

!A

\*\*\* End of Pass 1

\*\*\* End of Pass 2

```
0800          1          ttl "SWEET16 Source Code, SWEET16.L"
0800          2          ;
0800          3          ;
0800          4          ; SWEET16.L
0800          5          ;
0800          6          ;
0800          7          ; This is the version of SWEET16 that is included in the
0800          8          ; ROM2E.SW16GCR.8.ROM firmware.
0800          9          ;
0800         10          ; Build Version #8
0800         11          ;
0800         12          ; 2024 February 14
0800         13          ;
0800         14          ;
0800         15          ; DOS 4.5, Build 06
0800         16          ;
0800         17          ; 2024 February 14
0800         18          ;
0800         19          ;
0800         20          ; Start of Source Code: 0x4000
0800         21          ; Start of Symbol List: 0x7800
0800         22          ;
0800         23          ;
0800         24          ; Copyright (c) 2024 February 14 by
0800         25          ; Walland Philip Vrbancic Jr
0800         26          ;
0800         27          ; 6223 East Peabody Street
0800         28          ; Long Beach, California 90808
0800         29          ; Unitied States of America
0800         30          ;
0800         31          ; All Rights Reserved
0800         32          ;
0800         33          ; This software is the confidential and
0800         34          ; proprietary intellectual property of
0800         35          ; Walland Philip Vrbancic Jr
0800         36          ;
0800         37          ;
0000         38      R0L          epz $00
0001         39      R0H          epz $01
0018         40      R12L         epz $18
0019         41      R12H         epz $19
001C         42      R14L         epz $1C
001D         43      R14H         epz $1D
001E         44      R15L         epz $1E
001F         45      R15H         epz $1F
0800         46          ;
0800         47          enz
0800         48          ;
0000         49      ZERO          equ $00
00FF         50      NEGONE        equ $FF
0800         51          ;
FA78         52      SW16RTN       equ $FA78
0800         53          ;
FDED         54      COUT          equ $FDED
0800         55          ;
FF3F         56      RESTORE       equ $FF3F
FF4A         57      SAVE          equ $FF4A
0800         58          ;
0800         59          ;
C670         60          org $C670
```

```

C670      61      obj $900
C670      62      usr
C670      63      ;
C670      64      ;
C670      65      ;      dfs $C670-*,NEGONE
C670      66      ;
C670      67      ;
C670      68      ; This version of the Sweet 16 Metaprocessor has been
C670      69      ; revised from the original.
C670      70      ;
C670      71      ;
C670      72      ; Sweet 16 Registers
C670      73      ;
C670      74      ; Register      Description
C670      75      ; -----
C670      76      ;      R0      Sweet 16 Accumulator (ACC)
C670      77      ;      R1-R11  Sweet 16 user registers
C670      78      ;      R12      Sweet 16 subroutine return Stack Pointer
C670      79      ;      R13      Sweet 16 compare instruction results
C670      80      ;      R14      Sweet 16 Status Register (PR & carry flag)
C670      81      ;      R15      Sweet 16 Program Counter (PC)
C670      82      ;
C670      83      ;
C670      84      ; Sweet 16 Non-Register Opcodes
C670      85      ;
C670      86      ; Opcode      Mnemonic      Description
C670      87      ; -----
C670      88      ;      00      RTN      Return to 6502 mode
C670      89      ;      01      BR      adr      Branch always, PC+=adr+2
C670      90      ;      02      BNC      adr      Branch if no carry, PC+=adr+2
C670      91      ;      03      BC      adr      Branch if carry, PC+=adr+2
C670      92      ;      04      BP      adr      Branch if plus, PC+=adr+2
C670      93      ;      05      BM      adr      Branch if minus, PC+=adr+2
C670      94      ;      06      BZ      adr      Branch if zero, PC+=adr+2
C670      95      ;      07      BNZ      adr      Branch if not zero, PC+=adr+2
C670      96      ;      08      BM1      adr      Branch if minus 1, PC+=adr+2
C670      97      ;      09      BNM1      adr      Branch if not minus 1, PC+=adr+2
C670      98      ;      0A      SOUT      chr      Send "chr" to COUT
C670      99      ;      0B      RS      Return from subroutine, PC=(R12--)
C670     100      ;      0C      BS      adr      Branch to subroutine, (R12++)=PC
C670     101      ;      0D      RSNS      Return from subroutine, PC=R12
C670     102      ;      0E      BSNS      adr      Branch to subroutine, R12=PC
C670     103      ;      0F      SJMP      adr      Jump to address, PC=adr
C670     104      ;
C670     105      ;
C670     106      ; Sweet 16 Register Opcodes
C670     107      ;
C670     108      ; Opcode      Mnemonic      Description
C670     109      ; -----
C670     110      ;      1n      SET      Rn,C      Load Rn immediate (with constant)
C670     111      ;      2n      LD      Rn      Load LO ACC from Rn
C670     112      ;      3n      ST      Rn      Store LO ACC into Rn
C670     113      ;      4n      LD@      Rn      Load LO ACC indirectly using Rn, Rn+1
C670     114      ;      5n      ST@      Rn      Store LO ACC indirectly using Rn, Rn+1
C670     115      ;      6n      LDD@      Rn      Load ACC indirectly using Rn, Rn+2
C670     116      ;      7n      STD@      Rn      Store ACC indirectly using Rn, Rn+2
C670     117      ;      8n      POP@      Rn      Rn-1, load LO ACC indirectly using Rn
C670     118      ;      9n      STP@      Rn      Rn-1, store LO ACC indirectly using Rn
C670     119      ;      An      ADD      Rn      Add Rn to ACC with carry out
C670     120      ;      Bn      SUB      Rn      Subtract Rn from ACC, with carry out
C670     121      ;      Cn      POPD@      Rn      Rn-1, HO ACC, Rn-1, LO ACC indirectly

