```
; FILE      Macsbug-syms.text           ; symbol commands
;
;         Modification history
;
;          3-Sep-84 Added WhereAmI code (should integrate WH code here)
;          5-Sep-84 Modified WhereAmI to search for UNLK A6 - RTS/JMP (A0)
;          6-Sep-84 WhereAmI prints offset from start of proc
;          8-Sep-84 Changed to LookupPC, severe re-write
;         21-Sep-84 Multiple LookupPC fixes (make sure even PC, IO buffer ptr manip
stuff)
;         26-Sep-84 Added LookupTrap (from D.text)
;          1-Nov-84 Sets to -1 TrapNum in LookupName loop
;

;-----------------------------------------------------------------------------
;  Routine Name     LookupPC
;
;  Arguments        A0 (input)          : the value of the program counter to lookup
;                   A1 (input)          : where to write out the characters to.
;
;  Uses             D0-D2/A2
;
;  Function         Tries to give indication of where PC is.  If in ROM,
;                   prints out trap value, if in user space, attempts to
;                   do Lisa Pascal symbols/Duvall symbols (eventually)
;                   If no symbols found, sets EQ CC flag, else NE
;-----------------------------------------------------------------------------

LookupPC
          BSR.S     LookupEntry         ; save some regs, A0 on stack
          TST.B     showSyms            ; are we displaying symbols?
          BEQ.S     LookupExit          ; No, exit.

          CMP.L     RomBase,A0          ; where is the location?
          BLT.S     BelowRom            ; A0 < Rom => somewhere in Ram

LookupExit
          CLR.L     (SP)+               ; pop old A0
          MOVEM.L   (SP)+,D3-D7/A3-A4   ; restore regs
          RTS

LookupEntry                             ; trashes D0/A2, saves std regs, A0 on stack
          MOVE.L    (SP)+,A2            ; save return address
          MOVEM.L   D3-D7/A3-A4,-(SP)   ; save some regs
          MOVE.L    A0,D0               ; now make sure address is even
          BCLR      #0,D0
          MOVE.L    D0,A0
          MOVE.L    A0,-(SP)            ; save A0 (PC) on stack
          SF        LastRoutine         ; reset last procedure flag
          SF        SymFound            ; reset name found flag
          JMP       (A2)                ; and return

;Look for RST or JMP (A0) that ends Lisa Pascal routines & preceeds routine
;name.  First byte of name is offset by $80.  ??? should check the Lisa sym
;flag before attempting this search.
; New check to see if between procedures (LINK closer than UNLK)

BelowRom

; first look for LINK A6

          MOVEQ     #0,D0               ; clear counter
          MOVE.W    #200,D0             ; search for 200 words
          MOVE.W    #$4E56,D1           ; look for LINK A6
          MOVEQ     #2,D2               ; search forward by 2's
          BSR.S     Look4Word           ; and try to find it
          BNE.S     @0                  ; found it, A0 pts to LINK A6

; didn't find it, fake location as some huge number

          MOVE.L    MaskBC,A0           ; a large dummy value (FFFFFF)
@0        MOVE.L    A0,D3               ; save location of LINK A6 in D3

; now look for RTS & JMP (A0)

          MOVEQ     #0,D0               ; clear counter
          MOVE.W    #2048,D0            ; search 2K words for RTS
          MOVE.W    #$4E75,D1           ; set search value for RTS
          MOVEQ     #2,D2               ; set for positive word search
          MOVE.L    (SP),A0             ; set A0 = PC
          BSR.S     Look4Word           ; and try to find it
          BNE.S     @1                  ; found, A0 pts to RTS

; didn't find RTS, fake location

          MOVE.L    MaskBC,A0           ; fake large value
@1        MOVE.L    A0,D4               ; save location of RTS

; now look for JMP (A0)

          MOVEQ     #0,D0               ; clear counter
          MOVE.W    #2048,D0            ; search 2K words for JMP (A0)
          MOVE.W    #$4ED0,D1           ; set search value, D2 still valid
          MOVE.L    (SP),A0             ; set A0 = PC
          BSR.S     Look4Word           ; search for it
          BNE.S     @2                  ; found, A0 pts to JMP (A0)

; didn't find JMP (A0), fake location

          MOVE.L    MaskBC,A0           ; face large value

; now try to figure out which to use, RTS or JMP (A0)

@2
          CMP.L     A0,D4               ; which is 'closer'
          BGT.S     Try4UNLK            ; D4 > A0, RTS further away, use JMP loc (A0)

          MOVE.L    D4,A0               ; set last instruction loc to pt to RTS

; Maybe found either RTS or JMP (A0), first check whether we did find anything.  Next,
; see whether it's 'closer' than LINK A6 ... if not,
; assume we're above all procedures or in between two and bail out

Try4UNLK
          CMP.L     MaskBC,A0           ; if A0 = FFFFFF, then neither RTS or JMP found
          BEQ.S     LookupExit          ; didn't find either, bail out

          CMP.L     A0,D3               ; which is closer, RTS/JMP (A0) or LINK?
