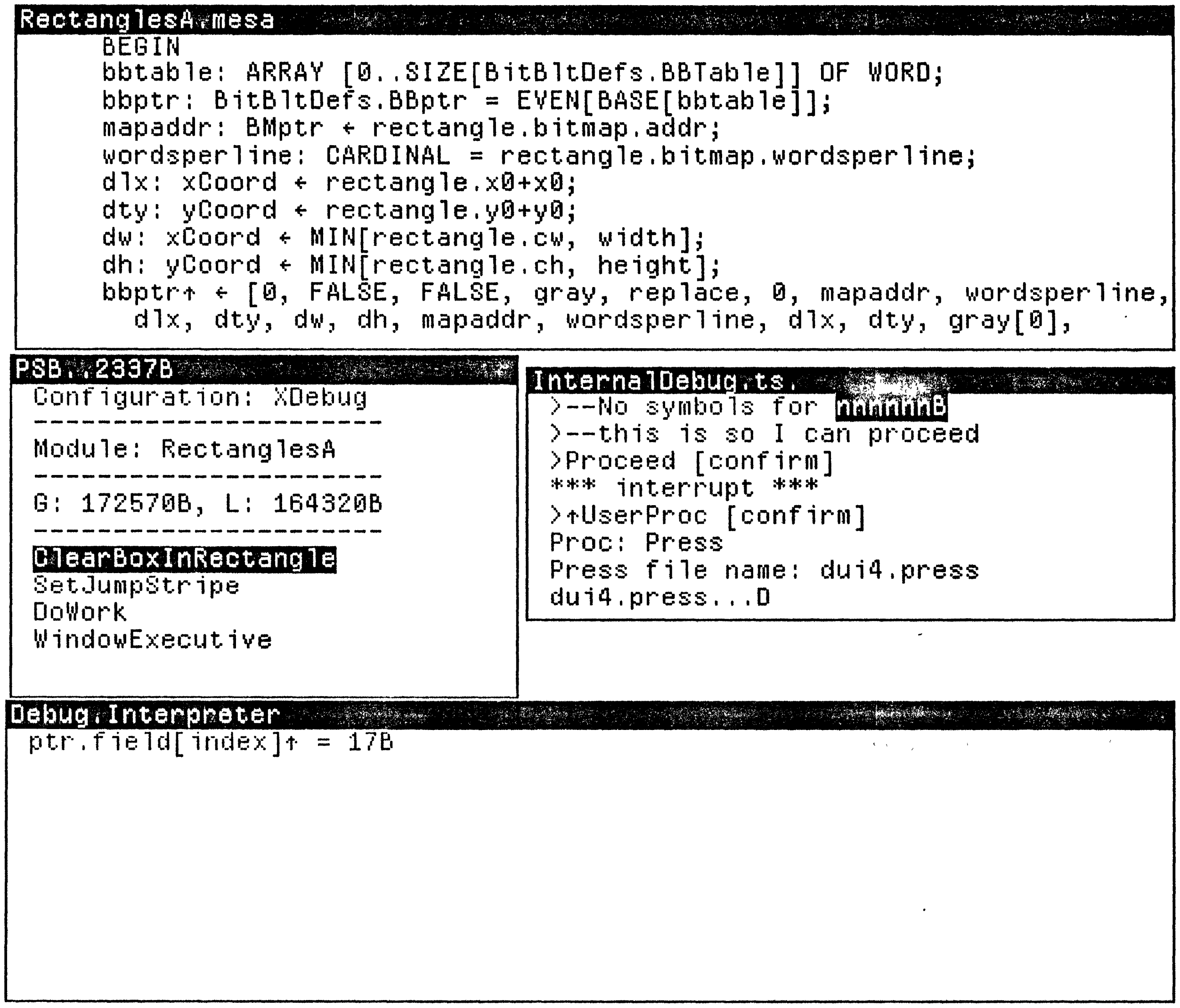


```
RectanglesA.mesa
     BEGIN
     bbtable: ARRAY [0..SIZE[BitBltDefs.BBTable]] OF WORD;
     bbptr: BitBltDefs.BBptr = EVEN[BASE[bbtable]];
     mapaddr: BMptr ← rectangle.bitmap.addr;
     wordsperline: CARDINAL = rectangle.bitmap.wordsperline;
     dlx: xCoord ← rectangle.x0+x0;
     dty: yCoord ← rectangle.y0+y0;
     dw: xCoord ← MIN[rectangle.cw, width];
     dh: yCoord ← MIN[rectangle.ch, height];
     bbptr↑ ← [0, FALSE, FALSE, gray, replace, 0, mapaddr, wordsperline,
```

```
PSB.:2337B
 Configuration: XDebug
 ----------------------
 Module: RectanglesA
 ----------------------
 G: 172570B, L: 164320B
 ----------------------
 ClearBoxInRectangle
 SetJumpStripe
 DoWork
 WindowExecutive
```

```
Debug.Interpreter
 ptr.field[index]↑ = 17B
```

```
InternalDebug.ts.
 *** interrupt ***
 >↑UserProc [confirm]
 Proc: Press
 Press file name: dui3.press
 dui3.press...D
```



```
RectanglesA.mesa
     BEGIN
     bbtable: ARRAY [0..SIZE[BitBltDefs.BBTable]] OF WORD;
     bbptr: BitBltDefs.BBptr = EVEN[BASE[bbtable]];
     mapaddr: BMptr ← rectangle.bitmap.addr;
     wordsperline: CARDINAL = rectangle.bitmap.wordsperline;
     dlx: xCoord ← rectangle.x0+x0;
     dty: yCoord ← rectangle.y0+y0;
     dw: xCoord ← MIN[rectangle.cw, width];
     dh: yCoord ← MIN[rectangle.ch, height];
     bbptr↑ ← [0, FALSE, FALSE, gray, replace, 0, mapaddr, wordsperline,
       dlx, dty, dw, dh, mapaddr, wordsperline, dlx, dty, gray[0],
```

```
PSB.42337B
 Configuration: XDebug
 ---------------------
 Module: RectanglesA
 ---------------------
 G: 172570B, L: 164320B
 ---------------------
 ClearBoxInRectangle
 SetJumpStripe
 DoWork
 WindowExecutive
```

```
InternalDebug.ts.
 >--No symbols for nnnnnnB
 >--this is so I can proceed
 >Proceed [confirm]
 *** interrupt ***
 >↑UserProc [confirm]
 Proc: Press
 Press file name: dui4.press
 dui4.press...D
```

```
Debug.Interpreter
 ptr.field[index]↑ = 17B
```