```

```

C670      122 ; Dn    CPR    Rn  ACC-Rn->R13, with carry out
C670      123 ; En    INR    Rn  Rn+1->Rn
C670      124 ; Fn    DCR    Rn  Rn-1->Rn
C670      125 ;
C670      126 ;
C670      127 ; The Status Register contains the previous register*2 in
C670      128 ; the LO byte and the carry flag in bit 0 of the HO byte.
C670      129 ;
C670      130 ; Branch opcodes depend on the results of the prior opcode.
C670      131 ; The register*2 of the prior opcode is saved in LO R14.
C670      132 ; It is the value currently in the prior register that is
C670      133 ; tested for the selected branch opcode.
C670      134 ;
C670      135 ; Before the BS/RS opcodes can be used, R12 must be
C670      136 ; initialized with the address of the stack containing the
C670      137 ; return from subroutine addresses.
C670      138 ;
C670      139 ;
C670      140 ; Preserve 6502 registers and init SW16 program counter.
C670      141 ;
C670 20 4A FF 142 SW16      jsr SAVE
C673      143 ;
C673 68      144          pla
C674 85 1E    145          sta R15L
C676      146 ;
C676 68      147          pla
C677 85 1F    148          sta R15H
C679      149 ;
C679      150 ;
C679      151 ; Interpret a single SW16 opcode, then fetch the next SW16
C679      152 ; opcode.
C679      153 ;
C679 20 7F C6 154 SW16B      jsr SW16C
C67C      155 ;
C67C 4C 79 C6 156          jmp SW16B
C67F      157 ;
C67F      158 ;
C67F      159 ; Increment the SW16 program counter.
C67F      160 ;
C67F E6 1E    161 SW16C      inc R15L
C681 D0 02    162          bne SW16D
C683      163 ;
C683 E6 1F    164          inc R15H
C685      165 ;
C685      166 ;
C685      167 ; Push the common HO routine address byte onto the stack
C685      168 ; and process the SW16 opcode.
C685      169 ;
C685 A9 C7    170 SW16D      lda /SETCMD
C687 48      171          pha
C688      172 ;
C688 A0 00    173          ldy #ZERO
C68A      174 ;
C68A      175 ;
C68A      176 ; Mask the specified SW16 register and double it for
C68A      177 ; non-register address table indexing or prior register.
C68A      178 ;
C68A B1 1E    179          lda (R15L),Y
C68C 29 0F    180          and #$0F
C68E      181 ;
C68E 0A      182          asl