          BGE.S     @0                  ; D3 >= A0, RST/JMP(A0) closer then LINK, all
gokay

          CMP.L     (SP),D3             ; If loc of LINK = PC, then @ first instruction
of routine
          BNE.S     LookupExit          ; not first instruction, exit
```

```
@0
          MOVE.L    A0,-(SP)            ; save location of last instr. (RTS/JMP) on
stack
          MOVEQ     #10,D0              ; search 10 words back for UNLK A6
          MOVE.W    #$4E5E,D1           ; set search value to UNLK A6
          MOVEQ     #-2,D2              ; set for backwards word search
          BSR.S     Look4Word           ; and try to find it

          BEQ.S     @1                  ; not found, conditional bail out
          MOVE.L    (SP)+,A0            ; set A0 back to position of last instruction
          BRA.S     PrintProcSym        ; found, print out string

@1        MOVE.L    (SP)+,A0            ; get last instruction
          ADDQ      #2,A0               ; bump to next instruction
          CMP.W     #$4E5E,(A0)+        ; is next instruction UNLK A6?
          BNE       LookUpExit          ; no, bail out
          ST        LastRoutine         ; yes, we're at very end of program w/RTS, UNLK
A6, RTS
          BRA.S     PrintProcSym        ; print the sucker


; Look4Word, searches through memory looking for the appropriate word value.
; ??? Should integrate with Find command at some point.
; A0 = search loc, D0 = count, D1 = word value to search for
; D2 = search increment (word),  sets NE flag if found

BumpCounter
          ADD.L     D2,A0               ; adjust A0

Look4Word
          CMP.W     (A0),D1             ; is it what we want?
          BNE.S     @1                  ; no, keep going
          MOVEQ     #1,D0               ; set NE CC flag
          RTS

@1        SUBQ.L    #1,D0               ; dec loop counter
          BNE.S     BumpCounter         ; keep looking
          MOVEQ     #0,D0               ; set flag
          RTS                           ; and return

;We have what we hope is a valid string of eight bytes, pointed at by 2(A0),
;transfer to print buffer and try to print proc location.
;Note : Proc symbols have first character offset by $80

PrintProcSyms

          ADDQ.L    #2,A0               ; bump past RTS or JMP (A0) instruction
          MOVE.L    A1,-(SP)            ; save old IO ptr
          MOVEQ     #7,D2               ; print out eight chars

@0        MOVE.B    (A0)+,D1            ; get nth char
          BCLR      #7,D1               ; clear high bit of byte
          CMP.B     #' ',D1             ; must be at least a space
          BLT       LookupFlush         ; nope, bail out
          BEQ.S     @1                  ; no spaces allowed, skip

          CMP.B     #'^',D1             ; max char is ^
          BGT       LookupFlush         ; out of range there too
          MOVE.B    D1,(A1)+            ; stuff the nth char

@1        DBRA      D2,@0               ; and loop
```

```
          ADDQ      #4,SP                 ; get rid of saved IO ptr
          ST        SymFound              ; we found a name

;now try to print out offset from start of procedure

          MOVEQ     #0,D0                 ; clear count
          MOVE.W    #2048,D0              ; look for 2K words back from initial PC
location
          MOVE.W    #$4E56,D1             ; look for LINK A6,xxxx
          MOVEQ     #-2,D2                ; look in reverse direction by words
          MOVE.L    (SP),A0               ; set saved PC as start loc
          BSR.S     Look4Word             ; look for it
          BEQ       LookupExit            ; not found, exit

          MOVE.L    (SP),D0               ; get the PC
          SUB.L     A0,D0                 ; D0 = PC - start of proc = offset
          MOVE.B    #'+',(A1)+            ; stuff '+'
          BSR       Print4Hex             ; print out hex offset D0 at IO loc A1
          BRA       LookupExit            ; and return

LookupFlush
          MOVE.L    (SP)+,A1              ; restore IO ptr (flush bad name)
          BRA       LookupExit            ; and return

          .IF       Tnames
;---------------------------------------------------------------------------
; Routine Name      LookupTrap
;
; Arguments         A0 (input)            : where to put the found trap name
;                   D0 (input)            : trap number (0..511)
;
; Uses              D0/D1
;
; Function          Finds the trap name in the trap name table corresponding
;                   to <trap number>.