```

```

C68F AA      183      tax
C690      184      ;
C690      185      ;
C690      186      ; Extract opcode.  If it is zero then process a
C690      187      ; non-register opcode routine.
C690      188      ;
C690 4A      189      lsr
C691      190      ;
C691 51 1E    191      eor (R15L),Y
C693 F0 0D    192      beq TOBR
C695      193      ;
C695      194      ;
C695      195      ; Save register*2 in LO status byte and clear HO status
C695      196      ; byte.  Form opcode*2 for register address table indexing.
C695      197      ;
C695 86 1C    198      stx R14L
C697 84 1D    199      sty R14H
C699      200      ;
C699 4A      201      lsr
C69A 4A      202      lsr
C69B 4A      203      lsr
C69C      204      ;
C69C A8      205      tay
C69D      206      ;
C69D      207      ;
C69D      208      ; Save the LO address onto the stack to process this SW16
C69D      209      ; register opcode.  Also C-flag = 0; N-flag = 0 for SETCMD.
C69D      210      ;
C69D B9 E1 C6 211      lda OPTBL-2,Y
C6A0 48      212      pha
C6A1      213      ;
C6A1 60      214      rts
C6A2      215      ;
C6A2      216      ;
C6A2      217      ; Increment the SW16 program counter for the expected
C6A2      218      ; branch address.  Save the LO address onto the stack to
C6A2      219      ; process this SW16 non-register opcode.
C6A2      220      ;
C6A2 E6 1E    221 TOBR      inc R15L
C6A4 D0 02    222      bne TOBR2
C6A6      223      ;
C6A6 E6 1F    224      inc R15H
C6A8      225      ;
C6A8 BD E2 C6 226 TOBR2     lda BRTBL,X
C6AB 48      227      pha
C6AC      228      ;
C6AC      229      ;
C6AC      230      ; Recall prior register*2 in LO status byte and set up
C6AC      231      ; carry flag from HO status byte for BC and BNC opcodes.
C6AC      232      ; Also N-flag = 0 and Y-reg = 0.
C6AC      233      ;
C6AC A6 1C    234      ldx R14L
C6AE      235      ;
C6AE A5 1D    236      lda R14H
C6B0 4A      237      lsr
C6B1      238      ;
C6B1 60      239      rts
C6B2      240      ;
C6B2      241      ;
C6B2      242      ; Y-reg set to 2 by SET opcode index*2.  First get HO byte,
C6B2      243      ; then LO byte for designated register.

```

```

C6B2          244 ;
C6B2 B1 1E    245 SETZ      lda (R15L),Y
C6B4 95 01    246          sta R0H,X
C6B6          247 ;
C6B6 88       248          dey
C6B7          249 ;
C6B7 B1 1E    250          lda (R15L),Y
C6B9 95 00    251          sta R0L,X
C6BB          252 ;
C6BB          253 ;
C6BB          254 ; Increment the SW16 program counter by 2.  C-flag = 0.
C6BB          255 ;
C6BB A5 1E    256          lda R15L
C6BD 69 02    257          adc #2
C6BF 85 1E    258          sta R15L
C6C1          259 ;
C6C1 90 02    260          bcc SET2
C6C3          261 ;
C6C3 E6 1F    262          inc R15H
C6C5          263 ;
C6C5 60       264 SET2      rts
C6C6          265 ;
C6C6          266 ;
C6C6          267 ; R12 must contain the stack address where to pop the
C6C6          268 ; return SW16 program counter.  Pop the H0 byte, then
C6C6          269 ; the LO byte.  Then R12 - 2 -> R12.
C6C6          270 ;
C6C6          271 ;
C6C6 A2 18    272 RSZ      ldx #R12L
C6C8          273 ;
C6C8 20 F7 C7 274          jsr DCRCMD
C6CB          275 ;
C6CB A1 00    276          lda (R0L,X)
C6CD 85 1F    277          sta R15H
C6CF          278 ;
C6CF 20 F7 C7 279          jsr DCRCMD
C6D2          280 ;
C6D2 A1 00    281          lda (R0L,X)
C6D4 85 1E    282          sta R15L
C6D6          283 ;
C6D6 60       284          rts
C6D7          285 ;
C6D7          286 ;
C6D7          287 ; Copy the current SW16 program counter in R15 to R12.
C6D7          288 ;
C6D7 A5 1E    289 BSNSZ    lda R15L
C6D9 85 18    290          sta R12L
C6DB          291 ;
C6DB A5 1F    292          lda R15H
C6DD 85 19    293          sta R12H
C6DF          294 ;
C6DF 4C 9C C7 295          jmp BRCMD
C6E2          296 ;
C6E2          297 ;
C6E2          298 ; The byte-pairs of this table have been swapped to remove
C6E2          299 ; the unused Fn+1 entry.
C6E2          300 ;
C6E2 00       301 BRTBL    byt RTNCMD-1      ; 0
C6E3 08       302 OPTBL    byt SETCMD-1      ; 1n
C6E4 9B       303          byt BRCMD-1      ; 1
C6E5 0E       304          byt LDCMD-1      ; 2n

```

```

C6E6 9C          305          byt BNCCMD-1          ; 2
C6E7 17          306          byt STCMD-1           ; 3n
C6E8 AD          307          byt BCCMD-1           ; 3
C6E9 20          308          byt LD@CMD-1          ; 4n
C6EA B0          309          byt BPCMD-1           ; 4
C6EB 2A          310          byt ST@CMD-1          ; 5n
C6EC B5          311          byt BMCMD-1           ; 5
C6ED 39          312          byt LDD@CMD-1         ; 6n
C6EE BA          313          byt BZCMD-1           ; 6
C6EF 42          314          byt STD@CMD-1         ; 7n
C6F0 C1          315          byt BNZCMD-1          ; 7
C6F1 52          316          byt POP@CMD-1         ; 8n
C6F2 C8          317          byt BM1CMD-1          ; 8
C6F3 62          318          byt STP@CMD-1         ; 9n
C6F4 D1          319          byt BNM1CMD-1         ; 9
C6F5 6B          320          byt ADDCMD-1           ; An
C6F6 DA          321          byt SOUTCMD-1          ; A
C6F7 79          322          byt SUBCMD-1          ; Bn
C6F8 0A          323          byt RSCMD-1           ; B
C6F9 4B          324          byt POPD@CMD-1         ; Cn
C6FA 8F          325          byt BSCMD-1           ; C
C6FB 7B          326          byt CPRCMD-1          ; Dn
C6FC DF          327          byt RSNSCMD-1         ; D
C6FD 32          328          byt INRCMD-1          ; En
C6FE 0C          329          byt BSNSCMD-1         ; E
C6FF F6          330          byt DCRCMD-1          ; Fn
C700 E8          331          byt SJMPCMD-1         ; F
C701             332          ;
C701             333          ;
C701             334          ; These routines must reside on the same page.
C701             335          ;
C701             336          ;
C701             337          ; RTN opcode returns to 6502 processing mode. Pop the
C701             338          ; SW16C return address, restore the 6502 registers, and
C701             339          ; return to 6502 mode using the SW16 program counter.
C701             340          ;
C701 68          341          RTNCMD    pla
C702 68          342          pla
C703             343          ;
C703 20 3F FF     344          jsr RESTORE
C706             345          ;
C706 4C 78 FA     346          jmp SW16RTN
C709             347          ;
C709             348          ;
C709             349          ; SET opcode gets a 2-byte immediate constant.
C709             350          ;
C709 10 A7        351          SETCMD    bpl SETZ          ; always taken
C70B             352          ;
C70B             353          ;
C70B             354          ; RS opcode is return from a BS subroutine call.
C70B             355          ;
C70B 10 B9        356          RSCMD     bpl RSZ          ; always taken
C70D             357          ;
C70D             358          ;
C70D             359          ; BSNS opcode is branch to subroutine using no address
C70D             360          ; stack.
C70D             361          ;
C70D 10 C8        362          BSNSCMD   bpl BSNSZ        ; always taken
C70F             363          ;
C70F             364          ;
C70F             365          ; LD opcode moves the contents of Rn to ACC.

```

```

C70F          366 ;
C70F B5 00    367 LDCMD      lda R0L,X
C711 85 00    368          sta R0L
C713          369 ;
C713 B5 01    370          lda R0H,X
C715 85 01    371          sta R0H
C717          372 ;
C717 60       373          rts
C718          374 ;
C718          375 ;
C718          376 ; ST opcode moves the contents of ACC to Rn.
C718          377 ;
C718 A5 00    378 STCMD      lda R0L
C71A 95 00    379          sta R0L,X
C71C          380 ;
C71C A5 01    381          lda R0H
C71E 95 01    382          sta R0H,X
C720          383 ;
C720 60       384          rts
C721          385 ;
C721          386 ;
C721          387 ; LD@ opcode loads the LO ACC indirectly from memory
C721          388 ; using the address in Rn. HO ACC is cleared. Make R0
C721          389 ; the prior register.
C721          390 ;
C721 A1 00    391 LD@CMD      lda (R0L,X)
C723 85 00    392          sta R0L
C725          393 ;
C725 A0 00    394          ldy #ZERO
C727 84 01    395          sty R0H
C729          396 ;
C729 F0 06    397          beq ST@3          ; always taken
C72B          398 ;
C72B          399 ;
C72B          400 ; ST@ opcode stores the contents of LO ACC indirectly to
C72B          401 ; memory using the address in Rn. Make R0 the prior
C72B          402 ; register. Fall into INRCMD.
C72B          403 ;
C72B A5 00    404 ST@CMD      lda R0L
C72D          405 ;
C72D 81 00    406 ST@2        sta (R0L,X)
C72F          407 ;
C72F A0 00    408          ldy #ZERO
C731          409 ;
C731 84 1C    410 ST@3        sty R14L
C733          411 ;
C733          412 ;
C733          413 ; INR opcode increments the register Rn.
C733          414 ;
C733 F6 00    415 INRCMD      inc R0L,X
C735 D0 02    416          bne INR2
C737          417 ;
C737 F6 01    418          inc R0H,X
C739          419 ;
C739 60       420 INR2        rts
C73A          421 ;
C73A          422 ;
C73A          423 ; LDD@ opcode loads the ACC indirectly from memory using
C73A          424 ; the address in Rn. The LO byte is loaded first. Make
C73A          425 ; R0 the prior register.
C73A          426 ;