;---------------------------------------------------------------------------

LookupTrap
          MULU      #10,D0                ; D0 = offset into trap name table

          LEA       TrapNames,A1          ; get address of trap name table
          ADD.L     D0,A1                 ; A1 = address of first char of name

          MOVEQ     #9,D1                 ; print out 10 chars

@0        MOVE.B    (A1)+,(A0)+           ; append next char
          DBRA      D1,@0

          RTS                             ; and return


;---------------------------------------------------------------------------
; Routine Name      LookupName
;
; Registers         A0 (input)            : Ptr to string to find
;                   D0.B (input)          : Length of input string
;                   D1 (output)           : Value of string
;
; Uses              D2
;
; Function          First looks through the trap name table for input string.
;                     If found, returns trap location, otherwise searches through
;                     the heap for a routine name match.  If that's found, returns the
;                     location of the first instruction of the routine, otherwise
;                     returns 0.
;-------------------------------------------------------------------------------

LookupName
          MOVEM.L   D0/D3,-(SP)           ; save off working regs
          MOVEQ     #-1,D2
          MOVE.W    D2,TrapNum            ; no trap names yet (also cleared in LookupName)

          LEA       TrapNames,A1          ; A1 = ptr to current position in trap name
table
          MOVEQ     #0,D2                 ; clear temp counter
          MOVEQ     #0,D1                 ; D1 = trap # counter

          CMP.B     #10,D0                ; force max length to 10
          BLE.S     SetDiff

          MOVEQ     #10,D0

SetDiff
          MOVEQ     #10,D3
          SUB.B     D0,D3                 ; D3 = diff (length) between test string & trap
names

@0        MOVE.L    A0,A2                 ; A2 = ptr to current position in test string
          MOVE.B    D0,D2                 ; D2 = current count

@1        CMPM.B    (A1)+,(A2)+           ; match?
          BNE.S     @2                    ; no, reset things

          SUBQ.B    #1,D2                 ; dec count
          BEQ.S     FoundTname            ; we've searched the full length, it's a match

          BRA.S     @1                    ; keep checking names

; we didn't match over the full name, so advance the trap name ptr to the next name,
; reset the current count, and bump the trap number counter.

@2

          ADD.L     D2,A1                 ; bump A1 by unfinished count amount
          ADD.L     D3,A1                 ; bump A1 by diff between count & 10
          SUBQ.L    #1,A1                 ; pt it to first char of next name

          ADDQ      #1,D1                 ; bump trap number counter
          CMP.W     #$200,D1              ; check for out of range
          BNE.S     @0                    ; still ok, keep checking

; Here we should (if symbols are on) check for a routine name match.
; Work through heap, finding all locked resource blocks, then check each one for
; a routine name that matches.

; For now, exit w/EQ CC set for failure

          MOVEQ     #0,D2                 ; null return value

LUTNexit
          MOVEM.L   (SP)+,D0/D3           ; restore D0/D3
          TST.L     D2                    ; set CC codes
          RTS                             ; and return
```

```
; we found the name.  D1 = trap number

FoundTname
          MOVE.W    D1,TrapNum            ; save value for parsing check

          MOVE.L    $28,-(SP)             ; save our A-trap dispatcher
          MOVE.L    SAVEA,$28             ; restore original one

          MOVE.L    D1,D0                 ; set up for call
          _GetTrapAddress                 ; get the trap address
          MOVE.L    A0,D1                 ; set up return value

          MOVE.L    (SP)+,$28             ; restore our A-trap dispatcher

          MOVEQ     #1,D2
          BRA.S     LUTNexit              ; and exit

          .ENDC
```