```



```

C73A 20 21 C7 427 LDD@CMD jsr LD@CMD
C73D 428 ;
C73D A1 00 429 lda (R0L,X)
C73F 85 01 430 sta R0H
C741 431 ;
C741 90 F0 432 bcc INRCMD ; always taken
C743 433 ;
C743 434 ;
C743 435 ; STD@ opcode stores the contents of the ACC indirectly to
C743 436 ; memory using the address in Rn. LO ACC is stored first.
C743 437 ; Make R0 the prior register.
C743 438 ;
C743 20 2B C7 439 STD@CMD jsr ST@CMD
C746 440 ;
C746 A5 01 441 lda R0H
C748 81 00 442 sta (R0L,X)
C74A 443 ;
C74A 90 E7 444 bcc INRCMD ; always taken
C74C 445 ;
C74C 446 ;
C74C 447 ; POPD@ opcode decrements Rn and gets the H0 byte
C74C 448 ; indirectly from memory using the address in Rn.
C74C 449 ; Fall into POP@CMD.
C74C 450 ;
C74C 20 F7 C7 451 POPD@CMD jsr DCRCMD
C74F 452 ;
C74F A1 00 453 lda (R0L,X)
C751 A8 454 tay
C752 455 ;
C752 2C 00 00 456 bit *-*
C755 457 dfs !-2
C753 458 ;
C753 459 ;
C753 460 ; POP@ opcode sets the H0 byte to zero, then decrements Rn
C753 461 ; and gets the LO byte indirectly from memory using the
C753 462 ; address in Rn. Store the H0 and LO bytes into the ACC.
C753 463 ; Make R0 the prior register.
C753 464 ;
C753 A0 00 465 POP@CMD ldy #ZERO
C755 466 ;
C755 20 F7 C7 467 jsr DCRCMD
C758 468 ;
C758 A1 00 469 lda (R0L,X)
C75A 85 00 470 sta R0L
C75C 471 ;
C75C 84 01 472 sty R0H
C75E 473 ;
C75E A0 00 474 POP@3 ldy #ZERO
C760 84 1C 475 sty R14L
C762 476 ;
C762 60 477 rts
C763 478 ;
C763 479 ;
C763 480 ; STP@ opcode decrements Rn and stores the LO ACC
C763 481 ; indirectly to memory using the address in Rn. Make R0
C763 482 ; the prior register.
C763 483 ;
C763 20 F7 C7 484 STP@CMD jsr DCRCMD
C766 485 ;
C766 A5 00 486 lda R0L
C768 81 00 487 sta (R0L,X)

```

```

C76A          488 ;
C76A 90 F2    489          bcc POP@3          ; always taken
C76C          490 ;
C76C          491 ;
C76C          492 ; ADD opcode sets Y-reg to 0 so ACC + Rn -> ACC, carry
C76C          493 ; saved to HO status byte. Make R0 the prior register.
C76C          494 ;
C76C A5 00    495 ADDCMD      lda R0L
C76E 75 00    496          adc R0L,X
C770 85 00    497          sta R0L
C772          498 ;
C772 A5 01    499          lda R0H
C774 75 01    500          adc R0H,X
C776          501 ;
C776 A0 00    502          ldy #ZERO
C778 F0 0E    503          beq CPR2          ; always taken
C77A          504 ;
C77A          505 ;
C77A          506 ; SUB opcode sets Y-reg to 0 so ACC - Rn -> ACC, carry
C77A          507 ; saved to HO status byte. Make R0 the prior register.
C77A          508 ;
C77A A0 00    509 SUBCMD      ldy #ZERO
C77C          510 ;
C77C          511 ;
C77C          512 ; CPR opcode leaves Y-reg set to 13*2 (SET opcode index*2)
C77C          513 ; so ACC - Rn -> R13, carry saved to HO status byte. Make
C77C          514 ; R13 the prior register.
C77C          515 ;
C77C 38       516 CPRCMD      sec
C77D          517 ;
C77D A5 00    518          lda R0L
C77F F5 00    519          sbc R0L,X
C781 99 00 00 520          sta R0L,Y
C784          521 ;
C784 A5 01    522          lda R0H
C786 F5 01    523          sbc R0H,X
C788          524 ;
C788 99 01 00 525 CPR2      sta R0H,Y
C78B          526 ;
C78B 84 1C    527          sty R14L
C78D          528 ;
C78D          529 ;
C78D          530 ; Save carry out bit.
C78D          531 ;
C78D 26 1D    532          rol R14H
C78F          533 ;
C78F 60       534          rts
C790          535 ;
C790          536 ;
C790          537 ; BS opcode is branch to subroutine. Save the current
C790          538 ; SW16 program counter indirectly to memory using the
C790          539 ; address in R12. R15L is saved first, then R15H; finally
C790          540 ; R12 + 2 -> R12. Fall into BRCMD.
C790          541 ;
C790 A2 18    542 BSCMD      ldx #R12L
C792          543 ;
C792 A5 1E    544          lda R15L
C794 20 2D C7 545          jsr ST@2
C797          546 ;
C797 A5 1F    547          lda R15H
C799 20 2D C7 548          jsr ST@2

```

```

C79C          549 ;
C79C          550 ;
C79C          551 ; BR opcode is branch always.  Fall into BNCCMD.
C79C          552 ;
C79C 18        553 BRCMD      clc
C79D          554 ;
C79D          555 ;
C79D          556 ; BNC opcode is branch if carry is clear from prior opcode.
C79D          557 ;
C79D B0 0E     558 BNCCMD     bcs BRTS
C79F          559 ;
C79F          560 ;
C79F          561 ; Get displacement byte.  If it is negative set Y-reg to
C79F          562 ; minus one for 2's compliment addition.  Y-reg = 0.
C79F          563 ;
C79F B1 1E     564             lda (R15L),Y
C7A1 10 01     565             bpl BR2
C7A3          566 ;
C7A3 88        567             dey
C7A4          568 ;
C7A4          569 ;
C7A4          570 ; Add displacement to program counter.
C7A4          571 ;
C7A4 65 1E     572 BR2        adc R15L
C7A6 85 1E     573             sta R15L
C7A8          574 ;
C7A8 98        575             tya
C7A9          576 ;
C7A9 65 1F     577             adc R15H
C7AB 85 1F     578             sta R15H
C7AD          579 ;
C7AD 60        580 BRTS       rts
C7AE          581 ;
C7AE          582 ;
C7AE          583 ; BC opcode is branch if carry is set from prior opcode.
C7AE          584 ;
C7AE B0 EC     585 BCCMD      bcs BRCMD
C7B0          586 ;
C7B0 60        587             rts
C7B1          588 ;
C7B1          589 ;
C7B1          590 ; BP opcode is branch if prior register's value is
C7B1          591 ; postive.
C7B1          592 ;
C7B1 B5 01     593 BPCMD      lda R0H,X
C7B3 10 E7     594             bpl BRCMD
C7B5          595 ;
C7B5 60        596             rts
C7B6          597 ;
C7B6          598 ;
C7B6          599 ; BM opcode is branch if prior register's value is
C7B6          600 ; negative.
C7B6          601 ;
C7B6 B5 01     602 BMCMD      lda R0H,X
C7B8 30 E2     603             bmi BRCMD
C7BA          604 ;
C7BA 60        605             rts
C7BB          606 ;
C7BB          607 ;
C7BB          608 ; BZ opcode is branch if prior register's value is zero.
C7BB          609 ;

```

```

C7BB B5 00      610  BZCMD      lda R0L,X
C7BD 15 01      611              ora R0H,X
C7BF F0 DB      612              beq BRCMD
C7C1            613  ;
C7C1 60         614              rts
C7C2            615  ;
C7C2            616  ;
C7C2            617  ; BNZ opcode is branch if prior register's value is not
C7C2            618  ; zero.
C7C2            619  ;
C7C2 B5 00      620  BNZCMD     lda R0L,X
C7C4 15 01      621              ora R0H,X
C7C6 D0 D4      622              bne BRCMD
C7C8            623  ;
C7C8 60         624              rts
C7C9            625  ;
C7C9            626  ;
C7C9            627  ; BM1 opcode is branch if prior register's value is
C7C9            628  ; negative one.
C7C9            629  ;
C7C9 B5 00      630  BM1CMD     lda R0L,X
C7CB 35 01      631              and R0H,X
C7CD            632  ;
C7CD 49 FF      633              eor #NEGONE
C7CF F0 CB      634              beq BRCMD
C7D1            635  ;
C7D1 60         636              rts
C7D2            637  ;
C7D2            638  ;
C7D2            639  ; BNM1 opcode is branch if prior register's value is not
C7D2            640  ; negative one.
C7D2            641  ;
C7D2 B5 00      642  BNM1CMD    lda R0L,X
C7D4 35 01      643              and R0H,X
C7D6            644  ;
C7D6 49 FF      645              eor #NEGONE
C7D8 D0 C2      646              bne BRCMD
C7DA            647  ;
C7DA 60         648              rts
C7DB            649  ;
C7DB            650  ;
C7DB            651  ; SOUT opcode sends the "chr" value to COUT.
C7DB            652  ;
C7DB B1 1E      653  SOUTCMD    lda (R15L),Y
C7DD            654  ;
C7DD 4C ED FD    655              jmp COUT
C7E0            656  ;
C7E0            657  ;
C7E0            658  ; RSNS opcode is return from a BSNS subroutine call using
C7E0            659  ; no address stack. Copy the saved SW16 program counter
C7E0            660  ; in R12 to R15.
C7E0            661  ;
C7E0 A5 18      662  RSNSCMD    lda R12L
C7E2 85 1E      663              sta R15L
C7E4            664  ;
C7E4 A5 19      665              lda R12H
C7E6 85 1F      666              sta R15H
C7E8            667  ;
C7E8 60         668              rts
C7E9            669  ;
C7E9            670  ;

```

```
C7E9      671  ; SJMP opcode gets a 2-byte immediate address for the SW16
C7E9      672  ; program counter.  Get the LO byte, then the HO byte, and
C7E9      673  ; save the address to R15.  Make R15 the prior register.
C7E9      674  ; Fall into DCRCMD.
C7E9      675  ;
C7E9 B1 1E  676 SJMPCMD  lda (R15L),Y
C7EB AA     677          tax
C7EC       678  ;
C7EC C8     679          iny
C7ED       680  ;
C7ED B1 1E  681          lda (R15L),Y
C7EF       682  ;
C7EF 86 1E  683          stx R15L
C7F1 85 1F  684          sta R15H
C7F3       685  ;
C7F3 A2 1E  686          ldx #R15L
C7F5 86 1C  687          stx R14L
C7F7       688  ;
C7F7       689  ;
C7F7       690  ; DCR opcode decrements the register Rn.
C7F7       691  ;
C7F7 B5 00  692 DCRCMD  lda R0L,X
C7F9 D0 02  693          bne DCR2
C7FB       694  ;
C7FB D6 01  695          dec R0H,X
C7FD       696  ;
C7FD D6 00  697 DCR2     dec R0L,X
C7FF       698  ;
C7FF 60     699          rts
C800       700  ;
C800       701  ;
```

BSAVE SWEET16,A\$0900,B,L\$0190

```
C800      702          usr SWEET16
C800      703  ;
C800      704  ;
C800      705          stt "SWEET16 Symbol Table"
C800      706  ;
C800      707  ;
C800      708          end 111
```

\*\*\* End of Assembly

Symbol List starts at 0x7800, ends at 0x7A9E, used 0x029E, remaining 0x3D12

### Symbols unsorted:

R0L	0000	R0H	0001	R12L	0018	R12H	0019	R14L	001C
R14H	001D	R15L	001E	R15H	001F	ZERO	0000	NEGONE	00FF
SW16RTN	FA78	COUT	FDDED	RESTORE	FF3F	SAVE	FF4A	SW16	C670
SW16B	C679	SW16C	C67F	SW16D	C685	TOBR	C6A2	TOBR2	C6A8
SETZ	C6B2	SET2	C6C5	RSZ	C6C6	BSNSZ	C6D7	BRTBL	C6E2
OPTBL	C6E3	RTNCMD	C701	SETCMD	C709	RSCMD	C70B	BSNSCMD	C70D
LDCMD	C70F	STCMD	C718	LD@CMD	C721	ST@CMD	C72B	ST@2	C72D
ST@3	C731	INRCMD	C733	INR2	C739	LDD@CMD	C73A	STD@CMD	C743
POPD@CMD	C74C	POP@CMD	C753	POP@3	C75E	STP@CMD	C763	ADDCMD	C76C
SUBCMD	C77A	CPRCMD	C77C	CPR2	C788	BSCMD	C790	BRCMD	C79C
BNCCMD	C79D	BR2	C7A4	BRTS	C7AD	BCCMD	C7AE	BPCMD	C7B1
BMCMD	C7B6	BZCMD	C7BB	BNZCMD	C7C2	BM1CMD	C7C9	BNM1CMD	C7D2
SOUTCMD	C7DB	RSNSCMD	C7E0	SJMPCMD	C7E9	DCRCMD	C7F7	DCR2	C7FD

### Symbols alphabetically sorted:

ADDCMD	C76C	BCCMD	C7AE	BM1CMD	C7C9	BMCMD	C7B6	BNCCMD	C79D
BNM1CMD	C7D2	BNZCMD	C7C2	BPCMD	C7B1	BR2	C7A4	BRCMD	C79C
BRTBL	C6E2	BRTS	C7AD	BSCMD	C790	BSNSCMD	C70D	BSNSZ	C6D7
BZCMD	C7BB	COUT	FDDED	CPR2	C788	CPRCMD	C77C	DCR2	C7FD
DCRCMD	C7F7	INR2	C739	INRCMD	C733	LD@CMD	C721	LDCMD	C70F
LDD@CMD	C73A	NEGONE	00FF	OPTBL	C6E3	POP@3	C75E	POP@CMD	C753
POPD@CMD	C74C	R0H	0001	R0L	0000	R12H	0019	R12L	0018
R14H	001D	R14L	001C	R15H	001F	R15L	001E	RESTORE	FF3F
RSCMD	C70B	RSNSCMD	C7E0	RSZ	C6C6	RTNCMD	C701	SAVE	FF4A
SET2	C6C5	SETCMD	C709	SETZ	C6B2	SJMPCMD	C7E9	SOUTCMD	C7DB
ST@2	C72D	ST@3	C731	ST@CMD	C72B	STCMD	C718	STD@CMD	C743
STP@CMD	C763	SUBCMD	C77A	SW16	C670	SW16B	C679	SW16C	C67F
SW16D	C685	SW16RTN	FA78	TOBR	C6A2	TOBR2	C6A8	ZERO	0000

### Symbols numerically sorted:

ZERO	0000	R0L	0000	R0H	0001	R12L	0018	R12H	0019
R14L	001C	R14H	001D	R15L	001E	R15H	001F	NEGONE	00FF
SW16	C670	SW16B	C679	SW16C	C67F	SW16D	C685	TOBR	C6A2
TOBR2	C6A8	SETZ	C6B2	SET2	C6C5	RSZ	C6C6	BSNSZ	C6D7
BRTBL	C6E2	OPTBL	C6E3	RTNCMD	C701	SETCMD	C709	RSCMD	C70B
BSNSCMD	C70D	LDCMD	C70F	STCMD	C718	LD@CMD	C721	ST@CMD	C72B
ST@2	C72D	ST@3	C731	INRCMD	C733	INR2	C739	LDD@CMD	C73A
STD@CMD	C743	POPD@CMD	C74C	POP@CMD	C753	POP@3	C75E	STP@CMD	C763
ADDCMD	C76C	SUBCMD	C77A	CPRCMD	C77C	CPR2	C788	BSCMD	C790
BRCMD	C79C	BNCCMD	C79D	BR2	C7A4	BRTS	C7AD	BCCMD	C7AE
BPCMD	C7B1	BMCMD	C7B6	BZCMD	C7BB	BNZCMD	C7C2	BM1CMD	C7C9
BNM1CMD	C7D2	SOUTCMD	C7DB	RSNSCMD	C7E0	SJMPCMD	C7E9	DCRCMD	C7F7
DCR2	C7FD	SW16RTN	FA78	COUT	FDDED	RESTORE	FF3F	SAVE	FF4A